Journal of Software Engineering

ISSN 1819-4311

Academic Journals Inc.

www.academicjournals.com

Journal of Software Engineering 1 (1): 13-21, 2007
ISSN 1819-4311

Stock Rate Prediction Using Backpropagation Algorithm: Results with Different Number of Hidden Layers

[1]Asif Ullah Khan, [1]Mahesh Motwani, [1]Sanjeev Sharma,
[2]Abdul Saboor Khan and [3]Mukesh Pandey
[1]School of Information Technology, [2]Center of Excellence in Information Technology
[3]School of Energy and Environment Management,
Rajiv Gandhi Technological University, Bhopal, M.P., India

Abstract: Much research on the applications of Neural Networks for solving business problems have proven their advantages over statistical and other methods that do not include artificial intelligence. Neural Networks methods, have become very important in making stock rate predictions (Gately and Edward, 1996). Results show that the use of neural network based backpropagation algorithm for predicting stock prices with two hidden layers are more accurate in comparison to single layer and three, four and five hidden layers.

Key words: Backpropagation, hidden layers, activation function, stock market prediction, multilayer feedforward network, technical analysis

INTRODUCTION

Prediction of financial markets has long been an attraction in the minds of equity investors. Technical Analysis (Mizuno *et al*., 1998) provides a framework for studying investor behavior and generally focuses only on price and volume data. Typically, traders using this type of approach concern themselves chiefly with timing and are generally unaware of a company's financial health. Traders using this approach have short-term investment horizons and access to only price and exchange data. With the advent of powerful computers much attention has been focused on this field.

Equity market prices depend on many influences. Key factors that influence future equity prices can be broadly divided into quantitative and qualitative types (Mizuno *et al*., 1998). Primary quantitative factors include open rate, high rate, low rate, close rate and volume for individual equities. Qualitative factors include socio-economic, political, international, regional and performance factors to name but a few (Stefan Zemke, 2002).

The ability of neural networks to learn from training data has not been overlooked and as such neural networks have been applied to a range trading market applications of equity markets (Wong *et al*., 1997).

Equity market prices depend on many influences. Key factors that influence future equity prices can be broadly divided into quantitative and qualitative types. Primary quantitative factors include open rate, high rate, low rate, close rate and volume for individual equities (Zirilli, 1997). Qualitative factors include socio-economic, political, international, regional and performance factors to name but a few.

Due to the difficulty in accurately retrieving and quantifying historical qualitative factors, network inputs used in the model presented here have been confined to readily available quantitative data. However, from quantitative factors the key qualitative factor of the market sentiment can be derived (Kanas, 2001). Market sentiment tells us if the market is bullish, where high of confidence and rising

Corresponding Author: Sanjeev Sharma, School of Information Technology, Rajiv Gandhi Technological University, Bhopal, M.P., India Tel: 91755268844 Fax: 91755678844

prices prevail, or bearish, where there is a lack of investor confidence and prices are in decline. Thus historical data quantitatively reflects qualitative market sentiment to some extent, which in turn should give indication of future price movements.

Numerous research and applications of Neural Networks in solving business problems (Ward *et al*., 1995) has proven their advantage in relation to classical methods that do not include artificial intelligence. According to Wong *et al*. (1997) , the most frequent areas of Neural Networks applications are production/operations (53.5%) and finance (25.4%).

The primary aim of this study is the creation of a neural network that, given a set of historical daily data, is capable of predicting the direction of the future price trend. The price trend prediction simply being whether market prices would be on increase or decrease relative to a subset of the daily training data.

The purpose of this study is also to identify how many hidden layers of Neural networks are needed in order to have a good accuracy. Our results shows that the Neural networks using backpropagation algorithms having two hidden layers gives more accurate results in comparison to other layers. That's why we also trained networks with three, four and five hidden layers and results shows that the two hidden layers is the most accurate in comparison with single, three, four and five hidden layers.

