Available online at www.sciencedirect.com
SCIENCE DIRECT®
Decision Support Systems 37 (2004) 567–581
Decision Support Systems
www.elsevier.com/locate/dsw

Neural network techniques for financial performance prediction: integrating fundamental and technical analysis

Monica Lam*

Management Information Science, College of Business Administration, California State University, Sacramento, 6000 J Street, Sacramento, CA 95819-6088, USA

Available online 12 July 2003

Abstract

This research project investigates the ability of neural networks, specifically, the backpropagation algorithm, to integrate fundamental and technical analysis for financial performance prediction. The predictor attributes include 16 financial statement variables and 11 macroeconomic variables. The rate of return on common shareholders' equity is used as the to-be-predicted variable. Financial data of 364 S&P companies are extracted from the CompuStat database, and macroeconomic variables are extracted from the Citibase database for the study period of 1985–1995. Used as predictors in Experiments 1, 2, and 3 are the 1 year's, the 2 years', and the 3 years' financial data, respectively. Experiment 4 has 3 years' financial data and macroeconomic data as predictors. Moreover, in order to compensate for data noise and parameter misspecification as well as to reveal prediction logic and procedure, we apply a rule extraction technique to convert the connection weights from trained neural networks to symbolic classification rules. The performance of neural networks is compared with the average return from the top one-third returns in the market (maximum benchmark) that approximates the return from perfect information as well as with the overall market average return (minimum benchmark) that approximates the return from highly diversified portfolios. Paired *t* tests are carried out to calculate the statistical significance of mean differences. Experimental results indicate that neural networks using 1 year's or multiple years' financial data consistently and significantly outperform the minimum benchmark, but not the maximum benchmark. As for neural networks with both financial and macroeconomic predictors, they do not outperform the minimum or maximum benchmark in this study. The experimental results also show that the average return of 0.25398 from extracted rules is the only compatible result to the maximum benchmark of 0.2786. Consequentially, we demonstrate rule extraction as a postprocessing technique for improving prediction accuracy and for explaining the prediction logic to financial decision makers.

Keywords: Neural networks; Financial performance; Forecasting; Technical analysis; Fundamental analysis; Postprocessing techniques; Data mining; Rule extraction

1. Introduction

Neural networks have become a popular tool for financial decision making [2,5,16–18,25,33,35,36,43,45]. There are mixed research results concerning the ability of neural networks to predict financial performance. Due to a variety of research design and evaluation criteria, it is difficult to compare the results of different studies [9,27]. Past studies in this area are subject to several problems. First, time horizons for experiments are short. When time hori-

* Tel.: +1-916-278-7037; fax: +1-916-278-6757.
E-mail address: lamsm@csus.edu (M. Lam).

0167-9236/03/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S0167-9236(03)00088-5

zons are short, experimental results may be tampered by situational effect and economic fluctuations. Second, sample sizes are small. When sample sizes are small, experimental results may be biased and cannot be generalized to the future. Third, many studies do not investigate the statistical significance of performance differences. Because variance is a significant factor in investment, ignoring performance variance in the forecasting process is undesirable at least [3,6]. Fourth, the selection of predictor attributes in past studies is based on either fundamental or technical analysis. Fundamental analysts believe that an investment instrument has its intrinsic value that can be derived from the behavior and performance of its company [7,8,14,22,28,29,34,40,42]. The fundamental approach utilizes quantitative tools, mainly the financial ratios compiled from financial statements as well as qualitative indicators, such as management policy, marketing strategy, and product innovation, to determine the value of an investment instrument. Technical analysts, on the other hand, believe that the trends and patterns of an investment instrument's price, volume, breadth, and trading activities reflect most of the relevant market information a decision maker can utilize to determine its value [20,24,37]. Instead of analyzing fundamental information about companies, the technical approach tries to identify turning points, momentum, levels, and directions of an investment instrument, using tools such as charting, relative strength index, moving averages, on balance volume, momentum and rate of change, breadth advance decline indicator, directional movement indicator, and detrended price oscillator. There are divergent opinions about what other trends in the macroeconomic, political, monetary, and societal sentiment spheres should be incorporated in technical analysis [4]. It is the purpose of this research project to address the above four problems for neural network as a data mining tool for financial forecasting. This project applies neural networks to a sample of 364 S&P companies for the period of 1985–1995. We attempt to present a formal study on the complex phenomenon of financial performance using company financial and macroeconomic data as predictor variables, neural networks as the data mining tool, and rate of return on common shareholders' equity as the to-be-predicted variable. Paired *t* tests are adopted to verify the statistical significance of performance differences between neural networks and the market's top performers as well as overall averages.

We believe that neural networks are an excellent tool for forecasting financial performance for the following reasons. First, neural networks are numeric in nature, which is especially suitable for processing numeric data such as financial information and economic indicators. The numeric nature of neural networks is in contrast to symbolic manipulation techniques such as ID3 [26] and AQ [21], which were designed for processing nominal variables. Because numeric data must be converted into nominal values before they can be used as input to symbolic manipulation techniques, there are the problems of losing information, inappropriate data intervals, and different conversion methods leading to different mining results. Neural networks, on the other hand, can accept numeric data directly as input for mining purposes. Second, neural networks do not require any data distribution assumptions for input data. This feature allows neural networks be applicable to a wider collection of problems than statistical techniques such as regression or discriminant analysis. Third, neural networks are an incremental mining technique that permits new data be submitted to a trained neural network in order to update the previous training result. In contrast, many symbolic manipulation and statistical techniques are batch-oriented, which must have both new and old data submitted as a single batch to the model, in order to generate new mining results. For financial applications with new data being available from time to time, neural networks can accommodate new information without reprocessing old information. Fourth, neural networks are model-free estimators. This feature allows interaction effect among variables be captured without explicit model formulations from users [13,45]. Basically, the more hidden layers in a neural network, the more complicated the interaction effect can be modeled.

Although neural networks as a data mining tool have the above merits, they have their fair share of problems. One common difficulty for neural network applications involves the determination of the optimal combination of training parameters including the network architecture (the number of hidden layers and the number of hidden nodes), the learning rate,

the momentum rate, the order of submitting training examples to the network, and the number of training epochs. There are various heuristic rules and common practices for selecting the parameters [44], but the selection process remains as an art rather than a science, and varies from problem to problem. Another common problem in financial applications is noisy data. Because data are collected empirically from different sources, they are subject to corruptions during the retrieval, encoding, transfer, and decoding process. Financial frauds are also potential sources for noisy data in the corporate world. In this research, we take a different approach in addressing the parameter selection and noisy data problem for neural network applications. Instead of addressing the problems before the data mining process, we address the problems after the data mining process. Data preprocessing is a common step for data mining, which can be used to reduce data noise and other irregularities in data sets. As for the parameter selection problem, the traditional approach is to try different parameter combinations on a subset of the available data, with the objective of identifying a satisfactory set of parameters for the entire mining process. Because an exhaustive test of all parameter combinations is impractical, there is no guarantee for an optimal solution. Instead of trying to identify a satisfactory set of parameters before the mining process, we adopt the technique of extracting rules from trained networks after the mining process. Rule extraction, as a postprocessing technique, has the potential ability to generate a more precise and accurate mining result by reducing redundant, conflicting, and erroneous information due to noisy data, input selection problems, and parameter misspecification.

