A Multi-Component Nonlinear Prediction System for the S&P 500 Index

Tim Chenoweth[1,2,3] Zoran Obradović[1]

tchenowe@eecs.wsu.edu zoran@eecs.wsu.edu

[1] School of Electrical Engineering and Computer Science

[2] Department of Management and Systems

[3] Department of Economics

Washington State University, Pullman WA 99164-2752

Phone: 509-335-6601, Fax: 509-335-3818

Abstract

The proposed stock market prediction system is comprised of two preprocessing components, two specialized neural networks, and a decision rule base. First, the preprocessing components determine the most relevant features for stock market prediction, remove the noise, and separate the remaining patterns into two disjoint sets. Next, the two neural networks predict the market's rate of return, with one network trained to recognize positive and the other negative returns. Finally, the decision rule base takes both return predictions and determines a buy/sell recommendation. Daily and monthly experiments are conducted and performance measured by computing the annual rate of return and the return per trade. Comparison of the results achieved by the dual neural network system to that of the single neural network shows that the dual neural network system gives much larger returns with fewer trades. In addition, dual NN experiments with the appropriately selected filtering and decision thresholds managed to achieve an almost twice larger annual rate of return when compared to that of the buy and hold strategy over a seventy month period. However, no claims can be made that the proposed system is better than the buy and hold strategy when considering transaction costs.

Keywords: Stock market prediction; Hierarchical systems; Hybrid systems; Neural Networks.

1. Introduction

Financial markets in general and the stock market in particular are extremely efficient, meaning that at any given point in time the market does a very good job reflecting the actual value of the underlying stocks. In fact, the efficient market hypotheses states that any new information which affects this value is accounted for by the market before the general public can make trades based on it [13]. This hypotheses seems reasonable *provided* that the relationship between the new information and the value of the underlying stock is fully understood. Most quantitative methods attempting to capture such relationships are based on simple stochastic linear time series models [2, 14, 16]. This research, similar to [12], hypothesizes that there might be nonlinear relationships between market information and the value of stocks that so far have not been identified and therefore are not reflected in stock prices. In other words, there might still be inefficiencies in the market. It is important to note that if this is true and nonlinear relationships do exits, once their existence (and the means for identifying them) becomes public knowledge, traders will have a better understanding of the relationship between information and stock values, which means both a more efficient market and the elimination of these nonlinear relationships. The long term results then will not be higher returns, but a more efficient market.

If these nonlinear relationships exist, it may be possible to capture them using a nonparametric machine learning approach of multilayer artificial neural networks (NN). Such NN's are powerful computational systems that theoretically can approximate any nonlinear continuous function on a compact domain to any desired degree of accuracy [9]. In addition, a NN can account for fundamental changes in the underlying function through incremental retraining. However, the stock market is a highly complex system which takes a very large quantity of information (fundamentals, news, etc.) and produces a price movement. What a NN is trying to do is model this system using only a very small portion of the total information. The other information not accounted for now plays the role of noise, resulting in a problem domain that is extremely noisy. In addition, due to

the continuous evolution of the market, historical data may represent patterns of behavior that no longer hold. This combination of a noisy environment and non-stationarity makes it very difficult for a NN to learn a function which generalizes when applied to new data [1].

Therefore, this paper proposes a hybrid multi-component nonlinear system for S&P 500 stock market predictions. The system goals are to earn a larger annual return than the buy and hold strategy and to keep the number of trades low to reduce transaction costs. The system details are explained in Section 2 followed by results and analysis in Section 3, and conclusions in Section 4. An extended abstract of this paper using only daily data appears in [6]. The reader is also referred to companion papers [5, 7] dealing with the specific issue of feature selection.

2. Methodology

The proposed system consists of a statistical feature selection component for identification of the most relevant data, a data filtering component for removing noise and splitting the remaining data, two specialized NN's for extraction of nonlinear relationships from the selected data, and high level decision rules for determining buy/sell recommendations (see Fig. 1).

