Interpreting neurally encoded financial trading strategies

A. Skabar, I. Cloete
School Of Information Technology, International University in Germany, Germany.

Abstract

Several sources have suggested the use of neural networks for discovering trading strategies for buying and selling financial commodities on a stock exchange. However, a problem with neural network based trading systems is understanding their behaviour. The trading strategy is imbedded in the network weights, and visual inspection of these weights does not usually reveal any particular regular structure, or a clear interpretation of what the agent does. This paper reports the analysis of neural network behaviour in predicting buy and sells points for the Dow Jones Industrial Average index over a two-year trading period. The results show that the network represents rules that are similar to those discussed in many texts in the technical analysis literature.

1 Introduction

Approaches to share trading fall broadly into two categories—those that rely on technical analysis, and those that rely on fundamental analysis. While fundamental analysis is based on information from the economic system surrounding the market (interest rates, world events, prices of other assets, etc.), technical analysis bases trading decisions on historical data only (past prices, volume of trading, etc.). The use of technical analysis goes against the grain of conservative academic opinion, which regards this behaviour as irrational given the efficient markets hypothesis, and its corollary that prices follow a random walk [1]. However, 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 the (historical) data, and that these patterns can be exploited in financial decision-making.
Several sources have reported on the use of neural networks in financial trading [2][3][4]. One approach is to use neural networks for time series prediction [5]. In this case the network is used to represent a regression function that is used to predict either a future value, or a future direction in the movement of the time series. Trading decisions can then be made on the basis of this prediction. An alternative approach is to have the network represent a trading strategy [6]. That is, rather than basing a trading decision on a predicted value or movement in the price of some asset, the network directly outputs a decision—buy, sell, or do nothing. Thus, in this second approach, the network does not represent a continuous-valued function of its inputs, but rather, a discrete classifier.

A problem with using neural networks to make trading decisions is understanding their behaviour. It is insufficient to measure the performance of the trader only in terms of the return that it is able to achieve; we also want to understand why the agent decided to buy or sell. The trading strategy is imbedded sub-symbolically in the network parameters (i.e., the weights), and inspection of these weights does not usually reveal any particular regular structure, or a clear interpretation of the decisions made by the network. By understanding the trading strategy we would expect to gain more insight into the time-varying nature of share-trading models.

The paper is organized as follows. Section 2 introduces the trading problem and outlines the methodology by which we represent, train, and apply the knowledge stored by a neural network. Section 3 deals with the issue of extracting humanly comprehensible knowledge from such a trained network and provides a summary of important previous results/observations. Section 4 presents empirical results, Section 5 provides a discussion of these results, and Section 6 concludes the paper.

2 Neural networks and financial trading strategies

In this paper we assume a one-point buying and selling model in which at any time, either all available capital is invested in a high risk financial asset, or alternatively it is invested in a low-risk fixed interest security. On the basis of some trading signal, 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 ‘not in the market’.

The network that we use consists of an input layer, a single hidden layer of sigmoidally activated units, and a single sigmoidally activated output unit 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. Inputs to the network typically include 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.).

Because we do not know a priori what is the optimal sequence of buy and sell labels over the training data, we cannot use gradient descent training [7]. An alternative approach is to search for a set of network weights indirectly using some criterion to direct search through the space of weight configurations, and genetic
algorithms provide one means of doing this. Genetic algorithms [8][9] are robust search and optimisation techniques based on the mechanics of natural selection and natural genetics. They 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 [10]. They have also been applied to neural network weight optimisation [11]. 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 the return on some initial investment, we formulate an objective function that determines the return made by the agent over some training period.

The return achieved over a period of \( N \) days can be expressed as:

\[
r_N = \prod_{t=1}^{N} \left( \delta_{t-1} r_{m,t} + (1-\delta_{t-1}) r_f \right) \times \left[ 1-\delta_{t-1} c \right]
\]

where \( r_N \) is the total return at day \( t \), \( r_f \) is the return rate of the fixed interest security calculated daily, \( r_{m,t} = \frac{P_t - P_{t-1}}{P_{t-1}} \) is the market return at day \( t \) where \( P_t \) is the share price at time \( t \), \( \delta_{t-1} \) 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 \( \delta'_{t-1} \) 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. the market return rate, or the fixed interest return rate), and the second factor provides an adjustment for the cost of transactions.