NEURAL NETWORKS

A neural network is a non-linear type of model that receives its inspiration from the neural architecture of the human brain. A single neuron from the brain has three basic components - dendrites that provide for input from neighboring neurons, a soma for processing and axons for output to other neurons (Beale and Jackson, 1990). Each neuron may be connected to thousands of neighboring neurons via this network of dendrites and axons. This network allows the brain to achieve speed and power through massive parallel processing. Similarly, an artificial neural network contains inter-connections analogous to dendrites, processing nodes similar to somas in many regards and output connections that represent the axons. One of the prime advantages to neural network modeling is that there are no previous assumptions made about the relationship being modeled. Rather, the neural network examines data involving the phenomena in question and maps an approximation of the relationship (i.e., the underlying function) using a learning or training algorithm.

Use of Neural Network

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques (Ward *et al*., 1995). A trained neural network can be thought of as an expert in the category of information it has been given to analyze. This expert can then be used to provide projections given new situations of interest and answer "what if" questions.

Learning by Human Brain

The basic building block of a brain and the neural network is the neuron. The basic human neuron adapted from (Beale and Jackson, 1990) is shown in Fig. 1.

As described by (Beale and Jackson ,1990), all inputs to the soma (cell body) of the neuron arrive along dendrites. Dendrites can also act as outputs interconnecting interneurons. Mathema-tically, the dendrite's function can be approximated as a summation. Axons, on the other hand, are found only on output cells. The axon has an electrical potential. If excited past a threshold it will transmit an electrical signal. Axons terminate at synapses that connect it to the dendrite of another neuron. When the

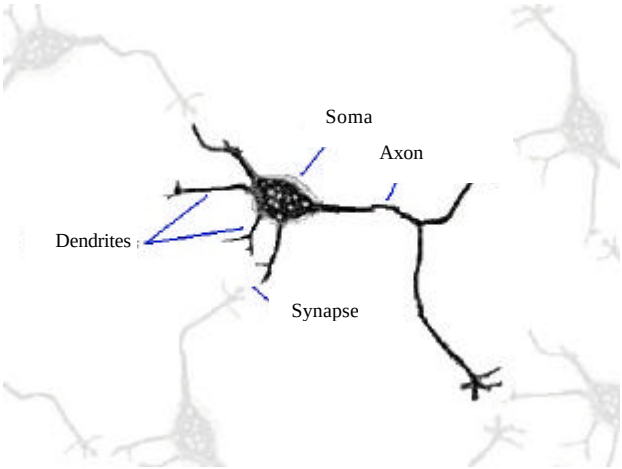

Fig. 1: Components of a neuron

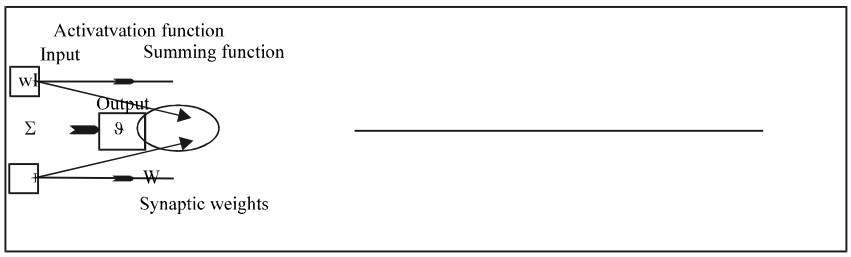

Fig. 2: Artificial neuron model

electrical input to a synapse reaches a threshold, it will pass the signal through to the dendrite to which it is connected. The human brain contains approximately 10 ^{10} interconnected neurons creating its massively parallel computational capability.

From Human Neurons to Artificial Neurons

The artificial neuron was developed in an effort to model the human neuron. The artificial neuron depicted in Fig. 2 was adapted from (Kartalopoulos and Stamatios, 1996) and (Haykin and Simon, 1994). Inputs enter the neuron and are multiplied by their respective synaptic weights. They are then summed and processed by an activation function. The activation function dampens or bound's the neuron's output. Figure 3 represent common activation functions which also happened to be used by the network tested during this research. The logistic or sigmoid function $f(x) = 1/(1+\exp(-X))$.

