Stock Prediction using Neural Network

B. Manjula
Department of Informatics,
Kakatiya University,
Warangal, A.P, India
manjulabairam@gmail.com

S.S.V.N. Sarma
Department of CSE,
Vaagdevi College of Engineering,
Warangal, A.P, India.
ssvn.sarma@gmail.com

R. Lakshman Naik
Department of CSE,
BITS, Warangal, A.P, India
lakshman432@yahoo.com

G. Shruthi
Department of CSE,
JITS, Warangal, A.P, India
shruthiguda@gmail.com

Abstract---**Today neural networks have been integrated into most fields and are a very important analytical tool. Neural networks are trained without the restriction of a model to derive parameters and discover relationships, driven and shaped solely by the nature of the data. This has profound implications and applicability to the finance field. Multilayer neural network has been successfully applied to the time series forecasting. Steepest descend, a popular learning algorithm for back propagation network, converges slowly and has the difficulty in determining the network parameters.**

In fact, artificial neural networks have been widely used for forecasting financial markets. However, such applications to Indian stock markets are scarce. This paper applies neural network models to predict the daily returns of the BSE (Bombay Stock Exchange) Sensex. Multilayer perceptron network is used to build the daily return's model and the network is trained using Multiple linear regression (MLR) provides a better alternative for weight initialization. It is found that the predictive power of the network model is influenced by the previous day's return than the first three-day's inputs. The study shows that satisfactory results can be achieved when applying neural networks to predict the BSE Sensex. However, the proposed Multilayer perceptron network with MLR weight initialization requires a lower computation cost and learns better than steepest decent with random initialization.

***Keywords*: - Artificial Neural network; Stock prediction; Multilayer perceptron network; MLR weight initialization**

I. INTRODUCTION

Statistical methods and neural networks are commonly used for time series prediction. Empirical results have shown that Neural Networks outperform linear regression [1, 2, 3] since stock markets are complex, nonlinear, dynamic and chaotic [4]. Neural networks are reliable for modeling nonlinear, dynamic market signals [5]. Neural Network makes very few assumptions as opposed to normality assumptions commonly found in statistical methods. Neural network can perform prediction after learning the underlying relationship between the input variables and outputs. From a statistician's point of view, neural networks are analogous to nonparametric, nonlinear regression models.

Back propagation neural network is commonly used for price prediction. Classical back propagation adopts first order steepest descent technique as learning algorithm. Weights are modified in a direction that corresponds to the negative gradient of the error surface. Gradient is an extremely local pointer and does not point to global minimum. This hill-climbing search is in zigzag motion and may move towards a wrong direction, getting stuck in a local minimum. The direction may be spoiled by subsequent directions, leading to slow convergence.

In addition, classical back propagation is sensitive to the parameters such as learning rate and momentum rate. For examples, the value of learning rate is critical in the sense that too small value will make have slow convergence and too large value will make the search direction jump wildly and never converge. The optimal values of the parameters are difficult to find and often obtained empirically. Customary random weight initialization does not guarantee a good choice of initial weight values. The random weights may be far from a good solution or near local minima or saddle points of the error surface, leading to a slow learning. To overcome the deficiencies of steepest descent learning and random weight initialization, some researches [6, 7] have investigated the use of Genetic Algorithms and Simulated Annealing to escape local minimum. Some [8, 9] have attempted Orthogonal Least Squares. Some have adopted Newton-Raphson and Levenberg-Marquardt.

II. NEURAL NETWORK STRUCTURE

A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. Artificial neural networks are computers whose architecture is modeled after the brain. It resembles the brain in two respects: knowledge is acquired by the network through a learning process and inter-

neuron connection strengths known as synaptic weights are used to store the knowledge. They typically consist of many hundreds of simple processing units, which are wired, together in a complex communication network. Each unit or node is a simplified model of a real neuron, which fires (sends off a new signal) if it receives a sufficiently strong input signal from the other nodes to which it is connected. The strength of these connections may be varied to enable the network to perform different tasks corresponding to different patterns of node firing activity.

The basic element of a neural network is the perceptron as shown in Fig.1 below. Frank Rosenblatt first proposed it in 1958 at Cornell University. The perceptron has 5 basic elements: n-vector input, weights, summing function, threshold device and an output. Outputs are in the form of .1 and/or +1. The threshold has a setting, which governs the output based on the summation of input vectors. If the summation falls below the threshold setting, -1 is the output. If the summation exceeds the threshold setting, +1 is the output.

