Forecasting Stock Market Trend using Prototype Generation Classifiers

PETR HÁJEK
Institute of System Engineering and Informatics
Faculty of Economics and Administration
University of Pardubice
Studentská 84, 532 10 Pardubice
CZECH REPUBLIC
petr.hajek@upce.cz

Abstract: - Currently, stock price forecasting is carried out using either time series prediction methods or trend classifiers. The trend classifiers are designed to predict the behaviour of stock price's movement. Recently, soft computing methods, like support vector machines, have shown promising results in the realization of this particular problem. In this paper, we apply several prototype generation classifiers to predict the trend of the NASDAQ Composite index. We demonstrate that prototype generation classifiers outperform support vector machines and neural networks considering the hit ratio of correctly predicted trend directions.

Key-Words: - Stock price, stock market index, forecasting, prediction, learning vector quantization, prototype generation, support vector machines, neural networks, particle swarm optimization.

1 Introduction

Most of the studies dealing with stock price forecasting have been focused on the accurate prediction of the value of the underlying stock / stock market index and low attention is paid to the prediction of the trend (direction) of stock market index movement [1].

As shown in preceding literature, this problem has to be perceived as complex since stock market index time series are highly non-linear with a changing volatility and many micro- and macroeconomic determinants. Therefore, several soft computing methods have been applied to address these issues. Especially, support vector machines (SVMs) [2] and related hybrid models have shown promising results due to its good generalization performance. Using soft computing methods such as SVMs and neural networks (NNs), it is possible to model the noise and non-linearity in the stock price time series.

In this paper we will demonstrate that the behaviour of stock price's movement can be effectively predicted using prototype generation classifiers. These methods are based on building new artificial prototypes from the training data set. In this way the performance of nearest neighbour based classification is improved. We hypothesise that the prediction of the stock market index trend can be derived from a set of prototypes generated from training patterns. We will show that these methods are significantly more accurate in the prediction of the NASDAQ Composite index movement compared to other soft computing and machine learning approaches such as NNs or SVMs.

The remainder of the article is organized as follows. First, we present a brief review of related literature. Then, input variables based on technical analysis will be designed for the prediction model. Further, basic notions of prototype generation methods will be introduced. In the experimental results' section, the performance of these methods is compared to selected models of NNs and SVMs.

2 Literature Review

The prevailing literature has dealt with the prediction of the value of the underlying stock so far. It has been proven that non-linear predictors from the fields of soft computing and machine learning are more accurate in forecasting stock prices, see e.g. [3]. For example, the capability of predicting power of multi-layer perceptron NN (MLP) compared to multivariate statistical methods was examined by Yoon and Swales [4]. In a similar manner, MLPs have been employed to predict Malaysian [5], Japanese [6] or American indexes [7]. The other architectures of NNs applied to stock price forecasting include generalized regression NNs [8], radial basis function (RBF) NNs [9, 10] and related SVMs [11].

The advantages of individual soft computing methods have further been examined in hybrid systems. Tsaih et al. [12] combined NNs with rule-based systems, Armano et al. [13] merged the advantages of NNs and genetic

algorithms, and Kuo et al. [14] employed genetic algorithm based fuzzy NN to formulate the knowledge base of fuzzy inference rules for Taiwan stock market. For similar reasons, traditional methods for time series analysis like ARIMA were combined with soft computing methods [15].

The other approach to stock market forecasting is represented by the prediction of the trend (direction) of stock market index movement. Contrary to the above mentioned studies this approach is a classification task where the target classes stand for positive / negative trend or buy / sell (buy / hold / sell) decision. The movement of the stock market index is calculated as $(P_{t+d} - P_t)/P_t$) for a d-day forecast.

The classification problem requires different methods to be used compared to prediction (regression) one. Chang et al. [1] employed fuzzy decision trees to predict the trend in S&P 500 index. They found that this approach is significantly more accurate (with the hit ratio of 91.95%) compared to random walk (51.06%), ARIMA (56.13%) and MLP (69.78%). Moreover, it is reported that previous studies using SVMs [2], [16], and [17] were outperformed as well. Concretely, Kim [2] achieved the hit ratio of 78.65%, Yu et al. [17] 82.66% and Yu et al. [16] 84.57%. In a similar manner, evolving fuzzy decision trees were used to predict the trend of several stocks in Taiwan Stock Exchange Corporation [18].

Leung et al. [19] compared the performance of probabilistic NN (PRNN) against traditional statistical classifiers on several stock market indexes. PRNN outperformed discriminant analysis, Probit and Logit in the case of S&P 500 (63%) and FTSE 100 (61%) while discriminant analysis was more accurate for Nikkei 225 stock market index (68%).

By combining several methods including SVMs and MLPs, Huang et al. [20] achieved a higher accuracy of 75% compared to individual classifiers (random walk 50%, linear discriminant analysis 55%, quadratic discriminant analysis 69%, MLP 69% and SVM 73%).

3 Data Set

The original time series used in this study is depicted in Fig. 1. It represents the closing values of the NASDAQ Composite Index (IXIC) from 1.1.2007 to 31.12.2010, i.e. 4 years and 992 datum points. The stock market index IXIC experienced both slight and substantial increased and decreases during this time period.

The indicators of technical analysis represent commonly used predictors of stock market movement. They are based on previous patterns drawn from stock price time series. In prior literature it has been shown that suitable combinations of technical indicators can relatively accurately detect the turns in stock price movement [21].