The remaining of this paper is organized as follows. Section 2 reviews the relevant research in the literature. Section 3 presents the experimental design for this study. The experimental results are described in Section 4 and are discussed in Section 5. The last section concludes the paper by summarizing the findings and suggesting some research directions in this area.

2. Literature review

Dropsy [4] adopted neural networks as a nonlinear forecasting tool to predict international equity risk premia. His study used macroeconomic variables to predict equity risk premia in the stock markets of Germany, Japan, United Kingdom, and United States for iterative monthly periods between January 1971 and December 1990. He concluded that both linear and nonlinear forecasts are superior to random walk forecasts, but nonlinear forecasts do not significantly outperform linear forecasts. Kryzanowski et al. [16] adopted the Boltzmann machine for neural network training to classify stock return as negative, neutral, or positive. Their study used financial data from companies, industry, and macroeconomy to predict the classification of stocks in the Canada stock market for monthly periods between 1984 and 1989. They found that for two-state output of positive and negative returns, neural networks correctly classify 72% of all test cases. For three-state output of positive, negative, and neutral, neural networks perform at a lower level than two-state output, but above random guess. Using macroeconomic variables as predictors, Lin and Lin [18] applied neural networks to forecast Dow Jones Industrial average change for monthly periods between August 1991 and March 1992. They reported error rates between 3.8% and 6%, but concluded that their methodology could not yet be used for investment decisions. Applying technical analysis, Trippi and DeSieno [41] investigated the effectiveness of a specific neural network trading system for S&P 500 index futures contracts. Their training period had 1168 days from January 1986 to June 1990, and the test period had 106 days from December 20, 1990 to May 31, 1991. They found that synthesized results from several networks outperform individual networks and the index. White [46] applied a simple neural network to perform time series analysis on the IBM common stock daily returns. His study covered a pre- and postsample period of 500 days. He concluded that the simple network fails to model the nonlinearity of the time series accurately and suggested that networks be evaluated and trained using profit and loss in dollars from generated trades, but not squared forecast error as used in traditional algorithms.

Rule extraction techniques for neural networks are mainly developed to explain learning results of neural networks. Gallant [11] suggested a hybrid system called connectionist expert system that uses a feedfor-

ward neural network to acquire knowledge and perform inference. Human experts provide dependence information about attributes to configure the network before training. Connection weights are represented using only positive or negative integers. The output of a node is clamped into 0, 1, or −1, corresponding to the logical meaning of unknown, true, or false, respectively. The network is trained using the pocket algorithm that is a modification of the perceptron learning method [30]. The rule extraction procedure is to identify contributing nodes which can determine the value of an output node and then use the contributing nodes to form the premise in an IF–THEN rule. This method is limited to networks with output values 0, 1, or −1.

The subset method for rule extraction [10,31] differs from Gallant's method in its utilization of threshold units in neural networks. For each hidden and output node in a neural network, the subset method carries out an exhaustive search to identify all subsets of contributing nodes which have a summation value being greater than the threshold unit of the hidden or output node. Then each subset of contributing nodes is used as the premise in a rule for the hidden or output node. Because the search procedure has to be carried out iteratively to reach the final output nodes in the output layer, the number of rules extracted from the network can grow exponentially as the number of connection weights increases. Heuristics can be applied to limit the search at the expense of the accuracy of extracted rules.

Another method *N*of*M* [39] used groups of weights rather than individual weights as the building blocks for rule premises. Rules generated by *N*of*M* has the format: IF (*N* of the following *M* antecedents are true), THEN (class *x*). *N*of*M* method is suggested as the final step for a knowledge refinement process. The entire process is first to insert domain knowledge into a neural network, then train the network, and, finally, extract rules from the network using the *N*of*M* algorithm. To extract rules, *N*of*M* first collects similar connection weights for each hidden and output node into groups. Then all the weights in the same group are set to the average of the group. Groups of weights which do not significantly affect the state (i.e., on or off) of the hidden or output node are deleted. After the above changes to connection weights, the network has to be retrained in order to reoptimize threshold units for hidden and output nodes. At last, one rule can be formed for each hidden and output node by collecting all contributing nodes into the *M* antecedent group. The value of *N* is determined by the magnitude of the connection weight from the threshold unit to the hidden or output node. *N*of*M* is developed for networks configured by domain knowledge. Some other developments for rule extraction can be found in BRAINNE [32], RX [19], RuleX [1], and VIA [38], which share similar characteristics as the Subset and *N*of*M* methods in one or more aspects.

GLARE [12], as a rule extraction algorithm for neural networks trained by backpropagation, has the following unique and advantageous characteristics. First, GLARE does not require the insertion of domain knowledge into the network before training. Compared with *N*of*M*, GLARE is applicable to classification problems without domain knowledge. Second, GLARE preserves the learning power of partially activated nodes by interpreting both the output of nodes and the connection weights. Because of this feature, GLARE avoids the problem of distorting the output values during the process of converting them into 0, 1, or −1. Third, GLARE extracts only one composite rule for each class, which simplifies the classification procedure for new cases and also explains the classification procedure succinctly. Fourth, GLARE establishes a direct relationship between input attribute nodes and output class nodes, which eliminates the need for coding hidden nodes in classification rules. Compared with other methods, GLARE eliminates the problem of exponential combinations among input, hidden, and output nodes during the rule formulation process. Because of above benefits, this research adopts GLARE as the rule extraction algorithm in the experiments. Appendix A shows the input, processing, and output of the GLARE algorithm.

