Back Propagation as a Test of the Efficient Markets Hypothesis

George Tsibouris

Department of Economics
University of Wisconsin
Madison, WI 53706

Matthew Zeidenberg

Department of Computer Sciences
University of Wisconsin
Madison, WI 53706

Abstract

This paper presents some research on the application of artificial neural networks to economic modeling. The Efficient Markets Hypothesis (EMH) states that at any time, the price of a security fully captures all known information about that stock, so the price behaves like a random walk in time, except when there are changes in information. We test whether a nonlinear statistical method, error back propagation, can do better than chance in forecasting stock trends.

An error back propagation model is estimated at different levels of time aggregation (daily and monthly) on stock price and stock index returns. This paper brings forth some new and encouraging results on the ability of neural network models to predict the direction of stock price movements and to account for some of the nonlinearities found in stock return data.

1 Introduction

Specification of financial models has traditionally been done in partial equilibrium. Both capital asset pricing models (CAPM) and arbitrage pricing theory (APT) derived models attempt to describe asset price behavior based on optimization in partial equilibrium. Such models have been very useful in expanding our understanding of stock price behavior. Nevertheless, many empirical financial anomalies have remained unexplainable. It is possible that this may be due to the partial equilibrium nature of these models. Attempting to model financial markets in a general equilibrium framework still remains analytically intractable.

Because of their inductive nature, neural networks, like other statistical models, can bypass the step of theory formulation altogether. They can infer complex non-linear relationships between input and output variables. There are three types of neural networks that suggest themselves for the modeling of time- series data: feedforward back propagation networks, recurrent back propagation networks, and networks based on the temporal difference method of Sutton. We explore the use of feedforward back propagation networks in the case of stock market time-series data. Neural network models are fitted to daily stock prices, daily and monthly stock market indices. The estimated parameters are used to obtain one-step-ahead forecasts.

Section 2 explains neural network models and introduces the reader to some of the terminology that is commonly used. Section 3 details the econometric methodology. The estimation results are presented and discussed in Section 4. A filter rule trading strategy based on the back propagation algorithm is devised in Section 5. This strategy is compared to the buy-and-hold rule. Possible directions for further study are outlined in Section 6. The last section, Section 7, contains some concluding remarks.

2 Artificial Neural Networks

A neural network is a computational model that is a directed graph composed of nodes (often referred to as units or neurons) and connections between the nodes. With each node is associated a number, referred to as the node's activation, or output. Similarly, a number is also associated with each connection in the network, called its weight. These are (very roughly) respectively based on the firing rate of a biological neuron and the strength of a synapse (connection between two neurons). There are usually some neurons which take their activations from the outside environment or teacher; there may be, in addition, some nodes that are distinguished as output nodes. In the field of neural networks, which is also called connectionism, the terms node, neuron, and unit are used interchangably.

Each node's activation is based on the activations of the nodes that have connections directed at it, and the

0073-1129-1/92 $3.00

weights on those connections. The rule that changes activations in a network at each time step is called the update rule. A neural network model is a parallel model; typically, many activations could be updated simultaneously, although this depends on the topology of the network. Linear, sigmoidal, or Gaussian functions of a dot product of the weights and activations of the inputting units to a given unit are typically used as an update rule for that unit.

There are two main reasons for interest in neural network models: first, such models have been notably successful in modeling a variety of phenomena, such as disease diagnosis [1], speech synthesis [2], and pattern recognition of various types. Secondly, since these models are somewhat brain-like, the hope is that human-like cognitive capabilities may be achieved.

Learning in a neural network typically occurs by adjustment of the weights, via a learning rule. The network is typically trained either to complete an input pattern, to classify an input pattern, or to compute a function of its input. The goal is typically to get the output units to match some target pattern which is the desired output of a corresponding input pattern. At the beginning of learning, with the weights all "wrong", the network performs badly at its task; at the end, with the weights adjusted, one hopes that it will perform better. Typically the update and learning rules do not change over the course of learning, only the weights. After learning, the weights are usually not changed further.

The data are typically partitioned into two sets, a training set of input and target patterns, and a test set. The training set is used for learning. After learning, performance is gauged on both the training and test sets. Performance on the test set is what is most critical, since this tests the inductive model's validity on other data, that is, its generalizability.

