MASSEY UNIVERSITY
COLLEGE OF SCIENCES

Declaration Confirming Content of Digital Version of Thesis

I confirm that the content of the digital version of this thesis

Title: **Equity Trend Prediction with Neural Networks: An Empirical Analysis**

is the final amended version following the examination process and is identical to this hard bound paper copy.

Have you included published articles/material from your thesis? (Yes) / No

If yes, have you received copyright permission from the copyright holder (usually the publisher) to include this material with your thesis? (Yes) / No

Student's Name: **Russell James Halliday**

Student's Signature: RJHalliday

Date: **25/02/2012**

Equity Trend Prediction with Neural Networks: An Empirical Analysis

A thesis presented in partial fulfilment of the requirements for the degree of

Master of Engineering
in
Computer Systems Engineering

at Massey University, Albany,
New Zealand.

Russell James Halliday

2012

Acknowledgements

I wish to give my utmost gratitude to my supervisor Dr Mohammad Rashid and Dr Chris Messom. Dr Chris Messom gave me the guidance and input in steering the initial direction of the thesis, while Dr Rashid was instrumental in guiding it's final form.

I also wish to thank my wife Angie for her encouragement, support, patience and understanding at all times. Finally I thank my parents, brother and sisters for their unwavering support.

Abstract

This thesis presents results of neural network based trend prediction for equity markets. Despite a breadth of research which has focused on the prediction of various equity and currency exchange markets, much has focused on the use of specific techniques in such predictions. Few bodies of work have compared a wide range of equity market data preprocessing and technical analysis techniques in creating a prediction model based on feed-forward and recursive neural networks.

To achieve a broad-based prediction model, the work in this study was broken into three distinct parts. Firstly, the neural networks goal is defined as finding whether a stock will be higher or lower than the previous trading period. Subsequent to this, a variety of input data scaling and network topologies are looked at. This includes the use of Self Organising Maps (SOM) as a data classification method to limit neural network inputs and training data requirements. Feed Forward and Elman networks of various topologies are used to narrow down the best network combinations. The resulting simulation is a neural network that can predict whether the next trading period will be, on average, higher or lower than the current.

Secondly, the topology and preprocessing lessons learned during the first phase are applied to two types of neural network. Technical analysis is applied to the input data in an attempt to verify the usefulness of conventional stock indicators as inputs to neural networks. The two types of networks trained are, for the purposes of this thesis, dubbed indicator-predictive and price-predictive networks, meaning that technical analysis inputs are used to predict the next trading days technical indicator or future stock price direction respectively.

Finally, a combined network is trained which takes the inputs from the price-predictive networks in an attempt to gain better results. The hypothesis with this network is that the combined neural network should learn which of its inputs are more indicative of a stock price movement, and thus more accurately predict the future direction of the stock.

Table of Contents

List of Abbreviations

Various specialised abbreviations are used in this thesis as listed below:

ADL..... Accumulation Distribution Line
ANN.....Artificial Neural Network
BSE.....Bombay Stock Exchange
BSH.....Buy Sell Hold
MLP.....Multi Layer Perceptron
MACD.....Moving Average Convergence Divergence
NYSE.....New York Stock Exchange
OBV.....On Balance Volume
OHLCV.....Open High Low Close Volume
RNN.....Recursive Neural Network
RMSE.....Root Mean Square Error
SOM.....Self Organising Map

List of Matlab Functions

List of Tables

List of Figures

CHAPTER 1 - INTRODUCTION

1.1 Introduction

Prediction of financial markets has long been a holy grail in the minds of equity investors. With the advent of powerful computers, much attention has been focused on this field. The ability of neural networks to learn from training data has not been overlooked, and as such neural networks have been applied to a range of trading market applications from equity markets to currency markets. Contrasting this is a level of scepticism surrounding the ability of a system to predict future prices of trading markets.

1.2 Background

Equity market prices depend on many influences. Key factors that influence future equity prices can be broadly divided into quantitative and qualitative types. Primary quantitative factors include open, high, low, close and volume data for individual equities, market segments (equity groups), indexes and exchange markets as a whole. Qualitative factors include socio-economic, political, international, regional and performance factors to name but a few (Senanayake and Janaththanan, n.d.).

1.3 Formulation of the research problem

Due to the difficulty in accurately retrieving and quantifying historical qualitative factors, network inputs used in the model presented here have been confined to readily available quantitative data. However, from quantitative factors the key qualitative factor of the market sentiment can be derived. Market sentiment tells us if the market is bullish, where high levels of confidence and rising prices prevail, or bearish, where there is a lack of investor confidence and prices are in decline. Thus historical data quantitatively reflects qualitative market sentiment to some extent which in turn should give indication of future price movements.

Traditional trading systems are almost exclusively mechanical systems that apply mathematical formulae to securities data to produce turnery buy, sell and hold indicators. The weakness of this traditional approach is that the trading system must be

programmed to make explicit use of certain trading rules.

This thesis initially hypothesises that neural networks should be able learn from training data and in turn make use of data that is intrinsically present in the input data set without additional technical analysis.

1.4 Scope and limitation

The primary objective of this thesis was the creation of a neural network that, given a set of historical daily data, preprocessed with simple inter-day differencing or with a range of advanced technical analysis techniques, is capable of predicting the direction of the future price trend. The price trend prediction simply being whether market prices would on average increase or decrease relative to a subset of the daily training data.

The secondary objective of this thesis was the creation of an additional neural network which, cascaded with the output of the previously created networks, is able to increase the accuracy of the result in order to make a better predictor of stock price direction.

1.5 Methodology

Simulation data was sourced from Yahoo Finance (Yahoo, 2007). The data used for network training and verification is comprised of daily figures for individual equities listed on the New York Stock Exchange (NYSE) from January 1st 1985 to December 31st 2000.

The simulations carried out in this thesis take a series of stock data and, through network-parameter changes, are able to determine the best arrangement for each neural network with the net effect of increasing the accuracy of the overall output.

The preprocessing, neural networks and analysis used to produce results presented in this thesis were implemented entirely under Mathworks high-level language, Matlab.

1.6 Thesis Organisation

This thesis is organised into the following chapters:

- Chapter 2 provides a background to neural networks and financial market including a literature review that is focused on these areas.

- Chapter 3 looks at network design options and analyses the effectiveness of various topologies and methodologies to equity market predictions.

- Chapter 4 discusses the final topology of the neural network by drawing on the analysis undertaken in Chapter 4.

- Chapter 5 takes the basic topology decided upon in the previous chapter and analyses predictive effectiveness for the various technical indicator neural network combinations.

- Chapter 6 utilises a neural network to combine the networks from the previous chapter with the aim of improving the overall predictive power to be greater than a single neural network indicator combination.

- Chapter 7 gives the overall conclusion to the work and suggests future research direction.

CHAPTER 2 - ARTIFICIAL NEURAL NETWORKS AS A PREDICTION TOOL

Artificial Neural Networks (ANN) are inspired after the biological nervous system of the brain. Unlike more artificial devices such as computers which represent singular processing, artificial neural networks use a highly interconnected design to achieve a result.

Like a biological brain, an ANN learns through a learning process whereby the network is given a series of examples with appropriate responses to those examples collectively known as training data.

Through this training, the interconnections between neural network elements are either strengthened or weakened in response to training data. Furthermore Multi Layer Perceptrons (MLP) make use of several nodes placed in between the output and inputs, know as hidden nodes; the purpose of hidden nodes is to essentially further extract useful data from the input nodes before feeding it through to the network outputs (Swingler, 1996). A diagram depicting a simple MLP network is given in Figure 2.1.

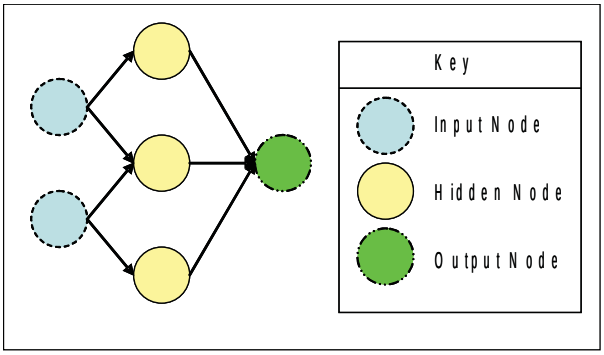

Figure 2.1: Simple Multilayer Perception

There are several neural network structures, the ones considered in this thesis were the two most fundamental types, Elman and Feed Forward (Mendelsohn, September 1993). Feed forward networks are simple networks without any form of internal feedback, such

as that shown in Figure 2.1.

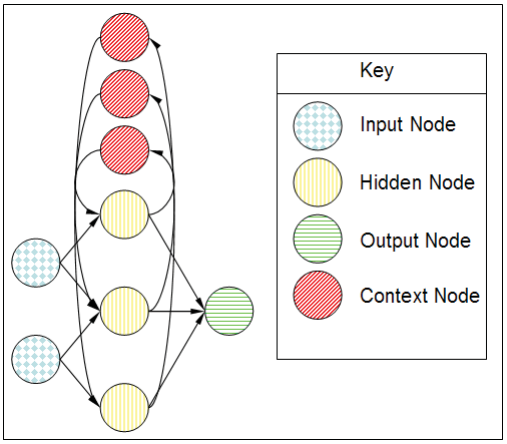

Figure 2.2: Elman Network

An Elman network uses internal feedback between network layers, allowing the detection and generation of time-varying patterns in network input. Simply put, Elman networks are appropriate for time varying data by making use of a recurrent context layer which acts as a network memory (Figure 2.2). Further information on Elman networks can be found in the prominent work by Elamn (1990).

For implementation, full use was made of the existing Matlab neural network toolbox, including for the creation of Elman and feed-forward networks (Matlab *elman* and *newffd* functions respectively).

Network training is undertaken by a standard algorithm known as back propagation learning (Hagan et. al, 1996). The steps in back propagation are as shown in Figure 2.3. As training functions exist in Matlab, all network training was implemented using the Matlab *train* function.

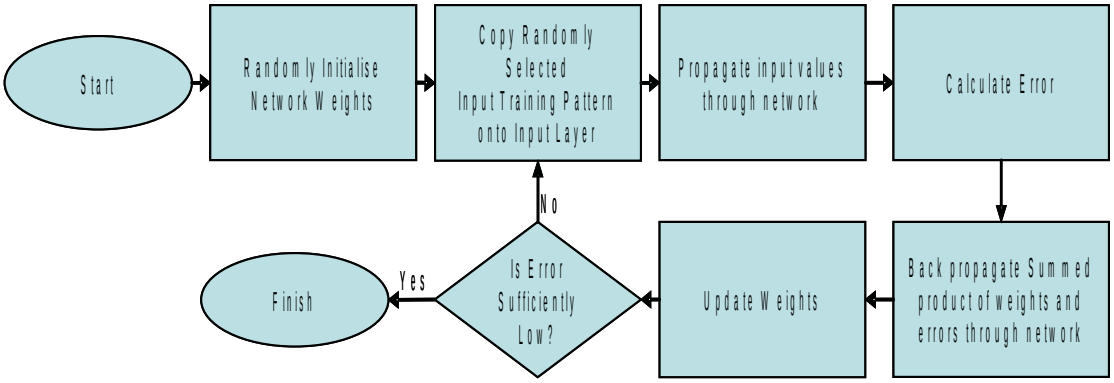

Figure 2.3: Simplified Diagram Illustrating Back Prorogation Learning (Tveter, 2001)

Self Organising Maps are essentially a form of compression or vector quantisation. A simple two-dimensional SOM is shown in Figure 2.4.

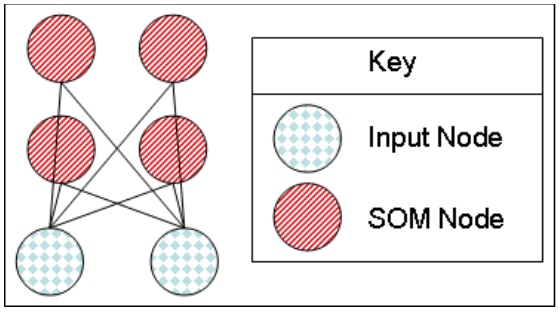

Figure 2.4: Self Organising Map Topology

Unlike Elman and Feed Forward networks, a SOM network does not need a target output. A SOM initially starts with random weights and, after training, stabilises into a 'map' of stable zones. Features are typically classified by the zone that is effected by their stimulus, so that even previously unseen inputs will stimulate similar zones to that of other similar inputs. The simplified training algorithm for a SOM is shown in Figure 2.5 and can be read in detail in Kohonen (1994).

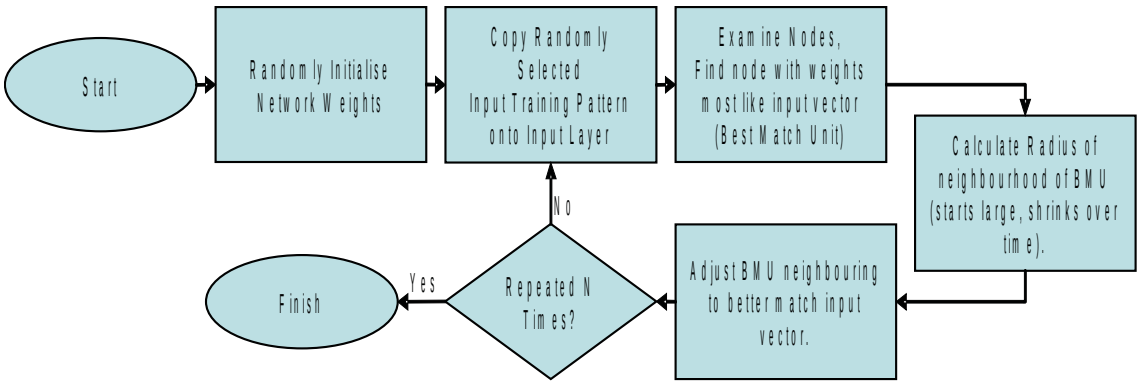

Figure 2.5: Self Organising Map Training (KOHONEN94)

2.1 Stock Market

Malkiel (1996) suggests that the stock market by it's very nature is random. Malkiel's random-walk theory an important background to this field, stating that equity markets are random, that the size and direction of changes in the share market or an individual stock can not be predicted from the size and direction of previous price movements.

While random walk has not been conclusively proven or disproved, there is strong empirical evidence against it, most notably the seminal study by Brown et.al (1998), which demonstrates positive returns over the period from 1902 to 1929, directly contradictory to the random walk theory.

Furthermore, it is widely accepted that there are inefficiencies in the market meaning “not all investors are equally well informed, nor are they all informed at the same time” (Dalton, 1996). These inefficiencies also go against the random walk theory and form an important basis for research into the prediction of future prices of equities.

2.2 Equity Markets

Equities or stocks are traded daily on various stock exchanges around the world. Stock markets like any other market have buyers and sellers bidding and offering different prices, when a bid price meets an offering price then the exchange of equity will occur. Once an exchange has occurred the now highest bidder may or may not be at the asking

price of the lowest sellers offer price. This process repeats many times per day per stock with buyers and sellers attempting to get the best price for their stock at that moment.

A third party stock broker facilitates the transaction, with the buyer or seller giving them instructions as to how much and at what respective maximum or minimum price they are willing to accept. It is then left to the broker to bid to buy and sell shares on their clients behalf.

2.3 Investment versus Trading

The study and investment or trading in the stock market can be broken into two very different schools of thought: technical analysis, which uses mathematical formula and rules to create market predictions, and fundamental analysis, which relies upon research of a company's financial and other operations in an attempt to objectively select the best stocks to invest in . The outlook of these two methodologies is different with technical analysis typically focusing on the short term and fundamental analysis giving a medium to long term perspective. This thesis focuses entirely on technical analysis principles wherein the mathematical nature lends itself to the use neural networks.

Daily data from the stock market is recorded as the market opening price, daily high, daily low, closing price and volume. For the purposes of this thesis we will call this data OHLCV representing the Open, High, Low, Close and Volume respectively.

2.4 Technical Analysis

Technical analysis is ‘the study of price activity – more specifically price patterns – to identify trade opportunities’ (Schwager, 1999). Essentially, technical analysis uses past pricing data to create indicators, which attempt to predict the future trading pattern of an equity.

Network indicators can be broken into two types, homogeneous, which use data derived from the equity in question, and heterogeneous which relies on other data such as industry and market indexes. Most indicators are typically specific to a certain equity and so are homogeneous by nature, however the output of these homogeneous indicators can be used heterogeneously as a means of comparison between stocks.

Where indicators are applied to indexes they can be considered heterogeneous; very few indicators are heterogeneous between two different stocks, but rather between a stock and it’s market or industry index.

2.5 Data Collection and Adjustment

Generally, technical indicators will give an indication of stock momentum, future price direction or buy/sell/hold recommendations. While specific details of technical indicators and their mathematical implementation are given in the Appendix, a simple technical indicator is given here for the sake of clarity.

One of the simplest and most common technical indicators is based on moving averages. Under this, a moving average of a stock's closing price is taken: this moving average signifies a baseline of the stock. A technical signal would occur when the stock price broke through the moving average, either dropping below it or rising above it. When this happens, it is a sign that the stock has broken above (or below) the current market trend, and as such there has been a change in the market sentiment for the stock. For example, when a stock which is trending upwards begins to level off or even decline, it is a clear indication that the trend has been broken. Typically speaking, a single technical indicator does not sit in isolation, as in this example, a break in trend may signal only a temporary levelling off of an overall increasing trend or the beginning of a down movement for the stock in question. Gerald Appel created the Moving Average Convergence Divergence (MACD) in the late 1970, an important technical indicator that employs the use of moving averages (Gerald, 1999).

Common occurrences in equity markets are so-called stock splits and reverse splits. Price data has been pre-adjusted to reflect these price abnormalities. Without such correction, trading data would experience unexpected price jumps up and down for reverse splits and splits respectively (SEC, July 2010). For example, if a stock selling at $2 were split on a 2:1 basis then a downward price jump of $1 would be shown after the split, while the opposite scenario with a jump from $2 to $4 would be true for a reverse split. Without adjustment, this would create abnormalities in the training data, thereby adversely effecting the training and performance of the neural network.

Schwager (1999) highlights the importance of training data length in correctly assessing the systems ability to achieve accurate predictions in reasonable range of market conditions. 10 – 20 years is proposed by Schwager (1999) as a reasonable range for system assessment. Sufficient predictive power to facilitate price trend movements is further highlighted by Swingler (1996).

2.5.1 Pre-processing

Bellman (1961) first coined the term the 'curse of dimensionality' which refers to the exponential increase in volume as a function of dimensionality. Drawing upon Singler (1996), the curse of dimensionality affects neural networks with the number of training examples required to train a neural network increasing dramatically with the number of network weights This sometimes-exponential increase puts strain on both data collection and computation requirements.

A financial time series may consist of up to 80% noise (Burgess, 1995). The use of Elman networks in for noisy environments has been suggested by Giles et. al (2001) while Mendelsohn (1991) highlights that the quality and preprocessing of data is essential in limiting the effect or filtering out much of this noise.

Consider Figure 2.5.1; if a network were trained over section A of the data, then it would be unlikely to be able to generalise for the data covered in section B. Senanayake and Janaththanan (n.d.) propose that the solution to this problem is the use of simple fist order differences (Equation 1).

$$\delta_t = D_t - D_{t-1} \qquad (1)$$

where Dt is the raw input data.

By taking first order differences, the network is trained only on the magnitude of the inter-period difference, rather that the actual value of the contributing points Giltes et. al, 2001).

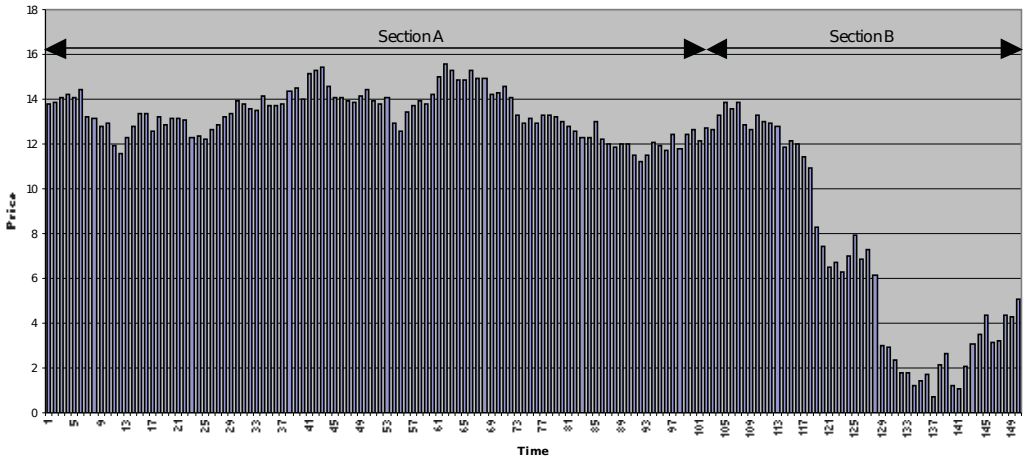

Figure 2.5.1: Difficulty in generalising from raw training data. Generalisation of section B from A is difficult given the entirely different trend and magnitude of the data.

Normalisation is a key part of data pre-processing for neural networks and should enable more accurately predict future price trends. Mendelsohn (October, 1993) discusses the use of normalisation to ensure that the distribution of input data is approximately uniform. Conversely LeBaron and Weigend (1997) argue that ANN perform reasonably only when the test period is similar to the training period.

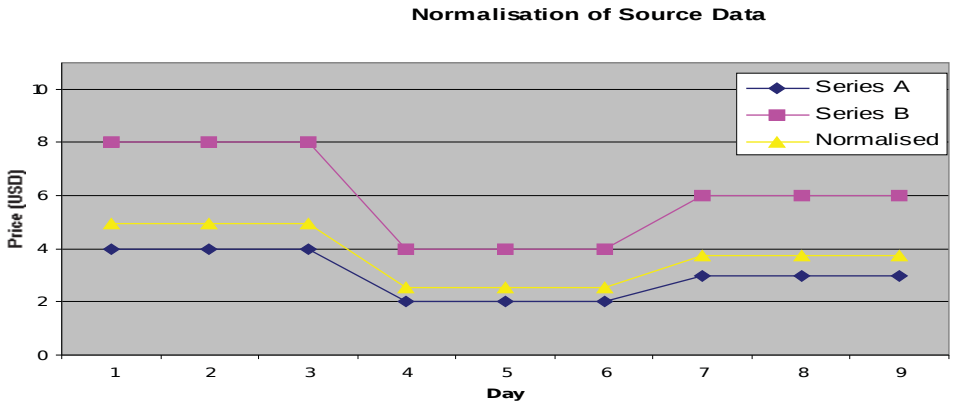

Figure 2.5.2: Normalisation of Source Data

Consider Figure 2.5.2 (above), both equities follow the same price changes, but on a different scale of magnitude. Pring (2001) highlights that the important feature of data is the relative changes in daily stock prices rather than the absolute stock price.

Normalisation yields an identical price graph for both. Before normalisation, both data sets exhibit the same qualities at different orders of magnitude. By normalising the data, the trend prediction neural network can be trained to identify generic trends in data rather than specific data arrangements.

Drawing upon Kohonen (1994), Self Organising Maps (SOM) can be used for dimensionality reduction in network implementation.

2.5.2 Training

Neural networks can suffer from over training. By over training, a network its ability to generalise is diminished. Schwager (1999) points out that this phenomenon is common to all forms of neural network training in addition to human tuning of mechanical trading systems. Furthermore, Shadbolt (2002) raises the importance of using a separate data set to verify data and stopping network training once the training set beings to increase.

2.6 Neural Network Design and Feasibility

Many considerations must be examined in developing a neural network. Attention needs to be given to these design considerations before beginning the network implementation phase. 'Time series forecasting in stock market is characterized by data intensity, noise, non-stationary, unstructured nature, high degree of uncertainty, and hidden relationships' (Chang, 2009). Design considerations include feasibility of the proposed network, data collection limitations, pre-processing and training. All of these aspects influence the effectiveness of the desired goals. Ultimately it must be remembered that neural networks are the tool or vehicle, but the objective must be both measurable and viable given available data and computational limitations.

Swingler (1996) proposes that feasibility is an important factor in the assessment of any neural network project. A primary feasibility aspect is that of information inherent in training data. Like any statistical tool, neural networks are limited by the intrinsic information in input data. It is not possible to predict information that is not reflected in training set. A simple example of this is real world events, such as company announcements, that, due to lack of pertinent information beforehand, cannot be

anticipated by the market.

Even in isolation of external factors, it is unrealistic to presume that any set of historical data points inherently contain required information suitable for precise prediction of future market prices. An important study by Giles et. al (1997) proved that for financial time series forecasting Recurrent Artificial Neural Networks outperform stateless networks. At best, a trading system, be it mechanical or artificial intelligence by design, can only aim to maximise pertinent information extraction from a historical data set. With a neural network approach, we can at best hope the system derives information otherwise obscured in the training set.

Due to the large amount of research that has gone into neural networks, the function of Multi-Layer Perceptions can be implemented relatively easily using conventional programming techniques.

Once the function of the neural network is implemented in software, the task of input pre-processing and network training can be easily achieved on any modern computer.

2.7 Literature Review

Before implementation of input data pre-processing, the neural network itself and training, a wider body of research in this field must be examined. This literature review considers the foundation provided by existing research with the aim of guiding the ultimate direction and implementation of this study.

Efficient market hypothesis proposes that financial markets and thus equity prices reflect all relevant information when determining equity prices. The idea of an efficient market was first suggested by Fama (1965) who said that new information on intrinsic values will be reflected instantaneously in action prices. An early research of neural networks was White (1988), whose research was aimed at testing the efficient market hypothesis, did not find evidence against it.

Contrary to the findings of White (1988), Bosarge (1993) among others found significant non-linearities in various time series and was able to improve the quality of forecasting. As such efficient market hypothesis is a hotly debated topic, with even Peter Lynch claiming “Efficient markets? That's a bunch of junk, crazy stuff.” (Para & McDonald, 1995). Market prediction relies on the use of non-linear models, such as neural networks, to improve prediction of the market.

A seminal study in the area of financial markets is Swanson and White (1997), who found that only marginal improvements in forecasting of macroeconomic indicators using ANN models. As such in the work presented in this thesis, focus is instead placed on stock price instead of macroeconomic indicators.

Given a wide body of research focusing on the prediction of equity and currency markets, there are few bodies of work that broadly look at the effectiveness of a range of traditional technical indicators when used as inputs to neural networks.

Atsalakis and Valavanis (2009) surveys more than 100 papers and articles which focus on neural networks and neuro-fuzzy techniques. The authors find that 60% of articles utilised feed-forward networks, while only 20% focused used technical analysis factors as inputs. Finally the authors find that 30% of articles use stock closing prices or index.

The application of feed-forward networks combined with the lack of research utilising technical indicators and stock closing prices found in Atsalakis and Valavanis (2009) was one of the primary reasons for selecting feed forward networks and a combination of technical and stock market closing price data in this thesis.

A relevant work by Lawrence (1997) looks at the application of neural networks in the forecasting of stock market prices. Specifically the author suggests that “a network’s performance is often measured on how well the system predicts market direction”, an approach adopted in this paper. The author further states that “neural networks are able to partially predict share prices”, confirming the validity of a neural network approach to stock market prediction.

McCluskey (1993) undertook a comparison of many different network architectures including both back-propagation, recurrent and hand-coded methodologies. Results shows that recurrent neural networks performed best, showing that it is appropriate to use recurrent networks in stock market prediction.

Atsalakis, G. S. and Valavanis, K.P. (2009) surveyed nearly 100 papers focused on stock market forecasting using “neural and neural-fuzzy techniques”. Specifically the study found that 60% of surveyed papers utilised feed-forward or recurrent networks. It is on this basis that feed-forward and recurrent networks were focused on in this study.

A notable work by Soni (2011), examines recent literature in the field of stock market prediction including machine learning techniques and artificial intelligence. The author draws a number of conclusions, including that ANN's have an ability to to “extract useful information from large set of data” and as a result “play very important role in stock market prediction” (Soni, 2011, p. 82). The author further concludes that “ANN's are more accurate than other competitive models and algorithm i.e. genetic algorithm , multiple linear regression analysis models for stock market prediction ” (Soni, 2011, p.82). This paper affirms the usefulness and appropriateness of investigating ANN's in the stock market.

Abdelmouez et. al (2007) looks at the use of technical analysis techniques using reinforcement learning and genetic programming in foreign exchange markets. These authors similarly note the narrow focus of academic studies focus on “testing popular trading rules in isolation”. This body of research however was limited to only eight technical analysis trading indicators, and did not consider the effectiveness of a broader range or technical indicators in assessing their effectiveness. The result of Abdelmouez e. al (2009) was that all four approaches considered were capable of achieving successful trading strategy. Specifically the authors note that “techniques investigated here return positive results both in-sample and in out-of-sample back-tests implies that there is useful information in technical indicators that can be exploited”.

Tanaka-Yamawaki and Tokuoka (2007) look at the use of intra-day rather than more common inter-day stock data in the prediction of the stock market. This paper focuses on the “optimal combination of a few indicators” for stocks using evolutional computation. As a result of their approach the authors were able to achieve a 66% prediction rate on the eight stocks selected for this paper. While this work took the less traditional approach of focusing on intra-day data, the authors did not compare the effectiveness of a wide range of indicators.

Chen et. al (2008) selected Self Organising Maps as the centre piece of their equity fund prediction model. This work was focused on the use of the SOM in mutual funds and additionally tracking the wider stock market according to macroeconomic indicators. The model used in this work resulted in a 122 percent return despite a fall in the weighted index of funds. This body of works shows the efficiency and effectiveness of SOM and as such these were selected as a primary component utilised.

Huang et. al (2007) reviews literature surrounding the use of neural networks in finance and economics forecasting. The paper examines the input variables and models in the prediction of the stock market and exchange rates. As the authors point out, technical indicators typically multivariate while neural networks are often single variate. Huang et. al (2007) concludes that there is "no doubt that the prediction performance of neural networks is improved by integrating it with other technologies". The authors highlight the need for multivariate inputs whereby data outside of the time series itself is used in prediction. Finally Huang et. al (2007) examines a range of neural network models, including MLP and RNN's, with MLP being sited as the most common due to it's ability to create arbitrary input-output mappings. The paper goes on to look at market efficiency and the relationship between different exchanges. Ultimately the author argues that prediction using a non-linear ANN using inputs from individual forecasting models is generally better than a linear model.

Thawornwong et. al (2003) focuses on the use of technical indicators as inputs to neural networks. The authors examine three neural networks in the prediction of the trend signals of three stocks. The authors conclude that technical indicators predictive ability

of is improved through the use of neural networks. Furthermore the authors suggest that neural networks are apt in uncovering irregularities in stock signals and profits. More specifically the authors suggest that the use of neural networks to make decisions on trades allows investors to trade given contradictory signals from technical indicators. Thawornwong et. Al (2003) falls short of a thorough analysis of technical indicators and thus opens the door to further study and analysis of the use of neural networks in stock market prediction.

Mohan et. al (2005) examines the use of stock market neural network prediction in the context of the Bombay Stock Exchange (BSE). The authors use two neural networks each with three hidden layers into which the weekly closing, moving average and oscillator values are fed. The authors results give a Root Mean Square Error (RMSE) of below 5% for both neural networks. While the results of this research are notable the authors selected only a limit range of inputs and fixed topology for the ANN used; it would be invaluable to explore the effect of ANN topology and input variations in the predictive ability of the ANN.

Pacelli (2008) utilises ANN to assess the influence that various phenomena and variables have on the stock returns of bank. Similar to other research the author selects a fixed ANN topology of 27 inputs, one output and three hidden layers of 15, 10 and 10 neurons respectively. The author concludes that only bank management has "confidential and qualitative information that should be used as input of an artificial neural network". While the author does look at a range or inputs to the ANN, they do not explore the variations in the ANN topology and what affect these may have on the accuracy of the result.

Papers that include technical indicators as network inputs include Yao and Tan (2000) who evaluated the effectiveness of a time series model on foreign exchange markets. The authors incorporated moving average technical indicators into their model. Results shows that with the exception of the Yen - US Dollar exchange the use of technical indicators improved results.

Dempster et al. (2001, July) considers genetic programming and reinforcement learning for the purposes of intra-day foreign exchange trading. Specifically the paper considers trading rules in terms of eight technical indicators as suggested by Jones (1999), Dempster & Jones (2000, Dec) and Dempster & Jones (2001b). The paper uses price channel breakout, adaptive moving average, Relative Strength Index (RSI), scholastics, Moving Average Convergence Divergence (MACD) crossover, momentum oscillator and commodity channel index. Variations of many of these indicators used by Dempster et al. (2001, July) were utilised in the simulations in this paper; such changes were necessary due to the difference in application, namely genetic programming versus the neural networks used in this paper.

2.8 Summary

There is no substantial research which utilises a range of technical indicators in the prediction of financial time series. Based on this there is certainly scope to explore the relative effectiveness of various topologies when applied to a wide range of technical indicators in the prediction of equity market price movements.

Chapter 3 looks at a variety of neural network implementations and considers their effectiveness as a precursor to Chapter 4 where the final network topology is set.

CHAPTER 3 - PRELIMINARY NETWORK TESTING AND ANALYSIS

Initially a range of network architectures were implemented to determine the best type of architecture of the data. The network architecture initially implemented in this thesis is shown in Figure 3.1.

