Manuscript Number: APAD 01

Artificial Neural Network Models for Forecasting Stock Price Index in Bombay Stock Exchange

Neeraj Mohan, Pankaj Jha, Arnab Kumar Laha[1] and Goutam Dutta[2]

Indian Institute of Management,
Ahmedabad-380015, India

ABSTRACT

Artificial Neural Network (ANN) has been shown to be an efficient tool for non-parametric modeling of data in a variety of different contexts where the output is a non-linear function of the inputs. These include business forecasting, credit scoring, bond rating, business failure prediction, medicine, pattern recognition, and image processing. A large number of studies have been reported in the literature with reference to use of ANN in modeling stock prices in the western countries. However, not much work along these lines has been reported in the Indian context.

In this paper we discuss modeling of Indian stock market (price index) data using ANN. We study the efficacy of ANN in modeling the Bombay Stock Exchange (BSE) SENSEX weekly closing values. We develop two networks with three hidden layers for the purpose of this study which are denoted as ANN1 and ANN2. ANN1 takes as its inputs the weekly closing value, 52-week Moving Average of the weekly closing SENSEX values, 5-week Moving Average of the same, and the 10-week Oscillator for the past 200 weeks. ANN2 takes as its inputs the weekly closing value, 52-week Moving Average of the weekly closing SENSEX values, 5-week Moving Average of the same, and the 5-week volatility for the past 200 weeks. Both the neural networks are trained using data for 250 weeks starting January, 1997.

To assess the performance of the networks we used them to predict the weekly closing SENSEX values for the two year period beginning January, 2002. The root mean square error (RMSE) and mean absolute error (MAE) are chosen as indicators of performance of the networks. ANN1 achieved an RMSE of 4.82% and MAE of 3.93% while ANN2 achieved an RMSE of 6.87% and MAE of 5.52%.

---

[1] E-mail : arnab@iimahd.ernet.in

[2] E-mail : goutam@iimahd.ernet.in

1. Introduction

This paper is an attempt to explore the emerging field of applying Artificial Neural Network (ANN) to the complex task of modeling security prices in the Indian context. There have been number of attempts to apply ANN to the task of modeling security prices (see for example - Lam (2004), Nygren (2004), Kaastra and Boyd (1995), Cao et. al. (2005), Jasic and Wood(2004)). When it comes to performing a predictive analysis of the security prices, it is very difficult to build one general model that will fit every market and every security. Experience has shown that such models tend to be specific to markets and asset classes and a general model may not be applicable across markets and asset classes. Similarly, there may be some temporal changes as well which means that the models may need to be modified over time in order to preserve their effectiveness. One of the major problems faced in modeling financial market movements is the fact that information comes in from very large number of sources, and at least some of the price movements are a direct result of the expectation by market participants of *that* very movement (Soros, 1987). It has been observed that financial markets get affected by virtually anything that has a bearing on the economy. Thus, economic indicators (Andersen, 2004), political developments (Beaulieu et. al., 2006, McGillivray, 2003), terrorist attacks (Maillet and Michel, 2005), etc., apart from factors relevant to individual securities, have a bearing on market movements.

The paper is organized as follows : in section 2, we briefly discuss the fundamental concepts of ANN and its applications in various fields; in section

3, we discuss construction of ANNs for modeling BSE SENSEX data; in section 4, we discuss the performance of the constructed ANNs; and in section 5 we conclude.

2. Artificial Neural Networks

In recent years some work has been reported on the use of ANNs for analysis of financial markets. In what follows we provide a very brief introduction to ANNs following Stern(1996). The three essential features of an ANN are basic computing elements referred to as neurons, the network architecture describing the connections between the neurons, and the training algorithm used to find values of the network parameters for performing a particular task. Each neuron performs a simple calculation, a scalar function of a scalar input. Suppose we label the neurons with positive integers and denote the input to the k^{th} neuron as i_k and the output from the k^{th} neuron as o_k. Then $o_k = f_k(i_k)$ where $f_k(.)$ is a specified function that is typically monotone but otherwise arbitrary. The neurons are connected to each other in the sense that the output from one unit can serve as part of the input to another. There is a weight associated with each connection; the weight from unit j to unit k is denoted as w_{jk}. Let N(k) denote the set of units that are connected to unit k. The input to unit k is then $i_k = \sum_{j \in N(k)} w_{jk} o_j$. Network architecture refers to the organization of the neurons and the types of connections permitted. In the multilayer feedforward network, the type used in this paper, neurons are organized in a series of layers. Information flows only in one direction; units receive information only from units in higher layers of the network. A

