Neuro-Genetic System for DAX Index Prediction

Marcin Jaruszewicz and Jacek Mańdziuk

Faculty of Mathematics and Information Science, Warsaw University of Technology,
Plac Politechniki 1, 00-661 Warsaw, POLAND
{jaruszewicz,mandziuk}@mini.pw.edu.pl

Abstract. The task of stock index prediction is presented in this paper. The data is gathered at the target stock market (DAX) and two other markets (KOSPI and DJIA). The data contains not only raw numerical values from the markets but also indicators pre-processed in terms of technical analysis, i.e. oscillators and patterns. Statistical analysis and the genetic algorithm are used to create the proper sequence of inputs from all available variables. Selected data is input to a neural network trained with backpropagation with momentum. The prediction goal is the next day's closing value of German stock market index DAX with consideration of Korean and USA stock markets' indexes. The prediction is performed within a tight time-window in order to protect the model against changing relationships between variables. For each time-window the best neural network is evolved and applied. The evaluation is repeated for every time-window in order to discover a new set of proper input variables.

1 Introduction

Despite a lot of effort that has already been devoted to financial time series predictions based on various techniques (e.g. mathematical models [1], support vector machines [2, 3], neural networks [4–6]) or the use of genetically developed prediction rules [7] prediction of a stock market index remains an uneasy goal. The main reason of the complexity of this task is the lack of the autocorrelation of an index value changes even in a period of one day [8]. Moreover, the instability of prediction is increased by the influence of non-measurable incidents like economic or political situation, catastrophe or war that may change the value of stock market index [8]. The other problem concerns proper selection of input variables that are taken into account in the prediction process. Although there exist some general practice which supports selection of inputs (financial indicators) [7] proper choice of this data remains a challenge for both human and artificial agents. This issue, in our opinion, is one of the impediments in building efficient financial prediction systems. The idea of how to approach this problem presented in the paper relies on defining short-term suboptimal sets of inputs which are changing in time based on the repeatable application of genetic optimization procedure.

This paper is a continuation of our prior work [9] where the efficacy of a large modular neural network was examined. Here we propose a different approach relying on the use of genetic algorithm and simple neural network for short history based learning. The role of genetic algorithm is to make pre-selection of input variables for further neural network learning.

In the next sections the data pre-processing method and genetic algorithm for selecting input variables are described in detail. Finally, based on the above two steps the final prediction system is created and tested.

2 Data pre-processing

The data was collected on a daily basis. For each day the sample was composed of the date, the opening, the closing, the highest and the lowest values[1].

Having values of a stock market index statistical variables were calculated: percentage changes in different periods and weighted moving averages for different periods. The closer look at these variables can by found in [9]. Finally technical analysis data was evaluated. Eight different oscillators [10] were calculated: MACD, Williams, Impet, Rate of Change, RSI, Stochastic Oscillator, Fast Stochastic Oscillator, 2 Averages.

Moreover patterns known in technical analysis [10] were extracted. An extraction is based on an analysis of the shape of the chart of the index value. In the technical analysis theory these patterns forecast change or continuation of a trend. Information about patterns and signals of trend changes generated according to them were also included in the learning data as "struct" and " type of struct"

All "buy" signals were coded as 0.8 value and all "sell" signals as -0.8. Information about signal was copied into samples through the next 5 days with values linearly decreasing in time. It was done to prevent the existence of too many samples with no signal value. Anyway, in practice, the signal is usually correct during the next few days after its appearance. Signals were cumulated when appeared more than once in a single record. All above data was generated for three stock markets from different continents: KOSPI, DJIA, DAX and for the rate of exchange USD to JPY and EUR to USD. After all the above calculations 2 420 records were created covering dates 1995/01/31 – 2004/09/02. In the experiment the last 200 records out of the above number were selected for prediction. A single prediction was done with time-window of 150 records. The first 140 records were learning records. The next 5 records constituted validation set. The last 5 records were the test samples (and were not used in the training phase). The stopping condition in genetic algorithm depended on the number of training epochs, a diversity of the population, and the number of training epochs without finding the new best chromosome. The stopping condition in neural networks' learning depended on either the number of iterations or the increasing of prediction error on validation set.

[1] Unfortunately the volume values were not available.

3 System architecture

Generally speaking the proposed idea of building the short-term prediction system is to **initially provide a large number of different variables and then select the locally suboptimal set of variables as the input data for a small ensemble of simple neural networks**.