3. Experimental design

This section describes the data set used, the selection of predictor attributes, the training and test

procedure for neural networks, and the statistical test for experiments. The data set consists of 364 S&P companies for the period of 1985–1995 from the CompuStat database. We extracted annual financial statement data from the Industrial Annual file and annual macroeconomic variables from the Citibase file. Based on recommendations from previous studies [4,15,16,23,28,37], we selected 16 financial statement variables and 11 macroeconomic variables as the predictor attributes. Appendix B shows the variable recommendations from previous studies. The definitions of the variables used in this study are given in Tables 1 and 2. In order to have cases with all the required variables, we eliminate cases with missing data or use proxies as much as we can. Please see Table 1 for proxy definitions. This process gives a sample of 364 companies from the S&P 500 list. The to-be-predicted variable is the rate of return on common shareholders' equity, which is defined as (Net Income − Preferred Dividend)/Common Shareholders' Equity. We classify cases into high, medium, and low rate of return by choosing the top 120 companies into the high, the next 122 companies into the medium, and the last 122 companies into the low categories. From the viewpoint of positive and negative examples, the high category represents positive examples and the medium and low categories represent negative examples. In this study, we are interested in selecting only positive cases.

Table 1
Financial data as predictor attributes

Variable	Definition
v_1	Current Assets/Current Liabilities Proxy: (Cash + Marketable Securities + Net Receivables)/Current Liabilities
v_2	Net Sales/Total Assets
v_3	Net Income/Net Sales
v_4	(Long-Term Debt + Short-Term Debt)/Total Assets
v_5	Total Sources of Fund/Total Uses of Fund Proxy: (Cash + Marketable Securities + Net Receivables)/Current Liabilities
v_6	Research Expense
v_7	Pretax Income/Net Sales
v_8	Current Assets/Common Shareholders' Equity
v_9	Common Shares Traded
v_{10}	Capital Expenditure
v_{11}	Earnings per Share
v_{12}	Dividend per Share
v_{13}	Depreciation Expense
v_{14}	Tax Deferral and Investment Credit
v_{15}	Market Capitalization = Stock Price × Common Shares Outstanding
v_{16}	Relative Strength Index (RSI) = 100 − 100/(1 + RS) RS = Average of *m* periods' up closes/Average of *m* periods' down closes

Table 2
Macroeconomic variables as predictor attributes

Variable	Definition
v_{17}	Federal Budget/Gross Domestic Product
v_{18}	Government Spending/Gross Domestic Product
v_{19}	Money Supply 1
v_{20}	Money Suppy 2
v_{21}	Short-Term Interest Rate
v_{22}	Spread between Short-Term and Long-Term Interest Rate
v_{23}	Consumer Price Index
v_{24}	Trade Balance/Gross Domestic Product
v_{25}	Current Account Balance/Gross Domestic Product
v_{26}	Effective Exchange Rate
v_{27}	Purchase Price of Crude Oil

The experimental procedure is a training and test process. For each complete training and test cycle, we have a pair of training and test set. The training process adjusts the parameters in the network based on the input–output paired examples from the training set. Then in order to validate the training result, the trained network is used to predict the test set. Each training example or test case has an input vector which comprises the 16 financial statement variables (and 11 more macroeconomic variables in some sessions) and an output classification which is high, medium, or low rate of return. Because our interest is in the selection of high-return companies, instead of using the percentage of correctly classified cases (correct classification rate) as a measure for prediction accuracy, we calculate the average return of all the selected high-return companies from the neural network and compare that average with the average of the top one-third (the 33% with the highest returns) companies as well as the average of all companies in the test set. We use the average of all companies as a benchmark to represent the highly recommended investment strategy of diversified and balanced portfolios which approximates the

overall market average. Investors adopting the diversified strategy should be able to get the minimum return of the market average. On the other hand, investors with the perfect information can get the highest returns from the market. We use the average return of the top one-third (33%) companies in the market to represent the benchmark of perfect information or the maximum return for investors. The choice of using the top one-third returns as the maximum benchmark is to follow the practice of classifying the top one-third companies into the high-return category in training sets. If the trained network does not select any company as high-return company, then the medium-return companies will be used, which happened in three training results. If the trained network does not select any company as high- and medium-return companies, then the low-return companies will be used, which happened in one training result.

There are five experiments in this study. Experiment 1 uses 1 year's financial data to predict the classification in the next year. The training set has financial data (input) from year n and classification (output) from year $n+1$. The test set has financial data from year $n+1$ and classification from year $n+2$. Each pair of training and test set requires financial data from 3 years. Starting from the first year (i.e., 1985) of the sample period, we have 85, 86, 87 as the first pair of training and test set, then 86, 87, 88 as the second, etc. The sliding training and test window creates nine sets of training and test samples for Experiment 1. Experiment 1 has 16 financial variables in the input vector. Each training and test set has 364 companies.

Experiment 2 uses 2 years' financial data to predict the classification in the next year. The training set has financial data (input) from years n and $n+1$ and classification (output) from year $n+2$. The test set has financial data from years $n+1$ and $n+2$ and classification from year $n+3$. Each pair of training and test set requires financial data from 4 years. Starting from the first year (i.e., 1985) of the sample period, we have 85, 86, 87, 88 as the first pair of training and test set, then 86, 87, 88, 89 as the second, etc. The sliding training and test window creates eight sets of training and test samples for Experiment 2. Experiment 2 has 32 financial variables (16 from year n and another 16 from year $n+1$) in the input vector. Each training and test set has 364 companies.

Experiment 3 uses 3 years' financial data to predict the classification in the next year. The training set has financial data (input) from years n, $n+1$, and $n+2$ and classification (output) from year $n+3$. The test set has financial data from years $n+1$, $n+2$, $n+3$ and classification from year $n+4$. Each pair of training and test set requires financial data from 5 years. Starting from the first year (i.e., 1985) of the sample period, we have 85, 86, 87, 88, 89 as the first training and test set, then 86, 87, 88, 89, 90 as the second training and test set, etc. The sliding training and test window creates seven sets of training and test samples for Experiment 3. Experiment 3 has 48 financial variables (16 from year n, 16 from year $n+1$, and 16 from year $n+2$) in the input vector. Each training and test set has 364 companies.

Experiment 4 has both financial and macroeconomic data as predictors. It uses 3 years' financial and macroeconomic data to predict the classification in the next year. Because the macroeconomic variables are the same for all companies in a certain year, in order to vary the input macroeconomic variables against the output return, we match 1 year's financial and macroeconomic data with the next year's rate of return. In other words, the training set has input–output patterns from three different periods including the pair of input from year n and output from year $n+1$, the pair of input from year $n+1$ and output from year $n+2$, and the pair of input from year $n+2$ and output from year $n+3$. The above design yields 27 variables (16 from financial data, and 11 from macroeconomic data) in the input vector and 1092 cases (364 cases $\times$ 3 years) in the training set. The test set has financial and macroeconomic data as input from year $n+3$ and output classification from year $n+4$. The test set has 364 cases and 27 variables in the input vector. The sliding training and test window creates seven samples for Experiment 4.