multilayer feedforward network with one hidden layer for the non-linear XOR function – x_1 XOR x_2 = 1 if exactly one of x_1 and x_2 is 1 and = 0 otherwise; x_1 and x_2 can take only two values 0 or 1 - is given in Figure 1 below:

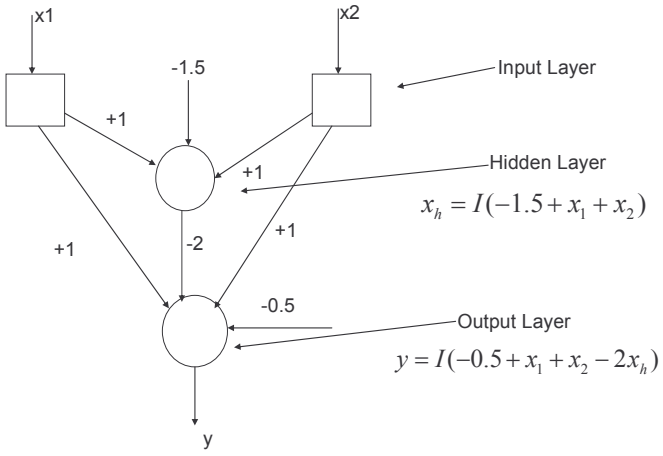

Figure 1 : Multilayer Feedforward Network for XOR function

The ANN can be "trained" to perform a specific task by adjusting these weights. The weights are continually adjusted by comparing the output of the network with the target until the output of the network "matches" the target in the sense that the error function measuring the difference between the target and the output is minimised. Many pairs of input and output are used to train the network and this mode of adjustment is called "supervised" learning. Figure 2 below summarizes the methodology:

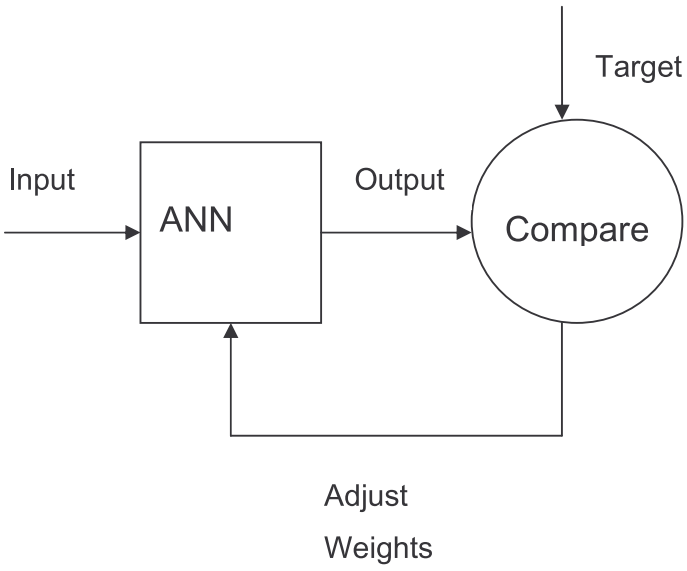

Figure 2: Methodology of Supervised Learning

There are several approaches for minimization of the error function including standard optimization techniques like conjugate gradient and Newton-type methods. However the most popular algorithm in this context is the backpropagation algorithm of Rumelhart, Hinton and Williams (1986). The algorithm is extremely simple to program but tends to converge slowly.

Training of a network can take two forms, “incremental” or “batch”. In case of incremental training, individual pairs are fed one at a time to the network. Output is compared with the target for each input and adjustments of the weights are made using a training algorithm, most often the backpropagation algorithm. The next pair is then fed to the network after making the adjustment for the previous pair and the same process is repeated. Incremental training is sometimes referred to as "on line" or "adaptive" training. In case of batch

training, all the pairs of input and output are fed to the network at the initial stage and the weights are adjusted.

