City University
MSc in Artificial Intelligence
Project Report
2007

An Application of Neural Networks, Propositional Logic and Technical Analysis to Predict the Stock Market

A. T. Mooney

Supervised by: A.S. Avila Garcez

January 2007

Abstract

This thesis used FTSE100 data and combined it with stock market technical indicators to produce boolean data sets. These data sets were then used to create three training data sets, through two simple forward looking strategies and Q Learning. The generated training data was found not to be suitable for training of the neural networks. A second strategy of evolving a neural network through an elementary evolutionary approach allowed the trained network to gain a 35% increase in "profit" generated over that of the base line. The P value for this method was calculated as 0.16.

Keywords: technical indicators, artificial neural networks, propositional logic, prediction.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisor, Dr. Artur Garcez for all his help during this thesis, but mostly Diane for her patient, support and proof reading of “something I really didn’t want to known about”.

Chapter 1

Introduction

1.1 Domain Background

1.1.1 Stock Markets

The stock market is a highly volatile and dynamic system with substantial sums of money involved. It is desirable to automate many trading decisions to improve consistence. Tools that are capable of showing logical processing and provide reliable results can give an advantage. There has therefore been a large amount of research towards this goal. To date however no examples of applications using connectionist logic within this domain have been done. This thesis combines these AI techniques with the aim of providing some new incites into the stock market.

1.1.2 Predicting Stock Markets

Stock markets are non linear systems which can make many AI techniques ineffective. The auto correlation on a day to day basis is very low and the behaviour of the stock market is very similar to that of a random walk (Hellstrom & Hellstrom 1998). Technical analysts using serial correlation to determine how well the past price of securities predicts the future price, concluded the serial correlation of stock prices were economic and statically insignificant (Hawawini & Keim 1995), (Malkial 2003).

1.1.3 The Approach Used

By applying Artificial Neural Networks (ANNs) with propositional logic this thesis aimed to solve the above problems. Many previous researchers, Singh & Fieldsend (2001), Nygren (2004), Albanus & Batchelor (2002) for example examine ANNs ability to predict the stock market using data generated only from stock price movements. Often significant preprocessing was done to reduce the domain space, then technical data was directly entered into the ANN. This places a large burden upon the network to "understand" the meaning of the figures. In fact this requires the network not only recognise all the associations a human expert might but also deal with a wide range of interacting variables.

The aim of this thesis is to reduce the complexity of the input data to simple boolean values, so the network could concentrate on forming associations. Input data can be reduced to boolean values by introducing a layer of abstraction. In this case, some of the financial rules that a human expert uses. To do this, these financial rules are converted into propositional symbols and logical rules which can be used to generate an ANN. The ANN is then trained using historical data. During the training the ANN will relearn the logical symbols relationships. Finnally new financial rules can be extracted from the network.

1.2 Objectives

The objective of this thesis is to compare various ANNs ability to generate a "profit" using FTSE100 historical data based on two different learning approaches.

1.3 Steps required to meet objectives

To achieve the above aim the following steps were taken. The following C++ libraries and programs where created;

- *Neural Network Library:* An implementation of neural networks utilising matrices for efficient calculations.
- *Propositional Logic Library:* A simple proposition logic library allowing basic logical sentences including the boolean operators; AND, OR, NOT, IMPLIES, IFANDONLYIF.
- *Logic/Neural Factory Library:* Generated a neural network from a given propositional logic program.
- *Reinforcement Agent/Environment Library:* Utilised an agent based approach. The environment specified the domain of the problem and the agent specialises to it.
- *Technical Analysis Library:* Used raw FTSE100 data to generate technical stock data.
- *Test programs:* Created for all libraries to ensure that each library was behaving correctly.
- *Stock market simulator:* Used historical stock movements to allow ANN to buy and sell shares in an attempt to make a profit.

1.4 Tests

To test the objectives the following scenarios were proposed.

1. An ANN was defined with only input/output logical symbols and trained using back-propagation.
2. An ANN was defined with only input/output logical symbols and trained using an evolutionary strategy.

3. An ANN was defined with logical symbols and the initial weight connections were set so as to define some financial rules.

To refine the ANN predictive capabilities these scenarios are first trained with sections of FTSE100 historical data. The ANNs are then tested with previously unseen FTSE100. Finally the results from the scenarios are compared with a base line experiment to see if there has been any statistically significant change in the ability of the ANN to predict the FTSE100.

1.5 Project Modifications

The objectives of the project changed slightly from the initial concept detailed in the appendix 7.1. This was due to difficulties discussed in the proposal being realised. The initial project intended to apply financial logical rules to an ANN and then use further data to refine the networks. After this, the intention was to use logic extraction algorithms to improve the initial financial rules sets. Difficulties in obtaining adequate training data meant this final step could not be accomplished. Instead more time was devoted to creating a training data set to use with back propagation and learning methods which did not need training data.

1.6 Technical Analysis and Fundamental Analysis

A large amount of research has been conducted on out performing the stock market. It was necessary to understand these techniques before any AI techniques were employed.

Fundamental analysis uses a wide range of relevant information to calculate companies true stock value. This data is diverse and includes financial reports and non financial information, for example competitors success; country demographics; changes in consumer preferences or advances in technology. It is clear from such disparate information quantifying results is hard. This makes fundamental analysis a reflection of an analysts history, knowledge and interpretation of the company. This is subject to human error and individual judgements. An example of this could be considered the late 90's "dot com bubble" which had internet only companies overpriced. It took a number of years of speculation before the market correction took place.

Technical analysis ignores the company's fundamentals but examines the patterns of stocks price variations. Much of technical analysis can be seen as the analysis of human behaviour through the medium of the stock market. Technical analysis can be applied to any commodity which is governed by supply and demand. There are many technical indicators an analyst will use to predict trends. These are based on the stock's open, close, high, low prices and volumes traded.

Technical analysis falls into two categories, several examples of each are now presented.

1.6.1 Technical Indicators

Technical indicators are time series data computed from a stock's price information, volumes and indexes. Examples of these are;

- Moving averages, simple, exponential and weighted: These use stock prices averaged over a defined number of days. Simple takes the mean value of the stock price over the defined number of days. Weighted and exponential averages, use functions to give recent prices a heavier weighting in the final average.
- Momentum indicator: Measures the amount of change that has occurred in a stock's price over a given time.
- Moving Average Convergence Divergence (MACD): Is the measure of the difference between two moving averages. A positive MACD indicates the short term moving average is higher than the long term moving average.

1.6.2 Price Patterns

Price patterns are formations that appear in a stock's price information. Examples of these are.

- Trend: When the price is heading in one particular direction. For example an upward trend would have the consecutive highs continually increasing.
- Resistance and Support: If a stock reaches a new high then falls, traders will sell if the price reaches the previous high again, to avoid missing the opportunity of selling at the new high. This, has a tendency to create a ceiling in the share price and is known as a resistance level. A similar pattern can develop with the buying of shares when a new low is reached, this is known as the support level.
- Head and Shoulders: This pattern consists of three peaks with an initial upward trend and where the centre peak is the highest. This is a reversal pattern and signals a change in the direction of the market. The two lows between the three peaks indicate the "neck line". The volumes stocks are traded in, help confirm the pattern is a head and shoulders. The new support level is set as the price difference between the head and neck line, subtracted from the neck line.

This thesis concentrates on technical indicators as a starting point. This was because including pricing patterns would add significant complexity without having tested the initial concept.

1.7 Basics of Logics

Modern day logic is rooted in mathematical definition. A logical language has a syntax which defines the parameters of the language structure. This allows sentences to be interpreted by the rules of the language, this, is the concept that a sentence must be well formed. Semantics of a sentence, refers to the meaning of a sentence, in logic this is defined as true or false. A possible world or model, is the scenario in which the logic is operating. Finally, entailment is the concept that one logical sentence follows logically from another sentence. For example, a entails b if and only if, in every model in which a is true, b is also true.

1.7.1 Propositional Logic

Propositional logic is the most basic mathematical logic and consists of symbols and connectives. A symbol can be true or false and represents any idea or fact. Sentences are defined as either atomic sentences, a single symbol, true/false value or a complex sentence. A complex sentence is a combination of atomic sentences combined through logical connectives. Common forms of propositional logic use five logical connectives; Negation, a logical NOT, alters the boolean meaning of the sentence; Conjunction combines two sentences using a logical AND; Disjunction combines two sentences using a logical OR; Implication combines two sentences and means the first sentence implies the second sentences but not vice versa; Finally Bi-conditional combines two sentences and means if and only if, the first sentence implies the second sentences and vice versa (Russel & Norvig 2003).

Having defined a problem in propositional symbols, logical mathematics can then be applied to solve it. The truth of a sentence is computed from its basic symbols and propagates outwards.

A knowledge base is used to contain all currently known facts about the model. It is defined as a list of sentences. The logical equivalent of a knowledge base is a single sentence joining, all sentences in the knowledge base by conjunctions.

Logic calculus makes use of logical equivalences. Equivalences enable algebra in logic and can be utilised to reform logical sentences from human readable to easily computationally solvable sentences.

Propositional logic enables a detailed description of problems or domains and can solve very complex problems such as logic circuits but lacks expressive power, the ability to relate objects to each other.

1.8 Basics of Artificial Neural Networks

The first neural networks where conceived by McCulloch & Pitts (1943). An artificial neuron has 3 basic qualities; a set of inputs which have associated weights; an adding function, summing up all the inputs and an activation function which describes the firing function of the neuron and is based in theory on a biological neuron. The effect of the activation function is to limit the output of the neuron and is sometimes referred to as a squashing function. The neurons are connected together in a network with weights associated with each connection. More significant pathways are expected to have greater weights attached and thus a greater influence on the connecting neuron.

A perceptron is a network of one or more neurons that do not have a hidden layer. Applications of perceptrons are limited and it has been proved that perceptrons can only classify in linear separable problems (Minsky & Papert 1972). ANN therefore utilise a hidden layer between the input and output layers. This, combined with a non linear activation function allows the network to approximate any function, (Hornik et al. 1989).

The typical layout of an artificial neural network utilities a layering system. The first layer is known as the input layer and its primary function is to transmit information to the next layer. The second layer is know as the hidden layer. There can be as many hidden layers as required, however a single hidden layer can approximate any function. The third layer is the output layer and effectively collects the output from the hidden layer and presents the results.

The layout of the network is also heavily influenced by the domain problem. A typical example for an ANN classifying system would be to have a large number of input nodes. These feed forward to a smaller number of hidden nodes, which feed to the output nodes, with each of the output nodes representing a class. A well trained neural net will then activate an output node to classify the input data.

There are many types of activation functions that can be used, only a few are listed here. The choice of activation function once again depends on the domain. The identity function, the activations function used for the input neurons, output the same values as the net inputs. The binary threshold function returns zero output until the threshold limit is reached and gives an output of one. The sigmoid function, which is a commonly used function, outputs a continuous range from zero to one but not in a linear correlation.

1.8.1 Back-Propagation

A common method of teaching a neural network is supervised learning. This can be done via the back propagation algorithm. Input patterns are repeatably shown to the network with corresponding output patterns available. The difference between the calculated output and the desired output is determined. This difference, the error, is propagated back through the network and the weights are adjusted accordingly. In adjusting the weights a learning rate η is used between 0 and 1. A large learning rate, will help the net initially learning faster. This should however be reduced as the learning progresses, to fine tune the network. To try to prevent the network from becoming trapped in a local minima some implementations use a momentum parameter α between 0 and 1. The effect of this is a large correction in one direction will still have an impact in the following learning iteration. The full derivation of the back propagation rule is beyond the scope of this thesis but begins by expressing the error derivative and arrives at equation 1.1, which defines the change in weights after every iteration. In equation 1.1 δ_j refers to the gradient of the error.

$$\Delta w_{ij}(n+1) = \eta(\delta_j o_i) + \alpha \Delta w_{ij}(n) \tag{1.1}$$

Preprocessing of data for neural networks is essential. The most common form of preprocessing is to use a linear normalisation to ensure all the data is between 0 and 1 or -1 and 1. This keeps the data in a manageable range for the network. If there is only a small amount of data at the extremes, this results in the majority of the data being concentrated in a small range in the centre. This increases the difficulty for the network to discover patterns, so a non linear normalisation process may be chosen instead. The choice of the training data is also important and it may be necessary not to use historic data due to excess noise or a lack of examples within the data. This introduces further complexities, as it is important when preparing the training data not to add any bias or remove important features.

1.8.2 Inherent Implications of Using Artificial Neural Networks

ANNs are noise tolerant and can accurately classify when data is missing or incorrect. ANN can also be easily retrained while utilising its current knowledge. Another benefit of neural networks is, implementations can be completely free of prior assumptions, so the model does not depend on the author's, possibly incorrect, understanding of the domain. Finally neural nets are highly parallel computation devices and can be implemented in hardware making them exceptionally fast.

There are however, several draw backs of neural networks. Predominately networks operate as "black boxes". Decisions made by the network in the classifying process are not known to the user. Research is being conducted to address this in special cases (Garcez et al. 2001). The training of the network can be very computationally intensive and data inefficient. The design of the network can be quite arbitrary, guidelines on how many neurons a layer should have or many layers a network should have are available but do not guarantee best results making it a trial and error process. Over fitting can occur, when the networks is too highly trained and is unable to generalise with real data. This can also happen if an incorrect network design has been used. Careful testing is necessary to prevent over training from occurring.

1.9 Evolutionary Computing

Evolutionary computing is an umbrella term used for a variety of AI techniques inspired by Darwinian Evolution. A simple evolutionary algorithm will; initialise a set of candidate solutions; evaluate the solutions; select parents from the candidates; recombine parents to produce new candidates; evaluate and repeat until a termination criteria is reached. In this manor using parent selection, survivor selection and recombination techniques each generation becomes a more optimised solution.

Evolutionary techniques do not require training data, as they are modified by the environment. This can however make them computationally expensive solutions, often requiring several hundred evaluations for a single incremental improvement (Eiben & Smith 1998).