The training algorithm is backpropagation for feedforward neural network. Backpropagation training has the major parameters of network architecture (the number of hidden layers and the number of hidden nodes in each hidden layer), learning rate, momentum rate, and the number of training epochs. Based on our past experience with backpropagation and recommendations from the literature, we selec-

tively evaluated some parameter values in preliminary training. According to the result from preliminary training, we decided to use 1, 2, and 3 hidden layers, 10 hidden nodes in hidden layer 1, 7 hidden nodes in hidden layer 2, and 3 hidden nodes in hidden layer 3 for Experiments 1 to 4, respectively. All four experiments have 0.5, 0.1, and 1000 as the learning rate, momentum rate, and training epochs, respectively. The training program is written in C language. All experiments were carried out in a desktop computer.

Experiment 5 is designed to compare the performance of neural networks per se and of the rules extracted from neural networks using the GLARE algorithm. GLARE allows only 1 hidden layer in the network and only dummy variables in the input layer. The continuous input variables are bucketed into five categories (very low, low, medium, high, and very high) with equal distance between any two adjacent categories. The neural network for this experiment has 80 (16 variables × 5 5 categories) input nodes, 1 hidden layer, and 30 hidden nodes. For each pair of the nine training and test sets in Experiment 1, we performed the following steps:

(1) The same procedure as Experiment 1.
(2) Apply conventional backpropagation to the training set with dummy input variables. Train the network with 1000 epochs. Apply GLARE to the trained network to extract classification rules. Use extracted rules to predict the test set with nominal input values. Calculate the average return of all the high-return companies as selected by the rules.
(3) This step is the same as Step (2) above except that the number of training epochs is reduced to 5.

For the GLARE algorithm, we set the parameter NR (number of input attribute values a case must match in order to be classified as a certain class) to 10, and the parameter NW (number of weights used in building the reduced rankings for all input nodes to a hidden node) to 30.

Many past studies on neural networks do not evaluate the statistical significance of the experimental results. A simple comparison between some yardstick measures and neural networks can be misleading because it does not take into consideration the variance and distribution. In this study, we adopt the paired t test to evaluate the statistical significance of the mean differences. Experiments 1 and 5 have 18 pairs of means from 1987 to 1995 with the year indicating the output year for the test set. Experiment 2 has 16 pairs of means from 1988 to 1995. Experiments 3 and 4 have 14 pairs of means from 1989 to 1995. Half of the mean comparisons are between neural network and the overall market average minimum benchmark, and half of the mean comparisons are between neural network and the top one-third average maximum benchmark.

4. Experimental results

The average rates of return for high-return companies from neural networks as well as paired t-test results are presented in Tables 3–8. Table 3 reports

Table 3
Rate of return and paired t-test results for using 1 year's financial data to predict the next year (Experiment 1)

Predicted year	All (min)	Top 33% (max)	nn1y1h	nn1y2h	nn1y3h
1987	0.13041	0.24367	0.13281	0.12675	0.10733
1988	0.10535	0.26027	0.03268	0.18497	0.10535
1989	0.14292	0.29383	0.13529	0.14182	0.12121
1990	0.15247	0.33445	0.20769	0.20191	0.30084
1991	−0.00144	0.23918	0.20036	−0.08882	−0.09401
1992	0.05856	0.26384	0.14703	0.15298	0.19015
1993	0.08487	0.24856	0.16121	0.16487	0.17458
1994	0.15980	0.32967	0.23751	0.22779	0.22575
1995	0.14363	0.29397	0.24482	0.23544	0.19183
Average	0.10851	0.27860	0.16660	0.14974	0.14700
p Value for nn vs. min			0.028	0.038	0.092
p Value for nn vs. max			0.00018	0.00065	0.00096

nn: neural network.
1h: 1 hidden layer; 10 hidden nodes in hidden layer 1.
2h: 2 hidden layers; 10, 7 hidden nodes in hidden layers 1, 2.
3h: 3 hidden layers; 10, 7, 3 hidden nodes in hidden layers 1, 2, 3.
1y: 1 year's financial data as predictors.

the average returns and *t*-test results for Experiment 1, which uses 1 year's financial data to predict the return in the next year. The average return for all companies for the period of 1987–1995 is 0.10851, and the average return for the top one-third companies is 0.2786. The average returns from neural networks with 1 hidden layer, 2 hidden layers, and 3 hidden layers are 0.1666, 0.14974, and 0.147, respectively, which are all higher than the average for all companies (minimum), but not the average for the top one-third (maximum). The *p* values for neural network vs. minimum ("nn vs. min") paired *t* test for 1 hidden layer, 2 hidden layers, and 3 hidden layers are 0.028, 0.038, and 0.092, respectively. Using 0.05 as the level of significance, neural networks with 1 and 2 hidden layers achieve significantly higher returns than the average of all companies.

Table 4 shows the average returns and *t*-test results for Experiment 2, which predicts the return in the third year using data from the previous 2 years. Similar to Experiment 1, the results from 1 (0.17608), 2 (0.15219), and 3 (0.18612) hidden layers are all higher than the market average (0.10577), but not the average of the top one-third (0.28297). The paired *t* test confirms the statistical significance of the average differences between the 1 hidden layer network ($p = 0.04$) and the market average as well as the 3 hidden layers ($p = 0.024$) and market average. In other words, the 1 hidden layer and 3 hidden layers networks significantly outperform the market average at the significant level of 0.05. On the other hand, none of the neural network performs significantly better than the top one-third benchmark.

In Table 5, we present the average returns and *t*-test results for Experiment 3 that predicts the next year's return based on the previous 3 years' financial data. All the average returns from neural networks, i.e., 0.18135 from 1 hidden layer, 0.20076 from 2 hidden layers, and 0.18371 from 3 hidden layers, are all higher than the market average return 0.10583 during the same period. This is the experiment that we have all neural networks with different number of hidden layers achieve significantly higher average returns than the market average at

Table 4
Rate of return and paired *t*-test results for using 2 years' financial data to predict the next year (Experiment 2)

Predicted year	All (min)	Top 33% (max)	nn2y1h	nn2y2h	nn2y3h
1988	0.10535	0.26027	0.20211	0.05857	0.21259
1989	0.14292	0.29383	0.09424	0.24513	0.23977
1990	0.15247	0.33445	0.10518	0.13705	0.30557
1991	−0.00144	0.23918	0.20194	−0.06128	0.21283
1992	0.05856	0.26384	0.03189	0.23096	0.20628
1993	0.08487	0.24856	0.20984	0.20451	0.0433
1994	0.1598	0.32967	0.29912	0.28073	0.15037
1995	0.14363	0.29397	0.26436	0.12184	0.11826
Average	0.10577	0.28297	0.17608	0.15219	0.18612
p Value for nn vs. min			0.040	0.097	0.024
p Value for nn vs. max			0.00764	0.00390	0.00427

nn: neural network.
1h: 1 hidden layer; 10 hidden nodes in hidden layer 1.
2h: 2 hidden layers; 10, 7 hidden nodes in hidden layers 1, 2.
3h: 3 hidden layers; 10, 7, 3 hidden nodes in hidden layers 1, 2, 3.
2y: 2 years' financial data as predictors.