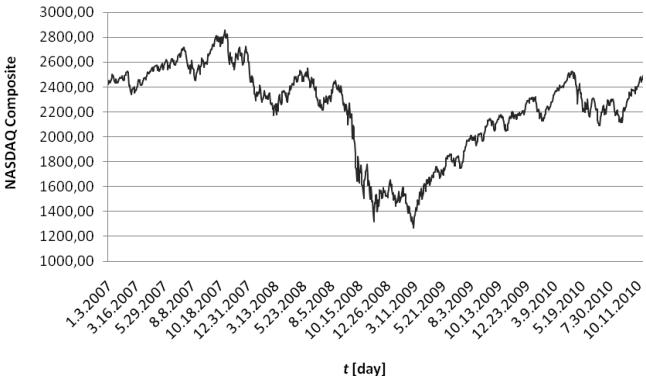


Fig. 1 NASDAQ Composite Index for the period 2007-2010

Therefore, we develop a set of technical indicators to predict the sign of the stock market index movement. We use the following technical indicators:

- Simple moving average (SMA);
- Weighted moving average (WMA);
- Momentum (MOM);
- Standard deviation (STD).

They are defined in this way:

$$SMA_{t,n} = \frac{1}{n}\sum_{t=1}^{n} P_{t,n}\,, \tag{1}$$

where n is the number of the datum points (period) and $P_{t,n}$ is the closing price in the t-th day.

$$WMA_{t,5} = \frac{1}{35}(-3P_{t-2} + 12P_{t-1} + 17P_t + 12P_{t+1} - 3P_{t+2})\,. \tag{2}$$

Moving averages emphasize the direction of a trend by smoothing out (eliminating) the price and volume fluctuations. The WMA indicator represents a convolution of values using the chosen weighting function. Here a central weighting function is used for n=5, where the weights decrease symmetrically in both directions from the central datum point.

$$MOM_{t,n} = (P_t - P_{t-n}) \times 100\,. \tag{3}$$

The indicator MOM is developed to monitor and indentify cyclical fluctuations and short-term trends in the movement of stock prices. It is used to estimate the change the direction of the movement.

$$STD_{t,n} = \sqrt{\frac{\sum_{t=1}^{n} (P_t - SMA_{t,n})^2}{n}}. \quad (4)$$

The indicator STD measures the difference between the closing price and SMA. Thus, the stock price nature of returning back to average price is analyzed. The technical indicators used in this study are presented in Fig. 2 – Fig. 5. The length of the period was set to n=5 due to using working days' values and to address related issues (the Monday Effect).

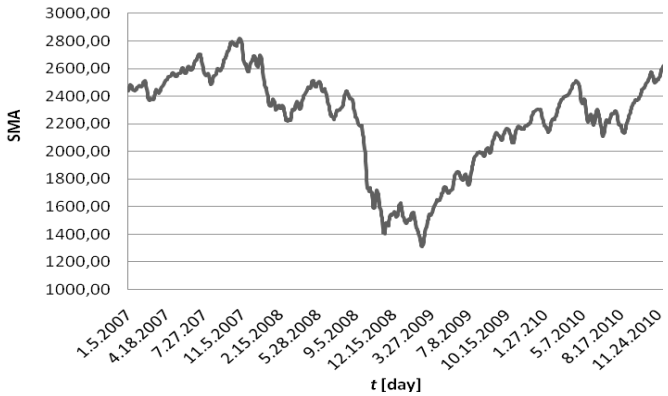


Fig. 2 Simple moving average for n=5

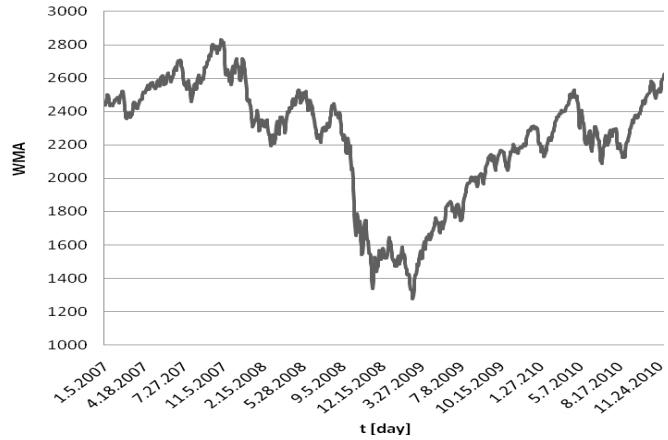


Fig. 3 Weighted moving average for n=5

Thus, the forecasting model has the following form:

$$y_{t+5} = f(P_t, SMA_{t,5}, WMA_{t,5}, MOM_{t,5}, STD_{t,5}), \quad (5)$$

where

$$y_{t+5} = (P_{t+5} - P_t) / P_t \quad (6)$$

and the target classes are defined as c_1 if $y_{t+5} > 0$ (the positive sign of the trend) and c_2 if $y_{t+5} \leq 0$ (the negative sign of the trend).