Back Propagation

Back Propagation (BP) is a method for training multilayer feedforward networks (Fadalla *et al*., 2001). It works by training the output layer and then propagating the error calculated for these output neurons, back though the weights of the net, to train the neurons in the inner (hidden) layers. To understand how BP works consider the network shown in Fig. 4.

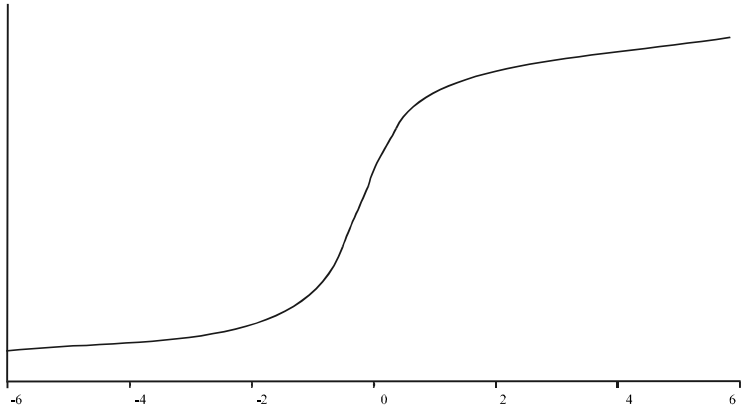

Fig. 3: Logistic activation function

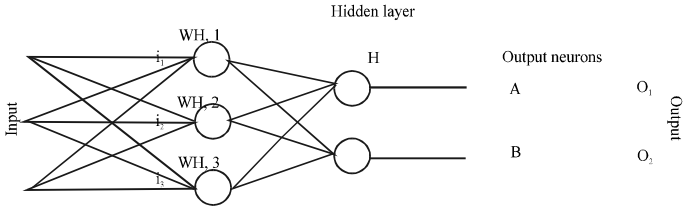

Fig. 4: General multilayer feedforward network

To show the general technique of BP training, consider the neuron in the hidden layer H. This neuron has three weights feeding into it, $w_{H,1}$ $w_{H,2}$ and $w_{H,3}$. The training technique applied to this one neuron can be applied to all the other neurons throughout the network.

First, an input is applied to the network and the output is calculated. The output is then compared with the target and an error is calculated. The error then multiplied by the derivative of the activation function (for example the sigmoid function).

$$\frac{\partial O}{\partial S} = O(1-O) \tag{1}$$

so:

$$\delta_A = O_A(1-O_A)(T_A - O_A) \tag{2}$$

This can be used to calculate Δ_A by multiplying learning rate l, then calculate the new weights for neuron A, as follows

$$\Delta_A = l\,\delta_A \tag{3}$$

$$w_A^+ = w_A + \_\Delta_A \tag{4}$$

Then propagate the value of δ calculated for the output neurons, back through the weights, to the neurons of the hidden layer and hence calculate a value of δ for these. This is done by multiplying the

value of δ from each output layer by the weight connecting that output layer to the hidden layer neuron and adding all the contributions together.

For the neuron H:

$$\delta_H = O_H(1 - O_H).(\delta_A.w_{A,H} + \delta_B.w_{B,H}) \quad (5)$$

and hence, having calculated δ for the neurons in the hidden layer, one can use Eq. 3 and 4 to calculate the weights associated with this neuron ($w_{H,1}$,$w_{H,2}$ and $w_{H,3}$). The process is then repeated for all the hidden layer neurons.

