SICE-ICASE International Joint Conference 2006
Oct. 18-21, 2006 in Bexco, Busan, Korea

# Stock Price Forecasting using Back Propagation Neural Networks with Time and Profit Based Adjusted Weight Factors

Nguyen Lu Dang Khoa[1], Kazutoshi Sakakibara[2] and Ikuko Nishikawa[2]

[1] Graduate School of Science and Engineering, Ritsumeikan University, Japan
[2] College of Information Science and Engineering, Ritsumeikan University, Japan
E-mail: {ludangkhoa, sakaki, nishi}@sys.ci.ritsumei.ac.jp

**Abstract**: In this paper, we showed a method to forecast the stock price using neural networks. Predicting the stock market is very difficult since it depends on several known and unknown factors. In recent years, one of the techniques that have been used popularly in this area is artificial neural network. The power of neural network is its ability to model a nonlinear process without a priori knowledge about the nature of the process. We used both feed forward neural network and simple recurrent neural network, trained by time and profit based back propagation algorithm with early stopping to make the prediction. The integration of profit and time factors with training procedure made an improvement in forecasted results for feed forward neural network. Moreover, the simple recurrent neural network with its 'time capture' capabilities had better forecasted results than feed forward neural network in all experiments.

**Keywords:** feed forward neural network, recurrent neural network, back propagation, time dependent directional profit

## 1. INTRODUCTION

Stock price forecasting is an important and hot topic in forecasting financial time series data. Since once the forecasting is successful, we could determine a suitable trading strategy and could get a very high profit. However, predicting the stock market is very difficult since it depends on several known and unknown factors, and frequently the data used for forecasting is noisy, uncertain, and incomplete [1].

Two typically used techniques in stock price analysis are fundamental analysis and technical analysis. Most studies using those techniques attempt to capture the relationship between the available data and the stock prices using linear assumptions. However, the relationship between stock prices and the financial and economic variables is not linear. Therefore, nonlinear models could produce more reliable and correct predictions of stock prices. Although a number of nonlinear statistical techniques have been used to produce better predictions of stock prices, most techniques (model-driven approaches) require that the nonlinear model have to be specified before the estimation of parameters can be determined [1].

In recent years, one of the techniques that have been used popularly in this area is artificial neural network. The novelty of neural network lies in its ability to model a nonlinear process without a priori assumptions about the nature of the process [1].

Using back propagation algorithm, the network can be trained with available data to model an arbitrary system. The trained network is then used to predict the movements in the future.

Traditional training algorithms are based on goodness-of-fit to evaluate the performance. However, in the context of financial forecasting, the fitness between the forecasts and the targets is less important than and sometime does not correspond to the profits. Moreover, in financial data, the structural relationship between an asset price and its determinants changes gradually over time as the economic environment evolves [3]. In order to increase the forecasting ability in terms of profit earning, Yao *et al.* [3] proposed a profit based adjusted weight factor for back propagation in which factors containing the profit, direction, and time information were added to the error function. The new training procedure did improve the forecasting ability of neural network models, for the financial domain.

This paper investigates the systematic ways to use neural networks to predict stock market prices in the future. We examined both feed forward neural network (FFN) and recurrent neural network (RNN) using the above time and profit based back propagation algorithm with early stopping. The network inputs, structures, training strategies were decided based on our experimental results.

## 2. BACK PROPAGATION NEURAL NETWORKS

Neural networks can approximate any nonlinear function. As such flexible function approximators, they are powerful for pattern recognition, classification, and forecasting. The most commonly used neural networks in stock price prediction are FFN and RNN. FFN is common because it is simple, easy to train and has a good performance. RNN has some benefits over FFN because the feedback feature in its network structure could extract time relations in the data, especially in financial domain, and thus improve the prediction. Fig. 1 is a simple RNN proposed by Elman [2].