2.1. Feature Selection

The objective of the feature selection component is to identify a small subset of the most relevant features from a larger pool for designing the system in a manner that preserves as much information as possible. This issue is important because fewer features per pattern lead to faster computation and require less training patterns for successful generalization. Most feature selection processes rely on fundamentally sound statistically based techniques [11]. These techniques are practical, easy to understand, and easily implemented. However, they suffer from *instability* problems, meaning that small data perturbations lead to drastic changes in the final reduced feature set [3]. This problem is especially pronounced for stock market models because the data is non-stationary and very noise. For these reasons, the feature selection procedure adopted in this paper is to use several

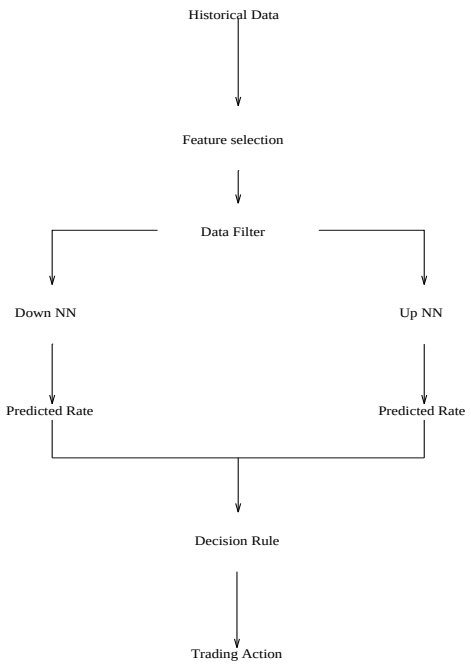

Figure 1: System Architecture

selection techniques and criteria, then combine the partial results using a ranking process proposed in [7]. Techniques are discussed in Section 2.1.1, criteria in Section 2.1.2, and the ranking process in Section 2.1.3.

2.1.1. Selection Techniques

Each feature selection technique is a search algorithm that attempts to determine a subset of the existing features which maximizes the differences between the classes based on some criteria. This section briefly describes the selection techniques used in the proposed feature selection process, all of which are described in more detail in [11].

The *Sequential Forward Search* selection technique is a greedy algorithm that begins with an empty feature set and adds features to it one at a time. The first feature added is the one deemed to be the best according to the selection criteria. The next feature added is the one which results in the largest improvement when considered in conjunction with the first feature. Similarly, the i^{th} feature added is the one that results in the largest improvement when considered in conjunction

with the previous $i-1$ features. The *Sequential Backward Search* selection technique is similar to the sequential forward search, except that the initial set contains all the features, and features are removed from this set one at a time. The first feature removed is the one that results in the smallest degradation when the remaining features are considered together. This process repeats until the feature set reaches a predetermined size.

Although both the Sequential Forward Search and the Sequential Backward Search consider features in combination and as such are fine grained techniques, both require significant computing time. In addition, with Sequential Forward Search once a feature is added to the features set it cannot be removed. With Sequential Backward Search once a feature is removed it cannot be added later. As such, neither Sequential Forward Search nore Sequential Backward Search guarantee that an optimal set of features is achieved.

2.1.2. Selection Criteria

Each feature selection criteria is based either on a measure of the distance between classes or an estimation of the classification error. Therefore the *selection criteria* objective is to either maximize the separation between classes using some distance measure of intra-class separation or to minimize the estimated classification error. This Section describes the selection criteria used in the proposed feature selection process, all of which are described in more detail in [11].