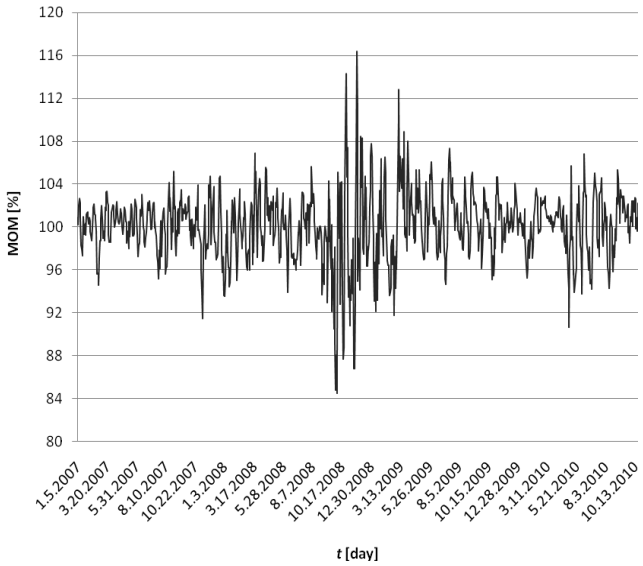


Fig. 4 Momentum for n=5

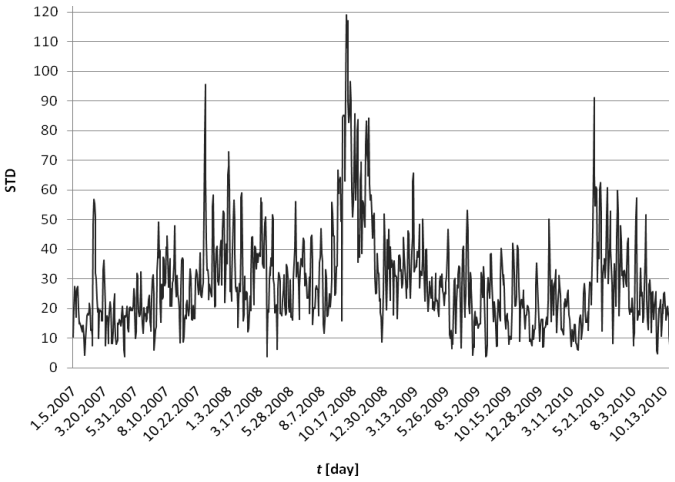


Fig. 5 Standard deviation for n=5

The aim of the developed model is to achieve the highest hit ratio (classification accuracy). Out of the total 992 data samples, 559 were with a positive sign (class c_1) and 433 with a negative sign (class c_2). The output classes for the underlying time series is presented in Appendix A.

3 Prototype Generation Methods

The prototype generation models used in this study are based on nearest neighbour algorithm and its derivatives, for their overview see [22]. They are represented by the set of methods that predicts the target class from training data only and does not develop learning models. A wide range of problems have been modelled using the nearest neighbour based methods in the field of finance [23-26] and other research fields [27, 28].

In the 1-nearest neighbour classification, the most similar (with the lowest distance) neighbour is used to classify data samples. In the *k*-nearest neighbours (*k*-NN), the predicted class is set to the most frequent true class among the *k* nearest training samples.

To mention the drawbacks of the *k*-NN classifiers, they often suffer from high computational cost and low tolerance to noise.

A successful approach to tackle the mentioned drawbacks is based on data reduction [29]. These techniques aim to obtain a representative training set with a lower size compared to the original one and with a similar or even higher classification accuracy [22]. This reduction can be realized using either prototype selection or prototype generation methods. In contrast to prototype selection methods, prototype generation methods aim at creating new artificial data which allows filling regions in the domain of the problem, which have no representative examples in original data.

The overview of the prototype generation methods is presented in Appendix B.

This taxonomy is based on several criteria. The first is type reduction. We can start with an empty set and add new prototypes (incremental) or start with an original set and reduce (modify) it (decremental). A mixed approach combines the two methods. Finally, fixed reduction is based on setting the final number of prototypes (usually as percentage retention of original data set).

The resulting generation set may retain border (condensation), central (edition) or both types of points (hybrid). The generation mechanism may also be based on several approaches. Class relabeling approach aims to change the class labels of suspicious samples from the original data set. Centroid based technique merges similar samples into new artificial prototypes. Space splitting uses heuristics to partition the feature space. Finally, positioning adjustment aims to correct the position of a subset of prototypes. The search criterion depends on the use or non-use of *k*-NN in the evaluation process. In filter approach, the *k*-NN is not used at all, while in wrapper approach the *k*-NN fully guides the search process.

The list of the applied prototype generation methods is presented in Table 1.

4 Experimental Results

For the experiments, data were divided into 10 parts of the same size and then trained with 9 parts and tested with 1 remaining part. This procedure was repeated 10 times for all parts. In the results, we refer to mean classification accuracy (hit ratio) and standard deviation of this estimation.

In Table 2, we present the results for the methods using positioning adjustment generation mechanism. In Table 3, there are resulting hit ratios and standard deviations for the remaining prototype generation methods. For the fixed reduction technique we tested percentage retention *pr* [%] for $pr=\{10,40,70,100\}$.

Table 1: The list of the used prototype generation methods with references

Name	Abbr.	Ref.
Learning vector quantization 1	LVQ1	[30]
Learning vector quantization 2	LVQ2	[30]
Learning vector quantization 3	LVQ3	[30]
Decision support mapping	DSM	[31]
Vector quantization	VQ	[32]
Gradient descent and deterministic annealing	MSE	[33]
Evolutionary nearest prototype	ENPC	[34]
Prototype selection – clonal selection algorithm	PSCSA	[35]
Particle swarm optimization	PSO	[36]
Adaptive Michigan PSO	AMPSO	[37]
Adaptive vector quantization	AVQ	[38]
Modified Chang's algorithm	MCA	[39]
Chen algorithm	Chen	[40]
Bootstrap technique for nearest neighbour	BTS3	[41]
Generalized editing using nearest neighbour	GENN	[42]
Generalized modified Chang's algorithm	GMCA	[43]
Integrated concept prototype learner 2	ICPL2	[44]
Adaptive condensing algorithm based on mixtures of Gaussians	MixtGauss	[45]
Prototype nearest neighbour	PNN	[46]
Pairwise opposite class nearest neighbour	POC	[47]
Reduction by space partitioning 3	RSP3	[48]
Self-generating prototypes	SGP	[49]