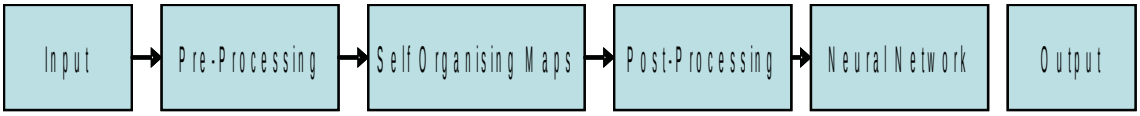

Figure 3.1: Network Architecture

This basic design was proposed and implemented in Giles et. al (2001), however the application has been transferred from international exchange markets to equity markets. Extension to this basic model has been performed in the ability of the network to undertake various types and combinations of neural network components and parameters. Pre-processing has been extended to include a wider range of alternatives. The SOM stage has been trained with a broad range of dimensions in addition to a stage of post-processing before SOM data is fed into the main neural network. The neural network phase was simulated with both standard Feed-Forward and Elman network types.

The trend prediction neural network was programmed in Matlab and took 23 training parameters. A wrapper function was written that called the trend predication neural network training program with various combinations of parameters. This design allowed better control of trend prediction in addition to a separation of functionality.

3.1 Data Selection

Fifteen years of historical data was used for the analysis. The 15 years of data is comprised of all stocks listed on the New York Stock Exchange from January 1st 1985 to December 31st 2000. Swingler (1996) and Schwager (1999) suggest that this length of data is sufficient in forecasting price trend movements. Data was obtained from Yahoo Finance (Yahoo, 2007).

Equity data of a homogeneous nature was selected on which to do analysis. Equity markets and their accurate prediction is an area where much research has been undertaken. The advantage to using equity markets is that traditional market prediction techniques can be used in combination with neural networks. When compared to other continuous data such as tides or river levels, equity markets do not necessarily follow such clear trends and are inherently more volatile in nature than these other systems, the causes of which, such as the cycle of the moon for tidal levels, make their prediction easier.

As stated previously, Yahoo Finance was selected as the source of market information. Yahoo Finance is a well-established and reliable source of equity prices on markets around the world.

3.2 Data Pre-processing

Simple normalisation was implemented to ensure that the mean and standard deviation of the data were zero and one respectively (Equation 2),

$$I = \delta_t - \frac{\overline{\delta}}{\sigma(\delta)} \qquad (2)$$

where $\overline{\delta}$ is the mean of the input data in the current input window and $\sigma(\delta)$ is the standard deviation of the data in the current input window differences.

Another form of pre-processing implemented in this thesis is logarithmic scaling. Logarithmic scaling makes better use of input data scope by evenly spreading data across the input range (Equation 3). This is a useful approach when reducing the effect of outliers (Giles et. al, 2001). Due to the sequential nature of equity price data and the importance of inter-day price changes outlier elimination, as outlined in Swingler (1996), was not used.

$$I_t = \begin{bmatrix} \log(|(\delta_t)|) & , & \delta \geq 0 \\ -\log(|(\delta_t)|) & , & \delta < 0 \end{bmatrix} \qquad (3)$$

Pre-processing can greatly influence the effectiveness of a network. Careful pre-process selection can increase the success of a networks output. In line with this, various combinations of pre-processing were trialled. The various pre-processing combinations are further explored under the implementation details (See Section 3.4 below).

Self Organising Map (SOM) stage was introduced to reduce the number of inputs to the neural network. This methodology groups input data into classifications according to data similarity, which in turn limits the number of input weights required.

The addition of noise to neural network training data helps to reduce the risk of over-training therefore allowing the network to generalise. While raw market data is inherently noisy, data passed into the neural network from the optional SOM stage was assessed with various amounts of post-processing including both the addition of random noise as well as normalisation.

A separate set of data was used for network error verification with network training being stopped once the verification set began to increase (Shadbolt, 2002).

In this implementation, up to two stages of pre-processing were applied. To determine the effectiveness of various combinations of pre-processing each stage could also be turned off, in effect acting in a simple pass-through manner.

Stage one of pre-processing calculates simple or logarithmic differences between days. A simple difference calculates the percentage between the current and previous days. Days on which trading did not take place were ignored (Yao and Tan, 2001). The logarithmic difference performs a log transformation on the simple difference (Equation 3). Logarithmic scaling also was undertaken on a simple difference between the current and first days trading (Equation 4). Again a simple pass-through mechanism was allowed (Equation 5).

$$I_t = \begin{bmatrix} \log(|(\gamma_t)|) & , & \gamma \geq 0 \\ -\log(|(\gamma_t)|) & , & \gamma < 0 \end{bmatrix} \quad \text{where} \quad \gamma_t = D_t - D_1 \qquad (4)$$

$$I_t = D_t \quad (5)$$

Stage two of pre-processing implemented pre-process normalisation (Equation 2). The mean and standard deviation were taken over the current input window and not over the entire input data.

The combination of these two stages allow for a total of 10 combinations of preprocessing. Testing of all combinations allowed for determination of the more optimal form of preprocessing.

3.3 Self Organising Map & Post Processing

A feature extracting Self Organising Map stage was optionally used to cluster similar input data. This allows for a large reduction in the number of inputs to the neural network. Each SOM works on one set of data only. SOM were used on a variety of combinations of source data and pre-processing functions. Self Organising Maps are able to organise input data into similar groups and this organisation reduces the number of neural network inputs, in turn limiting the effects of the curse of dimensionality. The size of the SOM was varied between 1-by-4, 1-by-8, 5-by-5 and 8-by-8. The SOM stage of the network was optionally removed.

Two SOM post-processing phases were added to the network design. The first stage of normalisation introduced noise to the SOM output, while the second optionally normalised the output of the SOM to a standard deviation of one and mean zero, similar to that performed in Equation 2.

3.4 Neural Network

Two varieties of neural network tested were Feed-Forward back propagation & Elman back propagation, representing non-recursive and recursive neural network types respectively. The input layer to the neural network was varied to include a number of data chunks either fed from the SOM or where no SOM was used, bypassing the SOM stage. These data chunks were allowed to be overlapping. Each chunk is created by taking a sample of *chunksize* days. An increment factor allows chunks to overlap i.e. if the increment factor is smaller than the *chunksize* then the next chunk will overlap by

(*chunksize – increment*) days. For example, three chunks of increment 10 and size 20 would cover 40 days as illustrated in Figure 3.4.1.

The number of chunks and *chunksize* parameters were kept constant at 4 and 10 respectively for the simulation.

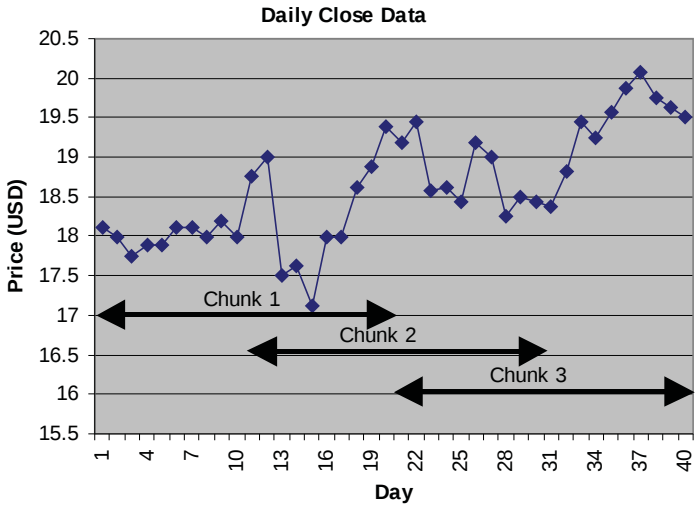

Figure 3.4.1: Thee chunks of 20 days with a 10-day increment

The Elman network was chosen as suggested by Giles et. al (2001). Elman networks have feedback to all hidden nodes from each hidden node (Figure 3.4.2). For comparative reasons, a standard feed-forward network was also used in the simulation. A feed forward network has no internal state memory therefore its predictive ability is limited to the data provided to the inputs at the current time instance.

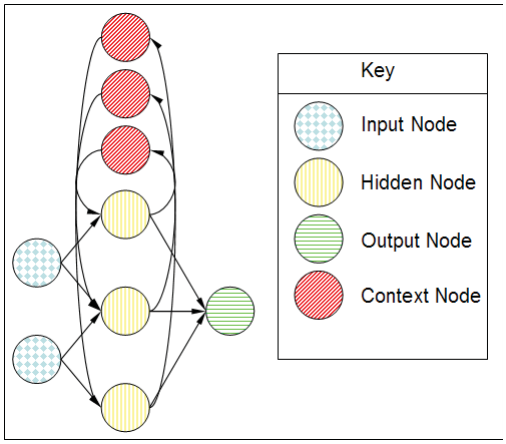

Figure 3.4.2: Elman Network

Various parameters for the neural networks were experimented with during training. Only one layer of hidden nodes was used in the experimentation. The number of nodes in the hidden layer was varied between 3, 5 and 10.

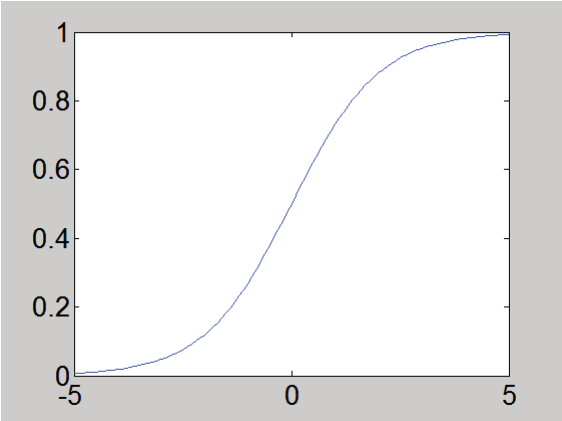

Figure 3.4.3: Log-Sigmoid Transfer Function

The transfer function used by the neural network was log-sigmoid (Figure 3.4.3). The transfer function was not modified during the training of the network.

3.5 Results

The network was trained to predict if a defined number of days ahead of time would be, on average, higher or lower than reference days in the training set. Thus the network was only expected to predict a ‘higher’ or ‘lower’ result from training data. Outputs to

the neural network were tested in both singular and dual arrangements. The singular configuration has only one output from the neural network; this output is set to near-one for a 'higher' prediction and near-zero for a 'lower' prediction (Equation 6). Singular output configuration is classified as correct if both the predicted and actual future values are both higher or lower than 0.5 (Equation 7), with a correct and incorrect prediction are defined as 1 and 0 respectively.

$$T_t = \begin{bmatrix} 0.8 & , & D_t \geq D_{t-1} \\ 0.2 & , & D_t < D_{t-1} \end{bmatrix} \quad (6)$$

where

Dt = Daily price data for day x

Tt = Target trend direction for day (t –1) to day t

$$Correct_t = \begin{bmatrix} 1 & , & P_t > 0.5 \wedge T_t > 0.5 \\ 0 & , & otherwise \end{bmatrix} \quad (7)$$

where

Px = the predicted trend direction from day (*x* –1) to day *x*.

In the case of dual outputs, these were simply set to alternate values (one high, one low) depending on if the target price was higher or lower than the reference days (Equation 8). Classification of correct results were categorised if actual and predicted results were of the same polarity (Equation 9). The target and predicted trend are defined as primary and secondary for reasons of clarity.

$$U_t = 1 - T_t \quad (8)$$

where

Ut = Secondary Target trend direction for day (x –1) to day x.

$$Correct_t = \begin{bmatrix} 1 & , & T_t \geq U_t \wedge P_t \geq Q_t \\ 0 & , & otherwise \end{bmatrix} \quad (9)$$

where

Qx = Secondary Predicted trend direction for day (x –1) to day x.

3.5.1 Mechanical Prediction

For comparative purposes, simple mechanical experimentation was carried out on the sample data collected. The intention of this analysis was to verify if a simple moving average could predict the next trading day. This tested whether the moving average increase from the previous day to the current day predicted that the following day would also be higher than the current (Equation 10) and visa versa for lower days. This test was carried out based on a 1, 5 and 15-day moving averages.

$$P_t = \begin{bmatrix} 0.8 & , & A_t \geq A_{t-1} \\ 0.2 & , & A_t < A_{t-1} \end{bmatrix} \qquad (10)$$

where

Pt = Artificially predicted trend direction

At = Moving average of daily values

Due to the enormous number of combinations of parameters that could be fed into the neural network, a batch layer was used to cycle through the possible combinations. Limitations in available computing time limited the number of combinations that could be tested. Fortunately, the addition of the batch-processing layer revealed quickly which parameters greatly impacted the performance of the network and which had little or no effect.

3.5.2 Neural Network Outputs

Outputs from the neural network were either dual or singular. Preliminary testing found a singular output gave vastly superior performance to that of dual outputs. In order to reduce the computational requirements, the results were limited to a singular network output only.

3.5.3 Data Pre-Processing

Both stages of pre-processing proved highly important in the predictive ability of the network. Results revealed that in the first stage of pre-processing, performance was not significantly altered between the simple-difference and logarithmic-difference preprocessing methodologies. However, the performance of the network was negatively

affected by setting this first pre-processing stage to pass-through. The average percentage of correct predictions with pass-through was just 55.7% versus 72.7% for other forms of pre-processing.

Normalisation of the second stage of pre-processing proved to have a significant impact on the results of the neural network. Without normalisation, the network predicted results with a lower accuracy of around 55% and narrow standard deviation of approximately 3% (Table 3.5.1). By contrast with normalisation, results gave a much higher average accuracy in the range of 73% but with a wider standard deviation of approximately 15%.

Table 3.5.1: Effectiveness of Pre-processing

Pre-Process (Stage 1)	*Pre-Process Normalisation (Stage 2)*	*Average Success (%)*	*Standard Deviation (%)*
None	*N*	*55.57*	*2.64*
None	*Y*	*55.85*	*2.98*
Relative Logarithmic Difference	*Y*	*73.60*	*14.68*
Origin Logarithmic Difference	*N*	*56.14*	*3.62*
Origin Logarithmic Difference	*Y*	*72.03*	*15.16*
Relative Simple-Difference	*Y*	*73.18*	*14.87*

These results show the effectiveness of normalisation on the output of the neural network (Table 3.5.1). Without normalisation, results were significantly degraded. Additionally, without any form of differencing, correct predictions were also low. It is interesting to note that while normalisation and pre-processing yielded a higher average result, the subsequent larger standard deviation shows that the consistency of the result was reduced.

3.5.4 Self Organising Maps

The addition of post-processing normalisation to the SOM did not yield a significant improvement or degradation in predictive ability of the network (Table 3.5.2).

Table 3.5.2: Effectiveness f Post-Processing Normalisation on Self Organising Maps

Pre-Processing	*SOM Post-Processing*	*Average Success (%)*	*Standard Deviation(%)*
Relative Simple-Difference	None	56.46	3.98
Origin Logarithmic-Difference	None	58.94	5.23
Relative Simple-Difference	Normalised	58.18	3.30
Origin Logarithmic-Difference	Normalised	57.97	3.22
Relative Logarithmic-Difference	Normalised	58.13	3.51

The effective of adding varying amounts of noise to the SOM was examined. The results show a minor drop in the average success of the neural networks predictive ability as noise is added (Table 3.5.3).

Table 3.5.3: Effect of Introduced Error on SOM Stage

Average Percentage Error Added	*Average Success (%)*	*Standard Deviation (%)*
0%	58.47	3.77
2.5%	58.41	3.36
5.0%	58.20	3.76
12.5%	57.21	3.40

Changes to the size of the SOM showed accuracy of the network output was improved with a larger 1-dimensional SOM (Table 3.5.4).

Table 3.5.4: Effectiveness of SOM dimensions

SOM Size	*Average Success (%)*	*Standard Deviation (%)*
1 x 4	57.82	3.56
1 x 8	58.30	3.64
5 x 5	51.86	1.47
8 x 8	55.74	3.29

Removal of the SOM from the network architecture was also tested. The results showed a significant increase in the average success of the network. This result is in contrast to that proposed by Giles et. al (2001). Note that SOM noise and normalisation post-processing phases are, of course, not used when the SOM phase is not present. Origin

relative logarithmic differencing yielded a slightly worse average success rate when compared to other forms of differencing. Additionally, the consistency of the result was compromised by the removal of the SOM layer, as revealed by an increase in standard deviation of results.

Table 3.5.5: Average Success with SOM Stage Removed

Pre-Processing	*SOM Post-Processing*	*Average Success (%)*	*Standard Deviation (%)*
Relative Simple-Difference	No SOM	77.32	14.05
Origin Logarithmic-Difference	No SOM	73.30	15.17
Relative Logarithmic-Difference	No SOM	77.58	13.81

The trend prediction network was very successful in its ability to predict price direction with prediction ability averaging 77% (Table 3.5.5).

3.5.5 Neural Network Type and Configuration

The network type was modified with and without a SOM stage to show the impact this would have. Results from simulation showed no significant difference in performance between Elman and standard Feed-Forward network topologies. Results demonstrated a direct correlation between the number of hidden nodes and average predictive ability when input was taken from the SOM stage. Conversely, when input was taken directly from the preprocessed data no significant difference in average performance was demonstrated, however a slight improvement in variation of the results was observed (Table 3.5.6).

Table 3.5.6: Effect of Neural Network Type and Number of Hidden Layers on Network Performance

Number of Hidden Layers	*NN Type*	*SOM*	*Average Success (%)*	*Standard Deviation (%)*
3	Elman	Y	58.35	3.34
5	Elman	Y	57.23	3.82
10	Elman	Y	52.68	2.40
3	Feed-Forward	Y	57.85	2.92
5	Feed-Forward	Y	58.77	4.13
10	Feed-Forward	Y	52.69	2.54
3	Elman	N	76.47	15.63
5	Elman	N	75.88	14.02
10	Elman	N	76.47	13.68
3	Feed-Forward	N	76.13	15.11
5	Feed-Forward	N	76.03	13.96
10	Feed-Forward	N	75.06	13.30

Finally, the effects of predictive range and reference were examined. Predictive range is the range of trading days over which the neural network is to predict the future trend. Predictive reference is the historical trading-day range used for calculating relative trend movement measures in days relative to the last trading day. For example, a predictive range and reference of 1-5 and 1-16 define that the network will try to predict whether the next 5 trading days (predictive range) will be higher or lower on average than the previous 16 trading days (predictive reference).

The results of varying the predictive range (horizontal) and predictive reference (vertical) of the neural network are shown below for 3, 5 and 10 hidden nodes (Table 3.5.7, 3.5.8 and 3.5.9 respectively). The data below was obtained without inclusion of the SOM stage due to the higher predictive ability of a SOM-free network architecture. Averaging was used to simplify data representation.

Table 3.5.7: Network Predictive Success with 3 Hidden Neurons

	1-1	1-2	1-5	1-10	Average
1-20	91.41	87.15	72.49	68.57	79.90
1-16	88.45	83.90	74.37	66.39	78.28
1-6	92.85	76.08	62.83	63.72	73.87
1-3	86.27	74.59	63.38	65.06	72.33
1-1	92.50	76.55	69.79	66.49	76.33
Average	90.30	79.66	68.57	66.05	76.14

Table 3.5.8: Network Predictive Success with 5 Hidden Neurons

	1-1	1-2	1-5	1-10	Average
1-20	91.21	84.36	73.65	68.54	79.44
1-16	90.09	78.52	71.45	66.94	76.75
1-6	95.25	80.61	68.68	63.00	76.88
1-3	88.16	79.33	69.95	65.62	75.76
1-1	94.35	79.22	66.37	64.29	76.06
Average	91.81	80.41	70.02	65.68	76.98

Table 3.5.9: Network Predictive Success with 10 Hidden Neurons

	1-1	1-2	1-5	1-10	Average
1-20	97.26	82.24	73.64	64.90	79.51
1-16	97.23	81.98	72.57	65.76	79.38
1-6	97.38	80.14	71.75	57.67	76.73
1-3	95.52	76.94	72.55	62.45	76.86
1-1	91.50	77.21	63.7	63.50	73.98
Average	95.78	79.70	70.84	62.86	77.29

Figure 3.5.1 shows the plotted averages for 3, 5 and 10 hidden layers across varying predictive ranges. Graphical representation of averages clearly shows the detrimental effect of an increased predictive range on the networks predictive ability.

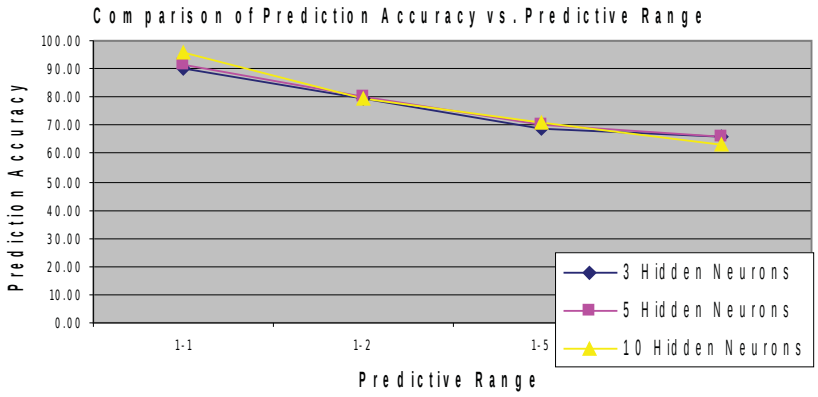

Figure 3.5.1: Predictive Accuracy vs. Predictive Range for Neural Network

Results of changes in predictive reference (Figure 3.5.2) show an increase in the predictive ability of the network as the size reference group is increased.

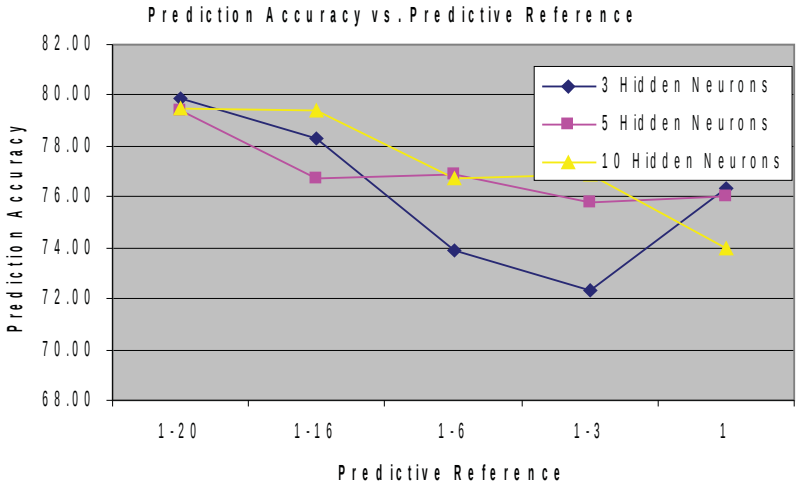

Figure 3.5.2: Predictive Accuracy vs. Predictive Reference for Neural Network

3.5.6 Mechanical Averaged-Based Prediction

Mechanical prediction as described in Chapter 3.5.1 (page 40) was carried out to verify if the network was simply carrying on the prevailing trend.

Table 3.5.10: Simple Average Trend Prediction

Average Type	*Average Success (%)*
1-day	*51.01*
5-day	*64.32*
15-day	*57.53*

This result demonstrates the effectiveness of a simple mechanical system. While an average success rate of 64% was achieved by the simple mechanical system when using 5-day averaging, it falls short of the result obtained by the neural network. The trained neural network was able to achieve a much higher level of accuracy with an average success rate exceeding 94% for a 1-1 predictive reference and range (Table 3.5.8). This demonstrates that the neural network was performing more than simple averaging to achieve its result.

3.6 Analysis

The results of experimental simulation have given tangible results for both pre-processing requirements and general network architecture. Results show that raw data

should always undergo some form of differencing, be it between the previous a day or a fixed reference point. The use of logarithmic function on this differenced data yielded no significant increase or decrease in the networks predictive ability. Furthermore, normalisation across the input window proved to be a critical element of the pre-processing procedure.

Self Organising Maps were found to decrease the network's predictive abilities. This result could be attributed to an over-simplification of training data. Subtleties in the training set time series was likely not reflected in SOM classification. Although a varying size of SOM were tested as suggested by Levine and Domany (2001), the poor result may be due to poor clustering and indicate that further simulation may could be considered with the aim of improving the result.

When using SOM's it was found that the addition of a post-processing phase was unnecessary. This post-processing phase added output normalisation and noise addition to the SOM-output data. While it was hypothesized that noise would reduce effects of over-fitting and therefore improve performance, it was found not to be the case in this circumstance. Noise was not added to input data as it is already inherently noisy. It was, however, found that when using a SOM stage, the size of the SOM had only a slight effect on the quality of the final neural network output.

Both Elman and standard Feed-Forward networks performed almost equally in the simulations. This suggests that information needed to predict the immediate direction of future prices is limited primarily to data contained in recent trading history. Comparatively, long term (seasonal) trends would very likely require more memory or a lower resolution such as weekly instead of daily statistics.

When the SOM stage was used, an increase in the number of hidden nodes in the second dimension resulted in a slight decrease in predictive ability of the network. However, when the SOM stage was excluded from the network, an increased number of nodes resulted in a very slight reduction in the variation of the network output.

To summarise, making use of normalised relative inputs into a neural network is significant in gaining an increase result. A SOM stage in processing gives worse results when predicting trend; however, when a SOM stage was used, it was best to keep a one-dimensional structure. Noise and normalisation added to the SOM stage had no effect on the predictive ability. Network type has no effect on network output, however, the use of more hidden layers proved beneficial to output variation when not using a SOM layer, and detrimental to average predictive ability when using a SOM layer.

3.7 Summary

The optimal configuration of the network has proven to be one without a SOM layer. Pre-processing should include relative-normalisation. To minimise the output variation, an increased number of hidden layers should be used. Best results were yielded with prediction calculated relative to a wider trading period for the following trading day only. This implies the difficulty in predicting multiple days into the future with such a model; however, next day price prediction, a more useful measure in real world applications, also yielded highly favourable results. When compared to a simple mechanical averaging form of trend prediction, the neural network trained achieved, on average, a higher success rate. Simple mechanical average trend prediction was only able to predict with 65% accuracy versus the optimally trained networks with predictive ability exceeding 90%.

Building on the results of this chapter, Chapter 4 discusses the final network topology.

CHAPTER 4 - INDICATOR, PRICE AND COMBINED NETWORK TOPOLOGY

The final network architecture used for this thesis is show below in Figure 4.1.

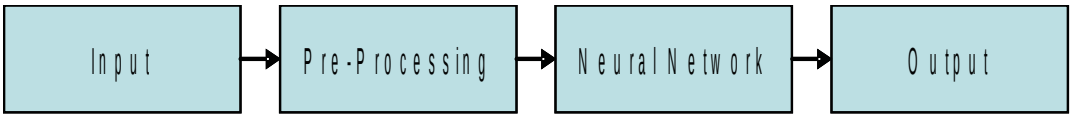

Figure 4.1: Final Neural Network Architecture chosen for prediction implementation

Two types of neural network were tested, indicator predictive and price predictive networks. Indicator predictive networks take inputs as preprocessed data in the form of technical indicators and attempt to predict the future direction of the indicator based on it's past. The second form of network implemented, namely price predictive, used these same indicators to predict the price direction of the stock in question. Price direction for these two network types were implemented on a strictly binary basis, with no analysis as to the magnitude of the prediction.

Due to it's poor performance, no SOM layer was used in this next phase of experimentation, nor was noise added to the inputs. As the first phase of results has determined that both the Elman and Feed forward architectures perform similarly, both were used in the next phase of training.

Various network sizes were tested as were the number of training days. According to Upadhyaya & Eryurek (1992) and Widrow & Lehr (1990) the amount of training data required increases with the number of network weights (Equation 11).

$$TrainingSetSize \geq \frac{Weights}{ErrorTarget} \qquad (11)$$

Seven network topologies, representing a broad range of structures, were chosen for the next stage of simulation. These topologies were:

a) 20-10-5-1 (20, 10 & 5 hidden nodes, 1 output node)
b) 10-5-1 (10 & 5 hidden nodes, 1 output node)
c) 5-1 (5 hidden nodes, 1 output node)

Based on previous results all networks used 30 days of inputs and a single output node, with the output node treated as giving a simple binary higher/lower signal (Equation 7).

As each layer's output is fed directly into each node of the next layer, the number of weights for each of these networks is the summation of the product of the *i*th layer multiplied by the (*i*+1)th layer. The amount of training data required varies from 20 days to 2550 days, or approximately 10 years of historical trading data.

Table 4.1: Weights and Training Data required for Feed Forward Network with a 0.1 Error Target

Network Topology	Weight Calculation	Weights Required	Training Data Required
20 - 10 - 5 - 1	20x10 + 10x5 + 5x1	255	2550
10 - 5 - 1	10x5 + 5x1	55	550
5 - 1	5x1	5	50

For an Elman network, the number of weights must account for the feedback loop from the output of the first layer of hidden nodes, calculated by adding the number of first layer hidden nodes squared to the number required for the comparative Feed-Forward network. This additional feedback layer increases the number of days of training data to increase to a maximum of 3550 days or approximately 14 years.

Table 4.2: Weights and Training Data required for Elman Network with a 0.1 Error Target

Network Topology	Weight Calculation	Weights Required	Training Data Required
20 - 10 - 5 - 1	(20x10 + 10x5 + 5x1) + 10x10	355	3550
10 - 5 - 1	(10x5 + 5x1) + 1x1	80	800
5 - 1	(5x1) + 1x1	6	60

The calculated number of training data required for the larger networks with hidden nodes is in contradiction to the 100 day sample size suggested by Giles et. al (2001) to minimise the effects of non-stationarity. It has also been found that 1-2 years of training

data will consistently outperform larger training sets (Walczak, 2001); thus, in order to determine the optimal number of training days was varied between 200, 400, 1000 and unlimited days. The unlimited days potentially allowed for up to 15 years of training data based; this varying in the time window allows determination of the most optimal window to allow for full representation of system dynamics (Frank, 2001). In situations where there were not a sufficient number of training days available to meet the training day requirements, the training and simulation based on that number of training days was not undertaken for the stock in question.

4.1 Pre-Processing - Technical Indicator Network Training

A financial time series may consist of up to 80% noise (Burgess, 1995) and, as such, the quality and preprocessing of data is essential in limiting the effect or filtering out much of this noise. Filtering has been limited to well known technical analysis functions and include such simple filtering features such as moving averages.

Various forms of preprocessing were analysed to determine the most effective preprocessing – neural network combination. Due to the sizeable amount of work undertaken in technical analysis, the pre-processing stage applied a number of well known technical analysis indicators and applied them to produce an output, which in turn, was then fed directly into a neural network. Not all technical indicators give the same form of output, but their outputs can be broken into the three basic categories of Buy/Sell/Hold, Top/Bottom and Open-High-Low-Close-Volume preprocessing.

The functions utilised by the neural network are show in Table 4.1.1:

Table 4.1.1: Network input functions

	Indicator	Appendix	Function	Function Type	Neural Network Type
1	Inter-day difference	A1	*InterdayClose(t)*	BSH	Price
2	Intra-day Difference	A2	*IntradayDiff(t)*	BSH	Price
3	Moving Average Crossover Indicator	A3	*CrossOver(t)*	BSH	Price
4	Moving Average Decayed Crossover Indicator	A4	*DecayedCrossOver(t)*	BSH	Indicator
5	Moving Average Difference Indicator	A5	*MovingAverageDiff(t)*	BSH	Price
6	Relative Strength Index	A6	*RSI(t)*	OHLCV	Price
7	Relative Strength Index RSI Analysis	A7	*TopBottom(t)*	BSH	Indicator
8			*BullishBearishA(t)*	BSH	Indicator
9			*BullishBearishB(t)*	BSH	Indicator
10			*BullishBearishC(t)*	BSH	Indicator
11			*BuySell(t)*	BSH	Indicator
12			*Local MaxMin(t)*	BSH	Indicator
13			*RSIAnalysis(t)*	BSH	Price
14	Bollinger Bands	A8	*BollingerDifference(t)*	OHLCV	Price
15	Volatility Analysis	A9	*CloseUpDownDay(t)*	OHLCV	Indicator
16			*VolumeUpDownDay(t)*	OHLCV	Indicator
17			*VolumePercentage(t)*	OHLCV	Indicator
18			*BollingerExpectation(t)*	OHLCV	Indicator
19			*BullishBearishVolatility(t)*	OHLCV	Price
20	On Balance Volume	A10	*OBV(t)*	OHLCV	Price
21	Accumulation Distribution Line	A11	*ADL(t)*	OHLCV	Price
22	Aroon	A12	*AroonOscillator(t)*	OHLCV	Price
23	Commodity Channel Index	A13	*CCI(t)*	BSH	Price
24	Bullish/Bearish Moving Average Convergence Divergence	A14	*BullBear(t)*	BSH	Price
25	Zigzag Filter	A15	*ZigZag(t)*	OHLCV	Price
26	Price Channels	A16	*BullBear(t)*	OHLCV	Price
27	Williams %R	A17	*WR(t)*	Trend	Price
28	Ultimate Oscillator	A18	*UO(t)*	BSH	Price
29	TRIX Data	A19	*Trix(t)*		Price
30	Standard Deviation	A20	*StdDev(t)*	OHLCV	Price
31	Money Flow Index	A21	*MoneyFlowIndex(t)*	BSH	Price
32	Chaikin Oscillator	A22	*ChaikinOscillator(t)*	Trend	Price
33	Rate of Change	A23	*ROC(t)*	OHLCV	Price
34	Money Envelope Analysis	A24	*MEA(t)*	BSH	Price

A brief explanation of the functions given in Table 4.1.1 is given below; a detailed description can be found in Appendix.

