An investigation of artificial neural networks applied to pairs trading in South Africa

by

Richard Otto Buhr

Submitted in partial fulfillment of the requirements for the degree

Magister Scientiae

in the Department of Mathematics and Applied Mathematics in the Faculty of Natural and Agricultural Sciences

University of Pretoria
Pretoria

June 2014

Abstract

Author Richard Otto Buhr

Supervisor Prof. E. Maré

Submitted April 2014

Qualification Magister Scientiae

The use of artificial intelligence models could enhance the process by which decisions are made for trading in the financial markets. In this project the aim was to define, create, apply and analyse the results of an artificial neural network based solution for so-called pairs trading.

A literature study is presented where a brief summary is given of the current body of knowledge with regards to artificial neural networks and their application in finance. Additionally an introduction of the software packages used in this project is given. In a separate chapter a short introduction on trading strategies and pairs trading, specifically, is included.

The design of the artificial neural network is discussed in detail with both training and execution results from experiments critically examined.

Finally the question 'Are artificial neural networks a viable tool applied to pairs trading in the South African market?' is answered based on the empirical evidence.

Acknowledgement

I would like to acknowledge and thank Prof. E. Maré for his invaluable support, advice and guidance during the completion of this dissertation.

Declaration

I, the undersigned, declare that the dissertation, which I hereby submit for the degree Magister Scientiae at the University of Pretoria, is my own work and has not previously been submitted by me for a degree at this or any other tertiary institution.

Signed:

Richard Otto Buhr

2014/06/05

Contents

List of Figures

List of Tables

List of Symbols

α ANN learning rate.

δ Indicates the rate of change of a variable.

σ Squashing function.

Θ Activation threshold level.

a Activation level.

d An output node's influence on the overall ANN error.

f A function.

g A function.

i An integer used for iteration.

K Activation function.

N An integer indicating a number of dimensions.

n An output node.

p An element of the training set.

s Squashing function.

t Target or true classification, can also indicate time.

w Weighting factor, can be a vector of weights.

x Input value or a vector of input values.

y Output value or a vector of output values.

List of Acronyms

ALSI All Share Index.

ANN Artificial Neural Network.

BESA Bond Exchange of South Africa.

FANN Fast Artificial Neural Network.

FX Foreign Exchange.

JIBAR Johannesburg Interbank Agreed Rate.

JSE Johannesburg Stock Exchange.

NN Neural Network.

PCA Principal Components Analysis.

TLU Transaction Logic Unit.

o

Glossary of Terms

Artificial Neural Network An artificial construct to mathematically model how the human brain works in a simplistic fashion.

Bloomberg A news and information system commonly used in the financial industry as a source of historical and real-time market data.

Bond A bond is a fixed income instrument. Bonds are issued by governments, municipalities and companies to raise cash money.

Buy The acquisition of one or a number of units of a financial instrument.

Deposit A deposit is an instrument that describes the interest rate earned if a sum of money were deposited in a bank.

Derivative A financial instrument that is contingent on some other financial instrument, the underlying.

Equity An equity inthe context of this dissertation, also known as a common stock or a common share, is a financial instrument that indicates a shareholding in a company. It has a value indicating the value of a fraction of a company's ownership.

Exposure The number of units or snapshot monetary equivalent of a position in a financial instrument, very similar to position. Mostly used in a risk measurement context.

Fixed Income Fixed income is a class of financial instruments that is interest rate based.

Financial Engineering Financial engineering is a multidisciplinary field involving financial theory, the methods of engineering, the tools of mathematics and the practice of programming. It has also been defined as the application of technical methods, especially from mathematical finance and computational finance, in the practice of finance.

Hold Maintaining a stable position in a financial instrument, i.e. no changes are made.

Index A number produced by using a weighted sum of constituent input values, for example the ALSI Index is made was made up of a weighted sum of 163 equity shares at the time of data collection for this work.

Instrument An entity in the financial markets that has some measurable metric to it, for example an interest rate, a bond, share or derivative.

Long To go long a share means to buy into that share and take a position. Being long a share means you own some of those shares, i.e. you have a positive position in and positive exposure to that share.

Long Position A long position indicates one owns that position.

Neural Network A shorted version of Artificial Neural Network.

Neuron Similar to a TLU it is a simplified model of a biological neuron as found in the brain.

Option A derivative that gives the right, but not the obligation, to exercise some predetermined financial transaction.

Pairs Trading A trading strategy where the trader is long one of two correlated shares and short the other, hoping to achieve some profit by entering the long-short position prior to a period where their correlation weakens in the market and selling off the position when their prices have diverged sufficiently.

Position Number of instrument units held or the equivalent monetary value at a specified time (which is also known as marked-to-market).

Price The market sourced price of a financial instrument, i.e. a share, bond or derivative.

Sell The relinquishing of one or a number of units of a financial instrument.

Share See Equity and Stock.

Short A 'short' can indicate either a short sale or short position. To go short a share means to borrow that share and take a negative position.

Short Position A short position indicates one owes that position to another party.

Spread In finance it generally means the difference between two related prices or other financial variables which is often used in trading activities such as a 'spread trade'.

Stock See Equity and Share.

TLU A Transaction Logic Unit is also known as a neuron of an artificial neural network. Its main function is to produce an output value based on given input values and its transfer function.

Underlying An underlying is a financial instrument which provides a basis for another financial instrument, for example and equity option is based on an equity share.

Value The equivalent monetary value of a position at a specified time (which is also known as marked-to-market).

Note

More detailed descriptions of financial instruments are given in Appendix A.

Chapter 1

Introduction

1.1 Overview

The purpose of this research was to investigate the use of artificial neural networks in financial engineering. As a practical application it was decided to develop and test artificial neural networks as applied to pairs trading and critically analyse the results.

Over the past few decades a lot of research has been done on so-called artificial intelligence, [1]. One form of artificial intelligence that allows for machine learning is the artificial neural network. The use of artificial neural networks in various industries has become quite widespread, with a large amount of research into financial applications; [2, 3, 4, 5] and [6] being a few of many.

This research topic was chosen as there is a lot of manual work currently involved in many different parts of financial engineering. Specifically the management of different types of portfolio trading strategies seem to comprise a lot of repetitive analysis and it is in this area that artificial neural networks can make a big difference.

By applying machine learning concepts to trading strategies, the management of complex trading strategies could possibly be done more efficiently than current methods by using an artificial neural network method for decision making support.

1.2 Executive Summary

The use of artificial intelligence models could enhance the process by which decisions are made for trading. In this project the aim was to define, create, apply and analyse the results of an artificial neural network based solution for pairs trading. This document's main sections are briefly discussed below.

In Chapter 2 a literature study is presented where the current body of knowledge with regards to artificial neural networks and their application in finance is briefly summarised. Additionally a discussion on the choice of software packages used in this research is included. Chapter 3 contains a high level introduction of trading strategies with a more detailed look at pairs trading.

Chapters 4 and 5 present the approach by which the data was prepared and the artificial neural network was designed for this application. Various explanations of the design principles and the choices made for this research are covered. While many financial artificial neural network applications use few unique data series during training, this research used a 'big bang' approach by using 165 different data series to attempt to distil salient information during training.

The results of the artificial neural networks applied to long position management and pairs trading experiments are presented and discussed in detail in Chapter 6. Both numerical and graphical results are included.

The crux of this research is discussed in Chapter 7, where the answer to the question 'Are artificial neural networks a viable tool applied to pairs trading in the South Africa market?' is given based on the empirical evidence. The experimental outcomes and possible future work based on it are considered.

The bibliography was compiled using the IEEE standard, with cited works appearing in order of appearance in the main document. In Appendix A the financial instruments mentioned in this dissertation are defined with some discussion on their respective attributes. Appendix B contains an overview of software used during this research. Appendix C lists the market variables taken into consideration used in the experiments with the source code developed for and used in this dissertation given in Appendix D.

All data used and software developed for this research is included in an attached compact disc.

1.3 List of Objectives

Chapter	Title	Objective
Chapter 1	Introduction	Present the research objectives and layout of this dissertation.
Chapter 2	Literature Study	Provide an overview of artificial neural networks and some of their applications in finance. Discuss the choice of software used in this dissertation.
Chapter 3	Trading Strategies	Introduce trading strategies with a more detailed look at pairs trading.
Chapter 4	Methodology	Present the methodology choices that were made in choice of data, preparing the data for the experiments and how the training and testing historical periods were defined are discussed.
Chapter 5	Design and Training	Provide the design of the artificial neural networks used in the experiments and discuss the results of the ANN training.
Chapter 6	Results	Present the results of the artificial neural networks applied to long position management and pairs trading experiments are presented and discussed.
Chapter 7	Conclusion	Conclude this dissertation with a discussion of the work done during this research, the experimental results, potential shortcomings detected and possible future work.
Appendix A	Financial Instruments	Provide an overview of the financial instruments mentioned or used in this research.
Appendix B	Software	Review the software used during this research.
Appendix C	Market Variables	List all market variables considered during the experiments.
Appendix D	Source Code	Present the source code developed during this research.

Table 1.1: Chapter objectives of this dissertation.

Chapter 2

Literature Study

2.1 Introduction

This chapter presents some basic theory of artificial neural networks and their application in finance. Information outside of the scope of this chapter is included in the appendices, for example, what software was used in this research.

2.2 Artificial Neural Networks

2.2.1 Origins

The original inspiration for the term Artificial Neural Network came from examination of central nervous systems and their neurons, axons, dendrites, and synapses, which constitute the processing elements of biological neural networks investigated by neuroscience. In an artificial neural network, simple artificial nodes, variously called 'perceptrons', 'neurons', 'neurodes' or 'processing elements' (PEs), are connected together to form a network of nodes mimicking the biological neural networks - hence the term 'artificial neural network', [7, 8, 9, 10, 11] and [12].

Because neuroscience is still full of unanswered questions, and since there are many levels of abstraction and therefore many ways to take inspiration from the brain, there is no single formal definition of what an artificial neural network is. Generally, it involves a network of simple processing elements that exhibit complex global behaviour determined by connections between processing elements and element parameters. While an artificial neural network does not have to be adaptive, per se, its practical use comes with algorithms designed to alter the

strength (weights) of the connections in the network to produce a desired signal flow, [9, 11] and [12].

According to a simplified account [13], the human brain consists of about ten billion neurons and a neuron is, on average, connected to several thousand other neurons. By way of these connections, neurons both send and receive varying quantities of energy. One very important feature of neurons is that they don't react immediately to the reception of energy. Instead, they sum their received energies, and they send their own quantities of energy to other neurons only when this sum has reached a certain critical threshold. The brain learns by adjusting the number and strength of these connections. Even though this picture is a simplification of the biological facts, it is sufficiently powerful to serve as a model for the artificial neural network, [9, 11] and [13].

An artificial neural network (ANN), sometimes called a neural network (NN), is a mathematical model or computational model that is inspired by the structure and/or functional aspects of biological neural networks. A neural network consists of an interconnected group of artificial neurons, and it processes information using a connectionist approach to computation. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase. Modern neural networks are non-linear statistical data modelling tools. They are usually used to model complex relationships between inputs and outputs or to find patterns in data, [9, 10, 11, 12] and [14].

2.2.2 Types of ANN

There are many types of artificial neural network (ANN) models in existence today.These models mimic the real life behaviour of neurons and the electrical messages they produce between input, processing by the brain and the final output from the brain. The main types of ANN to be found in most literature are listed below.

Feedforward neural network The feedforward neural network was the first and arguably most simple type of artificial neural network devised. In this network the information moves in only one direction: forwards. From the input nodes data goes through the hidden nodes (if any) and on to the output nodes, there are no cycles or loops in the network. Feedforward networks can be constructed from different types of units, the simplest being the perceptron. Continuous neurons, frequently with sigmoidal activation, are used in the context of backpropagation of error, [15, 16, 17, 18, 19].

Feedforward neural networks have been used in approximation of polynomials [20], forecasting exchange rates [21], non-linear principal component analysis [22], facial expression recognition [23] and motion control of autonomous underwater vehicles [24], amongst many other practical uses.

Radial basis function (RBF) network Radial basis functions are powerful techniques for interpolation in multidimensional space. Radial basis functions have been applied in the area of neural networks where they may be used as a replacement for the sigmoidal hidden layer transfer characteristic in multi-layer perceptrons. RBF networks have the advantage of not suffering from local minima in the same way as Multi-Layer Perceptrons. RBF networks have the disadvantage of requiring good coverage of the input space by radial basis functions, [15, 25].

Radial basis functions have been used for solving partial differential equations [26], a clustering technique for digital communications [27], as a numerical method for heat transfer problems [28] and interpolating implicit surfaces from scattered surface data [29].

Kohonen self-organizing network The self-organizing map (SOM) invented by Teuvo Kohonen performs a form of unsupervised learning. A set of artificial neurons learn to map points in an input space to coordinates in an output space. The input space can have different dimensions and topology from the output space, and the SOM will attempt to preserve these, [15, 25, 16, 17, 19].

SOMs have been used in feature recognition in 3D data [30], interpreting patterns of gene expression [31] and data exploration [32].

Hopfield network The Hopfield network (like similar attractor-based networks) is of historic interest although it is not a general recurrent ANN, as it is not designed to process sequences of patterns; instead it requires stationary inputs. It is an recurrent ANN in which all connections are symmetric. Invented by John Hopfield in 1982 it guarantees that its dynamics will converge. If the connections are trained using Hebbian learning then the Hopfield network can perform as robust content-addressable memory, resistant to connection alteration, [15, 16, 17, 18, 19].

Some applications of Hopfield networks include colour image segmentation [33], identification and control of dynamical systems [34] and image restoration [35].

Boltzmann machine The Boltzmann machine can be thought of as a noisy Hopfield network. Invented by Geoff Hinton and Terry Sejnowski in 1985, the Boltzmann machine is important because it is one of the first neural networks to demonstrate learning of latent variables (hidden units), [15, 17, 19].

Boltzman machines have been used in classification problems [36], simulated annealing and combinatorial optimization [37] and fault section estimation in power systems [38] amongst a number of other applications.

In this research the feed-forward back-propagation ANN was used. The reasoning was as follows: it is the simpler of the available models, the most flexible in terms of suitability to a wide range of applications and by far the most widely used ANN architecture, [39, 2]. Another important consideration was that this ANN model is well supported by the software that was deemed most suitable for use in this research (see Section 2.3). Finally, the author's previous ANN related work predisposed him to using a feed-forward back-propagation ANN, [30].

2.2.3 Basic Model

Neural network models in artificial intelligence are usually referred to as artificial neural networks (ANNs); these are essentially simple mathematical models defining a function $f : xy$ or a distribution over x or both x and y, but sometimes models are also intimately associated with a particular learning algorithm or learning rule. A common use of the phrase ANN model really means the definition of a class of such functions (where members of the class are obtained by varying parameters, connection weights, or specifics of the architecture such as the number of neurons or their connectivity), [9, 12] and [40].

The word network in the term 'artificial neural network' refers to the interconnections between the neurons in the different layers of each system, [40]. An example system has three layers; the first layer has input neurons, which send data via synapses to the second layer of neurons, and then via more synapses to the third layer of output neurons. More complex systems will have more layers of neurons with some having increased layers of input neurons and output neurons, [2] and [40]. The synapses store parameters called "weights" that manipulate the data in the calculations, [11, 12] and [40].

An ANN is typically defined by three types of parameters, [9, 11, 12, 14] and [40]:

1. The interconnection pattern between different layers of neurons;

2. The learning process for updating the weights of the interconnections;

3. The activation function that converts a neuron's weighted input to its output activation.

Mathematically, a neuron's network function $f(x)$ is defined as a composition of other functions $g_i(x)$, which can further be defined as a composition of other functions. This can be conveniently represented as a network structure, with arrows depicting the dependencies between variables. A widely used type of composition is the non-linear weighted sum, where $f(x) = K\left(\sum_i w_i g_i(x)\right)$, where K (commonly referred to as the activation function) is some predefined function, such as the hyperbolic tangent. It will be convenient for the following to refer to a collection of functions g_i as simply a vector $g = (g_1, g_2, \ldots, g_n)$, [9, 11, 13, 14] and [40].

The first step toward understanding artificial neural networks is to abstract from the biological neuron, and to focus on its character as a threshold logic unit (TLU). A TLU is an object that inputs an array of weighted quantities, sums them, and if this sum meets or surpasses some threshold, outputs a quantity, [41]. Let the inputs and their respective weights be: $x_1, x_2, \ldots, x_n$ and $w_1, w_2, \ldots, w_n$. Then, there are the $x_i w_i$ that are summed, which yields the activation level a, in other words:

$$a = (x_1 w_1) + (x_2 w_2) + \ldots + (x_i w_i) + \ldots + (x_n w_n). \tag{2.1}$$

The threshold is called θ (theta). Lastly, there is the output: y; when:

$$y = \begin{cases} 1, & \text{if } a \geq \theta; \\ 0, & \text{if } a < \theta. \end{cases} \tag{2.2}$$

Notice that the output does not have to be discontinuous, since it could also be determined by a squashing function, σ (or sigma), whose argument is a, and whose value is between 0 and 1. Then, $y = s(a)$, [9] and [13]. One of the main strengths of the TLU (sometimes called a perceptron, though a more accurate definition is given later) model is that it can classify different input data to produce output data as necessary, [42]. For instance, a TLU that has two inputs, with weights equal to 1, and $\theta = 1.5$. When this TLU inputs [0,0], [0,1], [1,0], and [1,1], it outputs 0, 0, 0, and 1 respectively. This TLU classifies these inputs into two groups: the 1 group and the 0 group. Insofar as a human brain that knows about logical conjunction (Boolean AND) would similarly classify logically conjoined sentences, this TLU exhibits something like logical conjunction, [13].

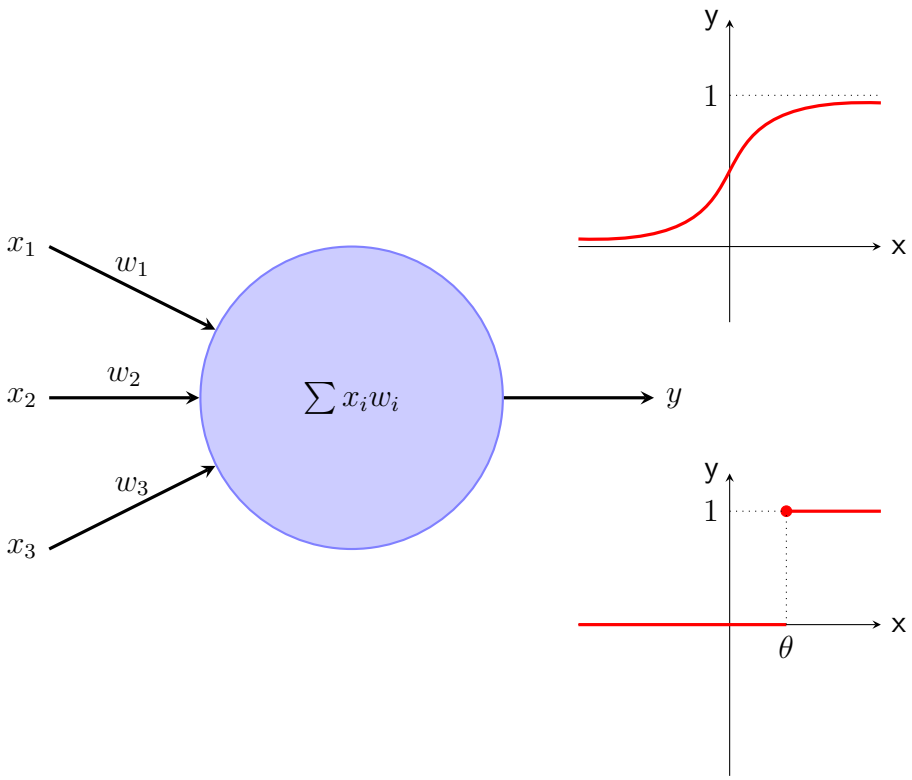

Figure 2.1: A graphical representation of a neuron model, also known as a threshold logic unit (TLU). The σ function is shown at the top right and the cut-off function at the bottom right.

This TLU has a geometric interpretation that clarifies what is happening. Its four possible inputs correspond to four points on a Cartesian graph. From the point where the classification behaviour switches:

$$x_1 w_1 + x_2 w_2 = \theta, \tag{2.3}$$

it follows that

$$x_2 = -x_1 + 1.5. \tag{2.4}$$

The graph of this equation cuts the four possible inputs into two spaces that correspond to the TLU's classifications. For a TLU with an arbitrary number of inputs, N, the set of possible inputs corresponds to a set of points in N-dimensional space. If these points can be cut by a hyperplane, an N-dimensional geometric figure corresponding to the line in the above example in other words, then there is a set of weights and a threshold that define a TLU whose classifications match this cut, [9] and [13].

Since TLUs can classify they can 'memorize' information. Artificial neural networks can thus 'learn'. Their learning mechanism is modelled on the brain's adjustments of its neural connections, [13]. A TLU learns by changing its weights and threshold, [9]. The critical point at which a TLU outputs 1 instead of 0 is when:

$$\sum i(x_i w_i) \leq \theta. \tag{2.5}$$

This is equivalent to saying that the critical point is when:

$$\sum i(x_i w_i) + (-1 \cdot \theta) \leq 0. \tag{2.6}$$

It is thus possible to treat -1 as a constant input whose weight (θ), is adjusted in learning (or training). In this case, [13]:

$$y = \begin{cases} 1, & \text{if } \sum i(x_i w_i) + (-1 \cdot \theta) >= 0; \\ 0, & \text{otherwise.} \end{cases} \tag{2.7}$$

During training a neural net receives the following inputs:

- A series of examples of the items to be classified.
- Their proper classifications or targets.

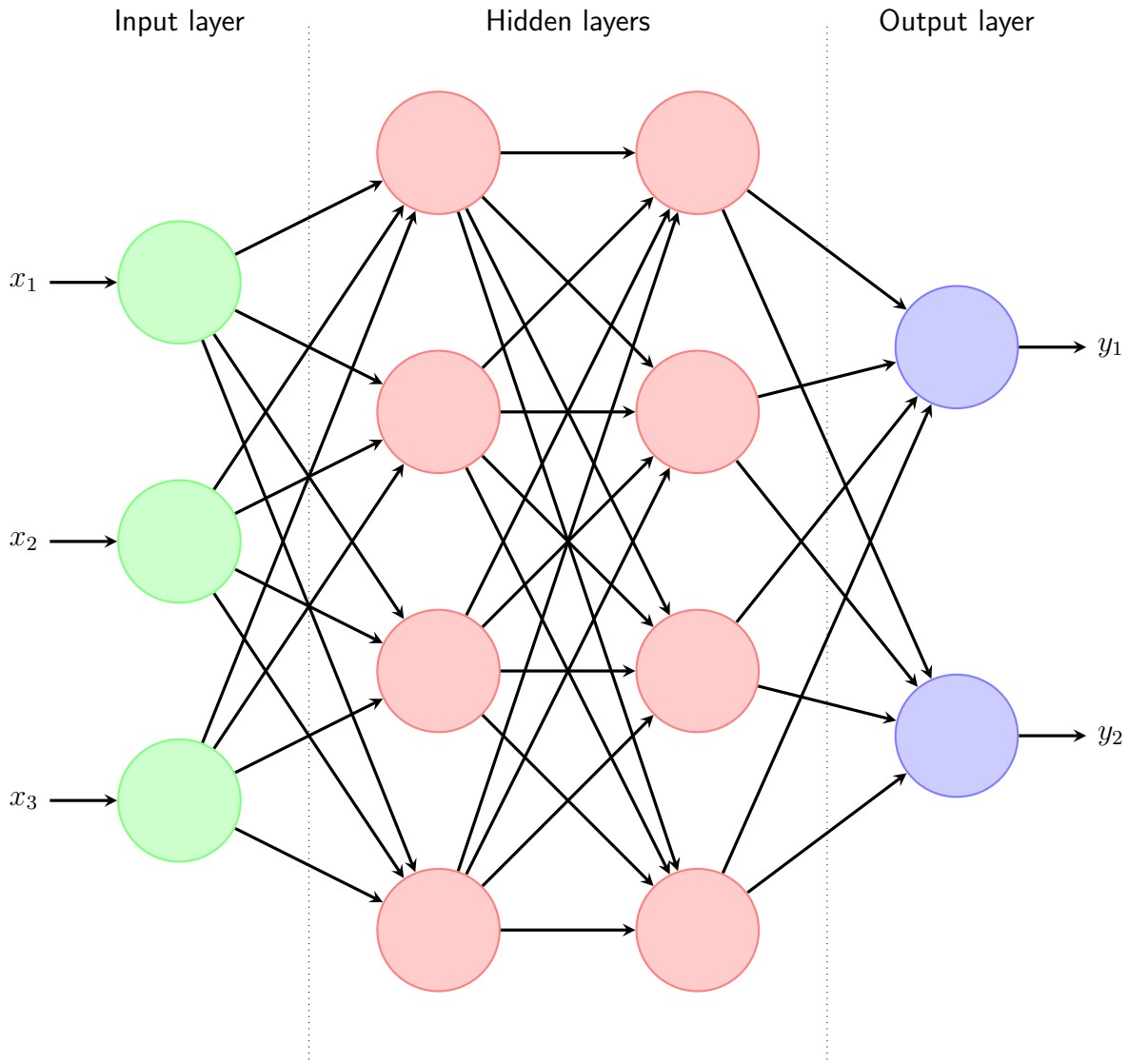

Figure 2.2: A graphical representation of an artificial neural network model.

These inputs can be viewed as a vector:

$$[x_1, x_2, \ldots, x_n, \theta, t], \tag{2.8}$$

where t is the target or true classification. The artificial neural network uses these inputs to modify its weights during training. The aim is to match its classifications with the targets in the training set as accurately as possible. This is known as supervised training, with examples accompanied by targets, whereas unsupervised training is based on statistical analysis, [2, 9] and [13].

2.2.4 Training

The Perceptron Learning Rule

The modification of weights during the training phase of an artificial neural network follows a prescribed learning rule, [43]. While there are many available training rules, one popular rule is based on the idea that weight and threshold modification should be determined by a fraction of $t - y$. This is accomplished by introducing $\alpha(0 < \alpha < 1)$, which is known as the learning rate. The change in a weighting is given by:

$$w_i = \alpha x_i (t - y). \tag{2.9}$$

When α is close to zero, the artificial neural network will engage in more conservative weight modifications, and when it is close to 1, it will make more radical weight modifications. An ANN that uses this rule is known as a perceptron, and this rule is called the perceptron learning rule. One result about perceptrons is that if a set of points in N-space is cut by a hyperplane, then the application of the perceptron training algorithm will eventually result in a weight distribution that defines a TLU whose hyperplane makes the wanted cut, [13, 40] and [44].

The Delta Rule

Another widely used training rule is the delta rule, [25]. The perceptron training rule is based on the idea that weight modification is best determined by some fraction of the difference between target and output, [45]. The delta rule is based on the idea of gradient descent, [9, 43] and [13]. This means that the change to the weight is proportional to its error, [9] and [46]. It is in essence used to minimize the sum of the mean square error, i.e. minimize the sum of deviation between the actual network output and the expected output, [2].

Let p be an element in a training set, and $t(p,n)$ be the corresponding target of output node n. Let $y(p,n)$ be determined by the squashing function s, where $a(p,n)$ is n's activation relative to p, $y(p,n) = s(a(p,n))$ or the squashed activation of node n relative to p. Setting the weights (w_i) for an artificial neural network also sets the difference between $t(p,n)$ and $y(p,n)$ for every p and n, and this means setting the network's overall error for every p. For any set of weights, there is therefore an average error, [13].

Backpropagation

Backpropagation is an algorithm that extends the analysis that underpins the delta rule to artificial neural networks with hidden nodes, [2]. This is a class of ANNs that have supervised learning rules, [10]. It is also based on the idea of gradient descent and the only change in the analysis of weight modification concerns the difference between $t(p,n)$ and $y(p,n)$, [9] and [13]. Generally, the change to w_i is:

$$\delta w_i = \alpha s'(a(p,n))d(n)x(p,i,n), \tag{2.10}$$

where $d(n)$ for a hidden node n, hinges on:

1. How much n influences any given output node;
2. How much that output node itself influences the overall error of the net.

The more n influences an output node, the more n affects the ANN's overall error. Additionally, if the output node influences the overall error less, then n's influence correspondingly diminishes. Where $d(j)$ is output node j's contribution to the ANN's overall error and $w(n,j)$ is the influence that n has on j, $d(j)w(n,j)$ is the combination of these two influences. However, n almost always influences more than one output node and it may influence every output node, [9] and [13]. So, $d(n)$ is:

$$\sum(d(j)w(n,j)), \forall j, \tag{2.11}$$

where j is an output node that takes input from n. Combining this provides a training rule. The first part: the weight change between hidden and output nodes, n and j, is:

$$\alpha s'(a(p,n))(t(p,n) - y(p,n))x(p,n,j). \tag{2.12}$$

The second part: the weight change between input and output nodes, i and n, is:

$$\alpha s'(a(p,n)) \sum(d(j)w(n,j))x(p,i,n), \tag{2.13}$$

where j varies over all the output nodes that receive input from n. The basic outline of a backpropagation algorithm is shown below, [13] and [44].

Step 1 Input training vector.

Step 2 Hidden nodes calculate their outputs.

Step 3 Output nodes calculate their outputs on the basis of Step 2.

Step 4 Calculate the differences between the results of Step 3 and targets.

Step 5 Apply the first part of the training rule using the results of Step 4.

Step 6 For each hidden node, n, calculate d(n).

Step 7 Apply the second part of the training rule using the results of Step 6.

Steps 1 through 3 are often called the forward pass, and steps 4 through 7 are often called the backward pass. Hence, the name: backpropagation, [9] and [13]. Backpropagation ANNs are the most commonly used, with over 80% of all applications using this design, [10].

2.2.5 Artificial Neural Networks in Finance

The best overview of artificial neural networks in finance that was discovered by the author of this document was in a book by A-P Refenes titled Neural Networks in the Capital Markets, [1]. Upon doing further research on this book it was also found that this one of, if not the, most highly praised books on the topic available.

The prevailing wisdom among financial economists is that price fluctuations not due to external influences are dominated by noise and can be modelled by stochastic processes. It is possible that these remaining price fluctuations could be due to non-linear processes at work in the marketplace. Given appropriate tools it is possible to understand much of the market's price structure on the basis of completely or partially deterministic but non-linear dynamics, [1].

Neural networks are a field of research which has enjoyed rapid expansion and great popularity in both the academic and industrial research communities, [47]. Neural networks are essentially statistical devices for performing inductive inference. From a statistician's point of view they are analogous to non-parametric, non-linear regression models. The novelty of neural networks lies in their ability to model non-linear processes with few a priori assumptions about the nature of the generating process, [14]. This is particularly useful in financial engineering

applications, where much is assumed and little is known about the nature of the processes determining asset prices, [1]. Kuo and Reitsch, [48], mention that ANNs provide meaningful predictions when independent variables are correlated or missing, while Denton, [49], puts forth that ANNs tend to outperform conventional regression analysis in the presence of ambiguities in independent variables. McClelland and Rumelhart, [50], also mention that ANNs can learn associations between input variables and can associate patterns, allowing them to generalise.

The case for the existence of non-linear dependencies in the context of financial markets can be made by a mix of observations on market micro-structures, feedback effects in market prices and empirical observations. Dissimilar micro-structures between asset markets and between spot and derivative markets, for example, could give rise to non-linear dependencies. Several financial economists have argued that price discovery takes place in the futures markets and the information is subsequently carried to the spot market through the process of arbitrage, [1]. Grudnitski and Osburn, [43], provide an application of artificial neural networks to financial futures price discovery.

Delays in transacting the stock market leg of the arbitrage imply that the immediate response in the mispricing would only be partial, reflecting the change in the future price alone. This may induce further arbitrage activity and could actually result in overshooting the arbitrage bounds. Moreover, restrictions on the short sales in the stock markets, for example, may lead to delays in executing arbitrage transactions and this may cause non-linear behaviour, [1].