The methods using positioning adjustment were trained with the following setting of learning parameters. LVQ1: # iterations = 100, learning rate = 0.1; LVQ2: # iterations = 100, learning rate = 0.1, window width = 0.2; LVQ3: # iterations = 100, learning rate = 0.1, window width = 0.2, stabilizing factor = 0.1; DSM: # iterations = 100, learning rate = 0.1; VQ: # iterations = 100, learning rate = 0.1, size of neighbourhood for the *k*-NN $k = 1$; MSE: $k = 3$, # initial centroids = 10, gradient step = 0.5, initial temperature = 100; ENPC: $k = 3$, maximum # iterations = 250; PSCSA: hypermutation rate = 2, clonal rate = 10, mutation rate = 0.01, stimulation threshold = 0.89, learning rate = 0.4; PSO: $k = 1$, swarm size = 20, particle size = 5%, maximum # iterations = 250, acceleration constant C1 = 1,

acceleration constant C2 = 3, maximum velocity of particles = 0.25, starting inertia weight = 1.5, ending inertia weight = 0.5; AMPSO: k = 1, swarm size = 10, maximum # iterations = 300, acceleration constant C1 = 1, acceleration constant C2 = 1, acceleration constant C3 = 0.5, maximum velocity of particles = 1, inertia weight = 0.1, constriction factor = 0.1, probability of reproduction = 0.1, probability of deletion = 0.0; AVQ: # iterations = 100, epsilon in the LBG algorithm = 0.1.

Table 2: Hit ratios for positioning adjustment generation methods

Method	Hit ratio ± Std. Dev.
LVQ1, *pr*=10%	60.66 ± 2.00
LVQ1, *pr*=40%	65.29 ± 1.37
LVQ1, *pr*=70%	66.23 ± 0.86
LVQ1, *pr*=100%	68.14 ± 1.10
LVQ2, *pr*=10%	63.22 ± 2.45
LVQ2, *pr*=40%	62.90 ± 1.85
LVQ2, *pr*=70%	63.34 ± 1.32
LVQ2, *pr*=100%	63.64 ± 0.93
LVQ3, *pr*=10%	62.13 ± 2.01
LVQ3, *pr*=40%	66.30 ± 1.77
LVQ3, *pr*=70%	67.78 ± 1.00
LVQ3, *pr*=100%	69.04 ± 1.07
DSM, *pr*=10%	63.81 ± 2.42
DSM, *pr*=40%	66.79 ± 0.66
DSM, *pr*=70%	69.06 ± 1.03
DSM, *pr*=100%	69.69 ± 1.00
VQ, *pr*=10%	61.02 ± 1.04
VQ, *pr*=40%	62.01 ± 1.27
VQ, *pr*=70%	63.19 ± 0.94
VQ, *pr*=100%	64.45 ± 0.99
MSE	64.82 ± 1.42
ENPC	**79.96 ± 1.07**
PSCSA	66.07 ± 1.84
PSO	72.33 ± 1.04
AMPSO	63.25 ± 1.47
AVQ, *pr*=10%	55.77 ± 3.48
AVQ, *pr*=40%	56.32 ± 6.49
AVQ, *pr*=70%	56.48 ± 6.05
AVQ, *pr*=100%	57.58 ± 6.53

The prototype generation with positioning adjustment did not show good results compared to previous studies with regard to average hit ratio. This measure ranges from 55.77% (AVQ, *pr*=10%) to 79.96 (ENPC).

The ENPC method outperformed the remaining positioning adjustment methods significantly (paired *t*-test at p=0.05). This method uses five different operators of genetic algorithms to adjust the positions of the generated prototypes. These operators define regions in the search space. Then, in a mutation phase prototypes are re-labelled with the most populate class in each region of the search space. In order to refine the set of generated prototypes a reproduction operator is introduced to generate new prototypes. The move operator relocates each prototype in the best expected place, i.e. it is moved to the centroid of its region. Finally, a die operator is used to remove prototypes that are not relevant. In this process the *k*-NN algorithm is used to classify the data samples.

Considering the percentage retention *pr* [%], it appears that highly complex models (with *pr*=70% or *pr*=100%) have to be constructed to predict the stock market index movement accurately. This implies that there are no sparse patterns presented in the data by which stock market index behaviour could be modelled effectively. Further, evolutionary based prototype generation methods such as ENPC and PSO outperformed the remaining positioning adjustment methods.

Table 3: Hit ratios for the rest of prototype generation methods

Method	Hit ratio ± Std. Dev.
MCA	**92.05 ± 0.52**
Chen, *pr*=10%	69.02 ± 0.75
Chen, *pr*=40%	74.44 ± 0.99
Chen, *pr*=70%	66.67 ± 1.31
Chen, *pr*=100%	64.42 ± 0.86
BTS3, *pr*=10%	62.50 ± 1.38
BTS3, *pr*=40%	64.08 ± 1.03
BTS3, *pr*=70%	65.85 ± 0.72
BTS3, *pr*=100%	65.24 ± 0.75
GENN	73.44 ± 0.93
GMCA	81.03 ± 1.14
ICPL2	71.07 ± 0.75
MixtGauss	60.29 ± 0.44
PNN	62.33 ± 1.59
POC	50.04 ± 1.48
RSP3	76.95 ± 0.77
SGP	52.86 ± 3.22

The methods using other generation mechanisms were trained with the following setting of learning parameters. MCA: no parameters; Chen: no parameters; BTS3: k = 1, random trials = 3; GENN: k = 2; GMCA: no parameters; ICPL2: # algorithms = 4, k = 3, filtering method = ENN, threshold for ACC filtering = 1;

MixtGauss: particle size = 5%; PNN: no parameters; POC: replacing method, learning rate = 0.2; RSP3: diameter subsets choice; SGP: pruning model.

The best results were provided by the MCA method with the average hit ratio of 92.05%. Both modified Chang's algorithms MCA and GMCA performed better than the best positioning adjustment method ENPC. These algorithms are centroid based techniques and work decrementally. The PNN and MCA share the same idea, concretely initialize resulting set with original set, and merge prototypes while classification accuracy does not decrease. The MCA method uses a distance matrix by classes to merge prototypes of the same class.