A neural network model is a statistical model. In statistical terminology, the input patterns correspond to the independent variable, the target patterns to the dependent variable, and the weights to the parameters of the model. The output patterns are the estimated values of the dependent variable, using the model.

The neural network model in most common use is the error back propagation model [3] [4] [5]. This type of model has a layer of input units, a layer of output units, and one or more layers of hidden units in between. Each layer is usually completely connected to the layer above it, although in some variants of the model, some of these connections are missing. Activation flows from input to hidden to output units. The input at any unit is the dot product of the weights and activations coming into it, flattened by a sigmoid.

At the output units, differences between the outputs and the targets, i.e. the errors, are computed. These errors are then used to "back propagate" hypothesized errors to the hidden units, using the weights. The error at a hidden unit is the dot product of the weights connecting it to the outputs with the errors at the respective outputs At any given time, the model is at a given point in its parameter (weight) space. The goal is to minimize error to maximize the goodness-of-fit of the model. In order to do this, gradient descent in weight space is performed. This is done by changing each weight in proportion to its partial derivative of the error. This is done repeatedly, cycling through the input/output pairs of the training set, or choosing from them randomly, until the error stabilizes at a minimum. After that, performance on the network on patterns in the test set is measured. This is the method used in this paper.

3 Econometric Methodology

The *efficient markets hypothesis* (EMH) has found general acceptance in the financial community. In its weak form, the EMH asserts that an asset price is a reflection of all the information that can be gleaned from past price patterns. This implies that the movement of an asset's price is completely unpredictable given its history. Thus, technical analysis cannot provide a financially profitable investment strategy. The intuitive reason behind the unpredictability of asset price returns is similar to the reason why one rarely finds money lying on the ground. Nobody disputes the fact that money does indeed fall out of people's pockets but, when it does, it tends to be picked up fairly quickly. Profit opportunities in the asset markets are taken advantage of at their inception.

Traditional tests of the EMH have involved estimating a linear auto-regressive model of asset returns and testing the joint null hypothesis that the coefficients of the lagged returns equalled zero. Nevertheless, the inability to refute the null hypothesis does not imply an affirmation of the EMH. It simply may be that there are nonlinear patterns that are linearly undetectable.

Neural networks are a rich class of functional forms that are capable of approximating arbitrary Borel measurable functions [6]. This property allows us to exploit non parametric estimation methods based upon these neural networks. These models can be viewed as an input-output system of a particular form. A neural network describes a mapping between a vector of inputs and a vector of outputs. The simplest

model is constructed in the following form: define a $1 \times r$ vector of inputs X and a $1 \times p$ vector of outputs Y. Their interaction can be described as:

$$Y = X \cdot \beta$$

where β is the connection strength matrix of size $r \times p$. If we assume that one of the input neurons exhibits a value of one at all times, we then have the multivariate linear statistical model with an intercept term. Each of the X inputs could be processed in parallel, as they are in the brain.

The family of models described above is only capable of approximating quasi-linear models. Even with a slightly more sophisticated underlying structure, say quadratic terms, it performs poorly. This led to the development of more complex networks by applying further analogies with neural function in the cerebral cortex. Intermediate processing layers were introduced between the input and output. These layers are called hidden since it is not possible to obtain empirical observations on them. The output of this network model can be described as follows:

$$Y_k = \psi_k \left(\sum_{j=1}^{q} \alpha_j (X\gamma_j) \beta_{kj} \right) \equiv f_k (X, \theta),$$

where $\beta' = (\beta_1, \ldots, \beta_p)$ and $\gamma' = (\gamma_1, \ldots, \gamma_q)$ are the connection strengths, ψ_k and α_j are known functions, and $\theta' = (\beta', \gamma')$.

In back propagation, a member of the family of smooth and monotonically increasing functions e.g. the sigmoid squashing function

$$\alpha_j (X\gamma_j) = \frac{1}{1 + e^{-X\gamma_j}}$$

is used as the activation function α_j. Without loss of generality and for computational simplicity we define the output processing function ψ_k to be the identity function.

Alternative choices of activation function are possible. One example that time-series econometricians are familiar with is letting α_j equal the sine function. We therefore get the spectral analysis model where a linear combination of sine functions at different frequencies can approximate any function arbitrarily closely.