Non-linear dependencies may also be explained in terms of non-linear feedback mechanisms in price movements alone. When the price of an asset becomes too high, self-regulating forces usually drive the price down. If the feedback mechanism is non-linear then the correction will not always be proportional to the amount by which the price deviates from the asset's real value. It is not unreasonable to expect such non-linear correction in the financial markets; they can be explained by the study of market psychology, where it is understood that investors and markets over-react to bad news and under-react to good news. There are many participants in the financial markets with complex motivations, reactions and interrelationships. Once non-linear feedback mechanisms were introduced in the market description, many price fluctuations could be explained without reference to stochastic effects, [1]. Masters, [51], goes on to add that neural networks are less sensitive to error term assumptions and they tolerate noise, chaotic components and heavy tails better than most other methods. Fadlalla and Lin, [2], also point out that ANNs are especially useful when data is non-linear, fuzzy or noisy.

Specifically, they refer to Lippman [52] who pointed out that ANNs 'provide a high degree of robustness and fault tolerance.'

The prevailing capital market model is based on the rationality of individual investors. It is assumed that investors are risk-averse, are unbiased when they set their subjective probabilities and always react to information as it is received. The implication is that the data-generating process is linear. In practice, however, investors may well be risk-seeking instead of risk-averse when, for example, taking gambles to minimise their losses. Moreover, they may have excessive faith in their own forecasts, thus introducing bias in their subjective probabilities, and may not react to information instantaneously but delay their response until their peers reveal their preferences. Therefore linear models may not be adequate in explaining market behaviour, [1]. Odom and Sharda, [53], discovered that ANNs predicted bankruptcy better than discriminant analysis.

Non-linear modelling techniques are the subject of increasing interest from market practitioners in quantitative asset management, with neural networks assuming a prominent role. Neural networks are being applied to a number of 'live' systems in financial engineering and have shown promising results. However, the development of successful applications with neural networks is not a straightforward task, [1].

The author of the above-mentioned book, A-P Refenes, goes on to include various applications of artificial neural networks to modelling stock returns, forecasting gold futures prices, modelling a stock market, forecasting exchange rates (also [54]) and applying other machine learning techniques such as genetic algorithms to foreign exchange trading, [47, 1]. Swanson and White, [41], apply artificial neural networks to forecasting future spot interest rates from forward rates. Kaastra and Boyd, [10], also list risk rating mortgages and fixed income investments, index construction and simulation, portfolio selection and diversification, identification of economic explanatory variables and economic forecasting as applications of artificial neural networks,citejsjgt:nnatfmpcpo. Zhang et al, [55], successfully applied ANNs to time series forecasting, while Van Eyden, [56], applied ANNs to forecasting security markets. There are many more applications of ANNs in finance and extensive lists are given by Johnson and Whinston [3] and Trippi and Turban [6].

Fadlalla and Lin, [2], performed an extensive review of financial applications of ANNs in 2001. Most of their findings are still perfectly applicable today and some of their results are

discussed below. As part of their investigation, they searched for key words 'neural networks' and 'finance' in journal and thesis abstracts using the Dialog database, which itself covers a number of research databases. The main database classifications covered were financial journals, thesis abstracts and technology. The rise of ANNs in research is shown in Table 2.1.

In order to categorize the application of ANNs in finance Fadlalla and Lin, [2], chose 40 representative papers that are frequently cited and whose authors went into detail on their specific applications. The distribution of these papers in terms of topic is shown in Table 2.2. Of these 40 articles the types of ANN training control strategy used are shown in Table 2.3 and their respective number of hidden layers used are shown in Table 2.4. In the case of Table 2.4 the total shown is larger than 40 because some researchers used multiple network topologies in their research.

2.3 Artificial Neural Network Software

There are a number of software packages available to perform artificial neural network modelling and implementation. Some of these packages are the Matlab Neural Network Toolbox [57], Stuttgart Neural Network Simulator [58], the SciPy and PyFANN combination (see Appendix B) and many more. PyFANN is the Python package that contains the bindings to the FANN C library.

While the Matlab Neural Network Toolbox and the Stuttgart Neural Network Simulator are excellent software packages for performing the experiments in this dissertation, the author elected to use Python [59], Scipy [60], and FANN [61] software. The reasoning was that while the author is proficient in both Matlab and Python, the general availability of open source software was preferable to the author. Hence the choice Python, PyFANN and FANN packages, which are available via the Ubuntu Linux [62] software package management tool.

2.4 Chapter Summary

This chapter has provided an overview of artificial neural networks and their applications in the financial world. It is clear that many minds have been applied to the use of ANNs in finance, with mixed results. The successes of some researchers are encouraging to future researchers of this topic. In the end of this chapter the reasoning behind using the specific software for modelling artifical neural networks in the dissertation was discussed.

Year	Financial Journals	Thesis Abstracts	Technology Journals	Column Total
1986	0	0	0	0
1987	0	0	3	3
1988	3	0	14	17
1989	2	0	40	42
1990	3	1	45	48
1991	20	3	61	84
1992	56	5	55	116
1993	73	6	76	155
1994	104	14	78	197
1995	124	18	43	185
1996	84	17	36	137
1997	90	13	27	130
Total	**559**	**77**	**478**	**1114**

Table 2.1: Annual search results for ANNs in literature from 1986 until 1997, showing a steadily increasing trend, [2].

Application Category	Number of Articles
Bankruptcy prediction	10
Stock market forecast	22
Credit analysis	4
Underwriting analysis	3
Business cycle recognition	1
Total	**40**

Table 2.2: Number of representative articles that were examined closely by Fadlalla and Lin on ANNs in Finance, [2].

Control Strategy	Number of Articles
Backpropagation	26
Recurrent	5
Unreported	9

Table 2.3: Number of representative articles that were examined closely by Fadlalla and Lin on ANNs in Finance, [2].

Hidden Layers	Number of Articles
0	2
1	29
2	7
Unreported	5

Table 2.4: Number of hidden layers used in the ANNs as per the 40 representative articles that were examined closely by Fadlalla and Lin on ANNs in Finance, [2].

Chapter 3

Trading Strategies

3.1 Introduction

This chapter contains a short overview of some trading strategies used in the financial markets, specifically pairs trading which can be part of algorithmic trading.

3.2 Trading Strategies

In finance, a trading strategy is a predefined set of rules for making trading decisions. Traders, investment firms and fund managers use a trading strategy to help make wiser investment decisions and help eliminate the emotional aspect of trading. A trading strategy is governed by a set of rules that do not deviate, [63]. Emotional bias is eliminated because the systems operate within the parameters known by the trader. The parameters can be trusted based on historical analysis (backtesting) and real world market studies (forward testing), so that the trader can have confidence in the strategy and its operating characteristics, [64].

When developing a trading strategy, many things must be considered: return, risk, volatility, timeframe, style, correlation with the markets, methods, etc. After developing a strategy, it can be backtested using computer programs. Although backtesting is no guarantee of future performance, it gives the trader confidence that the strategy has worked in the past. If the strategy is not over-optimized, data-mined, or based on random coincidences, it might have a good chance of working in the future, [64] and [65].

A trading strategy can be executed by a trader (manually) or automated (by computer).

Manual trading requires a great deal of skill and discipline, [66]. It is tempting for the trader to deviate from the strategy, which usually reduces its performance. An automated trading strategy wraps trading formulas into automated order and execution systems, [66]. Advanced computer modelling techniques, combined with electronic access to world market data and information, enable traders using a trading strategy to have a unique market vantage point. A trading strategy can automate all or part of your investment portfolio. Computer trading models can be adjusted for either conservative or aggressive trading styles, [64] and [66].

One of the advanced technologies that is discussed in a book by Ruggiero, [67], is the use of artificial neural networks in optimizing trading strategies.

3.3 Pairs Trading

Pairs trading, or Statistical Arbitrage, is a stock trading strategy that attempts to be market neutral and capture the spread between two correlated stocks as they return to the mean price. This is a form of a spread trade, [68] and [69]. It is known by some as 'statistical arbitrage', but 'pairs trading' is the more common name used to refer to this technique. Pair trading is sometimes also done with options, futures, and baskets of stocks, [70, 71, 72].

The pairs trading strategy developed in the late 1980s with the main assumption of the strategy being that two securities (or other assets), often competitors in the same sector, are correlated in their day-to-day price movements, [73]. When the correlation weakens, i.e. one stock moves up while the other moves down, the pairs trade would be to short the outperforming stock and to long the underperforming one, betting that the "spread" between the two would eventually converge again, [68] and [74].

Extensive research has been performed on the subject and a variety of authors have developed different ways of applying and analysing pairs trading strategies. Gatev, Goetzman and Rouwenhorst, [75], put forth using a returns based approach, for example, while Do and Faff [76, 77] propose selecting pairs from stocks in economically meaningful groups. This approach is also followed by a recent South African study of pairs trading on the JSE [78] by Mashele, Terblanche and Venter.

Simply stated, it is the buying and simultaneous selling of two stocks that follow each other when they diverge from the normal pattern; in the expectation that the normal pattern will

soon resume, [68] and [74]. Typically the trader would take a long position in the underperforming asset and an equal value short position in the outperforming asset, in essence funding one half of the trade with the other half, excluding trading costs, [72].

In Figure 3.1 the price histories of two correlated shares can be seen. Over time the 'spread' between the two price histories widens and eventually the two series converge again. This is indicative of the behaviour that provides a pairs trading opportunity. In this example one would buy (go long) Standard Bank (SBL SJ Equity) shares and sell (or short) ABSA (ASA SJ Equity) shares to an equal value, for arguments sake R10,000. This would mean buying R10,000 worth of Standard Bank shares and selling or owing R10,000 worth of ABSA shares.

3.4 Algorithmic Pairs Trading

Pairs trading is often conducted as an algorithmic trading strategy on some automated execution management system. The automated pairs trading strategy is typically built around models that define the spread between the two assets based on historical data mining and analysis. The algorithm monitors for deviations in price, automatically buying and selling to capitalize on market inefficiencies, [68, 79]. The advantage in terms of reaction time allows traders to take advantage of tighter spreads, [74, 79].

There is, unfortunately, immense operational risk attached to automated trading, which may more than offset the potential profit gained using this approach.

3.5 Artificial Neural Networks in Pairs Trading

There has been considerable work done on the use of artificial neural networks in various parts of quantitative finance, including pairs trading, [73]. Some practitioners have used artificial neural networks to match the shares used in a pairs trade from a basket of shares, then use a genetic algorithm to customise the actual trade (number of shares bought/sold etc.), [80].

NeuroShell (Ward Systems Group, Inc) is a company that sells software that uses artificial neural networks, genetic algorithms and fuzzy logic for building trading models. One of the trading models they purportedly can create and optimize with artificial neural networks is pairs trading, [81]. The author of the article referenced, [82], used artificial neural networks for predicting the closing prices of foreign exchange currency pairs in conjunction with principal

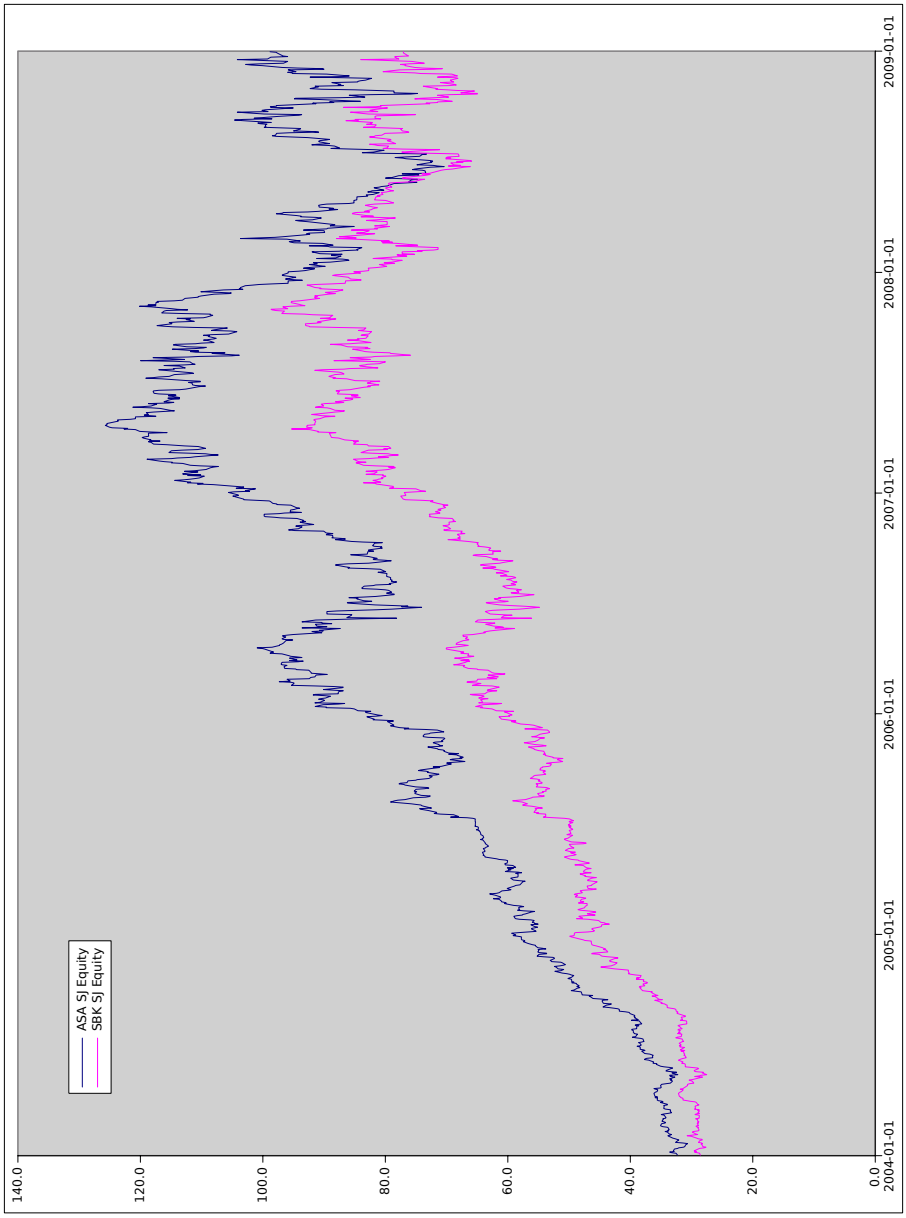

Figure 3.1: An example of two correlated shares where the 'spread' between the price histories widens over time and then converges again, indicating the behaviour that provides a profitable pairs trading opportunity.

components analysis (PCA).

A respected source of information for quantitative finance, the Wilmott forums, has a discussion on the use of artificial neural networks for use in pairs trading, though the general consensus in the discussion is that while artificial neural networks have their uses, they might not be entirely applicable to this task, [83]. They also mention that the specific model used by the author of the discussion might be unsuitable.

Chapter 4

Methodology

4.1 Introduction and Terminology

In this chapter the experimental data preparation and design of the trading strategies is discussed in detail. When discussing a financial instrument such as a currency, interest rate or share price it may be referred to as a risk factor. This is due to the fact that all these entities have inherent financial market risk.

In this document, when discussing a risk factor's class it means which broader type of risk factor it is, for example, whether it is an equity or an interest rate. When the risk factor type is used the meaning is whether the risk factor is a specific instrument, in other words while the JIBAR3M and the R157 are both of the interest rate class, the JIBAR3M is a deposit type instrument and the R157 is a bond type instrument, [84] and [85].

Another way to consider this is the XYZ:SJ equity share is of class equity and of type share. An option using XYZ:SJ share as an underlying would still be of class equity, but now of type option (which is also a financial derivative).

4.2 Trading Portfolio

The decision was made to perform the artificial neural network experiment in three stages. This purpose of this was to see how the ANN performed under different constraints. The three different trading scenarios were:

- Single share long trading position;

- Pairs trading in the Banking sector;
- Pairs trading in the Gold Mining sector.

The logic of each test case's portfolio management is discussed in the below sections. The purpose behind using different test cases was to train an artificial neural network to cater for each test case and see the capability of the respective ANNs in each case given that each case is subtly different.

4.2.1 Single Share Long Position

For the single share long position the criteria for portfolio management was to buy shares when the single share performed worse than the general market and hold the position while it performed better. The volume of shares to be bought was correlated to the level of disconnect in terms of market performance. This was measured by comparing the daily percentage moves of the designated share to the daily percentage move of the Johannesburg All Share Index (ALSI). Based on the comparative disparity an indicator was set up for 'Buy' and 'Hold', i.e. either 1 or 0. A second indicator was used for volume. This was set as an integer value of the number of designated shares to buy on that day. The fundamental principle of this strategy is to most efficiently (i.e. buy as low as possible) increase the long share investment over time and outperform the market (the ALSI market index).

For the single share position the stock chosen for analysis was OPT:SJ (for more information on Bloomberg tickers see Section B.1). This is the stock code for a company called Optimum Coal Holdings on the JSE. It was chosen for the single share analysis as it was deemed a fairly average share, not overly correlated with market movements but correlated enough such that the ANN would be able to use the other shares' information to attempt to predict OPT:SJ's movements. The actual correlation to the ALSI over the entire period described in Table 4.1 was 98.57%.

4.2.2 Pairs Trading Positions

A similar procedure to the single long share was followed for the pairs trading positions. One indicator was used to indicate being long the pairs position, when the indicator was 1 it ought to be owned and when it was 0 it ought not be owned. In other words it implied being in or out of the position at a given time. If the indicator was 0 and it became 1 then the pairs trade should be entered. For as long as the indicator remained at 1 the pairs trade was to be

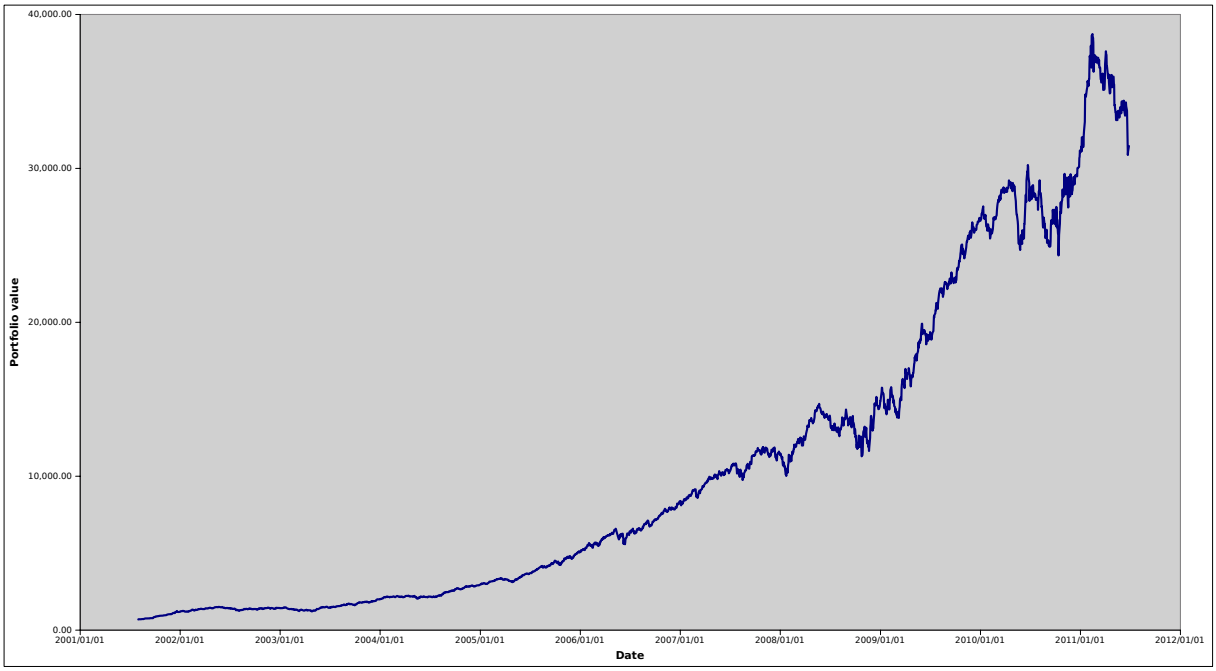

Figure 4.1: Portfolio value change over the duration of the Single Long Share Position training data.

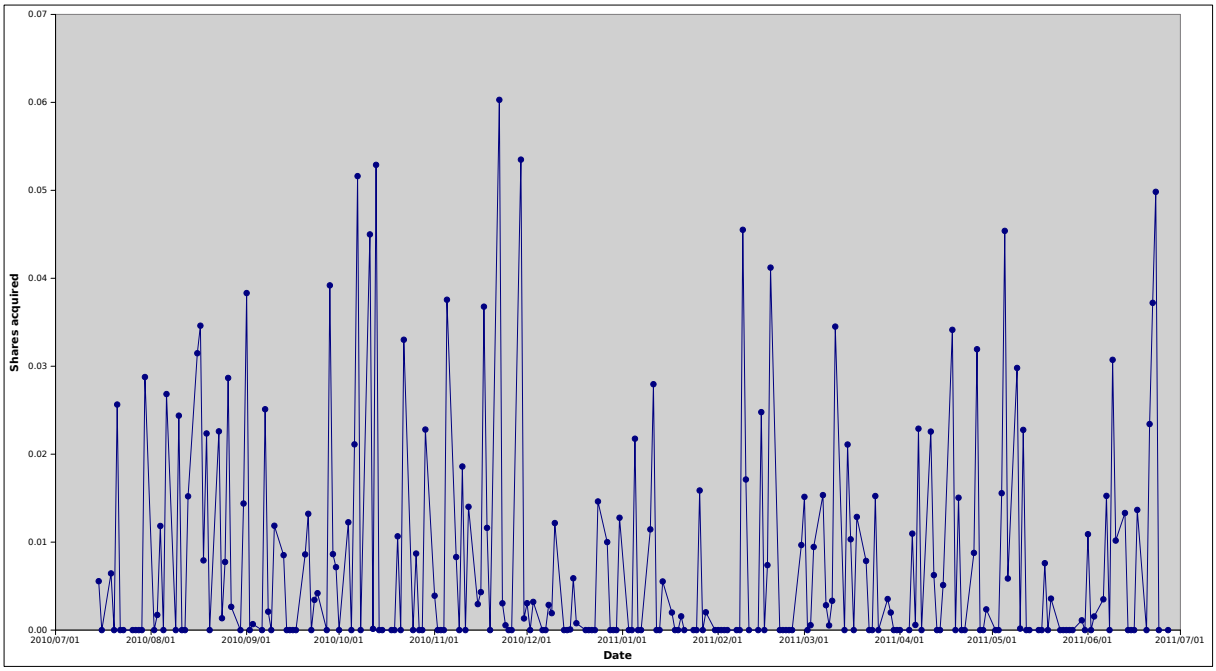

Figure 4.2: Share acquisitions over the last year of the Single Long Share Position training data.

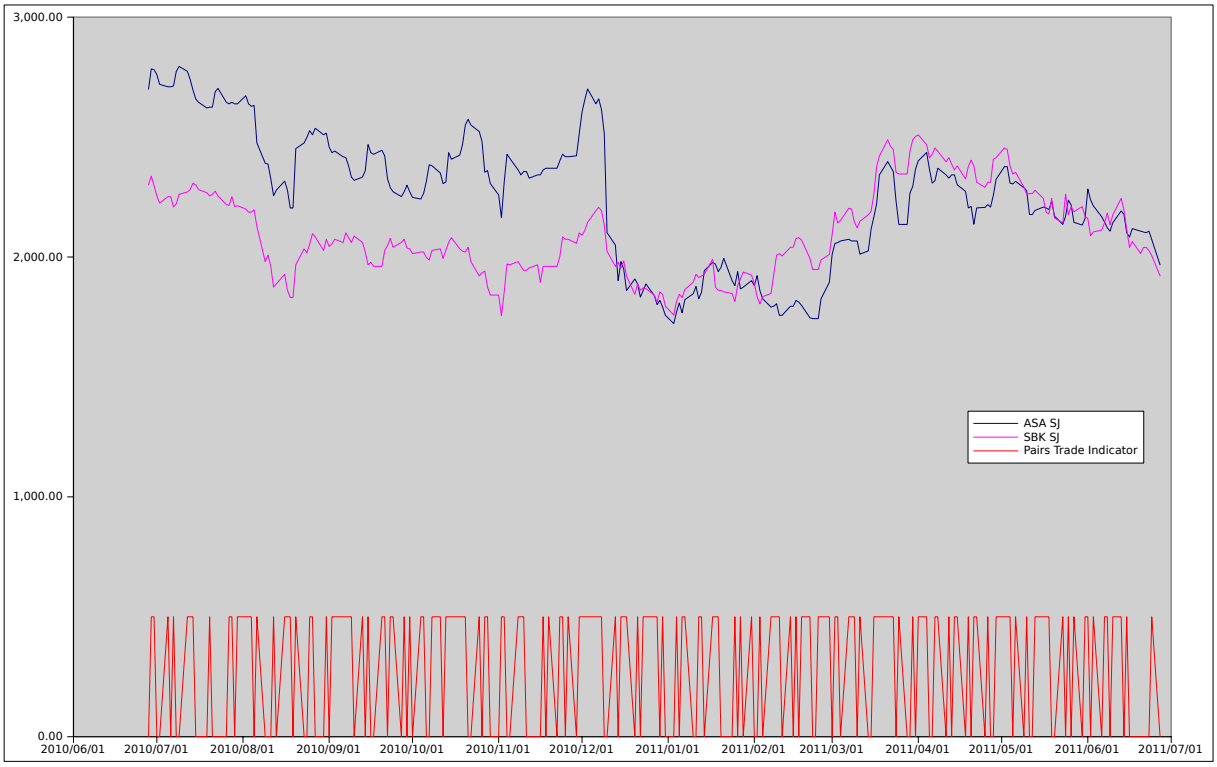

Figure 4.3: The two banking sector shares' fluctuations over the last year of the history used. The red line indicates the decision of when to enter a pairs position and when to exit. These decisions were made using simple logic in a spreadsheet and were used as inputs to the ANN.

held. As soon as the indicator became 0 the pairs trade was to be exited. This method was followed for both the banking sector and the gold mining sector test cases.

4.2.3 Portfolio Choice

The two pairs trading test cases mentioned in Section 4.2 were each chosen for their correlation properties and relatively small number of shares.

Banking Sector

In the banking sector the two shares chosen were ASA:SJ and SBK:SJ, ABSA Group Ltd and Standard Bank Group Ltd respectively. These were chosen on the basis of strong correlation of 98.87% to each other over the total time period considered in Table 4.1. ASA:SJ had a 93.63% correlation to the ALSI, while SBK:SJ had a 95.42% correlation to the ALSI over the same time period.

Gold Mining Sector

In the banking sector the two shares chosen were ANG:SJ and GFI:SJ, AngloGold Ashanti and Gold Fields Ltd. These two shares showed a correlation of 82.60% to each other over the aforementioned time period, with correlations of 72.70% and 54.88% to the ALSI respectively.

4.3 Data

4.3.1 Data Assumptions

1. Stock splits in the data were accounted for by propagating the new share value backwards through the historical data.

2. The day on day changes of input data may is more relevant in the training of a predictive ANN than the overall histories' absolute changes.

3. As bonds get closer to maturity, their term changes which may have an effect on their quoted yield and corresponding zero rate. This means that the interest rate implied by the yields do not remain constant for a specific bond over time. Since day on day changes were used in training, this was considered negligible in the greater scope of the model.

4. Trading costs were deemed negligible for the purpose of this project and as such were not included.

5. Stock dividends were also deemed negligible in terms of day on day share price changes and were not included.

4.3.2 Data Set Choice

The data used for the neural network training were price series histories from the Johannesburg Stock Exchange (JSE), [86], for the period in Table 4.1. The different price histories for share prices, government bond yields, Johannesburg Inter-Bank (JIBAR) deposit rates, foreign exchange values and share index levels were included to give the ANN exposure to multiple asset classes. The bond yields and JIBAR rates were chosen to give exposure to interest rate risk factors at various maturities.

The complete list of input risk factors is given in Appendix C, including the list of one hundred and sixty three (163) shares that made up the JSE All Share Index at the time of data preparation for this project.

4.3.3 Data Preparation

General

The data points between the two boundary dates were filled for each weekday over the entire period. When a risk factor's historical value was missing for a specific date, the previous known value was used (a method used in practice known as rolling the data).

During normalisation of data for use in the artificial neural network training, the concept of a record set is used. This specifically indicates the set of data at a specific observation time or snapshot. Using daily historical prices, a set of data would be the end of day closing rates for all the relevant risk factors.

Many methods of data normalisation exist [16], in this research a simplistic method was used as described below. When normalising a set of data (for a specific snapshot in time), the data could be normalised using one of two methods:

1. Normalise data using the maximum of a specific class's set of data for each specific class; or

2. Normalise the data using the maximum value out of the entire set of data.

The benefit of the first method is that granular, class specific information is preserved. This is useful if one wishes to track, for example when long-term interest rates become higher than short-term interest rates with no regard to other risk factor classes. The most obvious shortfall of this method is thus obvious: each class' normalised set data only contains information about that class. Any information relative to the overall system appears to be lost.

The converse is not necessarily true. While information for a specific class' movements relative to the overall system is stored in the artificial neural network, the granular information is not lost. Due to scaling of different inputs the ANN may have become less sensitive to this information during the training phase.

It is important to note, however, that to train the ANN to predict changes in end-of-day

values it is necessary to use day-on-day changes as input values for training. By using day-on-day changes for the training the question of scaling the inputs becomes a moot point, except in extreme cases which may be ignored, as most observed day-on-day changes were between 0.8 and 1.2.

These day-on-day changes are scaling factors that are determined by dividing the end-of-day value by the previous end-of-day value to arrive at a relative multiplicative change factor. These change factors were then normalised on a class basis such as in the first method described above. It was attempted to preserve class idiosyncrasies and change information by using day-on-day change factors and class based normalisation.

Shares

Share prices are given in one hundredth of a South African Rand, i.e. in cents. This results in a R300 (three hundred South African Rand) share having a listed price of 30000 cents.

Share Indices

Share index levels are the weighted total of various shares available from a stock exchange. For example the Johannesburg All Share Index (ALSI) from the Johannesburg Stock Exchange (JSE). It typically had a value in the region of 30,000 at the time the data was collected for this study.

Deposits

The deposit rates were listed as numerical values, i.e. a 9% interest rate was stored as 0.09. Short-term interest rates give an indication of interbank risk and in effect the cost of funding. They are usually quoted as a simple interest rate over a specified term.

Bonds

Bond yields were stored as percentages. In other words the raw data contained a value of 9 if the bond had a yield of 9%. This was converted to be in the same format as the deposit rates prior to normalisation. Bond information was sourced from the Bond Exchange of South Africa (BESA), [85].

Government bonds are usually quoted as a semi-annual yield rate. These bonds give an indication of long-term interest rates, specifically the 'risk-free' rate.

Foreign Exchange

All currency values were converted to be relative to the South African Rand (ZAR). This was necessary since the other currencies were all quoted relative to the United States Dollar (USD).

The foreign exchange (FX) rates are included to provide sensitivity in the ANN to foreign exposures a JSE listed company might have.

Data preparation summary

While it was not strictly speaking necessary to convert different risk factor types to be in the same format, it was decided to at least have the data in a consistent format across risk factor classes. The reason it was not strictly necessary to convert the data is that during the ANN training phase the individual inputs' weights would have been adapted to adjust for scale given that the required outcomes needed to be produced.

4.3.4 Training

Determination of Buy, Sell and Volume Signals

Single Share Long Position For the single long share position the correct behaviour for the neural network was input using buy and volume signals. The objective was to outperform the ALSI market index over time by buying the share when it was performing worse than the ALSI and holding the position when it was performing better than the ALSI. The basic strategy was to buy cheaply when possible and to maintain the position when not.

