A New Approach to Neural Network based Stock Trading Strategy

Miroslaw Kordos, Andrzej Cwiok

University of Bielsko-Biala, Department of Mathematics and Computer Science,
Bielsko-Biala, Willowa 2, Poland: mkordos@ath.bielsko.pl

Abstract. The paper presents an idea of using an MLP neural network for determining the optimal buy and sell time on a stock exchange. The inputs in the training set consist of past stock prices and a number of technical indicators. The buy and sell moments on the training data that will become the output to the neural network can be determined either automatically or manually by a user on past data. We discuss also the input space transformation and some improvements to the backpropagation algorithms.

Keywords: neural networks, stock prediction

1 Introduction

Prediction of stock prices is considered to be a very difficult problem. However, what we really need to trade effectively is not to predict the future stock price but the optimal moment to buy or sell the stock. On one hand machine learning methods can be used to predict the stock price and on the other hand technical analysis indicators determine the optimal transaction points. To all of that the human factor is added since it is the user of the trading software, who can finally decide whether to agree or not with the program decision.

In our approach we try to take into account the best features of all the three sources (machine learning, technical analysis and human factor) together to build a robust trading system.

The first market hypothesis was that stock prices follow a random walk [1]. However, researchers and economists were able to extract rules associated with the stock price movements. That kind of rules can significantly increase the quality of their predictions [1, 2].

Since stock prediction is a complex, non-deterministic issue, it is a good ground for the application of artificial intelligence. During the last two decades, stocks and futures traders stared to believe in the decisions made by various types of intelligent systems, based on different algorithms. The list of implemented solutions includes among others: genetic algorithms used to find optimal values of various technical indicators and their combinations thus reducing the computational complexity of the search [**?**,4], fuzzy logic controllers and fuzzy neural networks [5, 6], Hidden Markov Models to assess the parameters of a Markov process that best fits the stock price fluctuations [7]. Neural networks were the most widely used methods, because of their superior performance [8–13].

However, even the best model is only as good as the data it is trained on. In stock price prediction selection of the right input variables is a much more difficult problem than building and training the predictive model itself. In our work we choose to use MLP neural networks as the model and focus on searching for an optimal set of input variables, which allows to maximize the gains in the trade rather than only predict the stock price most accurately. We also discuss a simple improvement to the backpropagation algorithm.

2 Input Variables

In this section we describe the indicators we use as input variables and how we transfer and combine them to generate the training set for the neural network as well as the way of determining output variables. Technical indicators indicates the time when one should buy or sell stocks and by default they use daily closing prices. Only the candle formations use the opening, daily minimum, maximum and closing price. The indicators are only shortly listed here due to limited space. An in-depth analysis can be found in [14].

Simply Moving Average (SMA) shows the average value of stock price over a period of time, while Exponential Moving Average (EMA) pays more attention to recent prices:

$$SMA(t) = \frac{1}{n}\sum_{i=0}^{t} C_i \tag{1}$$

$$EMA(t) = \frac{2}{n+1}C_t + (1 - \frac{2}{n+1}) \cdot SMA(t) \tag{2}$$

where C_t is a stock closing price at day t. 15 days SMA is frequently the default. A moving average crossing the actual price from down when the price rises generates a buy signal, while a MA crossing the price from top generates a sell signal. Moving Average Convergence/Divergence (MACD) is a difference between two exponential moving averages: a shorter and a longer one (typically 12 and 26 days):

$$MACD(t) = EMAshort(t) - EMAlong(t) \tag{3}$$

Momentum oscillators measure the speed of change of a stock price. As the price is rising/falling, the momentum increases/decreases proportionally to the speed of the price rise/fall. Relative Strength Index (RSI):

$$RSI = 100 - [100/(1 + RS)] \tag{4}$$

where RS = (Average Gain of n-day up)/(Average Loss of n-day down). A high RSI means the market is rallying suddenly and a low RSI that the market is selling off suddenly. Rate of Change (ROC) indicator is at a high peak and is beginning to move down generates a sell signal. A ROC at a low peak that is beginning to move up is a buy signal. The advantage of a ROC oscillator in comparison to moving average based indicators is that it gives signals before the actual change in the direction of a stock price occurs.