In case of unsupervised learning, outputs for a given set of inputs are not available. Unsupervised learning is mostly used for networks that perform clustering and pattern recognition. Cheng and Titterington (1994) provides a comprehensive review of ANNs from a statistical perspective.

A well trained ANN can exploit the underlying non-linear relationships that drive security prices. Further, the networks can be retrained using newer data. Thus, a network can adapt to new information as it comes. This makes ANNs a very promising tool for security price modeling. The vast increase in computing power has made modeling of complex systems relatively easy and less time consuming than in the past.

ANNs are being used in various fields of application including business forecasting, credit scoring, bond rating, business failure prediction, medicine, pattern recognition, image processing, speech processing, computer vision, control systems etc. In the context of financial forecasting, Kuan and Liu(1995) discusses forecasting of foreign exchange rates using ANNs. They show that a properly designed ANN has lower out-of-sample mean squared prediction error relative to the random walk model. Jasic and Wood(2004) discusses the profitability of trading signals generated from the out-of-sample short-term predictions for daily returns of S&P 500, DAX, TOPIX and FTSE

stock market indices evaluated over the period 1965-1999. The out-of-sample prediction performance of neural networks is compared against a benchmark linear autoregressive model. They find that the buy and sell signals derived from neural network predictions are significantly different from unconditional one-day mean return and are likely to provide significant net profits for reasonable decision rules and transaction cost assumptions. Cao et. al. (2005) provides a comparison between the Fama and French's model and the ANN model in the context of prediction of the Chinese stock market. They report that the ANNs outperform the linear models from financial forecasting literature in terms of its predictive power. Tkacz (2001) provides an interesting study regarding the use of leading indicator neural network models for forecasting of Canadian GDP growth. It is reported that the neural network models yield statistically lower forecast errors for the year-over-year growth rate of real GDP relative to linear and univariate models.

Huang et. al. (2004) report a comparative study of application of Support Vector Machines(SVM) and Backpropagation Neural Networks (BNN) for analysis of corporate credit ratings. They report that the performances of SVM and BNN in this problem were comparable and both these models achieved about 80% prediction accuracy. Pendharkar(2005) discusses the application of ANNs for the bankruptcy prediction problem. It is reported that the ANNs perform better than the statistical discriminant analysis both for training and hold-out samples.

3. ANN for modeling BSE SENSEX

The most widely tracked and popular stock index in India is the BSE SENSEX. This index is extremely sensitive and has a high volatility. Though it is generally perceived as a not very reliable index of the overall market movement, the high liquidity of the stocks comprising the index and high level of sensitivity makes it an attractive choice for use as a proxy for stock price movements. We use weekly closing values of SENSEX for 250 trading weeks starting from January, 1997 for training the ANNs. The duration is long enough in order to model the prices accurately and has a "boom-bust" cycle. This is important since otherwise the ANNs will not be properly trained to handle a "boom-bust" cycle if that happens in future. Figure 3 below gives the weekly closing values of SENSEX for the period 1997-2003. The two year weekly closing values of SENSEX from January, 2002 onwards are used as the validation data set. It is to be noted that the Indian market witnessed a major trend change around May 2003, when the "bust" part of the cycle ended and a new "boom" started.

In this context, it may be noted that during the entire period 1997-2003 the process of liberalization of the Indian economy was carried forward by successive governments which came to power during this period. As a result, the period saw progressive easing of controls on capital and money market instruments, opening of equity markets further to international investors, removing foreign ownership ceilings in many sectors, deregulation of interest rates, reduction of import duty and tax rates etc. In our context, this would mean that the error rates for prediction observed on application of the ANNs

on the validation data set are likely to be somewhat higher than what it would be in a situation when there is no change in the economic environment between the periods when the input data and the validation data are collected.