Backpropagation Algorithm

```
Initialize all weights in network;
While terminating condition is not satisfied {
        for each training sample X in sample{
            // propagate the inputs forward:
         for each hidden or output layer unit j{
                I_j=∑_j W_ij O_i ; //Compute the net input of unit j with respect to the previous
                layer,i
            O_j=1/ (1+e^-Ij );}     //Compute the output of each unit j
        //Backpropagate the errors
        for each unit j in the output layer
            Err_j=O_j (1-O_j) (T_j – O_j);//Compute the error
      for each unit j in the hidden layers,from the last to the first hidden layer
            Err_j=O_j (1-O_j)∑_k Err_k W_jk ; //Compute the error with respect to the next higher
layer,k
        for each weight Wij in network{
            ΔW_ij= (l)Err _j O_i //Weight increment
            W_ij=W_ij+ΔW_ij;}//weight update
    } }
```

Fig. 5: Backpropagation algorithm

In the Backpropagation algorithm which is shown in Fig. 5, the weights in the network are initialized to small random numbers (e.g., ranging from -1.0 to 1.0).Then propagate the inputs forward, the input and output of each unit in the hidden and output layers are computed. First, the training sample is fed to the input layer of the network. For unit j in the input layer, its output is equal to its input, that is, $O_j = I_j$ for input unit j. The net input to each unit in the hidden and output layers is computed as a linear combination of its inputs. The inputs to the unit are, in fact, the outputs of the units connected to it in the previous layer. To compute the net input to the unit, each input connected to the unit is multiplied by its corresponding weight and this is summed. Given a unit j in a hidden or output layer,the net input,I_j,to unit j is $I_j = \acute{O}_i W_{ij} O_i$, where w_{ij} is the weight of the connection from unit i in the previous layer to unit j; $O_i$is the output of unit I from the previous layer. Each unit in the hidden and output layers takes its net input and then applies an activation function to it. The function symbolizes the activation of the neuron represented by the unit. The logistic,or sigmoid function is

used. Given the net input Ij to unit j, then O_j, the output of unit j, is computed as $O_j = 1/(1+e^{-Ij})$ (Ward *et al*., 1995). This function is also referred to as a squashing function, since it maps a large input domain onto the smaller range of 0 to 1.

The error is then propagated backwards by updating the weights to reflect the error of the network's prediction (Kryzanowski *et al*., 1993). For a unit j in the output layer, the error Err_j is computed by $Err_j = O_j (1-O_j) (T_j-O_j)$, where O_j is the actual output of unit j and T_j is the true output, based on the known class label of the given training sample. $O_j (1-O_j)$ is the derivative of the logistic function.

To compute the error of a hidden layer unit j,the weighted sum of the errors of the units connected to unit j in the next layer are considered. The error of a hidden layer unit j is $Err_j = O_j (1-O_j)\sum_k Err_k W_{jk}$, where W_{jk} is the weight of the connection from unit j to a unit k in the next higher layer and Err_k is the error of unit k. The weights are updated to reflect the propagated errors. Weights are updated by the following equations, where Δw_{ij} is the change in weight w_{ij}: $\Delta w_{ij} = (l) Err_j O_i$

$$w_{ij}=w_{ij}+ \Delta w_{ij}.$$

The variable l is the learning rate, a constant typically having a value between 0.0 and 1.0. Training stops when

- All Δw_{ij} in the previous epoch were so small as to below some specified threshold, or
- The percentage of samples misclassified in the previous epoch is below some threshold, or
- A prespecified number of epochs has expired.

RESULTS

The algorithm was implemented and tested on a PC with a Celeron D Processor of 2.53 GHZ and 256 MB of main memory running the Windows98SE operating system. We have used Visual Basic and Microsoft Access as Front End and Back End Tool. Simulation data was sourced from Stock Broker Shreekant Agarwal. The data used for network training and verification is comprised of daily figures for individual Equities listed on the National Stock Exchange (NSE) from December 1st 2000 to April 12th 2005. We used close rate,volume of stocks as our input and next stock rate as our output by training networks with 1, 2, 3, 4 and 5 hidden layers by using backpropagation algorithm mentioned above. Normalisation is a key part of data pre-processing for neural networks and should enable more accurate predictable rates (Zirilli, 1997). Normalised data is used for training neural network with backpropagation algorithm. We normalize inputs so that input values lies between 0 and 1. We develop three modules. First module is used for calculating weights by using backpropagation algorithm.Second module is used to find the predictable stock rate after denormalization by using weights we got from first module. Third module is used for getting buying and selling signals from predicted stock rates and the available stock rates. We got predicted stock rate from trained network with different hidden layers, we generate signals to know whether that stock should be buy or sell and then compare it with the actual stock rate signals on those trading days. By this comparison we got correctness of our prediction. After performing this experiment we got different results with their correctness as shown in Fig. 6-10. We also compared the correctness of prediction with hidden layer1, 2, 3, 4 and 5 as shown in Fig. 11.