89-950038-5-5 98560/06/$10 

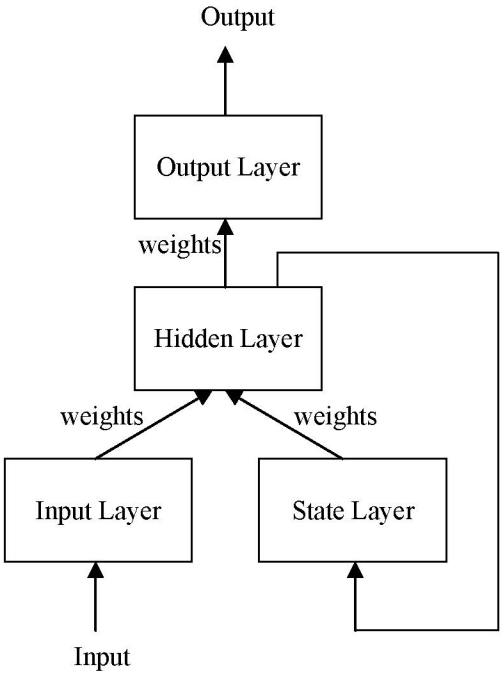


Fig.1. A simple recurrent neural network

### 2.1 Traditional back propagation algorithm

Back propagation is the most widely used algorithm to train neural networks. The training is based on a simple concept: if the network gives a wrong answer, then the weights are corrected so that the error is lessened and as a result, future responses of the network are more likely to be correct [8]. A neural network using back propagation algorithm to train is often called a back propagation neural network.

Back propagation corresponds to a propagation of errors backwards through the network. It involves an iterative procedure for minimization of an error function, with adjustments to the weights. Firstly, the derivatives of the error function with respect to the weights must be evaluated. Then the derivatives are used to compute the adjustment to be made to the weights. The simplest such techniques involves gradient descent [7]. The weights are updated using:

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

where $\Delta w_{t+1}$, $\Delta w_t$ are the weight changes at time $t$+1 and $t$, $\eta$ is the learning rate, $\partial E/\partial w$ is the derivative of the error function with respect to the weights, and $\alpha$ is the momentum coefficient.

### 2.2 Back propagation training using time dependent directional profit

The learning of traditional back propagation neural network used least squared error function:

$$E = \frac{1}{2}\sum_{p=1}^{N}(t_p - o_p)^2 \qquad (2)$$

where $N$ is the number of patterns in the training set, $p$ is the pattern number, $t_p$ is the target value, $o_p$ is the network output value.

However, in financial forecasting, profit gain is the major goal. Taking account of profit in training process could improve the forecast. Yao *et al.* in [3] integrated a Directional Profit (DP) adjust factor with the error function:

$$E_{DP} = \frac{1}{2}\sum_{p=1}^{N} f_{DP}(p)(t_p - o_p)^2 \qquad (3)$$

where $f_{DP}$ is the profit adjust factor which is a function of changes and directions as follows:

$$f_{DP}(p) = \begin{cases} a_1 \text{ if } \Delta t_p \Delta o_p > 0 \text{ and } |\Delta t_p| \le \sigma \\ a_2 \text{ if } \Delta t_p \Delta o_p > 0 \text{ and } |\Delta t_p| > \sigma \\ a_3 \text{ if } \Delta t_p \Delta o_p < 0 \text{ and } |\Delta t_p| \le \sigma \\ a_4 \text{ if } \Delta t_p \Delta o_p < 0 \text{ and } |\Delta t_p| > \sigma \end{cases} \qquad (4)$$

where $\Delta t_p = t_p - t_{p-1}$ and $\Delta o_p = o_p - t_{p-1}$ consist of the information of changes and directions of the network and target values of price. $\sigma$ is a threshold of the changes. The authors chose $\sigma$ as the standard deviation of the training set. According to [3], $a_1 = 0.5$, $a_2 = 0.8$, $a_3 = 1.2$, and $a_4$=1.5.

In Eq. (3), the weights will be adjusted more if a wrong direction is forecasted for a big change. The weights will be adjusted less if a right direction is forecasted for a big change.

Moreover, [3] said that Refenes *et al.* proposed the Discounted Least Squared (DLS) function:

$$E_{DLS} = \frac{1}{2}\sum_{p=1}^{N} w(p)(t_p - o_p)^2 \qquad (5)$$

where $w(p)$ is an adjustment of the contribution of observation $p$ to the overall error:

$$w(p) = \frac{1}{1 + e^{\left(a - \frac{2ap}{N}\right)}} \qquad (6)$$

Refenes *et al.* emphasized the recent information. Thus recent observations should be weighted more heavily than older observations. In the Eq. (6), $a$ is the discount rate. Refenes *et al.* suggested a = 6.