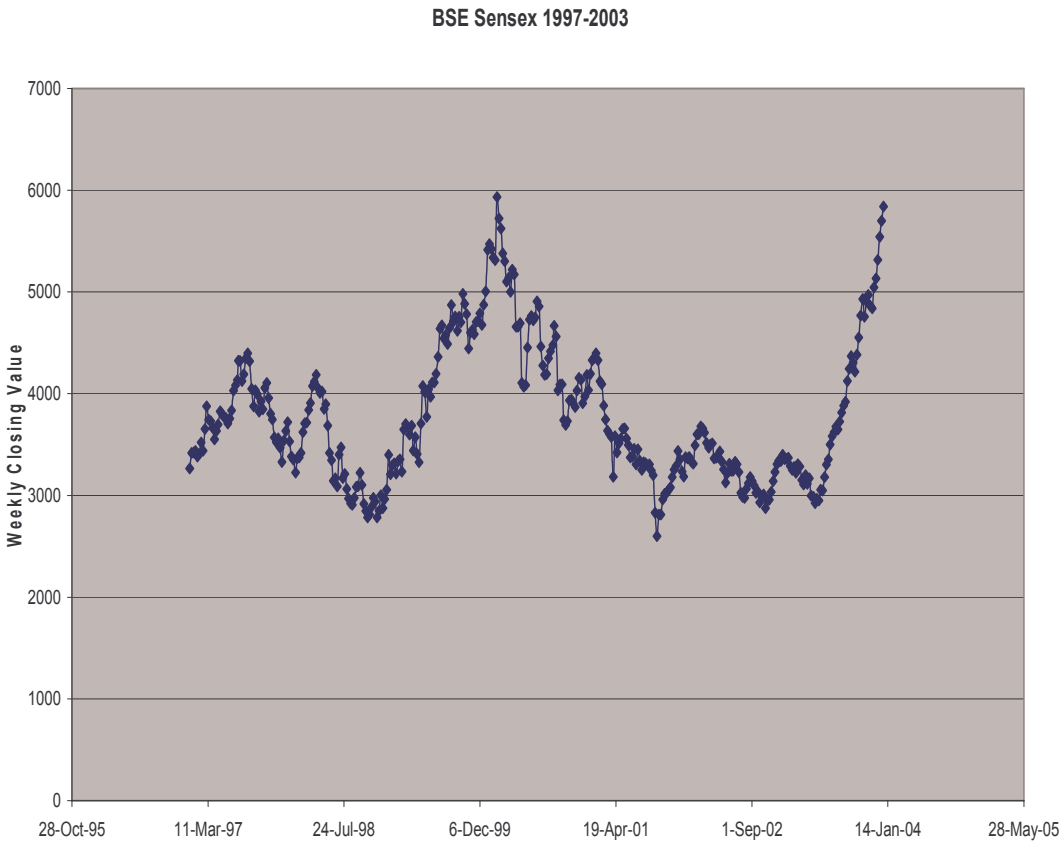

Figure 3 : BSE SENSEX weekly closing values for the period 1997-2003

In the remaining part of this section we discuss, with specific reference to the above data, construction of ANNs that perform reasonably well on the validation data set. We use in addition to the weekly closing values of SENSEX, some indicators which are well known in Technical Analysis and which can be easily calculated based on the historical price data, as input to

the ANNs. Feed Forward Backpropagation Networks are chosen for modeling purposes. Feed forward networks with one sigmoid layer and one linear layer are known to be able to model non-linear relationships of great complexity (Bishop, 1995). Depending on the complexity of the task, hidden layers can be added to the network in order to achieve desired level of accuracy. The software chosen for creating, training and testing the networks is MATLAB Neural Network Toolbox, which has an extensive capability in terms of creating and training different types of networks. The BSE SENSEX data is quite complex and even with a large number of neurons in the two main layers it could not be modeled effectively.

The need to increase the number of layers was felt in order to improve the performance. Initially, the number of sigmoid layers was increased to improve the performance. ANNs containing up to 8 sigmoid layers with up to 200 neurons each were tried in order to approximate the relationship. The inputs used were varied from using just preceding week's closing value to up to 26 weeks' closing values (going backwards from the week preceding the target week) and using a single target week's closing value as the output. It was seen that quite often the output was close to a single arbitrary number and showed very little fluctuation. Upon analysis, the problem was found in the structure of the network itself. A sigmoid layer can have one of the two transfer functions - tan sigmoid function (tansig) or log sigmoid (logsig) function. The tansig transfer function takes inputs over a wide range and provides output in the range of -1 to 1, while logsig function gives output in the

range of 0 to 1. Thus, the sigmoid layer takes an input over a wide range and provides an output over a limited range.