4.1.1 Buy / Sell / Hold

This indicator gives a direct indication of whether now is a good time to Buy, Sell or Hold (take no action) with the equity in question. These indicators are referred to as BSH indicators. Overbought and oversold indicators similarly signal time to sell and buy an equity respectively.

BSH indicators are commonplace, but suffer from a high amount of false signals and trashing. Thrashing is the situation in which the BSH indicator gives a series of buy or sell signals in a short time frame. Even if these buy and sell signals are technically optimal, trade transactional costs such as brokerage and the inability of a trader to trade precisely at these instances in time makes what is theoretically a small profit into a very real loss.

It is because of whipsaws and other false signals that a single BSH indicator cannot be solely relied upon, however a number of these indicators can form the basis for a broader trading strategy.

Inter-day Difference (Appendix A1)

The Inter-day Difference function calculates the difference between the the current and previous days close, with a positive output for a day where the close was higher than the previous day, and negative result for a day when the daily close was lower than the open.

Intra-day Difference (Appendix A2)

The Intra-day Difference function calculates the difference between the days open and close, giving a measure of the days volatility, with a positive output for a day where the daily close was higher than the open, and a negative result for a day when the daily close was lower than the open

Moving Average Crossover Indicator (Appendix A3)

The MACD function generates buy or sell signal when the 5-day moving average crosses above or below a 10-day moving average respectively.

Moving Average Decayed Crossover Indicator (Appendix A4)

As the MACD function output is strictly binary, giving a buy or sell indicator, a decaying factor of 0.5 was introduced. This decay factor essentially meant that the strength of the buy/sell indicator halved every day, resulting in the buy indicator being stronger when the moving averages first cross and becoming quickly weaker in the days following.

Moving Average Difference Indicator (Appendix A5)

The MACD difference indicator function returned the difference of the two moving averages, with the output being greater than, equal to or less than zero when the short-term moving average is above, equal to or less than the longer term moving average respecitvely. The motivation for this function was to greate an analog output that might be more neural network friendly.

Relative Strength Index (RSI) Analysis (Appendix A7)

The RSI Analaysi function applied to give a buy/sell/hold trigger based on well knowned properties of the RSI. The function itself was implemented the summation of the individual RSI properties.

Commodity Channel Index (Appendix A13)

The Commodity Channel Index (CCI) is a cyclical trend indicator developed by Donald Lambert in 1980. CCI values of above 100 and below -100 can be interpreted as overbought "buy" and oversold "sell" signals respectively.

Bullish/Bearish Moving Average Convergence Divergence (Appendix A14)

The Bullish/Bearish MACD function is a result of the MACD output divided by the a 9 day exponential moving average control line.

Money Envelope Analysis (Appendix A24)

Money Envelope Analysis indicates when a stock exceeds or drops below a moving maximum or minimum limit respectively. These limits are calculated as a percentage of

the current stock price; in this case 3% was selected.

Ultimate Oscillator (Appendix A18)

The Ultimate Oscillator uses multiple overlapping data-periods to measure overbought and oversold signals. The output of this function is a combination of three parameters made up of 7, 14 and 28 day overlapping periods.

Money Flow Index (Appendix A21)

Money Flow Index (MFI) is used to indicator the flow of money into or out of a stock. The output varies between 0 and 100 with an output above 80 considered an indication that the stock price will likely drop (a bearish signal); conversely a MFI of below 20 implies the opposite (a bullish signal).

4.1.2 Top / Bottom and Trend Indicators

Top-Bottom indicators signal when the equity in question is at a peak, or local maxima, and is likely to trend downward, or at a local minima, where the equity is likely to trend upwards. Obviously if an equity has reached its local maxima and is poised to fall, then it is a good time to sell, and conversely a local minima is a good time to buy.

As with BSH indicators, the Top-Bottom indicators cannot be relied upon in isolation, but does form the basis of a broader trading strategy for many traders. This form of indicator is an important means of predicting changes in the general trend of a stock, which in turn reflects the underlying market sentiment.

The Top / Bottom and Trend indicators used as inputs to the neural network include:

Williams %R (Appendix A17)

Williams %R shows the current closing price in relation to the low and high of the previous 14 days. This function outputted the raw output of the standard Williams %R function.

Chaikin Oscillator (Appendix A22)

The Chaikin function indicates the buying or selling pressure on a stock. The function is

calculated by taking the difference between the 3 day and 10 day exponential moving average of the ADL line.

4.1.3 Processed Open-High-Low-Close-Volume Data

Processed OHLCV (Open-High-Low-Close-Volume) refers to basic preprocessing of OHLCV data to present it in a form that is more suitable for a neural network. A simple example of this is that absolute values of a stock are of little relevance to a neural network, where a drop in an equities price from $10 to $5 being as significant as a drop from $100 to $50; preprocessing of such raw input data is a common step taken before feeding data into a neural network.

Relative Strength Index (Appendix A6)

The Relative Strength index gives an indication of a stocks recent gains versus it's recent losses and was applied without inter-day differencing.

Bollinger Bands (Appendix A8)

Bollinger Bands measure the volatility of a stock by effectively giving the output that is a function of the standard deviation of the stock's close. The larger the moving average, the more volatile a stock is.

Volatility Analysis (Appendix A9)

Volatility Analysis was implemented as the combination of a variety of various indicators to gauge the volatility of the market.

On Balance Volume (Appendix A10)

On Balance Volume (OBV) is a measure of volume flow of stocks traded and works on the premise that volume precedes price changes. The function simply adds or remove the daily volume to a running total.

Accumulation Distribution Line (Appendix A11)

The Accumulation Distribution Line, similar to OBV works on the premise that volume will increase or decrease before a price movement. ADL uses the close relative to the period range multiplied by the days volume.

Aroon (Appendix A12)

The AroonOscillator is a trend indicator, with an output higher or lower than zero being interpreted as an upwards or downwards trend respectively. The further the AroonOscillator from zero, the strong the trend.

Zigzag Filter (Appendix A15)

The ZigZag indicator is a form of extreme filter, removing all noise and following the basic trend of a stock. The ZigZag filter ignores all daily high to low changes below a certain percentage as specified by the sensitivity parameter of the filter. For simulation a 1.5% percent change inter-day low-to-low and high-to-high threshold.

Price Channels (Appendix A16)

Similar to Standard deviation price channels create upper and lower trend lines around a stock. For price channels, this trend line is based on the highest and lowest n-period close for the upper and lower n-period trend lines respectively.

Standard Deviation (Appendix A20)

A standard deviation indicator is simply a moving standard deviation performed to a stock to measure its volatility (Equation 104) and highlighted in Ford Equity Research (2002).

Rate of Change (Appendix A23)

The Rate Of Change (ROC) indicator is a simple momentum indicator that measure the rate of change of a stocks closing values (Schwager, 1999). ROC is calculated by taking a moving difference of a stock's current close with its close a number of periods ago (typically 10 days).

4.2 Summary

In Chapter 5 the input preprocessing functions and final network topology outlined here are used to train and analyse the best network topology. Details of the pre-processing functions can be found in the Appendix.

CHAPTER 5 - RESULTS AND ANALYSIS

Results of neural network training were compared against the two primary network components of network type and network topology. By adjusting other network parameters against these two primary parameters it enables detailed analysis of these primary parameters in addition to seeing the individual effects of changes to secondary parameters.

The network type was varied between price and indicator networks, and the neural network input training window was varied from previous day (“1-1”), previous five days (“1-5”) and the previous 15 day (“6-20”) networks. Training windows of 1-5 and 6-20 were only applied to price networks.

Indicator networks were not tested for 1-5 and 6-20 training windows. This is because indicator networks are designed to give a recommendation of actions that should be taken the next day trading day. These actions are based on known methods and criteria that are commonly employed among traders utilising technical analysis. Simply put, indicator networks are based on formula that focuses on specific data points, and not on a “window” of data as is used in price network analysis.

Finally, combined analysis of results separates the effect that primary parameters have against each other for given secondary configuration.

Results are tabulated and presented to show what percentage of the time a specific or given parameter gave the best results. By tabulating results in this manner, it makes it possible to determine which parameter changes give a discernible difference in the results, and which had little or no effect. These results are used to select the best network type or set of network parameters, with the topologies determined from these results later used to create sub-networks to feed into a combined network output.

5.1 Training Days

Training days are the number of days of training that are used to train the neural

network. Training days were varied between 250 and 2500 days, in addition to the number of days as suggested by Upadhyaya & Eryurek (1992) and Widrow & Lehr (1990) in Equation 11.

- As per theory (Equation 11)
- 250 (approximately 1 year of trading)
- 500 (approximately 2 years of trading)
- 1250 (approximately 5 years of trading)
- 2500 (approximately 10 years of trading)

The number of training days prescribed by Upadhyaya & Eryurek (1992) and Widrow & Lehr (1990) are given in Table 5.1.1. The required number of training days vary from only 55 days (11 weeks) to 8550 days (34 years).

Table 5.1.1: Calculation of training data required for given network types and topologies

Network Type	Input Perod	Network Topology	Requred Traininig Data Calculation	Weights Required	Training Data Required
Elman	10	10 - 20 - 10 - 5 - 1	10 x 20 + 20 x 10 + 10 x 5 + 5 x 1 + 20 x 20	855	8550
Elman	10	10 - 10 - 5 - 1	10 x 10 + 10 x 5 + 5 x 1 + + 10 x 10	255	2550
Elman	10	10 - 5 - 1	10 x 5 + 5 x 1 + + 5 x 5	80	800
Elman	5	5 - 20 - 10 - 5 - 1	5 x 20 + 20 x 10 + 10 x 5 + 5 x 1 + 20 x 20	755	7550
Elman	5	5 - 10 - 5 - 1	5 x 10 + 10 x 5 + 5 x 1 + + 10 x 10	205	2050
Elman	5	5 - 5 - 1	5 x 5 + 5 x 1 + + 5 x 5	55	550
Elman	1	1 - 20 - 10 - 5 - 1	1 x 20 + 20 x 10 + 10 x 5 + 5 x 1 + 20 x 20	675	6750
Elman	1	1 - 10 - 5 - 1	1 x 10 + 10 x 5 + 5 x 1 + + 10 x 10	165	1650
Elman	1	1 - 5 - 1	1 x 5 + 5 x 1 + + 5 x 5	35	350
Feed-Forward	10	10 - 20 - 10 - 5 - 1	10 x 20 + 20 x 10 + 10 x 5 + 5 x 1	855	8550
Feed-Forward	10	10 - 10 - 5 - 1	10 x 10 + 10 x 5 + 5 x 1 +	255	2550
Feed-Forward	10	10 - 5 - 1	10 x 5 + 5 x 1 +	80	800
Feed-Forward	5	5 - 20 - 10 - 5 - 1	5 x 20 + 20 x 10 + 10 x 5 + 5 x 1	755	7550
Feed-Forward	5	5 - 10 - 5 - 1	5 x 10 + 10 x 5 + 5 x 1 +	205	2050
Feed-Forward	5	5 - 5 - 1	5 x 5 + 5 x 1 +	55	550
Feed-Forward	1	1 - 20 - 10 - 5 - 1	1 x 20 + 20 x 10 + 10 x 5 + 5 x 1	675	6750
Feed-Forward	1	1 - 10 - 5 - 1	1 x 10 + 10 x 5 + 5 x 1 +	165	1650
Feed-Forward	1	1 - 5 - 1	1 x 5 + 5 x 1 +	35	350

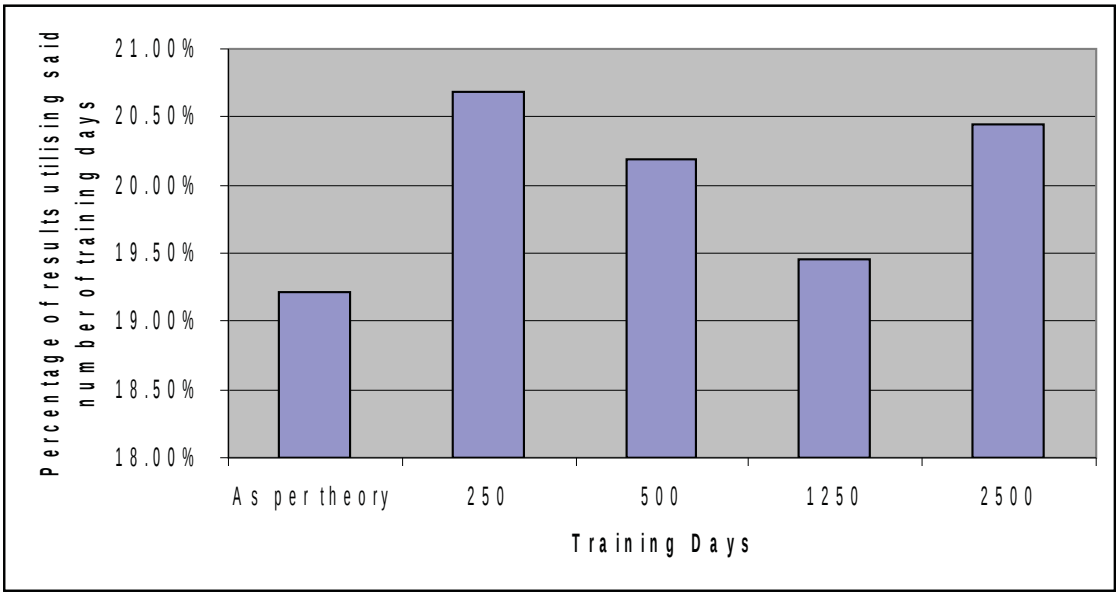

Figure 5.1.1:Distribution of results based on Performance summary variation in training days

Looking at the overall comparison (Figure 5.1.1) the most successful number of training days was 250 (1 year), with little evidence to support the training days described by Upadhyaya & Eryurek (1992) & Widrow & Lehr(1990).

Overall results worsen increasingly as the number of training days is increased. These results are roughly in line with Walczak (2001), who suggests a two year time-frame is optimal for neural network training.

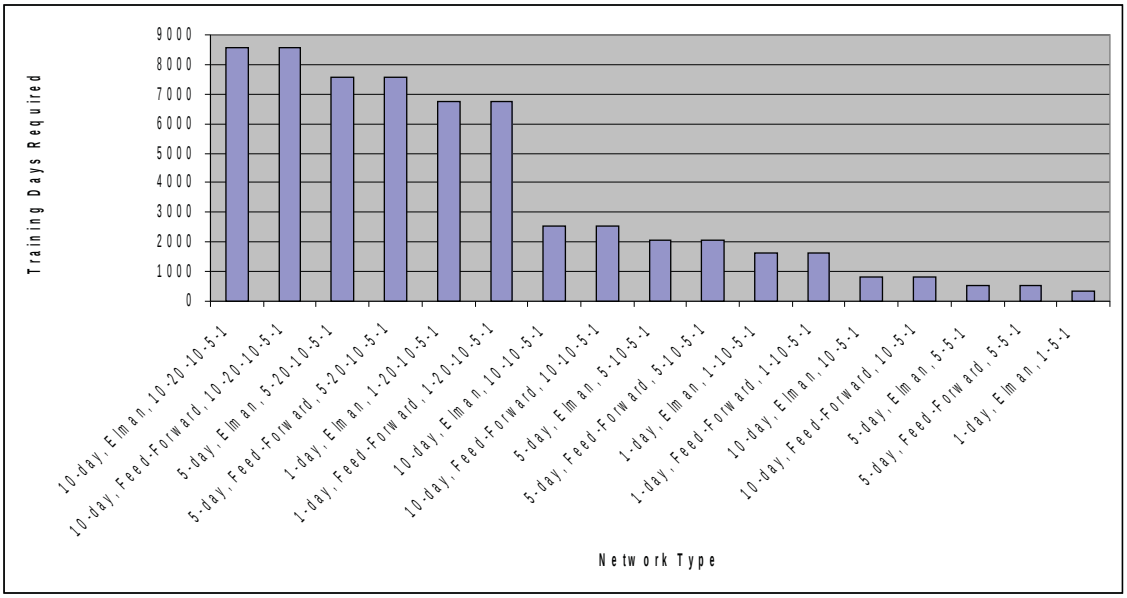

Figure 5.1.2: Number of training days required by versus network type

The noticeable rise in support for 2500 training days (Figure 5.1.1) correlates to the higher number of training days for the higher-complexity 20-10-5-1 network (Figure 5.1.2). This network should require on average 7600 days or approximately 30 years worth of training data if the theory prescribed by Upadhyaya & Eryurek (1992) and Widrow & Lehr (1990) is followed. It is realistic to assume that the results for this higher number of training days may not accurately reflect the performance of this network, due to the 15-year limitation on the training data.

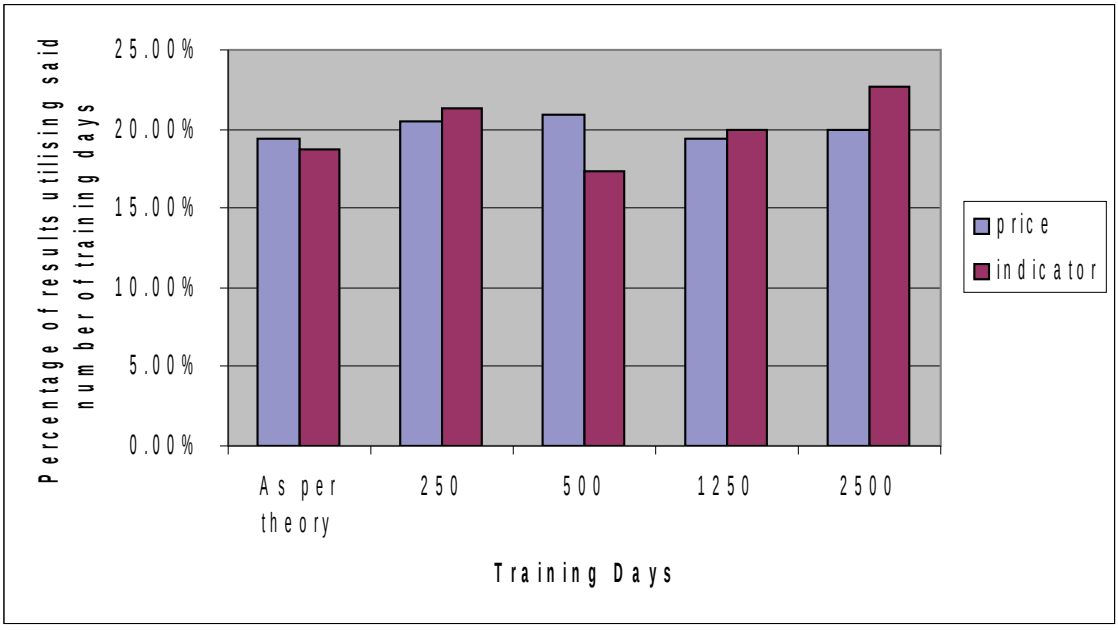

Figure 5.1.3: Distribution of Price and Indicator performance for various numbers of training days

Breaking down the results into price and indicator categories (Figure 5.1.3) shows that there was no significant difference that was gained by varying the number of training days. It should be noted that indicator networks performed slightly better if trained with 2500 training days when compared to the price networks.

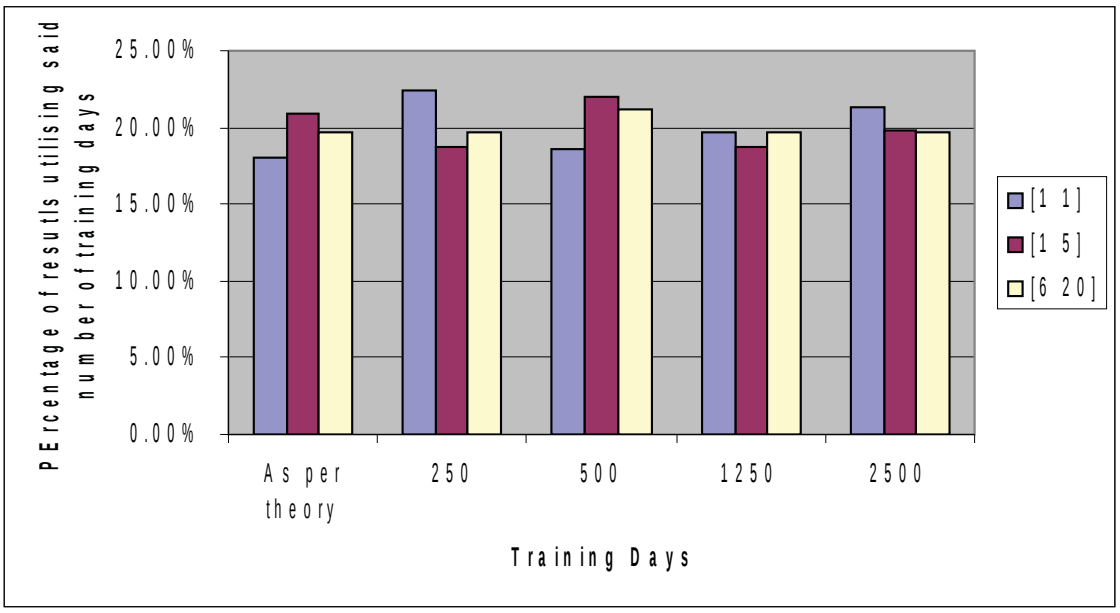

Figure 5.1.4: Distribution of results for Input period variation performance given variation in training days

Isolating results by network type (Figure 5.1.4) shows that 250 training days is more optimal for 1-1 networks, while 1-5 and 6-20 prediction periods are more suited to 500 training days.

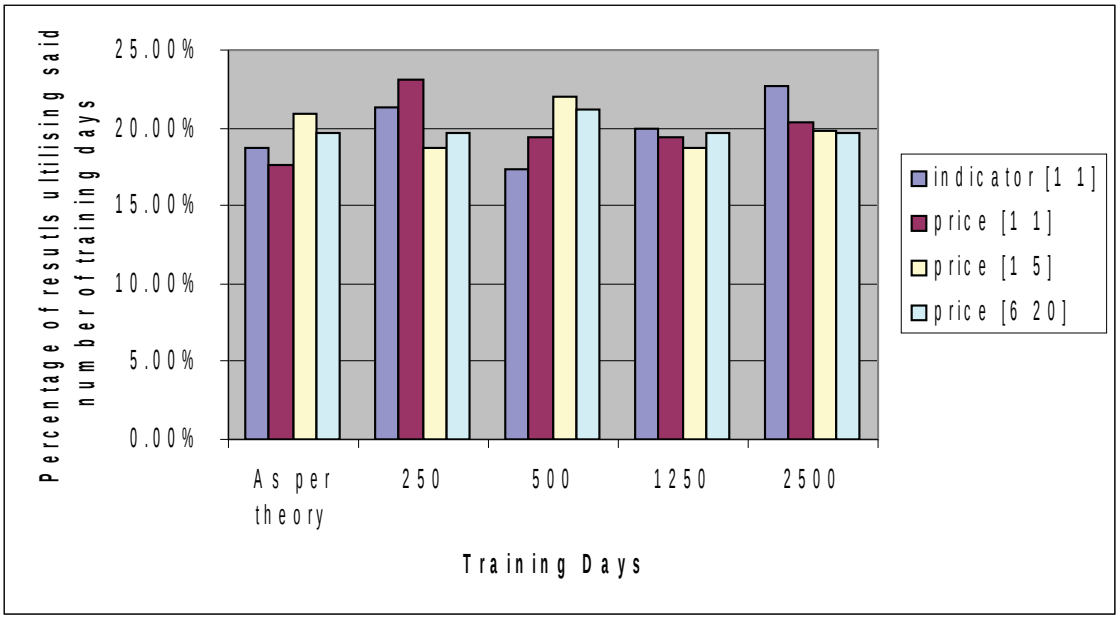

Figure 5.1.5: Price and Indicator combination performance given variation in training days

Finally, analysing training days by both topology and network type (Figure 5.1.5) reinforces the previous observations that 250 training days is more optimal for 1-1 price networks. 1-5 and 6-20 networks are better suited to 500 training days, while a larger proportion of results for indicator networks were in the category of 2500 training days.

5.2 Training Epochs

Training Epochs were kept static at 250 throughout the testing process.

5.3 Network Function

The network transfer functions of Tangent Sigmoid (tansig) and Logarithmic Sigmoid (logsig) were used for training. These transfer functions are used to effectively limit the output of the nth layer of a network before feeding the value as an input to the (n+1)th layer (Hecht-Nielsen, 1990).

The *tansig* and *log* functions are shown in Figures 5.3.1 and 5.3.2 respectively. *Tansi*g is the same shape as the hyperbolic tangent, *tanh*.

The function for *logsig*, as used in Matlab is as follows:

$$logsig\,(n) = \frac{1}{1+e^{-n}} \qquad (12)$$

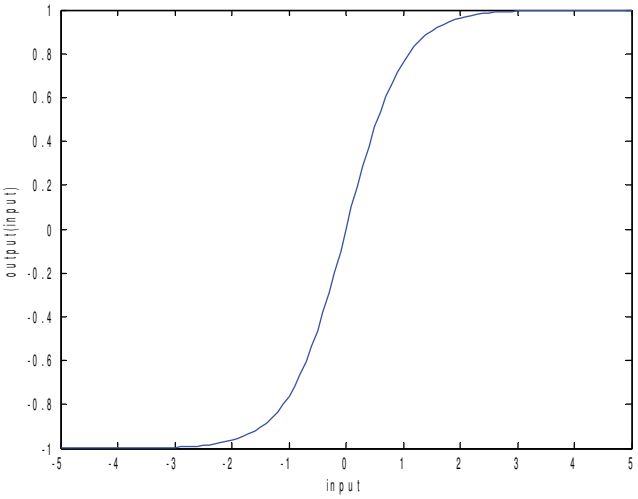

Figure 5.3.1: logsig Function

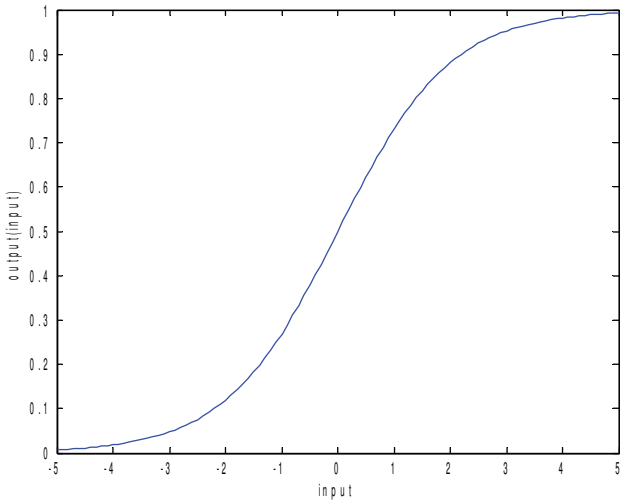

Figure 5.3.2: transit Function

From Figure 5.3.3 it can be seen that a slim majority of optimal results were derived from a *logsig* network function. The difference, however, is not sufficient to suggest that the *logsig* network function should be exclusively used instead of the *tansig* network function.

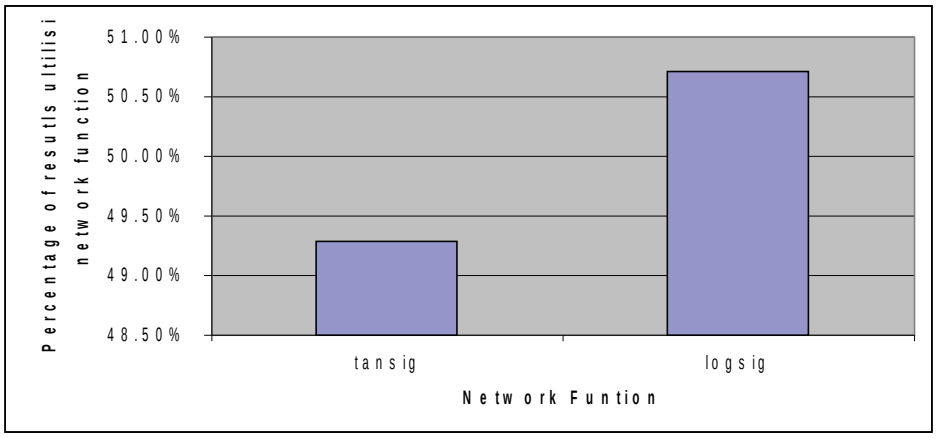

Figure 5.3.3: Distribution of results for variation in network function

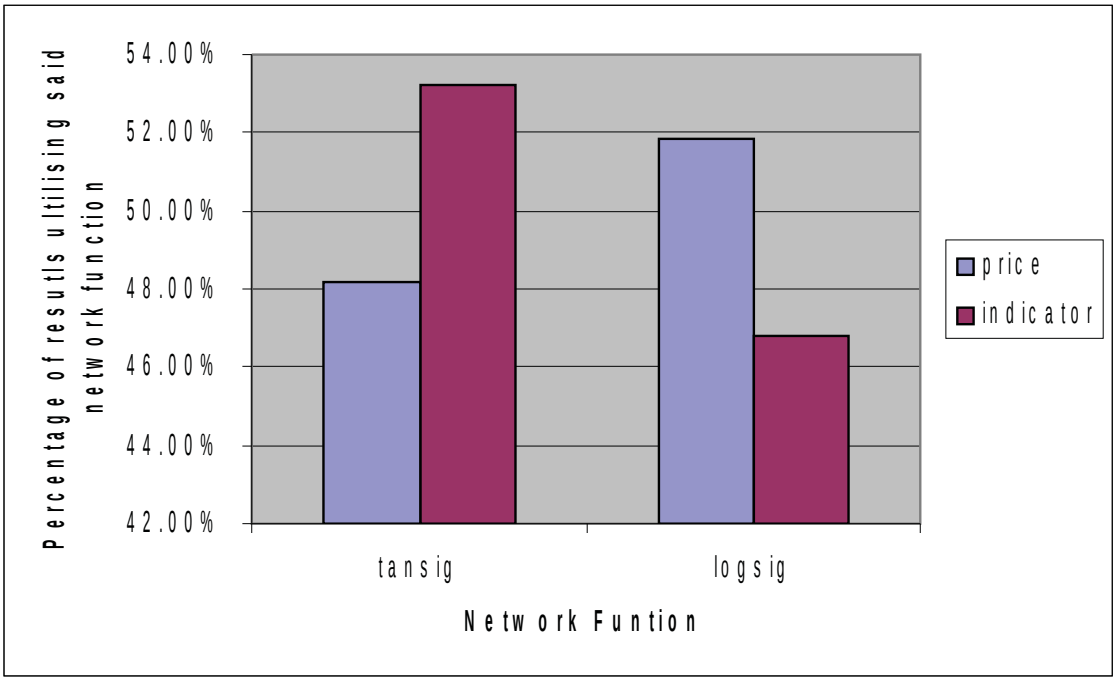

Figure 5.3.4: Comparison of results distribution between price and indicator networks

Analysis on the basis of price and indicator networks shows a converse relationship, with a larger proportion or optimal results for price network favouring the *logsig* network function, with the indicator networks biased towards a *logsig* network. This difference is notable and surprising, and may be linked to the less-continuous nature of the indicator network results.

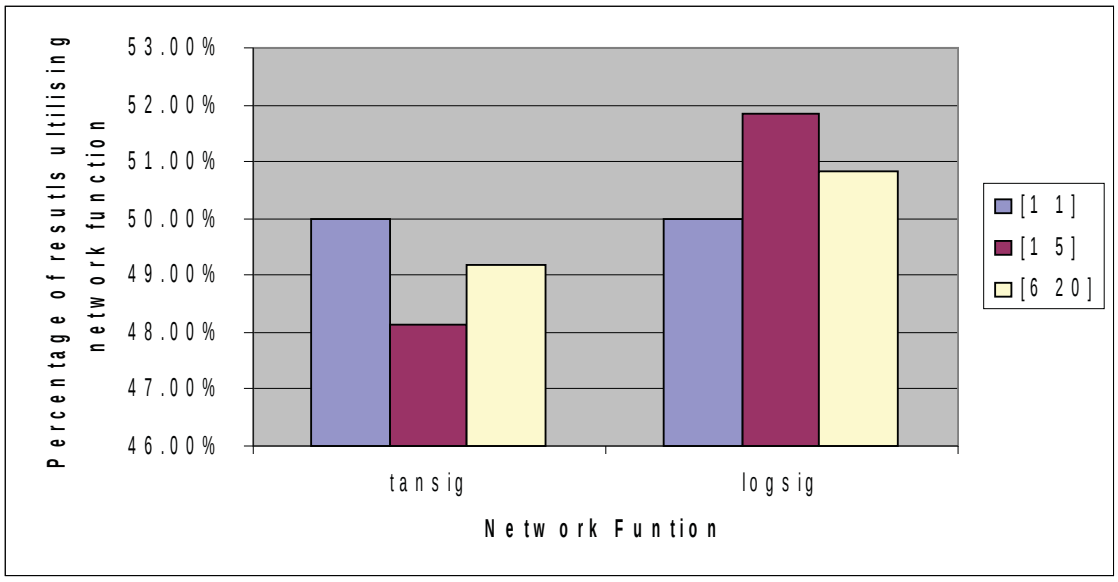

Figure 5.3.5: Comparison of results distribution for different network topologies

Analysis based on input period (Figure 5.3.5), shows that the neither the *tansig* and *logsig* networks were suited for a 1-day input period. The results also suggest that *logsig* networks are better suited for an input period of 5 and 15 days (1-5 and 6-20 networks respectively).