1.10 Reinforced Learning

Reinforcement learning uses the concept of learning through an agent interacting with the environment. It is different from supervised learning in that, there are no training examples and the agent learns through trial and error. A set of states are used to represent a model of the environment to the agent. A state which contains all relevant information is known as a Markov state and it is required that all states are Markov states. An example of this is a chess game, after every move, the chess board captures all relevant information about the game, preceding states are no longer relevant, the current state contains all the information needed to play the game. (Sutton & Barto 2000).

Chapter 2

Literature Review

The following chapter will review relevant literature and discus the basis for all the techniques used in the thesis.

2.1 Efficient Market Hypothesis (EMH)

The Efficient Market Hypothesis (EMH) Fama (1970) states that at any moment in time all the relevant information about a company is encapsulated by its stock price. Agents trading in stock prices act intelligently, in that they sell when they believe the stock is over valued and buy when they believe the stock in under valued. If, the EMH is to be believed the stock price already reflects an accurate value of the share price and therefore trading becomes nothing more than chance.

The "random walk hypothesis" is closely associated with the EMH, in that all the information about a stock is already reflected in the price. New information is immediately and efficiently incorporated into the price as it becomes available. As it is impossible to predict news and all current information about the stock is incorporated into the price, tomorrows change in price will only be effected by tomorrows news, therefore for any point in time, future prices can only be random.

Assuming there is a plausible argument and there is statical evidence to technical analysis, this does not guarantee economical sense once the share trading charges have been factored in and all agents have taken their cuts. Finally any successful prediction strategy will quickly be incorporated into all models thus rendering any advantage it might give ineffective.

If the EMH and random walk theories are correct, it implies that technical analysis is flawed and will not be able to predict any movements at all. The question remains "Why do people still use these tools?"

Many analysts believe that the market does not fully obey the EMH in its purest sense but rather a weaker version. Three adapted versions have been proposed. 1) The weak form of EMH states that all past market prices and data are fully reflected in stock prices and therefore technical analysis is of no use. 2) A semi-strong EMH asserts that all publicly available data is reflected in the stock price and therefore fundamental analysis is of no use. 3) Finally the strong version asserts that all information is incorporated in to the stock price, thus even insider information is of no use.

A technical analyst will assert that even if all information is already present in a stock's value, the future prices may still be predictable. In fact this is a benefit as there is too much information in the form of, news, economics, politics etc, for any trader to fully interpret. This means the solution to predicting stock process becomes a purely mathematical search for repeating patterns and understanding of stock movements. A technical analyst believes that market trends and while the market may be efficient, humans are not. The EMH assumes that agents will be acting intelligently and in their own best interests. Humans tend to behave in semi-predictable responses, i.e. they make panic buying or selling decisions. History repeats itself and the same mistakes are repeated time and time again. Technical analysis is therefore the study of human behaviour through the medium of stock pricing.

Stock markets are extremely noisy, even if there are weak under lying patterns there may be to too much noise in the system for the signals to be detected. On the other hand if the EMH is true, why are the markets so noisy, when they should reflect the current news and economic environment.

The EMH is a complex and controversial theory and a full discussion of it is beyond the scope of this thesis. This thesis will therefore only concern itself with looking for the evidence of repeatable patterns and not the EMH itself.

2.2 First Order Logic (FOL)

FOL is widely used within the AI community and overcomes some of the limitations of propositional logic. FOL describes the world in terms of objects, properties and functions and not true/false statements. The following is allowed in FOL, constants, predicates, functions, variables, connectives and quantifiers. A term is a constant, variable or a function. An atomic sentence is a predicate. Complex sentences are formed in the same manor as propositional logic by combining atomic sentences using connectives. In addition to the connectives, two quantifiers can also be used on a sentence. These are ∀xS which translates as, for all x it is the case that S, and ∃xS translates as there exists an x such that S. The descriptive power of FOL makes it a complex form of logic and not appropriate for an initial test of a theory, hence it is not used in this thesis.

2.3 Modal Logics

Modal logic refers to a group of logics which extend existing logical systems including propositional or first order logic, with concepts such as necessary and possibly. The broader umbrella term modal logics includes, Deontic Logic, Temporal Logic, Modal Logic and Doxastic Logic amongst others.

Modal uses two symbols to represent "It is necessary that.." and "It is possible that...". The operators can be expressed in terms of each other and behave in a similar manor to the ∀ and ∃ from FOL.

Propositional logic uses truth tables to evaluate atomic sentences from their symbols true/false values, this propagates upwards to complex sentences. Modal logic has possible worlds and each possible world has its own truth table were the value of a symbol may have changed, however normal propositional rules can be applied in these worlds.

Modal Logic and ANN have successfully been combined Garcez & Lamb (2006) and the concept of possible worlds attached with probabilities, lends itself to statistics and financial markets. There are many more types of logics which could be considered however from the three logic's briefly described, propositional logic is used in this thesis, due to its simplicity and existing research combining it with neural networks. Modal logic and FOL are a possible extension to this thesis.

2.4 Combining Neural Networks and Logic

There are many benefits from combining neural networks with logic programming these include; Using the parallelism of neural networks to allow fast computations; The generalisation abilities of ANN's and the ability to cope with noisy data sets. Logic programming on the other hand gives the ability to determine the reasoning of the system and define a known model of the domain. These hybrid systems fill in a gap between knowledge intensive ANNs and preprogrammed systems.

Holldobler & Kalinke (1994) proposed combining, the computation power of neural networks, with propositional logic. Using a multi-layered feed forward network, it was proved that a neural network can approximate any logic function which is associated with a logic program. Furthermore if the network is made a recurrent network, i.e. the output is connected to the input and another iteration occurs, the network will eventually stabilise. This stable position will be a solution to the logic program. Holldobler & Kalinke (1994) first proved that a hidden layer was required and that a logic program could not be approximated by a perception. A method of programming the neural network with logic sentences was then proposed with neural network using binary threshold activation functions. The method was as follows. The input and output layers are a series of binary threshold units of length 'n', where n is the number of symbols in the logic program. Each unit is given a threshold value of 0.5. Each sentence in the logic program was organised into clauses of the form $A \leftarrow L_1, ...L_k, K \geq 0$, for each clause the following procedure was followed. Add a binary unit to the hidden layer. Connect the unit to the unit representing 'A' in the output layer with a weight of 1. Next for each L (1 to K) connect the input layer representing L to the hidden layer unit with a weight of 1 if L is a symbol or -1 if L is a negated symbol. Finally the threshold value of the hidden unit is set to b 0.5, where b is the number of positive literal occurring in the clause. Holldobler & Kalinke (1994) showed that logic programs and neural networks are closely correlated. Also that logic programs can be inserted into a network efficiently. Furthermore the author states that for every neural network there exists an equivalent logic program and it should be possible to extract this program from the network.

Knowledge Based Artificial Neural Network (KBANN) is introduced in Towell & Shavlik (1994). This is a hybrid system which maps propositional logic programs into neural networks. The network is then refined using back propagation and found to out perform many alternative learning systems including standard back propagation, ID3, nearest neighbour, PEBLS, perceptron and Cobweb.

The KBANN system consists of two algorithms, the first inserts the logical program into the neural networks and the second refines the knowledge. To simplify the domain no cyclic dependencies were allowed. It is however, noted that there is no reason why this should not be attempted. KBANN is tested on two molecular biological problems, promoter recognition and splice-junction determination. With a confidence of 99.5% the results were found to be statistically significant with the exception of standard back propagation and PEBLS on the splice-junction problem. The paper states that this was probably due to the lack of understanding of splice junctions (Towell & Shavlik 1994). The background knowledge inserted into KBANN must have "sufficiently small" numbers of rules and antecedents in each rule to allow accurate encoding of the logic program.

Garcez & Zaverucha (1999) introduce the Connectionist Inductive Learning and Logic Programming System ($C-IL^2P$). To integrate the logical program to the network a similar method to that of Holldobler & Kalinke (1994) was used, however the algorithm used bipolar semi-linear neurons instead of binary threshold neurons. This meant the outputs of the neurons were real numbers from -1 to 1 opposed to either 0 or 1. This had the effect of allowing the network to learn from examples, as in Towell & Shavlik (1994). Due to the bipolar activation functions the $C-IL^2P$ system did not suffer from KBANN limitation with regards to "sufficiently small" background knowledge. Also using -1 instead of 0 to represent false, resulted in the network converging faster in almost all cases. The $C-IL^2P$ also allows for cyclic dependencies by using a recurrent network. Another important feature of the $C-IL^2P$ system was the logical program was always encoded into one hidden layer and not spread across multiple hidden layers, as in KBANN, thus reducing complexity. The $C-IL^2P$ was tested on the promoter recognition and splice-junction determination problems of molecular biology and was found to perform at least as well as KBANN but not significantly better over large data sets. With smaller data sets however $C-IL^2P$ generalised better as it converged faster.

2.5 Applications of Artificial Intelligence Techniques in Prediction of Stock Performance.

Every year more data is available in electronic form, from news stories to financial statements. This increasingly large pool of data makes it desirable to automate tracking, monitoring and in some cases decisions making. AI in the finance industry is currently being researched on many fronts.

Albanus & Batchelor (2002) used neural networks to preform a non linear Principle Component Analysis (PCA) and found non linear methods were superior to traditional PCA. The domain used to test the data "out performed" shares in the period of 1993-97 on the London Stock Exchange. A set of 38 accounting ratios were used for approximately 700 shares each year. The study found linear PCA caused massive loss of information as did a linear neural network, while the non linear neural network method was able to retain most of the information.

Nygren (2004) used a simple feed forward, back propagation network. This thesis used time series data of Ericsson B and Volvo B shares, closing highest, lowest prices and volumes traded. No further technical analysis of the stocks price was done, but several external data sources were also included, Dow Jones Stock Index, Swedish Stock Index, 3-month interest rate, 5-month interest rate and the exchange rates for the Swedish SEK/US Dollar and Swedish SEK/German DEM. Results were compared against the "naive strategy" of buy and hold. Unfortunately this biased the results. The possible gain or loss at the end of the test run was entirely based on the initial and final stock value. If the stocks happened to be overvalued at the start of the test run the algorithm would have appeared not to have performed well and vice versa. A larger and more varied data set with a rigorous statical analysis would have been more appropriate.

Lendasse et al. (2000), used non linear tools to preform a principle component analysis (PCA). Using non-linear methods to reduce the size of the input vector was a balancing act as it could result in over fitting. The paper used the largest range of variables as possible inputs then estimated the intrinsic dimension and non-linear projection to reduce them to as few as possible. The results were then used on the Bel 20 stock market index. Non-linear regression was preformed using a Radial Basis Function (RBF) network. The hidden layers had more layers than the input layer with the intention of projecting the inputs into a higher dimensional space where they might be linearly separable. A RBF model was chosen as it preforms an unsupervised clustering of data points and requires a minimum of user intervention. The variables used are not disclosed in the paper, but it was stated that a large number were required including past values of series and exogenous variables, that could influence the series. This process should also help with the generalisation properties of the model as the result should contain a smaller vector which incorporates the majority of the data's information. The Belgian Bel 20 index data was then used, 2600 daily data points over a 10 year period. The problem was defined as predicting the sign of the variation of the Bel 20, 5 days forward from the available data set (i.e. $t + 5$). PCA was carried out on 42 variables (technical indicators) and 95% of the variance was explained by 25 variables. Curvilinear Component Analysis (CCA), Demartines & Herault (1996), was used to reduce the state vector size still further and unlike Kohonen maps CCA do not make any assumptions about the projection space. This resulted in a 9 dimensional input vector. The input data was then split up into windows of sizes 500 points. This resulted in 2100 moving windows. The algorithm was then trained on each window and tested on the point directly after the last training example. 60.3% correct approximations in the training data was achieved, 57.2% in the test set. It was also found that some windows provide a higher accuracy than others. This paper acknowledged that more statical analysis needed to be preformed on the results to confirm the relevance of the results.

Continuing the work from Lendasse et al. (2000), Lendasse et al. (2001) compared PCA and CCA and found that the PCA produced slightly better results than CCA, this was attributed to a small data set and over fitting. In the domain of BEL20 Market Index comparable results were obtained using a linear projection method or a non-linear method, when a linear prediction tool was then used.

Singh & Fieldsend (2001), used a novel hybrid approach to exploit historic data. A Pattern Modelling and Recognition System (PMRS) was used. As patterns were detected information on past market behaviour was passed onto the network. The PMRS system used a nearest neighbour approach to classifying.

It selected a pattern of minimum size two. The nearest neighbour was then determined from the historic data and a forecast of high or low was predicted based on the similarity of the match found. This forecast was repeated for all patterns between two and five in size.

The neural network component consisted of two hidden layers with five sigmoid nodes in each, the networks were fully connected. The inputs to the neural network were six of the most recent lags of the time series. The learning rate of the network was set at 0.05 and momentum of 0.5. Training was done using back propagation which completed when the root mean square error fell to 0.025%. Data sets were cropped into 3 sections, 75% training data, 15% test data and 10% validations data. The best performing PMRS model with the test data was combined with the neural network, and 6 methods of error calculations were used to asses the performance.

The results were not directly comparable to other methods which use technical indicators to help the prediction. Singh & Fieldsend (2001) method was entirely based on pattern detection in a single data stream. The data sets used to test this method ranged from physiologic data to Dollar exchange rates of various currencies. Brazilian Reals to the US Dollar provided the best finical prediction of 64.4% while the Swiss Francs to US Dollar has only a 48.8% success rate, no better than chance. The approach to the problem is novel, but the statical analysis was lacking in the case of Brazilian Reals to the US Dollar to quantify the significance of the result.

Chan et al. (1999), used conjugate gradient learning oppose to steepest decent in the neural network. Back propagation is a hill climbing problem and thus it was possible for the solution to become trapped in a local minimum. Instead of using a standard randomisation of initial weights, multiple linear regression was used in an attempt to guarantee convergence. The weights between the input layer and the output layer were still randomised, the hidden layer and output layer were not however. Chan et al. (1999) used data from 11 companies from 1994-1996 collected from the Shanghai Stock Exchange. Ten technical indicators were selected as inputs to the neural network, no reason for these indicators was given. These were three exponential moving averages, relative strength index, MACD, MACD signal line, and 2 stochastic indicators (an oscillator that tracks close price to a recent high, low). The data sets were normalised from 0.05 to 0.95 to avoid neural network problems at end of range points. The learning rate was set to 0.1 and the momentum set to 0.5. Training was terminated when the mean square error was less than 0.5%. The first 500 values were used as the training data for each of the 11 companies. The last 150 points of data were used as the testing data. The correct direction was predicted 69% of the time.