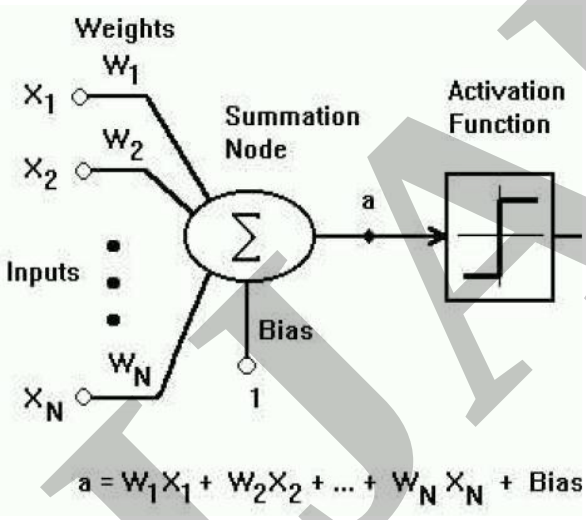

Figure 1: A Simple Perceptron

A more technical investigation of a single neuron perceptron shows that it can have an input vector X of N dimensions. These inputs go through a vector W of Weights of N dimension. Processed by the Summation Node, .a. is generated where .a. is the .dot product. of vectors X and W plus a bias. It is then processed through an activation function, which compares the value of .a. to a predefined threshold. If .a. is below the threshold, the perceptron will not fire. If it is above the threshold, the perceptron will fire one pulse whose amplitude is predefined. Neural networks itself represents a collection of artificial intelligence models which include, multilayer perceptron neural network, recurrent neural network, modular neural network, radial basis network. Each of these models has its own specific structure, training method and area of application. A thorough understanding on each of them is necessary to make the best choice of network structures for financial forecasting tasks.

The choice of relevant inputs for the networks would essentially be based on the expert knowledge in finance field. However, the best set of inputs is constrained by time factor and market factor thereby, demanding sufficient human expertise to choose the best inputs for a specific market at a specified time. Certain systematic methods give a helping hand to select a best sub-set of inputs from a large collection of possible inputs, including pruning of network, mutual information and embedding dimension. Equally important is the choice of outputs from the network to generate meaningful financial forecasting. This again rests on the expert financial knowledge.

Having chosen a particular network structure, network inputs, network outputs, objective function and historical data set of associated input-output pairs, an optimal set of network parameters with respect to that historical data set is then found by some numerical method. This process is termed .Training... It is possible that certain training methods might fail to arrive at an optimal set of parameters with respect to the data set. The possibility of finding optimal parameters varies from problem to problem. In general, if the objective function is highly complicated or if the noise in the data set is too high, it is more difficult to find the optimal parameters. However, there are certain methods to enhance the possibility of finding the optimal solution namely, genetic search, multiple training, hybrid algorithm of random search, back-propagation and multiple linear regressions and expanded range approximation. From a very broad perspective, neural networks can be used for financial decision making in one of the three ways:

- It can be provided with inputs, which enable it to find rules relating the current state of the system being predicted to future states.
- It can have a window of inputs describing a fixed set of recent past states and relate those to future states.
- It can be designed with an internal state to enable it to learn the relationship of an indefinitely large set of past inputs to future states, which can be accomplished via recurrent connections.

It follows that these three methods require decreasing sophistication in the choice of inputs coupled with stringent training methods. In most cases the optimal solution obtained from historical data set can be different from the optimal solution proposed for forecasting methods. For example, to know the returns of certain selected stocks in the past is quite a simple task. However, to forecast the same in the near future is a difficult task. It is interesting to note that the difference between the historical solution and the forecasting solution rests on the strength of the relation between the past and the current events. For financial forecasting tasks, the financial data is bound to contain lot of noise, which is not likely to repeat it in the future. Thus, the relation between the past and the future data needs to be strengthened to even out the differences between the historical solution and the forecasting solution. This would make way for the historical solution to be generalized for forecasting purposes.

III. DATA AND MODULE

The data consists of daily index values of the BSE Sensex. The data set consists of 3667 data points. The data has been obtained from Capitalize 2000 database that provides daily stock market data. The entire analysis has been done basically on the daily returns rather than the raw index value as such. The approach adopted to train the neural network using daily returns is explained in terms of the following:

A. Inputs and Outputs

The historical price is used as inputs to the network. The inputs to the neural network are basically the delayed coordinates of the time series. The number of inputs to the network is 4 and they are the 4 consecutive daily returns. The output is the prediction of the return on the fifth day.

B. Network Structure

The simple Structure of the neural network is as shown. Multi Layer Perceptron (MLP - Fig. 2) neural network is chosen for the purpose of prediction. It essentially consists of three layers of nodes namely, input, hidden and output layers. The first layer consists of the input data. The last layer is the output layer, which consists of the target values. All layers between the input and the output layers are called as the hidden layers.