Such a result again suggests that the *logsig* network performs better for more unpredictable training data, such as would be found in the prediction of network performance over the 1st to 5th days and 6th to 20th days.

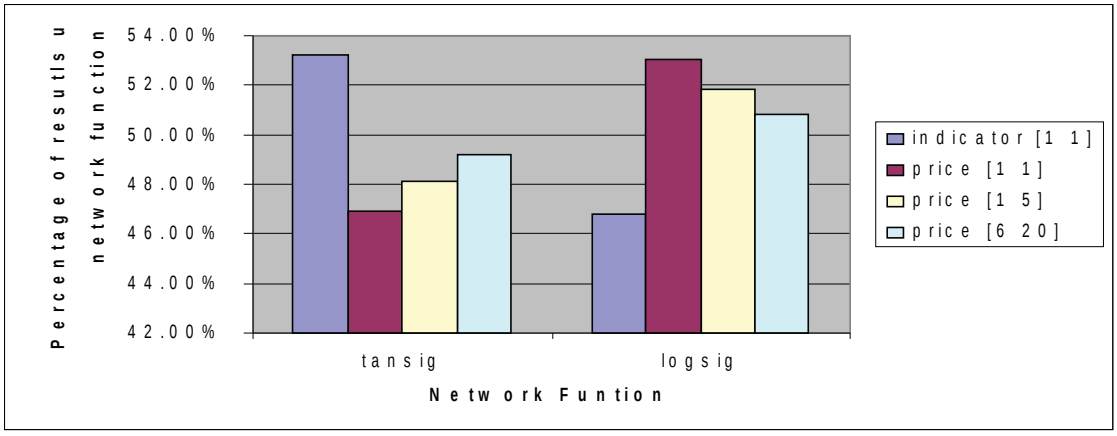

Figure 5.3.6: Comparison of results for different network topologies and type (price or indicator)

While indicator networks showed a strong bias towards *tansig* (Figure 5.3.6), the increase in the input period pushed the bias closer to *tansig* functions for price based networks. If the results for price networks were extrapolated, *tansig* networks may, in fact, out perform the *logsig* networks for long prediction price networks. Such long prediction is beyond what was assessed for these results and would require more in-depth analysis.

5.4 Training Goal

Training goals for the neural network were tested for 0.1%, 1%, and 5% error rates. By testing the most effective error rate, we are better able to ascertain an ideal level of training that neither under nor over trains the neural network.

Again training error rates were simulated against primary parameters of network type and network topology. 0.1%, 1%, and 5% error rates are denoted as 0.001, 0.01 and 0.05 respectively.

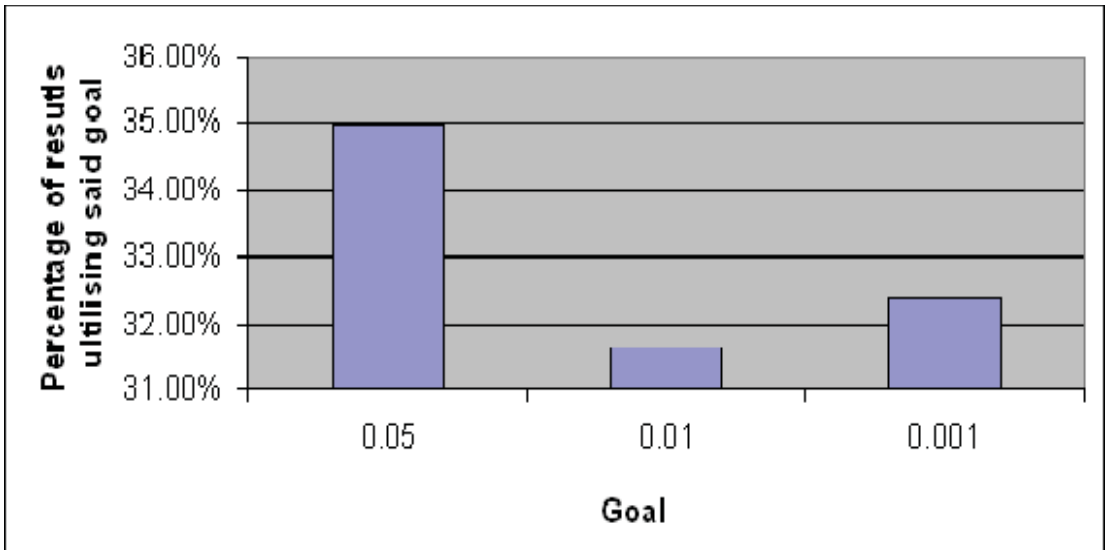

Figure 5.4.1: Distribution of results for training goals of 1%, 5% and 0.1%

The distribution of optimal results for 5%, 1% and 0.1% (Figure 5.4.1) shows that more optimum results were achieved with the 5% training goal. This suggests that a stricter training goal of 0.1% and 1% have led to over-training of the network.

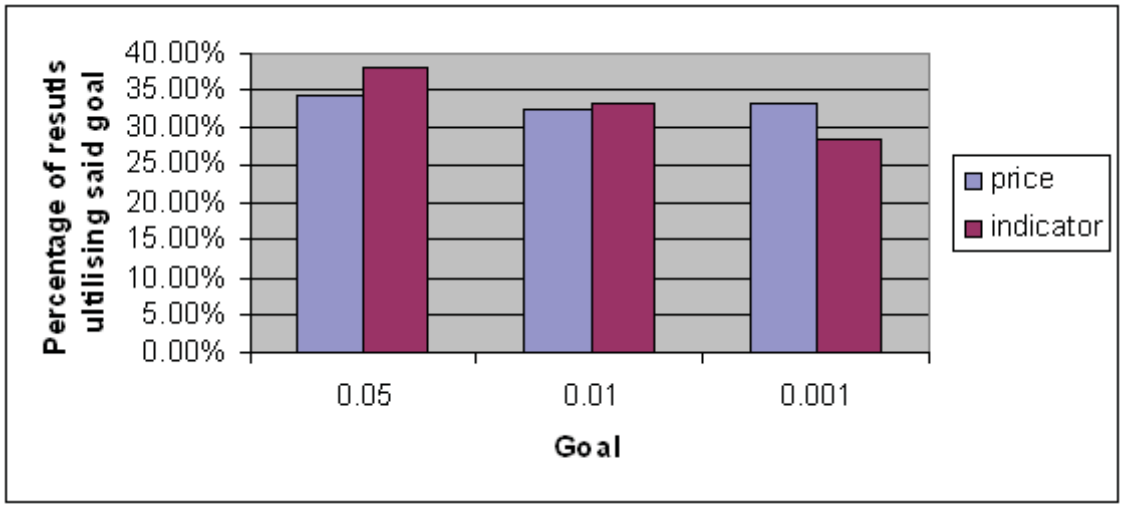

Figure 5.4.2: Comparison of price and indicator network type optimal results for given training goals

Analysis on a price – indicator level (Figure 5.4.2) shows that while price networks show no significant difference in the selection of the training goal, indicator networks consistently showed a stronger bias for the five-percent training goal.

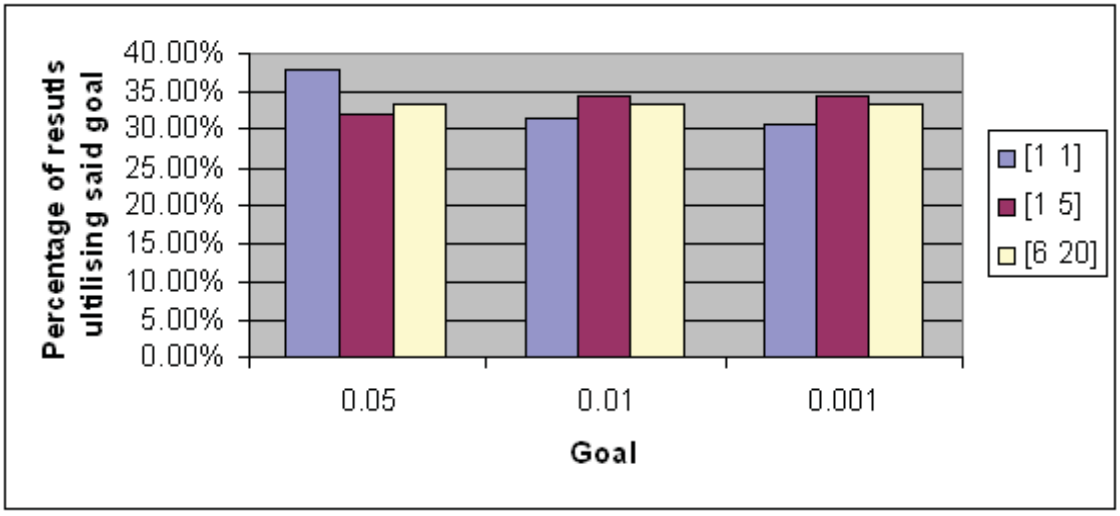

Figure 5.4.3: Comparison of network topology optimal results for given training goals

Looking at the input period analysis (Figure 5.4.3), it is evident that for a 1-day input period a five-percent training goal is best. 5-day and 15-day results do not show a significant proportion of optimal results for a particular training goal.

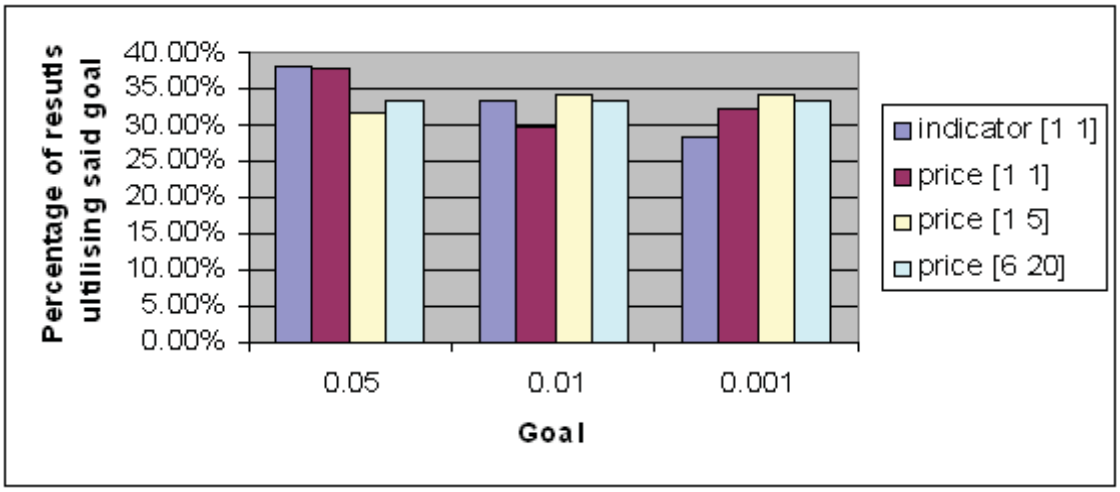

Figure 5.4.4: Comparison of network topology and type for given training goals

In summary, there is a bias towards the 5-percent training goal for 1-day indicator and

price networks (Figure 5.4.4), with little evidence of any preference for alternative 1-5 and 6-20 periods.

5.5 Input training period

The Input period is the size of the data block that was fed into the neural network. An input period of one means that only a single day was fed into the neural network, whereas a 10 day input period means that effectively 2 trading weeks worth of data (10 days) is fed into the network.

Results were compared to primary network components of network type and prediction period. Input periods of one day, one trading week (5 days) and two weeks (10 days) were tested.

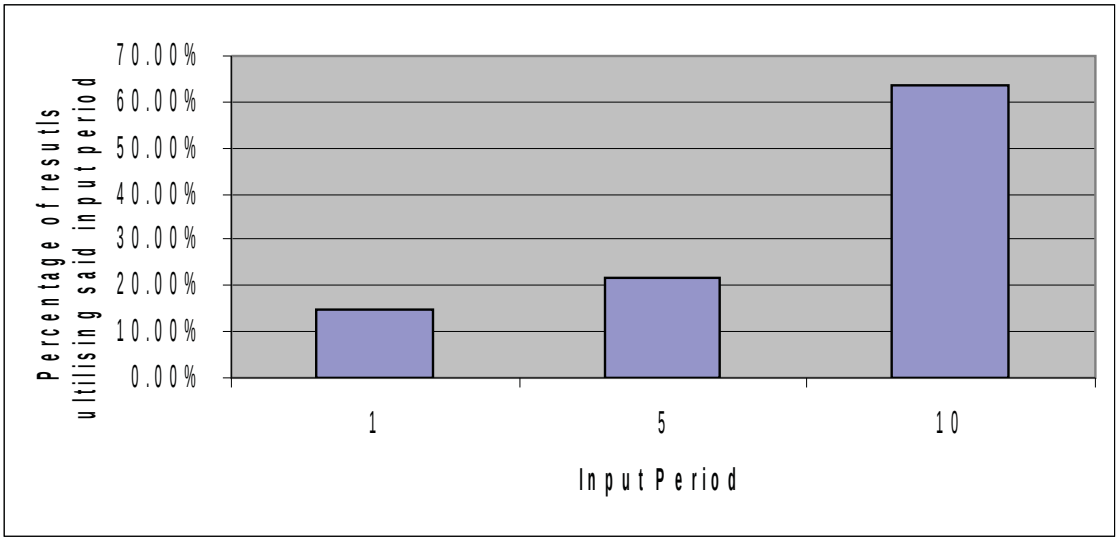

Figure 5.5.1: Input period optimal results distribution for given training periods of 1, 5 and 10 days

The summary of results for variation in the input period between 1 and 10 days (Figure 5.5.1) show a very strong bias towards longer input periods. This is consistent with the hypothesis that a larger input period is necessary to accurately predict the future direction of stocks.

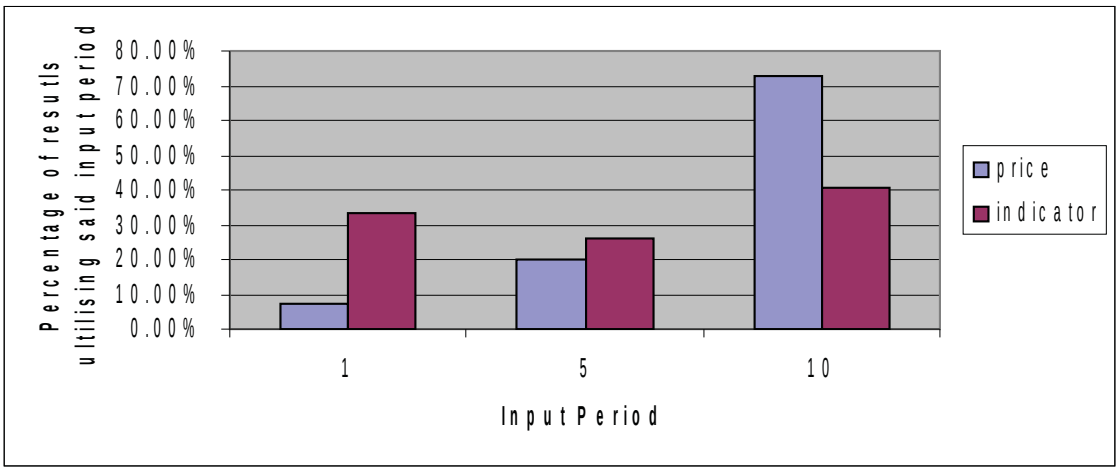

Figure 5.5.2: Input period optimal results distribution for price and indicator networks given training periods of 1, 5 and 10 days

Looking at the split between price and indicator networks (Figure 5.5.2) reveals a very strong bias towards a longer input period for price-based networks. Comparatively variation in the input period for indicator networks showed a lesser bias for either longer or shorter lengths.

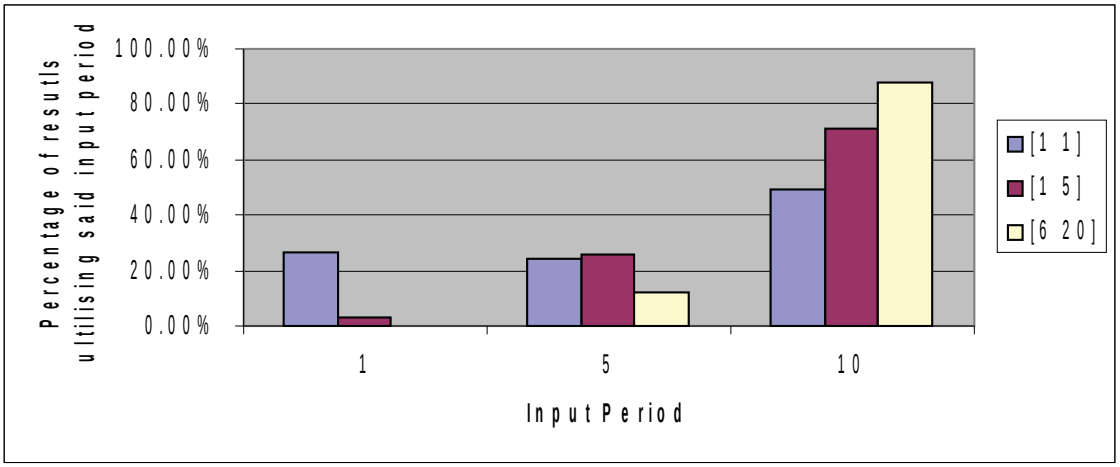

Figure 5.5.3: Network topology optimal results distribution for price and indicator networks given training periods of 1, 5 and 10 days

A closer look at the results when broken down into the prediction results (Figure 5.5.3) shows that a ten-day input period is very much superior to a lower input period. Closer examination of the results does reveal that while a 5 day input period was better than a 1-day period for 1-5 and 6-20 prediction periods, a “next day” 1-1 prediction period had

optimal results more evenly distributed between the input periods.

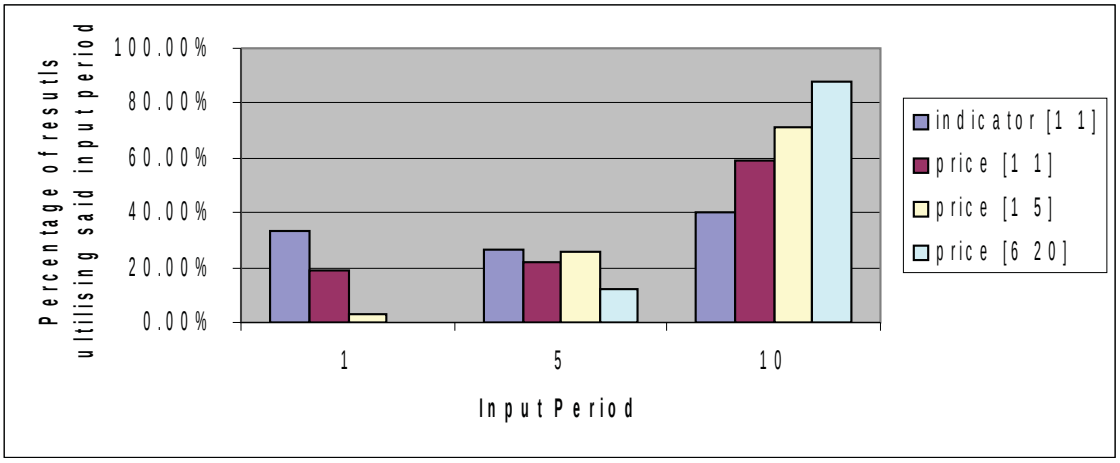

Figure 5.5.4: Input period and type optimal results distribution comparison for price and indicator networks given training periods of 1, 5 and 10 days

Finally, examining a comprehensive breakdown into indicative and price networks (Figure 5.5.4) shows that the indicator network was less affected than price networks to variation in training period. This sits reasonably well with typically shorter term outlook of many indicators, often with weighting that heavily favours recent data over historical data.

5.6 Input Scaling

Five types of Matlab input scaling functions were used:

- *Linear normalisation - linearnorm*
- *Linear scaled - linearscale*
- *Linear clipped - linearclip*
- *Logarithmic - logarithmic*
- *Softmax - softmax*

The distribution of results show that linear and logarithmic scaling comprised slightly more of the results than linear normalisation, linear clipping and softmax input scaling (Figure 5.6.1).

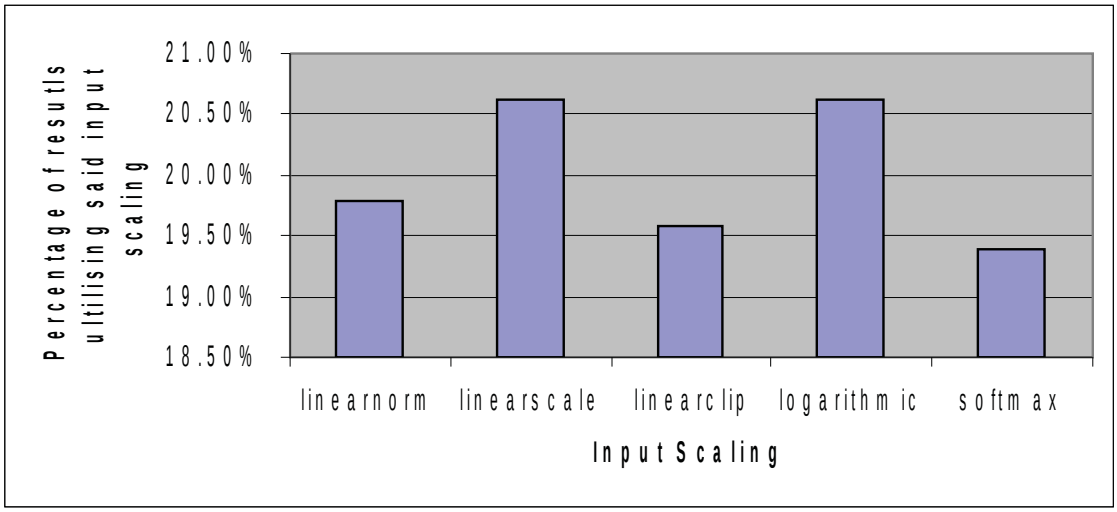

Figure 5.6.1: Comparison of optimal results for various types of input scaling

Results broken down by network types of price and indicator (Figure 5.6.2), show a slight bias towards linear normalisation and scaling for indicator networks.

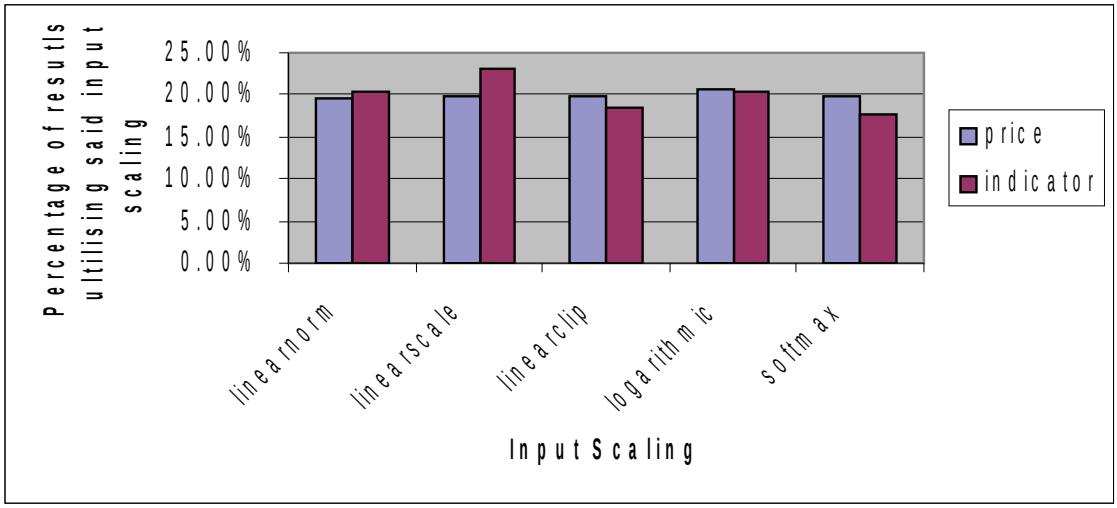

Figure 5.6.2: Distribution of optimal results for price and indicator networks given various input scaling functions Graph results

Further breakdown by input range (Figure 5.6.3) shows that logarithmic scaling was best used with a 5-day input period, while linear scaling was better used with a 1-day input period.

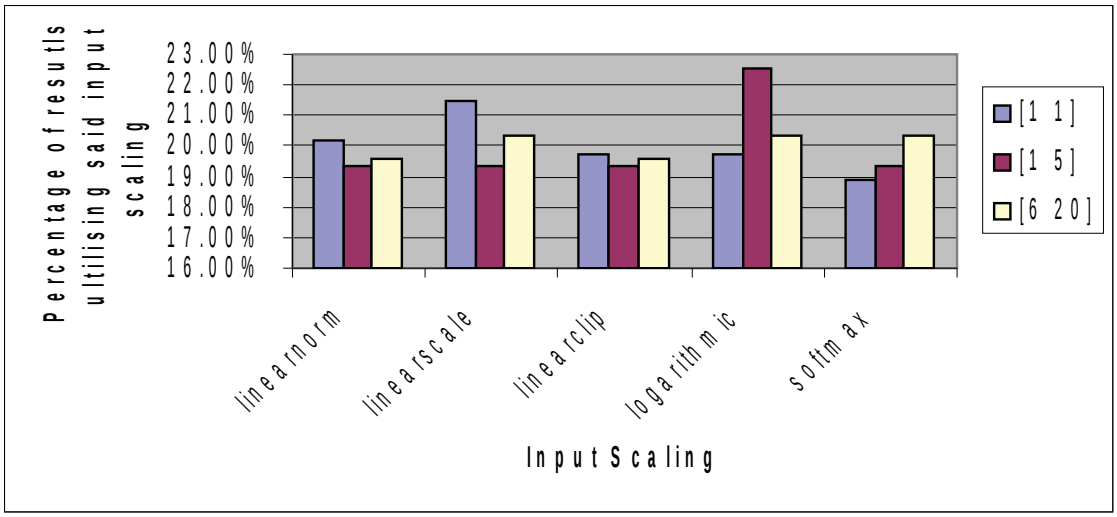

Figure 5.6.3: Distribution of optimal results for input period given various input scaling functions

Finally, breaking down the results into input period and network type (Figure 5.6.4) shows that linear input scaling is best used with 1-day input indicator networks, and logarithmic scaling is best used with 5-day price networks.

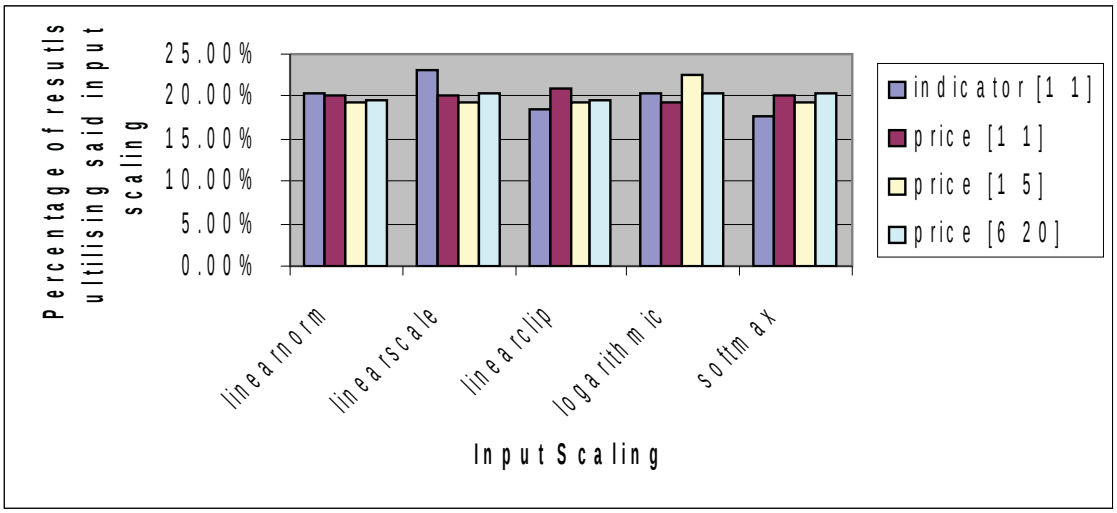

Figure 5.6.4: Distribution of optimal results for variations in network type and input periods given various input scaling functions

While there was not a strong bias for any particular type of input scaling, results do indicate that linear or logarithmic scaling are better used for indicator and 5-day price networks.

5.7 Training Step Size

The training step size, the number of steps between input sets to the neural network was kept constant at one step per iteration.

5.8 Target Scaling

Five types of target scaling were analysed versus network type and prediction periods.

- Binary Scaling - *binary*
- Linear Normalisation Scaling - *linearnorm*
- Linear Scaling - *linearscale*
- Linear Clipped Scaing - *linearclip*
- Logarithmic Scaling - *logarithmic*
- Softmax Scaling – *softmax*

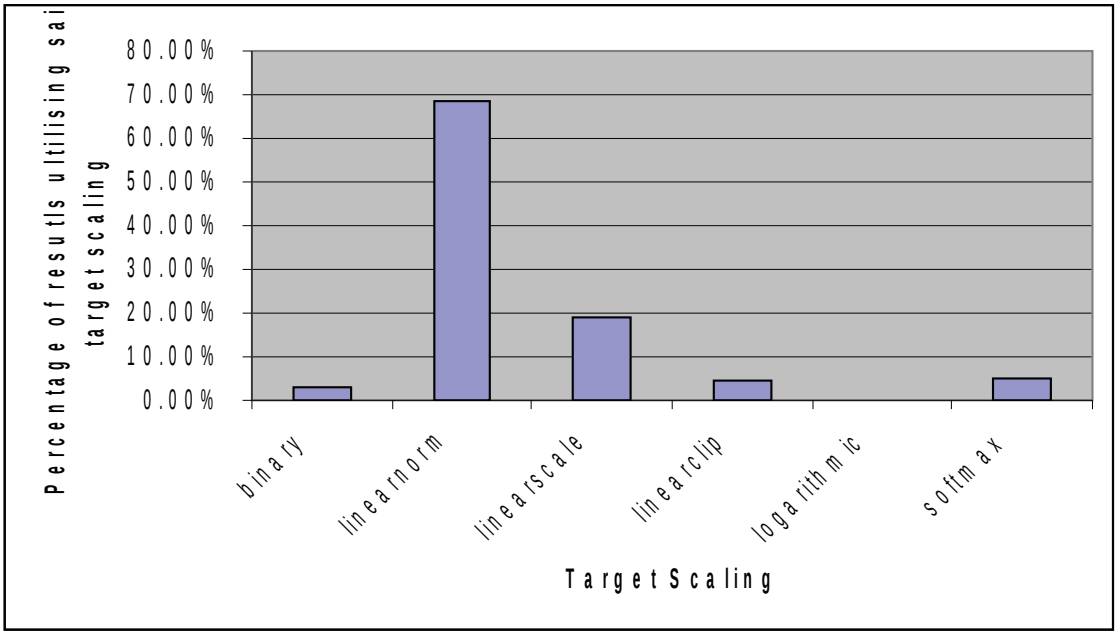

Figure 5.8.1: Distribution of optimal results given various types of target scaling

Summary results for target scaling (Figure 5.8.1) shows that linear normalisation was almost exclusively the best target scaling method.

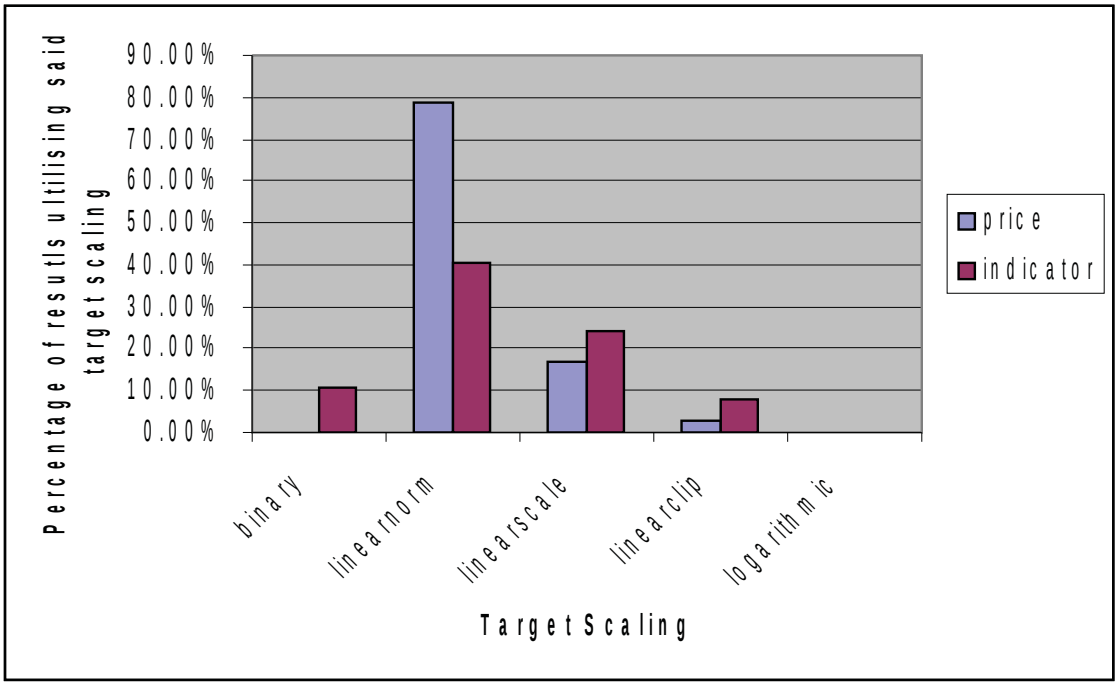

Figure 5.8.2: Distribution of optimal results for price and indicator networks given various types of target scaling

Price and Indicator performance analysis of the target scaling results shows that indicator optimal results were spread more evenly with linear scaling and linear clipping seen to be best in a significant number of cases (Figure 5.8.2).