Table 5
Rate of return and paired *t*-test results for using 3 years' financial data to predict the next year (Experiment 3)

Predicted year	All (min)	Top 33% (max)	nn3y1h	nn3y2h	nn3y3h
1989	0.14292	0.29383	0.13736	0.181	0.07283
1990	0.15247	0.33445	0.06749	0.18811	0.19024
1991	−0.00144	0.23918	0.20601	0.17359	0.19317
1992	0.05856	0.26384	0.19454	0.21593	0.15114
1993	0.08487	0.24856	0.15491	0.18299	0.14677
1994	0.1598	0.32967	0.3082	0.26316	0.29437
1995	0.14363	0.29397	0.20091	0.20056	0.23746
Average	0.10583	0.28621	0.18135	0.20076	0.18371
p Value for nn vs. min			0.045	0.002	0.024
p Value for nn vs. max			0.00826	0.00030	0.00303

nn: neural network.
1h: 1 hidden layer; 10 hidden nodes in hidden layer 1.
2h: 2 hidden layers; 10, 7 hidden nodes in hidden layers 1, 2.
3h: 3 hidden layers; 10, 7, 3 hidden nodes in hidden layers 1, 2, 3.
3y: 3 years' financial data as predictors.

the significance level of 0.05. The p values for 1, 2, and 3 hidden layers are 0.045, 0.002, and 0.024, respectively. However, the comparisons between neural networks and the top one-third averages show that neural networks cannot outperform the maximum benchmark.

Experiment 4 have both financial and macroeconomic variables as the predictors, and its results are presented in Table 6. Although the average returns from 1 (0.11532), 2 (0.11497), and 3 (0.13974) hidden layers are all higher than the average return from all companies (0.10583), none of the networks significantly outperforms the market average. The networks with 1 hidden layer, 2 hidden layer, and 3 hidden layers have the p values of 0.409, 0.418, and 0.152, respectively. Furthermore, neural networks do not outperform the maximum benchmark, which has 0.28621 as the average for the top one-third returns between 1989 and 1995.

Experiment 5 applies both conventional backpropagation and rule extraction to predict the next year's return based on financial data. Table 7 reports the results from Experiment 5 including the average returns from conventional backpropagation training (NN), extracted rules for 1000 training epochs (Rule1000), and extracted rules for 5 training epochs (Rule5). The results from the first three columns in Table 7 are the same as the first three columns in Table 3. The NN (0.1666), Rule1000 (0.19379), and Rule5 (0.25398) all achieve higher average returns than the average of all companies (0.10851), but lower than the top one-third companies (0.2786). In Table 8, we present the p values for paired t test of mean differences from Table 7. Using 0.05 as the level of significance, we find NN, Rule1000, and Rule5 are all significantly higher than the market average. In addition, Rule5 achieves significantly higher returns than Rule1000. While Rule5 significantly outperforms NN, Rule1000 does not. Comparing the top one-third average (maximum benchmark) with others, Rule5 can be considered as equivalent to the market's

Table 6
Rate of return and paired t-test results for using 3 years' financial and macroeconomic data to predict the next year (Experiment 4)

Predicted year	All (min)	Top 33% (max)	me3y1h	me3y2h	me3y3h
1989	0.14292	0.29383	0.01062	0.18043	0.13041
1990	0.15247	0.33445	0.19282	0.24169	0.10535
1991	−0.00144	0.23918	−0.13364	−0.16635	0.14648
1992	0.05856	0.26384	0.16295	0.20487	0.1914
1993	0.08487	0.24856	0.15314	−0.02881	0.04018
1994	0.1598	0.32967	0.27813	0.23231	0.20414
1995	0.14363	0.29397	0.14322	0.14068	0.16018
Average	0.10583	0.28621	0.11532	0.11497	0.13974
p Value for me vs. min			0.409	0.418	0.152
p Value for me vs. max			0.00386	0.00556	0.00026

1h: 1 hidden layer; 10 hidden nodes in hidden layer 1.
2h: 2 hidden layers; 10, 7 hidden nodes in hidden layers 1, 2.
3h: 3 hidden layers; 10, 7, 3 hidden nodes in hidden layers 1, 2, 3.
me3y: neural network using 3 years' financial and macroeconomic data as predictors.

Table 7
Rate of return from neural network and rule extraction (Experiment 5)

Predicted year	All (min)	Top 33% (max)	NN	Rule1000	Rule5
1987	0.13041	0.24367	0.13281	0.13843	0.23654
1988	0.10535	0.26027	0.03268	0.24203	0.25158
1989	0.14292	0.29383	0.13529	0.14281	0.28616
1990	0.15247	0.33445	0.20769	0.1274	0.24372
1991	−0.00144	0.23918	0.20036	0.2325	0.23006
1992	0.05856	0.26384	0.14703	0.26326	0.25975
1993	0.08487	0.24856	0.16121	0.11837	0.19832
1994	0.1598	0.32967	0.23751	0.33356	0.33852
1995	0.14363	0.29397	0.24482	0.14574	0.24119
Average	0.10851	0.27860	0.1666	0.19379	0.25398

NN: Conventional backpropagation training.
Rule1000: Training has 1000 epochs before rule extraction.
Rule5: Training has 5 epochs before rule extraction.

Table 8
p Value for one-tailed paired t test (data from Table 7)

	All (min)	Top 33% (max)	NN	Rule1000
NN	0.028	0.00018	–	–
Rule1000	0.024	0.00653	0.229	–
Rule5	0	0.02632	0.003	0.011

NN: Conventional backpropagation training.

top performers if the significant level is set at 0.01. To show an example of extracted rules, the following is the extracted rule from Rule5 for predicting 1994 return:

IF v_1 is low or high and
v_2 very low or low or high and
v_3 is very low or medium or high and
v_4 is very low or low or high and
v_5 is very low or low or medium or high or very high and
v_6 is very low or high or very high and
v_7 is very low, low, or medium and
v_8 is very low, low, or medium and
v_9 is very low or very high and
v_{10} is medium and
v_{11} is low or medium or very high and
v_{12} is very low or high or very high and
v_{13} is very low or medium or very high and
v_{14} is low or medium or very high and
v_{15} is very low or high or very high and
v_{16} is high or very high

THEN the company is a high return company.