This was done by creating two variables, one being a performance indicator and another a volume indicator. For a given day's market data the day's share price value was divided by the day's ALSI value, creating a relative performance factor R_t that indicated the relative value of the share versus the ALSI for time t. The following day the same process was used to create another factor R_{t+1}. If the second day's factor was greater or equal to the previous one, $R_{t+1} \geq R_t$, the share had performed better than the previous day and was deemed to be performing, with its performance indicator then set to 1, $P_{t+1} = 1$. Conversely if the share lost ground against the ALSI, i.e. $R_{t+1} < R_t$, the performance indicator was set to 0, $P_{t+1} = 0$. The binary opposite of the performance indicator P would be the buy indicator B, i.e. when the share did not perform it was required to buy some more of it. The indicator B was used in the ANN training.

For the volume indicator a value of 0 was assigned if the share was performing on that day since if the share was outperforming the ALSI no additional shares were to be bought, i.e. $V_t = 0$. When the share underperformed the market a volume for acquisition was calculated. It was decided that this volume only needed to realign the held share position's value with the ALSI movement so as to not lose relative value versus the market. The required volume V_t was calculated by using:

$$V_t = (R_{t-1} - R_t) \frac{ALSI_t}{S_t}. \tag{4.1}$$

Where R_t and R_{t-1} are the current and previous days' relative performance factors, with $ALSI_t$ the ALSI value and S_t the share price value for time t.

Since the value for V_t can be (and mostly was) a fraction, the astute reader would notice a discrepancy with market practice since it is (generally) not possible to trade a fraction of a share. The analysis was done using a single share as a basis which one would scale up by a much larger number of shares used in real trading, for example an initial starting position might be 1000 long shares with subsequent V_t values multiplied by 1000 and rounded to the nearest whole number.

Pairs Trading Positions Referring to the two strongly correlated shares used, they were designated as share A and share B. To shortly recap on pairs trading, the investor would go long share A and short share B to gain rewards on weaker correlation between the two shares from time to time, when it is expected that share A might outperform or share B underperform relative to each other.

For this case only one indicator was used, the 'live trade' indicator L. This indicator can be interpreted as a relative indicator since if one does not have a pairs position, then a value of $L = 1$ would indicate it necessary to enter the pairs trade. If one did have a pairs position a value of $L = 1$ would indicate to maintain the position, while if $L = 0$ it would be time to liquidate the pairs trade. If one had no current pairs trade position, then a value of $L = 0$ would indicate that it is not yet time to enter the trade.

For each observation t the value of A was divided by the value of B to get a relative value R_t. For two subsequent days t_0 and t_1, if $R_0 < R_1$ the spread between the two shares was increasing and $L_1 = 1$. If $R_0 > R_1$ then the spread is tightening, with $L_1 = 0$. In the case where the spread remains constant the previous position is maintained, i.e. for $R_0 = R_1$ then $L_0 = L_1$.

Training Data

To train the ANN to 'predict' an outcome based on a given set of day-on-day change factors, the corresponding outcome indicator was shifted one observation (or day) to the past prior to training. In other words, for observation $t = n$ the calculated outcome of $t = n + 1$ was used during training. This resulted in the single share positions' ANN training file containing records where the inputs for $t = n$ were matched with B_{n+1} and V_{n+1}. Similarly for the pairs trading cases the inputs for $t = n$ were matched with L_{n+1}.

4.3.5 Testing

The ANN was tested for each of the test cases by running some previously unseen data through the trained network and comparing the results with what would have been the correct decisions.

For the first experiment the testing set was taken as the last year of data while the training set was the entire available period before that, while the second experiment comprised rolling periods of four (4) years of training date followed by one (1) year of testing data staggered by one (1) year. For clarity, these periods are listed in Table 4.1. This is also graphically represented in Figure 4.4.

The training data comprised 2584 days of historical data, starting in 2001/08/01 and ending 2011/06/27. Mirmirani and Li, [40], used 6008 days of historical data with satisfactory results, though a number of authors mention good results with comparatively smaller training data sets, [43], [87] and [11]. Kang, [88], goes as far as to state that ANNs do not necessarily require a large data set to perform well.

	Training		Testing	
Period	**Start Date**	**End Date**	**Start Date**	**End Date**
Long data set	2001/08/01	2010/06/27	2010/06/28	2011/06/27
Data set 1	2001/08/01	2005/07/31	2005/08/01	2006/07/31
Data set 2	2002/08/01	2006/07/31	2006/08/01	2007/07/31
Data set 3	2003/08/01	2007/07/31	2007/08/01	2008/07/31
Data set 4	2004/08/01	2008/07/31	2008/08/01	2009/07/31
Data set 5	2005/08/01	2009/07/31	2009/08/01	2010/07/31
Data set 6	2006/08/01	2010/07/31	2010/08/01	2011/06/27

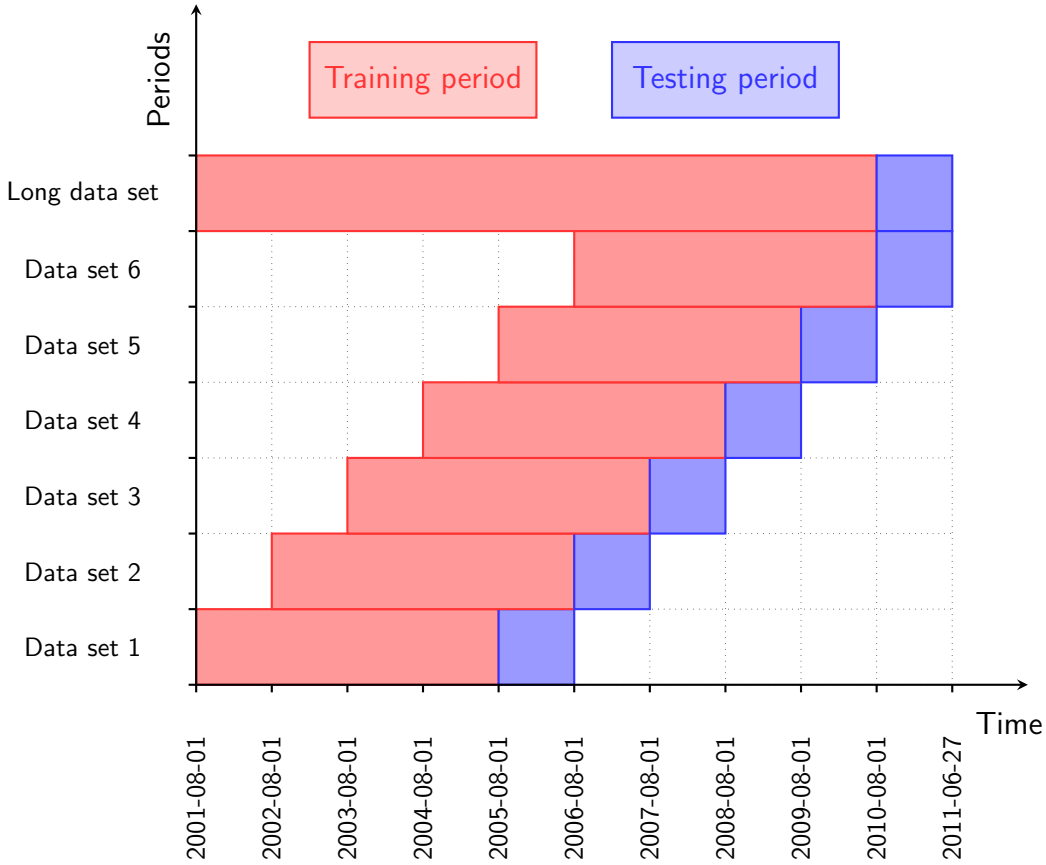

Figure 4.4: The division of training and testing data from the cleaned data set.

	Training		Testing	
Period	**Start Date**	**End Date**	**Start Date**	**End Date**

Table 4.1: The division of training and testing data from the cleaned data set.

4.4 Chapter Summary

In this chapter the data preparation method for the different test cases was discussed. The assumptions that went into various decisions that were made pertaining to data cleaning, normalisation, market behaviour and test case composition were discussed in detail.

Chapter 5

Design and Training

5.1 Introduction

This chapter discusses the choice of ANN type, the ANN design and training used in this research. The assumptions that were made when defining the topology of the neural network and the reasoning behind the training parameter choices are explained in detail.

5.2 ANN Type Choice

As discussed in Sections 2.2.2 and 2.3, the feed-forward back-propagation ANN was used. The reasoning was as follows: it is the simpler of the available models, the most flexible in terms of suitability to a wide range of applications, by far the most widely used ANN architecture and it is supported by the software that was deemed most suitable for use in this research.

5.3 Design Assumptions

The feed-forward back-propagation ANN is a simple and effective design that is well supported FANN library (see Section B.3). The most salient optional FANN inputs for the ANNs during training phases are given in Table 5.1. Most of the settings were kept as the standard recommended values, i.e. the default FANN network construction and training settings.

Setting	Value
Activation function	Sigmoid symmetric
Learning rate	0.700000
Connection rate	1.000000

Setting	Value
Network type	0
Learning momentum	0.000000
Training algorithm	2
Train error function	1
Train stop function	0
Cascade output change fraction	0.010000
Quickprop decay	-0.000100
Quickprop mu	1.750000
Rprop increase factor	1.200000
Rprop decrease factor	0.500000
Rprop delta min	0.000000
Rprop delta max	50.000000
Rprop delta zero	0.100000

Table 5.1: Salient FANN training options.

5.4 Topology

As discussed in Chapter 4 the ANN was designed to accept one hundred and sixty two (162) inputs for shares and a further eight (8) inputs for currencies, seven (7) inputs for bonds, five (5) inputs for deposit rates and another three (3) inputs for the Top40 Index, All Share Index and the gold price respectively.

Two sets of experiments were conducted, both making use of the three case studies mentioned later in this chapter. The first experiment consisted of training the ANN using the bulk of the data available and reserving one year's data for testing, generally referred to as the Single Long Period.

The second experiment used six (6) rolling periods of four (4) years' training data and one year of testing data immediately following the training data, generally referred to as the Rolling Periods. Each period started one year after the previous period started. This was done to test historical annual changes and to maximise utility of the available data. For the first experiment 10000 training epochs were used and for the second experiment 10, 100 and 1000 epochs were used to test the effect of more and less training with the same input data.

It was decided to use various network architectures to examine which would be most suitable for use with each experimental case. This was done by using a fixed number of input nodes (185) with two outputs for the single share case and one output for the pairs trading case, with varying numbers of hidden layers and numbers of hidden layers neurons.

The complete list of architectures tested during the first experiment is given in Table 5.2. By adding hidden layers one could theoretically enhance the complexity of the problems the ANN could solve with increased 'storage' available via the additional vector weightings in the network.

Input Neurons	Hidden Layers	Hidden Neurons	Output Neurons
185	5	5	1 or 2
185	10	5	1 or 2
185	5	10	1 or 2
185	10	10	1 or 2
185	20	20	1 or 2
185	20	40	1 or 2
185	40	20	1 or 2
185	50	50	1 or 2
185	100	50	1 or 2
185	50	100	1 or 2
185	100	100	1 or 2
185	185	185	1 or 2
185	10	185	1 or 2
185	185	10	1 or 2
185	500	100	1 or 2
185	100	500	1 or 2
185	500	500	1 or 2

Table 5.2: The ANN topologies considered during the first experiment. Output neurons are 1 for pairs trading or 2 for the single share case.

The list of of architectures considered during the second experiment are listed in Table 5.3.

Input Neurons	Hidden Layers	Output Neurons	Hidden Neurons	Training Epochs
185	2	1 or 2	1	10
185	2	1 or 2	1	100
185	2	1 or 2	1	1000
185	10	1 or 2	1	10
185	10	1 or 2	1	100
185	10	1 or 2	1	1000
185	2	1 or 2	10	10
185	2	1 or 2	10	100
185	2	1 or 2	10	1000
185	10	1 or 2	10	10
185	10	1 or 2	10	100
185	10	1 or 2	10	1000

Table 5.3: The ANN topologies considered during the second experiment. Output neurons are for pairs trading (1) or the single share cases (2).

5.5 Training

The ANNs were trained using the following parameters for the FANN library:

- Training epochs = 10000 (times retrained using the same data).
- Training error = 1×10^{-9} (see Section 2.2.4).

There were seventeen (17) topologies for each test case in the first experiment. Each of these was repeatedly trained for 10000 epochs until it was possible to minimise the 'Failed bits' value, usually to zero. These failed bits refer to the number of training data sets the ANN was unable to correctly produce outputs of training data for.

The reason that it would work to rerun a 10000 epoch training and obtain different training results is that in changing the weights of inter-neuron vectors random numbers are used make small adjustments to the weight vectors after each training epoch. The danger of overfitting the training data is present though.

The inherent danger of using many training epochs is over-fitting the ANN to the data available for training, rather than establishing slightly more general processing of input data. This would mean that it would tend to give better results with the training data than previously unseen data, possibly giving incorrect output. From a personal conversation of the author with an ANN expert, it was ascertained that it is a better approach to use many training data sets presented fewer times to the ANN rather than a smaller number of sets presented many times. This observation was partially tested in the second experiment where all four (4) basic topologies were tested using 10, 100 and 1000 training epochs.

Training the ANN with multiple hidden layers adds another level of complexity. While each hidden layer provides the potential to store more information, it also requires more robust training and data. Unfortunately this is another practical trade-off that is best determined via experimentation, hence the decision to experiment with various ANN configurations.

5.6 Chapter Summary

This chapter presented the logic and decisions behind the ANN designs and training used in this project. The assumptions that were made when defining the topology of the neural network and the reasoning behind the training parameter choices.

Chapter 6

Results

6.1 Introduction

This chapter contains the results obtained during the training phase in Section 6.2 with the testing phase's results in Section 6.3. These results are presented in table and graphical format with explanatory text on each section. Each test case as per Chapter 4 is treated individually, using the testing periods defined in Table 4.1.

In a typical training process the ANN would start off with thousands of failed bits and quickly decrease this number to close to zero, within a hundred or a thousand epochs. Graphical examples of the training are included for each test case. Similarly the testing sections of the experiments also have graphical representation.

The results presented below are split into testing and training sections, which in turn were classified by case type (Single Share Long Position, Baking Sector Pairs Position or Gold Mining Sector Pairs Position) and network topology. The results presented under the training section contain training errors which are an indication of the average mean square error (see Section 2.2.4) once the required amount of training passes of the test data had been completed. For comparison, the desired error is included too (see Section 5.5).

The testing sections below contain columns for the output variables described in Section 4.2 for each test case to give an indication of the comparative performance of the different topologies. When the data for training and testing was prepared the same methods were used for deriving what the ANN outputs should be. During testing the errors produced by comparing

the required test outputs to the realized values were calculated and those values are given under the output variable columns mentioned above.

For training and testing both experimental period types were reported as described in Table 4.1 and Figure 4.4; the single long period of training data and the multiple rolling periods of training data. In both cases a year's worth of data was used for testing. For additional variety and to simplify the experiment four (4) different network topologies were considered for the multi-period training compared to the seventeen (17) unique topologies used in the single long period training experiments.

It must be noted that the time given in seconds in the testing tables covers both the training and testing time and should be seen as a total time. The testing time for each topology was virtually the same, with the major variance in total time caused by the training time.

6.2 Training Results

6.2.1 Single Share Long Position

Single Long Period Training

The Single Share Long Position test case training results for the single long period training are shown in Table 6.1. From the results it can be seem that the best performance in the training set was displayed in the second row, i.e. with a topology of 185 input, 10 hidden layers with 5 neurons each and the required two outputs. As the topologies become more complex there was no increase in accuracy of the output, in fact the opposite was observed. For the Single Share Long Position it would appear that a simpler ANN was sufficient to produce the required output.

Input	Hidden	Output	Layers	Epochs	Desired Error	Training Error	Bit Fail
185	5	2	5	10000	1.00E-09	0.030729476	0
185	5	2	10	10000	1.00E-09	0.030377191	0
185	10	2	5	10000	1.00E-09	0.030819979	0
185	10	2	10	10000	1.00E-09	0.030584587	0
185	20	2	20	10000	1.00E-09	0.030799944	0
185	40	2	20	10000	1.00E-09	0.030525092	0
185	20	2	40	10000	1.00E-09	0.030633995	0
185	50	2	50	10000	1.00E-09	0.030757571	0

Input	Hidden	Output	Layers	Epochs	Desired Error	Training Error	Bit Fail
185	50	2	100	10000	1.00E-09	0.030714789	0
185	100	2	50	10000	1.00E-09	0.030703554	0
185	100	2	100	10000	1.00E-09	0.030785894	0
185	185	2	185	10000	1.00E-09	0.030930908	0
185	185	2	10	10000	1.00E-09	0.031392466	0
185	10	2	185	10000	1.00E-09	0.030535359	0
185	100	2	500	10000	1.00E-09	0.030826304	0
185	500	2	100	10000	1.00E-09	0.033163037	0
185	500	2	500	10000	1.00E-09	0.031307641	1

Table 6.1: Single Share Long Position training results.

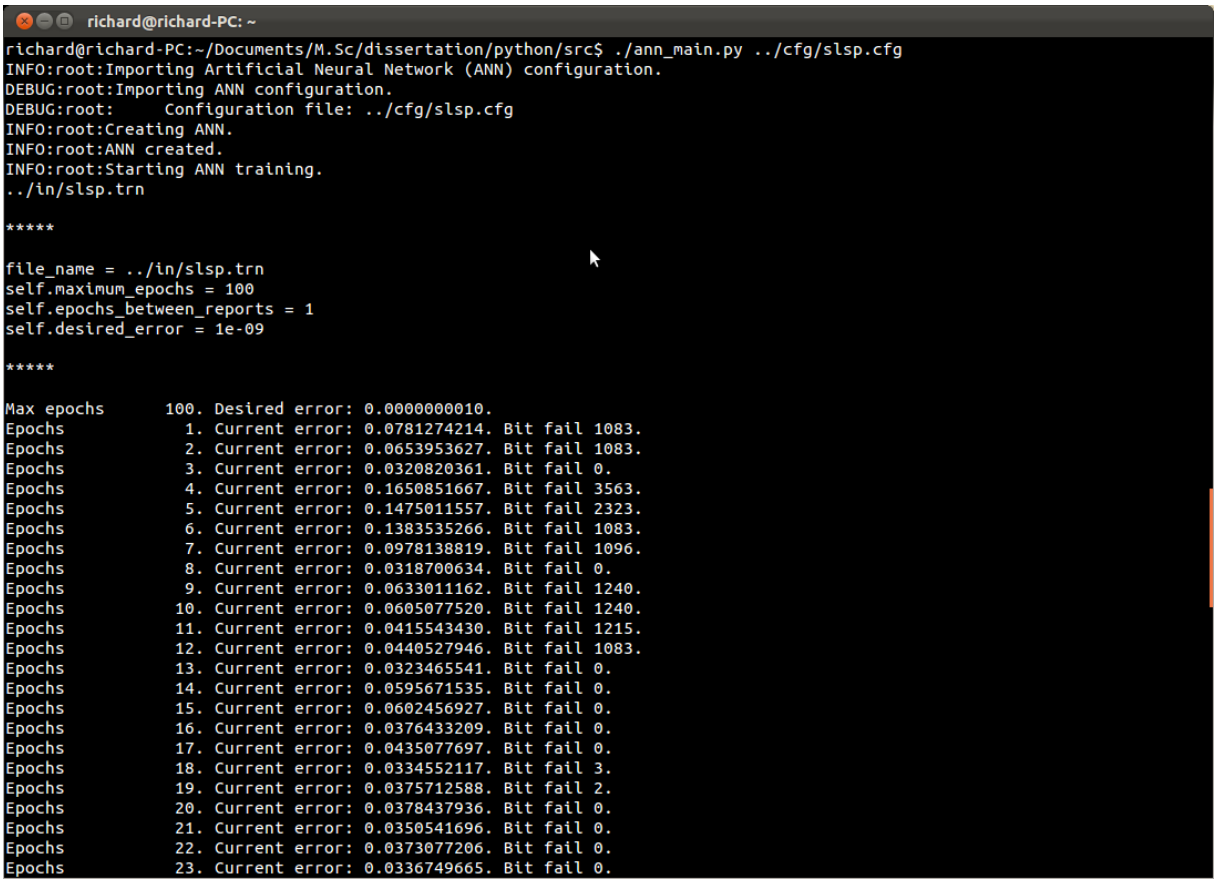

Figure 6.1: An example of Single Share Long Position single period training.

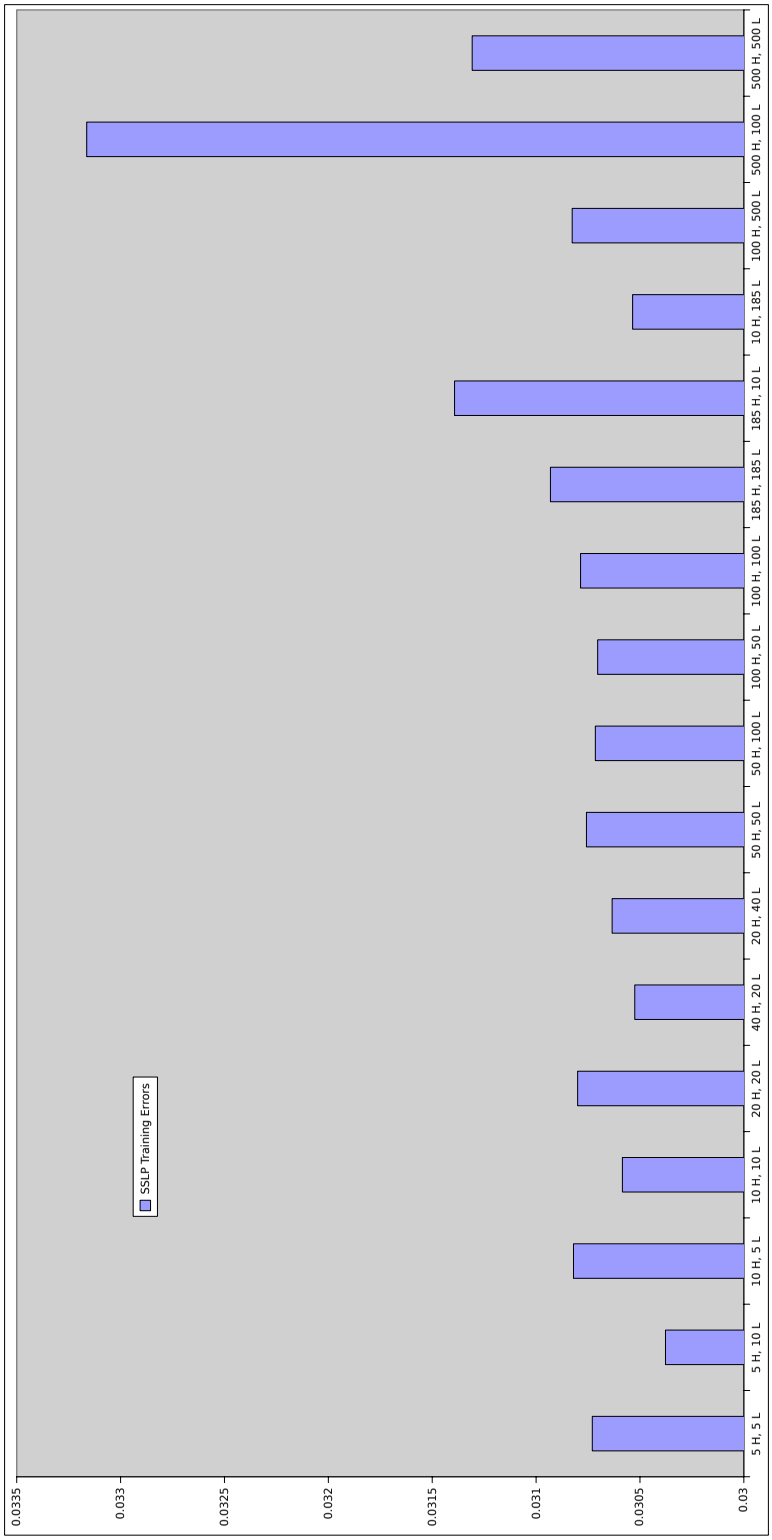

Figure 6.2: A graph of Single Share Long Position single period training errors.

Multi-Period Training

The training results of the multiple training periods are shown in Table 6.2 and Figure 6.3. As expected, the training error decreased as the number of training epochs increased. The comparative benefit of the additional time taken for training over additional epochs will be examined in the testing section. The best training result was gained by single hidden layer with 10 neurons after 1000 training epochs.

Period	Input	Neurons	Output	Hidden Layers	Epochs	Training Error	Bit Fail	Time [s]
1	185	2	2	1	10	0.031179598	0	0.07
1	185	2	2	1	100	0.030912770	0	0.17
1	185	2	2	1	1000	0.030602561	0	1.22
1	185	10	2	1	10	0.033465151	0	0.09
1	185	10	2	1	100	0.030917555	0	0.46
1	185	10	2	1	1000	0.029544571	14	4.14
1	185	2	2	10	10	0.031364948	0	0.06
1	185	2	2	10	100	0.030912325	0	0.17
1	185	2	2	10	1000	0.030237190	5	1.2
1	185	10	2	10	10	0.032011036	0	0.09
1	185	10	2	10	100	0.030913953	0	0.47
1	185	10	2	10	1000	0.030630581	0	4.13
2	185	2	2	1	10	0.031925913	0	0.08
2	185	2	2	1	100	0.030856095	0	0.18
2	185	2	2	1	1000	0.030702304	0	1.28
2	185	10	2	1	10	0.034834064	0	0.09
2	185	10	2	1	100	0.030843817	0	0.49
2	185	10	2	1	1000	0.030473357	0	4.27
2	185	2	2	10	10	0.030982317	0	0.06
2	185	2	2	10	100	0.030898694	0	0.18
2	185	2	2	10	1000	0.030646544	1	1.28
2	185	10	2	10	10	0.038562521	0	0.09
2	185	10	2	10	100	0.030896558	0	0.48
2	185	10	2	10	1000	0.030602798	0	4.25
3	185	2	2	1	10	0.031803422	0	0.07
3	185	2	2	1	100	0.030945303	0	0.17
3	185	2	2	1	1000	0.030604197	1	1.27
3	185	10	2	1	10	0.035040766	0	0.09
3	185	10	2	1	100	0.030870315	0	0.48
3	185	10	2	1	1000	0.030641381	0	4.29
3	185	2	2	10	10	0.031091604	0	0.06
3	185	2	2	10	100	0.030819576	0	0.18
3	185	2	2	10	1000	0.030862257	0	1.29
3	185	10	2	10	10	0.032734588	0	0.1

Period	Input	Neurons	Output	Hidden Layers	Epochs	Training Error	Bit Fail	Time [s]
3	185	10	2	10	100	0.030900501	0	0.48
3	185	10	2	10	1000	0.030567402	2	4.27
4	185	2	2	1	10	0.032324210	0	0.07
4	185	2	2	1	100	0.031043660	0	0.18
4	185	2	2	1	1000	0.030924682	0	1.29
4	185	10	2	1	10	0.035935376	0	0.09
4	185	10	2	1	100	0.030894686	0	0.48
4	185	10	2	1	1000	0.030384695	0	4.26
4	185	2	2	10	10	0.031538766	0	0.06
4	185	2	2	10	100	0.030936990	0	0.19
4	185	2	2	10	1000	0.030864434	2	1.28
4	185	10	2	10	10	0.036474809	0	0.1
4	185	10	2	10	100	0.030965613	0	0.48
4	185	10	2	10	1000	0.030633772	0	4.24
5	185	2	2	1	10	0.031569134	0	0.07
5	185	2	2	1	100	0.031049959	0	0.18
5	185	2	2	1	1000	0.030416518	0	1.28
5	185	10	2	1	10	0.031896003	0	0.1
5	185	10	2	1	100	0.030997466	0	0.48
5	185	10	2	1	1000	0.030661378	0	4.27
5	185	2	2	10	10	0.031455293	0	0.07
5	185	2	2	10	100	0.030677935	0	0.17
5	185	2	2	10	1000	0.030467482	0	1.29
5	185	10	2	10	10	0.031557929	0	0.09
5	185	10	2	10	100	0.030969262	0	0.48
5	185	10	2	10	1000	0.030928345	0	4.27
6	185	2	2	1	10	0.031238768	0	0.06
6	185	2	2	1	100	0.030936379	0	0.18
6	185	2	2	1	1000	0.030356996	1	1.28
6	185	10	2	1	10	0.031238958	0	0.09
6	185	10	2	1	100	0.030838624	0	0.48
6	185	10	2	1	1000	0.030759430	0	4.28
6	185	2	2	10	10	0.031983212	0	0.06
6	185	2	2	10	100	0.031086802	0	0.18
6	185	2	2	10	1000	0.030938892	0	1.28
6	185	10	2	10	10	0.032401089	0	0.1
6	185	10	2	10	100	0.031073093	0	0.48
6	185	10	2	10	1000	0.030457893	0	4.3

Table 6.2: Single Share Long Position multi-period training results.

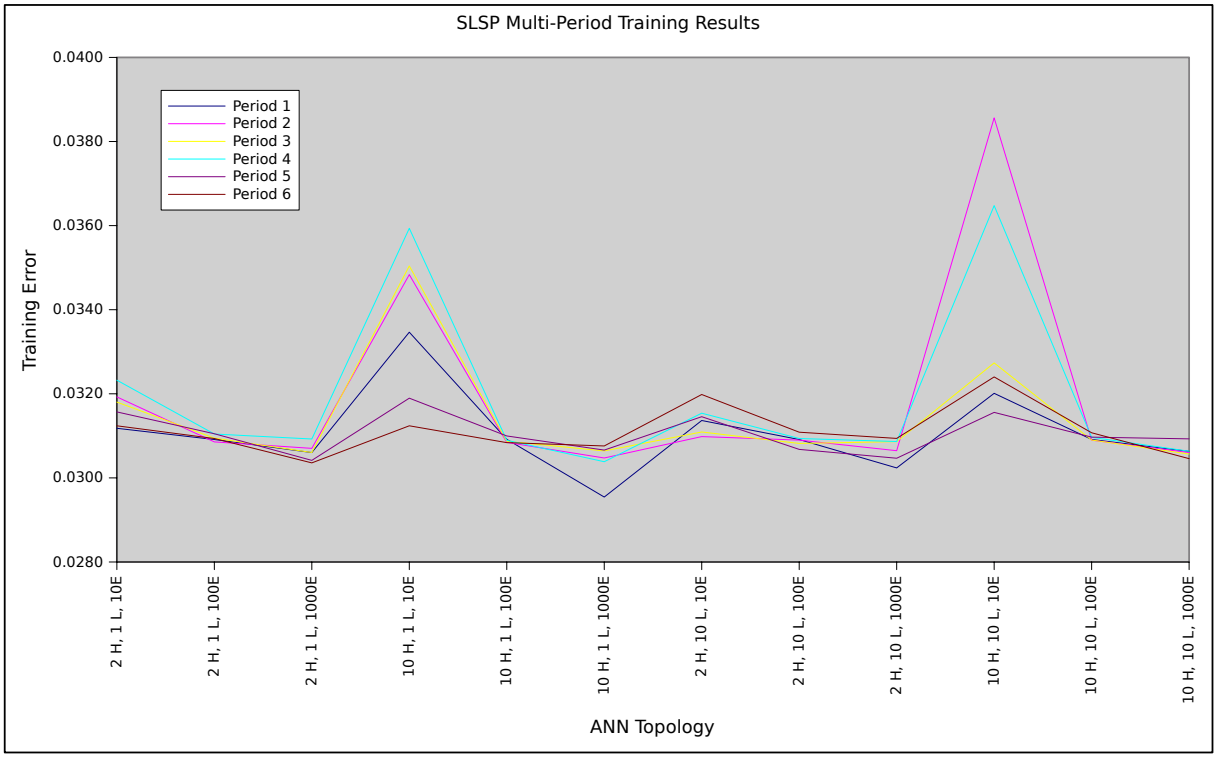

Figure 6.3: Graph of Single Share Long Position multi-period training errors.