The Euclidean, Patrick-Fisher, Mahalanobis, and Bhattacharyya distances used in this study are all means of measuring the multidimensional separation between classes. The *Euclidean* distance is measured as

$$\sqrt{(M_2 - M_1)^T (M_2 - M_1)}, \tag{1}$$

and the *Patrick–Fisher* distance as

$$\left(\frac{\Sigma_1 + \Sigma_2}{2}\right)^{-1} (M_2 - M_1), \tag{2}$$

where M_1 and M_2 are the mean vectors of class one and two respectively (e.g. M_1 is computed by averaging each feature's values for data in class one) and Σ_1 and Σ_2 are the corresponding covariance matrices. The *Mahalanobis* distance is measured as

$$(M_2 - M_1)^T \Sigma^{-1} (M_2 - M_1), \tag{3}$$

where the data from both classes is used to compute one covariance matrix Σ. And finally, the *Bhattacharyya* distance is measured as

$$\frac{1}{8}(M_2 - M_1)^T \left(\frac{\Sigma_1 + \Sigma_2}{2}\right)^{-1} (M_2 - M_1) + \frac{1}{2} \ln \frac{\left|\frac{\Sigma_1 + \Sigma_2}{2}\right|}{\sqrt{|\Sigma_1||\Sigma_2|}}. \tag{4}$$

Notice that the Bhattacharyya distance does not assume equal covariance matrices like the Mahalanobis distance. The first term in equation (4) measures the class separability due to the mean difference, while the second term measures the class separability due to the covariance-difference.

The final selection criteria used in this study is the *Estimated Minimal Error Probability* that estimates Bayes error for the data set by applying the K-nearest neighbor classifier [8] to the training set utilizing the leave one out approach. This tends to overestimate the error and as such gives a very conservative error estimate. The selection criteria then becomes finding a set of features that minimizes the estimated Bayes error.

2.1.3. The Ranking Feature Selection Process

The approach proposed here for dealing with the instability of statistically based feature selection techniques for noisy and non-stationary data is to perform feature selection using several combinations of selection techniques and selection criteria and to integrate the obtained partial results using a ranking process. The *ranking process* uses a specific technique and criteria combination to determine a rank ordering of the features from best to worst with scores assigned to the features based on this ordering (one is the best, p is the worst, where p is the total number of features). The process is repeated using combinations of other techniques and criteria explained earlier and

the scores are summed. The reduced feature set is then comprised of the q features with the lowest scores.

Had the initial data set been too large, this process would have been prohibitively expensive. For such cases, a two-stage ranking process is proposed in which the first phase reduces the initial feature set to an intermediate set of manageable size using coarser, computationally less demanding feature selection techniques. Then in the second phase the finer but more expensive Sequential Forward and Sequential Backward selection techniques are applied to the intermediate feature set to derive the final set [7].

2.2. Return Rate Prediction

The return rate prediction sub-system consists of a data filtering component and two NN's that are trained using the back-propagation algorithm and an on-line learning scheme (see Fig. 1). The filtering component's objective is to reduce the degree of noise in the data and then split the remaining patterns into two disjoint sets. The "up NN" is trained on the set containing patterns with positive target returns and the "down NN" on the set containing patterns with negative target returns. Once both NN's are trained, the test pattern is presented to each and the corresponding predictions are collected. A decision rule base is applied to these predictions and a buy/sell recommendation made as explained in Section 2.3.

The on-line learning scheme consists of a sequence of training/prediction sessions where the NN's are retrained after each session using more recent information. This is achieved by training the NN's using the back-propagation algorithm [15] and patterns from a fixed size window covering a continuous time segment of historic data. The target return for the time unit immediately following the window is predicted by both NN's and the predictions used by the rule base. Then the training window is shifted forward one time unit (i.e., one trading day or one month), the patterns from the new window used to retrain the NN's, and a prediction made for the next time unit. This process is repeated until the data set is exhausted.

For each training session the target return corresponding to each pattern in the window is compared to a *filtering threshold* value h. If the return is greater than h the corresponding pattern is added to the "up NN" training set, if the return is less than -h the pattern is added to the "down NN" training set. Any pattern with a target return between -h and h is discarded.

For example, suppose that the training window size is m and that at time t the test pattern is d_t, which means that the training window contains patterns d_{t-m} through d_{t-1}. First, the patterns in the training window (d_{t-m} through d_{t-1}) are separated into "up NN" and "down NN" training sets using the threshold value h as described. Next, both NN's are trained using their respective training sets, and asked to predict the target return for the test pattern d_t. Once the predictions are collected and sent to the decision rule base, the training window is shifted forward one time unit so that the new test pattern is d_{t+1} and the new training window contains patterns d_{t-m+1} through d_t, and the process repeated. This continuous until the end of the ordered data set is reached.