Hence, after pre-processing of raw data described in the previous section the autocorrelation matrix R of input vectors was calculated:

$$R = \frac{1}{n} \sum_{i=0}^{n} x_i x_i^T \tag{1}$$

where n is the number of all available input vectors x_i. The higher value in the matrix, the higher correlation of trends between corresponding variables in a vector. Using this matrix it is possible to find variables correlated with predicted variable. Based on this method $10\% - 20\%$ of all variables (the ones with correlation no less than 50% of maximum and 200% of average correlation) were initially selected for further steps. Furthermore, all variables from the target stock market were also added to the initial selection. These pre-selected input variables were available for genetic algorithm. Chromosomes coded the list of input variables of the final neural networks. The number of variables coded by a single chromosome was between 4 and 7. The number of hidden layers was set to 1 for all neural architectures.

In the genetic algorithm the first variable coded by each chromosome was forced to be the last available value of predicted variable (i.e. change of DAX closing value of the previous day). This data obviously seems to be crucial for prediction and therefore we've decided that it would always appear in the initial set of neural networks' input variables.

The crossover depends on the common parts of two parent chromosomes and random selection of the rest of variables. Two children are created as a result. During mutation variables coded by one chromosome are exchanged with randomly selected variables of all available ones. In each step of the algorithm the mutation can affect only one randomly selected variable, with some mutation probability (the detailed description follows). Selection of chromosomes for crossover was done by the rank method. Parents were exchanged with children only if the latter were better fitted. The fitness was calculated based on the results of training of the neural network coded by a chromosome. The smaller error on validation samples, the higher fitness of a chromosome. The fitness was calculated using the average prediction error of 3 neural networks (corresponding to the coded architecture) with random initial weights.

It is important to note, that in the initial set of randomly generated chromosomes almost all coded neural networks were unable to learn efficiently. Because of the lack of coherence between input variables the error value did not decreased during learning process. Such chromosomes were called *not-alive*. Chromosomes which coded networks with ability to learn efficiently were called *alive*. For both

types of chromosomes the probability of crossover was equal to 1. The probability of mutation depended on a current situation in a population. If there were any *not-alive* chromosomes the mutation affected only them with probability 1. If more than 90% of chromosomes were *alive* the mutation was done with probability 0.05 for *alive* chromosomes. The reason of such behavior was a necessity to preserve alive chromosomes.

After 300 iterations of genetic algorithm the best chromosome from all iterations was selected for the last step.

After the last iteration of genetic algorithm the specific *brute force* method was applied. Every variable coded by the best chromosome was sequentially (one at a time) changed by mutation and the fitness of a modified chromosome was calculated. Note, that also the first variable which was not allowed to change in the genetic algorithm could be impacted by this procedure.

Three instances of the network architecture coded by the best chromosome were then trained, each with randomly selected initial weights. Training in this step and during the genetic algorithm was performed with backpropagation with momentum. The input data was normalized and scaled to the range $(-1, 1)$. In all neurons sigmoidal activation function was used. Initially, all weights in the network were randomly chosen from the range $(-0.01, 0.01)$.

4 Experiment set-up

Ten experiments were performed in order to present usefulness of the above approach for prediction task in a short period of time. In each experiment learning and validation were based on 140 and 5 days (samples), resp. The test data was the following 5 days. Large part of the data was shared between experiments since in subsequent ones the time-window was shifted by 5 days forward (test samples from immediately previous experiment became validation ones and validation samples became the last part of the training days). Each experiment was repeated 10 times on disjunctive 5-day test records, as described above. Hence, the model was tested on the period of 50 days in total.

In the following discussion of results variables denoted by *value O, value C, value H* and *value L* denote open, close, highest and lowest values of a given day, resp.; *change O (%), change C (%)* - change of resp. open, close values in per cent within one day; *mov avg n* - moving average for n days; *n days change (%)* - change in per cent of open value in the period on n days; *DAY NO* - day of the week one day before prediction. The rest of the indicators (e.g. *pattern, MACD signal line, WILLIAMS buy/sell signal, ROC n, RSI n*, etc. are self-explained and denote popular technical analysis oscillators and signals [10].

5 Results

The goal of each day's prediction was the percentage change of closing value of DAX index. In each experiment, first an average error of prediction and corresponding average change of index value (volatility) are presented. More precisely

$$avg.err. = \frac{1}{10} \sum_{i=1}^{i=10} \left| (volatility^i_{correct} - volatility^i_{prediction}) \right|$$

where *volatility* means percentage change of an index value of the i-th sample. While the prediction task was a percentage change of an index value, we can compute an error as a difference between real and predicted change. Suppose that the prediction is always "no change". In that case the error would be the same as the volatility of an index value.