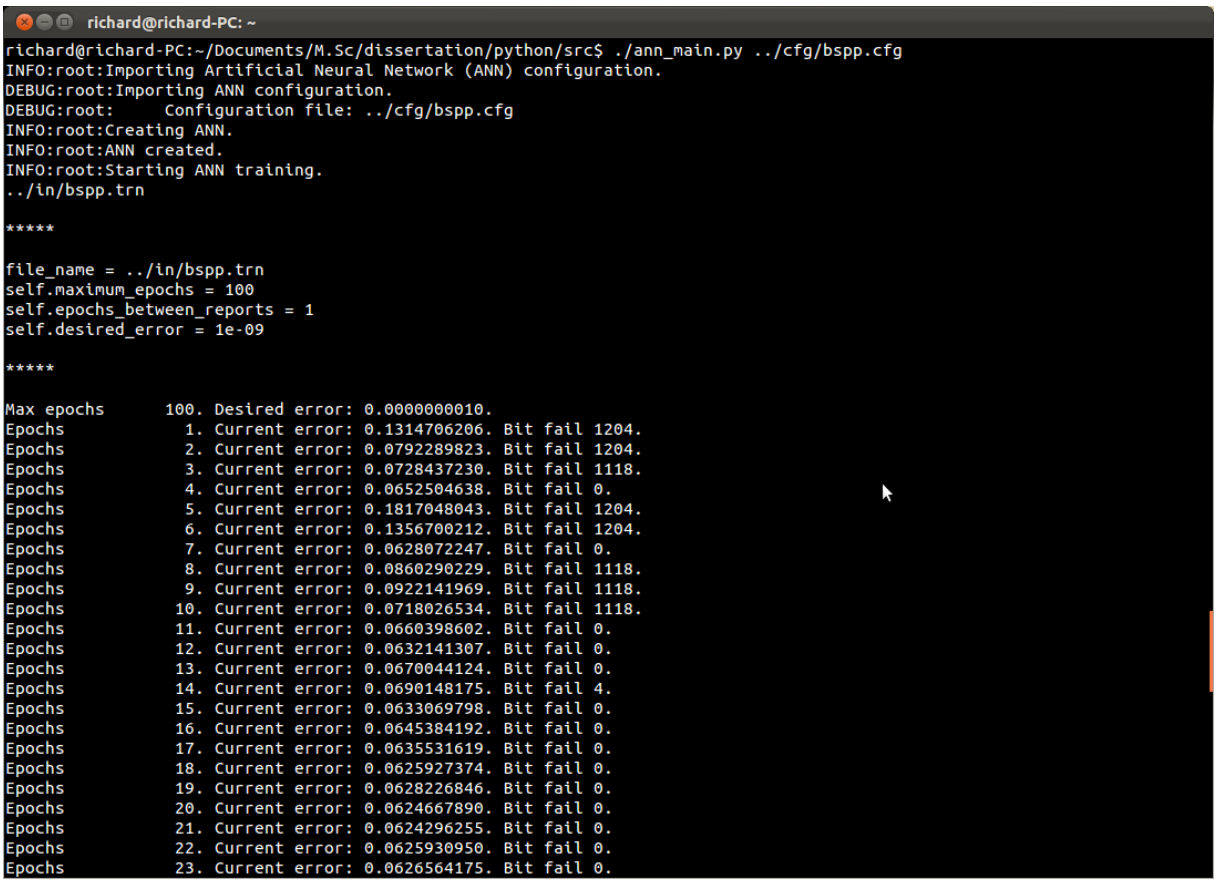

Figure 6.4: An example of Banking Sector Pairs Position training.

6.2.2 Banking Sector Pairs Position

Single Long Period Training

For the Banking Sector Pairs Position the single period training results indicated that the best performing topology during training was one with a square design of 100 hidden layers and 100 neurons per hidden layer. All the topologies tested had the same 185 inputs and a single output. The Banking Sector Pairs Positions test case training results are shown in Table 6.3. Once again it appeared that if the ANN's hidden layers and neurons became larger in number than the optimal performing topology, the training performance decreased significantly.

Input	Hidden	Output	Layers	Epochs	Desired Error	Training Error	Bit Fail
185	5	1	5	10000	1.00E-09	0.061798338	0
185	5	1	10	10000	1.00E-09	0.061462831	0
185	10	1	5	10000	1.00E-09	0.061957032	0

Input	Hidden	Output	Layers	Epochs	Desired Error	Training Error	Bit Fail
185	10	1	10	10000	1.00E-09	0.061327662	0
185	20	1	20	10000	1.00E-09	0.061806206	0
185	40	1	20	10000	1.00E-09	0.062220585	0
185	20	1	40	10000	1.00E-09	0.062082034	0
185	50	1	50	10000	1.00E-09	0.061760213	0
185	50	1	100	10000	1.00E-09	0.061285973	0
185	100	1	50	10000	1.00E-09	0.061696831	0
185	100	1	100	10000	1.00E-09	0.060885459	0
185	185	1	185	10000	1.00E-09	0.062141523	0
185	185	1	10	10000	1.00E-09	0.062265340	0
185	10	1	185	10000	1.00E-09	0.061789021	0
185	100	1	500	10000	1.00E-09	0.063053198	0
185	500	1	100	10000	1.00E-09	0.062476888	1
185	500	1	500	10000	1.00E-09	0.062380537	1

Table 6.3: Banking Sector Pairs Position training results.

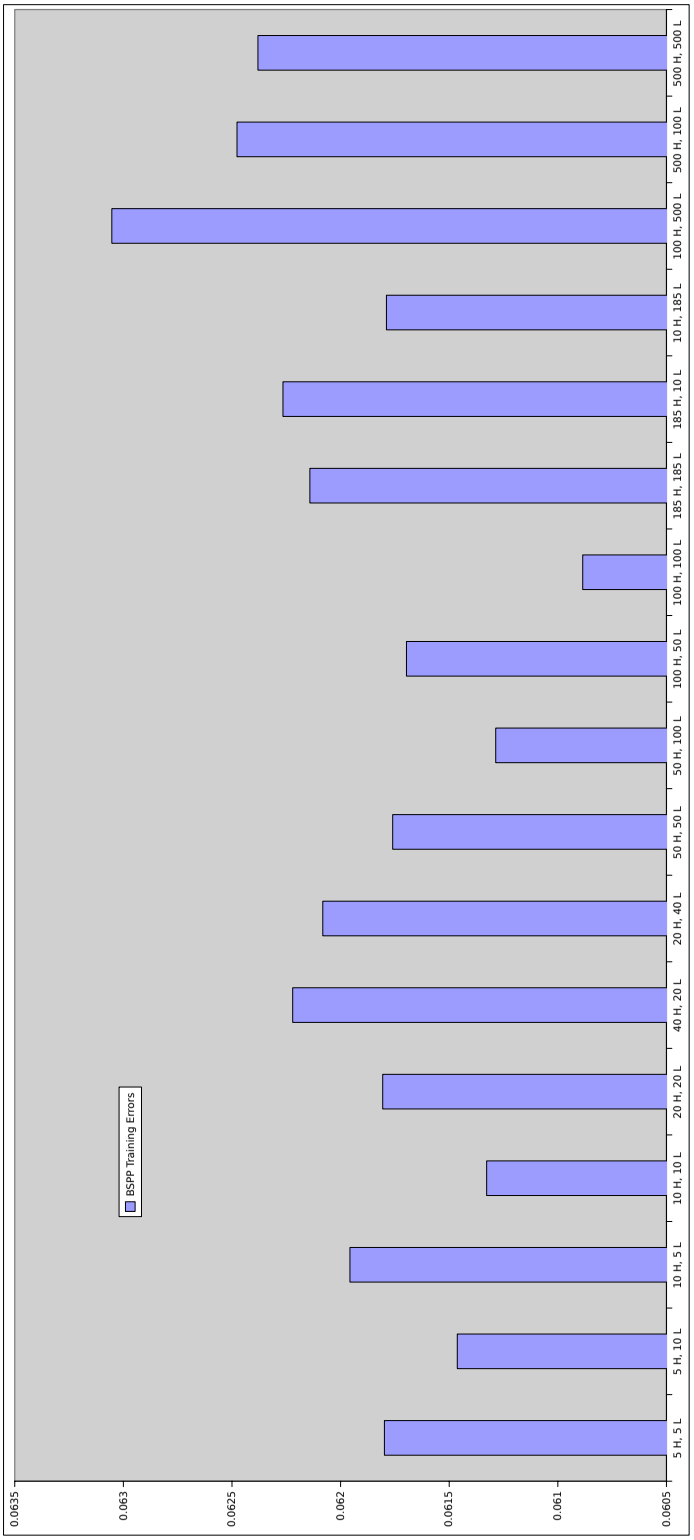

Figure 6.5: A graph of Banking Sector Pairs Position single period training errors.

Multi-Period Training

The training results of the multiple training periods for the Banking Sector Pairs Position test case are shown in Table 6.4 and Figure 6.6. The best training results were gained when there were 10 hidden layers after 1000 training epochs.

Period	Input	Neurons	Output	Hidden	Epochs	Training Error	Bit Fail	Time [s]
1	185	2	2	1	10	0.062429186	0	0.06
1	185	2	2	1	100	0.0621587113	0	0.16
1	185	2	2	1	1000	0.062429864	0	1.12
1	185	10	2	1	10	0.0624291785	0	0.09
1	185	10	2	1	100	0.0620858185	0	0.46
1	185	10	2	1	1000	0.0615396015	0	4.07
1	185	2	2	10	10	0.0628261119	0	0.07
1	185	2	2	10	100	0.0624299981	0	0.16
1	185	2	2	10	1000	0.0624299608	0	1.11
1	185	10	2	10	10	0.0626156628	0	0.09
1	185	10	2	10	100	0.0620692372	0	0.46
1	185	10	2	10	1000	0.0620009452	0	4.06
2	185	2	2	1	10	0.062294744	0	0.05
2	185	2	2	1	100	0.0620504282	0	0.16
2	185	2	2	1	1000	0.0622861795	0	1.12
2	185	10	2	1	10	0.0624071248	0	0.1
2	185	10	2	1	100	0.0619767457	0	0.46
2	185	10	2	1	1000	0.0602340139	0	4.06
2	185	2	2	10	10	0.0625289157	0	0.06
2	185	2	2	10	100	0.0619977042	0	0.16
2	185	2	2	10	1000	0.0622863844	0	1.11
2	185	10	2	10	10	0.0622579791	0	0.1
2	185	10	2	10	100	0.0619557761	0	0.47
2	185	10	2	10	1000	0.0611554794	0	4.06
3	185	2	2	1	10	0.0623825155	0	0.06
3	185	2	2	1	100	0.0623389110	0	0.16
3	185	2	2	1	1000	0.0623387210	0	1.1
3	185	10	2	1	10	0.0623475276	0	0.09
3	185	10	2	1	100	0.0620904118	0	0.45
3	185	10	2	1	1000	0.0613677911	0	4.04
3	185	2	2	10	10	0.0623477399	0	0.06
3	185	2	2	10	100	0.0623386800	0	0.17
3	185	2	2	10	1000	0.0623384789	0	1.1
3	185	10	2	10	10	0.0625070333	0	0.09
3	185	10	2	10	100	0.0621617325	0	0.47
3	185	10	2	10	1000	0.0610024035	0	4.03

Period	Input	Neurons	Output	Hidden	Epochs	Training Error	Bit Fail	Time [s]
4	185	2	2	1	10	0.0624197572	0	0.06
4	185	2	2	1	100	0.0623941682	0	0.16
4	185	2	2	1	1000	0.0623949468	0	1.1
4	185	10	2	1	10	0.0626947507	0	0.09
4	185	10	2	1	100	0.0621217936	0	0.46
4	185	10	2	1	1000	0.0611340217	0	4.04
4	185	2	2	10	10	0.0624432638	0	0.06
4	185	2	2	10	100	0.0623943582	0	0.17
4	185	2	2	10	1000	0.0623943917	0	1.1
4	185	10	2	10	10	0.0624140464	0	0.1
4	185	10	2	10	100	0.0622899458	0	0.46
4	185	10	2	10	1000	0.0610945038	0	4.03
5	185	2	2	1	10	0.0624024980	0	0.06
5	185	2	2	1	100	0.0623986460	0	0.16
5	185	2	2	1	1000	0.0623987876	0	1.1
5	185	10	2	1	10	0.0624557734	0	0.08
5	185	10	2	1	100	0.0621090531	0	0.46
5	185	10	2	1	1000	0.0610135607	0	4.04
5	185	2	2	10	10	0.0624057800	0	0.06
5	185	2	2	10	100	0.0615309849	0	0.16
5	185	2	2	10	1000	0.0609690063	0	1.13
5	185	10	2	10	10	0.0624286309	0	0.09
5	185	10	2	10	100	0.0621838607	0	0.45
5	185	10	2	10	1000	0.0619129911	0	4.05
6	185	2	2	1	10	0.0624951050	0	0.06
6	185	2	2	1	100	0.0618985370	0	0.16
6	185	2	2	1	1000	0.0620436706	1	1.12
6	185	10	2	1	10	0.0624487437	0	0.1
6	185	10	2	1	100	0.0618894435	0	0.45
6	185	10	2	1	1000	0.0617796779	2	4.03
6	185	2	2	10	10	0.0624385402	0	0.06
6	185	2	2	10	100	0.0617934354	0	0.16
6	185	2	2	10	1000	0.0616119169	0	1.11
6	185	10	2	10	10	0.0628165230	0	0.08
6	185	10	2	10	100	0.0622262284	0	0.45
6	185	10	2	10	1000	0.0622927248	0	4.04

Table 6.4: Banking Sector Pairs Position multi-period training results.

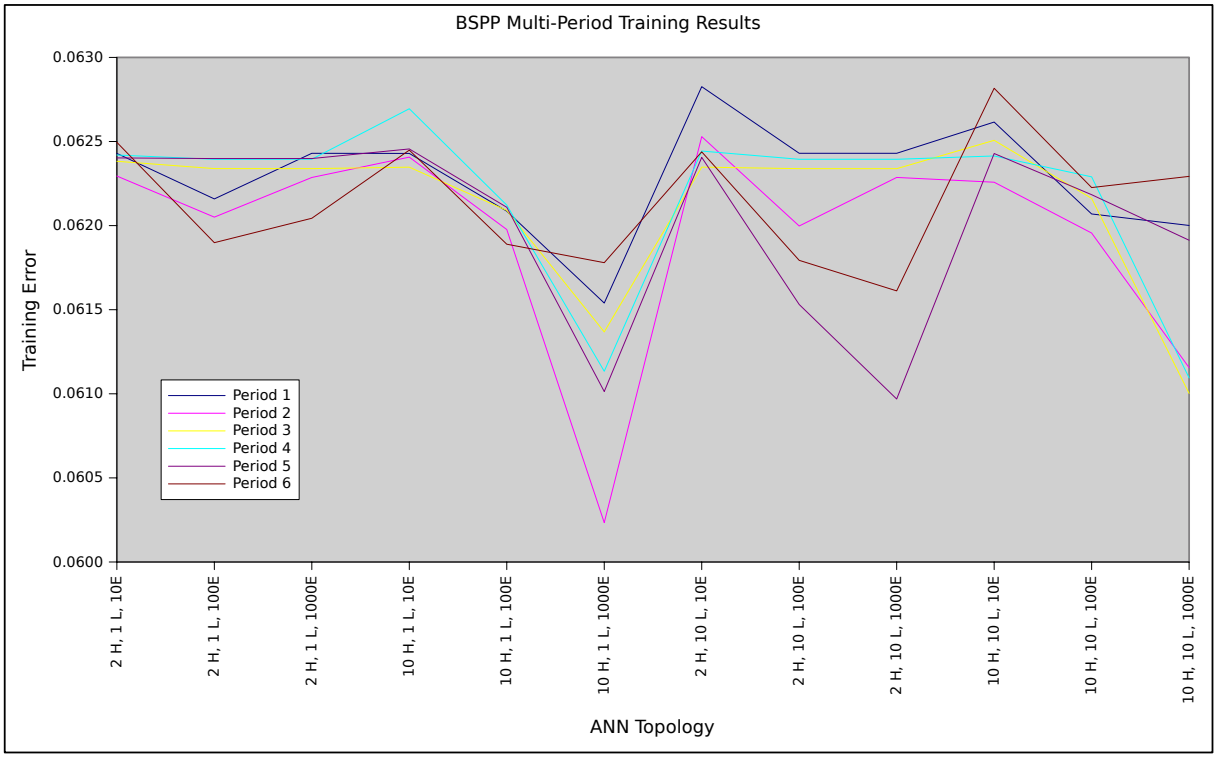

Figure 6.6: Graph of Banking Sector Pairs Position multi-period training errors.

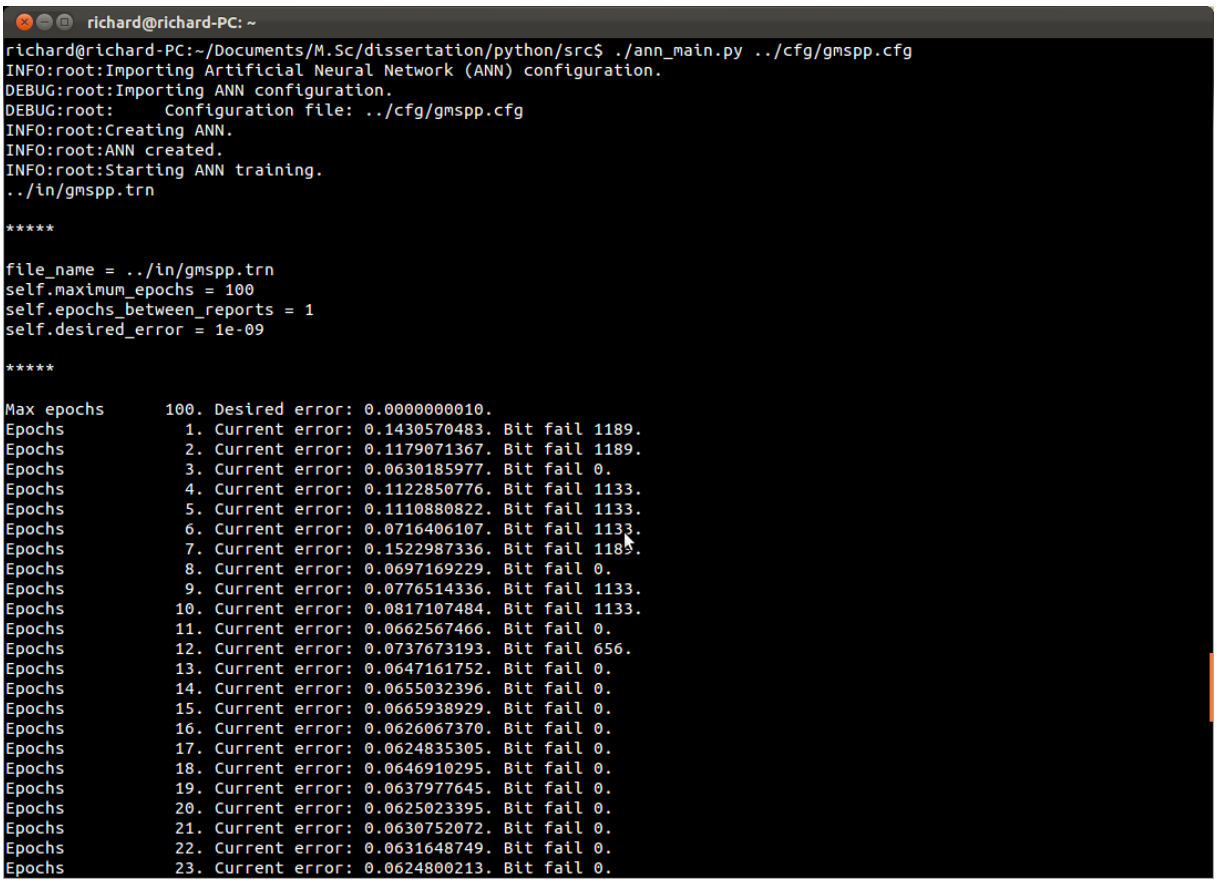

Figure 6.7: An example of Gold Mining Sector Pairs Position training.

6.2.3 Gold Mining Sector Pairs Position

Single Long Period Training

The Gold Mining Sector Pairs Position single period test case training results are shown in Table 6.5. From the table it can be seen that while the topology with 185 hidden layers and 185 neurons per hidden layer appears to have the lowest training error, it has 2 failed bits at the end of training. This means that two of the training cases did not give the correct output once training was completed. As such it should be discarded as a viable result. The result with no failed bits and lowest error was using a topology of ten hidden layers with 10 neurons per layer.

Input	Hidden	Output	Layers	Epochs	Desired Error	Training Error	Bit Fail
185	5	1	5	10000	1.00E-09	0.061606258	0
185	5	1	10	10000	1.00E-09	0.060560241	2

Input	Hidden	Output	Layers	Epochs	Desired Error	Training Error	Bit Fail
185	10	1	5	10000	1.00E-09	0.061096948	2
185	10	1	10	10000	1.00E-09	0.061251428	0
185	20	1	20	10000	1.00E-09	0.061565567	0
185	40	1	20	10000	1.00E-09	0.060907554	1
185	20	1	40	10000	1.00E-09	0.061690733	1
185	50	1	50	10000	1.00E-09	0.061472412	0
185	50	1	100	10000	1.00E-09	0.061620787	0
185	100	1	50	10000	1.00E-09	0.062080942	0
185	100	1	100	10000	1.00E-09	0.061578691	0
185	185	1	185	10000	1.00E-09	0.060374513	2
185	185	1	10	10000	1.00E-09	0.061769307	1
185	10	1	185	10000	1.00E-09	0.061561905	2
185	100	1	500	10000	1.00E-09	0.063234963	1
185	500	1	100	10000	1.00E-09	0.062274057	40
185	500	1	500	10000	1.00E-09	0.064508252	18

Table 6.5: Gold Mining Sector Pairs Position training results.

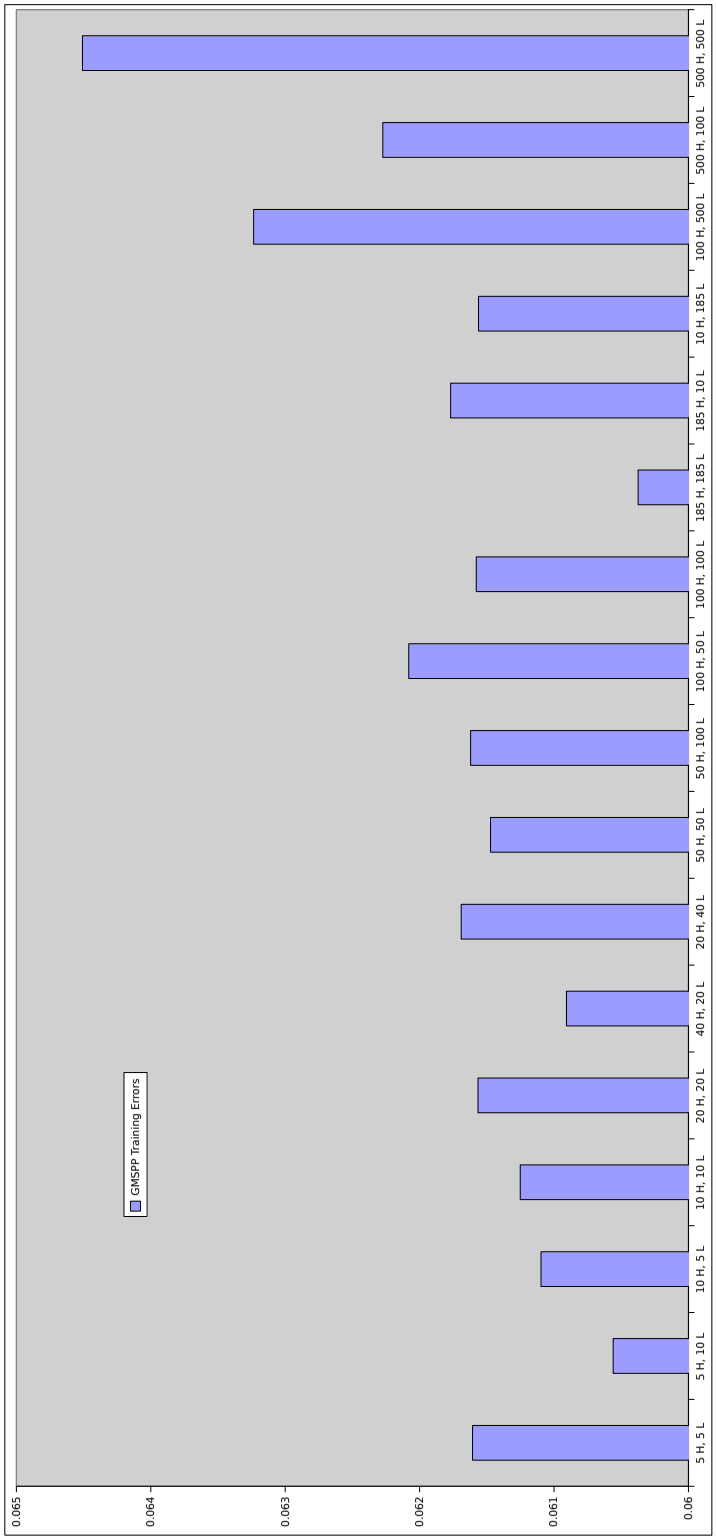

Figure 6.8: A graph of Gold Mining Sector Pairs Position training errors.

Multi-Period Training

The training results of the multiple training periods for the Gold Mining Sector Pairs Position test case are shown in Table 6.6 and Figure 6.9. The best training results were gained when there were 10 neurons per hidden layer and after 1000 training epochs.

Period	Input	Neurons	Output	Hidden	Epochs	Training Error	Bit Fail	Time [s]
1	185	2	2	1	10	0.0625080541	0	0.06
1	185	2	2	1	100	0.0624968857	0	0.16
1	185	2	2	1	1000	0.0624969788	0	1.1
1	185	10	2	1	10	0.0625197589	0	0.1
1	185	10	2	1	100	0.0620016605	0	0.45
1	185	10	2	1	1000	0.0619475953	0	4.02
1	185	2	2	10	10	0.0624987781	0	0.06
1	185	2	2	10	100	0.0622537173	0	0.17
1	185	2	2	10	1000	0.0617502630	0	1.13
1	185	10	2	10	10	0.0625271350	0	0.09
1	185	10	2	10	100	0.0622765981	0	0.45
1	185	10	2	10	1000	0.0609963983	0	4.03
2	185	2	2	1	10	0.0625028163	0	0.06
2	185	2	2	1	100	0.0622304417	0	0.16
2	185	2	2	1	1000	0.0625001043	0	1.09
2	185	10	2	1	10	0.0627761036	0	0.08
2	185	10	2	1	100	0.0622370765	0	0.45
2	185	10	2	1	1000	0.0611356385	0	4.03
2	185	2	2	10	10	0.0627314895	0	0.07
2	185	2	2	10	100	0.0624994226	0	0.16
2	185	2	2	10	1000	0.0611798130	0	1.12
2	185	10	2	10	10	0.0625962019	0	0.1
2	185	10	2	10	100	0.0622485578	0	0.46
2	185	10	2	10	1000	0.0611039847	1	4.04
3	185	2	2	1	10	0.0625426099	0	0.07
3	185	2	2	1	100	0.0621549003	0	0.16
3	185	2	2	1	1000	0.0624794960	0	1.12
3	185	10	2	1	10	0.0626132190	0	0.09
3	185	10	2	1	100	0.0620649345	0	0.46
3	185	10	2	1	1000	0.0608767755	1	4.05
3	185	2	2	10	10	0.0625051260	0	0.06
3	185	2	2	10	100	0.0618798696	0	0.16
3	185	2	2	10	1000	0.0623166524	0	1.11
3	185	10	2	10	10	0.0625714958	0	0.09
3	185	10	2	10	100	0.0621554106	0	0.46
3	185	10	2	10	1000	0.0617257655	0	4.04

Period	Input	Neurons	Output	Hidden	Epochs	Training Error	Bit Fail	Time [s]
4	185	2	2	1	10	0.0624312125	0	0.06
4	185	2	2	1	100	0.0624294505	0	0.17
4	185	2	2	1	1000	0.0624299571	0	1.12
4	185	10	2	1	10	0.0624342561	0	0.09
4	185	10	2	1	100	0.0620508231	0	0.46
4	185	10	2	1	1000	0.0617899336	0	4.11
4	185	2	2	10	10	0.0624549389	0	0.06
4	185	2	2	10	100	0.0620759092	0	0.16
4	185	2	2	10	1000	0.0617122315	0	1.16
4	185	10	2	10	10	0.0624987967	0	0.09
4	185	10	2	10	100	0.0619999282	0	0.46
4	185	10	2	10	1000	0.0611472540	0	4.09
5	185	2	2	1	10	0.0625620335	0	0.06
5	185	2	2	1	100	0.0624263287	0	0.16
5	185	2	2	1	1000	0.0624261908	0	1.12
5	185	10	2	1	10	0.0624540783	0	0.08
5	185	10	2	1	100	0.0613703541	0	0.46
5	185	10	2	1	1000	0.0606724657	0	4.06
5	185	2	2	10	10	0.0625908896	0	0.06
5	185	2	2	10	100	0.0624259003	0	0.16
5	185	2	2	10	1000	0.0624257959	0	1.1
5	185	10	2	10	10	0.0642919913	0	0.09
5	185	10	2	10	100	0.0616440885	0	0.46
5	185	10	2	10	1000	0.0606935918	0	4.07
6	185	2	2	1	10	0.0623879582	0	0.06
6	185	2	2	1	100	0.0623834096	0	0.17
6	185	2	2	1	1000	0.0623831898	0	1.12
6	185	10	2	1	10	0.0625463799	0	0.1
6	185	10	2	1	100	0.0619158931	0	0.45
6	185	10	2	1	1000	0.0605561435	0	4.13
6	185	2	2	10	10	0.0625890195	0	0.06
6	185	2	2	10	100	0.0623838194	0	0.15
6	185	2	2	10	1000	0.0623837672	0	1.15
6	185	10	2	10	10	0.0623893999	0	0.09
6	185	10	2	10	100	0.0619504489	0	0.46
6	185	10	2	10	1000	0.0608765967	0	4.11

Table 6.6: Gold Mining Sector Pairs Position multi-period training results.

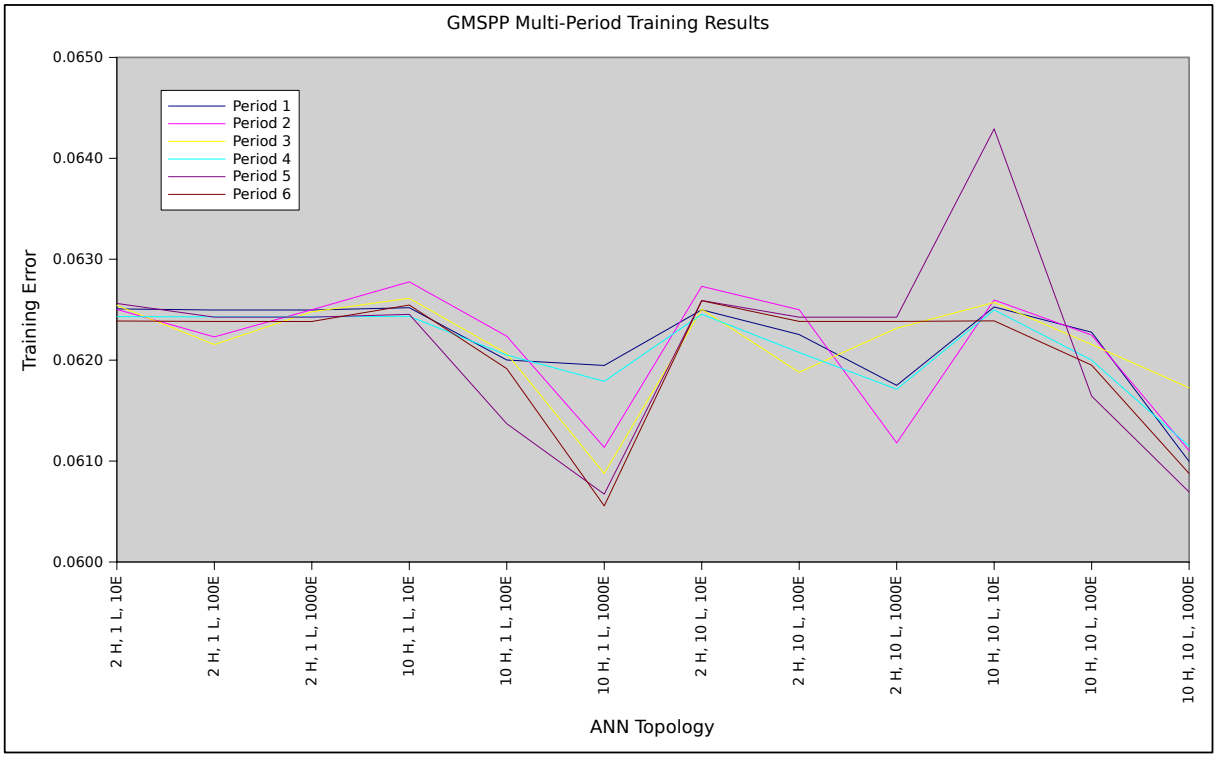

Figure 6.9: Graph of Gold Mining Sector Pairs Position multi-period training errors.