5. Discussions of experimental results

This section discusses the experimental results for neural networks as a data mining technique for predicting company return, the sensitivity of neural network architecture on prediction performance, the effect of integrating fundamental and technical analysis via neural networks, the validity of extracted rules from neural networks, and the contribution of this study towards the next-generation decision support systems.

5.1. Neural network as a data mining technique for predicting company return

Tables 3–6 show that all average returns from neural networks are higher than the minimum benchmark and 7 of the 12 higher returns are statistically significant. On the other hand, none of the neural networks can significantly outperform the maximum benchmark. The results show that neural network can perform better than the minimum, but not yet at the maximum return level. Using the minimum benchmark, neural networks predict consistently well under different market situations. If we look at Table 3, for prediction using 1 year's financial data, in the severe market downturn of 1991, the best return from neural networks is 0.20036 (from 1 hidden layer), which is 140 times of the market average (-0.00144). In other words, under most situations, neural network techniques can significantly outperform the highly recommended strategy of diversified and balanced portfolios for investment. In this application, neural networks' performance is attributed to the data preprocessing effect from variable selection, the integrative learning capabilities of neural networks, and the postprocessing power from rule extraction.

5.2. Effect of network architecture on prediction performance

The network architecture design for backpropagation training remains as an art rather than a science. Complex problems can be incrementally better modeled via additional hidden layers, but the incremental improvement comes with an associated cost in terms

of training time, interpretation problems, and data overfitting. As we can see from the experimental results, Table 3 (16 input variables) has the highest return from 1 hidden layer, Table 4 (32 input variables) from 3 hidden layers, Table 5 (48 input variables) from 2 hidden layers, and Table 6 (27 input variables) from 3 hidden layers. Experiment 1 in Table 3, the simplest in this study, requires only 1 hidden layer to generate the best result. On the other hand, although the 3 hidden layers training produce two of the highest returns, the returns are not exceedingly higher than the 1 or 2 hidden layers. The question seems to be how much complication to the model we are willing to take in order to increase a marginal unit of prediction accuracy. A systematic approach of identifying the point equating the marginal benefit with the marginal cost requires further research efforts. The experimental results also provide some evidence for the resounding principle of minimum description length, which states that everything being equal, the less complicated theory has better prediction power.

5.3. Integrating fundamental and technical analysis via neural networks

This study integrates the fundamental and technical analysis during the neural network training process via three different techniques. First, by increasing the predictor variables from 1 to 3 years in the input vector (from Experiments 1 to 3), we simulate the time series effect for analyses. Using neural networks, trend analysis and specific company information can be combined in a single model. For the next generation of decision support systems, in order to provide a consolidated recommendation to decision makers and to incorporate merits from different analyses, it is important to have integrative tools. It is shown in this project that neural networks, not only being an established and validated data mining tool, but also can serve the integrative role for different analytical approaches. Tables 3–5 represent the returns from gradually increasing the technical effect by incorporating the changing patterns of fundamental financial variables. The grand average of neural network returns is 0.1544 in Table 3 (average of 0.1666, 0.14974, and 0.14700), 0.1715 in Table 4 (average of 0.17608, 0.15219, and 0.18612), and 0.1886 in Table 5 (average of 0.18135, 0.20076, and 0.18371). The gradual increase in the grand average return shows that integrating technical analysis with fundamental analysis can improve the return level. The integration effect seems to work even better for recession periods. If we look at the returns for year 1991 in Tables 3–5, neural networks with 1 year's financial data yield the average return 0.00584 (average of 0.20036, −0.08882, and −0.09401), neural networks with 2 years' financial data yield the average return 0.11783 (average of 0.20194, −0.06128, and 0.21283), and neural networks with 3 years' financial data yield the average return 0.1909 (average of 0.20601, 0.17359, and 0.19317). The integration merit seems to manifest the most during market downturns.

The second integration technique is to have variable 16 in the input vector for neural network training. Variable 16 is the relative strength index measuring the relative stability of an investment instrument's market price, which is a tool for technical analysis. The third integration technique is to include macroeconomic variables in the input vector (i.e., Experiment 4). The average returns from Table 6 indicate that the addition of macroeconomic variables does not improve the network performance. Although the average returns from neural networks are higher than the average of all companies, none of them is statistically significant. The fact that the choice of the specific macroeconomic variables in this experiment does not add value to the prediction accuracy echoes the difficult issue on the selection of extraneous trends to facilitate the analysis. Results from Experiment 4 seems to suggest that adding unnecessary variables to a problem will actually deteriorate the network performance.

5.4. Validity of extracted rules on prediction

Table 7 presents the experimental results for rule extraction. The results are favorable to the prediction power of extracted rules from neural networks. Both Rule1000 (with 1000 training epochs) and Rule5 (with 5 training epochs) achieve higher average returns than neural networks per se. Rule5 achieves an astounding 0.25398 average return, which is 53% better than the average return from neural networks (0.16666), and also the best average return among all the experiments in this study. It is interesting to note

that Rule5 performs better than Rule1000. It seems that it takes only a few training cycles for the GLARE algorithm to capture the most important information for return prediction. The performance difference between Rule5 and Rule1000 may be due to the overlearning of Rule1000 in that training session. The average return from Rule5 (0.25398) is the only compatible performance (significance level of 0.01) to the maximum benchmark (0.2786) in this study, which shows the potential of extracted rules from neural networks to approximate the best performers in the market. A review of the extracted rule (in the Experimental Results section) sheds light on some important information for return prediction. For example, variable 5 (total sources of fund/total uses of fund) is not relevant for high return because it can be in any category; variable 9 (common shares traded) should be either very low or very high; variable 10 (capital expenditure), being the most restricted, must be medium; and variable 16 (relative strength index) must be either high or very high. Some extracted rules can be difficult to understand as the process of rule extraction utilizes more of local membership functions than both local and monotonic membership functions. However, attribute information from rule extraction is important for decision makers to feel comfortable with a recommendation as well as to contemplate, investigate, and react to the underlying prediction logic. Data mining results are beneficial only when they are understandable and usable. Moreover, one important feature of decision support systems is its ability to interact with decision makers to arrive at a decision. In order to interact, the results from data mining must be comprehensible to decision makers. Rule extraction can decipher the connection weights from a trained network by converting the numeric values into symbolic rules. The performance from rule extraction also demonstrates the effectiveness of rule extraction as a postprocessing technique to neural network training. When preprocessing is impractical, the data is noisy, and the training is subject to parameter misspecification, we should consider postprocessing techniques to compress and correct the data mining results.