Finally, we compared the results obtained using prototype generation methods with selected models of NNs (MLP, RBF and probabilistic neural network (PRNN)) and SVMs [50-52]. Since the results of NNs and SVMs are sensitive to the setting of learning parameters we automatically determined their values in the following way.

The MLP model was trained using conjugate gradient method with 10000 maximum iterations. Logistic activation functions were used in both hidden and output layer. The number of neurons (showing on the complexity of the MLP model) in one hidden layer was set from the set {2, 3, … , 20}.

The parameters of the RBF model was determined automatically using a genetic algorithm. The number of neurons in the hidden layer was chosen from the set {1, 2, … , 100}, the radius of RBF from <0.01, 400> and lambda from <0.001, 10>. The employed genetic algorithm worked with the population size of 200 and the maximum number of generations = 20.

In the PRNN, Gaussian kernel function was used with radius set for each variable individually from the set <0.0001, 10>.

The C-SVM algorithm was employed with linear, polynomial (degree = 3) and RBF kernel functions to predict the trend of the IXIC stock market index. The RBF kernel function proved to be the most appropriate for the prediction. Complexity parameter C was chosen from the interval <0.1, 50000> and gamma from <0.01, 20> using pattern search algorithm [53].

The results presented in Table 4 show that there is no significant difference in the performance across NNs and SVMs. The hit ratio of 71.27% for the MLP was achieved with only 4 neurons in the hidden layer. Thus, there is no need to develop highly complex MLPs for the prediction of stock market index movement. However, the performance of NNs and SVMs is significantly worse compared to the MCA and several other prototype generation methods.

Table 4: Comparison of prediction performance across NNs and SVMs

Method	Hit ratio ± Std. Dev.
MLP (# neurons = 4)	71.27 ± 0.93
RBF (# neurons = 36)	69.96 ± 1.71
PRNN	71.17 ± 0.81
SVM (C = 967, gamma = 0.14, # support vectors = 646)	70.87 ± 0.88

5 Conclusion

The article presents a forecasting model based on prototype generation classifiers. We examined the hit ratio of selected methods on the NASDAQ Composite stock market index.

Based on the obtained results we draw the following conclusions.

(1) Forecasting stock market index trend represents a complex and nonlinear problem (see e.g. [54]) where at least 70% of original data samples (or their modifications) has to be retained to achieve a high hit ratio.

(2) Decremental reduction of the set of prototypes is more effective than the incremental type of reduction. That implies that patterns necessary for stock market index movement prediction are present in the original data set. It is only necessary to merge the data samples with the same target class and, thus, generate new artificial prototypes representing these regions in the search space.

(3) Prototype generation classifiers were significantly more accurate compared to NNs and SVMs. The comparison with previous studies is difficult to assess since different time series were used (different time horizon, stock market index, inputs, etc.). However, our results are in line with [1], [16] and [17] since similar hit ratios were obtained for MLPs and SVMs. From this fact we may infer that the complexity of the stock market index trend forecasting is similar in an international scale. Moreover, using prototype generation methods seem to be a promising tool for this particular problem since they provided equal or better performance compared to the methods used in the mentioned prior studies such as random walk, SVMs, fuzzy decision trees or psychological line [1, 55].

The hit ratio (buy or sell) of the historical stock market index data could be detected with a high accuracy (92.05%) using the MCA method especially. Thus, this system might help investors to make better decision in trading stocks.

In the future, the proposed methods can be further investigated in several directions: more stock market indexes should be examined to verify the results; combinations (ensembles) of patterns obtained by individual prototype generation methods can be applied to get a more accurate forecasting system (see e.g. [56]); more diverse set of inputs can be examined taking into account also macroeconomic determinants of stock market indexes (see e.g. [57-59]); and, finally, the forecasting system may be combined with an agent based model to simulate the financial market more effectively [60, 61]. The two-class classification problem addressed in this paper can also be easily extended to a multi-class problem where buy / hold / sell may represent the target classes.

The experiments in this study were carried out in Keel 2.0 and DTREG in MS Windows 7 operation system.

References:

[1] P.Ch. Chang, Ch.Y. Fan, J.L. Lin, Trend Discovery in Financial Time series Data using a Case based Fuzzy Decision Tree, *Expert Systems with Applications*, Issue 5, Vol.38, 2011, pp.6070–6080.

[2] K.J. Kim, Financial Time Series Forecasting using Support Vector Machines, *Neurocomputing*, Vol.55, 2003, pp.307–319.

[3] G. Zhang, B.E. Patuwo, M.Y. Hu, Forecasting with Artificial Neural Networks: The State of the Art, *International Journal of Forecasting*, Vol. 14, 1998, pp.35–62.

[4] Y. Yoon, J. Swales, Prediction Stock Price Performance: A Neural Network Approach, *Proceedings of the 24th Annual Hawaii International Conference on System Science*, pp.156–162.

[5] J. Yao, H.L. Poh, Forecasting the KLSE Index using Neural Networks, *IEEE International Conference on Neural Networks*, Vol.2, 1995, pp.1012–1017.

[6] N. Baba, M. Kozaki, An Intelligent Forecasting System of Stock Price using Neural Networks, *International Joint Conference on Neural Networks*, Vol.1, 1992, pp.371–377.

[7] J.H. Choi, M.K. Lee, M.W. Rhee, Trading S&P 500 Stock Index Futures using a Neural Network, *Proceedings of the Annual International Conference on Artificial Intelligence Applications on Wall Street*, New York, 1995, pp.63–72.

[8] D. Enke, S. Thawornwong, The Use of Data Mining and Neural Networks for Forecasting Stock Market Returns, *Expert Systems with Applications*, Issue 4, Vol.29, 2005, pp.927–940.

