Neural Networks, Financial Trading and the Efficient Markets Hypothesis

Andrew Skabar & Ian Cloete

School of Information Technology
International University in Germany
Bruchsal, D-76646, Germany

{andrew.skabar, ian.cloete}@i-u.de

Abstract

The *efficient markets hypothesis* asserts that the price of an asset reflects all of the information that can be obtained from past prices of the asset. A direct corollary of this hypothesis is that stock prices follow a random walk, and that any profits derived from timing the market are due entirely to chance. In the absence of any ability to predict the market, the most appropriate strategy—according to proponents of the efficient markets hypothesis—is to buy and hold. In this paper we describe a methodology by which neural networks can be trained indirectly, using a genetic algorithm based weight optimisation procedure, to determine buy and sell points for financial commodities traded on a stock exchange. In order to test the significance of the returns achieved using this methodology, we compare the returns on four financial price series with returns achieved on random walk data derived from each of these series using a bootstrapping procedure. These bootstrapped samples contain exactly the same distribution of daily returns as the original series, but lack any serial dependence present in the original. Our results indicate that on some price series the return achieved is significantly greater than that which can be achieved on the bootstrapped samples. This lends support to the claim that some financial time series are not entirely random, and that—contrary to the predictions of the efficient markets hypothesis—a trading strategy based solely on historical price data can be used to achieve returns better than those achieved using a buy-and-hold strategy.

Keywords: Financial Trading, Neural Networks, Genetic Algorithms.

1 Introduction

Approaches to forecasting the future direction of share market prices fall broadly into two categories—those that rely on *technical analysis*, and those that rely on *fundamental analysis*. While technical analysis uses only historical data (past prices, volume of trading, volatility, etc.) to determine the movement in the price of some financial asset, fundamental analysis is based on external information; that is, information that comes from the economic system surrounding the market. Such information includes interest rates, prices and returns of other assets, and many other macro- or micro-economic variables. The use of technical analysis goes against the grain of conservative academic opinion, which regards this behaviour as irrational given the *efficient markets hypothesis* (Malkiel 1996).

The efficient markets hypothesis asserts that the price of an asset reflects all of the information that can be obtained from past prices of the asset. The argument is that any opportunity for a profit will be exploited immediately, and hence disappear. That is, the market is so efficient that no one can buy or sell quickly enough to consistently benefit. A consequence of the efficient markets hypothesis is that stock prices follow a random walk and are unpredictable based on any amount of historical data. The most appropriate investment strategy is thus a buy-and-hold strategy.

Despite the implications of the efficient markets hypothesis, many traders continue to make buy and sell decisions based on historical data. These decisions are made under the premise that patterns exist in that data, and that these patterns provide an indication of future movements. If such patterns exist, then it is possible in principle to apply automated pattern recognition techniques such as neural networks to the discovery of these patterns.

Several sources have reported on the simulation of trading agents based on Artificial Neural Networks (ANNs) (White 1988; Kimoto *et al* 1990; Yoon & Swales 1991; Weigend & Gershenfeld 1994). While the traditional approach to supervised neural network weight optimisation is the well-known backpropagation algorithm (Rumelhart & McClelland 1986), Beltratti, Margarita and Terna (1996) report on the use of genetic search for neural network weight optimisation in this domain. One of the advantages of genetic search as a weight-optimisation technique is that it allows flexibility in the choice of criteria that can be used as an objective function to guide search through the space of weight configurations. Thus, rather than making buy/sell decisions on the basis of a numerical prediction of the next day's price, genetic weight optimisation allows a trading regime to be discovered that optimises the financial *return* over some training period.

In this paper we describe the methodology by which neural networks can be trained indirectly, using a genetic algorithm based weight optimisation procedure, to determine buy and sell points for financial commodities traded on a stock exchange. In order to test the significance of the returns achieved using this methodology, we compare the returns on four financial time series with returns achieved on random walk data derived from each of these time series using a bootstrapping procedure. The bootstrapped samples contain exactly the same distribution of daily returns as the original series, but lack any serial dependence present in the original. Our results indicate that on some price series the return achieved is significantly greater than that which can be achieved on the bootstrapped samples. This lends support to the claim that some financial time series are not entirely random, and that—contrary to the predictions of the efficient markets hypothesis—a trading strategy based solely on historical

price data can be used to achieve returns better than those achieved using a buy-and-hold strategy.

The paper is organized as follows. Section 2 introduces the trading problem and outlines the methodology that we use to train the network to represent a financial trading strategy. Section 3 describes the experimental design and includes a description of the bootstrapping procedure used to create random samples and the performance benchmark used for testing. Section 4 presents empirical results, Section 5 provides a discussion of these results, and Section 6 concludes the paper.

2 Neural Networks for Automated Trading

One approach to developing neural network trading models is to first train the neural network to predict the value of the closing price of some asset one or more days into the future. An entry/exit (i.e. *buy* or *sell*) decision can then be made on the basis of this prediction. This section describes an alternative approach that does not attempt exact numeric prediction of the asset value, but rather, attempts to recognize patterns in the input data that can provide clues as to the optimal points to make buy or sell decisions.

The neural network buying and selling agent we use consists of an input layer, one hidden layer of sigmoidally activated units, and a single sigmoidally activated output that is thresholded such that output values above 0.5 are interpreted as a *buy* signal, and all other values are interpreted as a signal to *sell.* The inputs to the network are typically the price of the asset at the close of trade on the previous trading day, and variables derived from this. These could include moving averages, various delayed inputs (price two days prior, etc.).[1] The network is shown schematically in Figure 1.