The back propagation method is implemented on θ using recursive m – estimation, namely:

$$\theta_{t+1} = \theta_t - \eta_t \nabla_\theta f_k (X, \theta)' (Y - f_k (X, \theta))$$

where η_t is the learning rate, ∇_θ is the gradient and θ_0 is an arbitrary starting value.

4 Empirical Results

We selected two stock index data sets and six stock price data sets. They are the New York Stock Exchange Transportation Index (TRANS), the Standard and Poor's Composite Index (SP), and the stock prices of IBM, Procter and Gamble (PG), Coca-Cola (CC), Boeing (B), Mobil (M), and 3M. We use two levels of temporal aggregation: monthly for TRANS and daily for all the rest. The observations for IBM, PG, SP, CC, B, M, and 3M span the period from January 14, 1987 to December 29, 1989. The total number of daily observations is 750. The observations for TRANS span the period from January 1966 to August 1989. The total number of monthly observations is 284. All our data is from the Center for Research in Security Prices (CRSP), except for the TRANS data, which came from the Citibase database.

We divided each data set into a two portions. The first portion, which, following general practice, we call the training set, is used to construct the network model. The second portion is used to test the validity of the model on data that was not used to produce the model, that is, what is commonly referred as "generalization" performance.

Table 1 presents summary statistics of the data. It shows the division of each data set into two portions, with dates, sample means and standard deviations for the training and test sets in each of the four cases. The variables in Table 1 are one day (month in the case of TRANS) returns for holding the the stock or index in question. The closing values are used to represent the daily price.

The average daily return of the six stocks, IBM, PG, B, CC, 3M, and M for the training period is 0.0002% and for the test period is 0.0007%. The most volatile stock during the training period was CC with a standard deviation of 0.0242. The most volatile stock during the test period was PG with a standard deviation 0.0343.

Using the TRANS data set as a benchmark, we implemented the back propagation algorithm using the signs of stock returns as inputs and outputs.[1] For each of the six data sets IBM, PG, SP, CC, B, M, and 3M we used a network of 9 input units, 5 hidden units, and one output unit. This represents a model of $(9 * 5) + (5 * 1) = 50$ weights, or parameters. Back propagation is much more parsimonious in number of parameters than the equivalent linear polynomial expansion with $2^9 - 1$ parameters.

[1] An alternative specification, using signed magnitudes as inputs and signs and magnitudes as two separate outputs was attempted. It did not perform very well.

Variable	*Sample*	*Start date*	*End date*	*Mean*	*Std. Dev.*
B	Train	1/14/87	1/4/89	0.0005	0.0181
	Test	1/5/89	12/29/89	0.0003	0.0256
CC	Train	1/14/87	1/4/89	0.0006	0.0242
	Test	1/5/89	12/29/89	0.0022	0.0151
IBM	Train	1/14/87	1/4/89	0.0003	0.0191
	Test	1/5/89	12/29/89	-0.0010	0.0108
3M	Train	1/14/87	1/4/89	-0.0007	0.0313
	Test	1/5/89	12/29/89	0.0011	0.0099
M	Train	1/14/87	1/4/89	0.0003	0.0229
	Test	1/5/89	12/29/89	0.0013	0.0127
PG	Train	1/14/87	1/4/89	0.0004	0.0230
	Test	1/5/89	12/29/89	0.0001	0.0343
SP	Train	1/14/87	1/4/89	0.0003	0.0162
	Test	1/5/89	12/29/89	0.0010	0.0082
TRANS	Train	1/66	7/82	0.0023	0.0555
	Test	8/82	8/89	0.0154	0.0482

Table 1: Summary Statistics

We used the same technique for IBM, PG, SP, CC, B, M and 3M, so let us use IBM as an illustrative example. To predict the value of IBM at period (day) t (corresponding to the output unit), we used the signs of the returns for the past 5 periods $t-1,\ldots,t-5$, as well as the return between t and: $t-10$ (two weeks), $t-22$ (one month), $t-132$ (six months), and $t-264$ (one year), for a total of 9 inputs. Positive asset returns are coded as a 1 while negative asset returns are coded as a 0. We have not yet experimented with varying the number of hidden units. General experience has led researchers to choose a number of hidden units less than or equal to than the number of inputs (e.g. [2], [1]).

In the case of TRANS, which is a monthly data series, we again used as inputs the signs of the returns for the past 5 periods (in this case months), as well as the change in price between t and: $t-10$ and $t-20$, for a total of seven input variables.