Hui et al. (2000) provided a detailed description a hybrid system used, however due to a lack of formal analysis of the results, performance is unknown. The hybrid used an unsupervised Kohonen network to classify into clusters and also to provide a Guidance Factor to a Multilayer Perceptron.

Chan (2004) used a novel approach of a sliding window of the last 30 days. The last 30 days of a normalised share or index values were fed into the network and it was expected to predict the value on the 31st day. Once again no further technical indicators were used. The topology of the network was 30-8-8-1. The length of time for training was not detailed but the test data length was 300 days. The windowing technique was interesting, however using straight normalised stock prices is flawed. The neural network was not being “shown” any ratio’s or information regarding direction of the share price and was not therefore be able to generalise given new information. The general trend of stocks was increasing in value, as the network was trained on historic data and tested on more recent data the network will have not seen sufficient to be trained (i.e. the network maybe trained in the region of 0 to 0.4 but not when the inputs are 0.6 to 0.9).

Reviewing literature into this topic has shown that the quality of some of the research papers is substandard, some lack a detailed analysis of results. It has proven hard to obtain research of the required calibre, this might be due to the secretive nature of the topic.

Keogh & Lin (2003) focused on current methods of clustering of time series data and finds them to produce no “useful” results. That is the result data is independent of the input data. Sub-sequence Time Series (STS) clustering used a sliding window and preformed a clustering method on the data within the window. The paper initially focused on K means (using euclidean geometry as a distance measure) and stock market data (Poor’s 500 Index closing values). Random walk data was used as a control. The paper found no statistical significance in the clustering results, produced with the stock market data or random walk data. Hierarchical clustering was then also tested and the results found to be the same.

To verify the result was not due to the stock market data but the sliding window feature extraction method, the same experiment was performed on UCR ocean and buoy sensor data. The data sets were chosen due to their dissimilarity, however after analysis there were no discernible difference. A wide variety of tuning parameters were used in attempt to improve the results, but failed.

The paper was unable to provide a concrete explanation as to why STS fails to produce meaningful results. It was suggested STS clustering returned only a “set of basis functions that can be added together in a weighted combination to approximate the original data”. Finally an algorithm that produces meaningful results in some data sets was presented. The algorithm mines motifs, reoccurring sequences of discrete strings, and then a clustering algorithm on the motifs.

2.5.1 Agents - Q-Learning

Q-Learning developed by Watkins (1989), allows the optimal solution to be discovered and is computed with the following iteration; Initialise all state action values, Q values; Repeat the following until termination criteria is reached; Choose an action from the current state using a policy derived from Q values; Take the action and observe the returned value and new state; update the Q values according to equation 2.1.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma max(a)Q(s_{t+1}, a) - Q(s_t, a_t)] \tag{2.1}$$

Where $Q(s_t, a_t)$ is the Q value of state “s”, action “a” at time “t”. α is the step size parameter and γ is the discount rate. The advantage of Q learning is that it guarantees convergence to the optimal policy Q^*, and can find solutions not easily found using other methods for example monte carlo. Disadvantages are that in real world examples the state action sets can often be too large to allow convergence within acceptable time frames.

Kjall-Ohlsson (2005) modified the Q-learning algorithm (Watkins 1989) by incorporating updated physiological models on animal learning. The following four problems were examined: exploration versus exploitation, temporal discounting, generalisation and large-sized problems. The thesis proposed that by utilising the Rescorla-Wagner equation for classical learning and incorporating it into the Q-learning frame work, that all four research questions could be addressed.

To test the success of the developed algorithms, the Q-learning's performance was used as a benchmark in the following evaluation criteria; Guaranteed convergence to the global optimum solution; Speed of convergence; The ability to generalise.

The new algorithms were developed by placing the emphasis of evaluating the reward on the agent and not on the environment. Introducing internal drives to the agent and ensuring the agent approached an appetite stimuli and avoided aversive stimuli achieved this. Two new algorithms Trace and Lookahead were presented and tested within a grid environment in which the agent was run in three experiments.

Of the four research questions asked, the thesis concluded the following. Exploration-exploitation equilibrium, Stimulus $\rightarrow$ Stimulus (S $\rightarrow$ S) associations were incorporated successfully into a Q-learning frame work model, however implementing exploration as a drive was not achieved. It was also determined that the S $\rightarrow$ S associations evolved in exactly the same way as the Stimulus $\rightarrow$ (Response $\rightarrow$ Outcome) (S $\rightarrow$(R $\rightarrow$ O)) associations and therefore did not help in discounting unsuccessful exploratory options. Temporal discounting remained a problem, as the presented algorithms contained the equivalent of a Q-value error correction rule. Generalisation was achieved by implementing states as reducible entities and modifying the error correction rule. When states were reached which had similar entities to previously visited states the algorithms were able to utilising this knowledge. Larger-sized problems remained an issue, but improvement could be gained if the agent can generalise from previous smaller sized problems.

The Q-learning algorithm was used as a benchmark for the Trace and Lookahead algorithm and the evaluation criteria used were therefore correct, for the following reasons. Guaranteed convergence to the global optimum solution, this was a minimum requirement as the Q-learning algorithm has been shown mathematically to convergence to the global optimum solution (Watkins 1989). The author used deductive reasoning to explain that the ARL lookahead algorithm is guaranteed to converge under the given experimental conditions. This was then confirmed by the observations. ARL trace algorithm was found not to converge and was also explained via deductive reasoning.

Speed of convergence was used to directly rank the new algorithms against Q-learning. All algorithms discovered the optimum policy but at different rates. The learning rate for Q-learning, α, was set to 0.01 for all experiments (based on the ARL parameters α * β). This had a direct influence on the convergence rate. It was therefore not appropriate to compare the speed of convergence for the new algorithms with Q-learning. The default parameters may have unintentionally resulted in one algorithm being more optimised for the problem than the other. The optimal parameters should be found for each algorithm independently and the results of these runs compared.

The ability to generalise, the Q-learning cannot generalise, therefore any ability to generalise will be an improvement. The author employs deductive reasoning to show that the algorithms are capable of generalisation. Each experiment was run eight times and a hypothesis was formed on which to base an 'independent t test'. The statistics from the tests were then used to prove or disprove the hypothesis. A larger sample would have been preferable.

2.6 Design of an AI system

The representation of the domain to any AI system is essential. If poorly thought out the representation can prevent effective learning. An excellent example of a well designed ANN is by Tesauro (1990) and Tesauro (1994) where a simple but clever representation was used to present the state of a backgammon board to the network. There are 24 positions on a backgammon board, each position has 4 input nodes assigned to it for each player. There are two players, white and black. Consider from the perspective of the white player, if there was 1 white piece in a position the input was set to 1. For 2 pieces the input nodes 1 and 2 are set and 3 pieces the first 3 nodes are set to 1. Any more than 3 pieces the fourth node takes on the value (n 3) / 2 where n is the number of pieces. A similar encoding was used for black, this gives 192 input nodes. Two more nodes are used to encode the number of pieces on the bar for both black and white using the values n / 2. Another 2 nodes to indicate the number of pieces successfully removed from the board and the final 2 nodes to indicate the who turn it was, in a binary fashion. The best playing strategy depends heavily on how many pieces each players has on any one position. After the 4th piece the importance of information regarding more pieces at this position to the strategy becomes less relevant and thus it was not given a boolean value, but a strength indicator. This simple encoding allowed TD-Gammon to perform at near grandmaster level.

Most AI techniques employ some sort of high climbing algorithm, when searching the domain space it is important therefore that the algorithm is presented with a smooth search space with as few local minima as possible.

No examples of using financial based boolean inputs to a Artificial Neural Network for predicating the stock market have been located during the literature review process.

Chapter 3

Method

A large amount of time series share price data was needed. It was therefore decided to use the companies listed in FTSE100, as the data is freely available from numerous sources and provides information on 100 large stable companies.

3.1 Data Sets Inclusion / Exclusion Criteria

The raw stock data was pulled from the Yahoo! finance website Yahoo! (2006). Yahoo! (2006) provided the following data for all FTSE100 companies; date, open, high, low, close, volume and adjusted close. The adjusted close was discarded as this information is not available in real time. The following conditions were imposed to ensure the data's integrity.

- For each company the start date of the share information was variable depending on its inclusion into the FTSE100 index. To ensure enough data was available for trading decisions to be made, a minimum of 3 years was set. As the ANN were to be trained to look at middle to long term trends, it was thought any shorter interval would not provide enough time for the trader's decision to average out.
- Each share must have continuous data. There could be no breaks in the share's pricing information, eg. several days worth of data missing. This would have broken the condition that each price followed from the previous and was related.
- Any shares that had their stocks split during the time period which the data sets were used were excluded. This would have broken the condition that each price follows from the previous and is related.

From 100 companies, 58 fulfilled the inclusion criteria, these where split evenly into two groups, 29 for training the ANNs and 29 for testing the ANNs. Where the companies were referenced, the Yahoo! abbreviations are used instead. The appendix 7.2 contains the companies full name and abbreviations conversions.

3.2 Financial Technical Rules and Symbols

The next step was to determine the appropriate technical indicators, financial rules and logical symbols to be used. Once a technical indicator had been accepted the parameters used to create the indicator were defined, where appropriate default values were used. There were many possible technical indicators that could have been used, however resource limitations meant that it was not possible to use all of them. For each indicator there were also many interpretations of how they should be used, the simplest interpretation with minimal rules and symbols was chosen. To simplify the domain no cyclic dependencies in the rule set were allowed. The following items were considered when selecting indicators.

- A wide variety of signals and indicators must be used which are not based on the same raw data. For example if all chosen technical indicators use the closing price when generated it is not surprising if all indicators signal the same action.
- Indicators with minimal number of variables, to limit the number of arbitrary choices.
- Indicators that were known to work best in the medium time frame, the chosen time frame for the ANNs and trending markets.
- Ease of implementation.

The following technical indicates were included in the study. In the following definitions subscript “t” refers to time and subscript “t-1” refers to one time step earlier. Subscript “n” refers to the time period. A complete description of all symbols and rules used is given in the appendix 7.3.

3.2.1 Three Moving Average System (TMA)

The TMA used three separate moving averages on different time scales, slow, medium and fast. Trading signals were generated at cross over points of the moving averages. Moving averages (MA) were calculated as simple moving averages (SMA), exponential moving averages (EMA) or weighted moving averages (WMA). Figure 3.1 shows the three moving system being used on the SCTN.L. The SMA was calculated as follows;

1.
$$SMA_t = \frac{\sum_{k=1}^{n} Close_{t-k}}{n} \tag{3.1}$$

The EMA is calculated as follows;

1.
$$EMA\% = \frac{2}{n+1} \tag{3.2}$$

2.
$$EMA_t = (Close_t * EMA\%) + (EMA_{t-1} * (1 - EMA\%)) \tag{3.3}$$

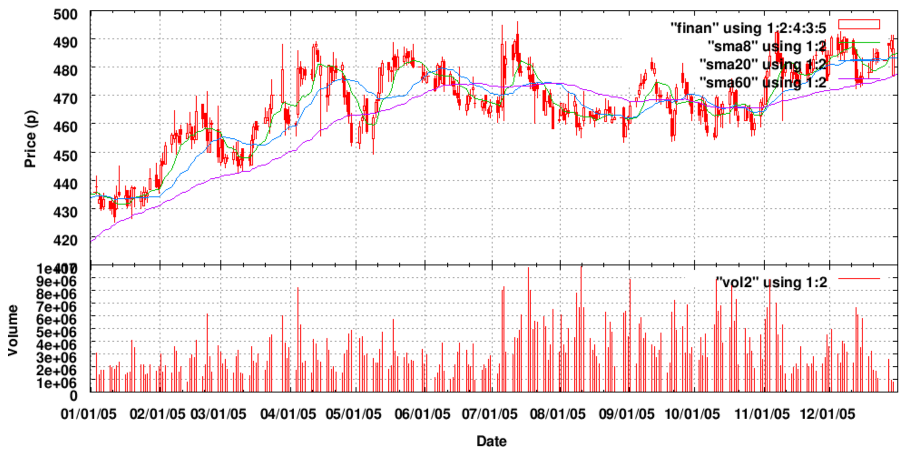

Figure 3.1: Three moving average, share SCTN.L

Symbol	Definition	Justification
A	medium MA > slow MA	n/a
B	fast MA > medium MA	n/a

Table 3.1: 3 Moving average symbol definitions

The trading rules of the three moving average system were as follows; Go long if the medium MA crosses to above slow MA from below and fast MA is above middle MA; Close long when fast MA crosses to below medium MA average from above; Go short when medium MA crosses to below slow MA from above and fast MA is below medium MA; Close short when fast MA crosses to above medium MA from below. Table 3.1 gives the break down of the rules to two logical symbols.

The following time periods were selected for the slow, medium and fast moving averages. 200 days or 40 weeks moving averages are popular amongst analysts for tracking longer cycles. 20 to 65 days or 4 to 13 weeks moving averages for medium cycles. 5 to 20 days for short cycles. This thesis uses 200 days for a long moving average, 42 days for a medium moving average and 13 days for a short moving average.

3.2.2 Moving Average Convergence Divergence (MACD)

The MACD is the difference between two moving average lines of the closing price, and was invented by Appel (1985). An EMA of the MACD was also computed and is known as the signal line. Figure 3.2 displays the computed MACD and signal line.