The buy and sell signals that are generated by the network, in conjunction with the particular trading strategy that is adopted, determines the trading position. The trading strategy that we adopt is a *one-point* buying and selling strategy. This means that all available capital is invested in shares, or all capital is invested in some low-risk fixed interest security. On the basis of the trading signal issued by the network, either the low-risk security is sold and shares are bought (*buy* signal), or vice-versa (*sell* signal). Note that shares can only be sold if the investor is currently 'in the market', and bought if the investor is "not in the market".

The most common approach to neural network weight optimisation is backpropagation training (Rumelhart & McClelland 1986). Backpropagation is a supervised training algorithm that relies on the availability of a set of labelled training data. However, direct (i.e. *supervised*) training of the network is not possible in this case, since we are not supplied with labelled training data. That is, we do not know *a priori* what are the optimal buy and sell

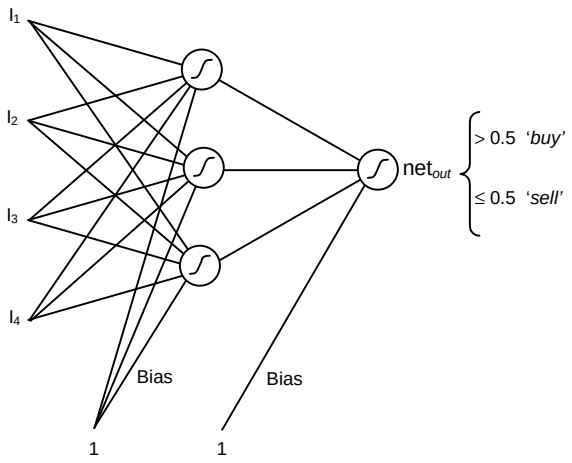

Figure 1. Neural network architecture.

points. An alternative approach is to discover a set of network weights indirectly by using some criterion to measure the performance of the trading decisions made by the agent, and to use this measure to guide search through the space of weight configurations. Genetic algorithms provide one means of doing this.

Genetic algorithms (Goldberg 1989; Holland 1975) have been extensively applied to complex parameter tuning problems in which various parameters of a system interact in unknown and non-linear ways resulting in a complex, irregular response surface (Bäck and Schwefel 1993). They have also been applied to neural network weight optimisation (Whitley 1995). Their performance relies fundamentally on the formulation of an objective function that is able to evaluate the success of competing individuals in solving the problem at hand. Since we wish to discover a neural network trading agent that is able to maximize returns, we must formulate an objective function that determines the return made by the agent over data representing the value of the asset price over some period of time. The optimality of the agent's decisions can be measured in terms of the financial return achieved in following the agent's decisions.

2.1 Calculating Financial Return

Assuming that an investor has a choice of investing her capital in either shares or a low-risk fixed interest security, the accumulation of wealth over a period of N days can be expressed as:

$$r_N = \prod_{i=1}^{N} \left\{ \left[\delta_{t-1} r_{m,t} + (1-\delta_{t-1})\, r_f \right] \times \left[1 - \delta'_{t-1} c \right] \right\} \qquad (1)$$

where r_N is the total return at day t, r_f is the return rate of the fixed interest security calculated daily, $r_{m,t} = (P_t - P_{t-1}) / P_{t-1}$ is the market return at day t where P_t is the share price at time t, δ_{t-1} is a delta function which equals 1 if capital is invested in shares at the completion of trading on day $t-1$ and 0 otherwise, c is the commission rate on a trade, and δ'_{t-1} is a delta function which equals 1 if a trade occurs at the end of day $t-1$ and 0 otherwise. Thus, the first factor appearing in Equation 1 represents the daily return rate that is applicable for the current day (i.e.

[1] Variables representing other information can also be used; e.g. the value of some other index or asset. In this study we restrict ourselves only to the prices of the asset under consideration, and variables that can be directly derived from this series.

the market return rate, or the fixed interest return rate), and the second factor provides an adjustment for the cost of transactions.

2.2 Genetic Weight Optimisation

A set of neural network weights can be represented as a binary string by encoding each (real valued) weight as a binary string, and then concatenating each of these individual strings together.[2] Genetic search proceeds as follows. A population of individuals, each representing a distinct neural network, is generated. Each of these networks is evaluated by following its trading predictions over the period represented by a set of historical training data and determining the return at the end of this period. The fitness of an individual is measured directly as the return that it is able to achieve. Reproduction, crossover and mutation operators are then applied to produce a new generation, with fitter individuals having a greater likelihood of contributing offspring to the next generation. This procedure is allowed to proceed until either a predetermined number of generations has been reached, or until there is no further increase in fitness. At the completion of search, the best network is used to make buy/sell decisions over some test period.

2.3 Moving-Windows Training/Testing

The procedure described above can be used to discover a set of network weights representing a trading strategy that performs well on a set of training data. The network can then be applied to some out of sample period to determine the placement of buy and sell decisions. In order to test the performance over some extended period we use a moving windows approach in which a pair of training/testing windows are advanced by *N* days after each training/testing cycle, where *N* is the number of days in each test period. This is shown schematically in Figure 2.

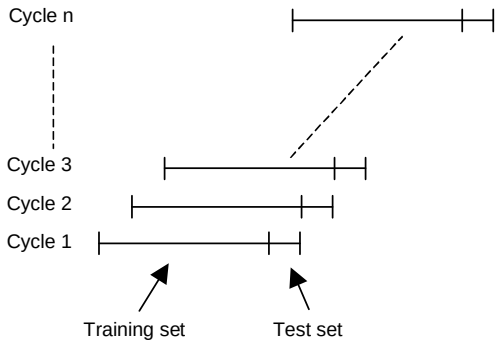

Figure 2. Moving windows. Testing the system over a large period of time can be achieved using a *moving-window* approach. In each cycle the training/testing window is advanced by *N* days, where *N* is the number of days in each forecast period.