$$ROC = [(Today'sclose - CloseNdaysago)/(CloseNdaysago)]) \cdot 100 \quad (5)$$

The Commodity Channel Index (CCI) is designed to identify cyclical turns in commodities. It is recommended to use 1/3 of a complete cycle as a time frame for the CCI. If the cycle runs 60 days (a low about every 60 days), then a 20-day CCI would be recommended. For the purpose of this example, a 20-day CCI is used:

$$CCI = (TP - SMATP)/(0.015 \cdot MD) \quad (6)$$

where : Typical Price (TP) = (H+L+C)/3 where H = high, L = low, ad C = close, SMATP is a 20 day-period SMA of the Typical Price (SMATP), MD is calculated as the mean deviation of the TP over the past 20 periods. From oversold levels, a buy signal is given when the CCI moves back above -100. From overbought levels, a sell signal is given when the CCI moved back below +100. The Average True Range (ATR) is calculated with the following steps: multiply the previous 14-day ATR by 13, add the most recent day's TR value, divide by 14. Extreme levels (both high and low) can mark turning points or the beginning of a move. Stochastic Oscillator (STO) is a momentum indicator that shows the location of the current price relative to the high/low range over a set number of periods. Price levels that are consistently near the top of the range indicate accumulation (buying pressure) and those near the bottom of the range indicate distribution (selling pressure).

$$STO = 100 \cdot (RecentClose - LowestLow)/(HighestHigh - LowestLow) \quad (7)$$

We use also two candle formation in our analysis, which shows the reversion of the current trend. A bullish hammer, which is a buy signal that occurs after an established downtrend and a bearish shooting star, which is the opposite of the hammer. Both formations indicate that the price extremum was reached during the day and the trend is now likely to reverse [15].

The training set consists of the following 15 features:

- Price change in 1,2 and 3 days in relation to the current price
- SMA with periods of 6, 9, 14 and 21 days
- ROC with periods of 6, 9, 14 and 21 days
- RSI with periods of 6, 9, 14 and 21 days
- CCI with periods of 6, 9, 14 and 21 days
- STO with periods of 6, 9, 14 and 21 days
- ATR with periods of 6, 9, 14 and 21 days
- The candle formation (hummer and shooting star)

That may seem an excessive number of parameters, but as our experiments showed it gives significantly better results than using only a single length period indicators (e.g. 14 days only). Moreover, neural networks have the ability to perform feature selection by setting the weights that connect the less important features to the network to very low values. While most other works try to predict the stock price, we try only to predict the optimal moment of buying and selling stocks.

The output of the training set can be determined automatically, but our software also enables the user to determine the output manually by choosing the optimal buy and sell moments, which will become the outputs in the training data set. In this way the user can balance different trading strategies, mostly the gain-risk trade-off according to their preferences.

We found that it very difficult to obtain good results when the network was trained on the exact data (that is with one day resolution). The result of that was, that different technical analysis indicators give signals in different days. Usually the difference is one or two days. However, if we consider only single day signals, they very rarely are enough strong to cause the neural network to generate a buy or sell signal.

Therefore if we determine the best day to buy or to sell stocks, first we shift the signal one day earlier (it gives better results) and then we spread the buy/sell signals (the network targets) two days backward and two days forward. However, the signal had the value of one on a given day, of 0.67 on the day before and after and of 0.33 on the days two day apart from the current date. We consider the network buy or sell signal correct if it appears on o given day or up to two days before or after. We do not discuss here the choice of the right stock to trade.

3 Neural Network Architecture and Training

We use two separate networks. One is trained to recognize the buy signals and the other one to recognize sell signals. When the stock is once bought it is kept until the first sell signal occurs, even if there are more buy signals in the meantime. We use a multilayer perceptron with two hidden layers and with hyperbolic tangent activation functions in all layers, including the input one [16]. A good practice is to standardize the data before the training to make particular inputs independent of their physical range. It may be also be beneficial to remove the outliers from the training set. Moreover, a model with higher sensibility in the intervals with more dense data may sometimes be preferred. To address the problem, the idea of transforming the data to make it distributed more evenly was proposed. For example, the data can be transferred by a hyperbolic tangent function.

The other advantage of such a transformation is the automatic reduction of the outliers' influence on the model. We do not consider the outliers as erroneous values and thus do not reject them, but rather reduce their influence on the final model. The neural network can either learn the optimal slopes of the transfer function during the training or they can be set a priori.

We tried three learning algorithms: standard backpropagation with momentum, modified backpropagation and variable step search algorithm [17]. In the modified backpropagation we used a directional minimization and a change of

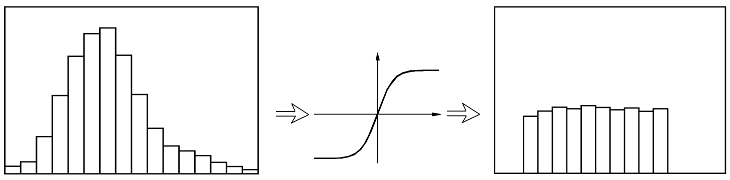

Fig. 1. The idea of transforming data from a Gaussian-like distribution to uniform distribution.