The trading rules of the MACD system were as follows; If the MACD is flat or stays close to the zero line, the market is ranging and signals are unreliable; If the market is trending then, go long when the MACD line crosses the signal line from below; If the market is trending then, go short when the MACD line crosses the signal line from above. Table 3.2 gives the break down of the rules

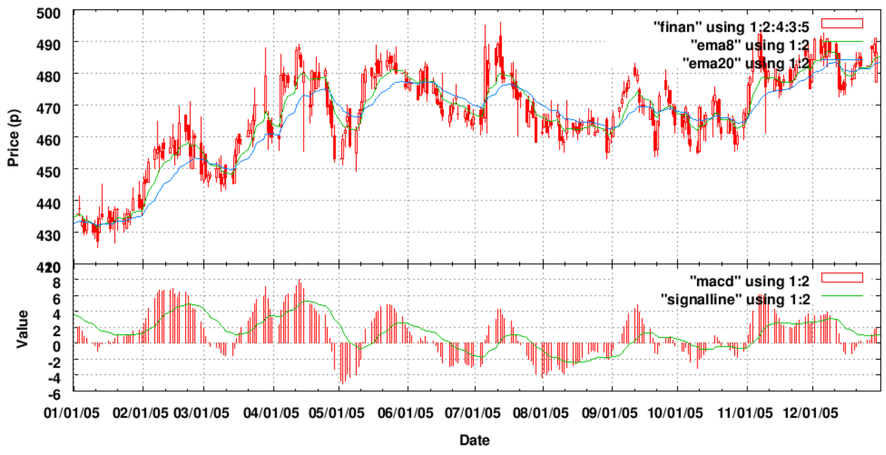

Figure 3.2: MACD, Graph 3.2.A shows the share price and two moving averages, graph 3.2.B displays the MACD and MACD signal line.

Symbol	Definition	Justification
A	$\|MACD\| > \% * Close$	trending/ranging
B	MACD > signal line	signal

Table 3.2: MACD symbol definitions

to two logical symbols.

3.2.3 Volume Oscillator (VO)

The calculation of the VO is similar to that of MACD except traded volumes were used. The VO was used to measure the strength of the signal generated but does not directly provide long or short signals. A positive VO signals a strong trend, where as a negative VO signal a weak trend.

The trading rules of the volume oscillator system were as follows; If the fast MA is above the slow MA the oscillator will be positive; If the fast MA is below the slow MA then the oscillator will be negative. Table 3.3 gives the break down of the rules to one logical symbol.

3.2.4 Bollinger Bands (BB)

Bollinger (2001) invented BBs. These were used to indicate break outs when shares were ranging. BBs are plotted as two lines either side of a moving average

Symbol	Definition	Justification
A	VO > 0	n/a

Table 3.3: VO symbol definitions

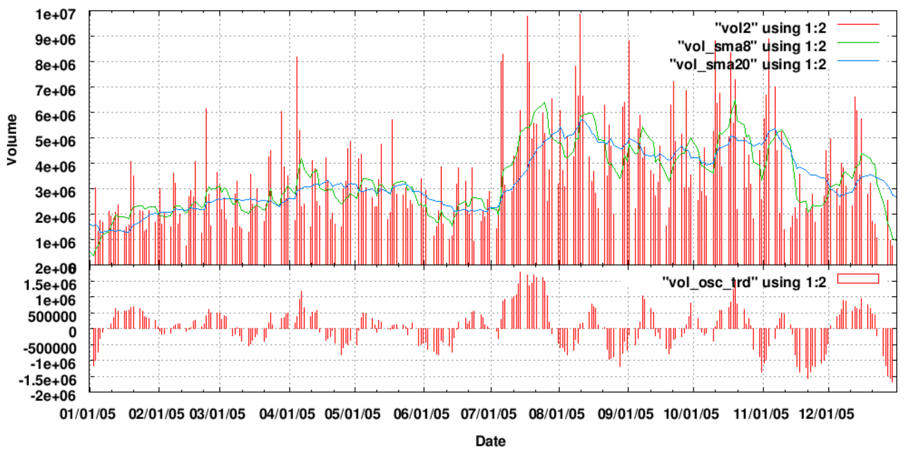

Figure 3.3: Volume Oscillator is seen in 3.3.B and fluctuates about the zero line.

Symbol	Definition	Justification
A	upper BB $<$ close	n/a
B	lower BB $>$ close	n/a

Table 3.4: BB symbol definitions

see figure 3.4. Bollinger originally used a 20 day SMA with the bands set at 2 standard deviations and stated this time period was suited to medium cycles.

$$\sigma = \sqrt{\frac{\sum_{j=1}^{N}(X_j - \overline{X})^2}{N}} \tag{3.4}$$

Equation 3.4 gives the standard deviation calculation for BB. Share prices climb up and slide down BBs in a breakout Bollinger (2001). There are no simple trading rules for BBs. Due to their importance in determining break outs, table 3.4 defines two logical symbols.

3.2.5 TRIX Indicator

The TRIX is calculated as follows.

1. EMA1 is calculated using the closing price
2. EMA2 is calculated on EMA1
3. EMA3 is calculated on EMA2
4. The TRIX is then calculated as;

$$TRIX_t = \frac{EMA3_t - EMA3_{t-1}}{EMA_{t-1}} \tag{3.5}$$

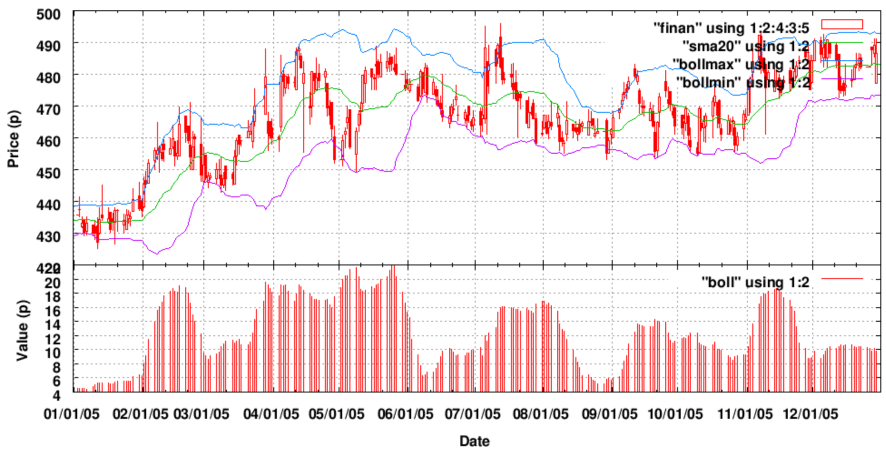

Figure 3.4: Bollinger Bands: Displays BB at 2 standard deviations about a SMA of 20 days.

Symbol	Definition	Justification
A	TRIX > 0	n/a
B	TRIX > signal line	n/a

Table 3.5: TRIX symbol definitions

As with the MACD, a signal line is computed to help reduce false signals. The trading rules of the TRIX system were as follows; If the TRIX crosses the signal line from below and the TRIX is greater than zero then go long; If the TRIX crosses the signal line from above and the TRIX is less than zero then go short. Table 3.5 gives the break down of the rules to two logical symbols. Figure 3.5 illustrates an example of the TRIX.

3.2.6 Directional Movement (DM)

The DM invented by Wilder Wilder (1978), attempts to check if the market is trending before providing a signal. The DM creates 3 indexes, ADX, +DI, and -DI and. It is computed as follows;

1. If price has increased, then $+DM = High_t - High_{t-1}$
2. If price has decreased, then $-DM = Low_{t-1} - Low_t$
3. Calculate True Range (TR), which is the highest value of, $High_t - Low_t$, $High_t - Close_{t-1}$, $Close_{t-1} - Low_t$
4. +DM = EMA of +DI
5. -DM = EMA of -DI
6. TR = EMA of TR
7. $+DI = \frac{+DI}{TR}$

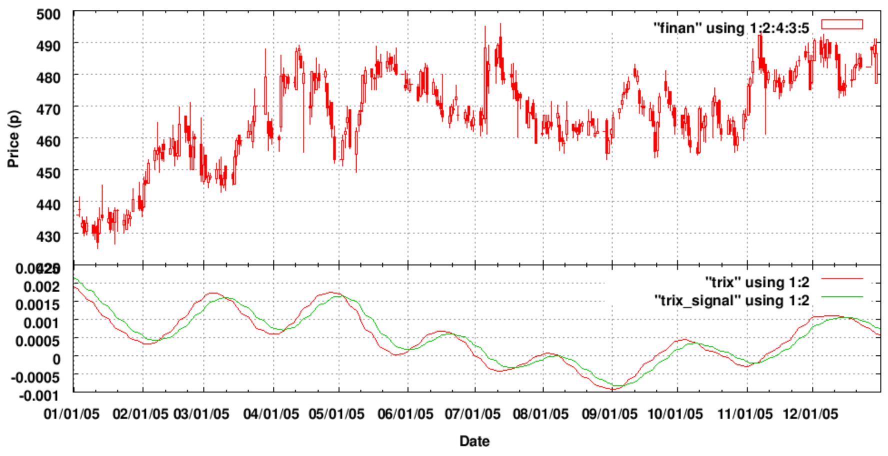

Figure 3.5: TRIX: top SCTN.L share price, bottom trix and signal line

Symbol	Definition	Justification
A	+DI > -DI	n/a
B	ADX > +DI	n/a
C	ADX > -DI	n/a

Table 3.6: DM symbol definitions

8. $-DI = \frac{-DI}{TR}$
9. DI = |+DI - -DI |
10. ADX = EMA of DI

$$MA_t = \frac{1}{n} * Value_t + \frac{n-1}{n} * MA_{t-1} \tag{3.6}$$

The moving average used in DM was computed slightly different to that of the standard EMA and is given by equation 3.6. The ADX indicated if the market was trending or ranging. When the ADX fell below both the +DI and -DI the market was ranging. When the ADX was above both +DI and -DI the market maybe becoming over heated. The +DI indicates the strength of the upward trend and -DI the strength of the downward trend.

The trading rules of the DM system, based on Elder (1993), are as follows; If +DI is greater than -DI and the ADX is increasing while the +DI and the ADX are above the -DI, then go long; If the +DI is greater than the -DI and the ADX turns up while below the +DI and the -DI, then go long; Exit when the +DI crosses below the -DI; If the -DI is greater than the +DI and the ADX is increasing while the -DI and the ADX are above the -DI, then go short; If the -DI is greater than the +DI and the ADX turns up while below the +DI and the -DI, then go short; Exit when the -DI crosses below the +DI. Table 3.6 gives the break down of the rules to three logical symbols.

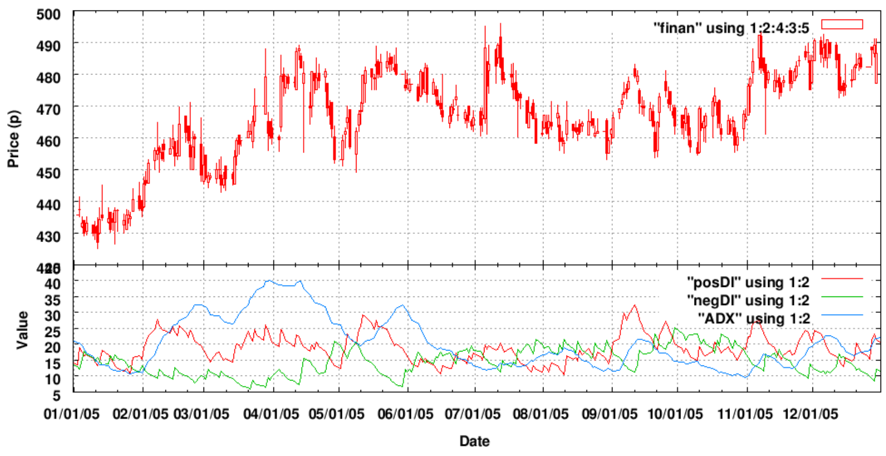

Figure 3.6: Directional Movement: top SCTN.L share price, bottom ADX, +DI, -DI

Symbol	Definition	Reason
A	ema8 > ema20	check for raising OBV

Table 3.7: OBV symbol definitions

3.2.7 On Balance Volume (OBV)

Developed by Granville Granville (1976), OBV measures the weight behind price movements and is calculated as follows:

1. If $Close_t$ is greater than $Close_{t-1}$ then $OBV_t = OBV_{t-1} + Volume_t$
2. If $Close_t$ is less than $Close_{t-1}$ then $OBV_t = OBV_{t-1} - Volume_t$
3. If $Close_t$ equals $Close_{t-1}$ then $OBV_t = OBV_{t-1}$

In a ranging market, rising OBV signals an upward breakout, while decreasing OBV signals a downward breakout. In a trending market the OBV can signal the market top or bottom. Similarly to Bollengier Bands there are no simple trading rules directly associated with the OBV. Table 3.7 gives the OBV symbol definition used in this thesis.

3.3 Generating Back Propagation Training Data

Once the input data of boolean lists was created, the training data for the back propagation algorithm was generated. Four potential methods of generating this were considered.

3.3.1 Human Expert

Using an expert to analyse several historical shares was considered. It was not however possible to get an individual with the required experience and

Date	Input Data	State no.	Price	Training Data
10/05/06	1 1 1 1 -1 1	1	100.0	
11/05/06	1 1 1 1 -1 1		105.0	
12/05/06	1 1 1 1 -1 -1	2	107.0	1
13/05/06	1 1 1 1 -1 1	1	95.0	-1
14/05/06	1 -1 1 1 -1 1	3	97.0	1

Table 3.8: Example of input data relating to state changes, 1 state ahead.

availability to participate in this thesis. A further problem is that humans are unreliable and do not repeatably reach the same conclusion from a given set of parameters. This inconsistency causes significant problems. It was therefore desirable to either generate a training data set which was consistent and reliable or use an AI technique which did not require supervision.

3.3.2 Shifting Historical Data

This was the simplest and fastest attempt to create reliable training data. The closing price from the historical data was taken and the difference between the current price and the 20 day future price was calculated. If the difference was positive, +1 was assigned to the training data and -1 to a negative difference.

3.3.3 Shifting Historical Data Based on a Change in State

The method described in section 3.3.2 was adapted further to forecast on a change in state. The input data at any time period was a vector of boolean values indicated by 1s and -1s. Within the input data list there may be many time steps where the state of the system does not change. For example table 3.8 shows a list of input data, at date 12/05/06 the input vector has changed and therefore the state. The change of price is computed and the training data is assigned the value 1. In the following time step another change in state has occurred, the difference is negative and so -1 is assigned to the training data.

The training data is not dependant on a fixed time step anymore, however it is dependent on a fixed number of states in the future.