[2] Because of this low-level encoding scheme, the standard genetic operators could be used without modification. However, we use a modified form of crossover called m*ulti-point restricted crossover*. This crossover operator works by selecting a crossover point within *each* weight, and restricting the crossover to *within* weights.

The advantage that using such a moving window approach has over that of using a single training/test cycle is that it allows for the fact that the prediction model may change over time. That is, a training strategy that was optimal in the past may not be optimal when projected too far into the future.

2.4 Previous Results

Based on previous work (Skabar & Cloete 2001; Cloete and Skabar 2001) we highlight the following observations. Firstly, there is usually a great variation in performance depending on the split between training and test sets. This variation arises not only due to the placement of starting and ending points for the test and training sets, but also from the size of these sets. For example, using a 24-day test window can lead to significantly different results from using a 25-day window. These observations are also supported by Le Baron & Weigend (1994) who state that "the variation due to different resamplings … is significantly larger than the variation due to different network conditions". However, the dependence on training window size does not appear as significant.

As a consequence of this, it is often possible to obtain good results by sufficient fiddling with the parameters, a form of so called *data-snooping*. Data-snooping occurs when a given set of data is used more than once—typically once for model parameter selection (e.g. selecting optimal values for training and testing size windows), and once for inferencing. The problem with using the same data for multiple purposes in this way is that it leads to the possibility that the results achieved may be due to chance rather than an inherent merit in the method.

A second set of observations concerns the complexity of the network required to represent the trading strategy. In Cloete & Skabar (2001) we generated a training set of buy/sell decisions using the network trained by genetic learning and then further trained this network using back-propagation with Structured Learning with Forgetting (Ishikawa 2000). The purpose of doing this was to obtain a network that was more amenable to human interpretation. We found that networks trained using a large number of input variables can usually be reduced to much smaller networks that only use two variables (e.g. a short term moving average and a longer term moving average). Furthermore, the results were found to often be not highly sensitive to the period of these moving averages. We also observed that networks with few units in the hidden layer are sufficiently complex to model a good trading strategy, and that in some cases, performance using linear rules (i.e. no hidden layer) can actually be better than that obtained using a hidden layer.

3 Experimental Design

As described in the introduction, a consequence of the efficient markets hypothesis is that price series follow a random walk, and hence any trading strategy based on timing or predicting the market will never consistently outperform a simple buy-and-hold strategy. However the trading strategy described above has been observed to outperform a buy-and-hold strategy on some financial

price series (Skabar & Cloete 2001; Cloete and Skabar 2001). How might we determine whether the observed returns achieved by using the neural network trading agent to time buy and sell points based on historical data are real or anomalous?

One way of testing this is to compare the performance of the procedure on real data with performance on one or more sets of random walk data. If performance on the random data does not differ significantly with that on real data, then we cannot claim to have discovered any real *predictability*. We first address the problem of generating random walk data.

3.1 Bootstrapping

We would like to compare the return performance on the real data with performance on random data that has the same empirical distribution of daily returns, but without the serial dependence—if any—that is present in the original series. If we knew the precise form of the distribution of daily returns present in the original data then we could simply sample from this distribution to construct our randomised datasets. However, we do not know the nature of this distribution. The difficulty of determining the precise distributional form can be avoided by direct sampling from the actual returns present in the original series. This technique is generally referred to as '*bootstrapping*' the original series (Efron and Tibshirani 1993; Levich and Thomas 1993). The bootstrapping technique that we use can be described as follows:

Step 1: Obtain the *return series* from the original price series. This is the series of daily returns, *r*, where *r* is the value of the index at the close of the current day's trading divided by the value at the close of the previous day's trading i.e., $r_t = P_t \div P_{t-1}$.

Step 2: Create a bootstrap *return* series by sampling (without replacement) from the return series obtained in step (1).

Step 3: Calculate the bootstrap *price* series for each bootstrap sample using $P_t = r_t \times P_{t-1}$.

The price series generated in this way are pseudo price series that retain all of the distributional properties of the original series, but do not contain the serial dependence present in the original series.

Note that in Step 2 we have the choice of sampling either *with* or *without* replacement. There is a distinct methodological advantage in sampling *without* replacement. Because the values of each price series will be identical both at the start and end of the test period, the buy-and-hold return for each pseudo series will be identical to that for the original series. This is important because it allows us to use as an independent performance benchmark the return of the network trained on the time series relative to the buy-and-hold return.[3]

At this point it is worth emphasizing that if the efficient markets hypothesis holds true (i.e. if financial time series are truly random), then each of the pseudo-series produced using the above procedure is as equally likely to be observed as is the original series. If each of the series are random, then we would not expect a significant difference between the returns achieved from applying the neural network trader to these (random) price series.

3.2 Hypothesis Testing

We are interested in determining whether the return achieved by applying the procedure of Section 2 to a real price series differs significantly to that achieved by applying it to the pseudo price series. Thus, the null hypothesis can be expressed as follows:

> H_0: *There is no significant difference between the return achieved when the procedure is applied to the real time series and the return achieved when the procedure is applied to the pseudo time series.*

The corresponding alternative hypothesis is thus:

> H_1: *There is a significant difference between the return achieved when the procedure is applied to the real time series and the return achieved when the procedure is applied to the pseudo time series.*

The null hypotheses can be tested by applying the procedure we have described in Section 2 to each of the pseudo time series that have been produced using the bootstrapping procedure. This will result in some empirical distribution of overall returns. The return on the *original* series can then be compared with this distribution of returns and a *p*-value obtained. The *p*-value simply provides the probability of observing a result as extreme, or more extreme, than that which would be expected if the null hypothesis were true; the smaller the *p*-value, the less likely the null hypothesis is true.[4] Rejection of a null hypothesis would allow us to accept the alternative hypothesis that the return achieved on the original price series is significantly different to that which we would be expected if the series was random. And this, in turn, would imply that there is some serial dependence in the original time series (which is not present in the pseudo time series), thus providing support against the efficient markets hypothesis.