[9] X.B. Yan, Z. Wang, S. Yu, Y. Li, Time Series Forecasting with RBF Neural Network, *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics*, Guangzhou, Vol.8, 2005 pp.4680–4683.

[10] K. Chen, H. Lin, T. Huang, The Prediction of Taiwan 10-year Government Bond Yield, *WSEAS Transactions on Systems*, Issue 9, Vol.8, 2009, pp.1051–1060.

[11] L.J. Cao , F.E.H. Tay, Support Vector Machine with Adaptive Parameters in Financial Time Series Forecasting, *IEEE Transactions on Neural Networks*, Issue 6, Vol.14, 2003, pp.1506–1518.

[12] R. Tsaih, Y. Hsu, C.C. Lai, Forecasting S&P 500 Stock Index Futures with a Hybrid AI System, *Decision Support Systems*, Issue 2, Vol.23, 1998, pp.161–174.

[13] G. Armano, M. Marchesi, A. Murru, A Hybrid Genetic-Neural Architecture for Stock Indexes Forecasting, *Information Science*, Issue 1, Vol.170, 2005, pp.3–33.

[14] R.J. Kuo, C.H. Chen, Y.C. Hwang, An Intelligent Stock Trading Decision Support System through Integration of Genetic Algorithm based Fuzzy Neural Network and Artificial Neural Network, *Fuzzy Sets and Systems*, Issue 1, Vol.118, 2001, pp.21–45.

[15] P.F. Pai, CH.S. Lin, A hybrid ARIMA and Support Vector Machines Model in Stock Price Forecasting, *Omega*, Issue 6, Vol.33, 2005, pp.497–505.

[16] L. Yu, S.Y. Wang, K.K. Lai, Mining Stock Market Tendency using GA based Support Vector Machines, *Lecture Notes in Computer Science*, Vol.3828, 2005, pp.336–345.

[17] L. Yu, H. Chen, S. Wang, K.K. Lai, Evolving Least Squares Support Vector Machines for Stock Market Trend Mining, *IEEE Transactions on Evolutionary Computation*, Issue 1, Vol.13, 2009, pp.87–102.

[18] R.K. Lai, Ch.Y. Fan, W.H. Huang, P.Ch. Chang, Evolving and Clustering Fuzzy Decision Tree for Financial Time Series Data Forecasting, *Expert Systems with Applications*, Vol.36, 2009, pp.3761–3773.

[19] M.T. Leung, H. Daouk, A.S. Chen, Forecasting Stock Indices: A Comparison of Classification and Level Estimation Models, *International Journal of Forecasting*, Vol.16, 2000, pp.173–190.

[20] W. Huang, Y. Nakamori, S. Wang, Forecasting Stock Market Movement Direction with Support Vector Machine, *Computers and Operations Research*, Issue 10, Vol.32, 2005, pp.2513–2522.

[21] S.W. Pruitt, R.E. White, The CRISMA Trading System: Who Says Technical Analysis Can't Beat the Market?, *Journal of Portfolio Management*, Issue 3, Vol.14, 1988, pp.55–58.

[22] I. Triguero, J. Derrac, S. Garcia, F. Herrera, A Taxonomy and Experimental Study on Prototype Generation for Nearest Neighbor Classification, *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, Issue 1, Vol.42, 2012, pp.86–100.

[23] N. Chen, B. Ribeiro, A. Vieira, J. Duarte, J.C. Neves, Extension of Learning Vector Quantization to Cost-sensitive Learning, *International Journal of Computer Theory and Engineering*, Issue 3, Vol.3, 2011, pp.352–359.

[24] P. Hajek, V. Olej, Municipal Creditworthiness Modelling by Kohonen´s Self-organizing Feature Maps and LVQ Neural Networks, *Proceedings of the 9th International Conference on Artificial Intelligence and Soft Computing*, Zakopane, Poland, 2008, pp.52–61.

[25] P. Hajek, Municipal Credit Rating Modelling by Neural Networks, *Decision Support Systems*, Issue 1, Vol. 51, 2011, pp.108–118.

[26] P. Hajek, Probabilistic Neural Networks for Credit Rating Modelling, *Proceedings of International Conference on Neural Computation*, Valencia, 2010, pp.289–294.

[27] M. Govindarajan, R. Chandrasekaran, Evaluation of K-Nearest Neighbor Classifier Performance for Direct Marketing, *Expert Systems with Applications*, Issue 1, Vol.37, 2009, pp.253–258.

[28] P. Hajek, V. Olej, Air Quality Modelling by Kohonen's Self-organizing Feature Maps and LVQ Neural Networks, *WSEAS Transactions on Environment and Development*, Issue 1, Vol.4, 2008, pp.45–55.

[29] D.R. Wilson, T.R. Martinez, Reduction Techniques for Instance based Learning Algorithms, *Machine Learning*, Issue 3, Vol.38, 2000, pp.257–286.

[30] T. Kohonen, *Self-Organizing Maps*, Springer Verlag, Berlin, Heidelberg, New York, 2001.

[31] S. Geva, J. Site, Adaptive Nearest Neighbor Pattern Classifier, *IEEE Transactions on Neural Networks*, Issue 2, Vol.2, 1991, pp.318–322.

[32] Q. Xie, C.A. Laszlo, R.K. Ward, Vector Quantization Technique for Nonparametric Classifier Design, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Issue 12, Vol.15, 1993, pp.1326–1330.

[33] C. Decaestecker, Finding Prototypes for Nearest Neighbour Classification by means of Gradient Descent and Deterministic Annealing, *Pattern Recognition*, Issue 2, Vol.30, 1997, pp.281–288.

[34] F. Fernandez, P. Isaci, Evolutionary Design of Nearest Prototype Classifiers, *Journal of Heuristics*, Issue 4, Vol.10, 2004, pp.431–454.