Yao *et al.* formula emphasizes on the profit while Refenes *et al.* formula emphasizes on time. Yao *et al.* proposed the Time dependent Directional Profit (TDP)

error function [3] which includes emphasis on both profit and time concept:

$$E_{TDP} = \frac{1}{2}\sum_{p=1}^{N} f_{TDP}(p)(t_p - o_p)^2 \qquad (7)$$

where $f_{TDP}(p) = f_{DP}(p) \times w(p).$ (8)

In all the above error formulas, the weight change is the same as the approach used in the traditional back propagation algorithm in Eq. (1) except that an adjustment factor is introduced.

$$\Delta w_{t+1} = -f_{TDP}(p)\eta \frac{\partial E}{\partial w} + \alpha \Delta w_t \qquad (9)$$

However, in general, big changes in prices between two consecutive periods are rare to occur. So a change in two consecutive prices is much smaller comparing with the standard deviation of the price. That means the inequality $|\Delta t_p| > \sigma$ in Eq. (4) rarely occurred. Instead of using such inequality, we used the new inequality $\Delta t_p / t_{p-1} > a_5$ as a variant of Eq. (4):

$$f_{DP_N}(p) = \begin{cases} a_1 \text{ if } \Delta t_p \Delta o_p > 0 \text{ and } \Delta t_p / t_{p-1} < a_5 \\ a_2 \text{ if } \Delta t_p \Delta o_p > 0 \text{ and } \Delta t_p / t_{p-1} > a_5 \\ a_3 \text{ if } \Delta t_p \Delta o_p < 0 \text{ and } \Delta t_p / t_{p-1} < a_5 \\ a_4 \text{ if } \Delta t_p \Delta o_p < 0 \text{ and } \Delta t_p / t_{p-1} > a_5 \end{cases} \qquad (10)$$

The parameter $a_5$= 0.05 in experiments gave best performance. That means 5% gain in profit was chosen as the threshold to choose the penalty for the error function. The parameters $a_1$, $a_2$, $a_3$, and $a_4$ were also chosen like Yao *et al.* [3].

The weight change is the same as Eq. (9)

$$\Delta w_{t+1} = -f_{TDP_N}(p)\eta \frac{\partial E}{\partial w} + \alpha \Delta w_t \qquad (11)$$

where $f_{TDP_N}(p) = f_{DP_N}(p) \times w(p).$ (12)

## 3. NEURAL NETWORKS METHODOLOGY

The networks tried to predict the Standard and Poor 500 (S&P 500) stock index one month in the future. S&P 500 index is the composite index reflecting the stocks price of 500 largest companies in the U.S. It is seen as the benchmark of the U.S. stock market.

### 3.1 Background of the financial prediction

For many years, the topic of market efficiency has been attracted a widespread interest and considerable research. The issues which have been hotly debated are the random walk and the efficient market hypothesis. The former theory assumes that prices are stochastic in nature, while the latter theory implies that profit opportunities do not exist in perfectly efficient markets [9]. That means both theories imply that in well functioning markets, prices are unpredictable.

However, there are theoretical arguments and empirical research seriously questioning these two theories, and there is still no general agreement over the validity of both the random walk and the efficient market hypotheses. Moreover, there is more and more considerable evidence to prove that markets are not fully efficient. In fact, many researchers provide evidence that stock market prices are predictable by publicly available information such as time-series data of financial and economic variables [9].

Forecasting is a process that produces a set of outputs by a given set of variables. The variables are normally past data. Basically, forecasting assumes that future values are based, at least in part, on past data. Past relationships can be found through study and observation. The ideas of forecasting using neural networks is to find an approximation of mapping between the input and output data through training. The trained neural networks are then used to predict the values for the future.

### 3.2 Inputs selection

The input variables for the financial domains can be divided into two categories: the first are inputs based on fundamental factors such as the prices of oil, the interest rates, inflation, GDP, and so on. The second contains technical indicators such as volatility, relative strength index, directional index, daily high, low and closing prices, moving averages, and similar indicators derived from the item to be forecasted [5].

Our goal is to develop a network that uses as few independent inputs as possible and still achieves the desired accuracy. Too many independent inputs do not necessary improve the learning capabilities of the network and allow a greater chance of over-fitting [5]. However, the presence of several technical indicators could lead to the better prediction. The following 3 indicators were considered:

- *The S&P 500 price* (*SP*)
- *The 3-month treasury bill* (*3TB*)
- *The 10-year treasury bond* (*120TB*)

The short term treasury bills (T-bills) and the long term treasury bonds (T-bonds) are interest rates issued by the U.S. government and reflect many factors of the U.S. economy that could affect to the stock market. From these indicators, the following 12 technical indicators were created as the network inputs:

- *The absolute difference between SP and its previous value, and the relative trend of SP. These inputs along with SP make up the first 3 inputs of the network.*

- *5-months moving average of SP and its absolute difference and relative trend.*
- *5-months moving average of 3TB and its absolute difference and relative trend.*
- *5-months moving average of 120TB and its absolute difference and relative trend.*

The use of moving average (the average of some lags value) could remove the seasonal features of the time series data [4]. The absolute difference is defined as $|P_t-P_{t-1}|$. The relative trend equals to -1 if $P_t<P_{t-1}$, to 0 if $P_t=P_{t-1}$ and to +1 otherwise.