3.3.4 Reinforced Learning, Q-Learning

Finally a training data set was produced via the reinforced Q-Learning algorithm. The inputs to the network were a simple set of binary input and thus for every possible input to the network there exits a finite state number. The total number of available states is 2^n where “n” is the number of input symbols to the network. Using the Q-Learning algorithm an optimal trading policy can be calculated. This method should not be dependent on a fixed time frame or a fixed state number.

The reward function of the Q-Learning algorithm was set as;

- If a trade has not been completed then $Reward = 0.0$
- If a trade has been completed $Reward = \frac{sellprice - buyprice}{buyprice}$

Several experiments where run with varying learning rates and discount rates to discover which allowed the most efficient learning. The egreedy and softmax exploration policies where tested with varying parameters.

3.4 Determining the Training Data's Effectiveness

Once the training data had been computed it was necessary to determine which was the most effective training data. The boolean lists were used by a "dumb trader" to generate a best case profit scenario. The training data sets were compared against a basic buy and hold strategy, the shares were brought at the beginning of the time period and sold at the end. To test the training data provided useful trading information a paired t-test was done against the buy hold strategy. If the data sets were not statically significant then the data was not used. By using a paired t-test it was assumed that the generated "profit" over all shares would form a Gaussian distribution.

The t-test checked that the "profit" generated through passing the share information through a function was not a chance occurrence causes by choosing a subset of the population, the null hypothesis.

3.5 Generating the Artificial Neural Networks From Logic Programs

To generate the ANNs from the logical program the method presented in Garcez & Zaverucha (1999) was used. First the rules were inserted into a knowledge base as a set of clauses in the form $A \leftarrow L_1, ...L_k$. The conversion of these logical symbols into a clause of this form is best illustrated with an example. Consider the MACD and MACD signal line, section 3.2.2. Using the financial rules presented and the symbols defined we can define the logical rules as;

$$Short \leftarrow A, !B \tag{3.7}$$

$$Long \leftarrow A, B \tag{3.8}$$

Equation 3.7 states that a short signal is generated when the market is not ranging and the MACD index is less than the signal line. Equation 3.8 states that a long signal is generated when the market is not ranging and the MACD index is greater than the signal line.

All the logical rules from section 3.2 were inserted into a knowledge base in this manor. See appendix 7.3 for the complete knowledge base. The knowledge base was then used to create an ANN. This was handled by the "Neural Network Factory" library.

Each clause used one neuron in the hidden layer to map the inputs $L_1, ...L_k$ to the output A. The algorithm was as follows;

1. Calculate $a = max_{knowledgebase}(k_1, ..., k_q, \mu_1, ..., \mu_q)$, the largest number of all k's and μ's in the logic program. Where k was the number of literals in

the body of the clause and μ was the number of clauses in the knowledge base with the same head.

2. Calculate $A_{min} > \frac{a-1}{a+1}$
3. Calculate the weight $W \geq \frac{2}{\beta}.\frac{ln(1+A_{min})-ln(1-A_{min})}{a(A_{min}-1)+A_{min}+1}$
4. Generate a network of size "a - b - c" where "a" is the input layer and is equal to the number of unique symbols in the body of the clause. "b" is the number of clauses in the knowledge. "c" was the number of unique symbols in the heads of the clauses. Assign each node in the input and output layers a symbol.
5. For each clause in the knowledge base; Connect the node to the symbols in the input layer which are in the body of the clause with a weight "W" for a positive connection and "-W" for a negative connection. The hidden node is then connected to the symbol in the output layer corresponding to it symbol at the head of the clause with a weight of "W".

$$h(x) = \frac{2}{(1 + e^{-\beta x})} - 1 \tag{3.9}$$

6. A standard linear activation function was used in the input layer, while equation 3.9 gave the semi linear activation function used in the hidden and output layers.
7. Finally connections not explicitly mentioned where set to zero.

3.6 Stock Market Simulator (SMS)

The SMS provided an interface for a "trader" to buy and sell shares on the historical data. The following restrictions were applied.

- Only a single transaction could occur at any one time.
- A trader could not take a short position unless it had already purchased, effectively it could only sell, if it had already brought a share.
- Transaction fees and the spread were not included in the calculations, as statical significance was being tested and not financial viability.
- Any unfinished trades at the end of the data set were not counted in the total profit for that share.

An iteration within the SMS only occurred when there was a change in state of the input data and not a time-step. Therefore there may have been multiple time-steps to each iteration of the algorithm. This served two purposes, the learning became more CPU efficient as the input data sets were reduced in size and secondly the ANNs was not repeated shown the same state, over representing it. Two extra states where prepended to the input vector, the long and short status of the trader.

3.6.1 Back Propagation ANNs

Only one case was tested, the ANN without predefined rules. It was decided to use the training data generated from the shifting of historical data based on state, see section 3.3.3, as this produced the best training data, section 4.1.2. The back propagation parameters where set as loops = 10,000, step size = 0.0001, momentum = 0.0.

3.6.2 Evolving ANNs

Starting with 100 randomly initialised ANNs, the candidate solutions, each candidate was run through the SMS and ranked according to its generated "profit". The selection criteria was the top ten performing ANNs were kept in the pool and the other 90 were discarded. The next generation of candidate solutions were generated from the parent 10 ANNs using a simple shift function which modified a real value by a small percentage. This shift function was randomly applied to the weights in the ANN. The process was repeated until no further improvements were detected in the last five generations.

3.7 Testing The Trained Networks

A similar test was used for the trained networks as for the validation of the training data. Once an ANN was fully trained either via back propagation or an evolutionary approach, the ANN was run with the testing data. The generated profits were compared against a basic buy and hold strategy and a paired t-test was applied to check for a statical significance.

3.8 Development Environment

The Stock Market Simulator (SMS) was developed within the following environment and utilised the given libraries and languages.

- *Operating System, Linux(2.6.17-10)* Chosen for it stability as a development platform.
- *STD C++ libraries* Chosen for its portability between platforms and its wide use as freely available standard.
- *GNU build tools, compiler, linker* Chosen for price.
- *PHP* Chosen for ease of development in creating scripts to interact with the Internet.
- *MySQL Database* Chosen as it is free to use, easy to manipulate, fairly reliable.
- *Assortment of open source libraries, libmysql, libcrypto, libuuid* Required to interact with various components, all open source and freely available.

3.9 System Overview

Figure 3.7 displays the system layout and the following steps were taken to operate the SMS.

1. Run the PHP scripts. This fetched data from the internet and inserted the raw stock data, open, close, high, low and volume into the local database.
2. Run the Data Generator. Using the raw data held in the local database relevant technical indicators were computed which were then incorporated with the logical symbols to produce boolean lists. The lists were then stored in the local database.
3. The three trader programs (future, optimal and state) were then run in any order to generate the training data sets.
4. Finally the SMS was run and depending on the configurations can train or test the neural networks.

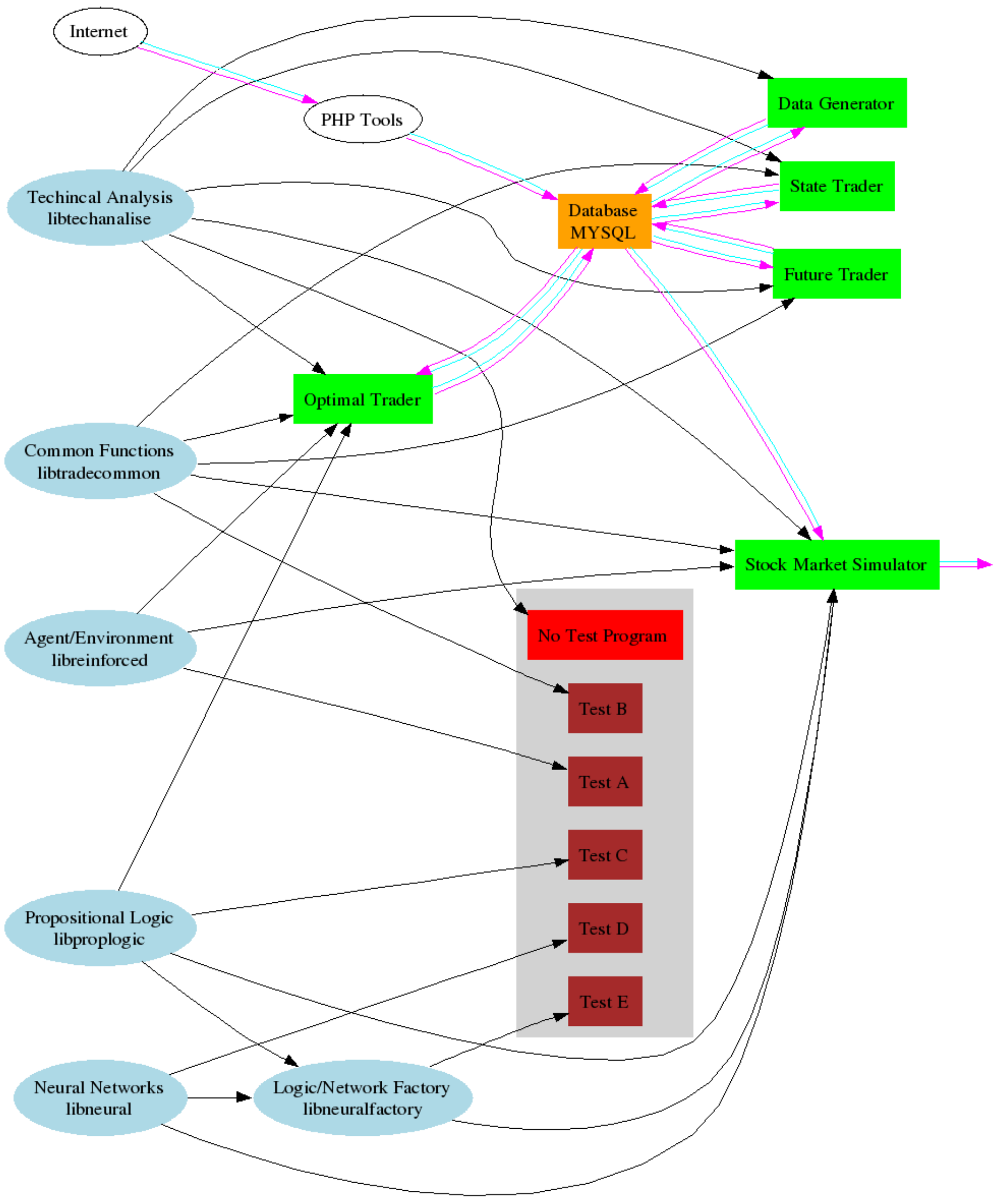
Internet
PHP Tools
Data Generator
State Trader
Database MYSQL
Future Trader
Techincal Analysis libtechanalise
Optimal Trader
Common Functions libtradecommon
Stock Market Simulator
Agent/Environment libreinforced
No Test Program
Test B
Test A
Test C
Test D
Test E
Propositional Logic libproplogic
Neural Networks libneural
Logic/Network Factory libneuralfactory

Figure 3.7: System layout, black lines indicate library dependencies, multi-coloured lines indicate the data flow. The green boxes are the executable programs and blue ovals are libraries.

Chapter 4

Results

4.1 Training Data

In this section the results for computing the training data are presented. Please note that in each figure of Q values variance, each data point is an average of 1000 iterations of the Q learning algorithm. Whereas, plots of "profit" are generated after each iteration of the entire data set.

4.1.1 Shifting Historical Data

For the full list of "profits" generated on a per company basis using the 20 day shift strategy see table 4.7.

Group	Buy Hold	20 Steps
Mean	233.3183	1383.7010
SD	569.9726	697.7959
SEM	105.8413	129.5774
N	29	29
P		< 0.0001

Table 4.1: Results from paired t-test using simple shift strategy with the training data.

A summary of the paired t-test results is shown in table 4.1. The P value indicates with a 95% confidence level there is a statical significant difference between the buy and hold strategy and the 20 day shift strategy. This method of generating training data was therefore successful.

4.1.2 Shifting Historical Data Based on State Change

For the full list of "profits" generated on a per company basis using the 1 state ahead strategy see table 4.7. Table 4.2 gives the "profits" at 5 different states ahead.

The 1 state ahead strategy produced the highest returned average profit and was therefore chosen as a possible training data set.

States Ahead	Average Profit
1	4157.25
2	2931.95
3	2451.8
4	2134.12
5	1924.81

Table 4.2: Average profit when trading on state changes with a fixed state ahead strategy.

Group	Buy Hold	1 state
Mean	233.3183	4157.2466
SD	569.9726	2094.9672
SEM	105.8413	389.0256
N	29	29
P		< 0.0001

Table 4.3: Results from paired t-test using the 1 state ahead strategy with the training data.

Table 4.3 provides a summary of the paired t-test results. The P value indicates with a 95% confidence level there is a statical significant difference between the buy and hold strategy and the 1 state ahead strategy. It is therefore possible to use this method to train the ANNs.

Further analysis of the state training data was done to examine possible ambiguities in the training data see section, 5.3 for discussion. The signal strength at each state was computed and is given by equation 4.1.

$$SignalStrength_{state} = \frac{BuySignals_{state} - SellSignal_{state}}{BuySignals_{state} + SellSignal_{state}} * 100 \tag{4.1}$$

Figure 4.1 displays the signal strength of the buy and sell signals at all states. Positive signal strengths indicate a buy signal while negative indicates sell signals. Signal strengths about the zero line indicates a very weak and unreliable signal where as large divergence from the zero line indicate a strong signal. States with zero hits are not shown. At position “y” there is a strong sell signal, indicated by the large amount of hits and a large negative value, while “x” indicates quite a strong buy signal. Three lines are plotted to show the signal strengths with hits greater than 0, 20 and 50. The majority of the signals have a low hit value, plotted in blue, and therefore have a low reliability. Note, A 50% signal strength does not indicate that the signal is correct 50% of the time, see the signal strength equation 4.1.