6.3 Testing Results

6.3.1 Single Share Long Position

Single Long Period Testing

From Table 6.7 it can be seen that the Single Share Long Position single period testing results that identifying the topology with the best performance appears to be a challenging task. The best performing topology would need to minimize the average error reported for both the Signal and Volume, preferably with low maximum errors in comparison to the other topologies. In addition the training data also plays a role, i.e. minimal error possible and with no bit failure.

By using the above arguments, the 'best' topology was identified as 5 hidden layers with 10 neurons each. This is unfortunately a difficult choice to make, though with different training parameters it may differ. This topology seemed to give the best aggregate performance under testing and training, though as can be seen in the tables a number of other topologies give almost similar results.

For clarification, since the Signal value was either 1 or 0, a Signal Average Error value of 0.500737 means that the average error measured over the entire testing set was roughly 0.5. Thus the ANN gave the correct prediction for roughly half the given input sets. This is similar to predicting what the markets would do tomorrow based on changes in some market data from yesterday to today.

A Volume Average Error of 0.008100 means that over the entire testing set the ANN's volume prediction was out by 0.008100 on average. Given that the average of required volumes in testing data was 0.008164, this is a disappointing result. It means that on average the predicted volume for trading was mostly wrong.

While the results indicate that this experiment was not as successful as had been hoped, it was not without merit either. The major benefit of these results is the indication that either the basic design of the experiment, data preparation or the training methods was problematic and would require adaptation to produce better results. This is discussed in more detail in Chapter 7.

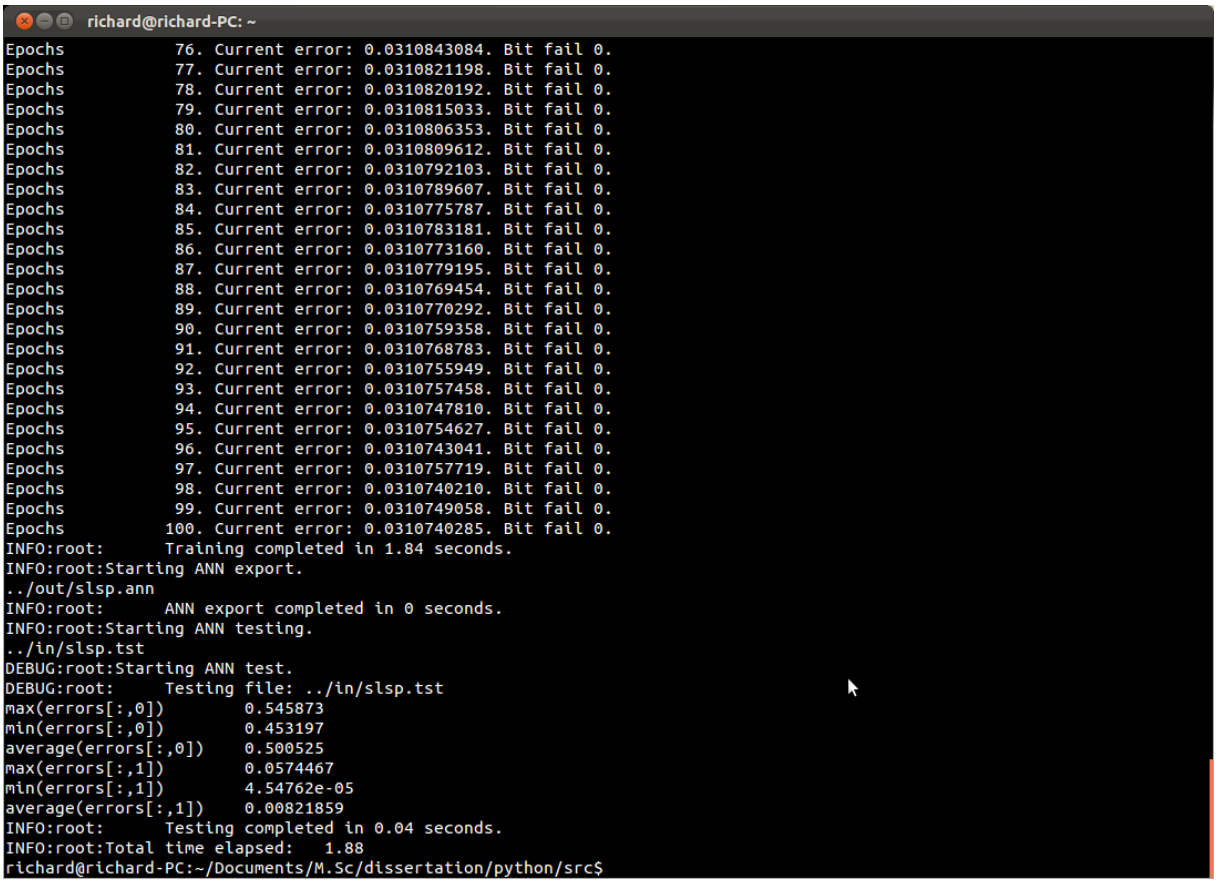

Figure 6.10: An example of Single Share Long Position testing.

				Signal			Volume			
Input	**Hidden**	**Output**	**Layers**	**Max. Error**	**Min. Error**	**Ave. Error**	**Max. Error**	**Min. Error**	**Ave. Error**	**Time [s]**
185	5	2	5	0.557851	0.439167	0.502249	0.055611	0.000121	0.008553	73.81
185	5	2	10	0.577986	0.402306	0.501706	0.051332	0.000049	0.009959	72.89
185	10	2	5	0.554258	0.443929	0.500737	0.057546	0.000001	0.008100	134.79
185	10	2	10	0.627390	0.416652	0.501880	0.055332	0.000084	0.008537	132.75
185	20	2	20	0.590774	0.394985	0.501607	0.059270	0.000012	0.008705	255.92
185	40	2	20	0.609669	0.397037	0.500093	0.039913	0.000060	0.013577	498.85
185	20	2	40	0.595050	0.400297	0.502354	0.056765	0.000033	0.008568	256.10
185	50	2	50	0.583015	0.410648	0.501369	0.073265	0.000031	0.022788	623.22
185	50	2	100	0.567452	0.423754	0.500070	0.052056	0.000047	0.010988	632.94
185	100	2	50	0.553768	0.424844	0.499248	0.046502	0.000007	0.011896	1240.17
185	100	2	100	0.566575	0.415470	0.499249	0.041404	0.000180	0.015860	1242.18
185	185	2	185	0.609533	0.394169	0.504711	0.066344	0.000009	0.014963	2280.57
185	185	2	10	0.563229	0.439014	0.500209	0.061993	0.000015	0.010938	2313.51
185	10	2	185	0.572275	0.408306	0.499964	0.051516	0.000023	0.009331	137.87
185	100	2	500	0.551440	0.427771	0.500977	0.059894	0.000028	0.012301	1258.76
185	500	2	100	0.541464	0.464311	0.501358	0.053355	0.000016	0.019826	6145.73
185	500	2	500	0.564020	0.424281	0.500219	0.065684	0.000358	0.042497	6162.05

Table 6.7: Single Share Long Position test results.

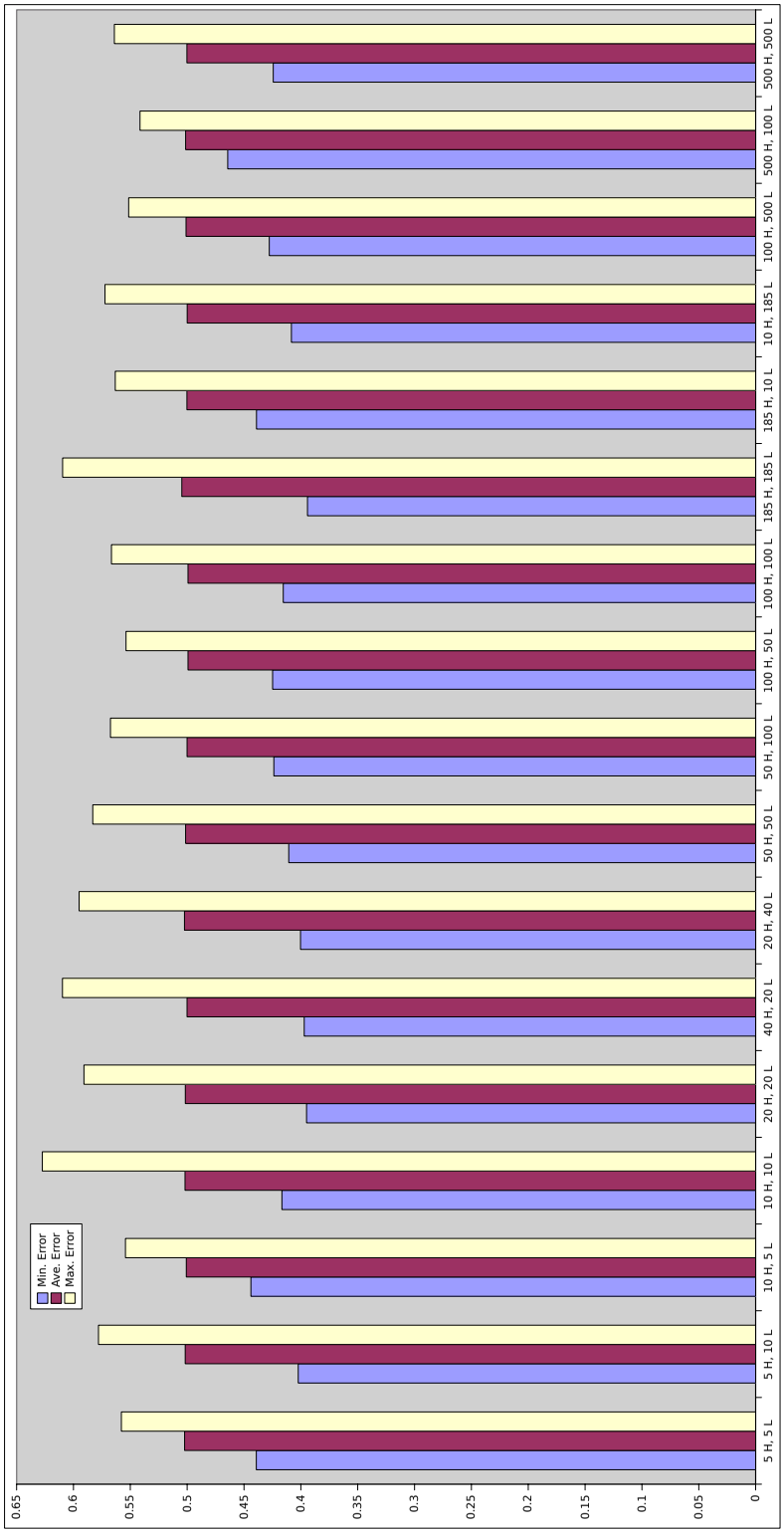

Figure 6.11: A graph of Single Share Long Position testing indicator errors.

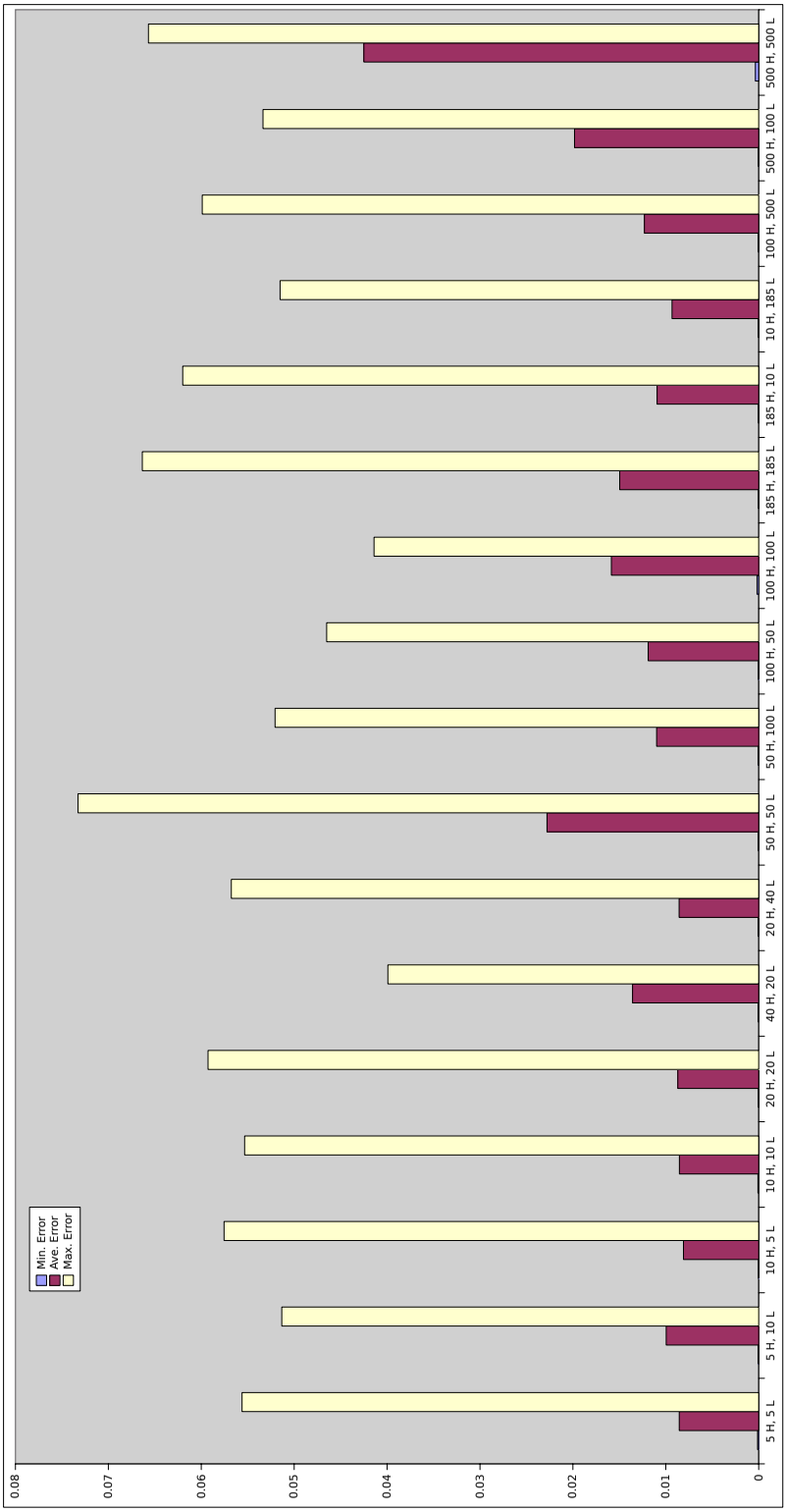

Figure 6.12: A graph of Single Share Long Position testing volume errors.

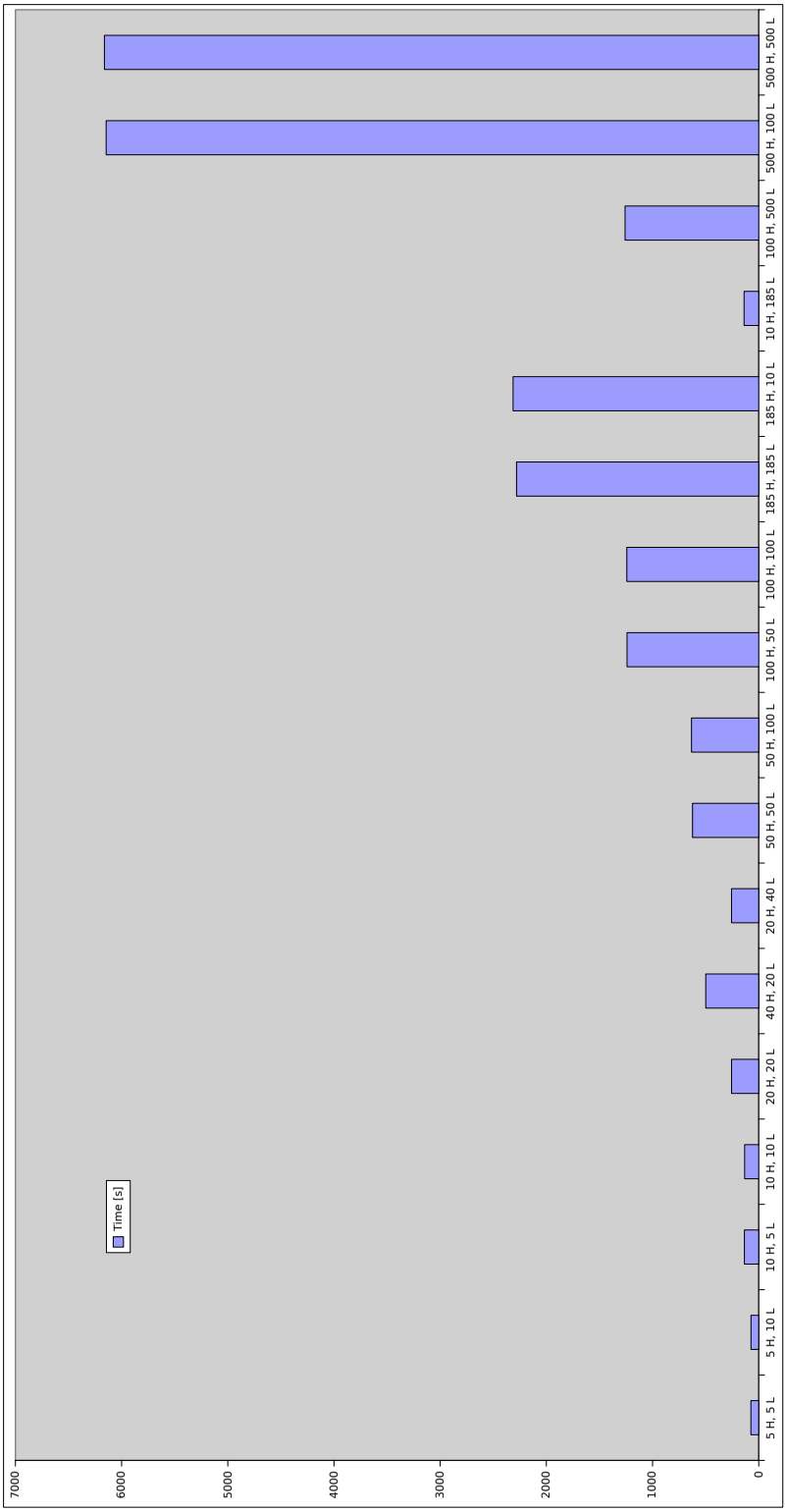

Figure 6.13: A graph of Single Share Long Position testing times.

Multi-Period Testing

The testing results of Single Share Long Position multiple testing periods are shown in Table 6.8 and Figure 6.14. The best results were given by comparatively simple networks, yielding average errors close to 0.4879. These were divided between 10 epochs and 1000 epochs and for different network topologies yet in both cases there were either 2 hidden layers with 10 neurons or 10 hidden layers with 2 neurons, seemingly indicating that a simpler network with 20 neurons performed best.

						Signal			Volume			
Period	Epochs	Input	Neurons	Output	Hidden	Max. Error	Min. Error	Ave. Error	Max. Error	Min. Error	Ave. Error	Time [s]
1	10	185	2	2	1	0.528196	0.471804	0.496445	0.019634	0.004881	0.017101	0.05
1	100	185	2	2	1	0.708245	0.335202	0.494681	0.011738	0.000049	0.002076	0.04
1	1000	185	2	2	1	0.705726	0.314801	0.497056	0.011791	0.000034	0.002083	0.05
1	10	185	10	2	1	0.595381	0.404619	0.487956	0.124145	0.104743	0.111928	0.04
1	100	185	10	2	1	0.632620	0.394231	0.495275	0.022868	0.000020	0.002446	0.05
1	1000	185	10	2	1	0.733950	0.265537	0.498677	0.019380	0.000014	0.003261	0.06
1	10	185	2	2	10	0.508762	0.491238	0.498892	0.044361	0.030838	0.032322	0.05
1	100	185	2	2	10	0.610115	0.387357	0.494982	0.016698	0.000003	0.003296	0.05
1	1000	185	2	2	10	0.842443	0.174935	0.495506	0.013843	0.000013	0.002423	0.05
1	10	185	10	2	10	0.515229	0.487938	0.498843	0.056108	0.026676	0.043806	0.05
1	100	185	10	2	10	0.635285	0.362452	0.495310	0.012291	0.000018	0.002061	0.04
1	1000	185	10	2	10	0.795817	0.269267	0.496204	0.025289	0.000028	0.003201	0.04
2	10	185	2	2	1	0.615390	0.384610	0.492484	0.114880	0.102904	0.113257	0.05
2	100	185	2	2	1	0.575814	0.417990	0.497124	0.010485	0.000023	0.001894	0.04
2	1000	185	2	2	1	0.637511	0.348173	0.497649	0.010540	0.000011	0.001895	0.05
2	10	185	10	2	1	0.511798	0.488202	0.499239	0.035016	0.021000	0.024568	0.05
2	100	185	10	2	1	0.601269	0.369971	0.498046	0.048875	0.000229	0.013822	0.05
2	1000	185	10	2	1	0.635609	0.344063	0.495912	0.010165	0.000012	0.002166	0.05
2	10	185	2	2	10	0.523870	0.476130	0.498445	0.033483	0.021508	0.023130	0.04
2	100	185	2	2	10	0.576305	0.421704	0.496863	0.010504	0.000006	0.001912	0.05
2	1000	185	2	2	10	0.632876	0.256688	0.496444	0.010367	0.000001	0.001946	0.06
2	10	185	10	2	10	0.508749	0.490993	0.500060	0.070651	0.012218	0.056994	0.05
2	100	185	10	2	10	0.571732	0.428200	0.496817	0.010647	0.000010	0.001848	0.05
2	1000	185	10	2	10	0.724193	0.285058	0.498476	0.010209	0.000022	0.001924	0.05
3	10	185	2	2	1	0.511310	0.488690	0.499309	0.073558	0.009299	0.013348	0.05
3	100	185	2	2	1	0.560883	0.443555	0.496896	0.062898	0.000034	0.004260	0.05
3	1000	185	2	2	1	0.711015	0.398139	0.495963	0.062877	0.000011	0.004233	0.04
3	10	185	10	2	1	0.747315	0.252945	0.484887	0.159166	0.093808	0.098694	0.06
3	100	185	10	2	1	0.607234	0.393596	0.496447	0.085600	0.000063	0.013235	0.05
3	1000	185	10	2	1	0.677547	0.407408	0.496537	0.062954	0.000007	0.004224	0.06
3	10	185	2	2	10	0.571418	0.428582	0.495639	0.058629	0.000972	0.054672	0.04
3	100	185	2	2	10	0.581884	0.419343	0.495602	0.064643	0.000002	0.004430	0.05
3	1000	185	2	2	10	0.572039	0.426965	0.496423	0.068807	0.000003	0.005044	0.05
3	10	185	10	2	10	0.713890	0.287268	0.487249	0.119468	0.051220	0.057811	0.05
3	100	185	10	2	10	0.570305	0.429959	0.496006	0.062708	0.000001	0.004304	0.04
3	1000	185	10	2	10	0.737427	0.281890	0.495771	0.062968	0.000045	0.004653	0.04
4	10	185	2	2	1	0.553714	0.446286	0.502238	0.078141	0.001624	0.062627	0.05
4	100	185	2	2	1	0.546155	0.450880	0.498071	0.091960	0.000031	0.013628	0.05
4	1000	185	2	2	1	0.577417	0.354372	0.496495	0.095811	0.000357	0.012935	0.05
4	10	185	10	2	1	0.736608	0.263269	0.490252	0.146976	0.030090	0.061892	0.06
4	100	185	10	2	1	0.599638	0.393089	0.497397	0.095275	0.000106	0.013677	0.04
4	1000	185	10	2	1	0.625665	0.347289	0.493578	0.095449	0.000220	0.012968	0.05
4	10	185	2	2	10	0.519110	0.480897	0.499265	0.125481	0.027783	0.040595	0.05
4	100	185	2	2	10	0.609141	0.396164	0.497082	0.095774	0.000006	0.012915	0.05
4	1000	185	2	2	10	0.650979	0.237641	0.487901	0.095028	0.000009	0.012925	0.05
4	10	185	10	2	10	0.687572	0.316559	0.492998	0.158950	0.059838	0.074325	0.04
4	100	185	10	2	10	0.680061	0.302930	0.495175	0.154017	0.000243	0.030729	0.06
4	1000	185	10	2	10	0.779916	0.352160	0.499025	0.097430	0.000032	0.014289	0.05
5	10	185	2	2	1	0.519905	0.480095	0.499234	0.070477	0.013977	0.020180	0.05
5	100	185	2	2	1	0.545342	0.454583	0.498472	0.051402	0.000084	0.007176	0.05
5	1000	185	2	2	1	0.669017	0.336250	0.493930	0.050533	0.000086	0.006922	0.05
5	10	185	10	2	1	0.559026	0.443323	0.497868	0.069821	0.009309	0.063408	0.04
5	100	185	10	2	1	0.554526	0.445443	0.497681	0.045826	0.000087	0.007255	0.06
5	1000	185	10	2	1	0.559675	0.419089	0.497138	0.049989	0.000002	0.007349	0.04
5	10	185	2	2	10	0.522739	0.477318	0.500912	0.112568	0.049277	0.062636	0.04

						Signal			Volume			
Period	Epochs	Input	Neurons	Output	Hidden	Max. Error	Min. Error	Ave. Error	Max. Error	Min. Error	Ave. Error	Time [s]
5	100	185	2	2	10	0.633220	0.342754	0.491300	0.079928	0.000015	0.010232	0.06
5	1000	185	2	2	10	0.647909	0.340791	0.494258	0.050392	0.000016	0.006923	0.05
5	10	185	10	2	10	0.541809	0.458191	0.498392	0.228973	0.172469	0.178675	0.05
5	100	185	10	2	10	0.557585	0.442391	0.497818	0.048318	0.000209	0.006954	0.05
5	1000	185	10	2	10	0.577980	0.420007	0.497613	0.051500	0.000079	0.006993	0.04
6	10	185	2	2	1	0.560999	0.439002	0.502584	0.036686	0.000178	0.018878	0.05
6	100	185	2	2	1	0.553608	0.441004	0.499617	0.039188	0.000038	0.017178	0.04
6	1000	185	2	2	1	0.656182	0.333316	0.499350	0.054814	0.000060	0.008844	0.04
6	10	185	10	2	1	0.680622	0.321177	0.492772	0.296755	0.231342	0.244581	0.05
6	100	185	10	2	1	0.572842	0.413124	0.499628	0.053853	0.000019	0.009675	0.04
6	1000	185	10	2	1	0.614179	0.376652	0.499474	0.054858	0.000009	0.009627	0.04
6	10	185	2	2	10	0.560021	0.439979	0.497457	0.083235	0.022946	0.030987	0.05
6	100	185	2	2	10	0.556285	0.441125	0.500578	0.053418	0.000322	0.009254	0.04
6	1000	185	2	2	10	0.596912	0.403704	0.499561	0.054425	0.000087	0.008968	0.05
6	10	185	10	2	10	0.686689	0.313441	0.507846	0.120891	0.060301	0.068630	0.04
6	100	185	10	2	10	0.568638	0.426469	0.500244	0.050011	0.000005	0.010737	0.04
6	1000	185	10	2	10	0.694158	0.276969	0.498883	0.055338	0.000125	0.008699	0.05

Table 6.8: Single Share Long Position multi-period testing results.

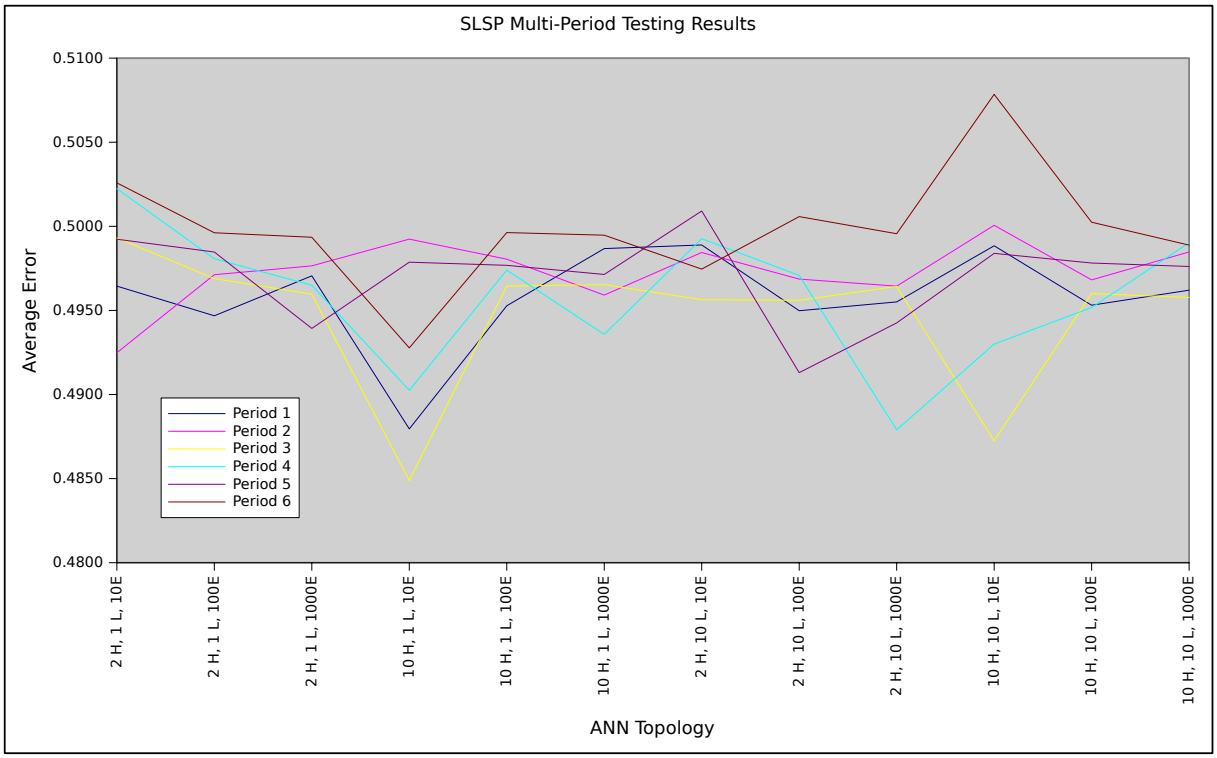

Figure 6.14: Graph of Single Share Long Position multi-period average indicator error during testing.