3.3 Benchmark Comparison

As described above, the decision to sample the original return series without replacement results in pseudo price series which have an overall return which is identical to that of the original price series over the test period. This means that the returns that are achieved can be compared directly to those of a buy-and-hold strategy. This is an advantage because we are often interested in determining how the return achieved by some trading strategy compares with the buy-and-hold return.

---

[3] In order to achieve this, we have actually had to perform a *double* bootstrap—one for the prices in the training period, and one for the process in the test period.

[4] The usual convention is to reject the null hypothesis if the *p*-value is less than 0.05; that is, if chance of observing such an extreme result is less than 1 in 20.

The experimental procedure that we adopt can be summarised as follows:

1. Obtain the series representing the price of the financial commodity of interest at the close of trading on a set of consecutive trading days.
2. Create *N* random walk pseudo price series by applying the bootstrapping methodology to the original price series (i.e. the series in Step 1 above).
3. Apply the neural network trading procedure described in Section 2 to determine the overall return on the original price series.
4. Apply the neural network trading procedure to determine the overall return on each of the random walk pseudo price series, thus generating an empirical distribution of returns for the random walk data.
5. Determine the probability of observing the return achieved in (3) given the empirical distribution obtained in (4).

Note that in order to avoid the possibility of data-snooping (i.e. *fluking* a good result), the return measured on the original price series (Step 3 above) should be calculated and averaged over several trials.

4 Empirical Results

This section describes the application of the above procedure to the following four financial indices: the Dow Jones Industrial Average, the Australian All Ordinaries, the S&P500, and the NASDAQ.[5] All of the experiments are based on daily close values of the indices over the 5-year period from 1 July 1996 to 30 June 2001. Out of sample results cover the last 4 years of this 5-year period.[6]

The network architecture that was used in all experiments consisted of a single layer containing 2 sigmoidally activated units, and a single sigmoidally activated output. In each experiment, only two network input variables were used: a 5-day moving average (MA_5) and a 30-day moving average (MA_{30}). Both input variables were scaled linearly between 0 and 1. The commission rate on each trade (the value of *c* in Equation 1) was set at 0.1%.

Figure 3 shows the four financial price series, each accompanied by two of the 25 pseudo series produced by bootstrapping the original series using the procedure outlined in Section 3.1. Note that the values of each price series are identical both at the start and end of the 40-day test period. This means that the buy-and-hold return for each pseudo series will be identical to that for the original series.

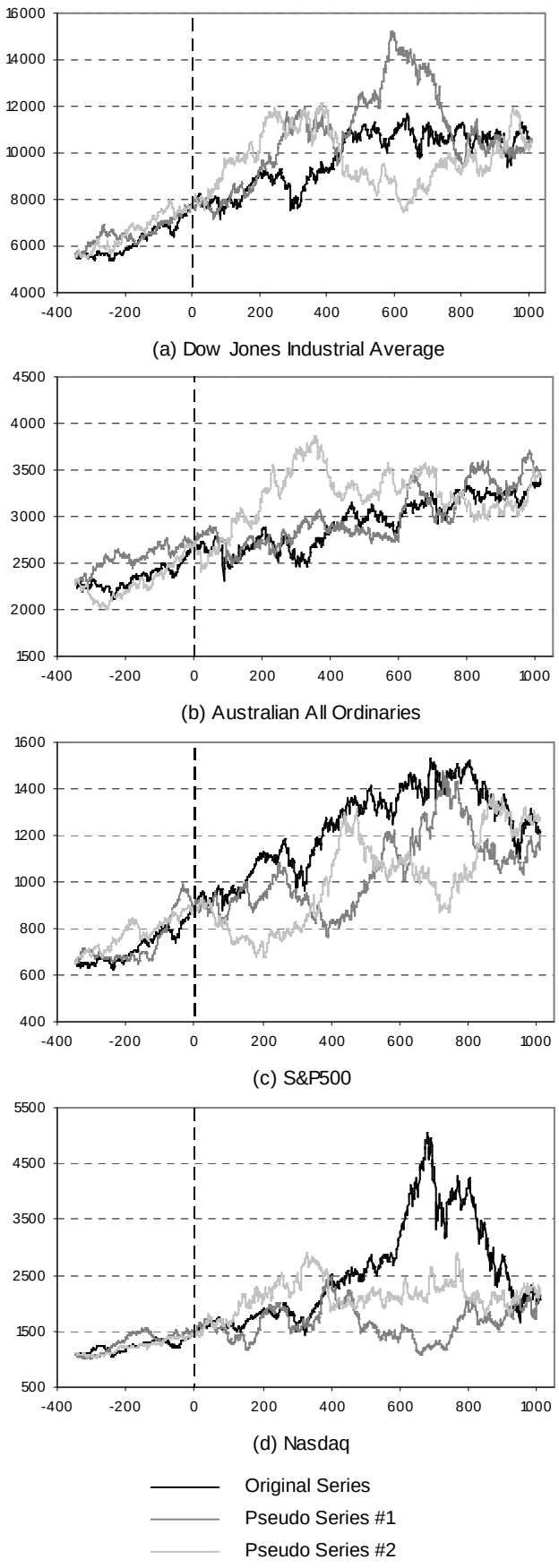

Figure 3. Original time series and two pseudo time series (a) Dow Jones Industrial; (b) Australian All Ordinaries; (c) S&P500; (d) NASDAQ. Pseudo series are constructed by sampling without replacement from the set of daily returns present in the original time series.