4.1.3 Reinforced Learning

Initial test runs were used to decide the best learning policy. From figure 4.2 it was noted that the softmax policy, while it did converge, was unable to produce

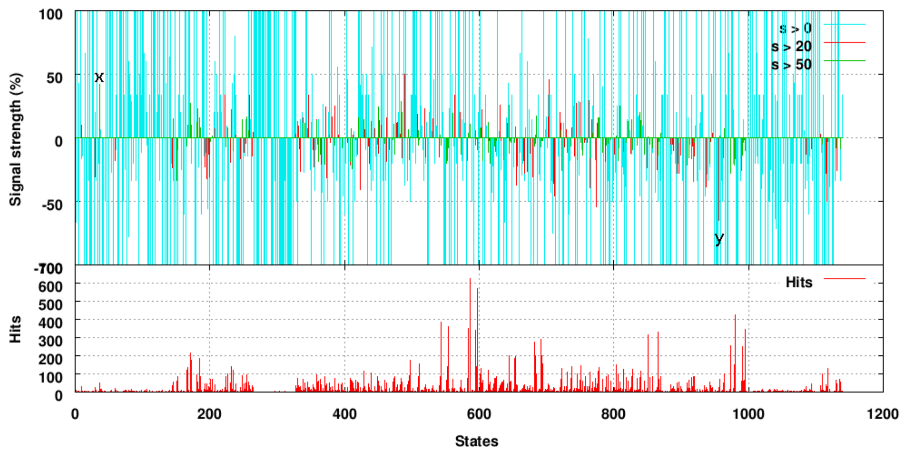

Figure 4.1: States signal strength.

results close to that of the basic buy and hold strategy. The softmax policy was therefore discounted and the egreedy method used.

Figure 4.3 displays the variance of the Q-values as a function of iterations across varying learning parameters. Figure 4.4 gives the average returned "profit" across all the stocks in the training data set. The ϵ which balances the exploration versus exploitation was kept low, as there were only three possible actions at all states, buy, hold, or sell. α, γ the step size and discount parameters were also kept low as a large number of iterations involved over what is a small action state space, this should result is many small changes.

Variance in Q-values were used to determine that the agent was converging while the "profit" was used as the measure of success in the trading strategy.

These showed that runs with high ϵ values tended to preform considerably worse than lower values, so several longer runs with extremely low ϵ values where done. The run times of the tests became unfeasibly long preventing much experimentation, the longest runs looped over 100,000,000 times.

The best optimal run was achieved with the parameters settings as $\epsilon = 0.002$, $\alpha = 0.1$ and $\gamma = 0.1$.

Group	Buy Hold	Optimal
Mean	233.3183	279.1234
SD	569.9726	462.3813
SEM	105.8413	85.8620
N	29	29
P		0.3610

Table 4.4: Results of paired t-test using optimal trader strategy with the training data.

Table 4.4 provides a summary of the paired t-test results and the P value indicates with a 95% confidence level there is not a statically significant difference between the buy and hold strategy and the optimal trader strategy. This

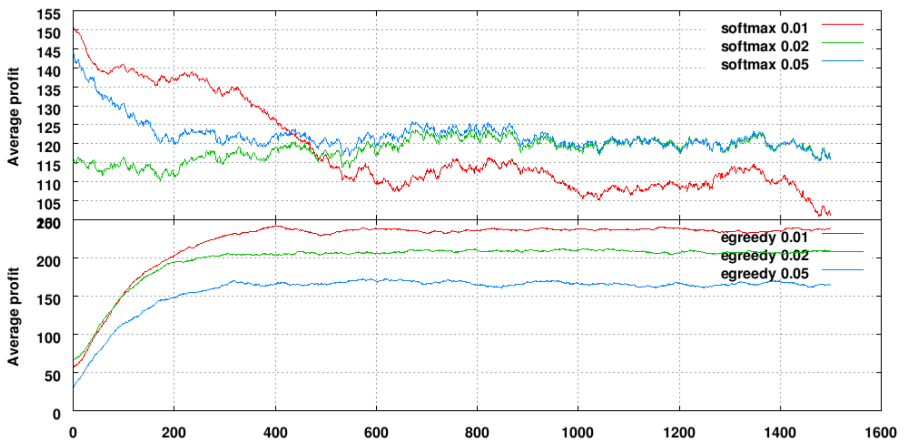

Figure 4.2: Profits with egreedy and softmax learning policies, smoothing used with $\lambda = 0.99$.

data was not therefore used.

4.2 Effectiveness of Training Data Generation

The 1 state ahead strategy produced greatest profit and was therefore chosen to train the ANNs in the following sections.

As the optimal strategy was not able to provide sustained high return values statically different to the buy and hold strategy, the inputs into the neural networks where doubled up, see section 5.3 for the full explanation. The second group of inputs contained the data from the previous iteration. Unfortunately the doubling of the input arrays meant that the financial rules no longer had any relevance to the design of the ANN. This test was therefore dropped from the project.

4.3 Training the Artificial Neural Networks

4.3.1 Back-Propagation Symbols Only

The average profit returned is 95.7p, this is below the basic buy and hold strategy which produces a 210.3p profit, see table 4.8. As it was not possible to get the ANN to successfully trade on the training data, it was applied to the testing data.

A single company's share information was run through the back-propagation algorithm to determine what was preventing the ANN from learning. Figure 4.7 displays the squared error of the target output versus the actual output on a single company's stock price movements. The generated "profit" is also shown.

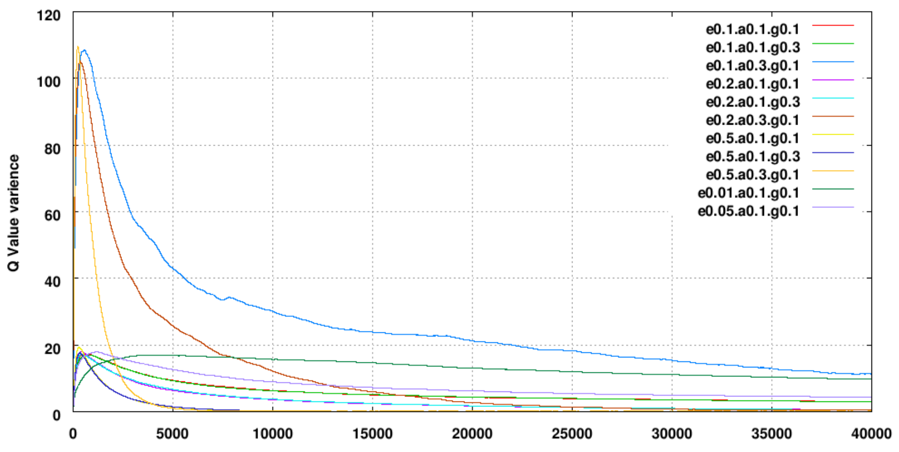

Figure 4.3: Variance on Q-values with different learning parameters

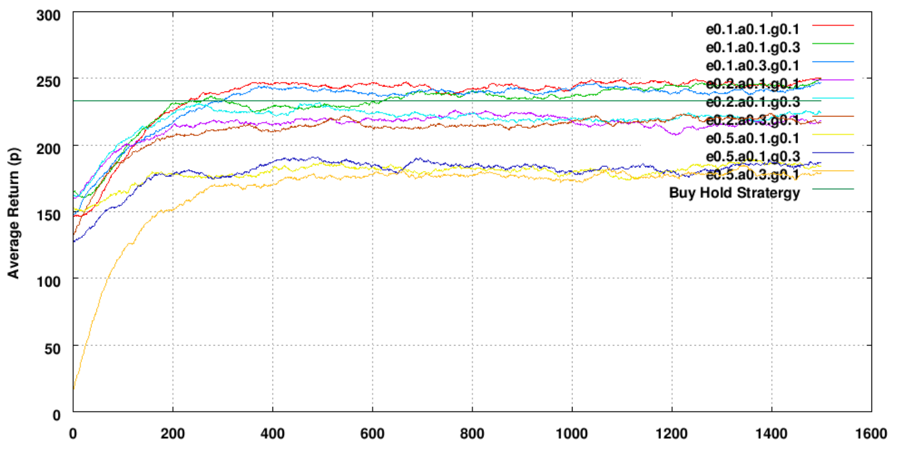

Figure 4.4: Generated profit with different learning parameters

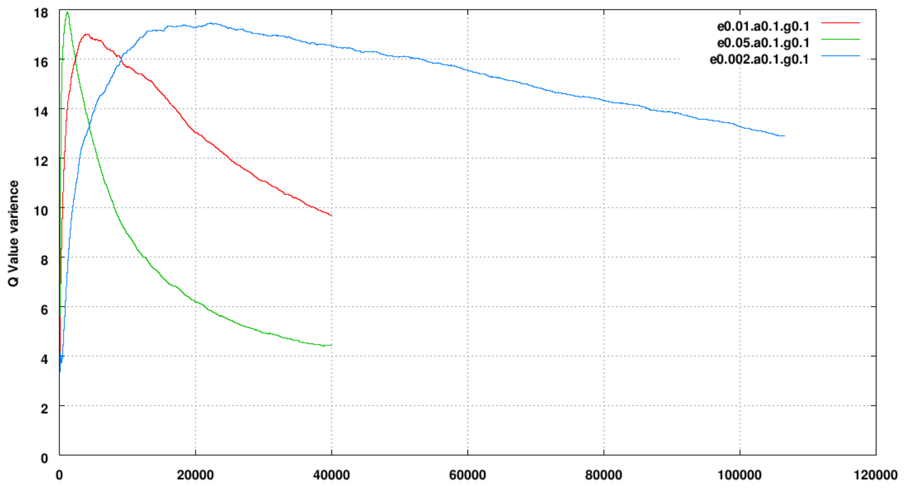

Figure 4.5: Varience on Q-values with low ϵ values

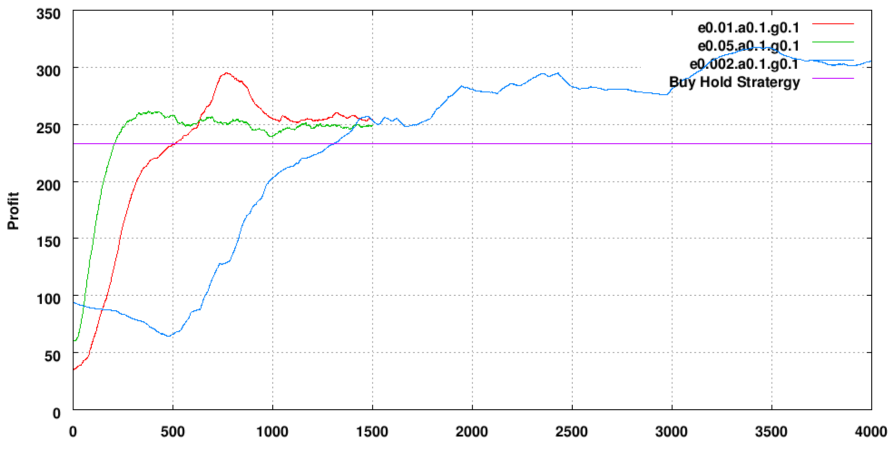

Figure 4.6: Generated profit with different with low ϵ values

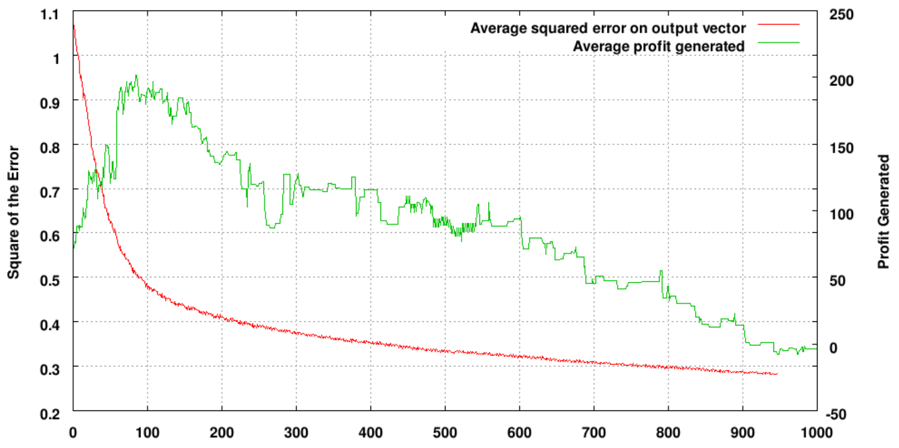

Figure 4.7: Symbol only ANN. The squared error on the output data on the training set and the generated profit

4.3.2 Evolved

Several runs with varying parameters and ANN designs were done see section 7.4 for full results. The ANN layout of 26-10-2 and a mutation frequency of 0.05% performed best.

Group	Buy Hold	Evolve
Mean	233.3183	497.84448
SD	569.9726	519.34654
SEM	105.8413	96.44023
N	29	29
P		0.0001

Table 4.5: Results of paired t-test using simple evolved strategy with the training data.

Table 4.5 provides a summary of the paired t-test results. The P value indicates with a 95% confidence level there is a statically significance difference between the buy and hold strategy and the evolved ANNs on the training data. The ANN had been successfully trained to out perform the basic buy and hold strategy.

4.4 ANN's Ability to Trade on Unseen Data

This section gives the results of the trained ANNs on the previously unseen testing data.

4.4.1 Evolved ANNs

Table 4.6 provides a summary of the paired t-test results. The P value indicates with a 95% confidence level there is not a statically significance difference be-

Group	Buy Hold	Evolve
Mean	210.2672414	284.4748279
SD	482.2206211	316.8765317
SEM	89.5461213	58.8424947
N	29	29
P		0.1578

Table 4.6: Result of paired t-test using simple evolved strategy with the testing data.

tween the buy and hold strategy and the evolved trader strategy on the testing data.