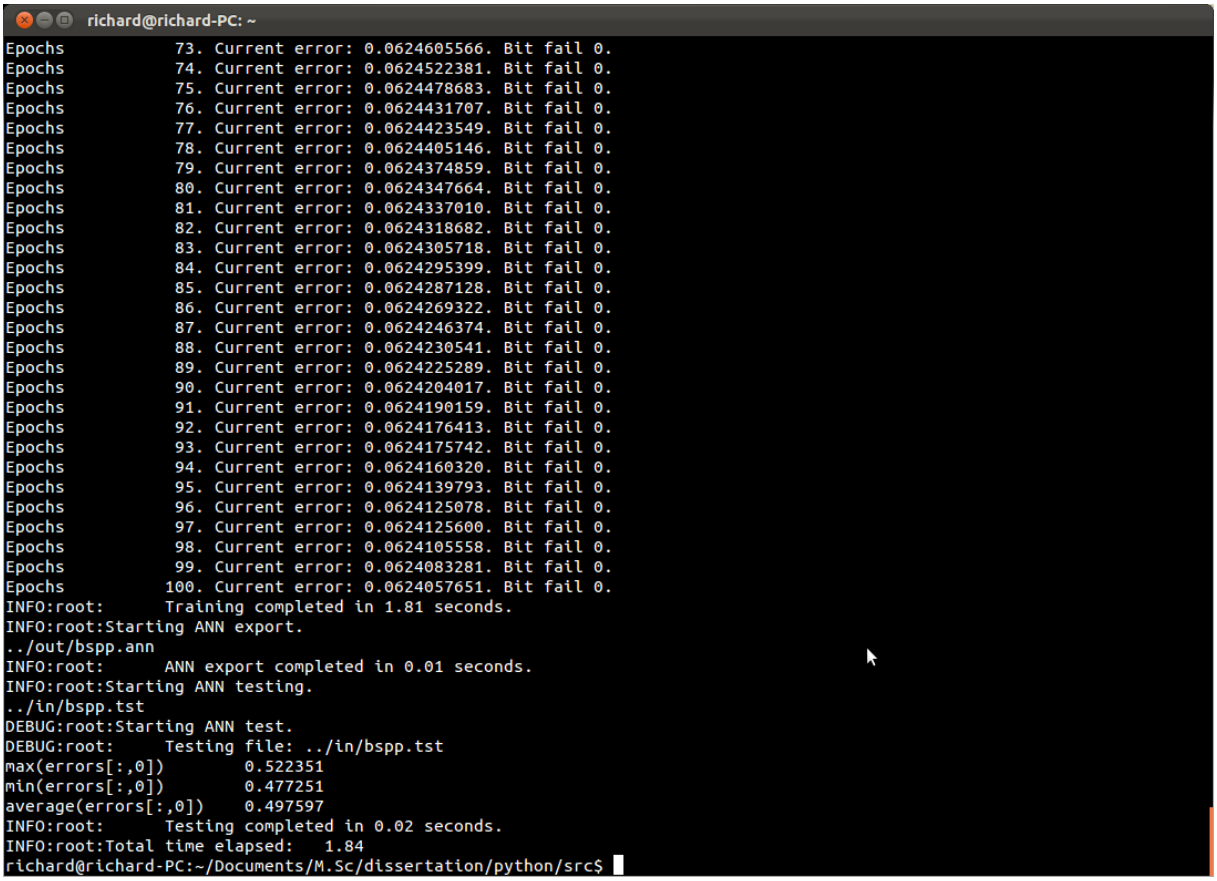

Figure 6.15: An example of Banking Sector Pairs Position testing.

6.3.2 Banking Sector Pairs Position

Single Long Period Testing

The Banking Sector Pairs Position test case single period testing results are shown in Table 6.9. Similarly to the Single Share Long Position table, the Average Error column indicates the average difference between the ANN's predicted value and the test data's realized value. The main difference between the Banking Sector Pairs Position's results and Single Share Long Position's results is that the former's Average Error is below 0.5 for all except one topology.

The lowest error value was 0.491708 for the topology of 185 hidden layers with 10 neurons per layer. The next lowest error topology has 20 layers with 20 neurons per layer, with an error of 0.493101. Considering that this implies that the ANN correctly predicted when to be in the pairs trade very slightly better than half of the test cases it had never seen before, the results are marginally better than in the Single Share Long Position test case.

Input	Hidden	Output	Layers	Max. Error	Min. Error	Ave. Error	Time [s]
185	5	1	5	0.566087	0.433307	0.499525	68.53
185	5	1	10	0.602041	0.418362	0.498767	68.28
185	10	1	5	0.547301	0.44143	0.500075	128.52
185	10	1	10	0.612166	0.392366	0.495152	129.49
185	20	1	20	0.613084	0.378556	0.493101	250.02
185	40	1	20	0.568891	0.439710	0.496796	491.14
185	20	1	40	0.560652	0.437752	0.494122	250.20
185	50	1	50	0.571395	0.425117	0.497349	614.59
185	50	1	100	0.667924	0.38337	0.497657	732.95
185	100	1	50	0.584364	0.430312	0.494264	1224.03
185	100	1	100	0.626816	0.381648	0.496929	1243.20
185	185	1	185	0.525893	0.414548	0.498371	2250.47
185	185	1	10	0.527137	0.415248	0.496913	2271.79
185	10	1	185	0.603203	0.376758	0.491708	130.95
185	100	1	500	0.554705	0.423991	0.497234	1233.86
185	500	1	100	0.520212	0.478274	0.497225	6131.03
185	500	1	500	0.50393	0.451216	0.499268	6179.52

Table 6.9: Banking Sector Pairs Position test results.

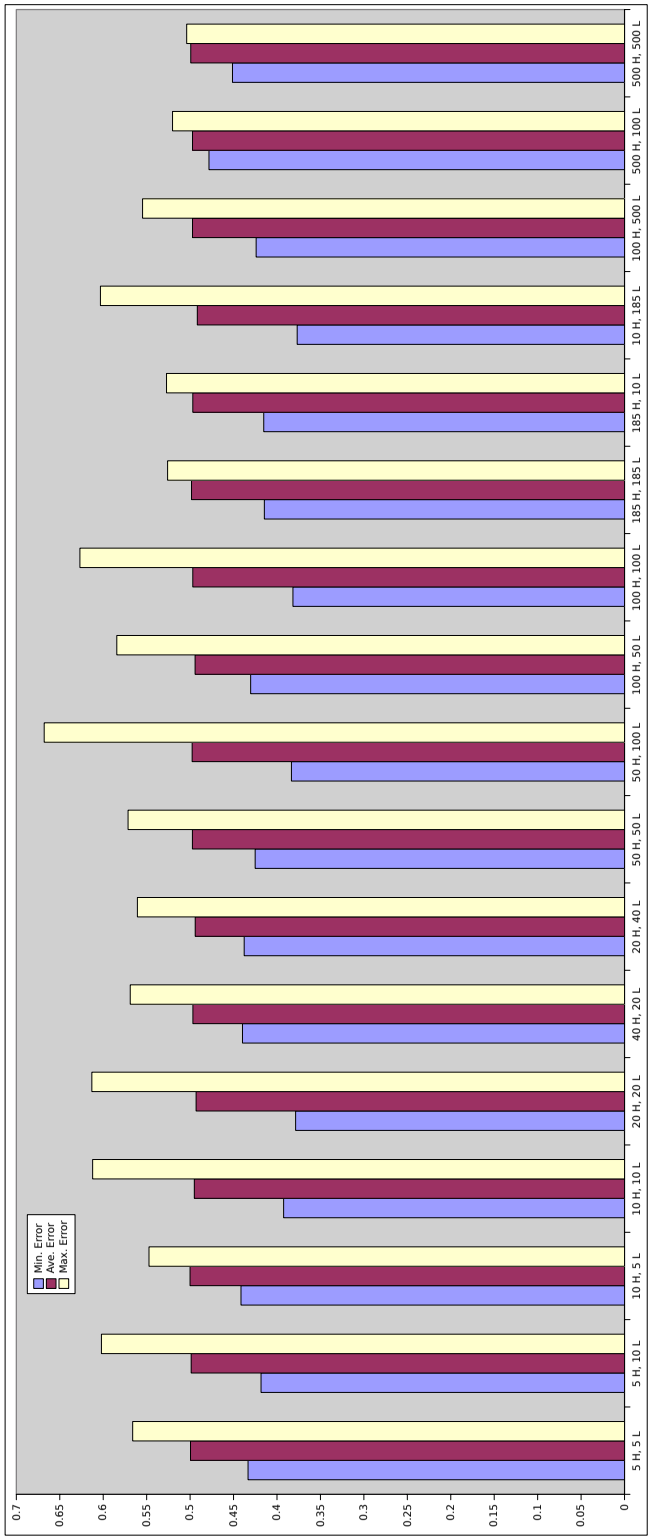

Figure 6.16: Graph of Banking Sector Pairs Position testing errors.

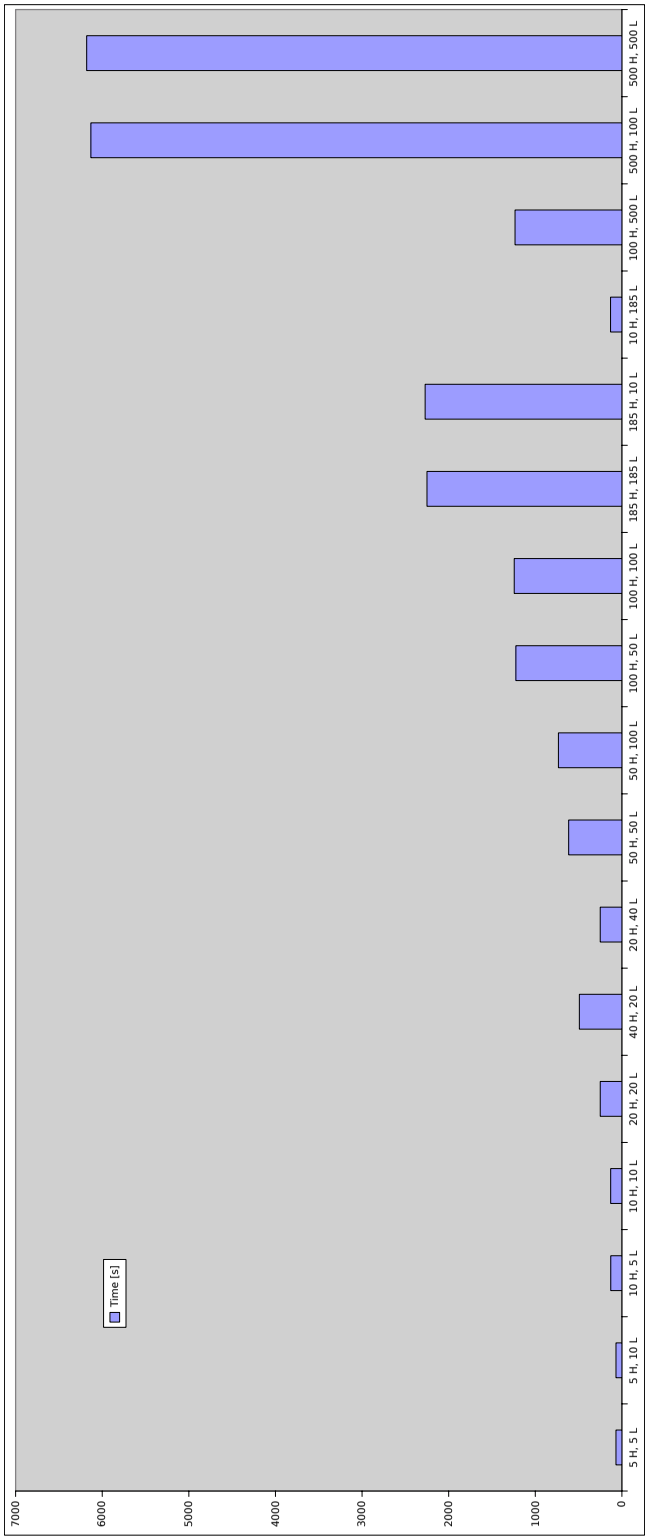

Figure 6.17: Graph of Banking Sector Pairs Position testing times.

Multi-Period Training

The testing results of the Banking Sector Pairs Position multiple testing periods are shown in Table 6.10 and Figure 6.18. There does not seem to be a particularly well performing period and on the whole very similar results to the Single Long Period experiment were noted.

Period	Input	Neurons	Output	Hidden	Epochs	Max. Error	Min. Error	Ave. Error	Time [s]
1	185	2	2	1	10	0.500374	0.499626	0.500021	0.05
1	185	2	2	1	100	0.653369	0.346663	0.496331	0.05
1	185	2	2	1	1000	0.51595	0.48405	0.499083	0.04
1	185	10	2	1	10	0.585866	0.414134	0.49506	0.05
1	185	10	2	1	100	0.634783	0.363434	0.497011	0.05
1	185	10	2	1	1000	0.943561	0.0360373	0.495343	0.05
1	185	2	2	10	10	0.530319	0.469678	0.501742	0.04
1	185	2	2	10	100	0.516149	0.483851	0.499072	0.05
1	185	2	2	10	1000	0.516202	0.483798	0.499069	0.04
1	185	10	2	10	10	0.568031	0.432648	0.496723	0.05
1	185	10	2	10	100	0.696467	0.302503	0.495838	0.05
1	185	10	2	10	1000	0.68099	0.31694	0.496719	0.05
2	185	2	2	1	10	0.548483	0.451517	0.498328	0.05
2	185	2	2	1	100	0.562807	0.436977	0.499548	0.04
2	185	2	2	1	1000	0.527963	0.472037	0.499036	0.05
2	185	10	2	1	10	0.566787	0.433213	0.497703	0.05
2	185	10	2	1	100	0.625493	0.351409	0.500747	0.04
2	185	10	2	1	1000	0.679638	0.293655	0.498073	0.05
2	185	2	2	10	10	0.566765	0.433235	0.497698	0.06
2	185	2	2	10	100	0.597722	0.390605	0.500959	0.05
2	185	2	2	10	1000	0.527999	0.472001	0.499035	0.04
2	185	10	2	10	10	0.617141	0.383364	0.503428	0.04
2	185	10	2	10	100	0.609544	0.379746	0.500412	0.05
2	185	10	2	10	1000	0.698985	0.298485	0.499744	0.04
3	185	2	2	1	10	0.534972	0.465028	0.500801	0.04
3	185	2	2	1	100	0.524432	0.475568	0.49944	0.05
3	185	2	2	1	1000	0.524419	0.475581	0.499441	0.04
3	185	10	2	1	10	0.569222	0.430758	0.498418	0.05
3	185	10	2	1	100	0.559724	0.440244	0.499643	0.05
3	185	10	2	1	1000	0.631961	0.370225	0.498878	0.04
3	185	2	2	10	10	0.544762	0.455238	0.498975	0.05
3	185	2	2	10	100	0.5244	0.4756	0.499441	0.04
3	185	2	2	10	1000	0.524486	0.475514	0.499439	0.05
3	185	10	2	10	10	0.572318	0.427682	0.498344	0.05
3	185	10	2	10	100	0.546815	0.459638	0.49939	0.04
3	185	10	2	10	1000	0.620404	0.391153	0.499933	0.04
4	185	2	2	1	10	0.55837	0.44163	0.497306	0.04
4	185	2	2	1	100	0.519759	0.480241	0.499088	0.04
4	185	2	2	1	1000	0.519815	0.480185	0.499085	0.04
4	185	10	2	1	10	0.651891	0.357492	0.506389	0.05
4	185	10	2	1	100	0.640001	0.395866	0.498374	0.04
4	185	10	2	1	1000	0.696145	0.231377	0.496918	0.05
4	185	2	2	10	10	0.554726	0.445274	0.497474	0.05
4	185	2	2	10	100	0.519598	0.480402	0.499095	0.04
4	185	2	2	10	1000	0.519591	0.480409	0.499096	0.04
4	185	10	2	10	10	0.582484	0.417516	0.496197	0.04

Period	Input	Neurons	Output	Hidden	Epochs	Max. Error	Min. Error	Ave. Error	Time [s]
4	185	10	2	10	100	1.6592	0.454344	0.504619	0.05
4	185	10	2	10	1000	0.774354	0.206528	0.489791	0.05
5	185	2	2	1	10	0.540826	0.459174	0.500942	0.04
5	185	2	2	1	100	0.519357	0.480643	0.499553	0.04
5	185	2	2	1	1000	0.519436	0.480564	0.499551	0.04
5	185	10	2	1	10	0.720831	0.279143	0.505024	0.06
5	185	10	2	1	100	0.548482	0.453596	0.498846	0.05
5	185	10	2	1	1000	0.556964	0.436858	0.50106	0.04
5	185	2	2	10	10	0.603624	0.396376	0.497609	0.05
5	185	2	2	10	100	0.583731	0.425754	0.499073	0.05
5	185	2	2	10	1000	0.903633	0.477077	0.502558	0.04
5	185	10	2	10	10	0.656587	0.343417	0.503583	0.04
5	185	10	2	10	100	0.531298	0.470329	0.499733	0.04
5	185	10	2	10	1000	0.576149	0.421787	0.498897	0.04
6	185	2	2	1	10	0.604979	0.395021	0.516014	0.03
6	185	2	2	1	100	0.584672	0.439742	0.501267	0.04
6	185	2	2	1	1000	0.578534	0.430526	0.498756	0.04
6	185	10	2	1	10	0.565162	0.431776	0.509699	0.05
6	185	10	2	1	100	0.584445	0.425754	0.499604	0.04
6	185	10	2	1	1000	0.581887	0.445655	0.498806	0.04
6	185	2	2	10	10	0.582792	0.417208	0.487371	0.04
6	185	2	2	10	100	0.603641	0.423458	0.500175	0.04
6	185	2	2	10	1000	0.671238	0.445773	0.503056	0.04
6	185	10	2	10	10	0.596051	0.403949	0.485348	0.04
6	185	10	2	10	100	0.591392	0.419732	0.497945	0.04
6	185	10	2	10	1000	0.543386	0.46779	0.497483	0.04

Table 6.10: Banking Sector Pairs Position multi-period test results.

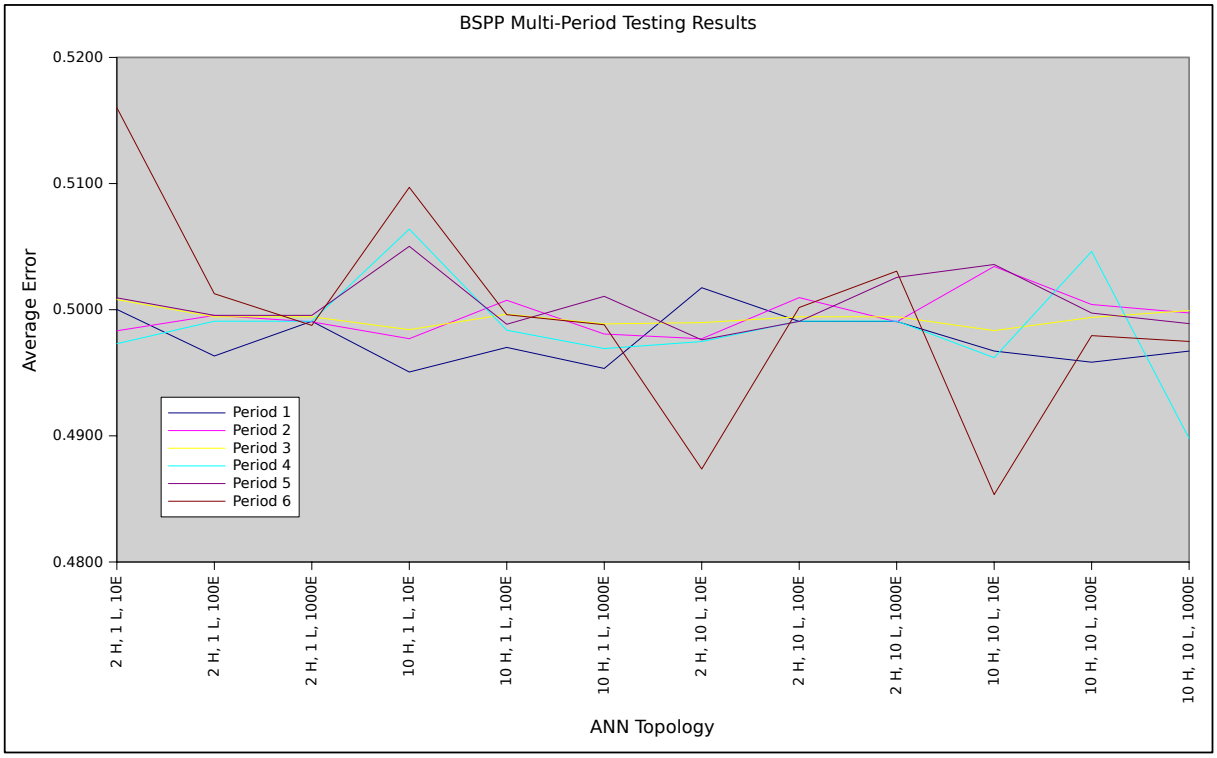

Figure 6.18: Graph of Banking Sector Pairs Position multi-period average error during testing.

6.3.3 Gold Mining Sector Pairs Position

Single Long Period Testing

The Gold Mining Sector Pairs Position test case results are shown in Table 6.11. The results of both the training and testing show similar but slightly less favourable results than for the Banking Sector Pairs Position.

Considering that nine of the topologies showed marginally lower than 0.5 errors, while the remaining eight were slightly over 0.5, it would appear that this test case also performed better than the Single Share Long Position test case. The average error over all topologies in this test case was 0.499442.

The best performing topology had 500 neurons for each of its 100 hidden layers with an error of 0.493061, however it is invalid for the purpose of this project as it had failed bits during training. The lowest error for a 0 bit fail topology was 0.498438, with 100 neurons per each of its 50 hidden layers.

Input	Hidden	Output	Layers	Max. Error	Min. Error	Ave. Error	Time [s]
185	5	1	5	0.593558	0.382161	0.499945	71.09
185	5	1	10	0.620158	0.391136	0.499071	70.57
185	10	1	5	0.617380	0.383910	0.500010	133.35
185	10	1	10	0.592176	0.416505	0.499666	131.85
185	20	1	20	0.609493	0.415265	0.499640	256.68
185	40	1	20	0.650035	0.353391	0.502142	503.67
185	20	1	40	0.592385	0.387810	0.499256	254.40
185	50	1	50	0.603557	0.395748	0.500632	614.72
185	50	1	100	0.620418	0.395773	0.500060	623.89
185	100	1	50	0.623224	0.375940	0.498438	1240.04
185	100	1	100	0.611335	0.379856	0.500475	1249.97
185	185	1	185	0.979814	0.185368	0.497024	2326.88
185	185	1	10	0.699383	0.31394	0.503321	2310.55
185	10	1	185	0.683262	0.359272	0.500166	133.50
185	100	1	500	0.680999	0.383617	0.500399	1256.44
185	500	1	100	1.0391	0.00841115	0.493061	6149.17
185	500	1	500	0.869168	0.143119	0.497712	6127.18

Table 6.11: Gold Mining Sector Pairs Position test results.

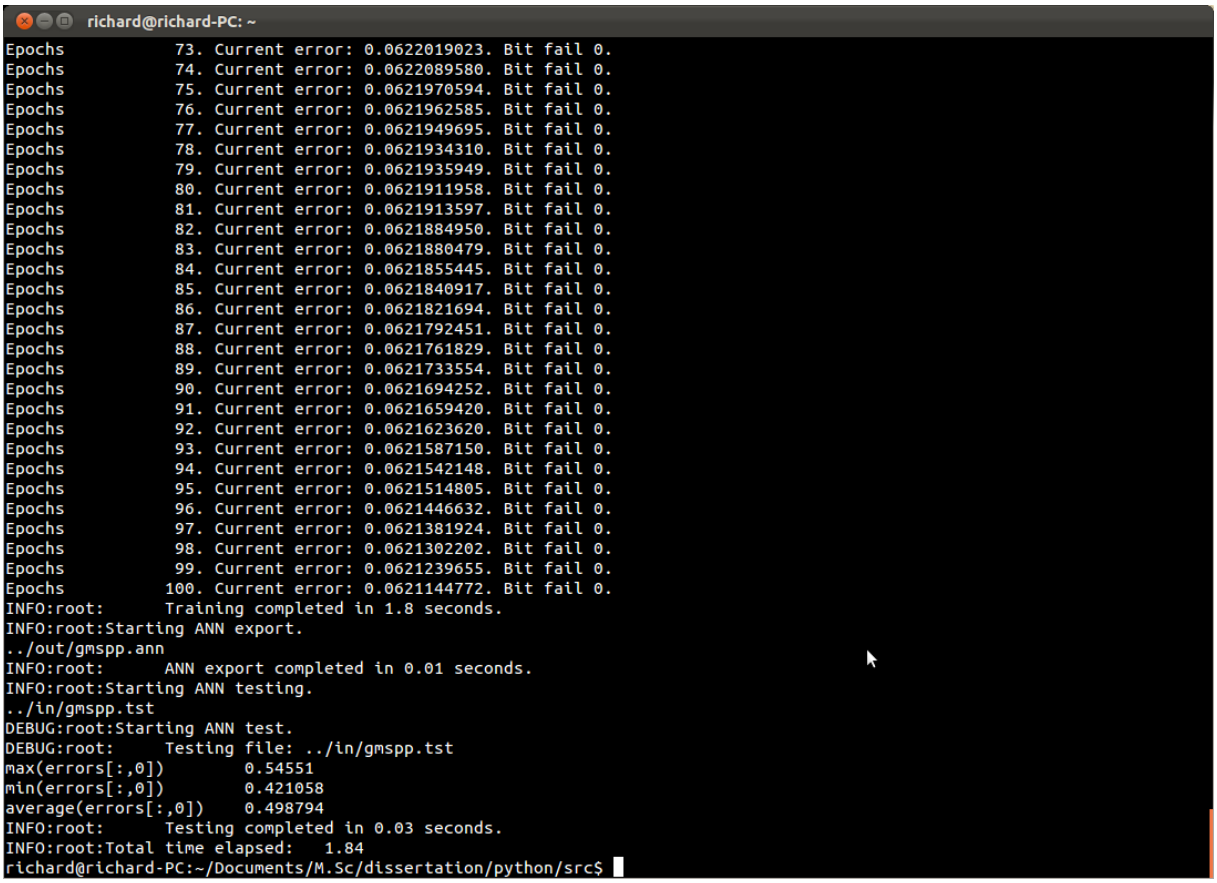

Figure 6.19: An example of Gold Mining Sector Pairs Position testing.

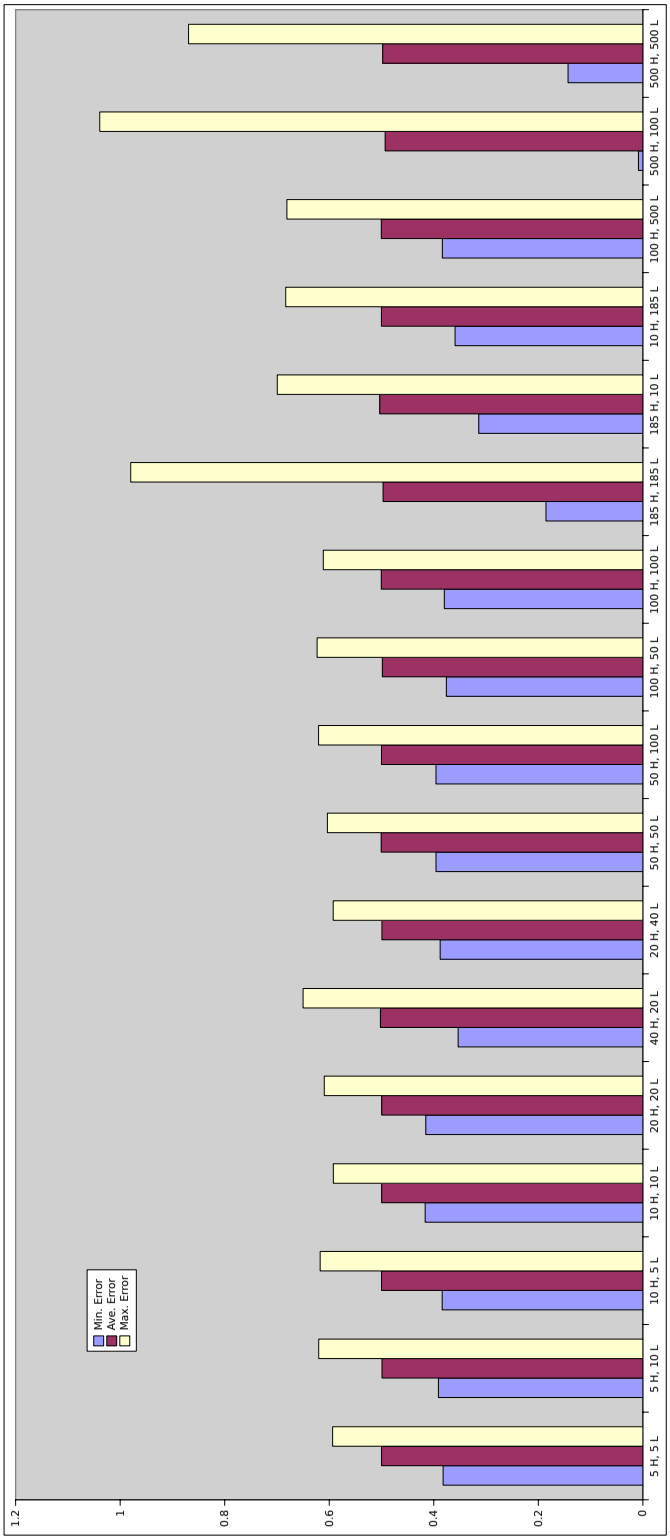

Figure 6.20: Graph of Gold Mining Sector Pairs Position testing errors.

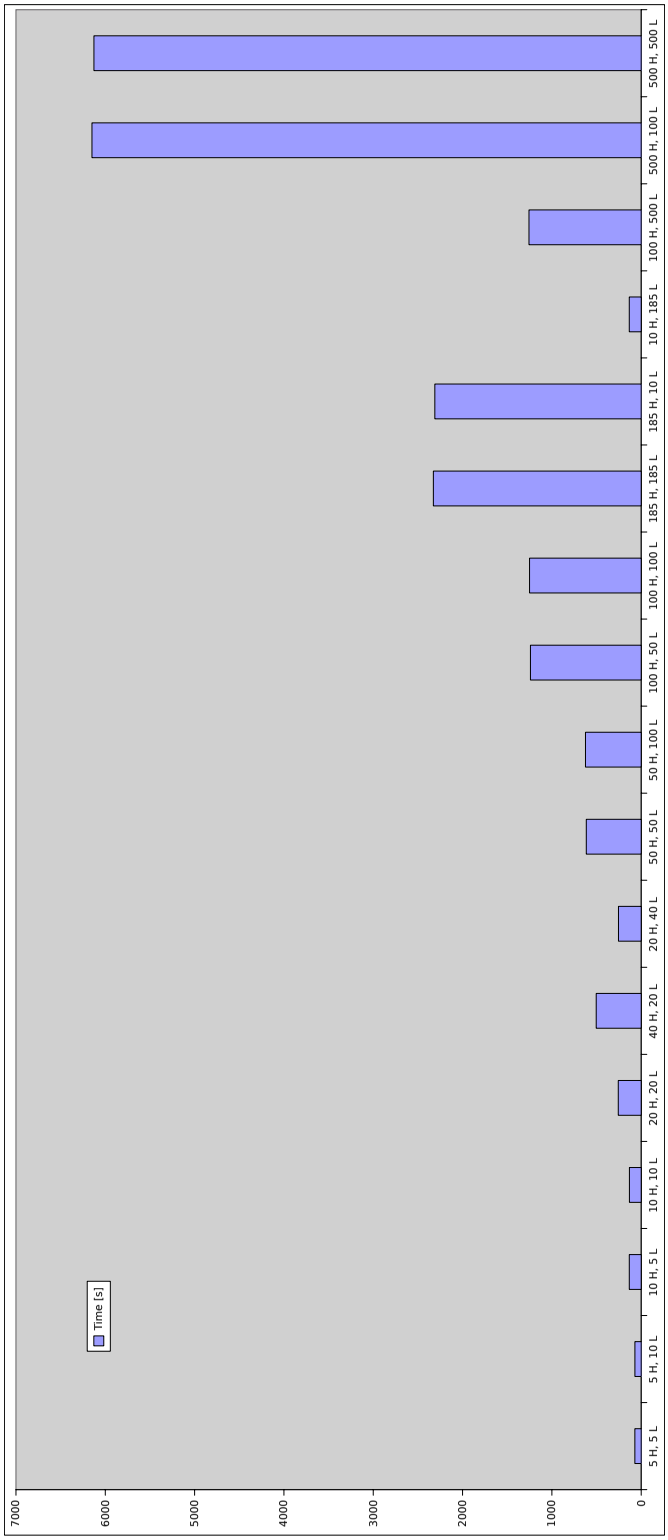

Figure 6.21: Graph of Gold Mining Sector Pairs Position testing times.