2.3. Decision Rule Base

The predicted returns from both NN components are used as input to the decision rule base component (see Fig. 1). This component analyzes the predicted returns and outputs a buy/sell recommendation that is used to establish either a long or short position in the market. A long position means purchasing an asset for later resale, while a short position means selling a borrowed asset now and purchasing it later.

This study examines three different decision rule bases. For each rule base the "up NN" prediction r_u is compared to the "down NN" prediction r_d. Each rule base recommends a long position in the market if $r_u > 0$ and $r_d \geq 0$, and a short position if $r_u \leq 0$ and $r_d < 0$. Otherwise the rule base computes the normalized difference *diff* as

$$diff = \frac{\max\{|r_u|, |r_d|\} - \min\{|r_u|, |r_d|\}}{\max\{|r_u|, |r_d|\}}, \tag{5}$$

compares this ratio to a predefined *decision threshold* value y, and determines a buy/sell recommendation as follows:

- **Rule Base 1: Maintain Current Position Until a Clear Buy/Sell Recommendation is Received.**

  This rule base specifies that if the system is unsure as to what recommendation to make, the action is to do nothing and maintain the old position. Under these rules, if $r_u \leq 0$ and $r_d \geq 0$ the system recommends maintaining the current position (i.e., do nothing). If $r_u > 0$, $r_d < 0$, and $diff > y$ the rule base recommends a long position providing $r_u > |r_d|$, and a short position providing $r_u < |r_d|$. Otherwise $diff \leq y$ and the recommendation is to maintain the current market position.

- **Rule Base 2: Hold a Long Position in the Market Unless a Clear Sell Recommendation is Received.**

  This rule base takes advantage of the common *a priori* knowledge that over the past 65 years the market has increased at an average annual rate greater than 10%. Stated another way, this means that given no other information the odds are that the market will increase. This is, in fact, the whole premise behind the buy and hold strategy. The difference between rule base two and rule base one is the actions taken under uncertainty. In this instance the action is to take a long position in the market. Under these rules, if $r_u > 0$, $r_d < 0$, $diff > y$, and $r_u < |r_d|$, the system recommends a short position. Otherwise the recommendation is to take a long position.

- **Rule Base 3: Stay Out of the Market Unless a Clear Buy/Sell Recommendation is Received.**

Again, the difference between rule base three and the previous rules is the action taken when the system is uncertain as to what recommendation to make. In case of uncertainty, the rule base three action is to exit the market. More precisely, if $r_u \leq 0$ and $r_d \geq 0$ the system recommends exiting the market (i.e., if the current position is long then sell, if it is short then buy). If $r_u > 0$, $r_d < 0$, and $diff > y$ the system recommends a long position providing $r_u > |r_d|$, and a short position providing $r_u < |r_d|$. Otherwise $diff \leq y$ and the recommendation is to exit the market.

2.4. Performance Measures

The most important criteria when measuring the performance of a stock market prediction model is whether it will make money and how much. Therefore the model's annual rate of return (ARR) is computed as

$$ARR = \frac{k}{n} \sum_{i=1}^{n} r_i, \tag{6}$$

where:

k is the number of trading time units per year (i.e., 253 for daily trading, 12 for monthly trading);

n is the total number of trading time units for the experiment (e.g 2,530 for daily trading, 120 for monthly trading in an experiment lasting 10 years);

r_i is the rate of return for time unit i.

The sum, $\sum_{i=1}^{n} r_i$, is computed by either adding, subtracting, or discarding the actual returns for the S&P 500 index. If the system recommends a long position, the actual return is added to the sum; if a short position is recommended, the return is subtracted; or if the recommendation is to exit the market, the return is discarded.