All the time series are monthly data and were collected from January 1990 to July 2005. They all were linearly scaled to the interval [0.1, 0.9]. A normalization technique where input variables are scaled within a specified range is particularly useful for modeling the neural network. It minimizes the effect of magnitude among the inputs and thus facilitates the neural network in learning the relevant relationships [1].

The Fig. 2 is the raw time series of the S&P 500 stock price.

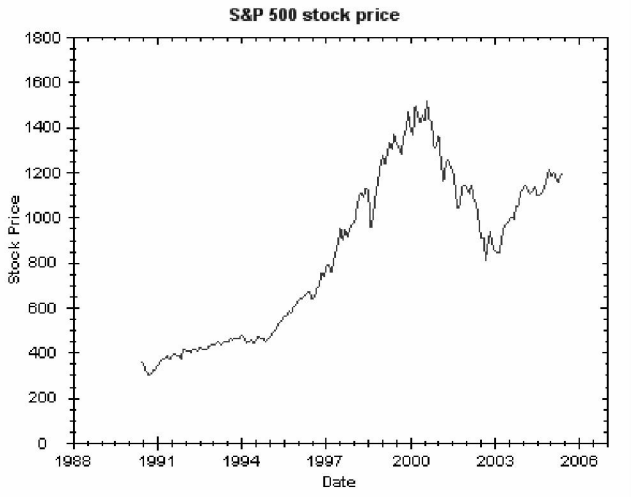


Fig.2. A S&P 500 time series

### 3.3 Network selection

We used FFN and RNN to examine the prediction. For RNN, we used a simple RNN proposed by Elman in Fig. 1. It is basically different from FFN in the sense that it works not only on an input space, but also on an internal state space. This network allows the neurons to depend not only on the input variables, but also on their own lagged values.

Designing the architectures of the networks involves much trial and error. Trial and error is used to find the initial weight range, the number of hidden units in the hidden layer, the learning rate, and the momentum coefficient, etc.

Both networks have 3 layers. The number of hidden units is chosen through the experimental performance. As the results, the hidden layer with 3 units performed best for FFN, while the hidden layer (and also the state layer) with 1 unit had the best performance for RNN.

### 3.4 Training strategy

The major problem in training a neural network is deciding when to stop training. If we stop the training unsuitably, the network may be over-fitting. Over-fitting occurs when the system memorizes patterns and thus looses the ability to generalize (or predict) input data that it has never seen. Over-fitting can occur by having too many hidden nodes or training for too much time.

The network was trained by back propagation algorithm with early stopping, a technique to avoid over-fitting [6]. It was trained with a training set and then was validated with a validating set. The validating set was used to check the generalization ability of the trained network. Every 200 training cycles, mean squared error (MSE) of the validating data was examined. The error told us whether the network was over-fitting or not. If it decreased, the training was continued. Otherwise the training was stopped.

## 4. EXPERIMENTAL RESULTS

In the experiment, all weights and thresholds were initialized in interval [-0.1,+0.1], the learning rate was 0.1, and the momentum coefficient was 0.5. The training set took 85%, the test set took 15% of the whole data set (total 181 patterns). The validating set took 20% of all training patterns. The training, validating, and testing sets were took in succession from the whole data set.

The data were trained with both FFN and RNN. In each experiment, the networks were trained and tested 50 times with different initial weights and we took the average results. The results were shown in Tables 1 and 2. In Table 1, the comparisons among the ordinary least square training (OLS), the time dependent directional profit training proposed by Yao *et al.* (TDP) [3], and our time dependent directional profit training ($TDP_N$) using FFN are shown. The results shown in Table 2 are also the comparisons among these 3 different training techniques using RNN. All the trainings used early stopping strategy.

In stock trading, the trend precision is very important (the trend precision is the ratio that the price trend is correctly predicted). Forecasted results will be meaningless if we predict the trend incorrectly no matter how close are the forecasted values to the target values. So we included the trend precision as the criteria to evaluate the results. Moreover, we calculated a trading profit as follows: when a forecasted value and its target value had the same trend, that is, increased or decreased at the same time comparing with the last target value, we added the target price change to the profit as a gain. When trends of a forecasted value and its target value were different, we subtracted the target price change from the profit as a loss. And then we took the monthly average value of the profit. The trading transaction costs were ignored in calculating the profits. The standard deviation is the deviation among different trials in the

experiments.