One of the problems associated with the sigmoid function is that towards the extreme, the gradient changes very little and hence, outputs change very little even for inputs which are quite different in value. Multiple layers with sigmoid transfer function clearly run into problems due to this reason. Successive layers of sigmoid transfer functions drive outputs to a constant number regardless of the variations in the inputs. Thus, in order to correctly model security price relationships, it is necessary to restrict the number of sigmoid layers to a minimum and if necessary, increase the number of linear layers only,

It is extremely important to choose the parameters of the sigmoid layer carefully as the network performance is extremely sensitive to these parameters. Again, this sensitivity arises from the nature of the transfer function. Since the gradient changes very little at the extremes of input values (and the backpropagation algorithm essentially relies on gradient changes), a transfer function with smaller range of outputs gives much poorer results. This effect was evident during the various experiments conducted by us, as the logsig transfer function performed rather poorly compared to the tansig transfer function. The reason possibly lies in the fact that tansig transfer function has double the range of outputs (-1 to +1) versus the logsig transfer

function (0 to +1). The tansig transfer function, therefore, allows a larger change in gradient for the extreme values of inputs.

An even larger impact of the transfer function appears on the input ranges chosen for the first layer of the network. In case of SENSEX, for example, the error during training dropped by almost 60% when input ranges were narrowed down to 2500-6000 from the initial range of 0-10000. Specifying an input range which is much larger than the possible values of inputs has a huge penalty in terms of accuracy and reliability of the network as gradient changes become much smaller due to larger input range, and network performance becomes poorer in line with increasing range. This creates some difficulty in modeling financial data. A narrow range around the training and validation data sets would reduce the errors made by the network in modeling the prices. But that reduces the usefulness of the network as the prices in future may move outside the range and the network starts giving erroneous results. A wide range, on the other hand, reduces accuracy but increases the usefulness of the network. It is necessary to choose the ranges keeping this trade-off in mind. The decision would depend on where the prices are at the time of the modeling (close to the extremes of the range or in the middle), for what time the model is intended to be used before retraining (whether retraining would happen after a quarter or a year) and so on. If the prices at the time of training the network, are in the middle of the price range exhibited by the input data, an input range very close to the price range is likely to serve the purpose very well as the prices are unlikely to move beyond the range in the short term. If, on the other hand, the prices are close to the

extreme of the price range exhibited by the input data, it would be necessary to provide some margin in terms of input range to that extreme so that even if prices move beyond the earlier range, the network still stays useful. Likewise, a network intended for short term use can do with narrower input ranges versus the network which is supposed to be trained for longer term use.

One of the difficult problems that are encountered while modeling complex systems is over-parameterization. Excessive number of parameters, coupled with variables and weights highly fine-tuned to the data available, can drive the error on test/ training data to extremely low levels. Over-parameterization, however, can impose a huge penalty in terms of subsequent validity of the model being created as the results can be significantly different from the actual observations. Some approaches to model choice typically uses a criterion function, analogous to the widely used Akaike Information Criterion (AIC) or Schwarz Information Criterion(SIC) in statistics, that contains a penalty term which increases with the number of parameters present in the model. The model which gives the best value according to the chosen criterion is then selected. However in the context of ANNs it is difficult to bring in a penalty function within the training process (Amari et. al., 2006). Instead, a validation data set may be used to select the network and the training algorithms. Over fitting problems in a Neural Network context would occur in the way of excessive number layers and of neurons in each layer. It is possible to keep on increasing the number of layers in the networks in order to drive the error down. By optimizing the network size based on the validation errors we have attempted to avoid the problem of over fitting. An approach

similar to that used in the context of genetic algorithms has been used to optimize the number of layers and neurons in the network. Networks with three different input profiles using different training algorithms were created with varying number of layers and neurons. While the number of sigmoid layers was kept to one, the number of linear layers was raised up to 10. The number of neurons was also varied between 100 and 1000 for the hidden linear layers and between 100 and 2000 for the sigmoid layer. Networks across all input profiles improved performance when the number of layers was raised from 1 in case of linear layers. But the gain dropped off around 3 hidden linear layers and 1 output linear layer. Performance, measured in terms of validation data set errors, dropped significantly after number of hidden layers went beyond six.