It is also important to minimize transaction costs by controlling excessive trading (e.g. a 10% return with 50 trades is more profitable than a 10% return with 100 trades). Therefore the break even transaction cost ($BETC$), which may be viewed as the return per trade, is computed as

$$BETC = \frac{1}{s}\sum_{i=1}^{n} r_i, \tag{7}$$

where s is the total number of trading transactions, while r_i and n are defined as previously [12]. A *trade* is defined as any action that changes a market position. For example, exiting the market constitutes a single trade (i.e., a buy trade to cover a short position or a sell trade to cover a long position), while switching from a short position to a long position constitutes two trades (i.e., one buy trade to cover the short position and another buy to establish the long position).

3. Results and Analysis

The system described in Section 2 is used for both daily and monthly S&P 500 stock market buy/sell recommendations. The daily historic data used in this experiment is 2,273 ordered financial time series patterns from the period January 1, 1985 to December 31, 1993. The first 1,000 patterns from January 1, 1985 to December 19, 1988 comprised the initial training window, whereas actual predictions were made for the 1,273 patterns from December 20, 1988 to December 31, 1993. Each pattern in the initial data set contained 24 monthly and 8 daily features as shown in the Appendix (Table 3). This was reduced to six features in the final data set, also shown in Table 3, by using the ranking feature selection process described in Section 2.

The monthly historic data consisted of an initial training window of 162 patterns formed using the data from January 1973 to February 1987 and actual predictions made for the 70 month period from March 1987 to December 1992. The initial monthly data set containing 29 monthly features was reduced to eight in the final data set by using the feature selection process described in Section 2. Both the initial and the reduced feature sets are shown in the Appendix (Table 4).

	Daily			*Monthly*		
Preprocessing	*ARR*	*BETC*	*Trades*	*ARR*	*BETC*	*Trades*
Initial Features	-2.16%	-0.00%	905	-1.67%	-0.00%	62
Reduced Features	2.86%	0.00%	957	-3.33%	-0.01%	56
Reduced Features and 0.5% Noise Removal	5.61%	0.01%	476	-2.97%	-0.01%	52

Table 1: Results for the Single NN System

3.1. Daily Results

The importance of the proposed feature selection and noise removal components were first tested using a *single NN system*. In this system the data was not split into up and down sets and a single NN replaced the dual NN component, making the decision rule base unnecessary. Results for a single NN system using the initial set of features, the reduced set, and the reduced set with a filtering threshold value of 0.5% are shown in Table 1 with the system parameters described in Table 2. In these experiments the ARR improves with feature reduction and noise removal. This provides evidence that in daily trading both feature selection and data filtering improve the NN predictive capability.

Several experiments with the dual NN system described in Section 2 were conducted using the reduced feature set from Table 3 and the system parameters from Table 2. Note that the training window size for the dual NN experiments is larger than for the single NN. This window size increase is needed since the training window for the dual NN system is split into 3 disjoint sets. The first set, consisting of all patterns with a target rate greater than a filtering threshold h, is used to train the "up NN". The second set, consisting of all patterns with a target rate less than -h, is used to train the "down NN." Finally, the third set, consisting of all patterns with a target return between -h and h, is discarded. Consequently, to ensure an adequately sized training set for both NN components in the dual NN system, it is necessary to have a larger window size. For the dual NN system, extensive experiments are conducted varying the filtering and decision thresholds h and y. The filtering threshold h is varied from 0.25% to 1.25% in increments of 0.25% with the

Parameter	*Daily*	*Monthly*
Topology (Single NN, Initial Features)	32-4-1	29-4-1
Topology (Single NN, Reduced Features)	6-4-1	8-3-1
Topology (Dual NN)	6-4-1	8-3-1
Training Window Size (Dual NN)	1000	162
Training Window Size (Single NN)	250	162
Activation Function	Tangent Hyperbolic	Tangent Hyperbolic
Learning Rate	0.3	0.05
Tolerance	0.00001	0.00001
Number of Iterations	5000	5000