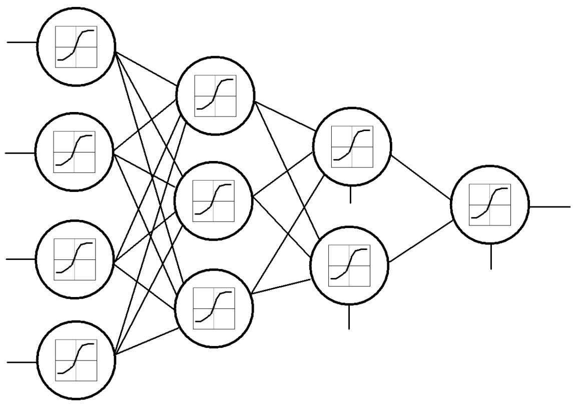

Fig. 2. Neural Network Architecture.

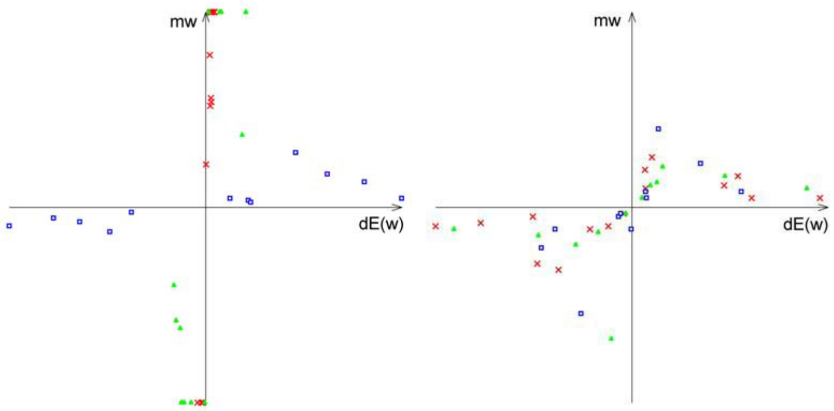

Fig. 3. Dependence between the gradient component dE(w) and the distance from the actual point actual point to the error minimum in a given weight direction *mw* at the beginning (left) and at the end of the training (right) cross = first hidden (counting from input), triangle = second hidden, square = output layer).

the direction by making the gradient components not linearly proportional to that given by the backpropagation algorithm, but rather proportional to the root square of them. Moreover, we multiplied the gradient component in the first hidden layer by four and in the second hidden layer by two to better optimize the trajectory direction (Fig. 3.) without reverting to the second order methods. Based on the experiments we conducted with several datasets as well for classification as for regression tasks (the current task is closer to classification) the following formula can be used to optimize the next step length $dS(w)$ in each weight direction in the backpropagation algorithm:

$$dS(w) = (1 + a \cdot \exp(-bT_c)) \cdot sign(dE(w))) \cdot \sqrt{(|dE(w)|))} \tag{8}$$

where a=0 for the output layer, $a = 4 \pm 1$ for the second layer (closer to the output), $a = 15 \pm 5$ for the first hidden layer weights, $b = 0.12 \pm 0.3$. $dE(w)$ is the gradient component in weight w direction and T_c is the training cycle (epoch). The simplest explanation of the equation is contained in Fig. 3.

4 Experimental Results

We created software for this project as a Windows Forms Application in C#. The application including the source code can be downloaded from our web site [19]. Using that application users can select with a mouse the moments used to train the network and can also obtain the results in a graphical form. We performed ten experiments with four stocks of the USA market: Amazon, Apple, Microsoft and Yahoo, the average results are reported in Table. 1. The training set comprised the data from 1/1/1995 to 12/31/2004 and the test set the data from 1/1/2005 to 1/1/2008.

We performed some simple tests in the Amibroker software [18], where we optimized the period of moving average (opt. SMA) on the training sets and then performed tests on test data. Then we compared the results also with the typical 15 day SMA and with a buy and hold strategy. However, as it can be seen from Table 1, the optimization of SMA does not cause any improvement in the prediction results. Sample signals, which the network generated on the Microsoft stock are shown in Fig 4. Only the MLP output signals that have the value of 0.8 or more are interpreted as effective buy or sell signals and only these signals are shown.

Table 1. The profit (in percent respect to the price on 1/3/2005) for the test period on four stocks.

stock	APPL	MSFT	YHOO	AMZN
buy and hold	518	33	-39	108
opt. SMA	12	17	-36	-1
SMA15	116	10	-20	8
Our Method	425	53	14	184