In terms of number of neurons a similar approach was used. Beyond a certain number of neurons, the results started showing performance losses. In case of sigmoid layer, the problem occurred after the number of neurons went beyond 1000. In case of hidden linear layers, the problem became visible after neurons went beyond 800 per layer.

Based on these initial experiments, the following combination was chosen for modeling:

Input Layer: Tan Sigmoid Transfer Function with 800 Neurons

Hidden Layers: Three Linear Layers with 600 Neurons each

Output Layer: Linear Layer with 1 Neuron

The above ANNs were trained using batch processing and the best performance was given by a quasi-Newton algorithm, One Step Secant (OSS) implemented in MATLAB. The technical indicators were chosen to represent different influences on the market prices; the medium term trends, short term fluctuations etc. The following variables/ indicators were used -

1. Weekly closing values of SENSEX for the past 200 weeks
2. 52-week moving averages of the weekly closing values for the past 200 weeks
3. 5-week moving averages of weekly closing values for the past 200 weeks
4. 5-week volatility of the weekly closing values (only in case of ANN2) for the past 200 weeks
5. 10-week Oscillator (or Momentum) (only in case of ANN1) for the past 200 weeks.

Moving averages are added as input to the ANNs on the premise that a moving average represents some sort of a trend in price for the given period. Different moving averages (5-week, 10-week, 13-week and so on) were tried to find the one that yielded significantly better results. Two moving averages were found to be performing better than the rest – 52-week moving average and 5-week moving average. When both of these are used in conjunction, significant gain in performance is noticed. Hence both are used as input to the ANNs. A possible explanation for this effect could be that the 52-week

moving average provides the long term trend information. The 5-week moving average then gives a reference point as to how far (and in which direction) values have been moving in the recent past (i.e. short term trend information).

10-week Oscillator (or Momentum) is an extremely simple indicator as it simply subtracts the value at 10 periods prior to latest value from the latest value. The resulting value is supposed to give information regarding the future direction of the values. Combined with the two moving averages, it is observed to improve performance of the ANN. We use the 10-week oscillator as an input variable for construction of ANN1.

Volatility computed over different periods- 5-week, 10-week, 13-week, 26-week and 52-week, was used to test the network performance. 5-week volatility gave the best performance. We use the 5-week volatility as an input variable for construction of ANN2.

Thus, in order to predict the SENSEX value for a given period, ANN1 uses the past 200 weeks information on the following: weekly closing values, 52-week moving average of the weekly closing SENSEX values, 5-week moving average of the same, and the 10-week Oscillator. On the other hand, ANN2 uses the past 200 weeks information on weekly closing values, 52-week moving average of the weekly closing SENSEX values, 5-week moving average of the same and the 5-week volatility.

4. Results

For ANN1, the Root Mean Square Error (RMSE) was 4.82% and the Mean Absolute Error (MAE) was 3.93% on the validation set. It was seen that the error towards the end of the validation set was higher than the earlier values. If the last 10 weeks' values are dropped from the validation set, the RMSE (MAE) drops to 4.40% (3.67%)

For ANN2, RMSE (MAE) was 6.87% (5.52%) on the validation set. As with ANN1, the error towards the end of the validation set was higher than the earlier values. If last 10 weeks' values are dropped from the validation set, the RMSE (MAE) drops to 6.596% (5.36%).

Considering all of the above, we find that the performance of ANN1 is better than that of ANN2 for predicting the weekly closing values of BSE SENSEX. With an MAE of 3.93% on the validation data set it is expected that ANN1 will work well for prediction purposes. However, it is well known that the performance of ANNs, with respect to forecasting, depends on a variety of factors (Zhang et. al., 1998). Thus, the performance of ANN1 cannot be compared easily with the published findings of other studies.