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 training period and determining the return. 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.

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 1. 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.
3 Previous work

One of the most important design decisions in applying neural networks concerns the physical structure of the network (i.e., the number of hidden layers and number of units in these layers) as it is this structure that is the prime factor determining the representational capacity of the network. A complex network may fit the training very well, but may subsequently perform poorly in generalising to holdout data. A network which is too simple may not even be able to represent the training data adequately. This phenomenon is known as the bias-variance dilemma [12]. However, simpler networks are easier to interpret, and it is useful to be able to convert some given network into a simpler, but equivalent, network.

One means of simplifying the structure of a network is Structural Learning with Forgetting (SLF) [13]. Structural learning with forgetting is a three-step backpropagation learning procedure which, in addition to the standard mean square (quadratic) objective function, uses different regularization terms in each step. These steps are: (i) learning with forgetting, in which a penalty term that favours small weights is used to cause unnecessary connections to fade away, thus allowing a skeletal structure to emerge; (ii) learning with hidden units clarification, in which the penalty term forces each of the hidden units to be fully active or fully inactive, thus dissipating distributed representations, and (iii) learning with selective forgetting, in which the mean squared error is reduced by using a criterion function that only allows decay in weights whose absolute values are below some threshold value.

In Cloete & Skabar [14] we used SLF to convert a trained network into a simpler network. The initial (complex) network was trained using the method described in Section 2. This network was then used to label each training example with a buy or sell training output label. The network was then further trained using back-propagation with SLF. Although the resulting networks were identical to the original networks with regard to their predictions, the networks resulting from SLF used fewer hidden layer units than the original network (usually only one or two), and required fewer input variables (usually two moving averages). Moreover, the results were not highly sensitive to the period of these moving averages. For example, using 5-day and 30-day moving averages gave very similar performance to using 10-day and 60-day averages. It is interesting to
note that the networks are usually less sensitive to structure and choice of variables than they are to factors such as the training and test window periods, and the positioning of starting and end points for these windows. These observations are supported by Le Baron & Weigend [15] 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.

Once a network has been be converted into a simpler form using SLF, it is sometimes possible to interpret the behaviour of the network by direct inspection of the weights. For example, in [14] we used a subset algorithm [16] to extract approximately correct symbolic rules from the skeletal network resulting after SLF. The subset algorithm searches for a combination of weight values for each unit in the ANN which causes the unit to fire. From these combinations, symbolic rules can be inferred. However, the problem here is that the trading strategy represented by the network varies over time, and it is more difficult to explain these changes using the symbolic rules than it is to appeal to some visual means of interpretation such as inspection of the decision boundary in input space.

Although we expect the trading strategy (as discovered by the network) to vary over time, it is not clear how much variation should be expected. That is, how similar will be the trading strategy based on, say, a 250 training period with an overlapping 250 day trading period offset by, say, 25 days. Also, we would like to see whether the transition across trading strategies is smooth, or whether there are abrupt leaps in the form and interpretation of those strategies. These issues are examined in the next section.

4 Empirical results

The procedure described in Section 2 was applied to the Dow Jones Industrial Average Index over a 28 month prediction period. Two input variables were used: a 5-day moving average and a 30-day moving average. Both input variables were scaled linearly between 0 and 1. The hidden layer of the network contained two sigmoidally activated units. The commission rate on each trade was set at 0.10% and the low risk fixed interest rate was set to 4.00%.