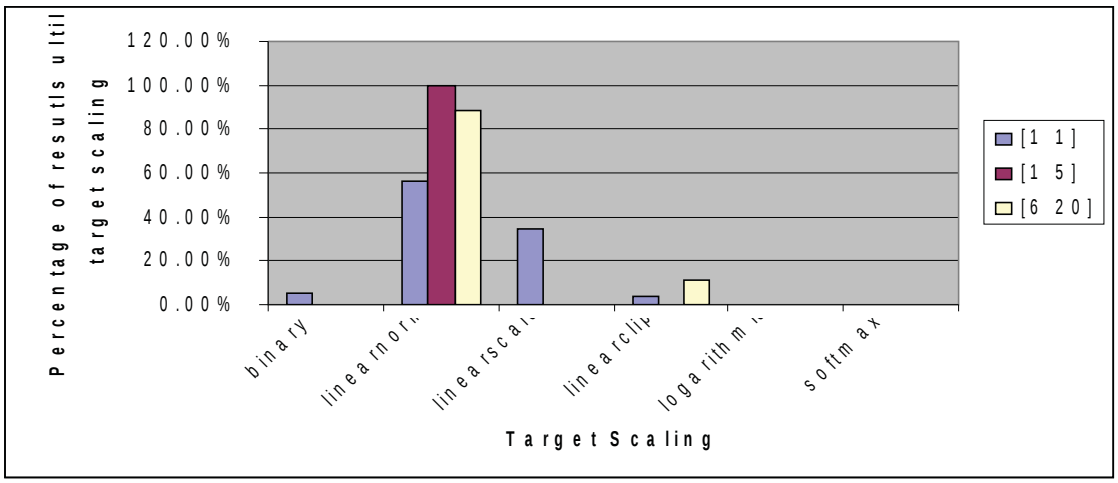

Figure 5.8.3: Distribution of optimal results of various network input periods given various types of target scaling

Looking at the breakdown based input period (Figure 5.8.3) shows that, in most cases, linear normalisation target scaling gave more optimal results. Linear scaling was consistently best used for 5-day input period networks, while linear scaled networks had slight support from both 1 and 15 day networks.

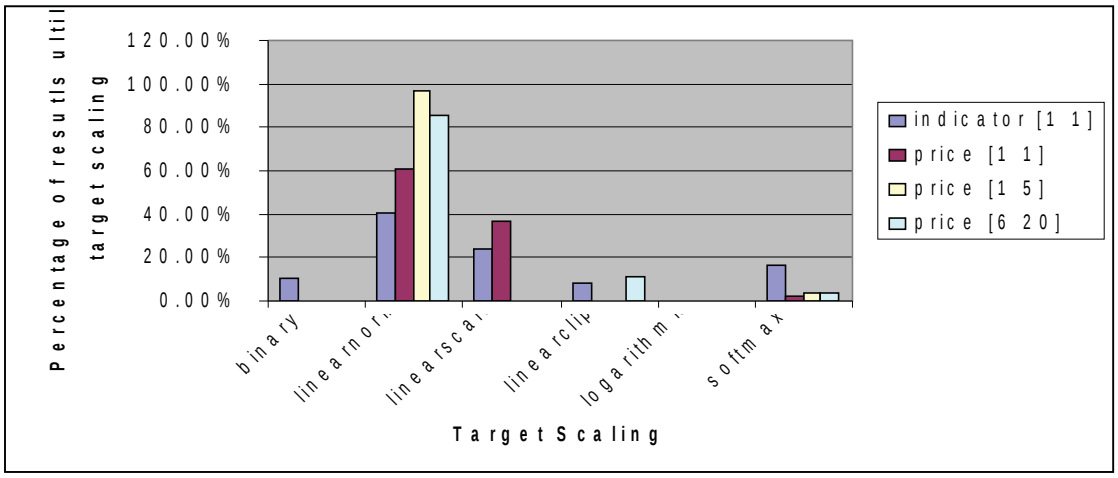

Figure 5.8.4: Distribution of optimal results for different network types and input period given various types of target scaling

Finally, a breakdown on input days and indicator/price type networks support the strong bias for linear normalisation target scaling. It should be concluded that linear normalisation is the best form of target scaling for all input periods (Figure 5.8.4).

5.9 Test Set Size

The test set size was set to 250 data pieces consistently.

5.10 Network Topology

Three network topologies were tested to test the effect on results:

20-10-5-1	20, 10 and 5 hidden nodes; 1 output
10-5-1-0	10 and 5 hidden nodes; 1 output node
5-1-0-0	5 hidden nodes; 1 output node

Results (Figure 5.10.1) showed a small bias for 5-1 networks over 10-5-1 and 5-1 networks.

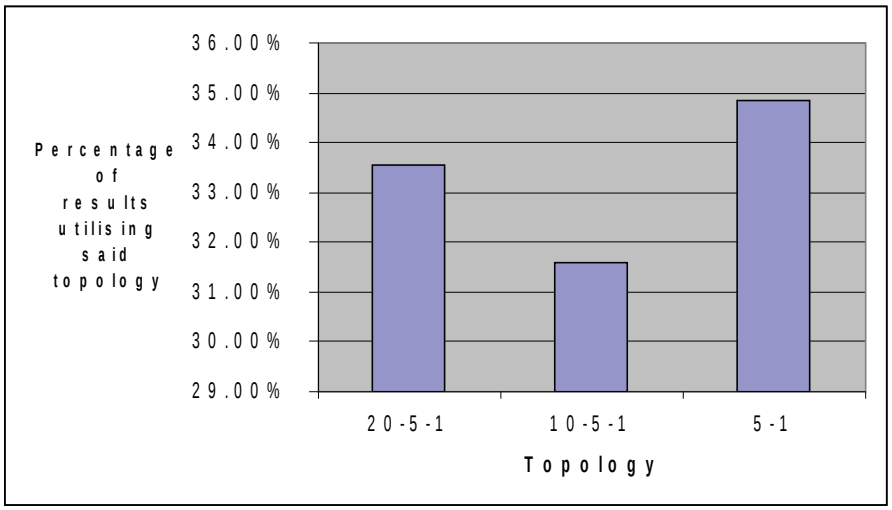

Figure 5.10.1: Distribution of optimal results given various types of network topology

Analysis by network type (Figure 5.10.2) shows that indicator networks are best used with either 20-10-5-1 or 5-1 networks. Price networks by comparison were better used with 5-1 networks.

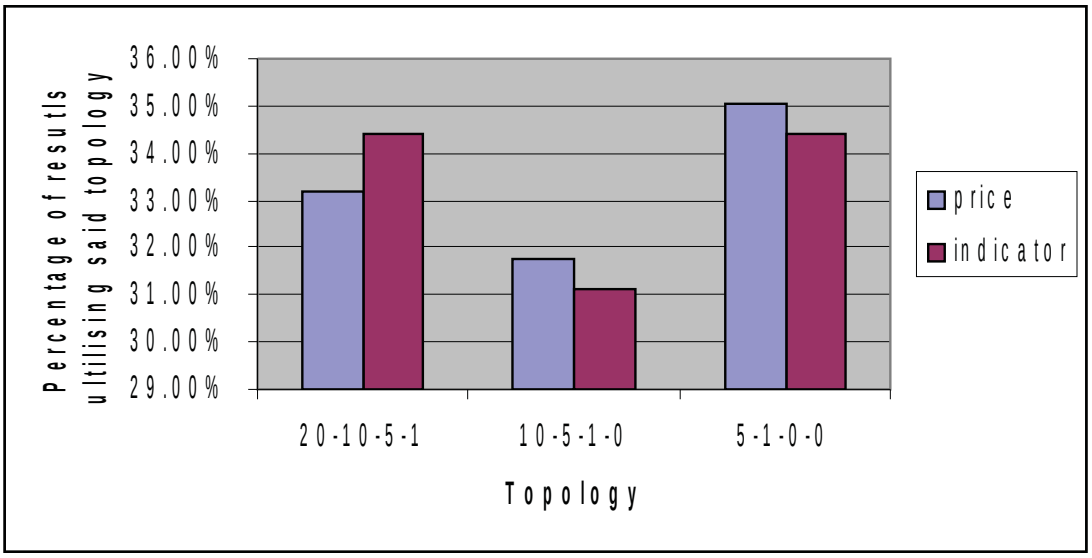

Figure 5.10.2: Price and Indicator performance given variation in topology

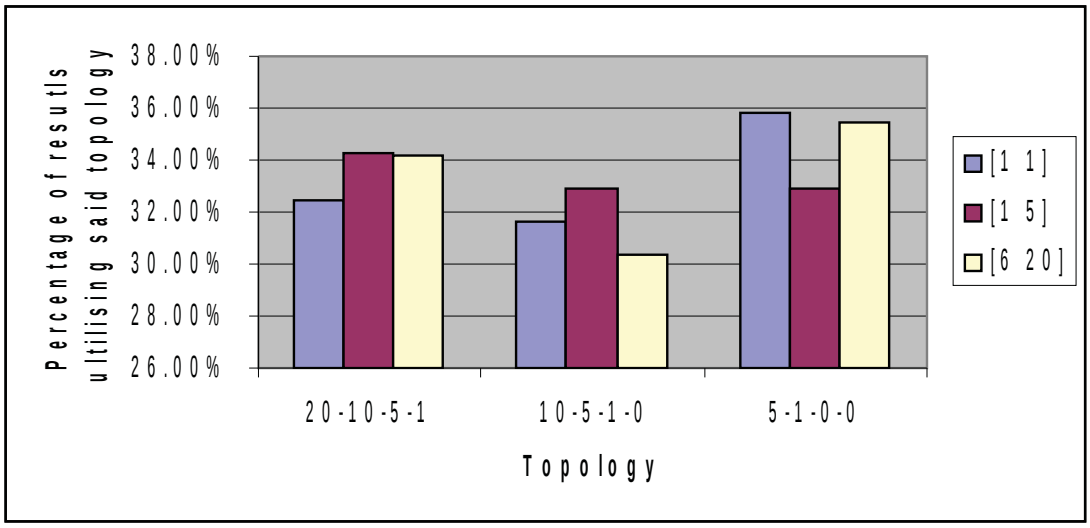

Figure 5.10.3: Distribution of optimal results given various input periods for different types of target scaling

Analysis by input period shows that both 1-day and 15-day networks were best used in with 5-1 network topology. (Figure 5.10.3).

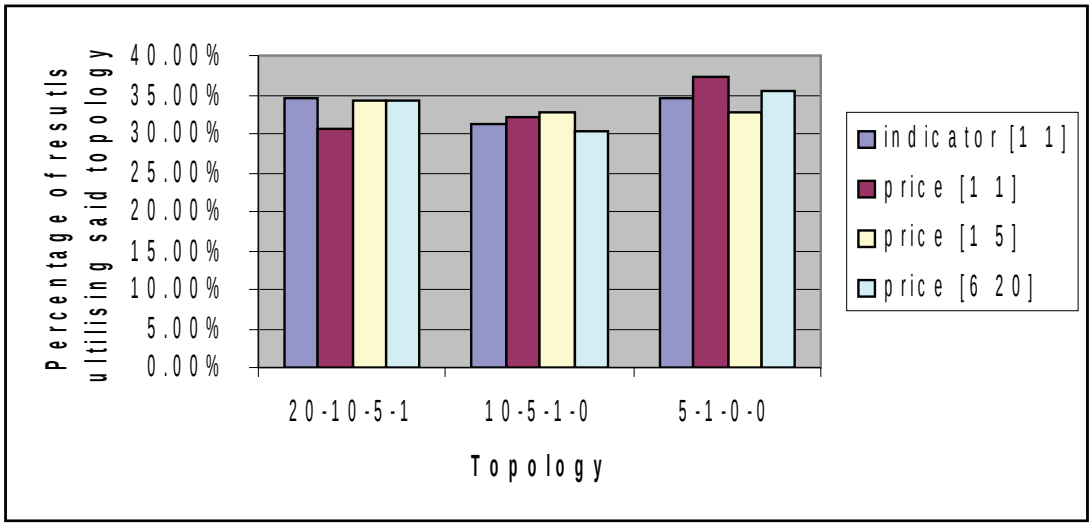

Figure 5.10.4: Distribution of optimal results for different network types and input periods given various types of target scaling

Break down by indicator and price mirror the bias for the use of less complex network topologies. Overall the results do not show that a single topology is best used in all cases, but rather that there is a slight trend towards the use of a less complex network for price networks (Figure 5.10.4).

5.11 Network Training Type

Three network training types were tested

- Standard back propagation - *traindx*
- Resilient Back Propogation - *trainrp*
- Levenberg-Marquardt Algorithm - *trainlm*

Standard Back Propagation – TRAINDX

This is a standard multilayer network training function, implementing the sigmoid "squashing" transfer function for hidden layers. The sigmoid function ensures that as the input gets large, the output of the sigmoid function approaches zero.

Resilient Back Propagation – TRAINRP

When using a steep descent to train the multilayer network, the gradient of the sigmoid function has only a very small magnitude, therefore, despite the far from optimal values of the train, causing only small weights on the network.

Under resilient back-propagation, only the sign of the derivative is used, while the magnitude is ignored, and a separate value is instead used. This separate update value is based upon a so-called "delta" value; this delta value is used to increase, maintain or decrease the update value based on whether the derivative signal is consistent, zero or opposite to that of the previous iteration respectively [ReBr93].

Levenberg-Marquardt Algorithm - TRAINLM

The Levenberg-Marquardt algorithm was designed to increase the speed of network training, while reducing the computational necessity to calculate the Hessian matrix (Hagan et. al, 1996). The Hessian matrix is approximated by the sum of squares (Equation 13)

$$H = J^T . J \qquad (13)$$

Whereby 'J' is the Jacquite matrix, containing the first network derivativeerrors with respect to the weights. The gradient is subsequently calculated as per Equation 14.

$$graident = error \times J^T \quad (14)$$

Further details of the Leven-Marquardt Algorithm can be found in Hagan et. al (1996).

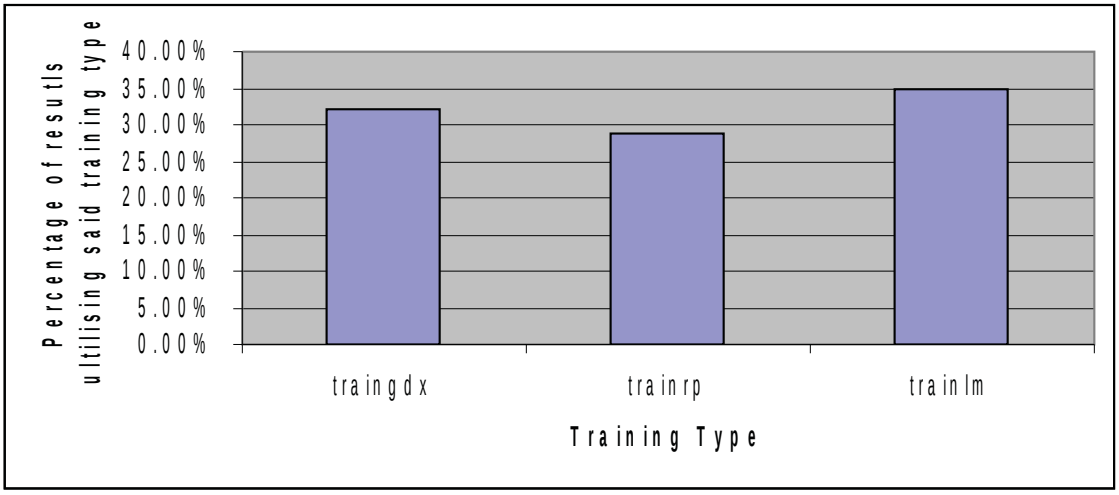

Figure 5.11.1: Distribution of optimal results given various network training functions

Training summary data shows that *trainlm* was the best training type, being optimal in 35% of results (Figure 5.11.1).

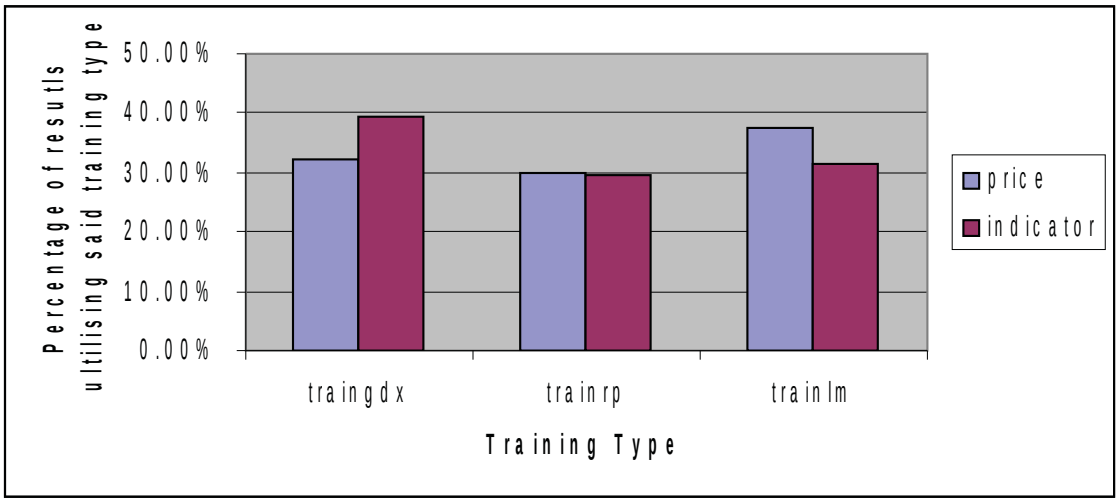

Figure 5.11.2: Distribution of optimal results for different network types given various network training functions

Analysis by price and indicator show that *traingdx* was more optimal for indicator networks at almost 40% of optimal results using it, while *trainlm* was better for price

networks (Figure 5.11.2). *trainrp* was almost exactly the same for both price and indicator networks with at 30% of results for both types of networks.

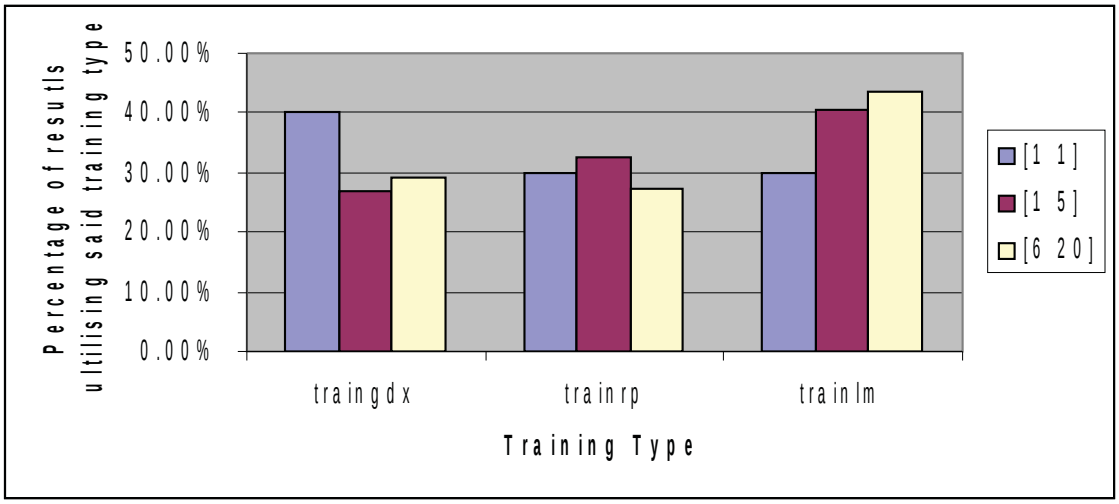

Figure 5.11.3: Distribution of optimal results for different training periods given various training functions

Analysis by input period shows that as the training period increased, so did the bias towards the use of *trainlm* (Figure 5.11.3). Conversely *traingdx* shows a bias towards a lesser training period, with trainrp again showing little difference varition based on training type.

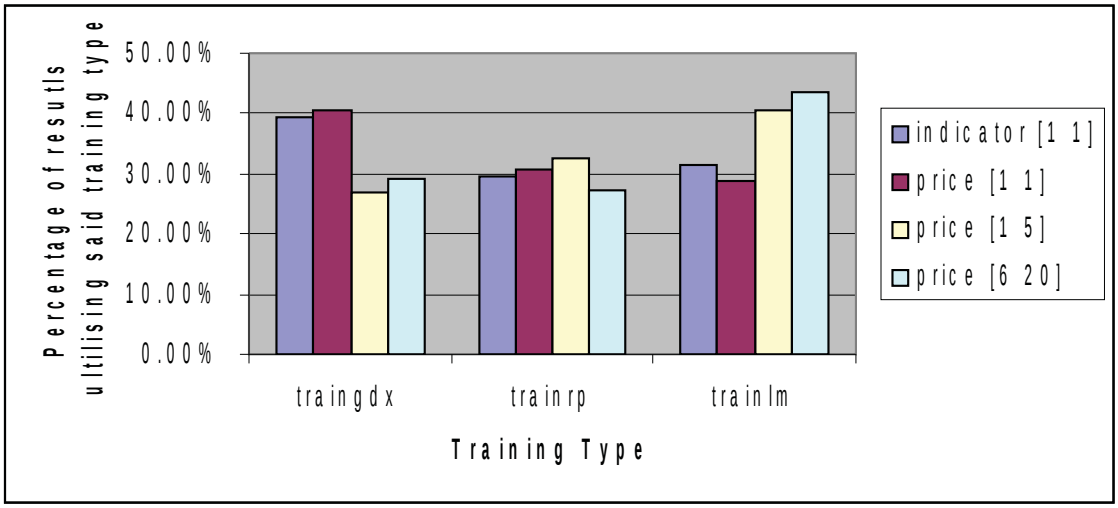

Figure 5.11.4: Distribution of optimal results given various network training functions for various training functions

Analysis by indicator/price period reinforce the summary data (Figure 5.11.4). It can be

further seen that there is a strong indication that indicator and price 1-day networks to favour *trainingdx* for optimal results.

5.12 Network Type

Two types of network types were tested, Elman networks and Feed forward networks.

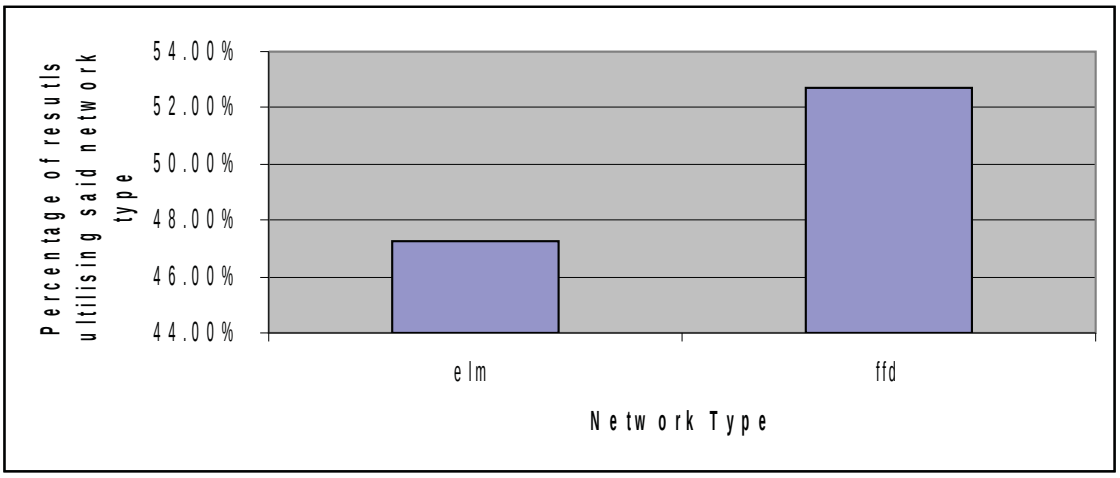

Figure 5.12.1: Distribution of optimal results given different network types (Elman and Feed Forward)

Analysis by network type show that feed-forward networks share the majority of optimal results (Figure 5.12.1).

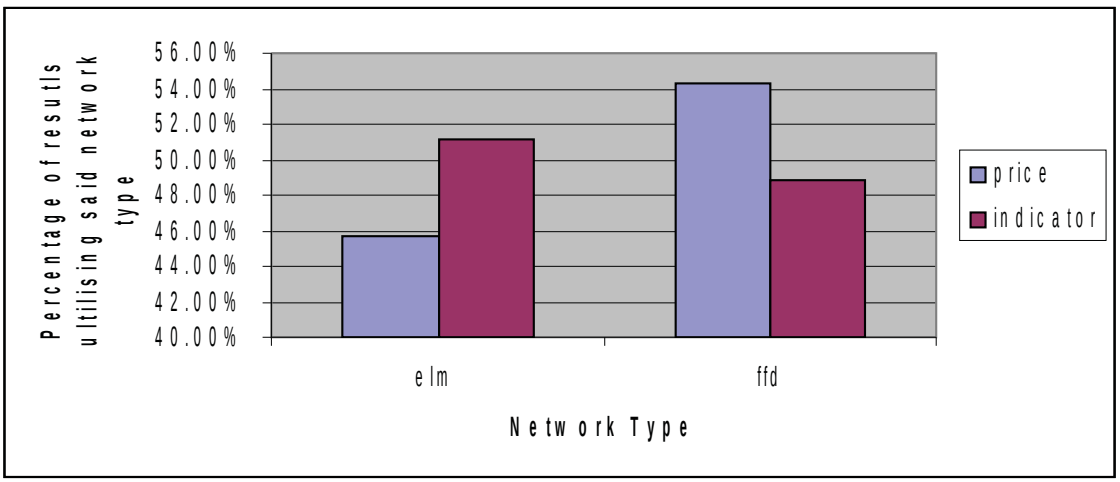

Figure 5.12.2: Distribution of optimal results for price and indicator networks given various network types

Breaking down into price and indicator networks show that price networks favoured feed-forward network types (Figure 5.12.2). This trend was the opposite for indicator networks which favoured Elman networks. This may be because indicator networks were restricted to 1-day input.

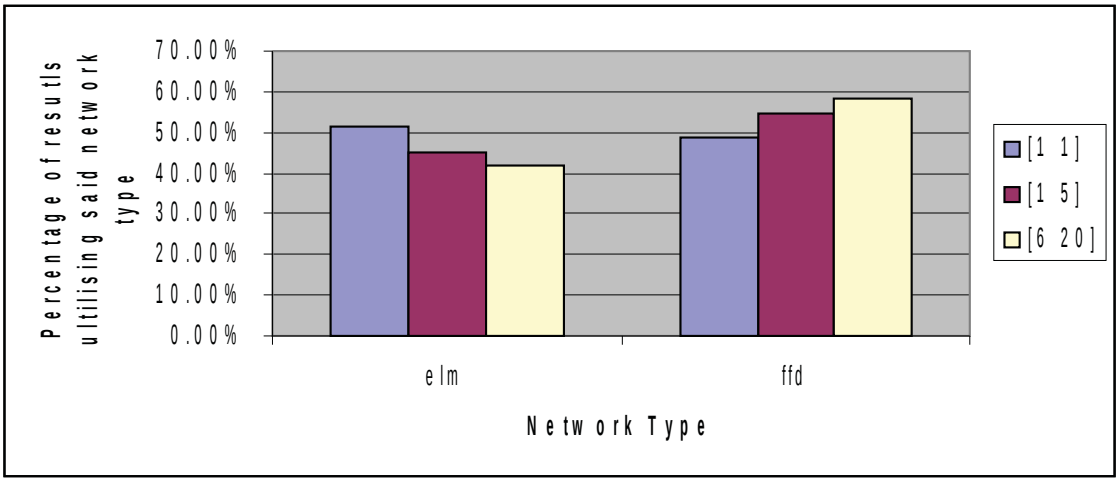

Figure 5.12.3: Distribution of optimal results for different training windows given various network types

Results broken down into input period shows that longer input periods favoured the feed-foward network (Figure 5.12.3).

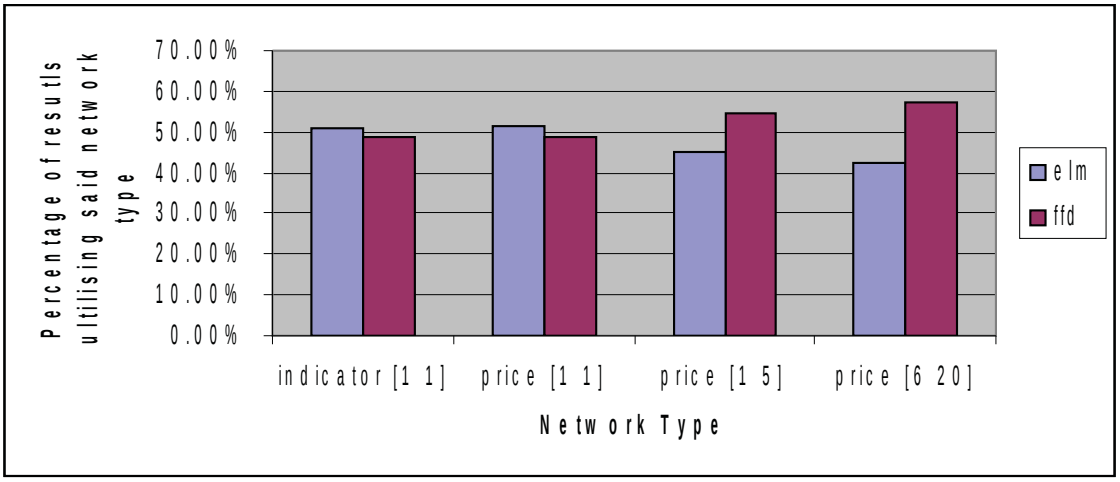

Figure 5.12.4: Distribution of optimal results for different network training windows and types given various network types

Results shown in Figure 5.12.4 highlight the effectiveness of feed-forward networks was better with longer input periods, while Elman networks gave a better result with 1

day input periods.

5.13 Validation Set Size

The validation set makes use of an improvement method known as early-stopping. Early stopping is based upon the creation of a 'validation set'. Unlike normal training, where the network is trained for a set number of iterations, or until the error of the training set increases, early stopping uses a separate validation to determine the point of early stopping. With a validation set, training is stopped once the error of the validation set increases for several consecutive iterations.

The use of a validation set prevents over fitting of the network to a specific data set, the situation whereby the network is not able to generalise for a wide range of data. It should be noted that a significant difference in the minimum point of the test and validation sets typically implies a poor division of the data set.

During simulation, a validation set equal to approximately one year (250 days) of training was used. A validation set of zero shows that the test set error was instead used. Results by validation set size shows that no validation set is better in more than half of cases (Figure 5.13.1).

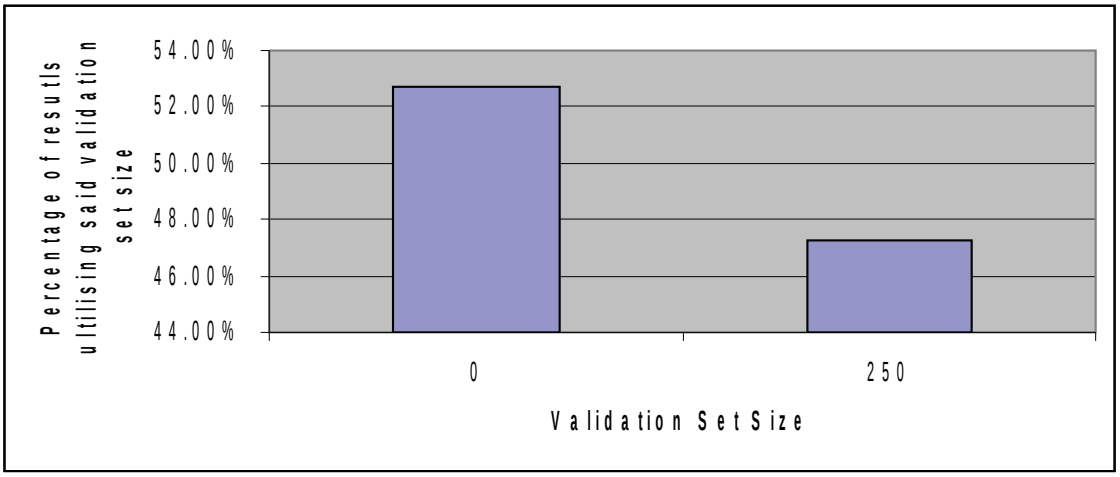

Figure 5.13.1: Distribution of optimal results with and without validation sets

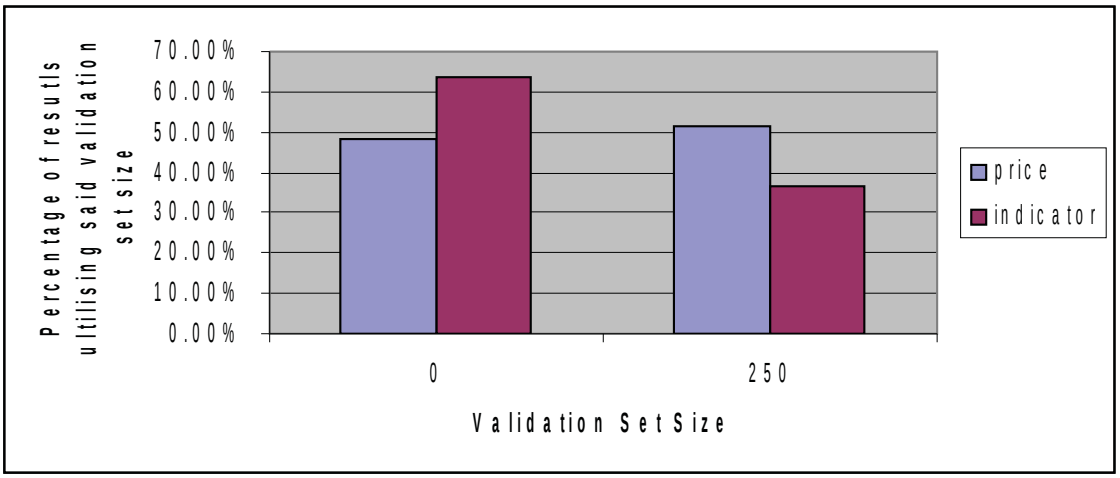

Figure 5.13.2: Distribution of optimal results for price and indicator networks with and without validation sets used for training

Analysis into price and indicator networks shows that while indicator networks strongly favour validation, price networks results were slightly biased towards the use of a validation set (Figure 5.13.2).