[5] This presumes the existence of an investment product (e.g. a unit trust) which purchases according to the make-up of the index (i.e. the investment fund manager invests in all companies represented in the index).

[6] It is not possible to provide test results over the entire 5 year period, since the training window size is 250 days (1 year).

	Dow Jones Industrial $r_{b\&h}$ = 1.369			Aust. All Ords. $r_{b\&h}$ = 1.257			S&P500 $r_{b\&h}$ = 1.383			NASDAQ $r_{b\&h}$ = 1.498		
Trial #	Trades	r_{net}	$r_{net}/r_{b\&h}$	Trades	r_{net}	$r_{net}/r_{b\&h}$	Trades	r_{net}	$r_{net}/r_{b\&h}$	Trades	r_{net}	$r_{net}/r_{b\&h}$
1	45	1.721	1.257	25	1.454	1.157	27	1.342	0.970	43	0.914	0.610
2	39	1.682	1.229	27	1.455	1.158	39	1.208	0.873	39	0.759	0.507
3	43	1.690	1.235	25	1.387	1.104	27	1.230	0.890	33	0.805	0.537
4	37	1.803	1.317	21	1.567	1.247	27	1.576	1.139	31	0.919	0.614
5	45	1.897	1.386	19	1.530	1.217	27	1.256	0.908	39	1.268	0.846
6	33	1.712	1.251	23	1.271	1.011	23	1.083	0.783	31	0.648	0.432
7	45	1.649	1.205	23	1.742	1.387	29	1.417	1.025	27	0.967	0.646
8	37	1.751	1.279	25	1.441	1.146	27	1.183	0.855	37	0.681	0.455
9	33	1.575	1.151	23	1.576	1.254	31	1.146	0.829	29	0.756	0.505
10	37	1.784	1.303	23	1.362	1.084	29	1.229	0.888	23	0.931	0.621
11	37	1.694	1.238	19	1.500	1.193	29	1.518	1.098	31	0.684	0.457
12	39	1.677	1.225	23	1.424	1.133	23	1.231	0.890	35	1.150	0.768
13	41	2.048	1.496	17	1.562	1.243	27	1.355	0.980	23	1.002	0.669
14	39	1.838	1.343	25	1.471	1.171	31	1.263	0.913	19	0.918	0.613
15	45	2.045	1.494	29	1.422	1.132	33	1.208	0.873	31	1.028	0.686
16	35	1.665	1.217	21	1.341	1.067	19	1.341	0.970	29	0.650	0.434
17	39	1.556	1.137	17	1.428	1.137	27	1.218	0.881	23	0.882	0.589
18	35	1.662	1.214	19	1.650	1.313	23	1.177	0.851	27	0.968	0.646
19	39	1.802	1.316	23	1.493	1.188	17	1.347	0.974	29	1.120	0.748
20	39	1.849	1.351	15	1.541	1.226	21	1.241	0.897	33	0.998	0.666
21	33	1.973	1.441	19	1.456	1.159	19	1.142	0.825	19	0.818	0.546
22	32	1.693	1.237	21	1.435	1.142	37	1.305	0.944	31	0.781	0.521
23	31	1.728	1.262	19	1.548	1.232	23	1.295	0.936	19	1.168	0.780
24	33	1.705	1.245	23	1.431	1.139	23	1.234	0.892	33	0.845	0.564
25	35	1.768	1.292	17	1.492	1.188	23	1.301	0.941	27	0.713	0.476
Mean	37.8	1.759	1.285	21.6	1.479	1.177	26.4	1.274	0.921	29.6	0.895	0.597
St Dev	4.3	0.127	0.093	3.5	0.099	0.079	5.3	0.112	0.081	6.4	0.170	0.114

Table 1. Returns achieved over 4 year out of sample period on original price series. 'Trades' is the total number of trades, 'r_{net}' is the overall return achieved by following the network output signals, and '$r_{net}/r_{b\&h}$' is the network return relative to the buy-and-hold return.