Company	Buy Hold	20 Steps	1 State	Best Optimal
AL.L	575	1839	5446.85	363.5
AVZ.L	-710	2341.25	6595	-638.75
AV.L	-277.5	1361.5	5194.05	47.5
BA.L	-2	788.75	2408	302.5
BLT.L	614.5	1879	5133.75	792.25
BB.L	178.75	771.75	2041.25	99.25
BAY.L	162	959.5	2766	172.75
CW.L	-812.02	855.59	2465.5	-582.5
CPG.L	-220.75	639.75	2063.25	-163
GLH.L	770.5	1301.25	3739.5	469.25
HMSO.L	1140.25	1743.39	4875.56	862.29
IPR.L	-100	590.25	1634	34.75
JMAT.L	388	2306	7297.5	518.5
LLOY.L	-90	1113.75	3740	-139
NRK.L	780	1796.25	4850.58	498
PRU.L	-258	1460.5	4643.25	-86.5
REL.L	-30.5	1140.75	3694.25	-75.75
REX.L	292.5	973.25	2807.9	274.75
RSA.L	-345.75	621.75	1685.25	-171
SGE.L	-268.5	695.75	2178.5	72.75
SSE.L	966	1418.5	4456.7	892.3
SVT.L	362.5	2323.5	7885.75	1198.75
SLOU.L	420.5	1081.02	2836.93	469.84
STAN.L	506	2169.5	6130	743.5
TSCO.L	181	502.75	1623.75	206.25
VOD.L	-116.25	340.5	1136.08	-28.1
WPP.L	-172	1569.06	5381.5	34
IMT.L	1315	2648	7029	809.5
RB.L	1517	2895.52	8820.5	1117
Average	233.318	1383.7	4157.25	279.123

Table 4.7: Potential profits per company using training data and the dumb trader.

Company	Buy Hold	Symbols Only (BP)	Evolved
AL.L	575	-40	276.85
AVZ.L	-710	184	120.25
AV.L	-277.5	-42.75	553
BA.L	-2	-69.25	215
BLT.L	614.5	21.25	930.25
BB.L	178.75	-16.75	96
BAY.L	162	50.25	331.75
CW.L	-812.02	-187.75	-126.75
CPG.L	-220.75	-7.25	100.5
GLH.L	770.5	336.5	752
HMSO.L	1140.25	336.84	1306.8
IPR.L	-100	88	162.75
JMAT.L	388	289.5	1077.5
LLOY.L	-90	22.5	180.25
NRK.L	780	71.75	685.25
PRU.L	-258	139.5	87.25
REL.L	-30.5	188.5	136.5
REX.L	292.5	110.75	246.9
RSA.L	-345.75	-32.25	-84.25
SGE.L	-268.5	-35.25	132
SSE.L	966	191.5	750
SVT.L	362.5	211	1415
SLOU.L	420.5	184.81	456.03
STAN.L	506	249.5	781
TSCO.L	181	11.75	201.75
VOD.L	-116.25	7	48.66
WPP.L	-172	-127.75	235.25
IMT.L	1315	472.5	1524
RB.L	1517	167	1846
Average	233.318	95.7034	497.844

Table 4.8: Profits per company, using a variety of learning strategies, with training data.

Company	Buy Hold	Symbols Only (BP)	Evolved
AB.L	304.5	N/A	354.95
ABF.L	331	N/A	67
BSY.L	-408	N/A	-81.25
BG.L	421	N/A	499
BP.L	-3	N/A	54.23
BATS.L	884	N/A	749.25
BLND.L	1157	N/A	710.25
CBRY.L	90.25	N/A	36.4
DGE.L	300	N/A	453
FP.L	24	N/A	44
GSK.L	-559	N/A	-106
HSBA.L	-121	N/A	213
KEL.L	538	N/A	372
LII.L	875	N/A	823.44
MRW.L	64.75	N/A	-7.39999
OML.L	0.5	N/A	50.4
PSN.L	1096	N/A	1071.5
RTR.L	-620.5	N/A	-304.5
RBS.L	402	N/A	791
SAB.L	610	N/A	350.75
SBRY.L	28	N/A	145.5
SCTN.L	46	N/A	270.25
SHP.L	-242	N/A	226.5
SMIN.L	276.5	N/A	363.5
TATE.L	482	N/A	366.25
UU.L	188.5	N/A	283.75
WOS.L	746	N/A	358.25
PSON.L	-745	N/A	96
ICI.L	-68.75	N/A	-1.25
Average	210.267	N/A	284.475

Table 4.9: Profits per company, using a variety of learning strategies, with testing data.

Chapter 5

Discussion

5.1 Technical Indicators

As stated previously there are many technical indicators, and financial rules available in the literature. The decisions on which rules to include was a balance of appropriate and frequently used rules, against practicality and resource management. It was not possible to test all available technical indicators. The conclusions of this thesis can therefore not be extended to include all technical indicators.

Including more indicators increases the search-able space for finding solutions, to reduce this, principle component analysis could be used to determine indicators which are redundant. Finally interpretation of the technical indicators and the logical symbols derived could be improved through the use of a human expert.

5.2 Training data

5.2.1 Shifting Historical Data

Section 4.1.1 presents the results of the shifted historical training data. A P value of less than 0.001 and a potential "profit" of 1383.7p when compared with the buy hold strategy means the 20 day shifted historical training data is statically significant. It provided a usable training set. In future studies a number of other factors should be considered.

This method does not utilise any information about the input parameters and is purely based on future market values. There is no guarantee there is a correlation between the input data and the training data. A benefit of this is the ANN may be able to make unforeseen associations. The drawback is there may not be a continuous and reliable, appropriate mapping of inputs vector to outputs vector.

Another assumption is that "t" has been fixed at a 20 time steps point. A set of logical parameters on day "t" may indicate a transaction signal but not necessarily at "t+20". This method assumes the value at "t+20" still contains information relevant to "t". This may be the case in a trending market but it should not be assumed.

The completely uncorrelated nature of the resultant training data makes it advisable to search for alternative sources.

5.2.2 Shifting Historical Data Based on Changes in State

Section 4.1.2 presents the results of the shifted historical data based on state change. This method does not suffer from the set time period problem as shifting historical data does. It does however suffer from a set number of states ahead. Table 4.2 shows that the 1 state ahead returns the greatest potential "profit".

A P value of less than 0.001 and a potential "profit" of 4157.25p means this strategy statically out performs the buy hold strategy and can be used as training data.

The strength of this training data is, it is time independent and partially correlated to the input vector, ie. the trading decisions are made at the same time intervals as the ANN. In this way the training data is as restricted as the ANN regarding when it can make a trading decision. It is not however fully correlated as the choice of decision could be either, buy or sell, for a given input vector at different times. In contrast a trained ANN will have to provide a consistent output signal for an input vector irrespective of time.

To verify that the data set was providing consistent trading information for the ANN to learn, figure 4.1 was produced. This figure shows the signal strength as a function of state. It illustrates that there are relatively few states which have a strong signal strength and a high number of hits. The majority of states consist of one or two hits, infact there are only a few states with over 20 hits and signal strength greater than 50%.

The training data was based on changes in state and yielded a profit over 3 times greater than that based on time steps alone and because of its partial correlation to the input data it is considered to be a far superior training guide for the ANN. The time steps based approach was there dropped as a potential training material.

5.2.3 Reinforced Learning, Q-Learning

The advantage of this method was the training data was correlated to the input vectors making it subject to all the restrictions the ANN has. In theory the Q-learning could discover the optimal policy and produce training data that would allow the ANN to achieve a similar level of performance. The ability of the ANN to generalise would then allow it, to trade successfully on the testing data. The disadvantage, is that Q-Learning is a computationally expensive process and as the number of input variables increases so do the possible states.

The softmax policies $\tau = 0.2$ and 0.5 did converge while the 0.1 did not. This was due to the exploration versus exploitation argument. The softmax used the known expected return values to calculate probabilities of choosing an action. In a large selection of cases the Q values were vastly different for each action and therefore the probability of exploring rapidly droped to zero. The egreedy method used a constant value ϵ, which ensures the algorithm will continue with the same frequency of exploration. The Q values did converge and the potential "profit" increased as expected. It was for this reason the egreedy method was chosen.

It was initially considered to have possible actions available to the agent as just buy or sell, so the data generated would be in the same form, as the time step strategy and the state change strategy. After considering the graph 4.1 which indicates there are many states which have no buy or sell action it was decided that a third action of do nothing should be permitted.

Table 4.2 shows the profit decreases for each state ahead of the current state, however the profit 2 states ahead is still significantly up. This leaves the possibility that a signal generated at state “s” has some relevance to the share price at state “s+2”. The Q learning algorithm is able to detect this as the value of the signal at “s” modified accordingly.

Section 4.1.2 presents the results of the Q-Learning training data. Table 4.4 the best Q-Learning agent achieved a profit of 279.12p with a P value of 0.3610. With a 95% confidence level this is judged not be statically significant. The question remains, why is there such little potential “profit” when compared with the two previous methods?

To guarantee convergence to the optimal policy Q learning algorithm relied on all states being Markov states. This rule was relaxed in this project as it is not certain that all relevant information is being passed. In fact, figure 4.1 indicates it has not. The Q learning algorithm could therefore be converging to a sub optimal policy. Figure 4.4 does not appear to confirm this. The learning parameters α, γ only effect the learning rate but not the end result. ϵ does however effect the end result, larger ϵ value made the agent explore more frequently where as with a small ϵ the agent exploited current knowledge. The smaller the ϵ value was, the better the end result, though the algorithm takes longer to converge.

The Q learning strategy was unable to produce a large enough difference in the potential “profit” to be statically significant. This coupled with figure 4.1 shows the information contained within the states is insufficient to adequately describe the future share movement. The top returned “profit” values calculated by the optimal trader reached a ceiling at about 290 indicating this is the maximum possible return value with this input vector.

5.3 Doubling the Input Data

To achieve a higher potential “profit” the information available to the ANN must be increased. This is done through increasing the input vector size. This can be achieved by introducing more financial rules in the hope that the new symbols might provide more relevant information. Alternatively the input symbols can be doubled by providing the symbols from the previous iteration. This option gives the agent a temporal perspective and allows it to learn when going from state “a” to “b”. This may be a better indication of a trend rather than just state “b”. Unfortunately it makes the domain space too large for the optimal strategy to be calculated using Q-Learning, see table 5.1.

Fortunately the highly parallel nature of the ANN means it is not a problem for an ANN to learn with this many inputs. The ANN could be trained with the most successful trading data available, the 1 state look ahead strategy. It was therefore decided to double up the inputs with the previous iterations input vector, however this meant the effect of adding financial logical rules to the ANN could not be tested.

Method	Input Symbols	States	Actions	Total States
Single symbols	12 + 2	16,384	3	49,152
Double symbols	24 + 2	67,108,864	3	201,326,592

Table 5.1: Number of states available for the optimal trader.

5.4 Training the Artificial Neural Networks

5.4.1 Back-propagation Symbols Only

Attempting to train the ANN using back-propagation resulted in a potential "profit" of 95.7p, well below the 210.3p of the buy and hold strategy. Numerous attempts at experiment variations where trialled, varying the learning rate, momentum, the ANN size and batch updating, no significant change where found. To discover the cause of the problem a small run was done using just one company's share information. From figure 4.7 the error in the output can be seen dropping as the ANN is trained, however as the error line levels off, the profits generated by the ANN drop rapidly. This indicates the training data is not providing the ANN with consistent information and as the ANN learns to accommodate the training data its own trading strategy fails.

5.4.2 Genetic Symbols Only

The ANN produced with the evolved strategy was able to predict the training data with an actual "profit" of 497.84p, 137% increase over the buy and hold strategy of 210p. This produced a P value of 0.0001 and is a statically significant difference.

5.5 Results on Testing Data

The evolved based solution was the only ANN eligible for use with the testing data. The ANN produced an actual "profit" of 284.5p compared with the buy and hold strategy of 210.3p. This is an improvement of 35% but with a P value of 0.16 it is not statically significant to a confidence level of 95%. It is important to note that statically significant is not the same as scientifically interesting. This means that there is a 16% probability that the observed difference is a random anomaly and is attributable to the subset of the population. There is however still a large probability there is a genuine difference in the means.

From table 4.9 we can see that with the trained ANN there is less variation in the profits generated, even on shares where the "buy hold" performs poorly, the network is improved. In fact there are only two companies where the ANN has a loss of over 100p, compared to six with the buy and hold strategy.

5.6 Overview

This thesis demonstrated that technical indicators may well provide an indication of future market direction. However it was unable to confirm this statically.

Results could be further improved by dedicating more resources to the evolutionary approach. The elementary algorithm could include cross over operators amongst parents and multiple populations with limited cross over. The algorithm currently has a fixed network structure, this could also be evolved in tandem with the network weights.

The generation of training data from early in the methodology proved to be a problem. The solutions employed led to unsatisfactory results. The most promising method, shifted state changes, would benefit from a more detailed mathematical study of the states and signal strengths. Allowing a third action of "do nothing" would increase its compatibility with the ANN, however this would require a search method capable of determining the appropriate action. This is however becoming very similar to the Q-Learning method.

The Q-Learning method while not producing statically significant results did manage to return a higher potential "profit" than the buy hold strategy and is probably close to the true profit capable of being generated with the given input vector. It would therefore be interesting to remove the least relevant inputs, determined through PCA and use previous iterations state inputs and attempt Q-Learning again. This should provide an improvement on the potential "profit". Lendasse et al. (2000) used PCA on the technical indicators of the Bel 20 index and reduced 42 input values to only 9, this will then allow the doubling up for the input vector and still allow Q-Learning.

Convergence does not appear to be the problem but it might be improved by using eligibility traces, where a parameter λ allows the reward value to have a direct effect on many states at once. To help speed convergence a variable ϵ value could be used which reduces as learning progresses.

An improvement to the reinforced learning method might be to employ the method used by Kjall-Ohlsson (2005). The modified Q-Learning algorithm is able to generalise by reducing the state to it constituent parts. The input vector is already the constituent parts of the state and therefore it should be easy to implement this improvement and any increase in speed to convergence would be a benefit.

The ANN may also be improved by using Elman and Jordan networks which use feedback from the network's middle or output layers from the previous iteration. This has a similar effect to adding input symbols from the previous iterations but the effect may be more general. Another avenue which has not been explored in this thesis is self training networks such as Self Organising Feature Maps (SOFM) and radial basis function networks should be considered in future research.

While it has not been possible to include the financial rules into the networks, the trained evolved network can still be interrogated to reveal its symbols associations. Using an approach such as Garcez et al. (2001) it would be possible to extract the logic program from the ANN. From section 4.1.3 it is known the states only provide enough information to get a profit of 279p on the training data. It is the doubling up of the inputs which allow the ANN to achieve 498p in the training data. This indicates that passing from state "a to b" provides considerably more information about future price movements than simple state "a" or "b" on their own.