Multi-Period Training

The testing results of the Gold Mining Sector Pairs Position multiple testing periods are shown in Table 6.12 and Figure 6.22. Much like the Banking Sector Pairs Position case study, the multi-rolling period experiment produced very similar results the single long period experiment. This is notwithstanding the different topologies and training epochs. The GMSPP case study appears to have produced worse results than the BSPP case study.

Period	Input	Neurons	Output	Hidden	Epochs	Max. Error	Min. Error	Ave. Error	Time [s]
1	185	2	2	1	10	0.566548	0.433463	0.498725	0.05
1	185	2	2	1	100	0.503341	0.496659	0.500064	0.04
1	185	2	2	1	1000	0.503312	0.496688	0.500063	0.05
1	185	10	2	1	10	0.603687	0.39737	0.498067	0.04
1	185	10	2	1	100	0.739804	0.273866	0.495436	0.05
1	185	10	2	1	1000	0.727496	0.349644	0.497756	0.04
1	185	2	2	10	10	0.562003	0.437997	0.501188	0.04
1	185	2	2	10	100	0.613537	0.389579	0.497265	0.05
1	185	2	2	10	1000	0.771449	0.269489	0.496241	0.05
1	185	10	2	10	10	0.749474	0.252381	0.495293	0.05
1	185	10	2	10	100	0.602888	0.404826	0.497779	0.04
1	185	10	2	10	1000	0.885484	0.274945	0.497131	0.05
2	185	2	2	1	10	0.559556	0.440444	0.493383	0.04
2	185	2	2	1	100	0.608996	0.390767	0.496262	0.04
2	185	2	2	1	1000	0.501455	0.498545	0.499838	0.05
2	185	10	2	1	10	0.544455	0.455545	0.495058	0.05
2	185	10	2	1	100	0.594221	0.405543	0.496443	0.05
2	185	10	2	1	1000	0.65013	0.352093	0.497549	0.05
2	185	2	2	10	10	0.532117	0.467883	0.503568	0.04
2	185	2	2	10	100	0.50132	0.49868	0.499853	0.05
2	185	2	2	10	1000	0.665819	0.330259	0.493677	0.04
2	185	10	2	10	10	0.556688	0.443312	0.493698	0.05
2	185	10	2	10	100	0.600892	0.39932	0.497153	0.05
2	185	10	2	10	1000	0.644546	0.317804	0.496823	0.05
3	185	2	2	1	10	0.513888	0.486112	0.500742	0.05
3	185	2	2	1	100	0.56093	0.437615	0.497727	0.04
3	185	2	2	1	1000	0.508689	0.491311	0.499536	0.04
3	185	10	2	1	10	0.560895	0.439106	0.503272	0.04
3	185	10	2	1	100	0.61923	0.379622	0.49881	0.05
3	185	10	2	1	1000	0.688982	0.288993	0.5017	0.05
3	185	2	2	10	10	0.590081	0.409919	0.495187	0.04
3	185	2	2	10	100	0.610841	0.386923	0.499132	0.05
3	185	2	2	10	1000	0.511997	0.101182	0.495537	0.04
3	185	10	2	10	10	0.633341	0.366684	0.507141	0.05
3	185	10	2	10	100	0.577075	0.422853	0.498642	0.04
3	185	10	2	10	1000	0.588447	0.409907	0.502811	0.05
4	185	2	2	1	10	0.5259	0.4741	0.500199	0.05
4	185	2	2	1	100	0.5161	0.4839	0.500124	0.05
4	185	2	2	1	1000	0.516214	0.483786	0.500125	0.04
4	185	10	2	1	10	0.625711	0.374289	0.500988	0.04
4	185	10	2	1	100	0.538283	0.463259	0.499138	0.05
4	185	10	2	1	1000	0.643223	0.426985	0.500587	0.05
4	185	2	2	10	10	0.567492	0.432508	0.500519	0.05

Period	Input	Neurons	Output	Hidden	Epochs	Max. Error	Min. Error	Ave. Error	Time [s]
4	185	2	2	10	100	0.682545	0.423544	0.500033	0.05
4	185	2	2	10	1000	1.15763	0.42571	0.503476	0.05
4	185	10	2	10	10	0.573214	0.426786	0.500619	0.05
4	185	10	2	10	100	0.646517	0.415225	0.499588	0.05
4	185	10	2	10	1000	0.607424	0.387984	0.495239	0.04
5	185	2	2	1	10	0.593659	0.406341	0.501441	0.04
5	185	2	2	1	100	0.516539	0.483461	0.499746	0.04
5	185	2	2	1	1000	0.516599	0.483401	0.499745	0.05
5	185	10	2	1	10	0.534754	0.465246	0.499465	0.05
5	185	10	2	1	100	0.591066	0.373471	0.497654	0.05
5	185	10	2	1	1000	0.657851	0.340761	0.502477	0.04
5	185	2	2	10	10	0.531451	0.468549	0.499516	0.05
5	185	2	2	10	100	0.516578	0.483422	0.499745	0.05
5	185	2	2	10	1000	0.516386	0.483614	0.499748	0.04
5	185	10	2	10	10	0.630237	0.369763	0.502033	0.04
5	185	10	2	10	100	0.568564	0.353219	0.49775	0.05
5	185	10	2	10	1000	0.622249	0.362928	0.496533	0.05
6	185	2	2	1	10	0.528829	0.471171	0.5	0.04
6	185	2	2	1	100	0.520802	0.479198	0.5	0.04
6	185	2	2	1	1000	0.520653	0.479347	0.5	0.04
6	185	10	2	1	10	0.528802	0.471198	0.5	0.04
6	185	10	2	1	100	0.564253	0.434753	0.499379	0.04
6	185	10	2	1	1000	0.616913	0.399368	0.501695	0.04
6	185	2	2	10	10	0.586817	0.413183	0.5	0.04
6	185	2	2	10	100	0.520772	0.479228	0.5	0.04
6	185	2	2	10	1000	0.520714	0.479286	0.5	0.04
6	185	10	2	10	10	0.522457	0.477543	0.500002	0.04
6	185	10	2	10	100	0.563421	0.43608	0.499612	0.04
6	185	10	2	10	1000	0.59178	0.405796	0.500189	0.04

Table 6.12: Gold Mining Sector Pairs Position multi-period test results.

6.4 Chapter Summary

In this chapter the experimental results obtained from training the respective artificial neural networks for each test case was presented along with the output from the actual testing of these artificial neural networks. In contrast with other research, for example Kryzanowski et al, [11], who achieved success rates between 66.4% and 71.7% with their ANN implementation for stock picking, the experiments of this dissertation achieved successful predictions between 49.9% and 51.0% of the time. While the results were indicative of only slightly better results than a simple coin flip, it is necessary to note that important insights were gained that may benefit future researchers in this area. These insights are discussed in more detail in Chapter 7.

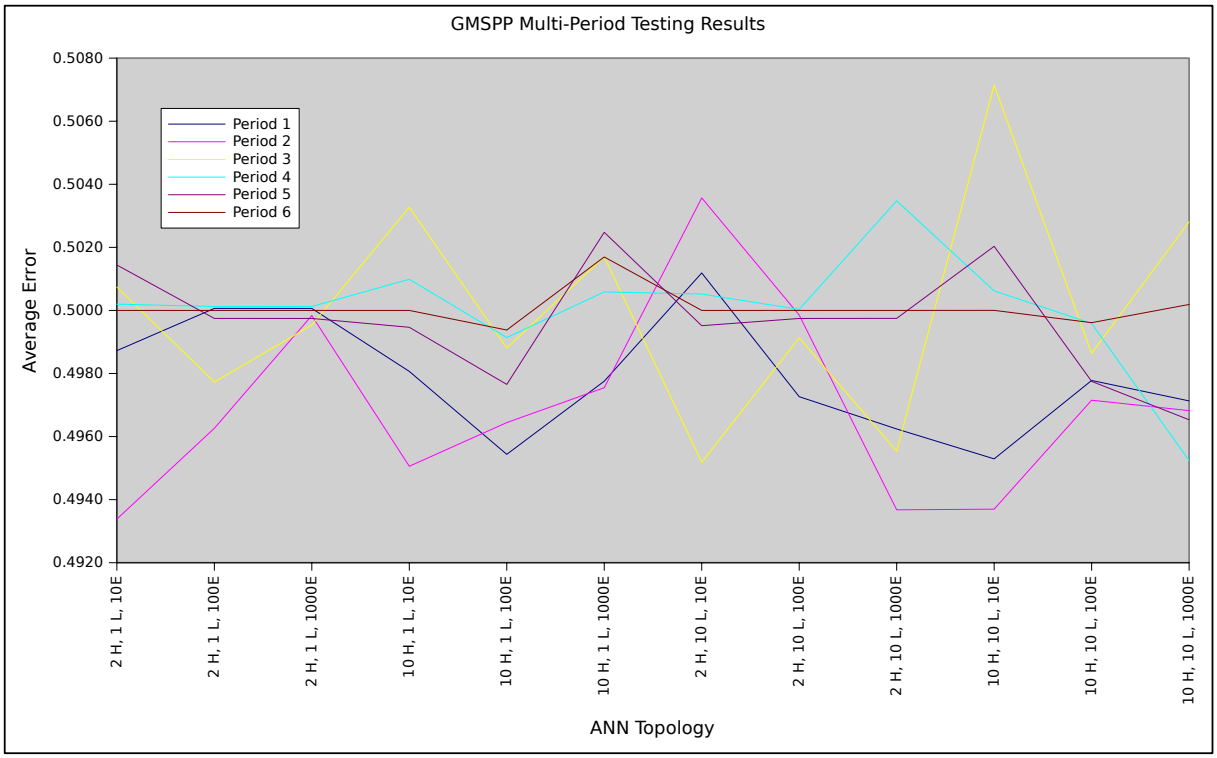

Figure 6.22: Graph of Gold Mining Sector Pairs Position multi-period average error during testing.

Chapter 7

Conclusion

7.1 Review

This chapter provides an overview of the work done and the important results from the experimental phase of the project. The first section is a brief list of the assumptions made in this project, which impacted design of the artificial neural networks as well as the design and methodology of the experiment.

This is followed by a brief review of the artificial neural network design process with a discussion on empirical training results thereof. Following this section a more traditional review of the actual experimental results if given, with explanation of these results and their implications.

A discussion section is included as a place to put the results into perspective and to discuss the measure of success of this research in terms of the viability of the use of artificial neural networks as decision making support tools in the pairs trading arena. The possible shortcomings of this study are also covered in this section.

A section on what potential future work could be done in the field of financial engineering using artificial neural networks is included and, finally, the document is concluded with a few words on the research results and their implications by answering the question posed in Chapter 1.

7.1.1 Important Assumptions

1. Stock splits in the data were catered for through rolling back the changes.

2. Sufficient data was available to sensibly train and test the artificial neural networks.
3. The changing time to maturity of the government bonds did not adversely affect the yield data, since day-on-day changes were used.
4. Trading costs were deemed negligible for the purpose of this project and as such were not included.
5. Stock dividends were also deemed negligible in terms of day on day share price changes and were not included.

7.1.2 Design

The artificial neural network experiments were designed so that different topologies could be compared during training and testing. From the results it would appear from that a very complex multi-layered network is not required to be able to produce the desired outputs, during training at least. This is corroborated by the findings of researchers that generally a single hidden layer, in other words a three layer network, can approximate any continuous mapping, [10] and [89].

Both experimental sets performed for each of the case studies appeared to indicate that a simpler or smaller network produced generally better results than larger or more complex networks.

7.1.3 Results

From the results chapter it can be seen that while the training phase was completed fairly well, the testing phase yielded results that were not quite as good as had been hoped for.

The Banking Sector Pairs Position test case performed better than the Single Share Long position and Gold Mining Sector Pairs Position test cases during testing. Considering that the Single Share Long Position test case had a more complex (two output variables versus one output variable for the pairs trading test cases) output requirement and that the Gold Mining Sector Pairs Position had some problems during the training, this does seem reasonable.

For the two Pairs Position test cases an average error of just below 0.5 seems to indicate that this specific implementation of artificial neural networks is only slightly better than a coin toss (0.5 probability of choosing the correct option). Other authors have reported successful

predictions in up to 70% test cases, [11], and as such the author had hoped to achieve similar results.

For all three cases to provide less than satisfactory results in both experimental sets, it is clear that the basic design of the experiments, data preparation or the training methods (or a combination of these crucial elements) were problematic and would require adaptation to produce better results.

7.1.4 Discussion

Zhang and Hu, [14], mention the inherent problems with over-fitting or over-training the ANN and also mention the importance of using the appropriate architecture for the ANN. Fadlalla and Lin, [2], reinforce the importance of not over-fitting data. Hawley et al, [90], point out that a user must rely on the output as a gauge of the systems' consistency and reliability, indicating that a critical review of the input data assumptions and architecture is required.

Not all researchers have reported success with ANNs, for example White, [91], reported disappointing results and that since ANNs are based on pattern recognition, one cannot make it produce acceptable results without revisiting how patterns are expected to be recognised.

The underwhelming results from this research seem to have been a product of various factors mentioned above. Some of these are listed below along with their possible causes and potentially mitigating suggestions.

Relatively little training data. It might be that ten years of daily data is insufficient for truly training an artificial neural network with the proper complexity to reliably predict the correct portfolio management decisions.

Possible neural network over-training. Given that the ANNs were trained for 10000 epochs, regardless of when the first successful training pass with no failed outputs (0 failed bits) occurred, it may be beneficial to rather incrementally increase the training epochs until the first successful training pass with all outputs correct.

The training data may have been insufficiently rich. Perhaps simply using a single set of day-on-day changes to predict market variable movements is not sufficient and additional historical data needs to be worked into the training data on a per day basis. In other words, each set of daily data may need to contain two or more days' worth of day-on-day changes.

The normalisation method of input data could have suppressed information. Considering that the input data needed to be normalised across numbers of various orders of magnitude, some information could have had a lesser impact than needed for satisfactory results. Perhaps by normalising the input in a different way this information might be more prominent to the ANN.

The chosen number of market variables could have introduced unnecessary complexity. It may well be that considerably less input variables are required, though it would require some study to identify which input variables are still required to produce satisfactory results.

Future work would need to address at least the following areas of interest that were not addressed in this project:

1. Optimal network topology for a given application;
2. Optimal amount of training epochs (number of times the training set is passed through the network). This is to minimise training time and avoid over-training;
3. Include more than one day's change to make a prediction. In other words, attempt to predict tomorrow's market state based on more than just the comparative change from yesterday to today;
4. Investigate different normalisation schemes; and
5. Potentially use less input variables, possibly using principal component analysis or a similar method to remove unrelated input variables.

7.2 Future Work

For related future research projects, a myriad of possibilities are created by considering the wide area of potential artificial neural network implementations. By slightly altering how data is fed into the artificial neural network and the topology of the actual network, vastly different applications are possible, for example:

- yield curve generation;
- credit risk analysis;
- decision making support in portfolio management;

- algorithmic and quantitative trading;
- analysis of illiquid instruments;
- risk management applications;
- identification of potential pairs trading shares; and
- any area where there is not enough data, no closed form solutions or where 'gut feel' is used due to the non-linear nature of available data.

7.3 Conclusion

While the experiments in this project have not produced very successful artificial neural networks, the work done by the author and the successes of various practitioners in the financial markets, the proven non-linear problem solving ability of artificial neural networks and the excellent results in other fields that there is definitely scope for further research into the use of artificial neural network in financial engineering.

The lack of results that strongly confirm the benefits of artificial neural network use in pairs trading decision support, however, is probably due to the experimental design and data preparation in this project. While one method has been examined, many more are possible that could yield more satisfying results.

Considering that most other ANN based analyses focus on a small number of shares and in this research the focus was on 165 shares and some other market data, it can be argued that the equity share market on the JSE is efficient due to the fact that the 165 shares that were considered during this research with no a priori information resulted in no obvious profitable trading patterns using the developed ANN.

Given the empirical evidence and the discussion above, it is now possible to answer the question in Chapter 1: 'Are artificial neural networks a viable tool applied to pairs trading in the South Africa market?'. The answer is, however, a bit vague in that it certainly appears possible, but the method examined in this research does not provide satisfactory results.

Bibliography

[1] A.-P. Refenes, *Neural Networks in the Capital Markets*, 1st ed. 111 River Street, Hoboken, NJ 07030-5774, USA: John Wiley & Sons, 1995.

[2] A. Fadlalla and C.-H. Lin, "An analysis of the applications of neural networks in finance," *INTERFACES*, vol. 31, no. 4, pp. 112 – 122, July - October 2001. [Online]. Available: http://www.jstor,org/stable/25062724

[3] J. Johnson and A. Whinston, "Advances in artificial neural networks in economics, finance and management," *Finance and Management*, vol. 1, 1994.

[4] J. Smith, "A neural network - could it work for you?" *Financial Executive*, vol. 6, no. 3, 1990.

[5] R. Gençay, "Optimization of technical trading strategies and the profitability in security markets," *Economics Letters*, vol. 59, pp. 249 – 254, 1998.

[6] R. Trippi and E. Turban, *Neural Networks in Finance and Investing*. Porbus Publishing Company, Chicago, Illinois, 1992.

[7] K. Mehrotra, C. Mohan, and S. Ranka, *Elements of Artificial Neural Networks*. Massachusetts Institute of Technology, 2000.

[8] B. Yegnanarayana, *Artificial Neural Networks*. New Delhi: Pearson Prentice Hall of India, 2006.

[9] W.-H. Steeb, *The Nonlinear Workbook*. World Scientific, 1999.

[10] I. Kaastra and M. Boyd, "Designing a neural network for forecasting financial and economic time series," *Neurocomputing*, vol. 10, pp. 215 – 236, 1995.

[11] L. Kryzanowski, M. Galler, and D. Wright, "Using artificial neural networks to pick stocks," *Financial Analysts Journal*, vol. 49, no. 4, pp. 21 – 27, July - August 1993. [Online]. Available: http://www.jstor,org/stable/4479664

[12] *Neural Network*, Wikipedia, Accessed 2010/03/16. [Online]. Available: http://en.wikipedia.org/wiki/Neural_network

[13] A. Blais and D. Mertz, *An introduction to neural networks*, IBM, Accessed 2010/03/16. [Online]. Available: http://www.ibm.com/developerworks/library/l-neural/

[14] G. Zhang and M. Hu, "Neural network forecasting of the british pound/us dollar exchange rate," *International Journal of Management Science*, vol. 26, no. 4, pp. 495 – 506, 1998.

[15] *Types of Artificial Neural Networks*, Wikipedia, Accessed 2010/03/16. [Online]. Available: http://en.wikipedia.org/wiki/Types_of_artificial_neural_networks

[16] K. Priddy and P. Keller, *Artificial Neural Networks: An Introduction*. Bellingham, Washington, 98227-0010 USA: SPIE - The International Society of Optical Engineering, 2005.

[17] P. Braspenning, T. Thuijsman, and A. Weijters, *Artificial Neural Networks: An Introduction to Theory and Practice*. Springer-Verlag Berlin Heidelberg, 1995.

[18] D. Graupe, *Principals of Artificial Neural Networks*, 2nd ed. 5 Toh Tuck Link, Singapore, 596224: World Scientific, 2007.

[19] N. Karayiannis and A. Venetsanopoulos, *Artificial Neural Networks: Learning Algortihms, Performance Evaluation and Applications*. Post Office Box 322, 3300 AH Dordrecht, The Netherlands: Kluwer Academic Publishers Group, 1993.

[20] F. Scarsellia and A. Tsoib, "Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results," *Neural Networks*, vol. 11, no. 1, pp. 15–37, 1998.

[21] C. Kuan and T. Liu, "Forecasting exchange rates using feedforward and recurrent neural networks," *Journal of Applied Econometrics*, vol. 10, no. 4, pp. 347–364, 1995.

[22] M. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE Journal*, vol. 37, no. 2, pp. 233–243, 2004.

[23] L. Ma and K. Khorasani, "Facial expression recognition using constructive feedforward neural networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 3, pp. 1588–1595, 2004.

[24] K. Venugopal, R. Sudhakar, and A. Pandya, "On-line learning control of autonomous underwater vehicles using feedforward neural networks," *IEEE Journal of Oceanic Engineering*, vol. 17, no. 3, pp. 308–319, 1992.

[25] M. Hassoun, *Fundamentals of Artificial Neural Networks*. Massachusetts Institute of Technology, 1995.

[26] C. Franke and R. Schaback, "Solving partial differential equations by collocation using radial basis functions," *Applied Mathematics and Computation*, vol. 93, no. 1, pp. 73–82, 1998.

[27] S. Chen, B. Mulgrew, and P. Grant, "A clustering technique for digital communications channel equalization using radial basis function networks," *IEEE Transactions on Neural Networks*, vol. 4, no. 4, pp. 570–590, 1993.

[28] M. Zerroukat, H. Power, and C. S. Chen, "A numerical method for heat transfer problems using collocation and radial basis functions," *International Journal for Numerical Methods in Engineering*, vol. 42, no. 7, pp. 1263–1278, 1998.

[29] B. Morse, T. Yoo, P. Rheingans, D. Chen, and K. Subramanian, "Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions," *Proceedings of the International Conference on Shape Modeling and Applications*, 2001.

[30] R. Buhr, "Feature recognition in 3d surface models using self-organizing maps," Master's thesis, University of Johannesburg, 2003. [Online]. Available: https://ujdigispace.uj.ac.za/handle/10210/1729

[31] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. Lander, and T. Golub, "Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 96, no. 6, pp. 2907–2912, 1999.

[32] S. Kaski, "Data exploration using self-organizing maps," Master's thesis, Finnish Academy of Technology, 1997. [Online]. Available: http://users.ics.aalto.fi/sami/thesis/thesis.html

[33] P. Campadellia, D. Medicia, and R. Schettini, "Color image segmentation using hopfield networks," *Image and Vision Computing*, vol. 15, no. 3, pp. 161–166, 1997.

[34] K. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4–27, 1990.

[35] J. Paik and A. Katsaggelos, "Image restoration using a modified hopfield network," *IEEE Transactions on Image Processing*, vol. 1, no. 1, pp. 49–63, 1992.

[36] H. Larochelle and Y. Bengio, "Classification using discriminative restricted boltzmann machines," *Proceedings Of The 25th International Conference On Machine Learning*, pp. 536–543, 2008.

[37] E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. John Wiley & Sons, 1988.

[38] T. Oyama, "Fault section estimation in power system using boltzmann machine," *Proceedings of the Second International Forum on Applications of Neural Networks to Power Systems*, pp. 3–8, 1993.

[39] R. Reed and R. Marks, *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. MIT Press, 1998.

[40] S. Mirmirani and H. Li, "Gold price, neural netowrks and genetic algorithm," *Computational Economics*, vol. 23, no. 2, pp. 193 – 200, March 2004.

[41] N. Swanson and H. White, "A model-selection approach to assessing the information in the term structure using linear models and artificial neural networks," *Journal of Business and Economic Statistics*, vol. 13, no. 3, pp. 265 – 275, July 1995. [Online]. Available: http://www.jstor,org/stable/1392186

[42] P. Rosenbloom, *On Computing: The Fourth Great Scientific Domain*. MIT Press, 2013.

[43] G. Grudnitski and L. Osburn, "Forecasting s&p and gold futures prices: An application of neural networks," *The Journal of Futures Markets*, vol. 13, no. 6, pp. 631 – 643, September 1993.

[44] O. Omidvar, *Progress In Neural Networks*. Intellect Books, 1995.

[45] J. Anderson, *An Introduction To Neural Networks*. MIT Press, 1995.

[46] *Delta Rule*, Wikipedia, Accessed 2010/03/16. [Online]. Available: http://en.wikipedia.org/wiki/Delta_rule

[47] J. Kamruzzaman, R. Begg, and R. Sarker, *Artificial Neural Networks in Finance and Manufacturing*. 3 Henrietta Street, Covent Garden, London, WC2E8LU: Idea Group Publishing, 2006.

[48] C. Kuo and A. Reitsch, "Neural networks vs. conventional methods of forecasting," *The Journal of Business Forecasting Methods & Systems*, vol. 14, no. 4, pp. 17 – 25, 1995/1996.

[49] J. Denton, "How good are neural networks for casual forecasting?" *The Journal of Business Forecasting Methods & Systems*, vol. 14, no. 2, pp. 17 – 23, 1995.

[50] J. McClelland and D. Rumelhart, *Explorations in Parallel Distributed Processing*. MIT Press, Cambridge, MA, 1988.

[51] T. Masters, *Practical Neural Network Recipes in C++*. Academic Press, New York, 1993.

[52] R. Lippman, "An introduction to computing with neural nets," *IEEE ASSP Magazine*, vol. 4, no. 2, pp. 4–22, 1989.

[53] M. Odom and R. Sharda, "A neural network model for bankruptcy prediction," *Proceedings of the IEEE International Conference on Neural Networks*, pp. 163–168, 1990.

[54] L. Yu, S. Wang, and K. Lai, *Foreign-Exchange-Rate Forecasting with Artificial Neural Networks*. 233 Spring Street, New York, NY 10013, USA: Springer-Science+Business Media, LLC, 2007.

[55] G. Zhang, B. Patuwo, and M. Hu, "Forecasting with artificial neural networks: The state of the art," *International Journal of Forecasting*, vol. 4, pp. 35 – 62, 1997.

[56] R. V. Eyden, *The Application Of Neural Networks In The Forecasting Of Share Prices*. Finance and Technology Publishing, 1995.

[57] *Neural Network Toolbox*, Accessed 2013/03/05. [Online]. Available: http://www.mathworks.com/products/neural-network/

[58] *Stuttgart Neural Network Simulator*, Accessed 2013/03/05. [Online]. Available: http://www.ra.cs.uni-tuebingen.de/SNNS/

[59] *Python Programming Language – Official Website*, Accessed 2009/02/05. [Online]. Available: http://www.python.org/

[60] *SciPy*, Accessed 2009/02/05. [Online]. Available: http://www.scipy.org/

[61] *Fast Artificial Neural Network Library*, Fast Artificial Neural Network Library, Accessed 2010/03/16. [Online]. Available: http://leenissen.dk/fann/

[62] *Ubuntu Home Page*, Ubuntu Linux, Accessed 2009/02/05. [Online]. Available: http://www.ubuntu.com/

[63] A. Brabazon and M. O'Neill, *Natural Computing In Computational Finance*. Springer, 2009.

[64] *Trading Strategy*, Wikipedia, Accessed 2011/03/01. [Online]. Available: http://en.wikipedia.org/wiki/Trading_strategy

[65] F. Hamzei, *Master Traders: Strategies For Superior Returns From Today's Top Traders*. John Wiley & Sons, 2010.

[66] R. Pardo, *The Evaluation And Optimization Of Trading Strategies*. John Wiley & Sons, 2011.

[67] M. Ruggiero, *Cybernetic Trading Strategies: Developing A Profitable Trading System With State-of-the-Art Technologies*. John Wiley & Sons, 1997.

[68] G. Vidyamurthy, *Pairs Trading: Quantitative Methods and Analysis*. John Wiley & Sons, Hoboken, New Jersey: John Wiley & Sons, 2004.

[69] *Spread Trade*, Wikipedia, Accessed 2011/03/01. [Online]. Available: http://en.wikipedia.org/wiki/Spread_trade

[70] P. Kaufman, *Alpha Trading: Profitable Strategies That Remove Directional Risk*. John Wiley & Sons, 2011.

[71] *A Basic Introduction to Pairs Trading*, TradingMarkets, Accessed 2011/03/01. [Online]. Available: http://www.tradingmarkets.com/.site/stocks/how_to/articles/-76543.cfm

[72] D. Ehrman, *THe Handbook of Pairs Trading: Strategies Using Equities, Options & Futures*. John Wiley & Sons, Hoboken, New Jersey: John Wiley & Sons, 2006.

[73] A. Pole, *Statistical Arbitrage: Algorithmic Trading Insights and Techniques*. John Wiley & Sons, Hoboken, New Jersey: John Wiley & Sons, 2007.

[74] *Pairs Trade*, Wikipedia, Accessed 2011/03/01. [Online]. Available: http://en.wikipedia.org/wiki/Pairs_trading

[75] W. G. E. Gatev and K. Rouwenhorst, "Pairs trading: Performance of a relative value arbitrage," *The Review of Financial Studies*, no. 19, pp. 797–827, 2006.

[76] B. Do and R. Faff, "Does simple pairs trading still work?" *Financial Analysts Journal*, no. 66, pp. 83–95, 2010.

[77] ——, "Are pairs trading profits robust to trading costs?" *The Journal of Financial Research*, no. XXXV, pp. 261–287, 2012.

[78] H. Mashele, S. Terblanche, and J. Venter, "Pairs trading on the Johannesburg stock exchange," *Investment Analysts Journal*, no. 78, pp. 13–26, 2013.

[79] E. Leshik and J. Cralle, *An Introduction to Algorithmic Trading: Basic to Advanced Strategies*. John Wiley & Sons, Hoboken, New Jersey: John Wiley & Sons, 2011.

[80] M. Sherald, *Neural Network Pair Trading*, Technical Analysis of Stocks & Commodities, Accessed 2011/03/01. [Online]. Available: http://www.traders.com/documentation/FEEDbk_docs/2010/02/Sherald.html

[81] *NeuroShell Trader*, NeuroShell, Accessed 2012/05/15. [Online]. Available: http://www.neuroshell.com/learnmore.asp

[82] *Neural Networks in Trading: Why One Pair and Not Another?*, Mechanical Forex, Accessed 2011/03/13. [Online]. Available: http://mechanicalforex.com/2011/04/neural-networks-in-trading-why-one-pair-and-not-another.html

[83] *Wilmott Forums: Question about a neural network*, Wilmott, Accessed 2011/03/15. [Online]. Available: http://www.wilmott.com/messageview.cfm?catid=4&threadid=80114

[84] *JSE - Bonds*, Johannesburg Stock Exchange, Accessed 2012/03/15. [Online]. Available: http://www.jse.co.za/HowToInvest/Bonds.aspx

[85] *BESA*, Bond Exchange of South Africa, Accessed 2012/03/15. [Online]. Available: http://www.bondexchange.co.za

[86] *JSE - Home*, Johannesburg Stock Exchange, Accessed 2012/03/15. [Online]. Available: http://www.jse.co.za/Home.aspx

[87] F. Fernández-Rodríguez, C. González-Martel, and S. Sosvilla-Rivero, "On the profitability of technical trading rules based on artificial neural networks: Evidence from the Madrid stock market," *Economics Letters*, vol. 69, pp. 89 – 94, 2000.

[88] S. Kang, *An Investigation Of The Use Of Feedforward Neural Networks For Forecasting*. Kent State University, Kent, Ohio, 1991.

[89] R. Hech-Nielsen, *Neurocomputing*. Addison Wesley, 1989.

[90] D. Hawley, J. Johnson, and D. Raina, "Artificial neural systems: A new tool for financial decision making," *Financial Analysts Journal*, vol. 46, no. 6, 1990.