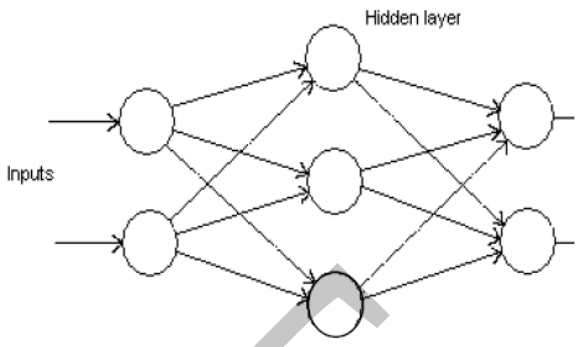

Fig. 2: A Simple MLP Network

C. Transfer Function

Transfer functions are those, which are used to relate the units of the different layers. They map the neurons input to an output, where the neuron is the link between the layers. The input functions are first multiplied by their respective weights, summed and then mapped to the output via the transfer function. The widely used transfer functions are the Sigmoid Function and the Hyperbolic Function. Both are very similar except that; the sigmoid function takes values between zero and one while the hyperbolic function takes values between minus one and plus one. A unit in the output layer determines its activity by following a two-step procedure.

1) First, it computes the total weighted input X_j, using the formula:

$$X_j = \sum_i y_i w_{ij} \quad \text{..... (1)}$$

Where y_i is the activity level of the j^{th} unit in the previous layer and W_{ij} is the weight of the connection between the i^{th} and the j^{th} unit.

2) Next, the unit calculates the activity y_j using some function of the total weighted input. Typically we use the sigmoid function:

$$y_j = 1/(1-e^{-xj}) \quad \text{..... (2)}$$

D. Training Scheme

Back-propagation is a hill-climbing technique. It runs the risk of being trapped in local optimum. The starting point of the connection weights becomes an important issue to reduce the possibility of being trapped in local optimum. Random weight initialization does not guarantee to generate a good starting point. It can be enhanced by multiple linear regressions. In this method, weights between input

layer and hidden layer are still initialized randomly but weights between hidden layer and output layer are obtained by multiple linear regression.

The weight W_{ij} between the input node i and the hidden node j is initialized by uniform randomization. Once input X_i^s of sample s has been fed into the input node and $W_{ij}{}^s$ have been assigned values, output value Rj^s of the hidden node j can be calculated as

$$R_j^s = f(\sum_i w_{ij}\ x_j^s) \quad \text{..... (3)}$$

Where f is a transfer function. The output value of the output node can be calculated as

$$y^s = f(\sum_j v_j R_j^s) \quad \text{..... (4)}$$

Where v_j is the weight between the hidden layer and the output layer.

Assume sigmoid function $f(x) = 1/(1+e^{-x})$ is used as the transfer function of the network. By Taylor's expansion,

$$f(x) = (1/2) + (x/4) \quad \text{.... (5)}$$

Applying the linear approximation in (5) to (4), we have the following approximated linear relationship between the output y and $v_j{}'s$:

$$y^s = (1/2) + ((1/4)(\sum_j^m v_j R_j^s)) \quad \text{..... (6)}$$

Or

$$4y^s - 2 = v_1 R_1^s + v_1 R_2^s + \ldots + v_m R_m^s$$

$s=1,2,\ldots N$ .... (7)

Where m is the number of hidden nodes; N is the total number of training samples.

The set of equations in (7) is a typical multiple linear regression model. $R_j{}'s$ are considered as the regressors. $V_j{}'s$ can be estimated by standard regression method. Once $v_j{}'s$ have been obtained, the network initialization is completed and the training starts.

IV. RESULTS

Information about the neural network used for predicting the daily returns is indicated in Table 1 and Table 2. Three layers were used with only one hidden layer. The input and the hidden layers have four nodes while the output layer has only one node. The network is fully connected. Sigmoid transfer functions have been used in the network. There were 3612 patterns, of which the last 1200 patterns were reserved for testing, which meant that the first 2412 patterns formed the training set.

Table 1: Network Information for Daily Returns

Network information	Input layer	Hidden layer	Output layer
Nodes	4	4	1
Transfer Function	Linear	Sigmoid	Sigmoid

Table 2: Training Information

Training Information	Daily Returns
Iterations	729.367
Training Error	0.002580
Test Set Error	0.002455
Learn Rates	0.001407
Training Patterns	2412
Test Patterns	1200

A. Prediction Accuracy

The network is trained by repeatedly presenting it with both the training and test data sets. To identify when to stop training, four parameters, namely the RMS error and the correlation coefficients of both the training and test sets were used. After 729.367 iterations, the RMS error and correlation coefficient of the training set was 0.002580 and 0.00823 respectively, while those of test set was 0.002455 and 0.01816. The training was stopped after 729.367 iterations as there was no significant decrease in either of the RMS errors and the correlation coefficient of the training set was also starting to form a plateau. The correlation coefficient of the test set was also oscillating within a small band. Thus, any further training was not going to be productive or cause any significant changes. The prediction accuracy of the training data is 96.3% and that of test data is 96.6% and is significant at .0.05 level of significance. The correlation between the input and predicted output has improved for the test data. As the values of RMS and correlation coefficient of the test data set are better than those of the training set, the model fits the test pattern better than the training pattern.