5.5. Next-generation decision support systems

Leigh et al. [17] described a feature of the next-generation decision support system as the ability to integrate different decision making and forecasting methods via powerful hardware and software components. As models or analytical approaches tend to focus on certain aspects of the reality, in order to capture the complex interrelationship among factors affecting financial activities in this global economy, hybrid analytical tools are critical for decision support systems to deliver results. This study presents and validates two features of neural networks contributing to the next-generation decision support systems in the financial world. The first feature points to the ability of neural networks to combine fundamental and technical analysis in the same model. The second feature involves the postprocessing power of rule extraction that can improve prediction performance and reveal the prediction logic and procedure.

6. Conclusion

In this study, we designed a series of experiments to test the prediction power of neural networks on financial performance. The experimental results support the ability of neural network to significantly outperform the minimum benchmark based on a highly diversified investment strategy. In addition, the technique of incorporating previous years' financial data in the input vector for neural network training can significantly increase the return level, which demonstrates the benefits of integrating fundamental analysis with technical analysis via neural network training. Moreover, the integration works especially well when the economy is in recession. As for the prediction efficacy of macroeconomic variables for financial performance, we did not find any confirmative evidence in this problem setting. The experimental results show that financial statement variables and macroeconomic variables together cannot generate significantly higher returns than the average index. We also extracted rules from trained neural networks and found that the extracted rules can predict significantly better than neural networks per se, can perform as accurately as the maximum benchmark, and can reveal the prediction logic to financial decision makers.

Among all the research studies on the abilities of neural networks to select high-return companies, this research provides some strong evidence on the benefits of neural networks. Will the promise hold up in

the coming 10 years? That remains to be confirmed by another research study. Another research direction in this area is to construct or manipulate the neural network to emphasize the relative importance of learning the positive cases correctly. One method can be changing the error function or the learning procedure to provide positive reinforcement for classifying the positive cases correctly. This is basically the procedure of recognizing differential costs for making different mistakes. As the focus of the problem is on selecting high-return companies, the relative cost of misclassifying a low-return company as high return can be more damaging than committing the reverse error.

Acknowledgements

This research project was supported by a research fellowship grant awarded by the California State University System.

Appendix A. The input/processing/output of GLARE [12]

A.1. Input

- Continuous attributes have to be centered and converted into nominal, then boolean attributes.
- Connection weights from a fully connected feedforward network trained by backpropagation.

A.2. Rule extraction process

For each output node *m* (each class in the data set), do the following:

1. Create Rank1_n which are rankings of all input nodes X_{ij} (category *j* of input attribute *i*, Σi is the total input attributes, Σj is the total category values for all input attributes) to hidden node *n* ($n = 1, 2, \ldots N$), based on the magnitude of connection weights from input nodes to hidden node *n* in descending order.
2. Create R1_n which are reduced Rank1_n from Step 1, based on the parameter a, which is the number of selected weights between input and hidden layer.
3. Calculate Index(H_{mn}) which are importance indexes for hidden node *n* to output node *m*.
4. Create Rank2_m which is the ranking of importance of hidden nodes to output node *m* in descending order, based on Index(H_{mn}) from Step 3.
5. Create *M*, which is an $N \times a$ matrix consisting of R1_n from Step 2, and adjusted based on Rank2_m from Step 4. Input nodes on the top rows and left columns in the *M* are more influential for determining the output of the output node *m*.
6. Create *R*, which is an $\Sigma i \times \Sigma j$ matrix consisting of elements -1, 1, or 0 based on *M* from Step 5.
7. Create the classification rule for node *m* based on *R* from Step 6.

A.3. Output rules

- Format:

```
If    attribute 0 = category 0 or 1   AND
      attribute 1 = category 2 or 3   AND ...
Then  class = 0.
```

- One composite symbolic rule for each class, and each rule is a conjunctive of disjunctives.
- The length of the conjunctive is the number of input attributes, and disjunctive *i* contains no more than the total categories in attribute *i*.

Appendix B. Suggested variables for analysis of financial performance from the literature

Dropsy [4], for international equity risk premia: Excess stock returns, ratio of government spending to GDP, growth rate of money supply M2, short-term interest rate, spread between long-term government bond yields and short-term interest rates, consumer inflation rate, ratio of trade balance to GDP, nominal effective rate of depreciation of the domestic currency, real oil price inflation rate.

Kiang et al. [15], for stock price: Shares outstanding, price ratio, return on equity, net current asset ratio, price-to-book ratio, relative strength, change in quarterly earnings, market capitalization.

Kryzanowski et al. [16], for stock return: Industrial production, gross domestic product, McLeod Young Weir corporate long bond index, 90-day Treasury bill

rate, Canada government long bond index, consumer price index, Montreal Exchange 25 index, gross margin, net profit margin, total asset turnover, fixed asset turnover, return on total assets, return on equity, debt ratio, debt-to-equity ratio, interest coverage ratio, long-term debt ratio, current ratio, quick ratio, accounts receivable turnover, accounts payable turnover.

Lin and Lin [18], for Dow average change: Consumer price index, price of crude oil, inflation rate, Dow change, unemployment rate, prime interest rate, consumer price index, political climate, time period.

Reinganum [28], for stock price: Price-to-book ratio, five-year growth rate, quarterly earnings, pretax profit margins, outstanding common shares, relative strength, selling price, O'Neil rating.

Thomsett [37], for stock price: Fundamental information, insider trends, industry-specific trends and changes, charting patterns of the individual company, price–earnings ratio, volatility, trading volume, major news of changes in the company, economic news affecting the company.

References

[1] R. Andrews, J. Diederich, A.B. Tickle, Rule extraction from a constrained error back propagation MLP, Proceedings of Fifth Australian Conference on Neural Networks, 1994, pp. 9–12.

[2] W. Cheng, B.W. McClain, C. Kelly, Artificial neural networks make their mark as a powerful tool for investors, Review of Business (1997 Summer) 4–9.

[3] A.K. Dixit, R.S. Pindyck, Investment Under Uncertainty, Princeton Univ. Press, New Jersey, 1994.

[4] V. Dropsy, Do macroeconomic factors help in predicting international equity risk premia? Journal of Applied Business Research 12 (3) (1996) 120–132.

[5] S. Dutta, S. Shekhar, Bond-rating: a non-conservative application of neural networks, Proceedings of the IEEE International Conference on Neural Networks 2 (1988) 443–450.

[6] E.J. Elton, M.J. Gruber, Modern Portfolio Theory and Investment Analysis, Wiley, New York, 1995.

[7] K.L. Fisher, Super Stocks, Dow Jones-Irwin, Illinois, 1984.