We performed back propagation learning on the training set for 200 iterations or epochs, where an iteration represents one complete pass through the input data, altering the weights in the network. We recorded the total sum-squared error (TSS) of all the training patterns. We considered, as is general practice, for the output unit to be predicting a positive sign for the subsequent return if it (a sigmoid unit) outputed above 0.5, and negative below 0.5. The performance is the percentage of the time that the predicted output agrees with the true output. We also calculated the RMS error, the number correctly predicted, and the average output of the network for each set. Table 2 presents the results of the neural network estimations on the financial data series. The TSS error and the RMSE are reported for all the asset price measures for both the training set and the test set. The average direction of asset price change, which corresponds to the average value of the output, is also presented. For the daily asset prices, the average output in the training set equals 0.4804 and in the test set equals 0.4922. This means that the neural network model predicts an upward movement approximately 48.04% of the time and a downward movement the remainder of the time (in the training set.) This is consistent with one of the implications of the "random-walk" hypothesis in which upward asset price movements should occur just about as frequently as downward movements.

The lower portion of Table 2 presents the actual performance statistics of the back propagation algorithm in and out of sample. The number of correct signs of the asset price returns coming from the model is compared to the actual data series. The percentage of correct signs is calculated for both the training and test runs. Standard deviations are also calculated for the percentage of correct signs under the null hypothesis of the model actually having a 0.5 probability of estimating the direction of change correctly. The probability of a Type I error, namely the probability that we would get the percentage of correct signs that we report given that the model was providing absolutely no explanatory power, is also presented.

The percentage of correct responses ranges from 51.6% for the B test set to 67.9% for the TRANS test set. The average percentage of correct responses for the training sets is 64.15% and the average percentage for the test sets was 56.13%. As is usually the case in many such experiments of this type, the performance on the training set (after training) usually exceeds that on the test set. This is because the training process tends to fit characteristics of the training set that are not generalizable (so called "overfitting"), that is, it fits "noise" in the training set as well as signal. Therefore, it is not desirable to train the training set beyond a certain number of iterations, since in the early stages of learning it learns the first-order characteristics of the training data, which are more likely to correspond to the true model. In subsequent iterations, the model learns second-order characteristics which vary from data set to data set and would prove to be of no help in attempts to generalize.

The estimated percentages of correct responses for the training sets are all more than two standard deviations from the value maintained under the null hypothesis. The standard deviations range from 2.59 for SP to 8.05 for B. These values correspond to very low probabilities of Type I errors in the training set with the highest value for all stock measures being 0.49%.

In the test sets, the estimated percentages of correct responses range from 0.5060 to 3.6682 standard deviations from the null of 0.50. Both indices, SP (aggregated daily) and TRANS (aggregated monthly) have correct percentages of 57.2% and 67.9% respectively. Their corresponding standard deviations are 2.28 and 3.27 and the Type I probabilities are 1.13% and 0.05%. M gives a correct percentage of 61.6%, and its standard deviation and type I probability are 3.67 and 0.01%.

These results provide strong evidence that back propagation models are able, using previous asset returns, to provide some predictive power for future returns. This result is evidence against the weak form of the EMH. With respect to IBM, PG, B, CC, 3M, the test sets did not provide evidence of prediction abilities. This result is in accordance with the result

of White [7] who was unsuccessful in forecasting IBM daily stock prices using neural networks.

Viewed jointly, all eight asset prices give a chi-square statistic of 115.84, with eight degrees of freedom. This statistic is significant at the 5% level. This represents a reasonable confidence level that our technique is doing significantly better than chance overall.

Figure 1 shows the performance of the back propagation model in the case of the TRANS stock index. The estimated outputs of the back propagation model are plotted in conjunction with the TRANS data, called the target in this instance. In January 1966, both data series are set to zero. For subsequent months, positive returns are recorded as $+1$ and negative returns are recorded as -1. These values are cumulated over the full training and test set. The Y-axis is drawn at the point where the training set ends and the test set begins. In the case of the target, you can think of the plot as representing the original TRANS data series where the magnitudes of all monthly returns are scaled to $+1$ or -1 depending on their sign. It is interesting to note how closely the output coincides with the target in the test set. Note that what matters in this figure is the degree in which the two curves follow the same trends, rather than how close they are to one another.