Chapter 6

Conclusion

Of the three data sets this thesis produced, the 20 days shift strategy was dismissed as inappropriate due to the uncorrelated method used to create the data. The state based approach produced interesting results indicating it is possible to determine appropriate times to trade, indicated by the large potential "profit". However it was not able to ascertain what trade should take place, this was subsequently confirmed by the results of training the ANN. The Q-Learning algorithm was not able to provide statical difference between the learnt trading strategy and the basic buy hold strategy. However the results do indicate this method could be further explored and that a carefully re-selecting of input symbols and fine tuning of the learning parameters should provide a significantly improved "profit". Unfortunately this meant it was not possible to further investigate the relationship between the input symbols and predefined technical rules.

The second strategy of using an evolved ANN with double the input vectors did allow it to map the input vector to the output vector in a "profitable" manor. The calculated P value was 0.16 at a confidence level of 95% which is not statical significant. This however is an extremely encouraging result when the nature of the domain is taken into account and compared with the available literature.

In this thesis we test for statical significant difference to prove that the results are better than the base line increase. If however the best possible "profit" that could be generate from the given input vectors falls closes to the base line increase and the ANN successfully exploits this knowledge, this does not mean the ANN is not successfully trading above the base line, only that we cannot statically prove it is.

In retrospect the scale of the thesis was too large it required too many subjects to be studied and perhaps did not allow a detailed enough analysis of a specific areas. However it is felt the thesis was reduced to its minimum possible size. The financial rule sets and symbols where reduced to the minimum size and most popular rules used. A simple implementation of ANN was created and efforts to create adequate training data were started. The integration of the optimal trader was only considered when all other methods where exhausted. Finally an evolutionary approach was adopted when all attempts to train the network using supervised methods had failed, in attempt to remove the need for training data.

The ANN which uses the logical symbols as defined by the technical indicators is by definition containing a finite amount of prior knowledge. At this stage I do not see the inclusion of financial rules into the ANN as essential, but feel that time would be better spent examining the various logical symbols to include and improve the Q-Learning and unsupervised methods of ANN training.

Bibliography

Albanus, G. T. & Batchelor, R. A. (2002), 'Predicting high performance stocks using dimensionality reductions technique', *Unknown* .

Appel, G. (1985), *The Moving Average Convergence-Divergence Trading Method*, Traders Pr.

Bollinger, J. (2001), *Bollinger on Bollinger Bands*, McGraw-Hill Publishing Co.

Chan, M., Wong, C. & Lam, C. (1999), Financial time series forecasting by neural network using conjugate gradient learning algorithm and multiple linear regression weight initialization, Technical report, Hong Kong Polytechnic University.

Chan, W. (2004), Forecasting stock prices using neural networks, Master's thesis, Carnegie Mellon.

Demartines, P. & Herault, J. (1996), 'Curvilinear component analysis: a self-organizing neural network for nonlinear mapping of data sets', *Transactions on Neural Network* .

Eiben, A. & Smith, J. (1998), *Introduction to Evolutionary Computing*, Springer.

Elder, A. (1993), *Trading for a Living: Psychology, Trading Tactics, Money Management*, Wiley.

Fama, E. (1970), 'Efficient capital markets: A review of theory and empirical work', *Journal of Financial Econmics* pp. 283 – 306.

Garcez, A. A., Broda, K. & Gabbay, D. (2001), 'Symbolic knowledge extraction from trained neural networks: A sound approach', *Artificial Intelligence* **125**, 155–207.

Garcez, A. A. & Lamb, L. (2006), 'A connectionist computational model for epistemic and temporal reasoning', *Neural Computation* **18**(7), 1711–1738.

Garcez, A. A. & Zaverucha, G. (1999), 'The connectionist inductive learning and logic programming system', *Kluwer Academic Publishers* pp. 59 – 77.

Granville, J. (1976), *Granville's New Strategy of Daily Stock Market Timing for Maximum Profit*, Simon and Schuster.

Hawawini, G. & Keim, D. (1995), 'On the predictability of common stock returns: World-wide evidence', *Handbooks in Operations Research and Mangement Science* **9**, 497–544.

Hellstrom, T. & Hellstrom, K. (1998), Predicting the stock market, Technical Report IMa-TOM-1997-07, Malardalen University.

Holldobler, S. & Kalinke, Y. (1994), 'Towards a new massively parallel computational model for logic programming', *unknown* .

Hornik, K., Stinchcombe, M. & White, H. (1989), 'Multilayer feed forward networks are universial approximators', *Neural Networks* **1**(2), 359 – 366.

Hui, S. C., Yap, M. T. & Prakash, P. (2000), A hybrid time lagged network for predicting stock prices, Master's thesis, Nanyang Technological University.

Keogh, E. & Lin, J. (2003), 'Clustering of time series subsequences in meaningless: Implications for previous and future research', *ICDM* .

Kjall-Ohlsson, N. (2005), Associatove reinforcement learning, Master's thesis, City Univeristy.

Lendasse, A., Bodt, E. D., Wertz, V. & Verleysen, M. (2000), 'Non-linear financial time series forecasting - application to the bel 20', *ESANN'2000 proceedings* .

Lendasse, A., Lee, J., Bodt, E. D., Wertz, V. & Verleysen, M. (2001), 'Input data reduction for the prediction of financial time series', *ESANN'2001 proceedings* pp. 2–930307–01–3, 237–244.

Malkial, B. (2003), The efficient market hypothesis and its critics, Master's thesis, Princeton University.

McCulloch, W. & Pitts, W. (1943), 'A logical calculus of the ideas immanent in nervous activity', *Mathematical Biophysics* **7**, 115 – 113.

Minsky, M. & Papert, S. (1972), *Perceptrons*, MIT Press.

Nygren, K. (2004), Stock prediction - a neural network approach, Master's thesis, Royal Institute of Technology, KTH.

Russel, S. & Norvig, P. (2003), *Artificial Intelligence A Modern Approach (Second Edition)*, Prentice Hall.

Singh, S. & Fieldsend, J. (2001), Pattern matching and neural networks based hybrid forecasting system, Master's thesis, University of Exeter.

Sutton, R. & Barto, A. (2000), *Reinforcement Learning An Introduction*, The MIT Press.

Tesauro, G. (1990), 'Neurogammon: A neural network backgammon program', *IJCNN Proceedings* **III**, 33 – 39.

Tesauro, G. J. (1994), 'Td-gammon, a self teaching backgammon program, achieves master level play', *Neural Computation* pp. 215 – 219.

Towell, G. G. & Shavlik, J. W. (1994), 'Knowledge-based artificial neural networks', *Artificial Intelligence* pp. 119 – 165.

Watkins, C. (1989), Learning from Delayed Rewards, PhD thesis, Cambridge University.

Wilder, W. (1978), *New Concepts in Technical Trading Systems*, Trend Research.

Yahoo! (2006), ‘http://finance.uk.yahoo.com’, Internet. Website.

Chapter 7

Appendix

7.1 Appendix A: Project Proposal

7.2 Appendix B: Data Sets

Table about the data sets, when they start, stop.

Training AL.L, AVZ.L, AV.L, BA.L, BLT.L, BB.L, BAY.L, CW.L, CPG.L, GLH.L, HMSO.L, IPR.L, JMAT.L, LLOY.L, NRK.L, PRU.L, REL.L, REX.L, RSA.L, SGE.L, SSE.L, SVT.L, SLOU.L, STAN.L, TSCO.L, VOD.L, WPP.L, IMT.L, RB.L

Testing AB.L, ABF.L, BSY.L, BG.L, BP.L, BATS.L, BLND.L, CBRY.L, DGE.L, FP.L, GSK.L, HSBA.L, KEL.L, LII.L, MRW.L, OML.L, PSN.L, RTR.L, RBS.L, SAB.L, SBRY.L, SCTN.L, SHP.L, SMIN.L, TATE.L, UU.L, WOS.L, PSON.L, ICI.L

BT-A.L	BT GROUP
III.L	3I GROUP
AL.L	ALL AND LEICS
AB.L	ALLIANCE BOOTS
AVZ.L	AMVESCAP
AAL.L	ANGLO AMERICAN
ANTO.L	ANTOFAGASTA
ABF.L	ASSOCIAT BRIT FOODS
AZN.L	ASTRAZENECA
AV.L	AVIVA
BG.L	BG GROUP
BP.L	BP
BA.L	BAE SYSTEMS
BARC.L	BARCLAYS
BLT.L	BHP BILLITON
BB.L	BRADFORD AND BINGLEY
BAY.L	BRITISH AIRWAYS
BATS.L	BRIT AMER TOBACCO
BLND.L	BRIT LAND CO REIT
BSY.L	B SKY B GROUP
CW.L	CABLE AND WIRELESS
CBRY.L	CADBURY SCHWEPPES
CNE.L	CAIRN ENERGY
CPI.L	CAPITA GRP
CCL.L	CARNIVAL
CNA.L	CENTRICA
CPG.L	COMPASS GROUP
CS.L	CORUS GROUP
DSGI.L	DSG INTERNATIONAL
DGE.L	DIAGEO
DRX.L	DRAX GROUP

Table 7.1: Yahoo! company symbols and company names

ETI.L	ENTERPRISE INNS
EXPN.L	EXPERIAN
FP.L	FRIENDS PROVIDENT
GLH.L	GALLAHER GRP
GSK.L	GLAXOSMITHKLINE
HMSO.L	HAMMERSON REIT
HNS.L	HANSON
HBOS.L	HBOS
HOME.L	HOME RETAIL GROUP
HSBA.L	HSBC HLDG
IAP.L	ICAP
ICI.L	ICI
IMT.L	IMPERIAL TOBACCO
IHG.L	INTERCONT HOTEL GRP
IPR.L	INTERNATIONAL POWER
ITV.L	ITV
JMAT.L	JOHNSON MATTHEY PLC
KAZ.L	KAZAKHMYS
KEL.L	KELDA
KGF.L	KINGFISHER
LAND.L	LAND SEC R.E.I.T.
LGEN.L	LEGAL AND GENERAL
LII.L	LIBERTY INT R.E.I.T
LLOY.L	LLOYDS TSB
LMI.L	LONMIN
EMG.L	MAN GROUP
MKS.L	MARKS AND SPENCER
MRW.L	MORRISON SUPERMKTS
NG.L	NATIONAL GRID
NXT.L	NEXT
NRK.L	NORTHERN ROCK
OML.L	OLD MUTUAL
PSON.L	PEARSON
PSN.L	PERSIMMON PLC

Table 7.2: Yahoo! company symbols and company names

PRU.L	PRUDENTIAL
RDSB.L	ROYAL DUTCH SHELL-B
RB.L	RECKITT BENCKISER
REL.L	REED ELSEVIER
RSL.L	RESOLUTION
RTR.L	REUTERS GROUP
REX.L	REXAM
RIO.L	RIO TINTO
RR.L	ROLLS-ROYCE GROUP
RBS.L	ROYAL BK SCOTL GR
RDSA.L	ROYAL DUTCH SHELL-A
RSA.L	ROYAL SUN ALLIANCE
SAB.L	SABMILLER
SGE.L	SAGE GRP
SBRY.L	SAINSBURY
SCTN.L	SCOTT AND NEWCASTLE
SSE.L	SCOT AND STHN ENERGY
SPW.L	SCOTTISH POWER
SVT.L	SEVERN TRENT
SHP.L	SHIRE
SLOU.L	SLOUGH ESTATES REIT
SN.L	SMITH AND NEPHEW
SMIN.L	SMITHS GROUP
STAN.L	STANDARD CHARTERED
SL.L	STANDARD LIFE
TATE.L	TATE AND LYLE
TSCO.L	TESCO PLC
ULVR.L	UNILEVER
UU.L	UNITED UTILITIES
VED.L	VEDANTA RESOURCES
VOD.L	VODAFONE GRP
WTB.L	WHITBREAD
WOS.L	WOLSELEY PLC
WPP.L	WPP GRP
XTA.L	XSTRATA

Table 7.3: Yahoo! company symbols and company names

7.3 Appendix C: Knowledge Base

Symbol	Definition	Justification
A	medium MA > slow MA	3 moving average
B	fast MA > medium MA	3 moving average
C	|MACD |> % * Close	MACD, trending/ranging
D	MACD > signal line	MACD, signal
E	VO > 0	Volume Oscillator
F	upper BB < close	Bollinger Bands
G	lower BB > close	Bollinger Bands
H	TRIX > 0	TRIX
J	TRIX > signal line	TRIX, signal line
K	+DI > -DI	Directional Movement
L	ADX > +DI	Directional Movement
M	ADX > -DI	Directional Movement
N	ema8 > ema20	check for raising OBV

Table 7.4: Knowledge base

7.4 Appendix D: Evolved ANN Results

Layout	Mutation Frequency	Double Inputs	Result
26-5-2	0.05	Y	325.742
26-5-2	0.1	Y	366.98
26-10-2	0.05	Y	497.844
26-10-2	0.1	Y	460.761
26-15-2	0.05	Y	470.113
26-15-2	0.1	Y	412.012
26-15-2	0.2	Y	432.666
26-20-2	0.05	Y	386.646
26-20-2	0.1	Y	377.634
26-20-2	0.2	Y	378.444

Table 7.5:

7.5 Appendix E: Setting up System

See README file in the source code directory for information on installing and compiling the code. A brief description of the programs is given.

Warning: All programs work well, however an amateur will find using the programs tough as no effort has been made into making them user friendly.

Simplenet, Runs a previously saved ANN through the data set. Data set is decided at compile time. The results of the run are given on the command line.

Backprop2, Attempts to train an ANN with training data, see script list.sh in same directory for possible options.

Calmaxprofit, Calculates the maximum profit on the data set with the given strategy.

Calbuyholdprofit, Calculates the maximum profit on the buy and hold strategy with the compiled data set.

GenData, Generates all the technical data from the raw stock prices

GenFutStateData, Generates the future state trading strategy for each stock.

Genetictest, Runs the evolved ANN run, see script list.sh in same directory for details of input parameters.

Optimaltrades, Generates an optimal strategy, see script list.sh in same directory for details of input parameters.