[8] J.A. Frankel, Financial Markets and Monetary Policy, MIT Press, Massachusetts, 1995.

[9] J.D. Freedman, Behind the smoke and mirrors: gauging the integrity of investment simulations, Financial Analysts Journal 48 (6) (1992) 26–31.

[10] L.M. Fu, Rule learning by searching on adapted nets, Proceedings of the Ninth National Conference on Artificial Intelligence, 1991, pp. 500–595.

[11] S.I. Gallant, Connectionist expert systems, Communications of the ACM 31 (1988) 152–169.

[12] A. Gupta, S. Park, M. Lam, Generalized analytic rule extraction for feedforward neural networks, IEEE Transactions on Knowledge and Data Engineering 11 (6) (1999) 985–991.

[13] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (1989) 359–366.

[14] M.C. Jensen, Risk: the pricing of capital assets, and the investment of portfolios, Journal of Business 42 (2) (1969) 167–185.

[15] M.Y. Kiang, R. Chi, K.Y. Tam, DKAS: a distributed knowledge acquisition system in a DSS, Journal of Management Information Systems 9 (4) (1993) 59–82.

[16] L. Kryzanowski, M. Galler, D. Wright, Using artificial neural network to pick stocks, Financial Analysts Journal (1993 June/August) 21–27.

[17] W. Leigh, R. Purvis, J.M. Ragusa, Forecasting the NYSE composite index with technical analysis, pattern recognizer, neural network, and genetic algorithm: a case study in romantic decision support, Decision Support Systems 32 (2002) 361–377.

[18] F.C. Lin, M. Lin, Analysis of financial data using neural nets, AI Expert (1993 February) 36–41.

[19] H. Lu, R. Setiono, H. Liu, Effective data mining using neural networks, IEEE Transactions on Knowledge and Data Engineering 8 (2) (1996) 957–961.

[20] T. Meyers, The Technical Analysis Course: A Winning Program for Stock and Future Traders and Investors, Probus, Chicago, 1989.

[21] R.S. Michalski, A theory and methodology of inductive learning, Artificial Intelligence 20 (1983) 111–116.

[22] W.J. O'Neil, How to Make Money in Stocks? McGraw-Hill, New York, 1995.

[23] J.A. Ou, The information content of non-earnings accounting numbers as earnings predictors, Journal of Accounting Research 28 (1990) 144–163.

[24] M.J. Pring, Technical Analysis Explained, McGraw-Hill, New York, New York, 1980.

[25] M. Qi, Nonlinear predictability of stock returns using financial and economic variables, Journal of Business and Economic Statistics 17 (4) (1999) 419–429.

[26] J.R. Quinlan, Induction of decision trees, Machine Learning 1 (1986) 81–106.

[27] J. Racine, On the nonlinear predictability of stock returns using financial and economic variables, Journal of Business and Economic Statistics 19 (3) (2001) 380–382.

[28] M.R. Reinganum, The anatomy of a stock market winner, Financial Analysts Journal (1988) 16–28.

[29] J. Ritchie, Fundamental Analysis: A Back-to-the-Basics Investment Guide to Selecting Quality Stocks, Probus, Chicago, 1989.

[30] F. Rosenblatt, Principles of Neurodynamics, Spartan, New York, 1962.

[31] K. Saito, R. Nakano, Medical diagnostic expert system based on PDP model, Proceedings of IEEE International Conference on Neural networks 1 (1988) 255–262.

[32] S. Sestito, T.S. Dillon, Automated Knowledge Acquisition, Prentice Hall, Upper Saddle River, New Jersey, 1994.

[33] J.R. Shah, M.B. Murtaza, A neural network based clustering

procedure for bankruptcy prediction, American Business Review 18 (2) (2000) 80–86.
[34] W.F. Sharpe, Mutual fund performance, Journal of Business 39 (1966) 119–138.
[35] K.Y. Tam, M.L. Kiang, Managerial applications of neural networks: the case of bank failure predictions, Management Science 38 (1992) 926–947.
[36] Z. Tang, C. de Almeida, P. Fishwick, Time series forecasting using neural networks vs. Box–Jenkins methodology, Proceedings of the First Workshop on Neural Networks: Academic/Industrial/NASA/Defense, 1990, pp. 95–100.
[37] M.C. Thomsett, Mastering Technical Analysis, A Kaplan Professional, Chicago, 1999.
[38] S.B. Thrun, Extracting probably correct rules from artificial neural networks, Technical Report IAI-TR-93-5, Institute for Informatik III Universitat Bonn, Germany, 1994.
[39] G.G. Towell, J.W. Shavlik, Interpretation of artificial neural networks: mapping knowledge-based neural networks into rules, in: J.E. Moody, S.J. Hanson, R.P. Lippmann (Eds.), Advances in Neural information Processing Systems, vol. 4, Morgan Kaufmann, San Mateo, California, 1992.
[40] J.L. Treynor, How to rate management of investment funds? Harvard Business Review 43 (1) (1965) 63–75.
[41] R. Trippi, D. DeSieno, Trading equity index futures with a neural network, Journal of Portfolio Management (1992 Fall) 27–33.
[42] L. Tso, Techniques for Uncovering Hidden-Value Stocks, Frederick Fell, New York, 1965.
[43] S. Walczak, An empirical analysis of data requirements for financial forecasting with neural networks, Journal of Management Information Systems 17 (4) (2001) 203–222.
[44] S. Walczak, N. Cerpa, Heuristic principles for the design of artificial neural networks, Information and Software Technology 41 (1999) 107–117.
[45] H. White, Consequences and detection of mis-specified nonlinear regression models, Journal of the American Statistical Association (1981 June) 419–433.
[46] H. White, Economic prediction using neural networks: the case of IBM daily stock returns, Proceedings of the Second IEEE International Conference on Neural Network, 1988, pp. II451–II458.

Monica Lam, a 1994 PhD graduate from the University of Wisconsin-Madison, is a professor in the Management Information Science Department of California State University at Sacramento. She has published in Annals of Operations Research, Neural Networks, IEEE Transactions on Knowledge and Data Engineering, Information Systems Management, European Journal of Operational Research, Journal of Operational Research Society, and International Journal of Business and Economics. She has recently coauthored an Accounting Information Case Book by Prentice Hall and is cowriting a second book on Enterprise Accounting Models for the same publisher. She holds a copyright on a rule extraction system for the backpropagation learning algorithm. She has also performed consulting and internship activities with Access Health, Wells Fargo Bank, and the Department of Parks and Recreation in California. She can be reached at lamsm@csus.edu and her Web address is http://www.csus.edu/indiv/l/lamsm.