[35] U. Garain, Prototype Reduction using an Artificial Immune Model, *Pattern Analysis and Applications*, Issues 3-4, Vol.11, 2008, pp.353–363.

[36] L. Nanni, A. Lumini, Particle Swarm Optimization for Prototype Reduction. *Neurocomputing*, Issues 4-6, Vol.72, 2009, 1092–1097.

[37] A. Cervantes, I. Galván, P. Isasi, An Adaptive Michigan Approach PSO for Nearest Prototype Classification, *Lecture Notes in Computer Science*, Vol.4528, 2007, pp.287–296.

[38] C.W. Yen, C.N. Young, M.L. Nagurka, A Vector Quantization Method for Nearest Neighbor Classifier Design, *Pattern Recognition Letters*, Vol.25, 2004, pp.725–731.

[39] J.C. Bezdek, T.R. Reichherzer, G.S. Lim, Y. Attikiouzel, Multiple Prototype Classifier Design, *IEEE Transactions on Systems, Man and Cybernetics – Part C*, Issue 1, Vol.28, 1998, pp.67–79.

[40] C.H. Chen, A. Józwik, A Sample Set Condensation Algorithm for the Class Sensitive Artificial Neural Network, *Pattern Recognition Letters*, Vol.17, 1996, pp.819–823.

[41] Y. Hamamoto, S. Uchimura, S. Tomita, A Bootstrap Technique for Nearest Neighbor Classifier Design, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Issue 1, Vol.19, 1997, pp.73–79.

[42] J. Koplowitz, T.A. Brown, On the Relation of Performance to Editing in Nearest Neighbor Rules, *Pattern Recognition*, Vol.13, 1981, pp.251–255.

[43] R.A. Mollineda, F.J.Ferri, E. Vidal, A Merge-based Condensing Strategy for Multiple Prototype Classifiers, *IEEE Transactions on Systems, Man and Cybernetics – Part B*, Issue 5, Vol.32, 2002, pp.662–668.

[44] W. Lam, C.K. Keung, D. Liu, Discovering Useful Concept Prototypes for Classification based on Filtering and Abstraction, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Issue 8, Vol.14, 2002, pp.1075–1090.

[45] M. Lozano, J. M. Sotoca, J. S. Sanchez, F. Pla, E. Pekalska, R.P.W. Duin, Experimental Study on Prototype Optimisation Algorithms for Prototype-based Classification in Vector Spaces, *Pattern Recognition*, Issue 10, Vol.39, 2006, pp.1827–1838.

[46] Ch.L. Chang, Finding Prototypes for Nearest Neighbor Classifiers, *IEEE Transactions on*

Computers, Issue 11, Vol.23, 1974, pp.1179–1184.

[47] T. Raicharoen, C. Lursinsap, A Divide-and-Conquer Approach to the Pairwise Opposite Class-Nearest Neighbor (POC-NN) Algorithm, *Pattern Recoginiton Letters*, Vol.26, 2005, pp.1554–1567.

[48] J.S. Sanchez, High Training Set Size Reduction by Space Partitioning and Prototype Abstraction, *Pattern Recognition*, Issue 7, Vol.37, 2004, pp.1561–1564.

[49] H.A. Fayed, S.R. Hashem, A.F. Atiya, Self-Generating Prototypes for Pattern Classification, *Pattern Recognition*, Issue 5, Vol.40, 2007, pp.1498–1509.

[50] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, 1999.

[51] N. Cristianini, J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press, Cambridge, 2000.

[52] S.S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice-Hall, Upper Saddle River, 1999.

[53] P. Hajek, V. Olej, Credit Rating Modelling by Kernel-based Approaches with Supervised and Semi-Supervised Learning, *Neural Computing & Applications*, Issue 6, Vol.20, 2011, pp.761–773.

[54] Ch.K. Volos, I.M. Kyprianidis, S.G. Stavrinides, I.N. Stouboulos, L. Magafas, A.N. Anagnostopoulos, Nonlinear Financial Dynamics from an Engineer's Point of View, *Journal of Engineering Science and Technology Review*, Vol.4, 2011, pp.281–285.

[55] Y. Zuo, E. Kita, Up/Down Analysis of Stock Index by Using Bayesian Network, *Engineering Management Research*, Issue 2, Vol.1, 2012, pp.46–52.

[56] P. Hajek, V. Olej, Municipal Revenue Prediction by Ensembles of Neural Networks and Support Vector Machines, *WSEAS Transactions on Computers*, Issue 11, Vol.9, 2010, pp.1255–1264.

[57] H. Lin, K. Chen, Soft Computing Algorithms in Price of Taiwan Real Estates, *WSEAS Transactions on Systems*, Issue 10, Vol.10, 2011, pp.342–351.