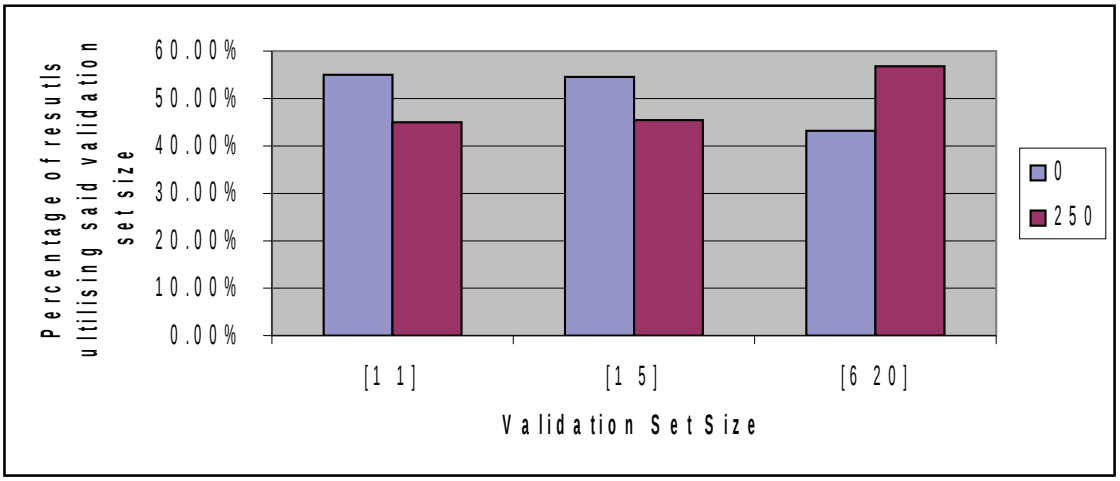

Figure 5.13.3: Distribution of optimal results for different training windows with and without validation sets used for training

Further breaking down the results into prediction period shows that as the prediction period increases, so does the bias for using a validation set (Figure 5.13.3).

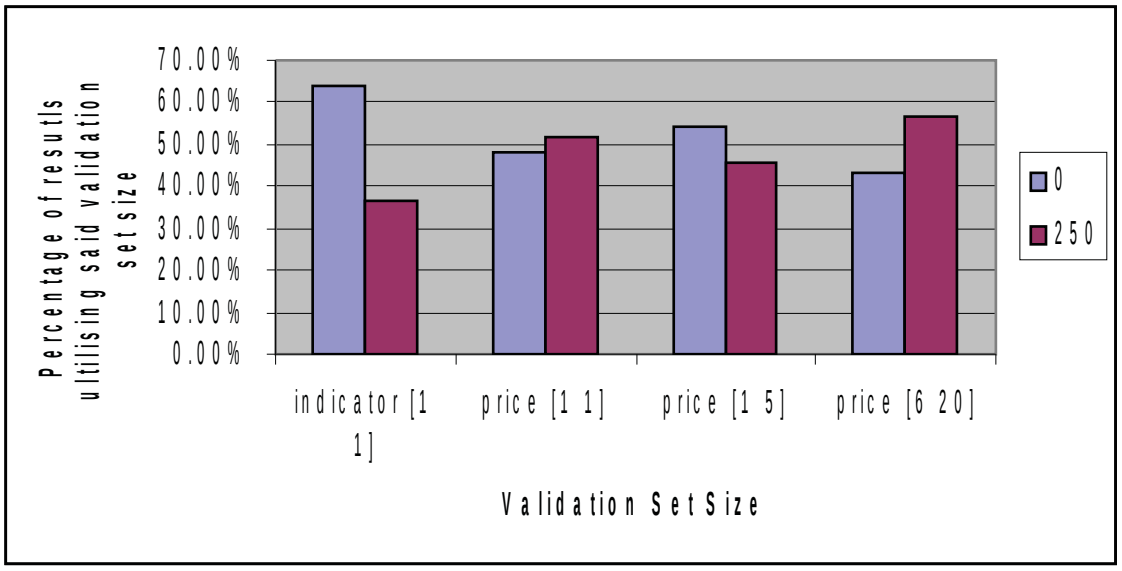

Figure 5.13.4: Distribution of optimal results for different network types and training windows, with and without validation sets used for training

Breaking down the results into indicator and price prediction periods shows the strong bias of indicator networks for non-validation, and the growing bias towards having a validation set for price networks with a longer prediction period (Figure 5.13.4)

5.14 Summary

The results by from this chapter are summarised in Table 5.14.1. Not all parameters yielded strong results, however where results were inconclusive these have been marked accordingly.

Table 5.14.1: Summary or results

	1-1 Indicator	1-1 Price	1-5 Price	6-20 Price
Training Days	2500	250	500	
Training Epochs	*Not varied: 250*			
Network Function	tansig	logsig		
Training Goal	5.00%	*Inconclusive*		
Input training period	10 day			
Input Scaling	linearscale	*Inconclusive*	logirithmic	N.P.
Training Step Size	*Not varied: One step per iteration*			
Target Scaling	Linear Normalisation			
Test set size	*Not varied: 250*			
Network Topology	*Inconclusive*			
Network Training Type	traindx		trainlm	
Network Type	*Inconclusive*		Feed Forward	
Validation Set Size	0	250		

Chapter 6 utilises the data from this chapter for the creation of a combined neural network. By training the combined network with output data gathered in Chapter 6 against the direction that the stock, price predictive networks which gave poor results will be weighted accordingly during the training process.

CHAPTER 6 - COMBINED NETWORK

The final set of training done was for the sake of simplicity were called the 'combined network'. Combined networks utilise previous Price and Indicator Predictive Network results in serving as inputs to a secondary neural network. The hypothesis of this application was that a secondary neural network layer would be better able to determine which neural networks were of use and which gave invalid results. For instance, a neural network which gave consistently poor results should be ignored by the secondary neural network, in favour of a more reliable indicator which gave consistently high results.

The combined network made use of 34 inputs each feeding into an separate inputs layer comprised of a pre-processing, price predictive neural network and scaling components. A simplified diagram of the combined network is shown in Figure 6.1, showing just 3 input layers.

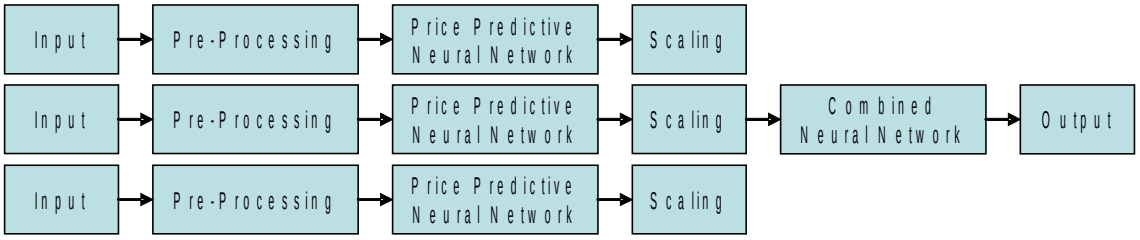

Figure 6.1: Simplified Topology of Combined Network

The 34 networks utilise the networks trained in the previous for each of the 34 Buy / Sell / Hold indicators detailed in Table 4.1.1.

Results from Chapter 5 were used to choose the neural networks which gave the best results. For Example in Chapter 5.12 Elman and Feed Forward Networks were found to be better for price and indicator networks respectively, thus price predictive neural network modules made use of Elman networks for Price Predictive networks and Feed Forward Networks for indicator networks.

6.1 Network Topology

During the training phase of the price and indicator networks the testing results of the network were saved to individual Matlab data files. This allowed time to be saved in retesting previously created networks. These results were converted to higher/lower signals by hard limiting so as to provide purely binary data to the network. To do this all, values above 0.5 were taken as a 'higher' indication and all values below as a 'lower' (Table 6.1.1).

Table 6.1.1: Labelling of data to 'higher' or 'lower' based on 0.5 reference

Data	*inter-day Difference*	*MACD*	*RSI*	*Moving Average*
1				
1.14	Higher	Higher	Higher	Higher
1.25	Higher	Higher	Lower	Higher
1.36	Higher	Lower	Higher	Higher
1.27	Lower	Lower	Lower	Lower
1.28	Higher	Higher	Higher	Lower
1.24	Lower	Lower	Higher	Lower
1.20	Lower	Lower	Lower	Lower
1.01	Lower	Lower	Higher	Lower
1.04	Higher	Lower	Higher	Higher
	Accuracy	70%	50%	90%

Higher and lower data points were then relabelled to 1 and 0 respectively (Table 6.1.2) and fed into the neural network.

Table 6.1.2: Labelling of 'higher' and 'lower' points identified in table 13 are assigned binary values of 1 or 0 respectively

Data	*inter-day Difference*	*MACD*	*RSI*	*Moving Average*
1				
1.14	1	1	1	1
1.25	1	1	0	1
1.36	1	0	1	1
1.27	0	0	0	0
1.28	1	1	1	0
1.24	0	0	1	0
1.20	0	0	0	0
1.01	0	0	1	0
1.04	1	0	1	1
	Accuracy	70%	50%	90%

This is represented in a matrix as per Equation 15.

$$CombinedNetworkInputMatrix = \begin{bmatrix} 1,1,1,1 \\ 1,1,0,1 \\ 1,0,1,1 \\ 0,0,0,0 \\ 1,1,1,0 \\ 0,0,1,0 \\ 0,0,0,0 \\ 0,0,1,0 \\ 1,0,1,1 \end{bmatrix} \quad (15)$$

Hard limiting is then applied to give reasonable set of maxima and minima.

$$CombinedNetworkInputMatrix = \begin{bmatrix} 0.8,0.8,0.8,0.8 \\ 0.8,0.8,0.2,0.8 \\ 0.8,0.2,0.8,0.8 \\ 0.2,0.2,0.2,0.2 \\ 0.8,0.8,0.8,0.2 \\ 0.2,0.2,0.8,0.2 \\ 0.2,0.2,0.2,0.2 \\ 0.2,0.2,0.8,0.2 \\ 0.8,0.2,0.8,0.8 \end{bmatrix} \quad (16)$$

This was then fed into the network which, as with previous implementations, had various parameters modified; these parameters were the network size and type. As with the Indicator and Price Prediction Networks, the network size was varied between 34-

10-5-1, 34-10-1, 34-1 and 34-5-1. Additionally, the Neural Network Type was varied between Elman and Feed-Forward networks for each size. The days of training were not modified for this testing due to the use of previous simulation outputs.

The hypothesis behind this training was that the neural network should learn that a certain combination of indicators should indicate a certain direction for the network.

These tests were performed hard limiting of 0.8 to 0.2 for buy and sell respectively, and without any hard limiting. Prior to simulation, it was hypothesized that the network, without hard limiting, may perform better as the neural network may be better able to respond to the real result of those individual inputs, using it as a degree of “certainty” in the inputs indication.

6.2 Results

The results of the training are given in Table 6.2.1 and 6.2.2 for variations in the Network Topology and Type respectively. Median results for using different network types ranged between 47.65% and 57.28%, with the higher results favouring wider and deeper networks.

Table 6.2.1: Combined Network Results for Network Size Variation (hard limits 0.8 & 0.2)

	20-10-5-1	*20-10-1*	*20-1*	*10-5-1*	*10-1*	*5-1*	*2-1*
max	69.05%	69.05%	69.05%	69.05%	69.05%	69.05%	69.05%
median	57.09%	57.28%	51.97%	55.70%	53.44%	52.57%	47.65%
min	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

Comparatively analysis based on network type variation show a somewhat better median result for Elman networks (Table 6.2.2).

Table 6.2.2: Combined Network Results for Network Type Variation (hard limits 0.8 & 0.2)

	Elman	*Feed Forward*
max	69.04%	69.05%
median	56.25%	54.95%
min	0	0

The implementation without hard limiting yielded the poorest results ranging between approximately 37% and 45% prediction accuracy. Smaller network topologies again had a negative effect on results (Table 6.2.3).

Table 6.2.3: Combined Network Results for Network Size Variation with no hard limits

	20-10-5-1	*20-10-1*	*20-1*	*10-5-1*	*10-1*	*5-1*	*2-1*
max	54.76%	54.76%	54.76%	54.76%	54.76%	54.76%	52.38%
median	38.95%	45.36%	36.79%	37.66%	40.87%	44.58%	41.54%
min	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

Finally Elman and Feed Forward networks without hard limiting similarly showed poor results of between 40% and 42% respectively (Table 6.2.4).

Table 6.2.4: Combined Network Results for Network Type Variation with no hard limits

	Elman	*Feed Forward*
max	54.76%	54.76%
median	41.91%	39.88%
min	0.00%	0.00%

6.3 Analysis

These results show that while changes in network topology caused a significant effect on the results, changes in network type were less severe. Additionally, the removal of hard limiting created networks which had a less than 50% probability of a correct market prediction; these unlimited results were from the previous neural network and were already bounded between 1 and 0 and thus it was expected that the neural network would be able to extract more information from the non-hard limited results.

Table 6.3.1: Summary of Average Network Performance for Combined Network

	20-10-5-1	*20-10-1*	*20-1*	*10-5-1*	*10-1*	*5-1*	*2-1*	*Elman*	*Feed Forward*
Hard limiting	57.09%	57.28%	51.97%	55.70%	53.44%	52.57%	47.65%	56.25%	54.95%
No Limiting	38.95%	45.36%	36.79%	37.66%	40.87%	44.58%	41.54%	41.91%	39.88%

As hard limiting had a more pronounced effect on the results than parameter modification, they were further scrutinised. A comparative graph of the hard limiting can be seen in Figure 6.3.1. This clearly shows the similarly in the effect that hard limiting had on all parameter combinations.

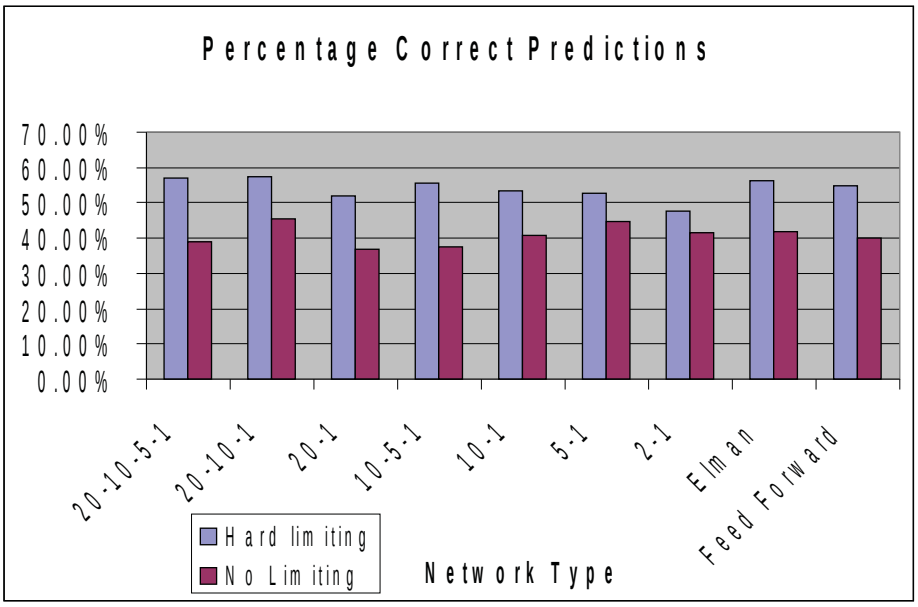

Figure 6.3.1: Average Network Performance for Combined Network

6.4 Summary

The combined network results gave an accuracy of less than 60%. A result of 50% is encouraging and expansion of this work is discussed in Chapter 7.

CHAPTER 7 - Conclusion and Future Work

Preliminary work showed that pre-processing should include relative-normalisation Furthermore these preliminary results also showed that use of Self Organising Maps on time series equity data yielded no significant improvement in performance. This can be attributed to information loss as the result of an overly restrictive SOM size; a broader and more in-depth look at SOM size and input data set size may improve this result.

For this study training epochs and test set size were kept constant at 250, while the training step size of one was used for all training.

Training days of 2500, 1250, 500 and 250 were tested; results showed that 2500 days were best used with indicator networks. A much smaller number of training days were better used for price networks, namely 250 days for next-day prediction and 500 days for 5 and 15 day prediction. Comparative studies show as few as 40 observations in the case of Koulouriotis et al. (2005) and as high as 2000 observations in the case of Thawornwong and Enke (2004) .

The tangent sigmoid and logarithmic sigmoid network transfer functions were used for training. Results showed that the tansig network function was found to be best for indicator networks, while logsig was best used for all price networks.

Training goals used in simulation were 0.1%, 1%, and 5% error rates. It was found that a 5% error rate was best used in indicator networks, which suggests that indicator networks are more easily over trained. Comparatively price networks showed generally similar results for all training goals.

The Input period is the size of the data block that was fed into the neural network. Input periods of one day, one trading week (5 days) and two weeks (10 days) were tested. Results showed that 10 days found to be better for all network types.

Five different types of input scaling were used in simulation, namely linear

normalisation, linear scaled, linear clipped, logarithmic and softmax. Results showed that linear scaling and logarithmic scaling were slightly better for indicator and 5-day price networks respectively. Results were not inconclusive for price networks of 1 and 15 days.

Six different types of target scaling were used, namely binary, linear normalisation, linear, linear clipped, logarithmic and softmax scaling. For both indicator and price networks linear normalisation was on average the best used.

Network topologies of 20-10-5-1, 10-5-1 and 5-1 were tested, however it was not found that any topology gave an overall better result. This may be because of the very wide range of data tested. In comparison other works with neural networks have shown great variation in the size of neural networks, from as high as 60-60-1 in the case of Kuo (1998) to as low as 2-1 in the case of Siekmann et al., (1999) .

Standard back propagation, resilient back propagation and Levenberg-Marquardt were the network training types used in simulation. Results shows that standard back prorogation was best used for next day prediction price and indicator networks, while Levenberg-Marquardt was best used for 5 and 15 day price prediction.

Both feed-forward and Elman network types were compared. Results showed that Feed-forward networks gave slightly better results with longer input periods, while Elman networks were better used for shorter input periods. Feed-forward networks are commonly used in literature (Ajith et al., 2003; Casas, 2001; Refenes et al., 1993) as too are recursive networks (Wikowska, D., 1995; Chaturvedi and Chandra, 2004; Zhang et al., 2004) .

Simulations were carried out both with and without a validation set; where a validation set was used it was kept at a size of 250. It was found that indicator networks were best served by using no validation set, while all price networks (1, 5 and 15 day) produced the best results with a validation set size of 250. Validation sets used by other works vary between none being used (Brownstone, 1996) or as low as 35 (Andreou et al.,

2000) and as high as 6000 (Pantazopoulos et al., 1998).

Analysis of overall results shows that the tansig network function, a 5% training goal and no validation set was best for indicator networks. Comparatively none of the traiining goals given for price networks gave a better result, however the logsig network function and a 250 day validation set size gave best results.

The training type was best used was trainingdx for next day prediction (both price and indicator networks) and trainlm for 5 and 15 day predictions.

Combined network results were such that correct predictions were able to be made with a median accuracy of between 50% and 60%, while the maximum accuracy obtained was 69%. While this level of accuracy may not seem significant, however any result higher than 50% can be seen as better than random chance.

The combined results fall short of that seen in some other works. A summary of results by other works is shown in Table 7.1. In it's current form a similarly trained network would not be reliable enough on it's own for an automated market prediction system but could be used as an aid in a trading environment. It is recommended that future work look at a more robust combined network topology to better extract an improved overall result.

Table 7.1: Comparison of results with other studies

Study	Result (testing data)
Yoon, Swales & Margavio (1993)	77.0%
Dutta & Shekhar (1988)	64.7%
Nikoo et. al (2007)	83.1%

Predicting the market 60% of the time or more does not necessarily guarantee success in market trading. For example, knowing the market is going to fall doesn't guarantee that the equity in question can be sold at an acceptable price if market volumes are too low. Similarly a network may not be able to predict drastic market rises or market crashes unless it is specifically trained to do so as is the case in Gomes et. al (2011), further

highlighting the importance of using more than a single neural network or technical indicator in making trading decisions.

Further tuning of such a network should minimise the output variation by using an increased number of hidden layers. Best results were yielded with prediction calculated relative to a wider trading period for the following trading day only. This implies the difficulty in predicting multiple days into the future with such a model; however, next day price prediction, a more useful measure in real world applications, also yielded highly favourable results. When compared to a simple mechanical averaging form of trend prediction, the neural network trained achieved, on average, a higher success rate. Simple mechanical average trend prediction was only able to predict with 65% accuracy versus the optimally trained networks with predictive ability exceeding 90%.

It is the recommendation of the author that further research network simulations be undertaken in a multi-threaded or parallel computing or environment. The use of Matlab seriously hampered the amount of simulation that was able to be achieved. Limitations of Matlab include the interpreted nature of Matlab code, the Java-based Matlab environment, the poor ability of the environment to deal with large (100's of megabytes) datasets and overall stability of the environment. These limitations forced some tasks to be broken into smaller sub-tasks and the functions themselves to be equipped with the ability to continuing processing after a crash.

Processing power limitations can not be simply resolved by using a more powerful computer as in reality several orders of magnitude of computational power is necessary. The use of a multi-threaded or parallel computing environment with functions implemented in a compiled and well tested language such as Fortran would be preferable. It would, in fact, be of benefit given the large library of Fortran code that is available due to the sheer age and wide use of the language.

A further extension of this research is to retrain the network on regular basis, perhaps even daily, to take into account new features in the stock data. For this to be a commercially viable option, it would require the central processing and training of the

neural networks which would then be sent as modules to individual trading client applications.

Another extension of this research would be to make use of heterogeneous data sources, that is, sources such as the market indexes, industry indexes, exchange rates and foreign market indexes. Further sources of easily quantifiable heterogeneous data are very industry specific but include data such as average rainfall, consumer power usage or even the birthrate 20 years previous.

This thesis has just scratched the surface of what is possible with neural networks in the stock prediction and time series as a whole. Extensions to this work are necessary to better understand the use of neural networks in similar environments.

REFERENCES

Abdelmouez, G., Hashem, S. R., Atiya A. F., & El-Gamal, M.A. (August, 2007). *Neural Network vs. Linear Models for Stock Market Sectors Forecasting.* IJCNN'07, Florida, USA, pp. 1365-1369.

Ajith, A., Baikunth, N. & Mahanti, P. K. (2003a). *Hybrid intelligent systems for stock market analysis.* Proceedings of International Conference on Computational Science.

Andreou, A. S., Neocleous, C. C., Schizas, C. N., & Toumpouris, C. (2000). *Testing the predictability of the Cyprus Stock Exchange: The case of an emerging market.* Proceedings of the International Joint Conference on Neural Networks, 360–365.

Appel, G. (2006, May). *Moving Average Convergence Divergence – MACD.* Stockvale.

Atsalakis, G. S. and Valavanis, K.P. (2009). *Surveying stock market forecasting techniques – Part II: Soft computing methods* . Expert Systems with Applications . 36 (2009) 5932–5941.

Baek J., & Cho S. (2002). *An Up-Trend Detection Using an Auto-Associative Neural Network : KOSPI 200 Future*. San 56-1, Shillim-Dong, Kwanak-Gu, 151-744, Seoul, Korea.

Bellman, R. (1961). *Adaptive Control Processes: A Guided Tour*, Princeton University Press.

Brown, S., Goetzmann, W., & Kumar, A. (1998). *The Dow Theory: William Peter Hamilton's Track Record Re-Considered.* New York University, Leonard N. Stern School Finance Department Working Paper Seires 98-013, New York University, Leonard N. Stern School of Business.

Brownstone, D. (1996). *Using percentage accuracy to measure neural network predictions in stock market movements.* Neurocomputing, 10, 237–250.

Burgess, N. (1995, 12 January). *Neuroforecasting.* Talk to Neural Computing Applications Forum.

Casas, C. A. (2001). *Tactical asset allocation: An artificial neural network based model.* Proceedings of the International Joint Conference on Neural Networks, pp 1811–1816.

Chaikin. (1994). *Chaikin Indicators.* Stocks and Commodities. Volume 12:1. p30-37. Technical Analysis Inc.

Chang, P., Liu, C., Lin, J., Fan C., & Ng, C.S.P. (2009). *A neural network with a case based dynamic window for stock trading prediction.* Expert Systems with Applications. p36, (6889–6898).

Chaturvedi, A., & Chandra, S. (2004). *A neural stock price predictor using quantitative data.* Proceedings of the 6th International Conference on Information Integration and Web-Based Applications Services, pp 27–29.

Chen, H., Huang, C., and Chen, A. (2008). *Application of Self-Organizing Mapping Neural Network for Discovery of Market Behavior of Equity Fund* . Institute of Information Management , Chiao Tung University , Taiwan. Retrieved from http://www.wseas.us/e-library/transactions/information/2010/89-257.pdf

Dalton, J.M. (1996). *How the Stock Market Works, 2nd edition.* New York Institute of Finance. p176.

Dempster, M. A. H. & Jones, C. M. (2000, Dec). *The profitability of intra-day FX trading using technical indicators.* Centre Financial Res., Judge Inst. Management, University Cambridge,

Dempster, M. A. H. & Jones, C. M. (2001b). *A real-time adaptive trading system using genetic programming.* Quant. Finance. Vol 1.

Dutta, S. and Shekhar, S. (1988). *Bond rating: A non-conservative application of neural net-works.* Proceedings of the IEEE International Conference on Neural Networks, San Diego, California, pp. 443–450.

Elman, J. L., (1990). *Finding Structure in Time.* Cognitive Science, 14, 179-211.

Fama, E. F. (1965, January). *The Behavior of Stock-Market Prices.* Journal of Business.

Frank, R. J., Davey, N., & Hunt, S. P. (2001). *Time Series Prediction and Neural Networks.* Journal of Intelligent and Robotic Systems, 31:91-103.

Using P/E Ratio Standard Deviation Bands to Improve Performance. (2002, 31 December). Ford Equity Research.

Bearish engulfing candlestick. (2010). FX Words. Retrieved from http://www.fxwords.com/b/bearish-engulfing-candlestick.html

Gerald, A. (1999). *Technical Analysis Power Tools for Active Investors.* Financial Times. Prentice Hall. p166.

Giles, C., Lawrence, S., & Tsoi, A.C. (1997). *Rule inference for financial prediction using recurrent neural networks*. Proceedings of IEEE/IAFE conference on Computational Intelligence for Financial Engineering (CIFE). p253–259.

Giles, C., Lawrence, S., & Tsio, A. (2001, July/August). *Noisy Time Series Prediction using a Recurrent Neural Network and Grammatical Inference*. Machine Learning. Volume 44, Number 1/2. p.7,161-183.

Gomes, C., Raposo, A., Diogo, M. & Marques, N.C. (2011). *A stock market crash alarm system based on a Hurst Index Neural Network* . ENTRIA/Departamento de Informˊtica, a Faculdade de Ciˆncias e Tecnologia, Universidade Nova de Lisboa e Quinta da Torre, 2829-516 Caparica, Portugal .

Jones, C. M. (1999, June). *Automated Technical Foreign Exchange Trading with High Frequency Data.* Ph.D. dissertation, Centre Financial Res., Judge Inst. Management, Univ. Cambridge.

Granville, J. (2000). *Granville's New Key to Stock Market Profits*. Prentice Hall.

Hagan, M., Demuth H., & Beale, M. (1996). *Neural Network Design*. PWS Publishing, Boston, MA.

Hecht-Nielsen, R. (1990). Neurocomputing, Addison-Wesley.

Hinton, G.E., Rumelhart, D. E. & Williams R.J. *Learning internal representations by error propagation.* Parallel Distributed Processing: Explorations in the Microstructures of Cognition. 1: 318-362. MIT Press, Cambridge, 1986.

Huang, W., Lai, K. K., Nakamori, Y., Wang, S. (2007). *Neural networks in finance and economics forecasting.* International Journal of Information Technology & Decision Making. Vol. 6, No. 1, p113–140.

Hutson, J. K., (1983). *Good Trix.* Stocks and Commodities Magazine, Volume 1.5, p105-108.

Kinder Jr, R. J. (1987, May). *Enhanced Williams %R.* Stocks and Commodities Magazine, Volume 5.5, p180-182.

Kohonen, T. (1994). *Self Organizing Maps.* Springer Series in Information Sciences. Springer.

Koulouriotis, D. E., Diakoulakis, I. E., Emiris, D. M., & Zopounidis, C. D. (2005). *Development of dynamic cognitive networks as complex systems approximators: Validation in financial time series.* Applied Soft Computing, 5, pp 157–179.

Kuo, R. J. (1998). *A Decision support system for the stock market through integration of fuzzy neural networks and fuzzy Delphi.* Applied Artificial Intelligence, 12, pp 501–520.

Lawrence , R. (1997). Using Neural Networks to Forecast Stock Market Prices. Department of Computer Science , University of Manitoba . Retrieved from http://people.ok.ubc.ca/rlawrenc/research/Papers/nn.pdf

Leung, M. T., Chen, A., Daouk, H. (2001). *Application of neural networks to an emerging financial market: forecasting and trading the Taiwan stock index.* Computers & Operations Research.

LeBaron, B., & Weigend, A. S. (1997). *A boot strap evaluation of the effect of data splitting on financial time series*. IEEE Transactions on Neural Networks. p213–220.

Levine, E. and Domany, E. (2001). *Resampling Methods for Unsupervised Estimation of Cluster Validity.* Neural Computation 13, 2573-2593, 2001.

Lim. (2002, July). *Money Flow Index (Macro Function).* Retrieved from http://www.lim.com/pdfdocs/money_flow.pdf

Malkiel, B. (1996). *A Random Walk Down Wall Street - 6th Edition.* W. W. Norton & Company.

McCluskey, P. G. (1993, September). *Feedforward and recurrent neural networks and genetic programs for stock market and time series forecasting.* Technical Report CS-93-36, Brown University.

Mendelsohn, L. (1991, June). *The Basics of Developing a Neural Trading System*. Stocks & Commodities, Volume 9:6. p.230-232, 1991

Mendelsohn, L. (1993, September). *Neural Network Development For Financial Forecasting*. Stocks & Commodities, Volume 11:9. p.355-359, 1993

Mendelsohn, L. (1993, October). *Preprocessing Data For Neural Networks*. Stocks & Commodities, Volume 11:10. p.416-420.

Mohan, N., Jha, P., Laha A., & Dutta, G. (2005). *Artificial Neural Network Models for Forecasting Stock Price Index in Bombay Stock Exchange*. Indian Institute of Management. Retrieved from http://www.iimahd.ernet.in/publications/data/2005-10-01_ALaha.pdf

Nikooa, H., Azarpeikanb, M., Yousefib, M.R., Ebrahimpourb, R. and Shahrabadia, Abolfazl . (December 2007). Using A Trainable Neural Network Ensemble for Trend Prediction of Tehran Stock Exchange . IJCSNS International Journal of Computer Science and Network Security, VOL.7 No.12.

Pacelli, V. (2008). *Utilizing Artificial Neural Network Model to Predict Bank Stock Returns*. Università degli Studi “La Sapienza” di Roma .

Pantazopoulos, K. N., Tsoukalas, L. H., Bourbakis, N. G., Bruen, M. J., & Houstis, N. (1998). *Financial prediction and trading strategies using neuro-fuzzy approaches.* IEEE Transactions on Systems Man and Cybernetics – Part B, 28(4).

Para, T.P. and McDonald, K.S. (1995, April 3). *Investing yes, you can beat the market with value stocks--and some discipline--it is possible to outrun the crowd, just like these pros.* Fortune Magazine. Retrieved from http://money.cnn.com/magazines/fortune/fortune_archive/1995/04/03/224106/index.htm

Pring, M. (2001). *Introduction to Technical Analysis*. McGraw-Hill. p.114-115

Refenes, A. N., Azeme-Barac, M., & Zapranis, A. D. (1993). *Stock ranking: Neural networks vs. multiple linear regression.* Proceedings of IEEE ICNN.