In order to interpret the decisions made by the network, we examine the decision boundary represented by the network in the two-dimensional input space. The sequence of diagrams in Figure 2 show the decision boundary represented by the network over 10 overlapping 250-day training periods commencing at May 2000). Each successive diagram represents a time shift of 25 days (i.e., the size of the prediction window). The horizontal axis represents the scaled value of the 5-day moving average, and the vertical axis the scaled 30-day moving average. Note that rescaling is performed after each train/test cycle. The circles represent assigned buy labels, and triangles represent sell labels. Although all points are assigned either a buy or a sell label, actual trades only take place if the buy and sell signals occur when the investor is respectively “out of” and “in” the market. Actual trades are shown in black. The average return over the ten 250-day training periods is 41.5% per annum.
Examination of the decision boundaries in the sequence of charts in Figure 2 reveals a number of interesting observations. Firstly, consider the linearity of the decision boundaries. In many of the charts, the decision boundary is very close to linear over the whole of the input space. In those charts where some curvature is observed, this curvature occurs only in unpopulated regions of the input space. This is evident, for example, in (g). Thus, while the network is capable of representing non-linear boundaries, it appears that the decision boundaries discovered by the network are effectively linear over the regions of interest.

A second observation concerns the slope of the decision boundary. Consider Figures 2 (a) to (d). In each of these, the decision boundary slopes down to the
right, and the position and gradient of this boundary is relatively constant over these four training periods. In contrast, Figures 2(e) to (h) contain a decision boundary sloping up to the right, and again, the position and slope of the boundary is relatively constant. It is interesting to note that Figure 2(e), in which the gradient first changes from negative to positive, happens to be the first training set in the sequence containing September 11, 2001. Figure 3 shows the placement of the actual buy and sell points, together with the moving averages, for the training sets corresponding to Figures 2(a) and 2(f). Note that in Figure 3(a), buy trades always occur when the short-term average is in an upward trend. Conversely, sells occur when the short-term average is in a downward trend. However, the precise point at which the trades occurs depends on the value of the longer-term average, and this relationship can be seen from the Figure 2. It is significant to note that Figure 3(b) (which corresponds to Figure 2(f)) contains far fewer trades than does 3(a). This is most likely due to the fact that the best profit in this case can be obtained when prices fall to low resulting from the events of September 11 (in the vicinity of day 560 on the chart). This low did not appear on the sequence of training sets in Figures (a) to (d), and it appears that this new low dominates the trading strategy from Figure 2(e) to 2(g), after which there is some alternation between the two models, possibly signalling a recovery to the relative stability observed earlier in the sequence.

Figure 3: Placement of buy/sell decisions and moving averages. Curves represent scaled values of moving averages: 35-day (black), 30-day (grey). Solid vertical lines represent buys; broken represent sells.
The ability of the network to represent a trading strategy that works well on the training data does not necessarily mean that the trading strategy will perform well when applied to out-of-sample data. When the trading strategy discovered over each 250-day training windows shown in Figure 2 is applied to the corresponding test window, the sum of returns over the ten 25-day windows is 4.47%. This is slightly above the fixed interest return of 4.15%, and much greater than the buy & hold return of –5.99% (i.e., a loss). Performance over the entire 28 month out-of-sample period from January 2000 to April 2002 is shown in Figure 4. Over this period, the return achieved by the trading was approximately 31%, compared with 9.8% for the low risk security and –13.5% for holding the index. The total number of trades over the 28 month period is 23.

Figure 4: Returns achieved over 28 month simulation period. Trading results in a return of 31.2%, compared with 9.8% for a fixed interest security and –13.5% for a buy & hold strategy.

In particular, note the exceptional performance for the 1-year period from approximately day 100 to day 350. A possible explanation for this is that the value of the index was relatively stable over the year preceding this period, and the network was able to utilize this stability to model a good trading strategy. Events such as September 11 appear to have a disruptive effect on the stability of the model, and prohibit network from being able to discover a trading strategy that generalises well to holdout data.

5 Conclusions

While neural networks are capable of representing non-linear decision boundaries, the boundaries observed in the space of moving averages tend to be linear. Analysis of decision boundaries over a partially overlapping temporal sequence of training sets showed that the model represent by the network alternated between two trading regimes. The network appears best able to model a strategy that generalises well to holdout data in periods of relative stability in index prices. Extreme changes in the value of the index can cause instability in the strategy. Future work in this area could focus on discovering indicators which can be used to detect the point at which an existing model ceases to be valid.
References