	Dow Jones Industrial $r_{b\&h}$ = 1.369			Aust. All Ords. $r_{b\&h}$ = 1.257			S&P500 $r_{b\&h}$ = 1.383			NASDAQ $r_{b\&h}$ = 1.498		
Trial #	Trades	r_{net}	$r_{net}/r_{b\&h}$	Trades	r_{net}	$r_{net}/r_{b\&h}$	Trades	r_{net}	$r_{net}/r_{b\&h}$	Trades	r_{net}	$r_{net}/r_{b\&h}$
1	38	1.219	0.891	55	1.689	1.344	25	0.898	0.649	17	2.310	1.541
2	19	1.149	0.839	27	1.083	0.862	37	1.154	0.835	48	1.416	0.945
3	38	1.158	0.846	41	0.880	0.701	25	0.829	0.600	30	1.090	0.728
4	47	0.890	0.650	19	1.179	0.938	48	1.769	1.279	27	1.247	0.832
5	33	1.452	1.061	35	1.430	1.138	25	1.367	0.989	37	1.262	0.842
6	25	1.576	1.151	31	1.091	0.868	29	1.348	0.975	21	1.204	0.803
7	26	1.351	0.987	31	1.240	0.987	38	1.013	0.733	29	1.277	0.853
8	38	1.606	1.173	18	1.463	1.165	46	1.376	0.995	32	1.137	0.759
9	29	1.107	0.809	36	1.365	1.086	27	1.299	0.939	19	1.277	0.852
10	32	0.886	0.647	20	1.032	0.821	27	1.741	1.259	23	1.033	0.689
11	27	1.832	1.339	31	1.263	1.005	30	1.146	0.828	39	1.810	1.208
12	25	1.379	1.007	32	1.229	0.978	29	1.201	0.868	24	1.016	0.678
13	33	1.063	0.777	31	1.286	1.024	30	1.075	0.777	21	1.251	0.835
14	31	1.709	1.249	33	1.131	0.900	39	1.017	0.735	24	1.085	0.724
15	25	1.271	0.929	30	0.947	0.753	29	1.703	1.231	25	1.196	0.799
16	26	1.010	0.738	24	1.256	1.000	23	1.332	0.963	29	0.917	0.612
17	31	1.254	0.916	39	0.903	0.718	29	1.733	1.252	37	0.882	0.589
18	28	1.055	0.771	27	0.953	0.758	26	1.241	0.897	31	1.614	1.077
19	29	1.571	1.148	25	1.558	1.240	32	1.181	0.854	25	1.891	1.262
20	21	1.171	0.855	44	1.061	0.844	26	1.127	0.815	45	0.966	0.645
21	21	1.719	1.256	23	1.223	0.974	25	1.191	0.861	24	2.033	1.357
22	20	1.277	0.933	25	1.456	1.158	38	1.396	1.009	21	0.721	0.481
23	32	1.125	0.822	22	1.020	0.812	21	1.257	0.909	25	1.708	1.140
24	31	1.242	0.907	29	1.132	0.901	25	1.029	0.744	22	1.204	0.804
25	18	0.921	0.673	31	1.084	0.863	28	1.065	0.770	22	1.389	0.927
Mean	28.9	1.280	0.935	30.4	1.198	0.954	30.3	1.260	0.911	27.9	1.317	0.879
St Dev	6.9	0.267	0.195	8.4	0.209	0.166	6.9	0.257	0.186	8.0	0.383	0.256

Table 2. Returns achieved over 4 year out of sample period on random walk pseudo price series. 'Trades' is the total number of trades, 'r_{net}' is the overall return achieved by following the network output signals, and '$r_{net}/r_{b\&h}$' is the network return relative to the buy-and-hold return.

In order to help avoid the possibility of data-snooping, the procedure described in Section 2 was applied to each of the 4 original time series over 25 trials. The number of days in the test period was different for each trial, and ranged from 25 (10% of the 250 training window period) to 49 days (just below 20% of the training period). In the case of the random walk pseudo series, one trial was performed for each pseudo series for each financial index. The test window size was randomly chosen between 25 and 49 (the same range as for the original price series). The genetic search population consisted of 40 individuals. The number of generations used in the training phase (for each training/testing cycle) was set at 250. Total processing time was approximately 12 hours.

Table 1 shows the financial return achieved by the neural trader on the original price series for each of the four financial indices. Table 2 shows the financial return achieved by the neural trader on the random walk pseudo

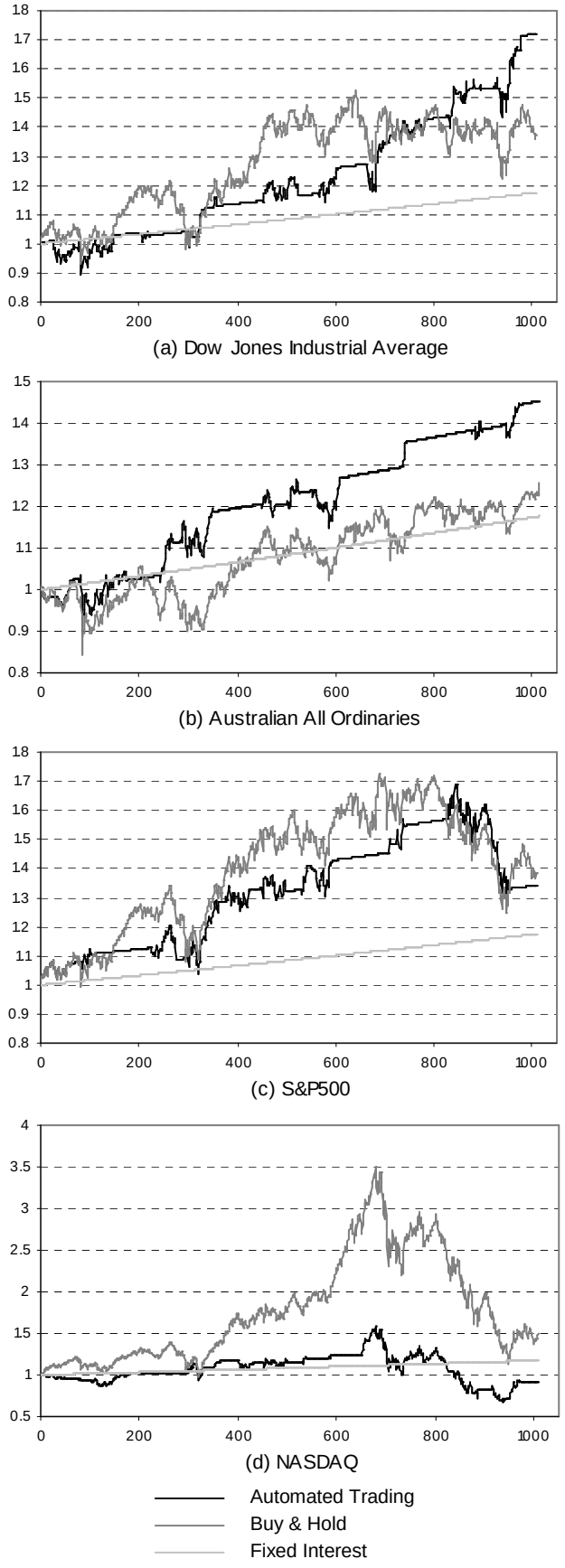

Figure 4. Automated trading return on original price series compared with buy and hold return and 4% fixed interest return. All data corresponds to Trial #1 from Table 1.

series. Means and standard deviations are provided at the bottom of each table. The buy-and-hold returns for each index are shown in the second line of each table.