Table 2: System Parameter Values

best results obtained using a filtering threshold $h = 0.5\%$. The decision threshold y is varied from 0 to 0.80 in increments of 0.05. The upper boundary valued of 0.80 is used because larger values did not improve the results. A comparison between the ARR for rule one and rule two and the number of trades for rule one and rule two using a filtering threshold value $h = 0.5\%$ are shown in Figures 2 and 3 respectively. The best annual rate of return was 13.35% and was obtained using rule base two with thresholds $h = 0.5\%$ and $y = 0.80$. In comparison, the annual rate of return for the same period using the buy and hold strategy is 11.23% and the best return for the single NN is only 5.61%. Results for decision rule three are significantly less than the results using decision rules one and two (best ARR and $BETC$ are 6.50% and 0.08% respectively) and as such are not presented.

3.2. Monthly Results

Results for a single NN system using the initial set of features, the reduced set, and the reduced set with a filtering threshold value of 0.5% are also shown in Table 1 while the system parameters for both single NN's are shown in Table 2. The reduced feature set gave results that were worse than those achieved using the complete feature set, but without drastic information loss. In addition,

Figure 3: Number of Trades Comparison Using Daily Data and Filtering Threshold $h = 0.5\%$

Figure 4: ARR Comparison Using Monthly Data and Filtering Threshold $h = 0.0\%$

noise removal did not significantly improve the results, indicating that noise removal may not be necessary for the smoother monthly data.

Several experiments with the dual NN system described in Section 2 were conducted using the reduced feature set from the Appendix (Table 4) and the system parameters from Table 2. For the dual NN system, extensive experiments are conducted varying the thresholds h and y. The filtering threshold h is varied from 0.0% to 3.00%. Observe that for daily experiment a corresponding upper bound is 1.25% versus the 3.00% used here. This 3.00% study interval is not feasible for daily data since it would remove too much data. For all rules the best results were achieved using a filtering value $h = 0\%$, meaning that there was no need for noise reduction as earlier indicated by the single NN results. The decision threshold y is varied from 0 to 0.95 in increments of 0.05. The upper boundary value of 0.95 is used because larger values did not improve the results. A comparison between the ARR for rule one and rule two and the number of trades for rule one and rule two are shown in Figures 4 and 5 respectively. The best annual rate of return was 16.39% and was obtained using rule base two with thresholds $h = 0\%$ and $y = 0.95$. In comparison, the annual rate of return for the buy and hold strategy was 8.76% and the best return for the single NN was -1.67%. Results

Figure 5: Number of Trades Comparison Using Monthly Data and Filtering Threshold $h = 0.0\%$

for decision rule three are significantly less than the results using decision rules one and two (best ARR and $BETC$ are 8.76% and 1.51% respectively) and as such are not presented.

4. Conclusions and Future Research

The system proposed in this paper is comprised of a preprocessing component for feature selection, a filtering component for noise removal and pattern separation, two specialized NN components for return predictions, and a decision rule component for buy/sell recommendation. Various experiments using this system to predict S&P 500 index movements were conducted and associated annual rates of return and returns per transaction computed.

Comparison of the results achieved by the dual-NN system to that of the single NN shows that the dual NN system gives a larger return with fewer trades. In addition, dual NN experiments with the appropriately selected filtering and decision thresholds managed to achieve an annual rate of return greater than that of the buy and hold strategy. Comparing the daily to the monthly experimental results, it seems evident that the proposed system yields better returns when using slower data sampling (e.g. monthly rather than daily data). Although the results are promising, it

is important to note that no claims can be made that the proposed system is better than the buy and hold strategy when considering transaction costs. No system configuration achieved a better return per transaction than the buy and hold strategy for the same period.

However, the proposed system is still in development and research in progress might lead to further improvements. For instance, in the current two NN system there was no attempt to optimize all the system parameters. It is possible that optimized learning parameters such as the learning rate may lead to better results. It also might be possible to improve results by further restricting the scope of each NN by incorporating additional NN's. For example, in a four network system the first NN can be trained on large down movements, the second NN on small down movements, the third on small up movements, and the fourth on large up movements. Further improvements might also be obtained by incorporating prior knowledge and constructive NN learning [10], or a recurrent network topology [4]. Finally, the current decision rule bases are fairly simplistic. Possible improvements might be obtained by incorporating technical information like moving averages and exponential averages into the system. Experiments using additional NN's, an expert system, and combinations of the two to analyze the existing system information and determine a market direction are in progress.