5. Conclusion

The RMSE and MAE achieved by ANN1 on weekly closing values of SENSEX in the validation data set is quite commendable given the reputation of

SENSEX being a volatile index. The 5-week (annualized) volatility of SENSEX varied between 6% and 63% during the training data period and between 6% and 31% during the validation data period. In such a volatile environment, predictions with the level of error reported above are likely to be quite useful. It has been observed that the error increases gradually during the validation period. Thus, an appropriate approach may be to retrain the network periodically (may be after every six months). There is considerable scope to build on these results further and build ANN models that can predict the security prices with higher level of accuracy.

Acknowledgement: The authors thankfully acknowledge the helpful comments of the anonymous referee which has led to improvement of this paper.

References

Amari, S., Park, H. and Ozeki, T. (2006): Singularities Affect Dynamics of Learning in Neuromanifolds, *Neural Computation*, 18, 5, 1007-1065.

Andersen, J. V. (2004): Estimating the Level of Cash Invested in Financial Markets, *Physica A*, 344, 1/2, 168-173.

Beaulieu, M-C, Cosset, J-C, Essaddam, N. (2006): Political Uncertainty and Stock Market Returns: Evidence from the 1995 Quebec Referendum, *Canadian Journal of Economics*, 39, 2, 621-641.

Bishop, C.M. (1995). *Neural Networks for Pattern Recognition.* Oxford University Press, New York:.

Cao, Q., Leggio, K. B., and Schniederjans, M. J. (2005): A comparison between Fama and French's model and artificial neural networks in predicting the Chinese stock market, *Computers & Operations Research*, 32, 2499-2512.

Cheng, B. and Titterington, D. M. (1994): Neural Networks: A Review from a Statistical Perspective, *Statistical Science*, 9, 1, 2-30.

Huang, Z., Chen, H., Hsu, C-J, Chen, W-H, and Wu, S. (2004): Credit Rating Analysis with Support Vector Machines and Neural Networks: A Market Comparative Study, *Decision Support Systems*, 37, 543-558.

Jasic, T. and Wood, D. (2004): The profitability of daily stock market indices trades based on neural network predictions: case study for the S&P 500, the DAX, the TOPIX and the FTSE in the period 1965-1999, *Applied Financial Economics*, 14, 285-297.

Kaastra, L. and Boyd, M. (1995): Designing a neural network for forecasting financial and economic time series, *Neurocomputing*(10).

Kuan, C-M and Liu, T. (1995): Forecasting Exchange Rates using Feedforward and Recurrent Neural Networks, *Journal of Applied Econometrics*, Vol. 10, 347-364.

Lam, M. (2004): Neural Network Techniques for Financial Performance Prediction: Integrating Fundamental and Technical Analysis, *Decision Support Systems*, 37(4). 567-581.

Maillet, B. B. and Michel, T. L. (2005): The Impact of the 9/11 Events on the American and French Stock Markets, *Review of International Economics*, 13(3), 597-611.

McGillivray, F. (2003): Redistributive Politics and Stock Price Dispersion, *British Journal of Political Science*, 33, 3, 367-395.

Nygren, K. (2004) : *Stock Prediction – A Neural Network Approach*, Master's Thesis, Royal Institute of Technology, KTH, Sweden.

Pendharkar, P.C. (2005): A Threshold-varying Artificial Neural Network approach for classification and its application to bankruptcy prediction problem, *Computers and Operations Research*, 32, 2561-2582

Rumelhart, D. E., Hinton, G.E. and Williams, R. J. (1986): Learning Internal Representations by Error Propagation, in *Parallel Distributed Processing: Exploration in the Microstructure of Cognition, Vol. I,* pp. 318-362, MIT Press, Cambridge, MA.

Soros, G. (1987). *The Alchemy of Finance: Reading the Mind of the Market*. John Wiley & Sons, Inc. , New York:

Stern, H. S. (1996): Neural Networks in Applied Statistics, *Technometrics*, 38, 3, 205-214.

Tkacz, G. (2001): Neural Network forecasting of Canadian GDP growth, *International Journal of Forecasting*, 17, 57-69.

Zhang, G., Patuwo, B. E., and Hu, M. Y. (1998): Forecasting with Artificial Neural Networks: The State of the Art, *International Journal of Forecasting*, 14, 35-62.