The tables show—for each of the four indices—the number of trades (Trades), the overall return achieved by following the buy/sell signals generated by the neural trading agent (r_{net}), and the ratio of the return achieved by the network to the buy-and-hold return ($r_{net\ /\ b\&h}$). Figure 4 shows how the returns achieved by following the signals of the network compare with the buy-and-hold return and the return on the fixed interest (4%) low-risk security. These results correspond to Trial #1 from Table 1.

In the case of both the Dow Jones Industrial Average and the Australian All Ordinaries, the return achieved by the network on the original price series was greater than the buy-and-hold return on each of the 25 trials. The network return exceeded the buy-and-hold return on the original S&P500 series on only 3 of the 25 trials, and for the original NASDAQ series the return was lower than the buy-and-hold return over all 25 trials. The average return over the 25 random walk pseudo price series was lower than the buy-and-hold return for each of the four financial indices (0.935 of buy-and-hold return for Dow Jones, 0.954 for Aust. All Ords., 0.911 for S&P500 and 0.879 for NASDAQ).

Is the good performance on the Dow Jones and the Australian All Ordinaries significant, or is it due to chance? This question can be answered by determining the likelihood of observing such a return, given the distribution of returns on the random walk pseudo series (i.e. the distribution of returns in Table 2). Table 3 provides the mean return on the original series, the mean and standard deviations of the returns on the 25 random walk pseudo series, and the 1-tail (right hand tail) and 2-tail *p*-values calculated from this data using a *t*-test with 24 degree of freedom.

	Mean Ret (original)	*MeanRet (sample)*	*St.Dev (sample)*	*p-value* 1-tail	2-tail
Dow Jones	1.285	0.935	0.195	0.046	0.092
Aust. All Ords.	1.177	0.954	0.166	0.099	0.199
S&P 500	0.921	0.911	0.186	0.478	0.957
NASDAQ	0.597	0.879	0.256	0.855	0.290

Table 3. *p*-values for each of the four financial indices.

How can these *p*-values be interpreted? Consider first the 1-tail test for the Dow Jones data. The 1-tail *p*-value is simply the probability of observing a value *greater than or equal* to 1.285, given the distribution of values obtained from the random walk data. The *p*-value in this case is 0.046. That is, $Pr\ (\ X \geq 1.285\ ) = 0.046$, where X is the mean return on the original prices series. The two-tailed value differs from the 1-tail value in that it represents the probability of observing a result which departs from the sample mean by such a degree *in either direction*. In this case we have $Pr\ (\ X \leq 0.585$ *or* $X \geq 1.285\ ) = 0.092$. As mentioned earlier, a *p*-value of less than 0.05 is regarded by convention as sufficient to reject the null hypothesis. Which is the appropriate *p*-value to consider—the 1-tial or 2-tail value? This of course, depends on the null hypothesis. In our case, the null hypothesis stated that "there is *no significant difference* between the

return achieved when the procedure is applied to the real time series and the return achieved when the procedure is applied to the pseudo time series" (see Section 3.2). This implies difference *in either direction*, and hence the appropriate *p*-value is the 2-tail value. Since the *p*-value in this case is 0.092, this (null) hypothesis cannot be rejected at the 0.05 level. However, a weaker null hypothesis which states that "the return achieved when the procedure is applied to the real time series is *not significantly greater* than the return achieved when the procedure is applied to the pseudo time series" can be rejected at the 0.05 level, since the 1-tail *p*-value is 0.046. Neither the original null hypothesis nor the weaker form of the null hypothesis can be rejected at the 0.05 level for any of the other three financial series considered.

These results suggest that the ability to time buy and sell decisions on the daily movements of the Dow Jones Index is not anomalous, but indicates some regularity in the time series which can be exploited to make future profitable decisions. In the case of the Australian All Ordinaries price series, while the neural trading agent consistently outperformed a buy-and-hold strategy over all 25 trials, we cannot claim that the difference between performance on the original data and performance on the random walk data is statistically significant. In the case of the S&P500 and NASDAQ data, there is no evidence to suggest that the neural trading agent can perform any better than a simple buy-and-hold strategy.

5 Discussion

As mentioned in Section 2.4, one of the main dangers in attempting to automate trading strategies is that of data-snooping. We believe that our methodology is reasonably free from criticisms of data-snooping for the following reasons. Firstly, every experiment that we conducted—across all four indices—used exactly the same network structure, the same inputs, the same learning parameters, and the same training/test set samplings. The only difference was that between the actual time series. Secondly, by performing many trials on the original price series using different test set windows, we can be quite confident that the average returns we achieved are not anomalous.