Acknowledgments

We would like to thank Dr. Wayne Joerding, Dr. Jeff Schlimmer, Dr. Bernie Han, Dr. Craig Tyran, and Radu Drossu for their constructive comments on the preliminary version of this manuscript. Partial support by the NSF research grant NSF-IRI-9308523 to Zoran Obradović is gratefully acknowledged.

References

[1] Y.S. Abu-Mostafa, "Financial market Applications of Learning from Hints," in A.N. Refenes, ed., *Neural Networks in the Capital Markets,* (Willey, England, 1994).

[2] F. Black and M. Scholes, "The pricing of Options and Corporate Liabilities," *Journal of Political Economy,* 81 (May-June 1973).

[3] L. Breiman, "The Heuristics of Instability in Model Selection," Technical Report No. 416, Statistics Department, University of California, Berkeley, CA, 1994.

[4] A. Burgess and D. Bunn, "The Use of Error Feedback Terms in Neural Network Modelling of Financial Time Series," *Proc. the 1994 Neural Networks in the Capital Markets Conference*, Pasadena, CA (1994).

[5] T. Chenoweth and Z. Obradovic, "Feature Selection for Predictive Models of the Stock Market," *Proc. the 1994 Neural Networks in the Capital Markets Conference*, Pasadena, CA (1994).

[6] T. Chenoweth and Z. Obradovic, "A Multi-Component Approach to Stock Market Predictions," *Proc. the Third International Conference on Artificial Intelligence Applications on Wall Street*, New York, NY (1995).

[7] T. Chenoweth and Z. Obradovic, "An Explicit Feature Selection Strategy for Predictive Models of the S&P 500 Index," To be published in a future issue of NeuroVe$st^{TM} Journal.

[8] T.M. Cover and P.E. Hart, "Nearest Neighbor Pattern Classification," *IEEE Trans. Inform. Theory*, IT-13 (1967) 21-27.

[9] G. Cybenko, "Approximation by Superpositions of a Sigmoidal Function," *Mathematics of Control, Signals, and Systems,* 2 (1989) 303-314.

[10] J. Fletcher and Z. Obradovic, "Combining Prior Symbolic Knowledge and Constructive Neural Networks," *Connection Science: Journal of Neural Computing, Artificial Intelligence and Cognitive Research,* 5 (3–4) (1993) 365-375.

[11] K. Fukunaga, *Introduction to Statistical Pattern Recognition,* (Academic Press, San Diego, CA, 1990).

[12] J. Hutchinson, *A Radial Basis Function Approach to Financial Time Series Analysis,* PhD Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1993.

[13] M. Jensen, "Some Anomalous Evidence Regarding Market Efficiency," *Journal of Financial Economics,* 6 (1978) 95–101.

[14] J. Markowitz, *Portfolio Selection: Efficient Diversification of Investments*, (John Wiley & Sons, New York, NY, 1959).

[15] Rumelhart, etc. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, 1 and 2, (MIT Press, Cambridge, MA, 1986).

[16] W. Sharpe, "Capital Asset Prices: A Theory of Market Equilibrium," *Journal of Finance* (September 1964).

Tim Chenoweth (tchenowe@eecs.wsu.edu) received a B.S. degree in Mathematics in 1981 from the Coast Guard Academy, a M.B.A in Finance from Washington State University in 1991, and is currently completing a M.S. in Computer Science and an Individual Interdisciplinary Ph.D. combining Business and Computer Science, both from Washington State University. He was an active duty officer in the Coast Guard from 1981 to 1989. The objective of his current research is to use advanced technologies to model financial markets.