The errors are followed by description of variables - each description is of the form *[stock_market; variable_name; connection]*, where *stock_market* can be either *Target (i.e DAX), DJIA, KOSPI, USD/JPY* or *EUR/USD* and each of the above can be calculated for any of the $t-n$ last days $n = 0, ..., 5$. *variable_name* describes the actual indicator (being either an oscillator, or signal, or numerical value, ...). Finally *connection* provides the information about the occurrence of *variable_name* in other experiments - this way some dependencies between variables in different periods are presented (in case of no such dependencies the sign "X" is placed in this field). Moreover the phrase *(sim.)* following the *exp. number* means that the meaning of the selected variables is almost the same (e.g. the average value of the past 20 days is almost the same for today and yesterday). Finally, the number after the *variable_name* (e.g. ROC 5) indicates oscillator's parameter. Furthermore information about source of chromosome is provided: 'GA' if it came directly from genetic algorithm, 'BF' if it was improved by the brute force method mentioned in section 3. It's interesting to note that 5 out of 10 best chromosomes were not improved by brute force algorithm, which means that they were 'locally optimal'.

exp. 1**; avg.err.** $0,00820$**; avg. volatility** $0,00828$**; source: GA**
[target dax [t]; close change (%); exp.1,3,4,6,7,8,9,10]
[target dax [t]; value o; sim.exp.2, sim.exp.9]
[eur/usd [t]; impet 20; x]
[target dax [t-5]; roc 5; sim.exp.2]
[eur/usd [t]; value l; x]
[eur/usd [t]; close change (%); x]

exp. 2**; avg.err.** $0,00574$**, avg. volatility** $0,00483$**; source: BF**
[eur/usd [t]; so; x]
[target dax [t-5]; close change (%); x]
[djia [t]; rsi 5; exp.9]
[target dax [t]; so; exp.5, sim.exp.6, exp.10]
[target dax [t-1]; roc 5; sim.exp.1, sim.exp.7]
[djia [t]; close change (%); exp.4, exp.8, exp.9]
[target dax [t]; value h; sim.exp.1, sim.exp.9]

exp. 3**; avg.err.** 0, 00532**, avg. volatility** 0, 00541**; source: GA**
[target dax [t]; close change (%); exp.1,3,4,6,7,8,9,10]
[target dax [t]; impet 5; exp.5, sim.exp.10]
[djia [t]; 5 days change (%); exp.4]
[target dax [t-1]; close change (%); x]
[target dax [t]; 2 avg's buy/sell signal; exp.7]
[usd/jpy [t]; 20 days change (%); x]

exp. 4**; avg.err.** 0, 00501**, avg. volatility** 0, 00555**; source: GA**
[target dax [t]; close change (%); exp.1,3,4,6,7,8,9,10]
[target dax [t-1]; 20 days change (%); exp.5, sim.exp.10]
[djia [t]; close change (%); exp.2, exp.8, exp.9]
[usd/jpy [t]; williams; x]
[djia [t]; 5 days change (%); exp.3]
[target dax [t]; change o(%); x]

exp. 5**; avg.err.** 0, 00875**, avg. volatility** 0, 00912**; source: BF**
[target dax [t-1]; 20 days change (%); exp.4]
[kospi [t]; so; x] - [djia [t]; impet 10; sim.exp.10]
[target dax [t]; impet 20; exp.10] - [target dax [t]; impet 5; exp.3]
[target dax [t]; so; exp.2, exp.10] - [target dax [t-5]; macd; x]

exp. 6**; avg.err.** 0, 00922**, avg. volatility** 0, 01052**; source: GA**
[target dax [t]; close change (%); exp.1,3,4,6,7,8,9,10]
[djia [t]; so; x] - [kospi [t]; impet 5; x]
[target dax [t]; roc 10; sim.exp.2, sim.exp.7]
[target dax [t-2]; rsi 5; x] - [target dax [t]; fso; sim.exp.2, sim.exp.10]
[target dax [t]; mov avg 10; sim.exp.8]

exp. 7**; avg.err.** 0, 00578**, avg. volatility** 0, 00579**; source: GA**
[target dax [t]; close change (%); exp.1,3,4,6,7,8,9,10]
[target dax [t]; roc 5; sim.exp.2, sim.exp.7]
[target dax [t]; 5 days change; x]
[target dax [t]; 2 avg's buy/sell signal; exp.3]
[target dax [t]; macd signal line; exp.9, sim.exp.10]
[usd/jpy; williams buy/sell signal; x]