We also ran the learning algorithm for 850 iterations, after which there was no appreciable decrease in the TSS error, signifying that the network had reached a minimum in error space.[2] This documented the overfitting effect, since performance on the test sets was uniformly better (in one case, the same) in the case of 200 iterations of training than in the case of 850. Also, performance on the training sets was uniformly worse in the case of 200 iterations than in the case of 850, as would be expected given the hypothesis of overfitting.

5 Trading Simulation

In this section, we develop and implement a trading rule strategy based on the back propagation algorithm. First, we consider a buy-and-hold strategy where funds are invested at the beginning of the test period in a particular stock and this long position is cashed out at the end of the test period. The alternative strategy is driven by the output of the back propagation algorithm. A positive prediction of an asset's daily return corresponds to going "in" the asset while a negative prediction is translated as a move "out" of the asset and into a risk-free asset such as a U.S. Treasury bill. The relative performance of these two strategies will be measured in terms of average daily excess rate of return. These kinds of filter rules are similar to those found in [9], [10].

The average rate of return for the buy-and-hold strategy is expressed as

$$\bar{R}_{BH} = \frac{1}{N}\sum_{t=1}^{N} R_{jt}$$

where R_{jt} is the return of asset j in time period t. The filter rule determined by the back propagation algorithm earns the return of the stock for the days "in" asset j and the risk-free rate of return for days "out" of asset j. The average rate of return for the filter rule strategy is

$$\bar{R}_F = (1-f)\frac{1}{N_{in}}\sum_{t\in I} R_{jt} + f\frac{1}{N_{out}}\sum_{t\in O} i_t$$

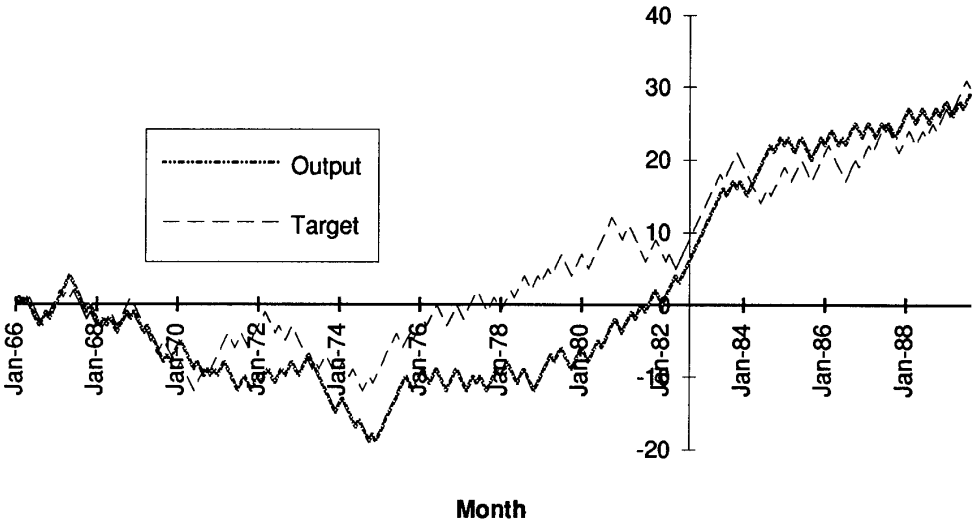
where f equals the proportion of days out of the asset relative to the total number of trading days, N_{in}, I, N_{out} and O denote the number and the set of trading days in and out of the asset respectively, and i_t is the risk-free rate of return. The arithmetic rate of return is used at the expense of the geometric one for the sake of clarity of exposition.

Some consideration must be given to transaction costs. These vary sharply according to the type of agent executing the transaction. Floor traders incur a clearing house fee which is approximately \$3 per transaction. Sweeney [10] estimates this transaction cost to translate to 1/20 of 1 percent for an average stock trade. This estimate is also corroborated by Fama and Blume [9]. Money managers and private agents face higher costs than floor traders. Sweeney conservatively estimates their transaction costs to be 1/5 and 2/5 of 1 percent respectively. We use the money managers' transaction cost of 1/5 of 1 percent in this paper.