A second criticism often directed at research which purports to have discovered a trading strategy that outperforms a buy-and-hold strategy is that the costs associated with trading have not been accounted for. Our experiments have been performed incorporating a trading cost of 0.1% per trade, which is currently the approximate commission for online trading. It is interesting to note the relatively low frequency of trading performed by the network, which ranges from a maximum of approximately 10 trades per year on the Dow Jones data to a minimum of approximately 5 trades per year on the Australian All Ordinaries. This trading frequency is significantly less than that of network traders based on *forecasting* numerical price movements.

What might be the cause of the differences in being able to successfully trade using these four indices. One possible explanation for this could be the fact that the Dow Jones and Australian All Ordinaries indices are *blue chip* indices. That is, they represent the averaged values of a large number of large, established, stable, and relatively secure companies. In contrast, the S&P500 and the NASDAQ include a significant proportion of *tech.* stocks, whose prices are known to have been much more volatile than blue chip companies, especially in recent years (recall the bursting of the technology stocks bubble). The inclusion of such volatile stocks in the makeup of these indices may make these series more chaotic, thus reducing the capacity to time trading decisions on past prices of these indices. However, this is highly speculative and experiments would need to be designed to test these ideas formally.

A rather obvious question that arises out of the results of this research is that if it is possible to exploit historical prices on the Dow Jones data to achieve a return better than a buy-and-hold strategy, how might we identify other series with this same property. That is, how might we determine *a priori* whether some given price series possesses such a desirable quality? One approach to this would be to perform the same experiments described in this paper over very many different price series and, on the basis of the results, assign each of these series some measure of what may cautiously be called 'predictability'. Patterns could then be sought between these so-called *predictability* values and measurable statistical properties of the price series (autocorrelation, Box-Pierce Q statistics, etc.). We leave this exploration for future work.

6 Conclusions

This paper has described a methodology by which neural networks can be trained indirectly, using a genetic algorithm based weight optimisation procedure, to determine buy and sell points for financial commodities traded on a stock exchange. In order to test the significance of the returns achieved using this methodology, the returns on four financial price series were compared with returns achieved on random walk data derived from each of these series using a bootstrapping procedure. These bootstrapped samples contain the same distribution of daily returns as the original series, but lack any serial dependence present in the original. The results indicate that on the Dow Jones Industrial Average Index, the return achieved over a four year out of sample period are significantly greater than that which would be expected had the price series been random. This lends support to the claim that some financial time series are not entirely random, and that—contrary to the predictions of the efficient markets hypothesis—a trading strategy based solely on historical price data can be used to achieve returns better than those achieved using a buy-and-hold strategy.

7 References

BÄCK, T. and SCHWEFEL, H.P. (1993): An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, vol. 1, 1993, pp. 1-23.

BELTRATTI, A., MARGARITA, S. and TERNA, P. (1996): *Neural Networks for Economic and Financial Modelling*. London, Thomson Computer Press.

CLOETE, I. and SKABAR, A. (2001); Feature Selection for Financial Trading Rules. *Proceedings of 13th European Simulation Symposium: Simulation in Industry*, Marseille, France, pp. 713-17.

EFRON, B. and TIBSHIRANI. R. (1993): *An Introduction to the Bootstrap*. Chapman and Hall.

GOLDBERG, D.E. (1989): *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, Addison-Wesley.

HOLLAND, J. (1975): *Adaptation in Natural and Artificial Systems.* Ann Arbor, University of Michigan Press.

ISHIKAWA, M. (2000):, Structural learning and rule discovery, in *Knowledge-Based Neurocomputing*. CLOETE, I. and ZURADA, J.M. (eds). Cambridge: MIT Press, pp.153-206.

KIMOTO,T., ASAKAYA, K., YODA, M. and TAKEOKA, M. (1990): Stock market prediction system with modular neural networks. *Proc. IEEE International Joint Conference on Neural Networks*, pp. I1-I6.

LeBARON, B. and WEIGEND, A.S. (1994), Evaluating Neural Network Predictors by Bootstrapping. Report-no: 9447 SSRI Working Paper, Paper EconWPA paper no. ewp-fin/9411002.

LeBARON, B., ARTHUR, W.B. and PALMER, R. (1999): Time series properties of an artificial stock market. *Journal of Economic Dynamics & Control*, vol. 23, pp. 1487-1516.

LEVICH, R. and THOMAS L., (1993), The Significance of Technical Trading Rule Profits in the Foreign Exchange Market: A Bootstrap Approach, *Journal of International Money and Finance*, 12, pp. 451-74.

MALKIEL, B.G. (1996): *A Random Walk Down Wall Street, 6th edn.* W.W. Norton, New York.

RUMELHART, D.E. and McCLELLAND, J.L. (1986): *Parallel distributed processing: exploration in the microstructure of cognition (Vols. 1 & 2).* Cambridge, MIT Press.

SKABAR, A. and CLOETE, I. (2001); Discovery of Financial Trading Rules. *Proc. Artificial Intelligence and Applications (AIA2001)*, Marbella, Spain, pp. 121-25.

WEIGEND, A.S. and GERSHENFELD, N.A. (eds) (1994): *Time Series Prediction: Forecasting the Future and Understanding the Past.* Reading: Addison-Wesley.

WHITE, H. (1988): Economic prediction using neural networks: the case of the IBM daily stock returns. *Proc. IEEE International Conference on Neural Networks*, pp. II-451-II458.

WHITLEY, D. (1995): Genetic algorithms and neural networks, in *Genetic Algorithms in Engineering and Computer Science.* WINTER, G. and PERIAUX, J. (eds). John Wiley, PP. 203-16).

YOON, Y. and SWALES, G. (1991): Predicting stock price performance: a neural network approach. *Proc. IEEE 24th Annual International Conference of Systems Sciences*, pp. 156-62.