Zoran Obradovic (zoran@eecs.wsu.edu) received the B.S. degree in Applied Mathematics, Information and Computer Sciences in 1985; the M.S. degree in Mathematics and Computer Science in 1987, both from the University of Belgrade; and the Ph.D. degree in Computer Science from the Pennsylvania State University in 1991. He was a systems programmer at the Department for Computer Design at the Vinca Institute, Belgrade, from 1984 to 1986, and has been a research scientist at the Mathematical Institute of the Serbian Academy of Sciences and Arts, Belgrade, since then. At present, he is an Assistant Professor in the School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164-2752, USA. The objective of his current research is to explore applicability of neural networks technology to large scale classification and time series prediction problems in very noisy domains.

Appendix

Initial Features	*Reduced*
Return on 30 year Government Bonds	X
30 year Government Bond index	
Rate of change in the return on Government Bonds	
Rate of change in the return on Government Bonds lagged 1 month	
Rate of change in the return on Government Bonds lagged 2 months	
Rate of change in the return on Government Bonds lagged 3 months	
Rate of change in the return on Government Bonds lagged 4 months	
Rate of change in the return on Government Bonds lagged 5 months	
Return on U.S. Treasury Bills	
U.S Treasury Bill Index	
Rate of change in the return on T-bills	
Rate of change in the return on U.S Treasury bills lagged 1 month	
Rate of change in the return on U.S Treasury bills lagged 2 months	X
Rate of change in the return on U.S Treasury bills lagged 3 months	X
Rate of change in the return on U.S Treasury bills lagged 4 months	
Rate of change in the return on U.S Treasury bills lagged 5 months	
The CPI	
The percentage increase in the CPI	
The rate of change in the percentage increase in the CPI	
The rate of change in the percentage increase in the CPI lagged 1 month	
The rate of change in the percentage increase in the CPI lagged 2 months	
The rate of change in the percentage increase in the CPI lagged 3 months	
The rate of change in the percentage increase in the CPI lagged 4 months	
The rate of change in the percentage increase in the CPI lagged 5 months	
The S&P Composite Index	
The S&P Composite Index lagged 1 day	
The S&P Composite Index lagged 2 days	
The return on the S&P Composite Index	X
The return on the S&P Composite Index lagged 1 day	X
The return on the S&P Composite Index lagged 2 days	X
The return on the S&P Composite Index lagged 3 days	
The return on the S&P Composite Index lagged 4 days	

Table 3: **Features for Daily Experiments**

Initial Features	*Reduced*
Return on 30 year Government Bonds	
30 year Government Bond index	X
Rate of change in the return on Government Bonds	
Rate of change in the return on Government Bonds lagged 1 month	
Rate of change in the return on Government Bonds lagged 2 months	
Rate of change in the return on Government Bonds lagged 3 months	X
Rate of change in the return on Government Bonds lagged 4 months	X
Rate of change in the return on Government Bonds lagged 5 months	X
Rate of change in the return on Government Bonds lagged 6 months	
Return on U.S. Treasury Bills	
U.S Treasury Bill Index	X
Rate of change in the return on T-bills	
Rate of change in the return on U.S Treasury bills lagged 1 month	X
Rate of change in the return on U.S Treasury bills lagged 2 months	
Rate of change in the return on U.S Treasury bills lagged 3 months	
Rate of change in the return on U.S Treasury bills lagged 4 months	
Rate of change in the return on U.S Treasury bills lagged 5 months	
Rate of change in the return on U.S Treasury bills lagged 6 months	
The CPI	X
The percentage increase in the CPI	
The rate of change in the percentage increase in the CPI	
The rate of change in the percentage increase in the CPI lagged 1 month	
The rate of change in the percentage increase in the CPI lagged 2 months	
The rate of change in the percentage increase in the CPI lagged 3 months	
The rate of change in the percentage increase in the CPI lagged 4 months	
The rate of change in the percentage increase in the CPI lagged 5 months	
The rate of change in the percentage increase in the CPI lagged 6 months	
The S&P Composite Index	X
The return on the S&P Composite Index	

Table 4: **Features for Monthly Experiments**