Table 3 presents the results of simulated trading using the back propagation filter rule for the six stocks sampled at a daily frequency. These results are compared to a buy-and-hold strategy. An amount of \$100 is invested using each one of the strategies on the first day of the test sample, January 5 1989. On the last

[2] The gradient descent algorithm does not ensure a global minimum. Nevertheless, multiple runs from different randomly selected initial parameter values did converge to the same minimum. This would suggest either a global minimum or a local minimum with a large basin of attraction. Simulated annealing on the learning rate has been shown to ensure convergence to the global minimum [8]. Nevertheless, this technique is very costly in terms of computing time.

Figure 1: TRANS Output Performance



Variable	*Set*	*Sample Size*	*TSS*	*RMSE*	*Avg. est.*
B	Train	500	96.780	0.4400	0.4553
	Test	250	77.315	0.5561	0.5046
CC	Train	500	106.373	0.4612	0.4519
	Test	250	77.085	0.5553	0.4546
IBM	Train	500	111.065	0.4713	0.4908
	Test	250	70.759	0.5320	0.4971
3M	Train	500	108.348	0.4655	0.4862
	Test	250	74.724	0.5467	0.4840
M	Train	500	104.277	0.4567	0.4540
	Test	250	57.0159	0.4776	0.4840
PG	Train	500	109.555	0.4681	0.4755
	Test	250	71.301	0.5340	0.4973
SP	Train	500	116.163	0.4820	0.5488
	Test	250	66.3106	0.5150	0.5239
TRANS	Train	200	40.174	0.4482	0.5180
	Test	84	19.832	0.4859	0.5149

Variable	*Set*	*Nr. correct*	*% correct*	*Std. Dev.*	*Prob.*
B	Train	340	68.0%	8.0498	0
	Test	129	51.6%	0.5060	0.3085
CC	Train	336	67.2%	7.6921	0
	Test	132	52.8%	0.8854	0.1894
IBM	Train	306	61.2%	5.0088	0
	Test	133	53.2%	1.0119	0.156
3M	Train	322	64.4%	6.4399	0
	Test	131	52.4%	0.7589	0.2266
M	Train	331	66.2%	7.2448	0
	Test	154	61.6%	3.66824	0.0001
PG	Train	322	64.4%	6.43988	0
	Test	131	52.4%	0.7590	0.224
SP	Train	279	55.8%	2.5938	0.0049
	Test	143	57.2%	2.2768	0.0113
TRANS	Train	132	66.0%	4.5255	0
	Test	57	67.9%	3.2733	0.0005

Table 2: Empirical Results

Variable	*Filter rule*	*Buy-and-Hold*
B	\$97.72	\$97.74
CC	\$105.40	\$171.67
IBM	\$73.84	\$77.16
3M	\$90.97	\$130.53
M	\$98.35	\$136.14
PG	\$101.76	\$81.45

Table 3: Trading Profits

day of the test period, December 29 1989, both positions are cashed out and the dollar amounts are presented in Table 3.

These results do not provide conclusive evidence of the profitability of the back propagation filter over the buy-and-hold strategy. In four cases, the buy-and-hold strategy is superior to the filter rule. For IBM, buy-and-hold is 4.5% more profitable than back propagation. In the case of Coca Cola, the buy-and-hold strategy performs 63% better. For Boeing, the two strategies perform equally well. In only one case, that of Procter and Gamble, does the filter rule outperform the buy-and-hold strategy. In percentage terms, the back propagation filter is 25.0% more profitable.

6 Directions for further work

In back propagation learning, or for that matter, in any algorithm that searches a parameter (or weight) space for an optimal set of parameters, the learning time increases with the number of parameters. Moreover, as Weigend et al. [11] and others have pointed out, Occam's razor would lead us to prefer models with fewer parameters. Several researchers have proposed methods for reducing the number of parameters in back propagation learning and variants thereof. In particular, Weigend has found that using one such method, which he calls weight-elimination, can yield improved performance in forecasting tasks such as the one that we have attempted in this paper. Weigend adds to the error a term that is roughly a normalized sum of the squares of the weights. Since we are minimizing the error, the back propagation algorithm will attempt to send to zero some of the weights at the expense of others, to minimize the total sum of squares of the weights. We have not yet tried this algorithm on our data sets, but it looks promising.