exp. 8**; avg.err.** 0, 01151**, avg. volatility** 0, 01231**; source:BF**
[target dax [t]; close change (%); exp.1,3,4,6,7,8,9,10]
[target dax [t-3]; so; sim.exp.10]
[target dax [t]; type of pattern; sim.exp.9]
[target dax [t-4]; value o; x]
[usd/jpy [t]; macd; x] - [djta [t]; close change (%); exp.2, exp.4, exp.9]
[target dax [t]; mov avg 20; sim.exp.6]

exp. 9**; avg.err.** $0,00488$**, avg. volatility** $0,00513$**; source:BF**
[target dax [t]; close change (%); exp.1,3,4,6,7,8,9,10]
[target dax [t]; rsi 5; exp.2]
[djta [t]; williams; x]
[target dax [t]; pattern; sim.exp.8]
[target dax [t]; value l; sim.exp.1, sim.exp.2]
[target dax [t]; macd signal line; exp.7, sim.exp.10]
[djta [t]; close change (%); exp.2, exp.4, exp.8]

exp. 10**; avg.err.** $0,00642591$**, avg. volatility** $0,00706417$**; source:BF**
[target dax [t]; close change (%); exp.1,3,4,6,7,8,9,10]
[target dax [t]; so; exp.2, exp.5, sim.exp.6, sim.exp.8]
[target dax [t-2]; 20 days change (%); sim.exp.4]
[djta [t]; impet 5; sim.exp.5]
[target dax [t-3]; Williams buy/sell signal; x]
[target dax [t-1]; macd signal line; sim.exp.9, sim.exp.7, sim.exp.9]
[target dax [t]; impet 20; sim.exp.3, exp.5]

6 Conclusions and directions for future research

In 9 out of 10 experiments the average error is smaller than volatility of test samples. In majority of (independent) tests several similarities between the choices of input variables in subsequent experiments are discovered. For example the Stochastic Oscillator or its extended version (FSO) for the target stock market DAX repeats as an input in experiments $2, 5, 6, 8, 10$. Also inputs from stock market DJIA repeat in several experiments. Close change (%) from DJIA is chosen in experiments $2, 4, 8$ and 9.

This implies that the evolutionary-based input variable selection and the choice of network's architecture are reasonable. Since the exactness of any financial indicator (oscillator, signal, etc.) varies in time, having some indicators shared between subsequent experiments is an expected and promising feature. Certainly, in each experiment some number of new variables is also expected to appear and replace the "used" and "not adequate" ones.

Since the mechanism of selecting input variables works quite tempting, the main future goal is to improve the exactness of the fitness function in genetic algorithm and the learning process of the final neural network, which should result in further decreasing of the error value.

Since many variables available for genetic algorithm are very similar to each other in their meaning and numerical properties another direction for future research is to introduce some kind of penalty for the algorithm for selecting similar variables.

References

1. Lajbcygier, P.: Improving option pricing with the product constrained hybrid neural network. IEEE Transactions on Neural Networks **15** (2004) 465–476
2. Cao, L., Tay, F.: Support vector machine with adaptive parameters in financial time series forecasting. IEEE Transactions on Neural Networks **14** (2003) 1506–1518
3. Gestel, T., Suykens, J., et al.: Financial time series prediction using least squares support vector machnies within the evidence framework. IEEE Transactions on Neural Networks **12** (2001) 809–820
4. Podding, T., Rehkegler, H.: A "world" model of integrated financial markets using artificial neural networks. Neurocomputing **10** (1996) 251–273
5. Kodogiannis, V., Lolis, A.: Forecasting financial time series using neural network and fuzzy system-based techniques. Neural Computing & Applications **11** (2002) 90–102
6. Tino, P., Schittenkopf, C., et al.: Financial volatility trading using recurrent neural networks. IEEE Transactions on Neural Networks **12** (2001) 865–874
7. Dempster, M., Payne, T., Romahi, Y., Thompson, G.: Computational learning techniques for intraday fx trading using popular technical indicators. IEEE Transactions on Neural Networks **12** (2001) 744–754
8. Mantegna, R., Stanley, E.: An Introduction to Econophysics. Correlations and Complexity in Finance. Cambridge University Press (2000)
9. Jaruszewicz, M., Mańdziuk, J.: One day prediction of nikkei index considering information from other stock markets. L. Rutkowski et al. (Ed.), ICAISC, Lect. Notes in Art. Int. **3070** (2004) 1130–1135
10. Murphy, J.: Technical Analysis of the Financial Markets. New York Instituite of Finance (1999)