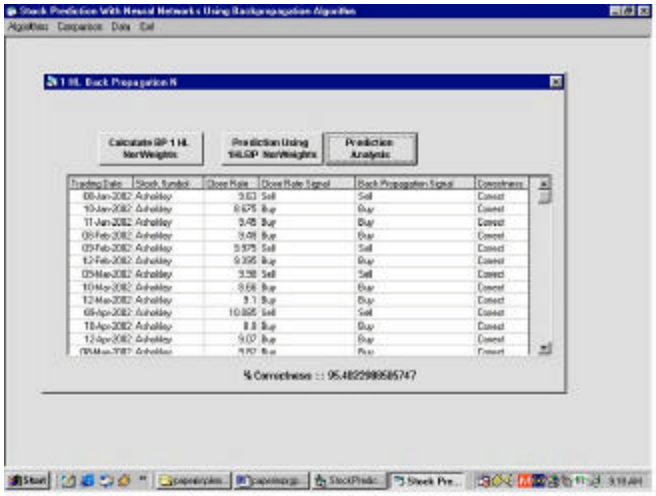

Fig. 6: Results of stock rate prediction using BackPropagation algorithm with one hidden layer

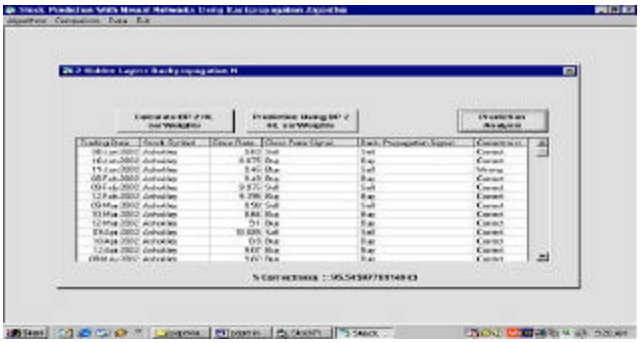

Fig. 7: Results of stock rate prediction using BackPropagation algorithm with two hidden layer

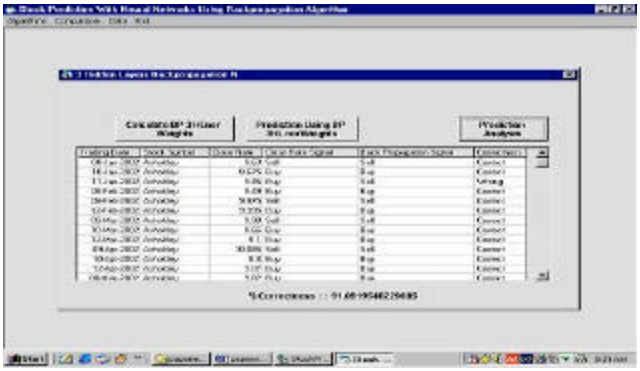

Fig. 8: Results of stock rate prediction using BackPropagation algorithm with three hidden layer

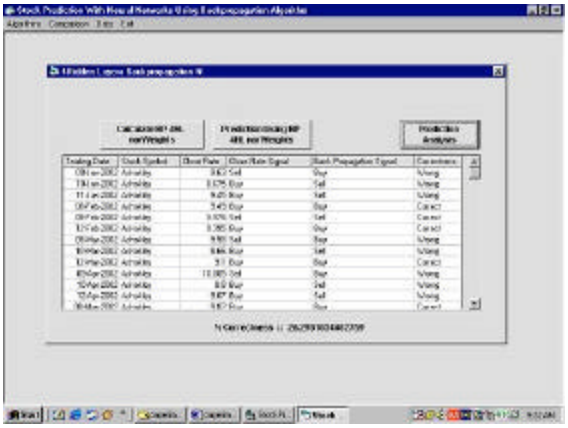

Fig. 9: Results of stock rate prediction using BackPropagation algorithm with four hidden layer