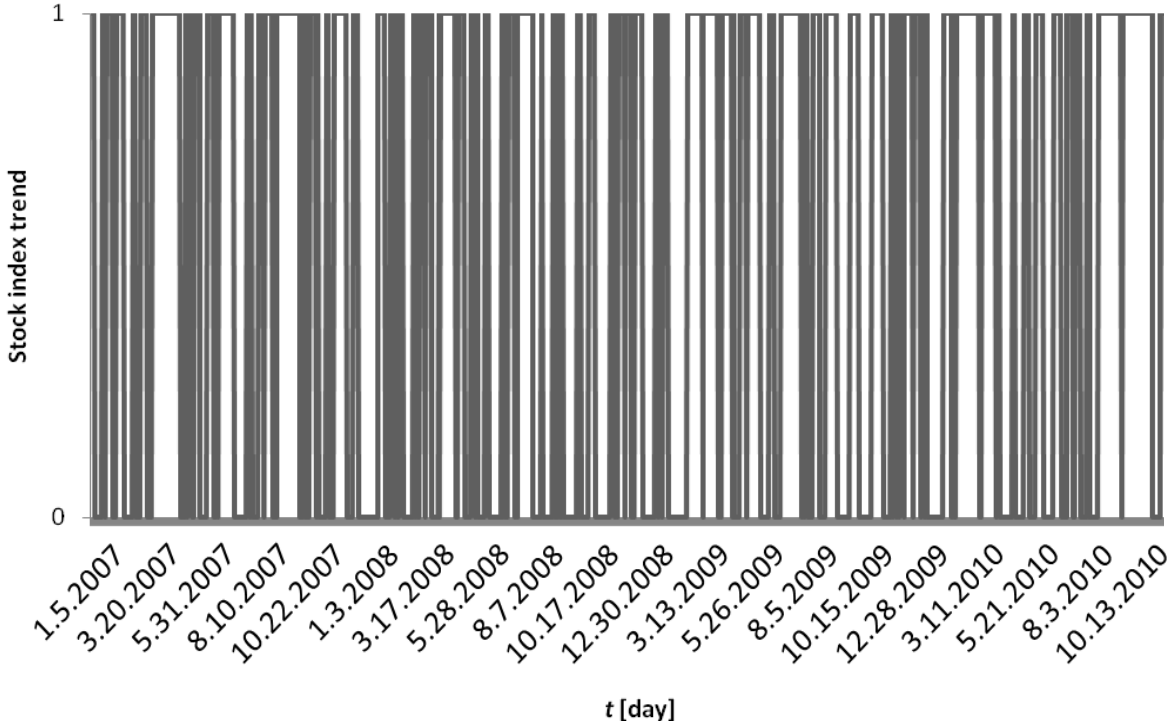
[58] V. Olej, Prediction of the Index Fund by Takagi-Sugeno Fuzzy Inference Systems and Feed-Forward Neural Network, *Proceedings of the 5th WSEAS International Conference on Artificial Intelligence*, *Knowledge Engineering and Data Bases*, Madrid, Spain, 2006, pp.7–12.

[59] K. Michalak, P. Lipinski, Prediction of High Increases in Stock Prices using Neural Networks, *Neural Network World*, Vol.15, 2005, pp.359–366.

[60] F. Neri, Learning Predictive Models for Financial Time Series by Using Agent Based Simulations, *Lecture Notes in Computer Science*, Vol.7190, 2012, pp.202–221.

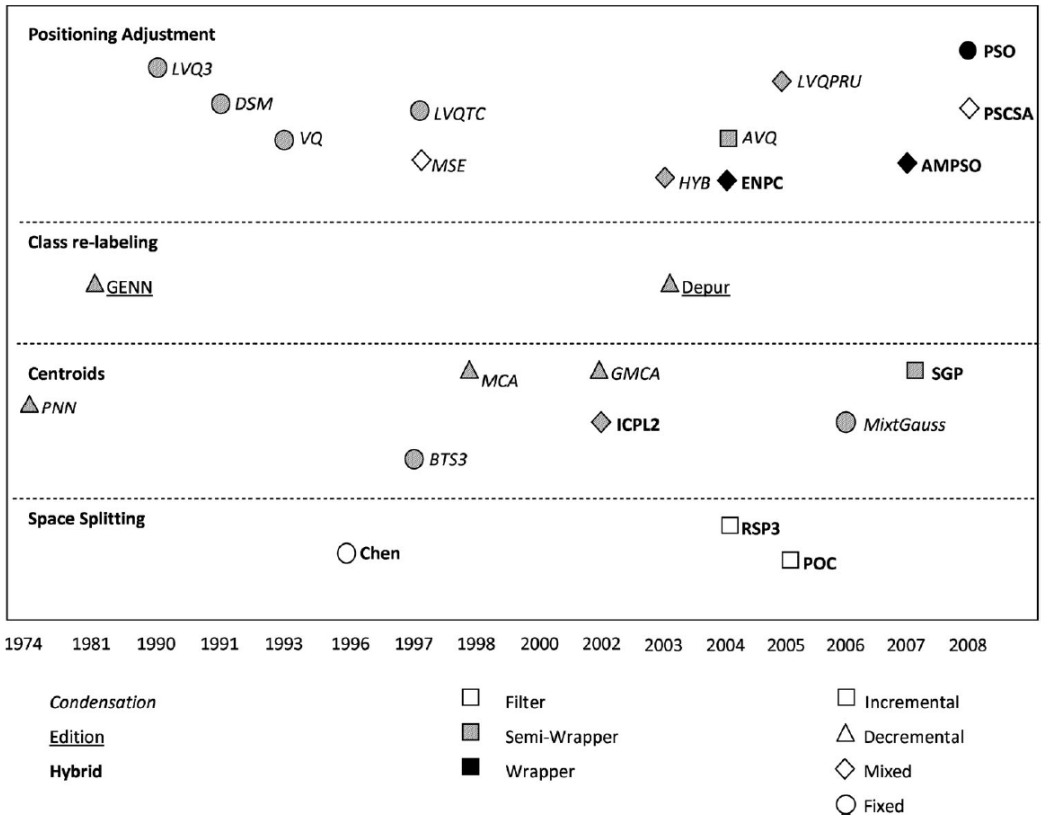
[61] F. Neri, Agent based Modeling under Partial and Full Knowledge Learning Settings to Simulate Financial Markets, *AI Communications*, Issue 4, Vol.25, 2012, pp.295-304.

Appendix A: Target classes of the forecasting model



Legend: stock index trend = 1 stands for a positive trend (class c_1) and stock index trend = 0 represents a negative trend (class c_2)

Appendix B: Overview of the used prototype generation classifiers



Source: [22]