Some of the following algorithms may also be useful. Fahlman and Lebiere [12] have proposed the Cascade-Correlation Learning Architecture as an alternative to standard back propagation. This algorithm adds new hidden units one at a time, each additional unit fitting some additional portion of the variance in the output signal. It is a cascaded model since each new unit introduced receives output from previous hidden units that have been added. Since this algorithm adds hidden units as needed until some error criterion is met, it also has the effect of minimizing the number of parameters in the model, given that the model must fit to some given level of accuracy. Ash [13] and Honavar and Uhr [14] have also proposed algorithms of this type; Ash's algorithm adds hidden units one at a time at the same level in the network, and Honavar and Uhr generate units which compute various functions of their input, which are added to the network if they reduce the error. We intend to test one or more of these algorithms on our data set and on subsequent data sets that we study.

Another algorithm has been developed explicitly for forecasting applications, the Temporal Difference Method of Sutton [15]. This model is based on the idea of computing an error between successive predictions rather than between output and target. Sutton finds that this method converges faster than ordinary back propagation, and we may apply it to our data sets.

We also plan on performing a factor and/or cluster analysis of each of the back propagation networks that we have developed, to attempt to account for its performance in terms that are theoretically understandable, instead of simply viewing the network as a black box. It may instruct us as to which input data are most relevant to the network.

7 Conclusion

We estimated a back propagation model on two stock price and two stock index returns. In the case of the stock indices, we found some evidence against the efficient market hypothesis both at the daily and monthly levels of aggregation. The evidence for the profitability of the back propagation filter rule was inconclusive when compared to the buy-and-hold strategy.

Traditional tests of the efficient market hypothesis have focused on linear models of asset prices. It is only recently, with the advent of non-linear modelling techniques, that this hypothesis has been put to a more stringent test. These new tests have provided some contradictory evidence to the efficient market hypothesis.

8 Acknowledgments

We would like to thank Duncan Chaplin for helpful discussions and assistance. This paper is a revised and updated version of a paper presented at an earlier conference [16].

References

[1] K. Saito and R. Nakano, "Medical diagnostic ex-

pert system based on pdp model," in *Proceedings of the IEEE International Conference on Neural Networks*, (San Diego, CA), IEEE, 1988.

[2] T. J. Sejnowski and C. Rosenberg, "Parallel networks that learn to pronounce english text," *Complex Systems*, vol. 1, pp. 145–168, 1987.

[3] Rumelhart, D.E., G. E. Hinton, and Williams, R. J., "Learning internal representations by error propagation," in *Parallel Distributed Processing: explorations in the microstructure of cognition* (D. Rumelhart and J. McClelland, eds.), Cambridge, MA: MIT Press, 1986.

[4] P. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, 1974.

[5] D. B. Parker, "Learning logic," Tech. Rep. 581-64, Office of Technology Licensing, Stanford University, 1982.

[6] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.

[7] H. White, "Economic prediction using neural networks: The case of ibm daily stock returns." University of California-San Diego, 1988.

[8] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[9] E. F. Fama and M. E. Blume, "Filter rules and stock market trading," *Journal of Business*, vol. 39, pp. 226–241, 1966.

[10] R. Sweeney, "Some new filter rules: Methods and results," *Journal of Financial and Quantitative Analysis*, vol. 23, pp. 285–300, 1988.

[11] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman, "Generalization by weight-elimination with application to forecasting," in *Advances in Neural Information Processing* (J. M. R. P. Lippmann and D. S. Touretzky, eds.), vol. 3, San Mateo CA: Morgan Kaufmann, 1986.

[12] S. E. Fahlman and C. Lebiere, "The cascade correlation learning architecture," Tech. Rep. CMU-CS-90-100, School of Computer Science, Carnegie-Mellon University, 1990.

[13] T. Ash, "Dynamic node creation in connectionist networks," Tech. Rep. 8901, Cognitive Science Institute, University of California, San Diego, 1989.

[14] V. Honavar and L. Uhr, "A network of neuron-like units that learns to perceive by generation as well as reweighting of its links," in *Proceedings of the 1988 Connectionist Models Summer School* (G. Hinton, T. Sejnowski, and D. Touretzky, eds.), (San Mateo, CA), Morgan Kaufmann, 1988.

[15] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, pp. 9–44, 1988.

[16] G. Tsibouris and M. Zeidenberg, "Predicting stock market fluctuations using neural network models." Presented at the Annual Meeting of the Society for Economic Dynamics and Control, Capri, Italy, 1991.