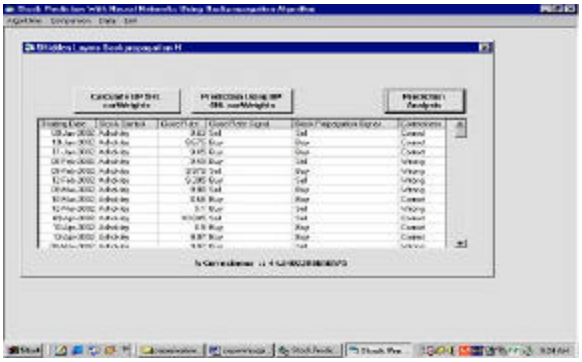

Fig. 10: Results of stock rate prediction using BackPropagation algorithm with five hidden layer

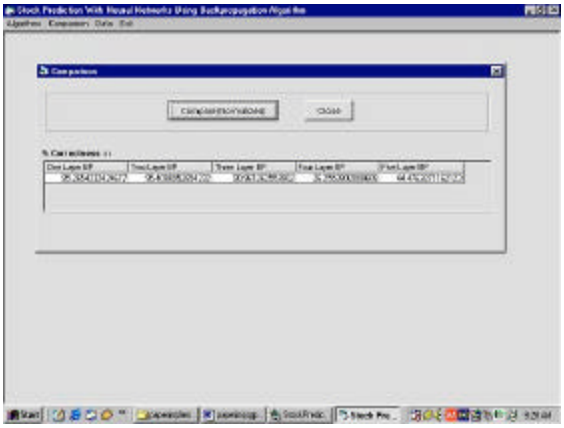

Fig. 11: Results showing comparison of stock rate prediction using BackPropagation algorithm with 1, 2, 3, 4 and 5 hidden layers

CONCLUSIONS

This study has compared the forecasting accuracies of neural networks with backpropagation algorithm using one, two, three, four and five hidden layers. Results showed that for this stock rate prediction problem, the neural networks with backprogation algorithms using two hidden layers has given most accurate prediction in comparison to other layers.

REFERENCES

Beale, R. and T. Jackson, 1990. Neural Computing: An Introduction, Adam Hilger, Bristol, England.

Fadalla, A., Lin and Chien-Hua, 2001. An Analysis of the Applications of Neural Networks in Finance. Interfaces, pp: 112-122.

Gately and J. Edward, 1996. Neural Networks for Financial Forecasting, John Wiley and Sons, New York.

Haykin and Simon, Neural Networks: A Comprehensive Foundation, Macmillian CollegePublishing Company, New York, New York.

Kanas, A., 2001. Neural network linear forecasts for stock returns. Int. J. Finance Econom., 6: 245-254.

Kartalopoulos and V. Stamatios, 1996. Understanding Neural Networks and Fuzzy Logic: Basic Concepts and Applications, IEEE Press, New York.

Kryzanowski, L., M. Galler and D.W. Wright, 1993. Using artificial networks to pick stocks. Financial Analyst's J., pp: 21-27.

Mizuno, H., M. Kosaka, H. Yajima and N. Komoda, 1998. Application of neural network to technical analysis of stock market prediction. Studies Inform. Control, 7: 111-120.

Ward, Steven, Sherald and Marge, 1995. The Neural Network Financial Wizards. Technical Analysis of Stocks and Commodities, Reprinted, Technical Analysis Inc., Seattle, Washington.

Wong, B.K., T.A. Bodnovich and Y. Selvi, 1997. Neural network applications in business: A review and analysis of the literature (1988-95). Decision Support Systems, 19: 301-320.

Zemke, S., 2002. On developing financial prediction system: Pitfalls and possibilities. DMLL Workshop at ICML-2002, Australia.

Zirilli, J.S., 1997. Financial Predictions Using Neural Networks. International Thomson Computer Press, London, pp: 135.