Schwager, J. (1999). *Getting Started in Technical Analysis*. John Wiley & Son. p.1-2,110,230,253-258

Shadbolt, J. (2002). *Overfitting, Generalisation and Regularisation*. Neural Networks and the Financial Markets. Springer-Verlag. p.56

U.S. Securities and Exchange Commission. (2010, 18 July). *Stock Splits*. Retrieved from http://www.sec.gov/answers/stocksplit.htm

U.S. Securities and Exchange Commission. (2010, 18 July). *Reverse Stock Splits*. Retrieved from http://www.sec.gov/answers/reversesplit.htm

Senanayake, A., & Janaththanan, J. (n.d.). *Efficient Neural Stock Market Prediction System.* Faculty of Science, University of Peradeniya, Peradeniya, Sri Lanka.

Siekmann, S., Gebhardt, J., & Kruse, R. (1999). *Information fusion in the context of stock index prediction.* Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty, 363–373.

Soni, S. (2011, March). *Applications of ANNs in Stock Market Prediction: A Survey* . International Journal of Computer Science & Engineering Technology (IJCSET) . ISSN : 2229-3345 . Vol. 2 No. 3 .

Swanson, N. R., & White, H. (1997). *A model selection approach to real-time macroeconomic forecasting using linear models and artificial neural networks*. The Review of Economics and Statistics. pg 79, 540–550.

Swingler, K. (1996). *Applying Neural Networks a Practical Guide*. University of Stirling, Stirling, Scotland and Neural Innovation Limited. Academic Press. p. 10,24-25,106, 191.

Tanaka-Yamawaki , M., and Tokuoka , S. 2007. *Adaptive use of technical indicators for the prediction of intra-day stock prices* . Department of Information and Knowledge Engineering, Tottori University, Japan .

Thawornwong, S., Enke, D, & Dagli, C. (2003). *Neural Networks as a Decision Maker for StockTrading: A Technical* Analysis Approach. International Journal of Smart Engineering System Design, 5:313–325.

Thawornwong, S., & Enke, D. (2004). *The adaptive selection of financial and economic variables for use with artificial neural networks.* Neurocomputing, 56, pp 205–232.

Tveter, D. R. (2001). *The Backprop Algorithm.* Retrieved from http://www.dontveter.com/bpr/public2.html

Upadhyaya, B. R., & Eryurek, E. (1992). *Application of neural networks for sensor validation and plant monitoring*. Neural technology (97). p.1780-176.

Walczak S. (2001, Spring). *An Empirical Analysis of Data Requirements for Financial Forecasting with Neural Networks.* Journal of Management Information Systems. Vol. 17, No. 4, p.203-222.

Widrow, B. & Lehr, M. (1990). *30 years of adaptive neural networks: preceptron madaline and back Prorogation.* Processings of IEEE. p1415-1451.

Williams, L. (1985, April). *Ultimate Oscillator.* Technical Analysis of Stocks and Commodities magazine.

White, H. (1988). *Economic Prediction using Neural Networks: The Case of IBM Daily Stock Returns*. Proceeding of the IEEE International Conference on Neural Networks II, 451–458.

Wikowska, D. (1995). *Neural networks as a forecasting instrument for the Polish Stock Exchange.* International Advances in Economic Research, 1(3), pp 232–242.

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

Yahoo Finance. Retrieved from http://finance.yahoo.com.

Yao, J. & Tan, C.J. (2000). *A Case Study on Using Neural Networks to Perform Technical Forecasting of FOREX*. Neurocomputing.

Yao, J. T. & Tan, C. L. (2001). *Guidelines for Financial Forecasting with Neural Networks.* Dept of Computer Science, National University of Singapore.

Yoon, Y. & Swales, G. (1993). *Predicting stock price performance: A neural network approach.* Neural Networks in Finance and Investing., chapter 19, pages 329–342. Probus Publishing Company, 1993.

Zhang, D., Jiang, Q., & Li, X. (2004). *Application of neural networks in financial data mining.* Proceedings of International Conference on Computational Intelligence, pp 392–395.

Appendix: Preprocessing Function Descriptions

This appendix gives details of the functions that pre-processed raw data, and fed as input into to the neural networks.

A1. Inter-day Difference

The inter-day Difference Function is a simple inter-day difference function which calculates the inter-day difference of the open, high, low, close and volume of a stock. inter-day differences present the network with the difference between consecutive days (see Section 4). Again, only a single function was fed into the neural for analysis, and thus the inter-day close difference was selected, a commonly used for inter-day differencing indicator. Function inputs and outputs in the Matlab implementation are given in Equation 17.

$$\begin{aligned} &\textit{Function Inputs:} \\ &\quad Close_t \\ &\textit{Function Outputs:} \\ &\quad InterdayClose_t \end{aligned} \tag{17}$$

where

$$InterdayClose_t = Close_t - Close_{t-1} \tag{18}$$

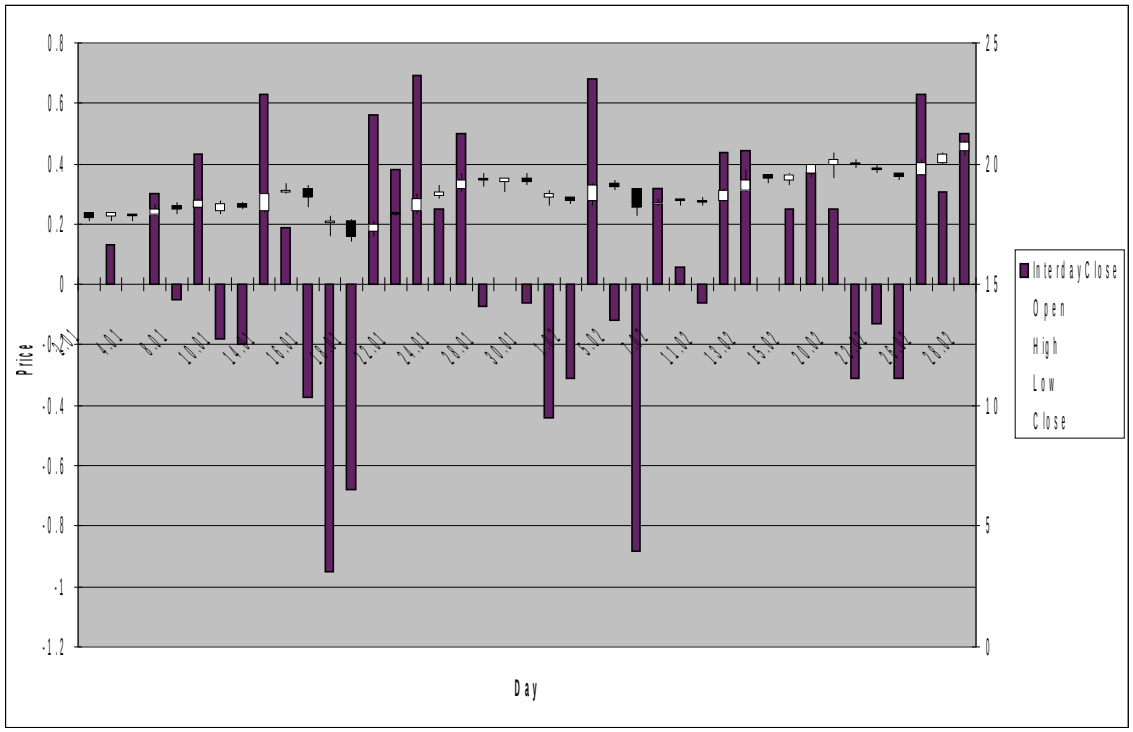

Figure A1.1: Inter-day Differencing for AMR Corporation Jan-Feb 1985

Inter-day differencing was used for target training data creation for indicator price prediction networks (Figure A1.1).

A2. Intra-day Difference

The intra-day difference function uses intra-day differences for prediction. This is calculated by recording the difference between the days open & close and high & low of the daily values; for analysis only the the open–close difference was considered so as to keep a single input into the neural network (Equation 19).

$$\begin{aligned} &\textit{Function Inputs:} \\ &\quad Close_t \\ &\quad Open_t \\ &\textit{Function Outputs:} \\ &\quad IntradayDiff_t \end{aligned} \tag{19}$$

where

$$IntradayDiff = Close_t - Open_t \tag{20}$$

The Intra-day Difference gives a measure of the days volatility, with a positive output for a day where the daily close was higher than the open, and a negative result for a day

when the daily close was lower than the open (Figure A2.1).

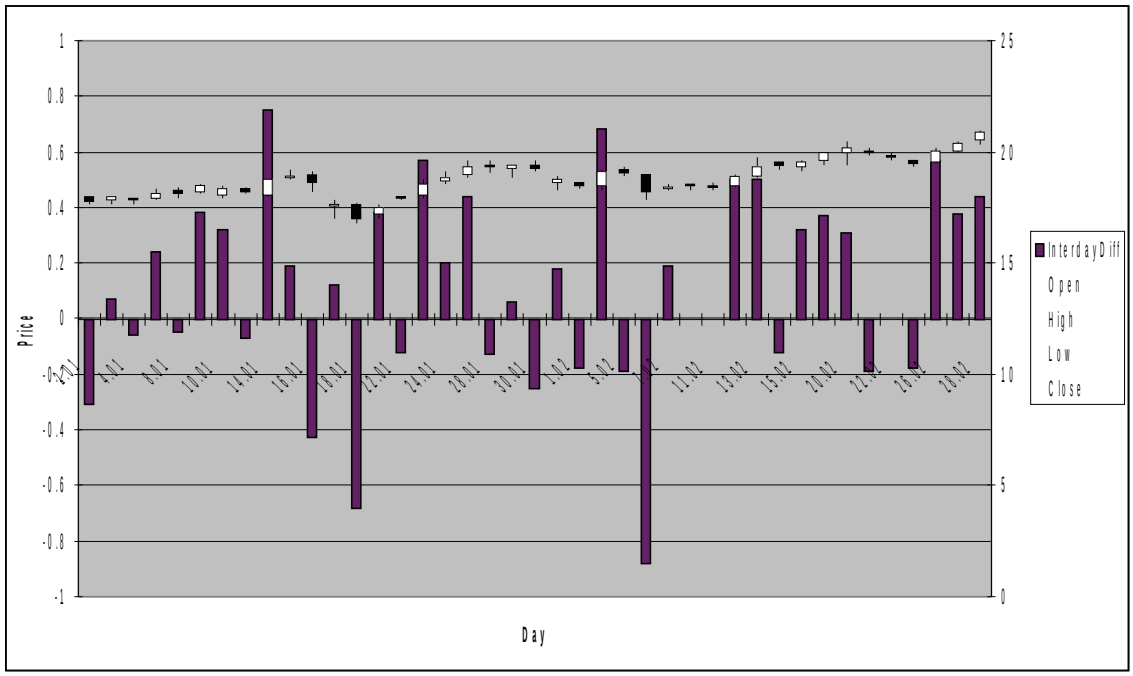

Figure A2.1: : Intra-day Difference Function applied to AMR Corporation Jan-Feb 1985

A3. Moving Average Crossover Indicator

Moving averages form the basis for many technical indicators. The primary use of moving averages is as a smooth function. Once data, especially that of a volatile nature, is smoothed, it can greatly aid in the spotting of trends.

For example, consider Figure A3.1 demonstrating sample stock data with and without smoothing; Notice that with the smoothed moving average it becomes possible to see the trend which otherwise was concealed by the noise of the original data.

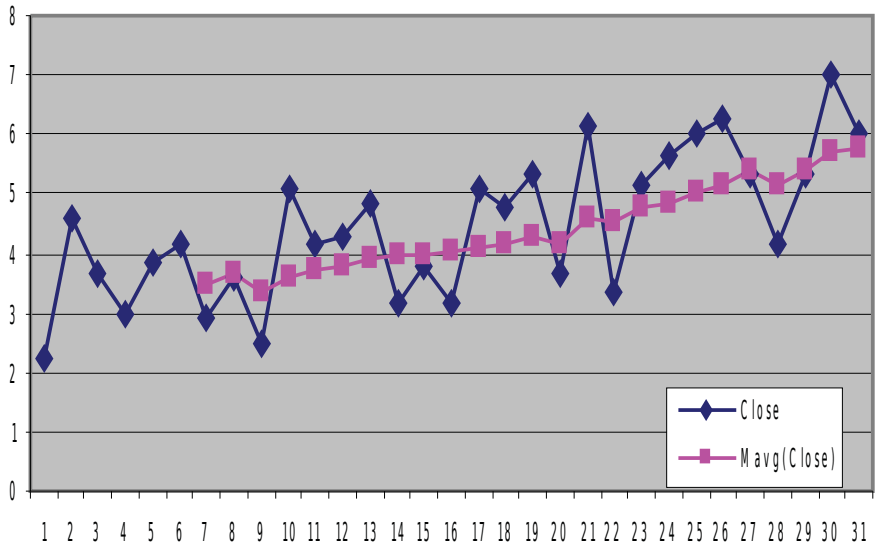

Figure A3.1: Moving Averages help to reveal the underlying trend of the data

The Moving Average Indicator function was created by calculating the cross over of two simple moving averages (Schwager, 1999). Two moving average sets were used, the 5 and 10 day moving average and 10 and 20 day moving average (Equation 22).

$$\begin{aligned} &\textit{Function Inputs:} \\ &\quad \textit{Close}_t \\ &\textit{Function Outputs:} \\ &\quad \textit{CrossOver}_t \end{aligned} \tag{21}$$

$$\textit{CrossOver}_t = \begin{bmatrix} 1 & , & \dfrac{\sum_{t-4}^{t} \textit{Close}_i}{5} < \dfrac{\sum_{t-9}^{t} \textit{Close}_i}{10} \wedge \dfrac{\sum_{t-5}^{t-1} \textit{Close}_i}{5} > \dfrac{\sum_{t-10}^{t-1} \textit{Close}_i}{10} \\ -1 & , & \dfrac{\sum_{t-4}^{t} \textit{Close}_i}{5} > \dfrac{\sum_{t-9}^{t} \textit{Close}_i}{10} \wedge \dfrac{\sum_{t-5}^{t-1} \textit{Close}_i}{5} < \dfrac{\sum_{t-10}^{t-1} \textit{Close}_i}{10} \\ 0 & , & \textit{otherwise} \end{bmatrix} \tag{22}$$

The BSH signals generated by the cross over function is illustrated in Figure A3.1. Notice that whenever the 5-day moving average crosses above or below the 10-day moving average, a buy or sell signal is generated respectively. The short-term moving average is intended to smooth and effectively minimise noise from the graph whilst the long term moving average is intended for represent the current longer term price of the

stock. When the short term crosses the long term moving average, it is signalling a change in market sentiment from bullish to bearish or vice versa.

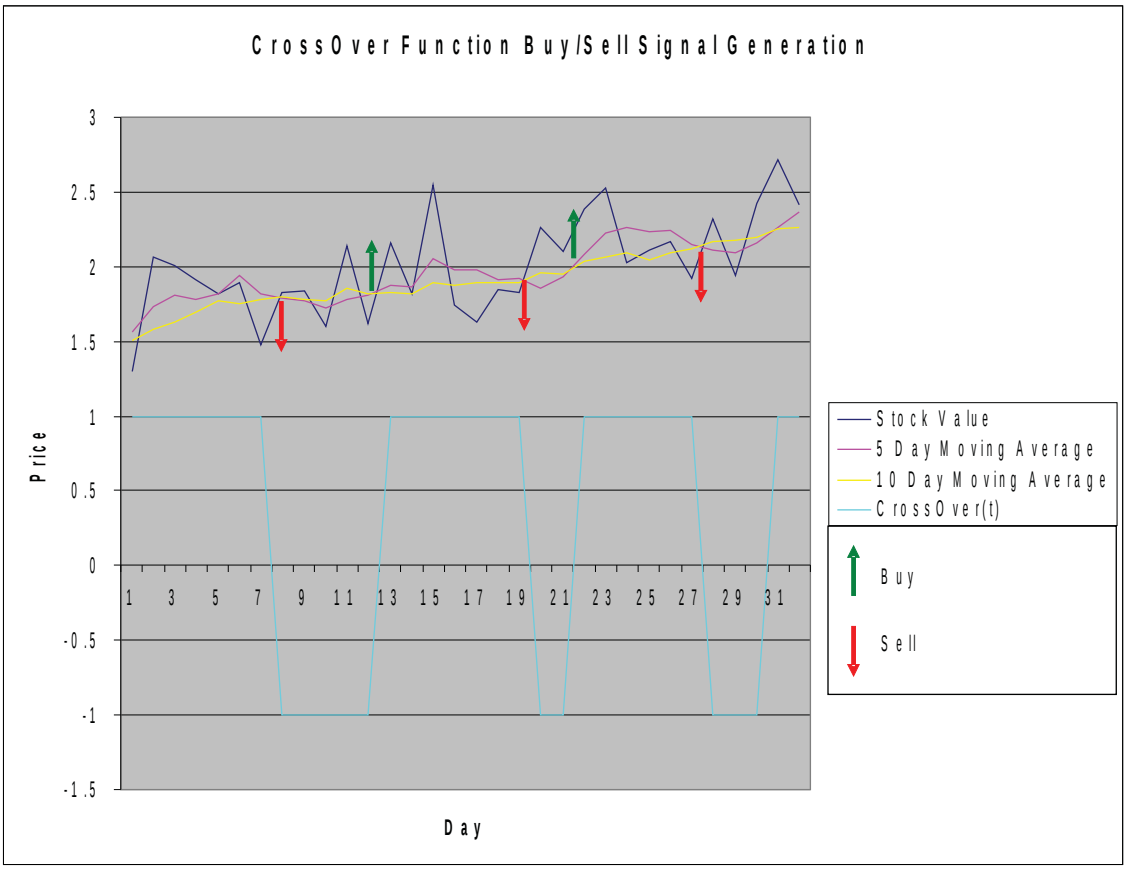

Figure A3.2: Crossover Function with Buy/Sell Signal Generation

A4. Moving Average Decayed Crossover Indicator

As a modification to the standard crossover indicator a decaying function was introduced to create more than simple binary data (Equation 23).

$$\begin{aligned} &\textit{Function Inputs:} \\ &\quad Close_t \\ &\textit{Function Outputs:} \\ &\quad DecayedCrossOver_t \end{aligned} \qquad (23)$$

Inputs to the data were simply the closing value of the stock with the output as the decayed cross over. The decayed cross over was created by halving the value of the Moving Average Crossover Indicator each time period until the output was once again set to high (-1) or low (1); this creates a decayed output with a half-life of one day

(Equation 24).

$$DecayedCrossOver_t = \begin{cases} 1 & , \frac{\sum_{t-4}^{t} Close_i}{5} < \frac{\sum_{t-9}^{t} Close_i}{10} \wedge \frac{\sum_{t-5}^{t-1} Close_i}{5} > \frac{\sum_{t-10}^{t-1} Close_i}{10} \\ -1 & , \frac{\sum_{t-4}^{t} Close_i}{5} > \frac{\sum_{t-9}^{t} Close_i}{10} \wedge \frac{\sum_{t-5}^{t-1} Close_i}{5} < \frac{\sum_{t-10}^{t-1} Close_i}{10} \\ \frac{DecayedCrossOver_{t-1}}{2} & , otherwise \end{cases} \quad (24)$$

This function was uniquely created as a modified form of standard cross over, to add a memory effect, such that the effect of a cross-over could have a continuing, but diminishing effect over the next period of time.

A comparison between the decayed and normal crossover functions is illustrated graphically in (Figure A4.1). Notice that the decayed output remains above 0.1 until the fifth day, so as to tapper off the effect of the cross over. Whilst the effect of using decayed output was compared to not using it, modification of the half-life of the decay was not studied.

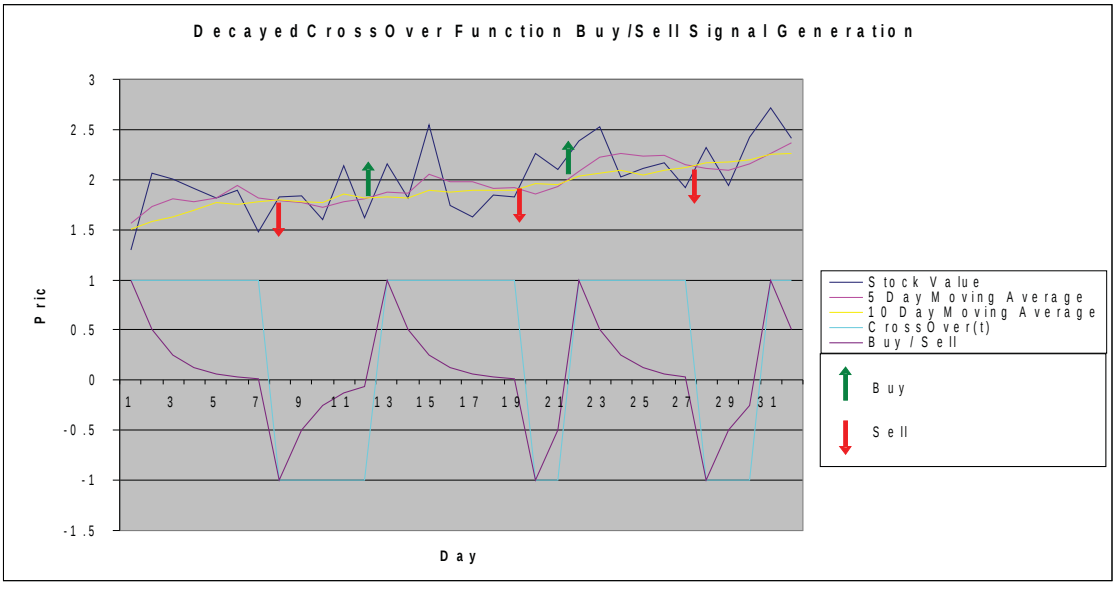

Figure A4.1: Decayed Cross Over Function with Buy/Sell Signal Generation

This decay factor acts as a form of artificial memory for the neural network whereby a Buy/Sell event has a continuously decreasing flow on effect after it is first signalled

A5. Moving Average Difference Indicator

The final moving average indicator was the Moving Average Difference Indicator. This function simply returned the difference of the two moving averages so as to provide a simple, but less binary input to the neural network than that of the buy/sell cross over alternative (Equation 25).

$$MovingAverageDiff_t = \frac{\sum_{t-4}^{t} Close_i}{5} - \frac{\sum_{t-9}^{t} Close_i}{10} \qquad (25)$$

This moving average function was applied without any artificial analysis to the two moving averages (Equation 26).

$$\begin{aligned} &Function\ Inputs: \\ &\quad CrossOver \\ &Function\ Outputs: \\ &\quad MovingAverageDiff_t \end{aligned} \qquad (26)$$

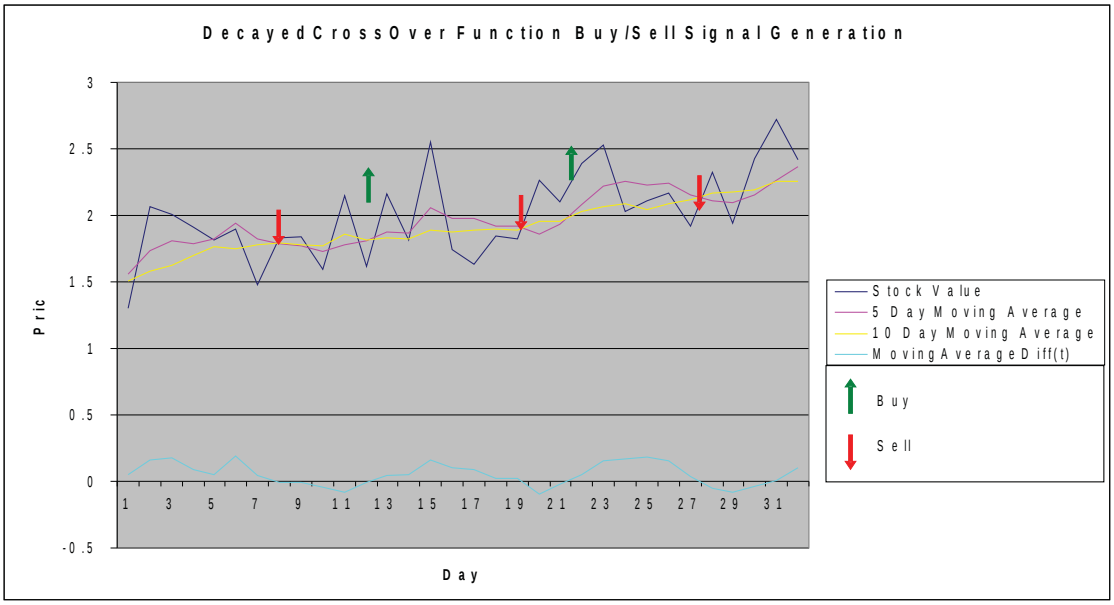

Figure A5.1: : Decayed Cross Over Function with Buy/Sell Signal Generation

A6. Relative Strength Index (RSI)

Relative Strength Index was introduced by Wilder (1978) as a means of quantifying a

stocks recent gains compared to it's recent losses. RSI can be stated as per Equation 27.

$$RSI_t = 100 - \frac{100}{1 + RS_t} \quad (27)$$

where

$$RS_t = \frac{UpDayAverage_t}{DownDayAverage_t} \quad (28)$$

An 'up' close is a day on which the close is higher than the previous day (Equation 31) while a 'down' close day is when the close is lower than the previous day's close (Equation 32). The 'up' day average is calculated by summing the total 'up' day inter-day differences and dividing by 14 (Equation 29). Similarly, the 'down' day average is calculated by summing the total 'down' day inter-day differences and dividing by 14 (Equation 30). To calculate RS the 'up' day average is divided by the 'down' day average (Equation 28).

$$UpDayAverage_t = \frac{\sum_{t}^{t-13} UpDay_t}{14} \quad (29)$$

$$DownDayAverage_t = \frac{\sum_{t}^{t-13} DownDay_t}{14} \quad (30)$$

$$UpDay_t = \begin{bmatrix} Close_t - Close_{t-1} & , & Close_t \geq Close_{t-1} \\ 0 & , & otherwise \end{bmatrix} \quad (31)$$

$$DownDay_t = \begin{bmatrix} Close_{t-1} - Close_t & , & Close_{t-1} > Close_t \\ 0 & , & otherwise \end{bmatrix} \quad (32)$$

RSI was outputted in its raw form without inter-day differencing, which was not applied as the value of the RSI, rather than the inter-day difference holds more information for the RSI.

$$
\begin{aligned}
&Inputs: \\
&\quad UpDayAverage_t \\
&\quad DownDayAverage_t \\
&Outputs: \\
&\quad RSI_t
\end{aligned}
\tag{33}
$$

A7. Relative Strength Index (RSI) Analysis

Analysis was applied to the RSI index to give trigger point based on well known technical properties of the RSI itself.

1. When the RSI is above 70 it is considered a topping formation (local maxima)
   When the RSI is below 30 it is considered to be a bottom (local minima) (Equation 34).

$$
TopBottom_t = \begin{bmatrix} 1 & , & RSI_t \le 30 \\ -1 & , & RSI_t \ge 70 \\ 0 & , & otherwise \end{bmatrix} \tag{34}
$$

2. When the RSI is below 70 and above 50 it is considered bearish.
   When the RSI is above 30 and below 50 it is considered bullish (Equation 35).

$$
BullishBearishA = \begin{bmatrix} 1 & , & RSI_t < 70 \wedge RSI_t > 50 \\ -1 & , & RSI_t > 30 \wedge RSI_t < 50 \\ 0 & , & otherwise \end{bmatrix} \tag{35}
$$

3. A 'buy' signal is generated the RSI moves out of an overbought area
   A 'sell' signal is generated the RSI moves out of an oversold area (Equation 36).

$$
BuySell = \begin{bmatrix} 1 & , & RSI_t < 70 \wedge RSI_{t-1} \ge 70 \\ -1 & , & RSI_t > 30 \wedge RSI_{t-1} \le 30 \\ 0 & , & otherwise \end{bmatrix} \tag{36}
$$

4. When the RSI reaches a local maxima it is considered bullish
   When the RSI reaches a local minima it is considered bearish (Equation 37).

$$
LocalMaxMin = \begin{bmatrix} 1 & , & RSI_t \ge max(RSI_{t-1} : RSI_{t-13}) \\ -1 & , & RSI_t \le min(RSI_{t-1} : RSI_{t-13}) \\ 0 & , & otherwise \end{bmatrix} \tag{37}
$$

5. When the security is at a 14 period high but the RSI is not it is bearish signal.

When the RSI is at a 14 period high but the security is not it is a bullish signal (Equation 38).

$$BullishBearishB = \begin{bmatrix} -1 & , & Close_t \geq max(Close_{t-1}:Close_{t-13}) \wedge RSI_t \leq max(RSI_{t-1}:RSI_{t-13}) \\ 1 & , & RSI_t \geq max(RSI_{t-1}:RSI_{t-13}) \wedge Close_t \leq max(Close_{t-1}:Close_{t-13}) \\ 0 & , & otherwise \end{bmatrix} \quad (38)$$

6. When the security is at a 14 period low but the RSI is not it is bullish signal. When the RSI is at a 14 period low but the security is not it is bearish signal (Equation 39).

$$BullishBearishC = \begin{bmatrix} 1 & , & Close_t \leq min(Close_{t-1}:Close_{t-13}) \wedge RSI_t > min(RSI_{t-1}:RSI_{t-13}) \\ -1 & , & RSI_t \leq min(RSI_{t-1}:RSI_{t-13}) \wedge Close_t > min(Close_{t-1}:Close_{t-13}) \\ 0 & , & otherwise \end{bmatrix} \quad (39)$$

Following the above parameters the inputs and outputs of the RSI Analysis function were trained as follows:

$$\begin{aligned} &Inputs: \\ &\quad RSI_t \\ &Outputs: \\ &\quad RSIAnalysis_t \end{aligned} \quad (40)$$

where *RSIAnalysis* is the sum of the individual indicators

$$\begin{aligned} RSIAnalysis_t = {} & TopBottom_t + BuySell_t + LocalMaxMin_t \\ & + BullishBearishA_t + BullishBearishB_t + BullishBearishC_t \end{aligned} \quad (41)$$

This sum of signals was done such to reduce the number of inputs to the neural network, which would otherwise increase six fold given the above equations.