B. Performance

The performance of scenarios mentioned below is evaluated by average number of iterations required for training, average BSE in testing phase and the

percentage of correct direction prediction in testing phase. The results are summarized in Table 3.

Table 3: Performance evaluation for two scenarios

	Average no. of iterations	Average BSE	% of correct direction prediction
SD/RI	497.818	0.001797	72.564
SD/MLRI	729.367	0.002580	69.303

All scenarios, except for steepest descent with MLR initialization, achieve similar average BSE and percentage of correct direction prediction. All scenarios perform satisfactory. The mean square error produced is on average below 0.258% and more than 69% correct direction prediction is reached.

In Multilayer perceptron network, MLR initialization requires less number of iterations required for training than random initialization, achieving similar BSE and direction prediction accuracy with random initialization. The positive result shows that regression provides a better starting point for the local quadratic approximation of the nonlinear network function performed by Multilayer perceptron network.

However, in steepest descent network, regression initialization does not improve performance. It requires more number of iterations for training, produces a larger BSE and fewer correct direction predictions than random initialization. The phenomenon is opposite to the case in Multilayer perceptron network. It is attributed to the characteristics of the gradient descent algorithm that modifies direction to negative gradient of error surface, resulting in spoils of good starting point generated by MLP by subsequent directions.

V. CONCLUSION

In this paper, an attempt is made to explore the use of neural network to predict the daily returns of BSE Sensex. Multi layer perceptron network is used to build the daily returns model and the network is trained using multiple linear regression weight initializations. The results show that the predictive power of the immediate previous day's return is higher than the first three-day's inputs. The experimental results show that it is possible to model stock price based on historical trading data by using a three layer neural network. In general, both steepest descent network and Multilayer perceptron network produce the same level of error and reach the same level of direction prediction accuracy. In regard to initial starting point, the experimental results show the good starting point generated by multiple linear regression weight initializations. The initialization scheme may be improved by estimating weights between input nodes and hidden nodes, instead of random initialization. Enrichment of more relevant inputs such as fundamental data and data from derivative markets may improve the predictability of the network. Finally, more sophisticated network architectures can be attempted for price prediction.

REFERENCES

[1] Arun Bansal, Robert J. Kauffman, Rob R. Weitz. Comparing the Modeling Performance of Regression and Neural Networks as Data Quality Varies: A Business Value Approach, *Journal of Management Information Systems,* Vol 10. pp.11-32, 1993.

[2] Leorey Marquez, Tim Hill, Reginald Worthley, William Remus. Neural Network Models as an Alternate to Regression, *Proc. of IEEE 24th Annual Hawaii Int'l Conference on System Sciences*, pp.129-135, Vol VI, 1991.

[3] White H. Economic Prediction Using Neural Networks: The Case of IBM Daily Stock Returns, *Proc. of IEEE Int'l Conference on Neural Networks*, 1988.

[4] Robert R. Trippi, Jae L. Lee. Artificial Intelligence in Finance & Investing, Ch 10, IRWIN, 1996.

[5] Guido J. Deboeck. Trading on the Edge - Neural, Genetic, and Fuzzy Systems for Chaotic Financial Markets, Wiley, 1994.

[6] Michael McInerney, Atam P. Dhawan. Use of Genetic Algorithms with Back-Propagation in training of Feed-Forward Neural Networks, *Proc. of IEEE Int'l Joint Conference on Neural Networks*, Vol.1, pp. 203-208, 1993.

[7] Timothy Masters. Practical Neural Network Receipes in C++, Ch 4-10, Academic Press.

[8] Chen S., Cowan C.F.N., Grant P.M. Orthogonal Least Squares LearningAlgorithm for Radial Basis Function Networks, *IEEE Trans. Neural Network* 2(2), pp. 302-309, 1991.

[9] Mikko Lehtokangas. Initializing Weights of a Multilayer Perceptron Network by Using the Orthogonal Least Squares Algorithm, *Neural Computation*, 1995.

[10] Collins, E., Ghosh, S. and Svofield. C. (1998), .An Application of a multiple Neural Network Learning System to Emulation of Mortgage Underwriting Judgements., Proceedings of the IEEE International Conference on Neural Networks, Vol.2, pp.459-466.

[11] Dutta, S., and Shekhar, S. (1998), .Bond Rating: A Non-conservative Application of Neural Networks., Proceedings of theIEEE International Conference on Neural networks, Vol.2, pp.443-450.