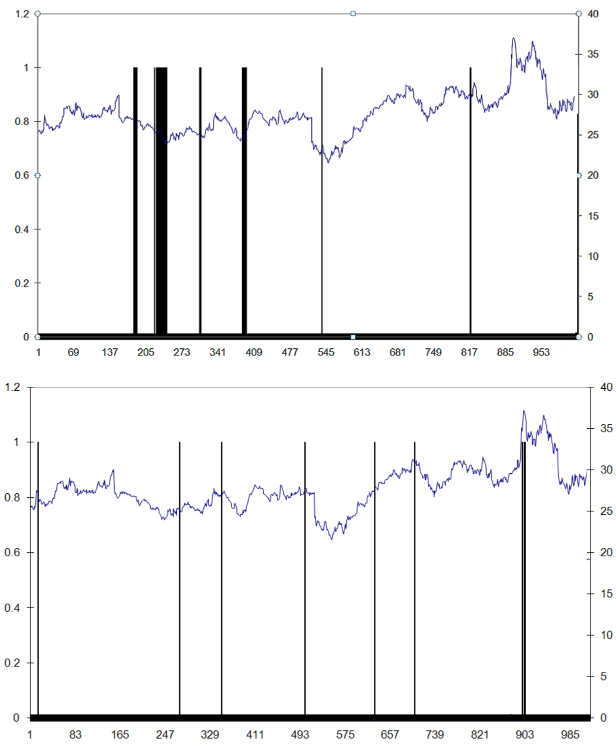

Fig. 4. Closing prices for Microsoft stock with buy (top) and sell (bottom) signals generated by the neural network.

5 Conclusions

We presented a method for determining optimal buy and sell moments at stock exchange using a neural network based prediction. The output of the training set is either chosen automatically or determined by a user, who chooses the optimal moments themselves. The input consists of past prices and a number of technical indicators with 6, 9, 14 and 21 days each. The method seems to be a very interesting approach that allows for including many different factors as well past prices and technical indicators, as some other information (e.g. financial condition of the company). We are currently working a feature selection method, which seems to be the most important task in improving the method.

Also the modification to the backpropagation direction connected with an approximate linear search along the modified step direction makes the backpropagation algorithm much more efficient in the terms of better convergence and convergence in cases where the standard backpropagation algorithms cannot find a satisfactory solution.

References

1. Gencay, R., Linear, non-linear and essential foreign exchange rate prediction with simple technical trading rules. Journal of International Economics, vol. 47, pp. 91–107 (1999)
2. Zhou, W. X., Sornette, D., Testing the stability of the 2000 US stock market antibubble. Physica A: Statistical and Theoretical Physics, vol. 348(15), pp. 428–452 (2005)
3. Tay, F. E. H., Cao, L. J., Modied Support Vector Machines in Financial Time Series Forecasting. Neurocomputing vol. 48(1-4), pp. 847–861 (2002)
4. Shu-Heng Che, Computationally intelligent agents in economics and finance. Science Direct, vol. 177, issue 5, pp 1153–1168 (2007)
5. Gradojevic,N., Non-linear, hybrid exchange rate modeling and trading profitability in the foreign exchange market. Journal of Economic Dynamics and Control, vol. 31, issue 2, pp. 557–574 (2007)
6. Pei-Chann Chang, Chen-Hao Liu, A TSK type fuzzy rule based system for stock price prediction. Expert Systems with Applications, vol. 34, issue 1, pp. 135–144 (2008)
7. Md. Rafiul Hassan, Baikunth Nath, StockMarket Forecasting Using Hidden Markov Model: A New Approach. ISDA, pp. 192–196 (2005)
8. Chandima, D., Tilakaratne, Musa A. Mammadov, Sidney A. Morris, Development of Neural Network Algorithms for Predicting Trading Signals of Stock Market Indices.
9. Khan, A., et. al., Stock Rate Prediction Using Backpropagation Algorithm: Results with Different Numbers of Hidden Layers, Journal of Software Engineering No. 1., pp. 13–21 (2007)
10. Mpofu, N., Forecasting Stock Prices Using a Weightless Neural Network. Journal of Sustainable Development in Africa, vol. 8 (2006)
11. Zorin, A., Stock Price Prediction: Kohonen Versus Backpropagation. In: Proceedings of the International Conference on Modeling and Simulation of Business Systems, pp. 115–119, Vilnius, Lithuania (2003)
12. Afolabi ,M., Olude, O., Predicting Stock Prices Using a Hybrid Kohonen Self Organizing Map (SOM). In: Proceedings of the 40th Hawaii International Conference on System Sciences (2007)
13. Klassen, M., Investigation of Some Technical Indexes inStock Forecasting Using Neural Networks. In: Proceedings of World Academy of Science, Engineering and Technology, vol. 5 (2005)
14. http://stockcharts.com/school/doku.php?id=chart_school
15. http://forex-candelistic.blogspot.com/2008/09/trade-with-candlestick-part-4.html
16. Kordos, M., Neural Network Regression for LHF Process Optimization. Lecture Notes in Computer Science, vol. 5506, pp. 453–460 (2009)
17. Kordos, M., Duch, W., Variable step search algorithm for feedforward networks. Neurocomputing, vol. 71, issues 13–15, pp. 2470–2480 (2008)
18. http://www.amibroker.com
19. Software used in this work, http://www.kordos.com/pdf/nnstock.zip