A8. Bollinger Bands

Bollinger bands is a simple measure of volatility created by three bands calculated as follows:

1. Simple Moving Average + 2 x Standard Deviation
2. Simple Moving Average
3. Simple Moving Average - 2 x Standard Deviation

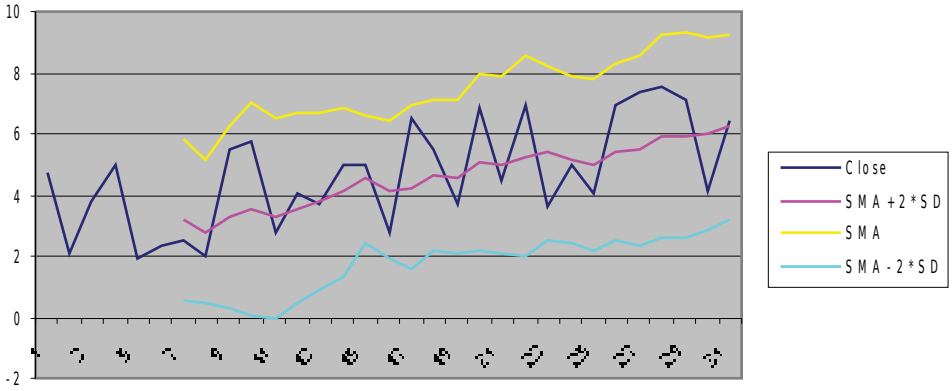

Figure A8.1: : Bollinger Bands

The output of the Bollinger Bands was simply done by taking the difference between the upper and lower bands (Figure A8.1 and Equation 42). The basis of this is that a larger difference indicates greater volatility and a lesser difference lower volatility. When the bands are far apart this is an indication that the current trend may be at an end.

$$\begin{aligned} &Function\ Inputs: \\ &\quad Close \\ &Function\ Outputs: \\ &\quad BollingerDifference_t \end{aligned} \tag{42}$$

$$BollingerDifference_t = BollingerTop_t - BollingerBottom_t \tag{43}$$

$$BollingerTop_t = BollingerTop(Close) \tag{44}$$

$$BollingerBottom_t = BollingerBottom(Close) \tag{45}$$

A9. Volatility Analysis

Volatility Analysis was implemented in a modified form as a preprocessing element. Volatility Analysis is a mixture of various indicators in an attempt to gauge the volatility of the market. The following outputs are given by volatility analysis:

1. Today is considered an 'Up Day' if today's close was higher than yesterday's
   Today is considered a 'Down Day' if today's close was lower than yesterday's

$$\textbf{CloseUpDownDay}_t = \begin{bmatrix} 1 & , & Close_t > Close_{t-1} \\ -1 & , & Close_t < Close_{t-1} \\ 0 & , & otherwise \end{bmatrix} \quad (46)$$

2. Today is considered an 'Up Day' if today's close was higher than yesterday's
   Today is considered a 'Down Day' if today's close was lower than yesterday's

$$\textbf{VolumeUpDownDay}_t = \begin{bmatrix} 1 & , & Volume_t > Volume_{t-1} \\ -1 & , & Volume_t < Volume_{t-1} \\ 0 & , & otherwise \end{bmatrix} \quad (47)$$

3. Today's volume is outputted as a percentage when compared to that of yesterday.
   If today's volume was higher than the 90-day moving average, then today is an 'Up Volume Day'
   If today's volume was lower than the 90-day moving average, then today is 'Down Volume Day'

$$\textbf{VolumeUpDownDay}_t = \begin{bmatrix} 1 & , & VolumePercentage_t > 0 \\ -1 & , & VolumePercentage_t < 0 \\ 0 & , & otherwise \end{bmatrix} \quad (48)$$

$$\textbf{VolumePercentage}_t = \frac{Volume_t - \frac{\sum_{t-89}^{t} Volume_i}{90}}{\frac{\sum_{t-89}^{t} Volume_i}{90}} \quad (49)$$

4. Today's Bollinger band width is outputted as compared to the 20 day average.
   1. If the Bollinger bands are wider than the average then it is considered wider-than-normal.
      If the Bollinger bands are narrower than the average then it is considered narrower-than-normal.

$$\textbf{BollingerWiderNarrower}_t = \begin{bmatrix} 1 & , & BollingerWidth > 0 \\ -1 & , & BollingerWidth < 0 \\ 0 & , & otherwise \end{bmatrix} \quad (50)$$

   2. If the Bollinger bands are twice as wide as the average then the stocks price

is likely to drop, otherwise called a 'consolidation'

If the Bollinger bands are 40% as wide as the average or less then this signals that future volatility can be expected.

$$\mathbf{BollingerExpectation}_t = \begin{bmatrix} 1 & , & BollingerWidth > 100 \\ -1 & , & BollingerWidth < -40 \\ 0 & , & otherwise \end{bmatrix} \quad (51)$$

$$\mathbf{BollingerWidth}_t = \left| \left(\frac{BollingerDiff_t - \frac{\sum_{t-89}^{t} BollingerDiff_i}{90}}{\sum_{t-89}^{t} BollingerDiff_t} \right) \right| \quad (52)$$

5. Today is considered bullish if:
    1. There have been less than six days since the last instance that the Bollinger bands were less than 40% of the average
    2. Today's volume is at least 25% higher than the 20 day moving average
    3. Today's price data is 'engulfing bull' of yesterday's, whereby today's open is higher than ' close, and the open-close span engulfs that of the previous day (FX Works).
6. Today is considered bearish if:
    1. There have been less than six days since the last instance that the Bollinger bands were less than 40% of the average
    2. Today's volume is at least 25% higher than the 20 day moving average
    3. Today's price data is 'engulfing bear' of yesterday's, whereby today's close is lower than today's open, and the close-open span engulfs that of the previous day (FXWORDS).

$$BullishBearishVolatility_t = x \wedge Volume_t \geq 1.25 \times VolumeAverage \wedge engulfingbear_t \quad (53)$$

$$VolumeAverage_t = \frac{\sum_{t-19}^{i=t} Volume_i}{20} \quad (54)$$

$$BollingerLessThanAverage_t = BollingerDiff_t < 0.6 \times BollingerWidth_t \quad (55)$$

$$BollingerDaysLessThanAverage_t = \sum_{t}^{t-5} BollingerLessThanAverage_t \quad (56)$$

$$BollingerDaysLessThanAverage_t = \begin{bmatrix} 1 & , & BollingerLessThanAverage_t > 0 \\ 0 & , & otherwise \end{bmatrix} \quad (57)$$

$$\boldsymbol{BullishBearishVolatility_t} = \begin{bmatrix} 1 & , & a_t < 40 \wedge b_t = true \wedge c_t \\ -1 & , & a_t < 40 \wedge d_t = true \wedge c_t > 1.25 \\ 0 & , & otherwise \end{bmatrix} \quad (58)$$

where

$$a_t = Min\left(BollingerExpectation_t : BollingerExpectation_{t-4}\right) \quad (59)$$

$$b_t = EngulfingBull_t \quad (60)$$

$$c_t = Volume_t > 1.25 * \frac{\sum_{t-19}^{t} Volume_i}{20} \quad (61)$$

$$d_t = EngulfingBear_t \quad (62)$$

These outputs were given in their raw form and passed onto the neural network to analyse it's ability to learn such parameters.

A10. On Balance Volume

On Balance Volume (OBV) is a measure of volume flow of stocks traded (Granville, 1963). OBV is based on the commonly assumed theory among technicians that volume precedes price changes. If volume is increasing in line with a positive or negative price move on an equity, it is said that the volume is supporting the move; for a rising price this means more money is being poured into the market in support for the increasing price, whilst for an decreasing price, the increasing volume implies that the down trend will continue (Equation 63).

$$OBV_t = \begin{bmatrix} Close_t > Close_{t-1} & , & OBV_t = OBV_{t-1} + Volume_t \\ Close_t = Close_{t-1} & , & OBV_t = OBV_{t-1} \\ Close_t < Close_{t-1} & , & OBV_t = OBV_{t-1} - Volume_t \end{bmatrix} \quad (63)$$

Conversely, when a stock is moving in the opposite direction to the volume, it is considered that the current trend will likely reverse or otherwise run out of momentum. As demonstrated in Equation 63, inputs to the OBV calculation are simply the stocks

Close and Volume with outputs as only the raw OBV (Equation 64).

$$\begin{aligned} &Inputs: \\ &\quad Close_t \\ &\quad Volume_t \\ &Outputs: \\ &\quad OBV_t \end{aligned} \tag{64}$$

A sample of the OBV technical indicator is show in Figure A10.1. As with other trend indicators a divergence suggests that the stocks current trend is not supported and will either reverse or level off. Close examination of Figure A10.1 shows that at several points the OBV stops supporting the prevailing price trend, which subsequently levels off after this divergence.

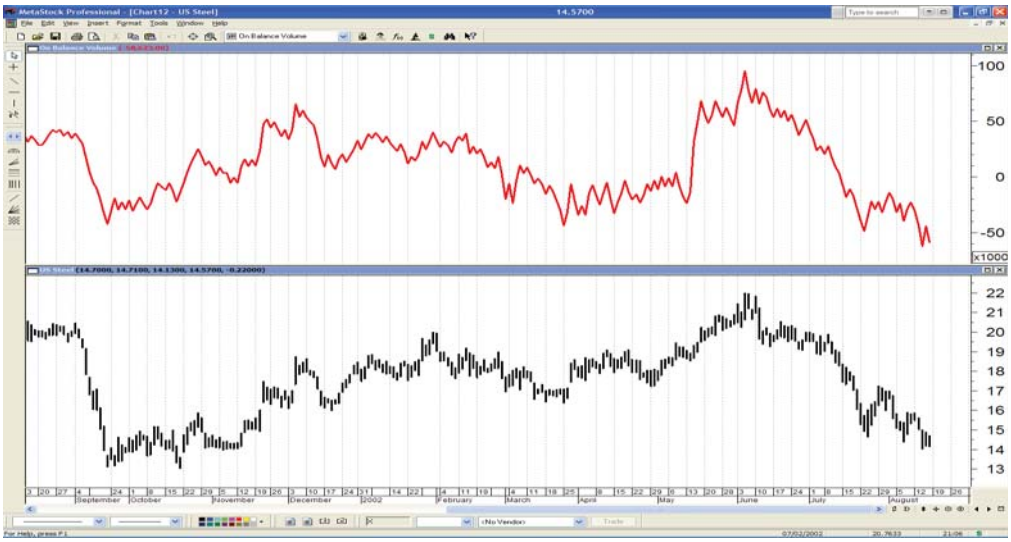

Figure A10.1: : On Balance Volume applied to US Steel (August 2001 - August 2002)

A11. Accumulation Distribution Line

The Accumulation Distribution Line is one of many indicators used to measure the flow of money into or out of a stock. As with many other flow indicators it is based on the idea that volume will increase or decrease before a price movement. Unlike the OBV flow indicator, which uses the inter-day price difference as a means of assessing the direction of money flow, ADL uses the intra-day price statistics primarily focusing on the location of the daily close versus the daily high and low to determine money flow.

Much like OBV the Accumulation Distribution Line (ADL) is based on the principle that volume proceeds price; For example it is not abnormal for the volume of a stock to increase prior to a major price move. Unlike OBV, which uses the inter-period (inter-day) values to set the volumes as positive or negative, ADL uses the close relative to the period range.

The value for ADL is calculated as follows:

$$ADL_t = CLV_t \times Volume_t \tag{65}$$

where

$$CLV_t = \frac{(Close_t - Low_t) - (High_t - Close_t)}{High_t - Low_t} \tag{66}$$

The CLV value works on the basis that if today's close is equal to the high or low of the period, then the CLV will be 1 or -1 respectively. A close at the mid point of the range will give a CLV of zero.

$$\begin{aligned} &Inputs: \\ &\quad Open_t \\ &\quad Low_t \\ &\quad High_t \\ &\quad Close_t \\ &\quad Volume_t \\ &Outputs: \\ &\quad ADL_t \end{aligned} \tag{67}$$

Figure A11.1: The Application of Accumulation Distribution Line (ADL) to Exxon Mobil Corp

Interpretation of ADL is two fold. Firstly ADL should always move in the direction of the prevailing trend to confirm the validity of that trend (Figure A11.1), "Confirmation of Advance"). Secondly, a positive divergence, whereby the ADL and stock price are trending upwards and downwards respectively, indicates that the stock price will again increase (Figure A11.1, "Positive Divergence"). ADL was implemented without analysis (Equation 67) leaving it to the neural network to extract these subtleties.

A12. Aroon Oscillator

Aroon is used to show the trend of a stock with the intention that a new trend can easily be spotted.

Aroon is calculated as follows:

$$AroonHigh_t = \frac{100 \times (Periods - PeriodsSinceHighestHigh_t)}{Periods} \quad (68)$$

$$AroonLow_t = \frac{100 \times (Periods - PweriodsLow_t)}{Periods} \quad (69)$$

where

Periods is the number of periods in the selected Aroon analysis

PeriodsSinceHighestHigh is the number of periods since the periods highest high

PeriodsSinceLowestLow is the number of periods since the periods lowest low

Interpretation of the Aroon indicator is based on the *AroonHigh* and *AroonLow* lines; when the *AroonHigh* or *AroonLow* line passes below 50 then it has lost its upwards or downwards momentum respectively; a passes above 70 indicates strong momentum. The Aroon was implemented the difference of the *AroonHigh* and *AroonLow* lines, known as the AroonOscillator (Equation 70).

$$AroonOscillator_t = AroonHigh_t - AroonLow_t \quad (70)$$

The *AroonOscillator* is simply interpreted as showing whether there is a current upwards or downwards trend when it is above or below zero respectively; the further the *AroonOscillator* from zero, the strong the trend. The Aroon function was implemented with periods of 10,30 and 70 days (Equation 71).

$$\begin{aligned} &Inputs: \\ &\quad High_t \\ &\quad Low_t \\ &\quad Periods \\ &Outputs: \\ &\quad AroonOscillator_t \end{aligned} \quad (71)$$

A13. Commodity Channel Index

The Commodity Channel Index (CCI) is a cyclical trend indicator developed by Donald Lambert in 1980. Calculation of CCI is complex as per Equations 72 - 75. CCI is based on a Period value which ultimately determines the volatility of the indicator.

$$TypicalPrice_t = \frac{High_t + Low_t + Close_t}{3} \quad (72)$$

$$TypicalPriceMovAvg_t = \frac{\sum_{i=t-19}^{t} TypicalPrice_i}{20} \quad (73)$$

$$MeanDeviation_t = \frac{\sum_{i=t-19}^{t} \left| (TypicalPrice_i - TypicalPriceMovAvg_i) \right|}{20} \quad (74)$$

$$CCI_t = \frac{TypicalPrice_t - TypicalPriceMovAvg_t}{0.015 \times MeanDeviation_t} \quad (75)$$

The raw CCI value was given raw to the neural network without further processing (Equation 76), however interpretation of CCI is quite straight forward; CCI values of above 100 and below -100 can be interpreted as buy and sell signals respectively. These same signals can also be interpreted as overbought and oversold. The raw CCI output was fed into the neural network as per Equation 76.

$$\begin{aligned} &Function\ Inputs: \\ &\quad Close_t \\ &\quad High_t \\ &\quad Low_t \\ &Function\ Outputs: \\ &\quad CCI_t \end{aligned} \quad (76)$$

A14. Bullish/Bearish Moving Average Convergence Divergence

Moving Average Convergence Divergence (MACD) is a well known technical indicator for producing bullish and bearish signals using lagging moving averages developed by in the 1970's (Appel, 2006) . MACD is simply the difference between the exponential 12 and 26 day moving averages (Equation 77). A control or trigger line is typically included in MACD which is made up of a 9 day exponential moving average (Equation 78).

$$\begin{aligned} MACD_t = &ExponentialMovingAverage(Close_t, Close_{t-1}, \ldots Close_{t-11}) \\ &- ExponentialMovingAverage(Close_t, Close_{t-1}, \ldots Close_{t-25}) \end{aligned} \quad (77)$$

$$Control_t = ExponentialMovingAverage(Close_t, Close_{t-1}, \ldots Close_{t-8}) \quad (78)$$

On a very basic level MACD can be understood by considering its simple moving average make-up; a positive MACD simply shows that the 12 day moving average is more than the 26 day moving average, and vice versa for a negative MACD. If the MACD is widening in an upwards direction this is caused by a more recent acceleration in the stock price; this contrasts to a constant increasing price which would give a more consistent MACD difference.

Simple Analysis of the MACD function was carried out with signals for crossover between the MACD 12 and 26 day moving averages and the control line, with a bullish/bearish indicator for when the MACD was above or below the control line respectively (Equation 79).

$$BullBear_t = \frac{MACD_t}{Control_t} \tag{79}$$

MACD was presented to the neural network in processed and unprocessed forms. The outputs and inputs to the unprocessed MACD function is given in Equation 80.

$$\begin{aligned} &Function\ Inputs: \\ &\quad Close_t \\ &Function\ Outputs: \\ &\quad BullBear_t \end{aligned} \tag{80}$$

A sample of MACD is shown in Figure A14.1; it can be seen that when the MACD crosses the above control line, it indicates a buy signal, where as a cross below signals a sell signal. The bullish or bearishness of the stock is signalled whenever the MACD or greater than or less than the control line respectively.

Figure A14.1: MACD Analysis Applied to Nike for May to August 2002

A15. Zigzag Filter

The ZigZag indicator is a form of extreme filter, removing all noise and following the basic trend of a stock. The ZigZag filter ignores all daily high to low changes below a certain percentage as specified by the sensitivity parameter of the filter. For simulation a 1.5% percent change inter-day low-to-low and high-to-high threshold.

For example a 5% sensitivity ZigZag indicator would create a trend line that only changes it's gradient once the 5% threshold movement in a day has been reached. The zigzag implementation was based on the stocks close price and function output a simple zigzag function of the close (Equation 81).

$$\begin{aligned} &Function\ Inputs: \\ &\quad Close_t \\ &Function\ Outputs: \\ &\quad ZigZag_t \end{aligned} \qquad (81)$$

Calculation of the ZigZag function is based on the inter-day high and low differencing. The ZigZag primary points are defined as per Equation 82; in-between points which do not qualify due to intra-day differences of less than 5% are calculated on subsequent runs as per Equations 83 and 84. These subsequent calculations are done only for points which were set to zero during the first data processing where start is the t day value for the previous primary point and end is the t value for the next primary point; this function calculates the points of a simple straight line plotted between the previous and next primary point.

$$ZigZag_t = \begin{bmatrix} High_t & , & High_t > 1.05 \times High_{t-1} \\ Low_t & , & Low_t < 0.95 \times Low_{t-1} \\ 0 & , & otherwise \end{bmatrix} \quad (82)$$

$$ZigZag_t = High_{start} + (t - start) \times \frac{High_{end} - High_{start}}{end - start} \quad (83)$$

$$ZigZag_t = Low_{start} + (t - start) \times \frac{Low_{end} - Low_{start}}{end - start} \quad (84)$$

A sample zigzag indicator is shown in Figure A15.1 demonstrates the filtering effect of the zigzag function.

Figure A15.1: Zigzag Analysis with 5% threshold

A16. Price Channels

Similar to Standard deviation price channels create upper and lower trend lines around a stock. For price channels, this trend line is based on the highest and lowest n-period close for the upper and lower n-period trend lines respectively.

Much like other channelised ceiling-floor indicators, Price Channels can easily be interpreted as a buy and sell respectively for an upside and downside breakout as exemplified in Figure A16.1.

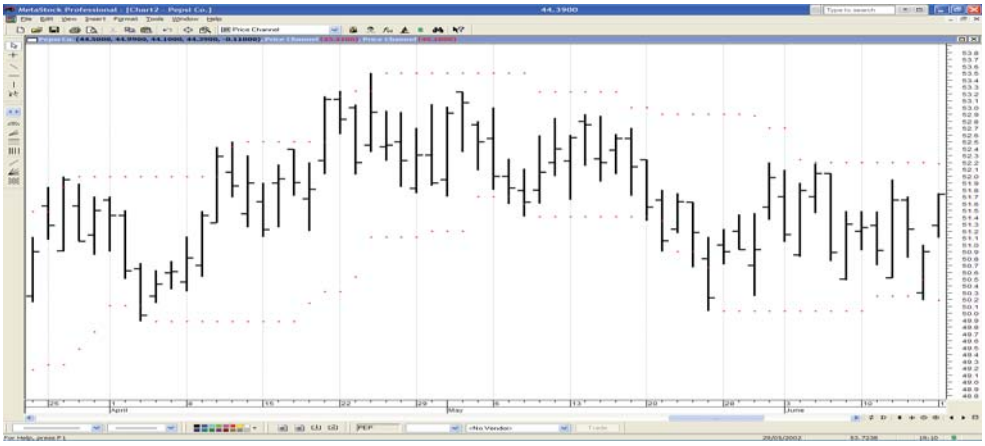

Figure A16.1: Price Channel Applied to Pepsi Co. with 10 day price channel.

Price Channels were implemented in their raw form (Equations 85 - 86) in addition to their analysed form (Equation 88).

$$PriceChannelHigh_t = Max(Close_{t-1}, Close_{t-2}, ... Close_{t-20}) \quad (85)$$

$$PriceChannelHigh_t = Min(Close_{t-1}, Close_{t-2}, ... Close_{t-20}) \quad (86)$$

The Analysed form signalled when the stock price breached the price channel above and below by signalling a bullish and bearish signal respectively (Equation 87).

$$BullBear_t = \begin{bmatrix} 1 & , & High_t > PriceChannelHigh_t \\ -1 & , & Low_t < PriceChannelLow_t \\ 0 & , & otherwise \end{bmatrix} \quad (87)$$

The Inputs and Outputs of the Analysed Price Channel is given in Equation 88.

$$\begin{aligned} &Function\ Inputs: \\ &\quad Close_t \\ &Function\ Outputs: \\ &\quad BullBear_t \end{aligned} \quad (88)$$

A17. Williams %R

Williams %R is a stochastic oscillator, originally described in Larry Williams (Kinder, 1987). Williams takes one parameter in periods (days, weeks, months) which ultimately changes the sensitivity of the indicator.

Calculation of Williams %R is as follows:

$$WR_t = \frac{Close_t - HighestHigh_t}{HighestHigh_t - LowestLow} \times -100 \quad (89)$$

$$HighestHigh_t = Max(High_t, High_{t-1} \ldots High_{t-periods+1}) \quad (90)$$

$$LowestLow_t = Min(Low_t, Low_{t-1} \ldots Low_{t-periods+1}) \quad (91)$$

A period of 14 is typically used for Williams %R but can be varied to adjust sensitivity of the data. Williams was fed into the neural network as per Equation 92.

$$\begin{aligned} &Function\ Inputs: \\ &\quad High_t \\ &\quad Low_t \\ &\quad Close_t \\ &Function\ Outputs: \\ &\quad WR_t \end{aligned} \quad (92)$$

Figure A17.1: Williams %R applied to Microsoft Corp from May to August 2002

From Figure A17.1 the Williams %R can be seen. Whenever Williams %R goes above -20 or below -80 it is seen as overbought (bearish) and oversold (bullish) respectively. From this example it can be seen that in the first two rises of Williams %r above the -20 level leads to a decline in the stock price; this occurred similarly for the dips of Williams %R below -80 (Equation 93).

$$\begin{aligned} Williams\%R \geq -20 \quad & OverBought \\ Williams\%R < -80 \quad & OverSold \end{aligned} \tag{93}$$

A18. Ultimate Oscillator

The Ultimate Oscillator, as described by Williams (1985), uses multiple overlapping data-periods to measure overbought and oversold signals. The three parameters that typically define the ultimate oscillator are 7, 14 and 28 day overlapping periods. The 14 day period includes the 7 day period, the 28 day period includes both the 14 and 7 day period meaning that the most recent periods have a more significant effect on the results than older periods.

The ultimate oscillator can be defined as follows:

$$UO_t = \frac{4\times\frac{BuyingPressureA_t}{TrueLowA_t} + 2\times\frac{BuyingPressureB_t}{TrueLowB_t} + 1\times\frac{BuyingPressureC_t}{TrueLowC_t}}{4+2+1} \tag{94}$$

where

$$BuyingPressureA_t = \sum_{i=t}^{t-27} Close_i - TrueLow_i \tag{95}$$

$$BuyingPressureB_t = \sum_{i=t}^{t-14} Close_i - TrueLow_i \tag{96}$$

$$BuyingPressureC_t = \sum_{i=t}^{t-7} Close_i - TrueLow_i \tag{97}$$

$$TrueLow_t = \begin{bmatrix} Low_t & , & Low_t < Close_{t-1} \\ Close_{t-1} & , & Low_t \geq Close_{t-1} \end{bmatrix} \tag{98}$$

$$TrueRange_t = \begin{bmatrix} High_t - Low_t & , & (High_t - Low_t) > (High_t - Close_{t-1}) \wedge (High_t - Low_t) > (Close_{t-1} - Low_t) \\ High_t - Close_{t-1} & , & (High_t - Close_{t-1}) > (High_t - Low_t) \wedge (High_t - Close_{t-1}) > (Close_{t-1} - Low_t) \\ Close_{t-1} - Low_t & , & (Close_{t-1} - Low_t) > (Close_{t-1} - Low_t) \wedge (Close_{t-1} Low_t) > (High_t - Close_{t-1}) \end{bmatrix} \tag{99}$$

The Matlab implementation of this function takes the following input and output parameters:

$$\begin{aligned} & Function\ Inputs: \\ & \quad High_t \\ & \quad Low_t \\ & \quad Close_t \\ & Function\ Outputs: \\ & \quad UO_t \end{aligned} \tag{100}$$

A19. TRIX Data

A Momentum indicator, TRIX filters stock movements of lesser significance by the application of a triple exponential moving average to the closing price of a security. The period length selected for the filters application effectively filters out high frequency price moments (Hutson, 1983).

Trix gives the percent rate of change of a stock, which, when compared to a signal line, can be used to predict the turning points in a stock's trend. The signal line is either the stock's closing price itself or more commonly a simple smaller period moving average. Trix is typically used with other indicators due to the high number of false signals generated during times when a stock is not moving strongly in an upwards or downwards direction. The equation for the calculation of TRIX is given in Equation 101 and 102.

$$Trix_t = \frac{c_t}{MAvgC_{(t-1)}} \tag{101}$$

where

$$MAvgC = \frac{\sum_{i=t}^{15} (MAvgB_i)}{15}$$

$$MAvgB = \frac{\sum_{i=t}^{15} (MAvgA_i)}{15} \tag{102}$$

$$MAvgA = \frac{\sum_{i=t}^{15} (Close_i)}{15}$$

Trix is calculated by performing a triple exponential moving average on the closing price of the stock while Equation 101 calculates the simple percentage increase / decrease in the movement of the indicator.

The Matlab implementation of this function takes the following input and output parameters.

$$\begin{aligned} &Function\ Inputs: \\ &\quad Close_t \\ &Function\ Outputs: \\ &\quad Trix_t \end{aligned} \qquad (103)$$

A20. Standard Deviation

A standard deviation indicator is simply a moving standard deviation performed to a stock to measure its volatility (Equation 104) and highlighted in Ford Equity Research (2002).

$$\begin{aligned} &Function\ Inputs: \\ &\quad Close_t \\ &Function\ Outputs: \\ &\quad StdDev_t \end{aligned} \qquad (104)$$

Figure A20.1 illustrates the standard deviation calculation; notice that as the volatility of the stock decreases, so too does the standard deviation. In this example, any sudden changes in price will result in a temporary spike in standard deviation, as observed in the price changes from $2 to $1 and back to $2. A continual decline or incline leads to a higher standard deviation with limited variation. From these observations, it is easy to see that generally the standard deviation indicator gives a good account of what has happened with little forward predictive ability.

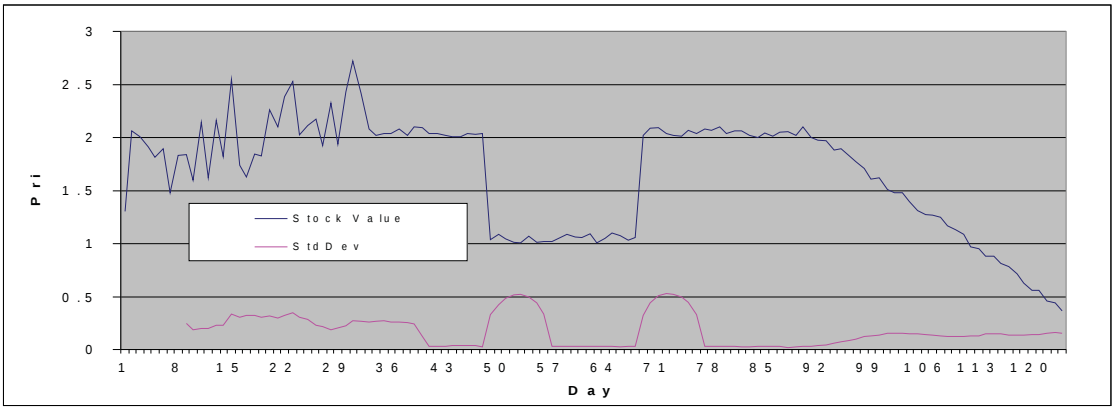

Figure A20.1: : Standard Deviation Function Output

A21. Money Flow Index

Money Flow Index (MFI) is used to indicator the flow of money into or out of a stock. Using a 0 – 100 scale and often calculated with a 14 day period and being volume

weighted considered a good indicator of the strength of money flow into or out of a stock (LIM, 2002). When the MFI increases to to above 80 it is considered an indication that the stock price will likely drop (a bearish signal); conversely a MFI of below 20 implies the opposite (a bullish signal).

To understand Money Flow Index, consider a situation where a stock's price and the money flow index are rising and falling respectively. In this situation, the flow of negative money is greater than that of positive money, implying that more money is associated with daily price falls than rises; this will ultimately lead to a reversal - the stock will begin to rise.

$$MoneyFlowIndex = 100 - \left(\frac{100}{1 + Money\ Ratio}\right) \quad (105)$$

$$MoneyRatio = \frac{PositiveMoney_t}{NegativeMoney_t} \quad (106)$$

MoneyRatio is a function of the so-called Money Ratio, a daily intra-day average calculated by averaging the daily close, high and low and multiplying the result by the daily volume (Equation 107).

$$MoneyFlow_t = \left(\frac{Close_t + High_t + Low_t}{3}\right) \times Volume_t \quad (107)$$

When the inter-day *MoneyFlow* drops it is considered Negative Money (Equations 108 and 109); conversely a Positive Day occurs when the inter-day *MoneyFlow* increases (Equations 110 and 111).

$$PostiveMoney_t = \sum_{i=t}^{t-13} PositiveDay_t \quad (108)$$

$$PositiveDay_t = \begin{bmatrix} MoneyFlow_t & , & MoneyFlow_t > MoneyFlow_{t-1} \\ 0 & , & otherwise \end{bmatrix} \quad (109)$$

And similarly calculated for Negative Money.

$$NegativeMoney_t = \sum_{i=t}^{t-13} NegativeDay_t \quad (110)$$

$$NegativeDay_t = \begin{bmatrix} MoneyFlow_t & , & MoneyFlow_t \le MoneyFlow_{t-1} \\ 0 & , & otherwise \end{bmatrix} \quad (111)$$

The function inputs and outputs for the Matlab implementation of the Money Flow Index are given in Equation 112. Note that the daily close is given as an output to the function as this is a necessary comparative element for interpretation of Money Flow Index. Bearish and Bullish interpretation and signal generation of the MFI was not carried out on the data.

$$\begin{aligned} &Function\ Inputs: \\ &\quad Open_t \\ &\quad High_t \\ &\quad Close_t \\ &\quad Volume_t \\ &Function\ Outputs: \\ &\quad MoneyFlowIndex_t \end{aligned} \quad (112)$$

A22. Chaikin Oscillator

The Chaikin indicator attempts to indicate the degree of buying or selling pressure through the location of the close relative to the high and low for the period (Chaikin, 1994). The Chaikin Oscillator is calculated by taking the difference between the 3 day and 10 day exponential moving average of the ADL line, or more simply applying MACD to the ADL line.

$$\begin{aligned} ChaikinOscillator_t = &ExponendialMovingAverage\left(adl_t, adl_{t-1}, \ldots adl_{t-9}\right) \\ &- ExponendialMovingAverage\left(adl_t, adl_{t-1}, adl_{t-2}\right) \end{aligned} \quad (113)$$

where adl is the accumulation distribution line (Equation 65).

The function inputs and outputs are given in Equation 114.

$$\begin{aligned} &Function\ Inputs: \\ &\quad High_t \\ &\quad Close_t \\ &\quad Volume_t \\ &Function\ Outputs: \\ &\quad ChaikinOscillator_t \end{aligned} \quad (114)$$

The basic premise behind the Chaikin Oscillator is that the current buying or selling pressure can be predicted by the relative positioning of the current high and low to the close.

A23. Rate of Change

The Rate Of Change (ROC) indicator is a simple momentum indicator that measure the rate of change of a stocks closing values (Schwager, 1999). This is simply calculated by taking a moving difference of a stock's current close with its close a number of periods ago (typically 10 days). This simple formula is a simple moving comparison of an equities price compared to ten periods ago and can be used to assess a stocks momentum.

The Rate of Change oscillator can be calculated as follows:

$$ROC_t = \frac{Close_t - Close_{t-9}}{Close_{t-9}} \times 100 \quad (115)$$

$$\begin{aligned} &Function\,Inputs: \\ &\quad Close_t \\ &Function\,Outputs: \\ &\quad ROC_t \end{aligned} \quad (116)$$

A24. Money Envelope Analysis

Money Envelope Analysis indicates when a stock exceeds or drops below a moving maximum or minimum limit respectively. These limits are calculated as a percentage of the current stock price; in this case 3% was selected. Envelope boundaries are calculated as follows:

$$\begin{aligned} &Function\,Inputs: \\ &\quad Close_t \\ &Function\,Outputs: \\ &\quad MEA_t \end{aligned} \quad (117)$$

Sensitivity simply sets the size of the envelope around the stock.

$$EnvelopeSensativity = 0.03 \quad (118)$$

$$MovingAverage_t = \frac{\sum_{i=t}^{15}(Close_i)}{15} \quad (119)$$

$$UpperEnvelope_t = MovingAverage_t \times (1 + EnvelopeSensativity) \quad (120)$$

$$LowerEnvelope_t = MovingAverage_t \times (1 - EnvelopeSensativity) \quad (121)$$

$$MEA_t = \begin{bmatrix} \left|\left(\frac{Close_t - UpperEnvelope_t}{UpperEnvelope_t}\right)\right| & , \quad Close_t > UpperEnvelope_t \\ -1 \times \left|\left(\frac{Close_t - LowerEnvelope_t}{LowerEnvelope_t}\right)\right| & , \quad Close_t > LowerEnvelope_t \\ 0 & , \quad otherwise \end{bmatrix} \quad (122)$$

The Money Envelope Analysis function was essentially implemented such that, when the stock price is within the ±3% moving average envelope, the out put is zero; When the stock price moves above or below the upper and lower envelope, the function output is the percentage difference between the price and the corresponding envelope respectively (Figure A24.1).

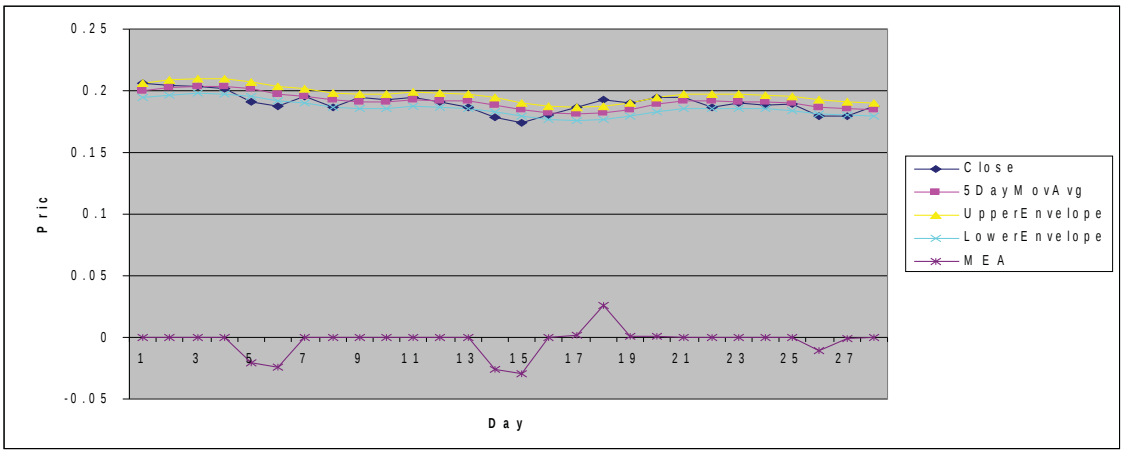

Figure A24.1: : Money Envelope Analysis