Table 1. Comparisons among traditional training and time dependent directional profit trainings using FFN

| | OLS | TDP | $TDP_N$ |
|---|---|---|---|
| MSE | 0.0021 | 0.0022 | 0.0022 |
| Trend precision | 68.93 % | 69.86 % | 70.00 % |
| Monthly profit | 12.98 | 13.39 | 13.47 |
| Standard deviation | 1.18 | 0.66 | 0.39 |

Table 2. Comparisons among traditional training and time dependent directional profit trainings using RNN

| | OLS | TDP | $TDP_N$ |
|---|---|---|---|
| MSE | 0.0006 | 0.0009 | 0.0040 |
| Trend precision | 75.43 % | 73.79 % | 73.00 % |
| Monthly profit | 16.22 | 15.19 | 14.68 |
| Standard deviation | 1.43 | 1.56 | 1.51 |
| Profit gain | 24.96% | 13.44% | 8.98% |

The profit gain in Table 2 is a gain in profit of the forecast using RNN comparing with the profit of the forecast using FFN trained by the same techniques.

The results in Table 1 and 2 show that for every training technique, RNN always forecasted better than FFN ranging from about 9% to 25% in profit gain.

In case of FFN, the time dependent directional profit training of Yao *et al.* (TDP) made higher profit than the traditional training (OLS). Our proposed training algorithm performed best with the most trading profit. That means the new weight adjustment training technique did focus on patterns which had the big changes in price in case of FFN. Moreover, the standard deviation among different trials of $TDP_N$ technique was the smallest, proving that it was the most stable technique.

However, the new techniques might not suitable for applying to RNN. Both the TDP and $TDP_N$ did not have better forecast than OLS technique. The feedback structure in RNN might worsen the factors integrated to the error function in the new training techniques.

## 5. CONCLUSION AND FUTURE WORKS

This paper showed a systematic method to forecast the times series stock price using neural networks with back propagation. The integration of profit and time factors to the training algorithm made improvements comparing with the traditional training for FFN. Moreover, simple RNN with its 'time capture' capabilities had better prediction results than FFN in all cases.

However, the new methods did not work for RNN and did not have considerable improvements in the experiments for FFN. Moreover, the weight adjustment function $f_{TDP_N}$ has not been examined completely yet. Therefore, the values of parameters in function $f_{DP_N}(p)$ and $w(p)$ should be inspected more to gain higher profit. Besides, an investigation of applying those techniques to other data sets should be considered.

In the future, we plan to integrate neural network with some other techniques such as genetic algorithm or fuzzy logic. Genetic algorithm can be used to identify optimal network architecture and training parameters. Fuzzy logic provides the ability to account for some uncertainty produced by the neural network predictions [1]. Their uses in conjunction with neural network could provide an improvement for stock market prediction.

## REFERENCES


[1] S. Thawornwong and D. Enke, "Forecasting stock returns with artificial neural networks". In G. P. Zhang (ed.) *Neural Networks in Business Forecasting* (Chapter 3, pp. 47-79). Idea Group Publishing, 2004.

[2] J.L. Elman, "Finding structure in time". *Cognitive Science*, 14, pp. 179-211, 1990.

[3] J.T. Yao, C.L. Tan, "Time dependent directional profit model for financial time series forecasting". *In proceeding of the IJCNN, Como, Italy,* Vol. 5, pp. 291 - 296, 2000.

[4] P. D. Mc.Nelis, *Neural networks in Finance. Gaining predictive edge in the market*, Elsevier Academic Press, 2005.

[5] E. Gately, *Neural networks for financial forecasting*, John Willey & Sons, 1996.

[6] S. Thawornwong and D. Enke, "The adaptive selection of financial and economic variables for use with artificial neural networks", *Neurocomputing*, Vol. 56, pp. 205-232, 2004.

[7] C. M. Bishop, *Neural networks for pattern recognition*, Oxford University Press, 1996.

[8] J. E. Dayhoff, *Neural network architectures. An introduction*, International Thompson Publishing, 1990.

[9] J. Shadbolt and J.G. Taylor, *Neural Network and the Financial Markets. Predicting, Combining and Portfolio Optimisation*, Springer, 2002.