[91] H. White, "Economic prediction using neural networks: The case of IBM daily stock returns," *IEEE International Conference on Neural Networks*, vol. II, pp. 451 – 458, 1988.

[92] *Stock*, Wikipedia, Accessed 2012/03/15. [Online]. Available: http://en.wikipedia.org/wiki/Stock

[93] J. Hull, *Options, Futures and Other Derivatives*, 7th ed. Pearson Education, Inc., Upper Saddle River, New Jersey, 07458: Pearson Prentice Hall, 2009.

[94] *Equity Market*, Investopedia, Accessed 2012/03/15. [Online]. Available: http://www.investopedia.com/terms/e/equitymarket.asp

[95] *Bank Deposits*, Investopedia, Accessed 2012/03/15. [Online]. Available: http://www.investopedia.com/terms/b/bank-deposits.asp

[96] *Johannesburg Interbank Agreed Rate - JIBAR*, Investopedia, Accessed 2012/03/15. [Online]. Available: http://www.investopedia.com/terms/j/jibarrate.asp

[97] *FTSE/JSE Africa All Share Index*, Bloomberg, Accessed 2012/03/15. [Online]. Available: http://www.bloomberg.com/quote/JALSH:IND

[98] *Foreign exchange market*, Wikipedia, Accessed 2012/03/15. [Online]. Available: http://en.wikipedia.org/wiki/Foreign_exchange_market

[99] *Foreign Exchange*, Investopedia, Accessed 2012/03/15. [Online]. Available: http://www.investopedia.com/ask/answers/08/what-is-foreign-exchange.asp

[100] *Spreads*, RiskGlossary, Accessed 2011/03/01. [Online]. Available: http://www.riskglossary.com/link/spreads.htm

[101] *Bloomberg - Business, Financial & Economic News, Stock Quotes*, Bloomberg, Accessed 2012/03/15. [Online]. Available: http://www.bloomberg.com

[102] *SA Bonds / Gilts*, Sharenet, Accessed 2012/03/15. [Online]. Available: http://www.sharenet.co.za/free/gilts.phtml

Appendix A

Financial Instruments

A.1 Overview

Some decsriptions of financial instruments too long for the glossary are included here for additional clarity.

A.2 Equity Shares

The capital stock (or stock) of a business entity represents the original capital paid into or invested in the business by its founders. It serves as a security for the creditors of a business since it cannot be withdrawn to the detriment of the creditors. Stock is different from the property and the assets of a business which may fluctuate in quantity and value. The stock of a business is divided into multiple shares. Used in the plural, stocks is often used as a synonym for shares, [92] and [93].

The equity market is the market in which shares are issued and traded, either through exchanges or over-the-counter markets. Also known as the stock market, it is one of the most vital areas of a market economy because it gives companies access to capital and investors a slice of ownership in a company with the potential to realize gains based on its future performance, [94] and [93].

A.3 Deposits

Money placed into a banking institution for safekeeping is known as a 'deposit'. Bank deposits are made to deposit accounts at a banking institution, such as savings accounts, checking accounts and money market accounts. The account holder has the right to withdraw any deposited funds, as set forth in the terms and conditions of the account. The 'deposit' itself is a liability owed by the bank to the depositor (the person or entity that made the deposit), and refers to this liability rather than to the actual funds that are deposited, [95] and [93].

In South Africa the money market rate that is used is known as the Johannesburg Interbank Agreed Rate (JIBAR). The rate comes in one-month, three-month, six-month and 12-month discount terms. The rate is determined as an average of the rates indicated by local and international banks. JIBAR is calculated as a yield and then converted into a discount. The rate is calculated daily after all of the rates are received by participating banks, [96] and [93].

A.4 Bonds

A bond is a financial instrument that promises that the borrower (a company or a government) will pay the holder (investor) interest over a period of time and repay the full amount of the loan on a predetermined maturity date, [84] and [93].

Bonds provide investors with a regular and steady income in the form of interest while preserving their initial investment amount (principal). They can help investors spread assets across different asset classes of the financial market, thereby minimising the risk of concentration in any one asset class, [84] and [93].

A.5 Equity Share Indices

The Johannesburg All Share Index (Bloomberg ticker: JALSH Index [97]) is a market capitalization weighted index. Companies included in this index make up the top 99% of the total pre-free float market capitalization of all listed companies on the Johannesburg Stock Exchange.

A.6 Foregn Exchange Rates

The foreign exchange market (forex, FX, or currency market) is a form of exchange for the global decentralized trading of international currencies. The FX market determines the relative values of different currencies. In a typical foreign exchange transaction, a party purchases a quantity of one currency by paying a quantity of another currency, [98] and [93].

In a free economy, a country's currency is valued according to factors of supply and demand. In other words, a currency's value can be pegged to another country's currency, such as the U.S. dollar, or even to a basket of currencies. A country's currency value also may be fixed by the country's government. However, most countries float their currencies freely against those of other countries, which keeps them in constant fluctuation. Foreign exchange, or Forex, is the conversion of one country's currency into that of another. The value of any particular currency is determined by market forces based on trade, investment, tourism, and geo-political risk, [99] and [93].

A.7 Spreads

In finance it generally means the difference between two related prices or other financial variables, [100] which is often used in trading activities such as a 'spread trade', [69] and [93].

Appendix B

Software

B.1 Bloomberg

Bloomberg is a company that provides a news and information service. Their primary offering provides historical and real-time data for business, financial & economic news supplemented by market data quotes, [101].

The Bloomberg tickers referred to elsewhere in this dissertation are of the form 'ABC:DE' for equities and simply 'ABC' for other instruments. The part of the ticker before the colon indicates the ticker name and the part after the colon the stock exchange it is traded on. For example BAW:SJ which is Barloworld Limited is listed on the Johannesburg Stock Exchange which is indicated by SJ.

B.2 Python, SciPy and NumPy

The programming language used for the computational aspect of this dissertation was Python, [59]. In particular, use was made of an additional extension module called Scientific Python (SciPy), [60]. The SciPy module provides the Numerical Python (NumPy) extension which is generally seen as the fundamental package needed for scientific computing with Python, [60]. Python itself is a dynamic object-oriented programming language that can be used for many kinds of software development. It offers strong support for integration with other languages and tools and comes with extensive standard libraries, [59].

B.3 Fast Artificial Neural Network Library

The Fast Artificial Neural Network Library (FANN) is a free, open source, artificial neural network library, which implements multilayer artificial neural networks in C with support for both fully connected and sparsely connected networks. Backpropagation training (RPROP, Quickprop, Batch, Incremental) and several different activation functions implemented. Cross-platform execution in both fixed and floating point are supported and it includes a framework for easy handling of training data sets. It is easy to use, versatile, well documented and fast. Bindings to more than 15 programming languages are available, [61].

Appendix C

Market Variables

C.1 Overview

This appendix contains the complete listing of market variables used in this dissertation. Each market variable is defined with its instrument type and class, with additional information as required for each instrument.

C.2 Input Risk Factor Lists

C.2.1 Equity Shares

Note that at the time of writing the Bloomberg ticker for **MVG:SJ** (New Bond Capital Ltd) had changed to **NBC:SJ**. Also note that in the table below BB Ticker refers to Bloomberg ticker.

Share Name	BB Ticker	Instrument	Class	Value Type
African Bank Investments Ltd	ABL:SJ	Share	Equity	Price
Arcelormittal South Africa Ltd	ACL:SJ	Share	Equity	Price
Acucap Properties Ltd	ACP:SJ	Share	Equity	Price
Advtech Ltd	ADH:SJ	Share	Equity	Price
Adcorp Holdings Ltd	ADR:SJ	Share	Equity	Price
Aveng Ltd	AEG:SJ	Share	Equity	Price
AECI Ltd	AFE:SJ	Share	Equity	Price
Afgri Ltd	AFR:SJ	Share	Equity	Price
African Oxygen Limited	AFX:SJ	Share	Equity	Price

Share Name	BB Ticker	Instrument	Class	Value Type
Anglo American Plc	AGL:SJ	Share	Equity	Price
Adcock Ingram Holdings Ltd	AIP:SJ	Share	Equity	Price
Allied Technologies Ltd	ALT:SJ	Share	Equity	Price
Anglo Platinum Ltd	AMS:SJ	Share	Equity	Price
Anglogold Ashanti Ltd	ANG:SJ	Share	Equity	Price
Aspen Pharmacare Holdings Lt	APN:SJ	Share	Equity	Price
African Rainbow Minerals Ltd	ARI:SJ	Share	Equity	Price
Astral Foods Ltd	ARL:SJ	Share	Equity	Price
Argent Industrial Limited	ART:SJ	Share	Equity	Price
ABSA Group Ltd	ASA:SJ	Share	Equity	Price
Assore Ltd	ASR:SJ	Share	Equity	Price
Allied Electronics Corp Ltd	ATN:SJ	Share	Equity	Price
Allied Electronics Corp Preference Share	ATNP:SJ	Share	Equity	Price
AVI Ltd	AVI:SJ	Share	Equity	Price
Avusa Ltd	AVU:SJ	Share	Equity	Price
Brait SA	BAT:SJ	Share	Equity	Price
Barloworld Ltd	BAW:SJ	Share	Equity	Price
Business Connexion Group	BCX:SJ	Share	Equity	Price
Bell Equipment Ltd	BEL:SJ	Share	Equity	Price
BHP Billiton Plc	BIL:SJ	Share	Equity	Price
Blue Label Telecoms Ltd	BLU:SJ	Share	Equity	Price
Brimstone Investment	BRN:SJ	Share	Equity	Price
Basil Read Holdings Ltd	BSR:SJ	Share	Equity	Price
Bidvest Group Ltd	BVT:SJ	Share	Equity	Price
Financiere Richemont	CFR:SJ	Share	Equity	Price
City Lodge Hotels Ltd	CLH:SJ	Share	Equity	Price
Clover Industries Ltd	CLR:SJ	Share	Equity	Price
Clicks Group Ltd	CLS:SJ	Share	Equity	Price
Combined Motor Holdings Ltd	CMH:SJ	Share	Equity	Price
Coronation Fund Managers Ltd	CML:SJ	Share	Equity	Price
Cipla Medpro South Africa	CMP:SJ	Share	Equity	Price
Comair Limited	COM:SJ	Share	Equity	Price
Capitec Bank Holdings Ltd	CPI:SJ	Share	Equity	Price
Capital Property Fund	CPL:SJ	Share	Equity	Price

Share Name	BB Ticker	Instrument	Class	Value Type
Ceramic Industries Limited	CRM:SJ	Share	Equity	Price
Cashbuild Limited	CSB:SJ	Share	Equity	Price
Capital Shopping Centre	CSO:SJ	Share	Equity	Price
Distribution & Warehousing	DAW:SJ	Share	Equity	Price
Datacentrix Holdings Ltd	DCT:SJ	Share	Equity	Price
DRDGOLD Ltd	DRD:SJ	Share	Equity	Price
Discovery Holdings Limited	DSY:SJ	Share	Equity	Price
Datatec Ltd	DTC:SJ	Share	Equity	Price
Emira Property Fund	EMI:SJ	Share	Equity	Price
EOH Holdings Ltd	EOH:SJ	Share	Equity	Price
Eqstra Holdings Ltd	EQS:SJ	Share	Equity	Price
Exxaro Resources Ltd	EXX:SJ	Share	Equity	Price
Famous Brands Ltd	FBR:SJ	Share	Equity	Price
Fortress Income Fund Ltd	FFA:SJ	Share	Equity	Price
Fountainhead Property Trust	FPT:SJ	Share	Equity	Price
FirstRand Ltd	FSR:SJ	Share	Equity	Price
Gold Fields Ltd	GFI:SJ	Share	Equity	Price
Grindrod Ltd	GND:SJ	Share	Equity	Price
Grand Parade Investments Ltd	GPL:SJ	Share	Equity	Price
Group Five Ltd	GRF:SJ	Share	Equity	Price
Growthpoint Properties Ltd	GRT:SJ	Share	Equity	Price
Harmony Gold Mining Co Ltd	HAR:SJ	Share	Equity	Price
Hosken Cons Investments Ltd	HCI:SJ	Share	Equity	Price
Hudaco Industries Ltd	HDC:SJ	Share	Equity	Price
Hulamin Ltd	HLM:SJ	Share	Equity	Price
Hospitality Property Fund	HPA:SJ	Share	Equity	Price
Hospitality Property Fund	HPB:SJ	Share	Equity	Price
Hyprop Investments Ltd	HYP:SJ	Share	Equity	Price
Iliad Africa Ltd	ILA:SJ	Share	Equity	Price
Illovo Sugar Ltd	ILV:SJ	Share	Equity	Price
Impala Platinum Holdings Ltd	IMP:SJ	Share	Equity	Price
Investec Ltd	INL:SJ	Share	Equity	Price
Investec Plc	INP:SJ	Share	Equity	Price
Imperial Holdings Ltd	IPL:SJ	Share	Equity	Price

Share Name	BB Ticker	Instrument	Class	Value Type
Invicta Holdings Ltd	IVT:SJ	Share	Equity	Price
JD Group Ltd	JDG:SJ	Share	Equity	Price
JSE Ltd	JSE:SJ	Share	Equity	Price
KAP International Holdings L	KAP:SJ	Share	Equity	Price
Kagiso Media Ltd	KGM:SJ	Share	Equity	Price
Kumba Iron Ore Ltd	KIO:SJ	Share	Equity	Price
Liberty Holdings Ltd	LBH:SJ	Share	Equity	Price
Lewis Group Ltd	LEW:SJ	Share	Equity	Price
Life Healthcare Group Holdin	LHC:SJ	Share	Equity	Price
Lonmin Plc	LON:SJ	Share	Equity	Price
Mediclinic International Ltd	MDC:SJ	Share	Equity	Price
Telimatrix	MIX:SJ	Share	Equity	Price
MMI Holdings Ltd	MMI:SJ	Share	Equity	Price
Metmar Limited	MML:SJ	Share	Equity	Price
Mondi Ltd	MND:SJ	Share	Equity	Price
Mondi Plc	MNP:SJ	Share	Equity	Price
Mr Price Group Ltd	MPC:SJ	Share	Equity	Price
Merafe Resources Limited	MRF:SJ	Share	Equity	Price
Massmart Holdings Ltd	MSM:SJ	Share	Equity	Price
Metair Investments Ltd	MTA:SJ	Share	Equity	Price
MTN Group Ltd	MTN:SJ	Share	Equity	Price
Metorex Ltd	MTX:SJ	Share	Equity	Price
Murray & Roberts Holdings	MUR:SJ	Share	Equity	Price
New Bond Capital Ltd	MVG:SJ	Share	Equity	Price
Mvelaserve Ltd	MVS:SJ	Share	Equity	Price
Nedbank Group Ltd	NED:SJ	Share	Equity	Price
Northam Platinum Ltd	NHM:SJ	Share	Equity	Price
Nampak Ltd	NPK:SJ	Share	Equity	Price
Naspers Ltd	NPN:SJ	Share	Equity	Price
Netcare Ltd	NTC:SJ	Share	Equity	Price
Oceana Group Ltd	OCE:SJ	Share	Equity	Price
Octodec Investments Ltd	OCT:SJ	Share	Equity	Price
Old Mutual Plc	OML:SJ	Share	Equity	Price
Omnia Holdings Ltd	OMN:SJ	Share	Equity	Price

Share Name	BB Ticker	Instrument	Class	Value Type
Optimum Coal Holdings	OPT:SJ	Share	Equity	Price
Palabora Mining Co Ltd	PAM:SJ	Share	Equity	Price
Petmin Ltd	PET:SJ	Share	Equity	Price
Pioneer Foods Ltd	PFG:SJ	Share	Equity	Price
Peregrine Holdings Limited	PGR:SJ	Share	Equity	Price
Phumelela Gaming & Leisure	PHM:SJ	Share	Equity	Price
Pick N Pay Stores Ltd	PIK:SJ	Share	Equity	Price
Premium Properties Ltd	PMM:SJ	Share	Equity	Price
Pinnacle Technology Hldgs	PNC:SJ	Share	Equity	Price
Pretoria Portland Cement Co	PPC:SJ	Share	Equity	Price
PSG Group Ltd	PSG:SJ	Share	Equity	Price
Royal Bafokeng Platinum Ltd	RBP:SJ	Share	Equity	Price
Rainbow Chicken Ltd	RBW:SJ	Share	Equity	Price
Raubex Group Ltd	RBX:SJ	Share	Equity	Price
Redefine Properties Ltd	RDF:SJ	Share	Equity	Price
Reinet Investments SA	REI:SJ	Share	Equity	Price
Remgro Ltd	REM:SJ	Share	Equity	Price
Resilient Property Income	RES:SJ	Share	Equity	Price
Redefine Properties International Ltd	RIN:SJ	Share	Equity	Price
Reunert Ltd	RLO:SJ	Share	Equity	Price
RMB Holdings Ltd	RMH:SJ	Share	Equity	Price
RMI Holdings	RMI:SJ	Share	Equity	Price
Sabmiller Plc	SAB:SJ	Share	Equity	Price
SA Corporate Real Estate Fun	SAC:SJ	Share	Equity	Price
Sappi Limited	SAP:SJ	Share	Equity	Price
Standard Bank Group Ltd	SBK:SJ	Share	Equity	Price
Sacoil Holding Ltd	SCL:SJ	Share	Equity	Price
Sasfin Holdings Ltd	SFN:SJ	Share	Equity	Price
Steinhoff Intl Holdings Ltd	SHF:SJ	Share	Equity	Price
Shoprite Holdings Ltd	SHP:SJ	Share	Equity	Price
Sanlam Limited	SLM:SJ	Share	Equity	Price
Santam Ltd	SNT:SJ	Share	Equity	Price
Sentula Mining Ltd	SNU:SJ	Share	Equity	Price
Sasol Ltd	SOL:SJ	Share	Equity	Price

Share Name	BB Ticker	Instrument	Class	Value Type
Super Group Ltd	SPG:SJ	Share	Equity	Price
Spar Group Limited/The	SPP:SJ	Share	Equity	Price
Stefanutti Stocks Holdings Ltd	SSK:SJ	Share	Equity	Price
Sun International Ltd	SUI:SJ	Share	Equity	Price
Spur Corp Ltd	SUR:SJ	Share	Equity	Price
Sycom Property Fund	SYC:SJ	Share	Equity	Price
Tiger Brands Ltd	TBS:SJ	Share	Equity	Price
Foschini Ltd	TFG:SJ	Share	Equity	Price
Telkom SA Ltd	TKG:SJ	Share	Equity	Price
Tongaat	TON:SJ	Share	Equity	Price
Truworths International Ltd	TRU:SJ	Share	Equity	Price
Vukile Property Fund Ltd	VKE:SJ	Share	Equity	Price
Vodacom Group (Pty) Ltd.	VOD:SJ	Share	Equity	Price
Wilson Bayly Holmes	WBO:SJ	Share	Equity	Price
Wesizwe	WEZ:SJ	Share	Equity	Price
Woolworths Holdings Ltd	WHL:SJ	Share	Equity	Price
Zeder Investments Ltd	ZED:SJ	Share	Equity	Price

Table C.1: JSE all share index constituents on 2011/06/30.

C.2.2 Currencies

Currency Name	Currency Code	Instrument	Class	Value Type
Australian Dollar	AUD	Exchange rate	Foreign Exchange	Price
Canadian Dollar	CAD	Exchange rate	Foreign Exchange	Price
Swiss Franc	CHF	Exchange rate	Foreign Exchange	Price
European Euro	EUR	Exchange rate	Foreign Exchange	Price
British Pound	GBP	Exchange rate	Foreign Exchange	Price
Japanese Yen	JPY	Exchange rate	Foreign Exchange	Price
New Zealand Dollar	NZD	Exchange rate	Foreign Exchange	Price
United States Dollar	USD	Exchange rate	Foreign Exchange	Price

Table C.2: A list of the main currencies considered for this study.

C.2.3 Short Term Interest Rates

Name	Instrument	Class	Term	Value Type
JIBAR ON	Deposit	Interest rate	Overnight	Percentage
JIBAR 1M	Deposit	Interest rate	1 months	Percentage
JIBAR 3M	Deposit	Interest rate	3 months	Percentage
JIBAR 6M	Deposit	Interest rate	6 months	Percentage
JIBAR 12M	Deposit	Interest rate	12 months	Percentage

Table C.3: Short term interest rates used in this study were sourced from JIBAR deposit instruments.

C.2.4 Medium and Long Term Interest Rates

Name	Instrument	Class	Term	Maturity Date	Value Type
R 157	Bond	Interest rate	4 years	2015/09/15	Yield
R 186	Bond	Interest rate	15 years	2026/12/21	Yield
R 201	Bond	Interest rate	3 years	2014/12/21	Yield
R 203	Bond	Interest rate	6 years	2017/09/15	Yield
R 204	Bond	Interest rate	7 years	2018/12/21	Yield
R 206	Bond	Interest rate	3 years	2014/01/15	Yield
R 207	Bond	Interest rate	9 years	2020/01/15	Yield

Table C.4: South African government bonds are included as an indication of longer term interest rates. The listed terms are given as roughly relative to 2011, the year this document was written.

The bond maturity information was sourced from Sharenet, [102].

Appendix D

Source Code

D.1 Overview

This appendix contains the complete source code developed for this dissertation. The first section contains the configuration files used as inputs for the source code presented in the second section.

It must be noted that the software was developed on a GNU/Linux workstation, to be able to use this software on Microsoft Windows the FANN libraries along with the Python wrapper library is required. In addition, SciPy needs to be installed as various modules of it is used.

On GNU/Linux the wrapper script **ann_main.py** can be called with a command line parameter for the configuration file, for example using the command **./ann_main.py ../cfg/slasp.cfg**.

Another way of invoking the software is by calling it via the Python command, this will work on both GNU/Linux and Microsoft Windows. This can be done by issuing the command **python ann_main.py [configuation file path]** at the console.

D.2 Configuration Files

slsp.cfg

```
1 [creation]
2 number_input: 185
3 number_neurons_hidden: 185
4 number_output: 2
```

```
number_layers: 2

[training]
maximum_epochs: 100
epochs_between_reports: 1
desired_error: 1e-9

[files]
training_file: ../in/slsp.trn
ann_file: ../out/slsp.ann
testing_file: ../in/slsp.tst
export_file: ../out/slsp.exp
```

bspp.cfg

```
[creation]
number_input: 185
number_neurons_hidden: 20
number_output: 1
number_layers: 20

[training]
maximum_epochs: 100
epochs_between_reports: 1
desired_error: 1e-9

[files]
training_file: ../in/bspp.trn
ann_file: ../out/bspp.ann
testing_file: ../in/bspp.tst
export_file: ../out/bspp.exp
```

gmspp.cfg

```
[creation]
number_input: 185
number_neurons_hidden: 20
number_output: 1
number_layers: 20

[training]
maximum_epochs: 100
epochs_between_reports: 1
desired_error: 1e-9

[files]
training_file: ../in/gmspp.trn
ann_file: ../out/gmspp.ann
testing_file: ../in/gmspp.tst
export_file: ../out/gmspp.exp
```

D.3 Python Source Code

ann_main.py

```
#! /usr/bin/python

"""
Richard Buhr
richard.buhr@gmail.com

This is the executable script of a module to generate, train, run and test
artificial neural networks using the FANN library.

It requires a configuration file name as command line argument and prints
test results to the terminal.
"""

# Standard module import
import csv
import logging
import numpy
import sys
import time

# Custom module import
import ann_class
import ann_export
import ann_test
import config_import

# Start the clock
start_time = time.clock()
logging.basicConfig(level=logging.DEBUG)

# Read configuration file
logging.info("Importing Artificial Neural Network (ANN) configuration.")
config_file = sys.argv[1]
ann_config = config_import.config_import(config_file)

# Create the network object
logging.info("Creating ANN.")
ann = ann_class.NeuralNetwork()
ann.set_creation_parameters(ann_config.getfloat("creation","number_input"), \
                    ann_config.getfloat("creation","number_neurons_hidden"), \
                    ann_config.getfloat("creation","number_output"), \
                    ann_config.getfloat("creation","number_layers"))

# Create the ANN instance
ann.create()
logging.info("ANN created.")

# Train the ANN
logging.info("Starting ANN training.")
```

```
50 snapshot_time = time.clock()
51 ann.set_training_parameters( ann_config.getint("training","maximum_epochs"), \
52                     ann_config.getint("training","epochs_between_reports"), \
53                     ann_config.getfloat("training","desired_error"))
54
55 print ann_config.get("files","training_file")
56 ann.train(ann_config.get("files","training_file"))
57 logging.info("\tTraining completed in %g seconds." % (time.clock() - \
58     snapshot_time))
59
60 # Export the trained ANN
61 logging.info("Starting ANN export.")
62 snapshot_time = time.clock()
63 print ann_config.get("files","ann_file")
64 ann.export(ann_config.get("files","ann_file"))
65 logging.info("\tANN export completed in %g seconds." % (time.clock() - \
66     snapshot_time))
67
68 # Test the ANN
69 logging.info("Starting ANN testing.")
70 snapshot_time = time.clock()
71 print ann_config.get("files","testing_file")
72 ann_test.ann_test(ann_config.get("files","ann_file"), \
73     ann_config.get("files","testing_file"))
74 logging.info("\tTesting completed in %g seconds." % (time.clock() - \
75     snapshot_time))
76 """
77 # Export ANN test results
78 logging.info("Exporting ANN test results.")
79 snapshot_time = time.clock()
80 print ann_config.get("files","export_file")
81 ann_export.ann_export(ann_config.get("files","export_file"))
82 logging.info("\tANN test results exported in %g seconds." % (time.clock() - \
83     snapshot_time))
84 """
85 # Stop the clock
86 end_time = time.clock()
87 logging.info("Total time elapsed:\t%g", end_time - start_time)
```

ann_class.py

```
1  """
2  Richard Buhr
3  richard.buhr@gmail.com
4
5  This is a simple class which defines a wrapper for the PyFANN (Python) wrapper
6  to FANN (C++), an open-source artificical neural network library:
7  http://leenissen.dk/fann/wp/
8  """
9
10 from pyfann import libfann
11
12 class NeuralNetwork:
13     """
```

```
14      The docstring.
15      """
16      def __init__(self):
17          """
18          Create neural network instance
19          """
20          self.ann = libfann.neural_net()
21
22      def set_creation_parameters(self, number_input = 1, \
23          number_neurons_hidden = 1, number_output = 1, number_layers = 3):
24          """
25          Set up neural network creation parameters
26          """
27          self.number_input = number_input
28          self.number_neurons_hidden = number_neurons_hidden
29          self.number_output = number_output
30          self.number_layers = number_layers
31
32      def set_training_parameters(self, maximum_epochs = 10, \
33          epochs_between_reports = 1, desired_error = 0.001):
34          """
35          Set up neural network training parameters
36          """
37          self.desired_error = desired_error
38          self.maximum_epochs = maximum_epochs
39          self.epochs_between_reports = epochs_between_reports
40
41      def create(self):
42          """
43          Create the neural network from given parameters
44          """
45          # Create the neural net
46          self.ann.create_standard_array((self.number_input, \
47              self.number_neurons_hidden, self.number_output))
48
49
50          # When choosing an activation function it is important to note that the
51          # activation functions have different range.  FANN_SIGMOID is e.g. in
52          # the 0 - 1 range while FANN_SIGMOID_SYMMETRIC is in the -1 - 1 range
53          # and FANN_LINEAR is unbound.
54          # The default activation function is FANN_SIGMOID_STEPWISE.
55
56          # Activation function options:
57          # FANN_LINEAR                 [unbound]
58          # FANN_SIGMOID                [0, 1]
59          # FANN_SIGMOID_STEPWISE       [?, ?]
60          # FANN_SIGMOID_SYMMETRIC      [-1, 1] <== seems to be best option based on
61          #                                 testing all the parameters' perfomance.
62
63          # Set hidden layer activation function
64          self.ann.set_activation_function_hidden(libfann.SIGMOID_SYMMETRIC)
65          # Set output layer activation function
66          self.ann.set_activation_function_output(libfann.SIGMOID_SYMMETRIC)
67
```

```
68      def create_from_file(self, file_name):
69          """
70          Create the neural network from a file
71          """
72          # Create the neural net
73          self.ann.fann_create_from_file(file_name)
74
75      def train(self, file_name):
76          """
77          Train the neural network
78          """
79
80          print "\n*****\n"
81          print "file_name = %s" % file_name
82          print "self.maximum_epochs = %g" % self.maximum_epochs
83          print "self.epochs_between_reports = %g" % self.epochs_between_reports
84          print "self.desired_error = %g" % self.desired_error
85          print "\n*****\n"
86
87          self.ann.train_on_file(file_name, self.maximum_epochs, \
88              self.epochs_between_reports, self.desired_error)
89
90      def export(self, file_name):
91          """
92          Export neural network to file
93          """
94          # Save the neural net to file
95          self.ann.save(file_name)
```

ann_test.py

```
1   """
2   Richard Buhr
3   richard.buhr@gmail.com
4
5   This module tests the ANN using a given test file and prints the results to the
6   terminal.
7   """
8
9   # Standard module import
10  import csv
11  import logging
12
13  from numpy import abs, array, average, empty, max, min, vstack
14  from pyfann import libfann
15
16  # Custom module import
17  import ann_export
18
19  def ann_test(ann_file, testing_file):
20      logging.debug("Starting ANN test.")
21
22      # Test that the file exists etc.
23      logging.debug("\tTesting file: %s", testing_file)
```

```
24
25      # Create neural network object from file
26      ann = libfann.neural_net()
27      ann.create_from_file(ann_file)
28
29      # Read testing file and loop through entries
30      try:
31          input_file_object = open(testing_file)
32          input_reader = csv.reader(input_file_object)
33
34          # Skip the first line, get the required length and reset the iterator
35          input_reader.next()
36          required_length_1 = len(array(map(float, input_reader.next()[0].split())))
37          required_length_2 = len(array(map(float, input_reader.next()[0].split())))
38          input_data = empty((0, required_length_1))
39          test_data = empty((0, required_length_2))
40          input_file_object.seek(0)
41          input_reader.next()
42
43          for row in input_reader:
44              row_1 = row
45              row_2 = input_reader.next()
46              row_1_array = array(map(float, row_1[0].split()))
47              row_2_array = array(map(float, row_2[0].split()))
48              input_data = vstack((input_data, row_1_array))
49              test_data = vstack((test_data, row_2_array))
50      except csv.Error:
51          sys.exit('\n\tTesting file read error!\n')
52
53      # Loop through input data
54      output_data = empty(test_data.shape)
55      for i, input_row in enumerate(input_data):
56          koos = ann.run(input_row)
57          output_data[i,:] = koos
58
59      # Run tests
60      errors = abs(test_data - output_data)
61      print "max(errors[:,0])\t%g" % max(errors[:,0])
62      print "min(errors[:,0])\t%g" % min(errors[:,0])
63      print "average(errors[:,0])\t%g" % average(errors[:,0])
64      # If there is more than one output, add additional lines like below.
65      #print "max(errors[:,1])\t%g" % max(errors[:,1])
66      #print "min(errors[:,1])\t%g" % min(errors[:,1])
67      #print "average(errors[:,1])\t%g" % average(errors[:,1])
68
69      #for z in range(0,10):
70      #    a = test_data[z,0]
71      #    b = output_data[z,0]
72      #    c = a - b
73      #    d = abs(c)
74      #    print "Required ouput = %g\tActual output data = %g\tDifference = %g\tError = %g" % (a, b, c,
```

config_import.py

```
"""
Richard Buhr
richard.buhr@gmail.com

This module imports a configuration file for use with ANN wrapper.
"""

# Standard module import
import ConfigParser
import logging
import sys

def config_import(config_file):
    logging.debug("Importing ANN configuration.")

    # Test that the file exists etc.
    logging.debug("\tConfiguration file: %s", config_file)

    # Import configuration
    configuration = ConfigParser.RawConfigParser()
    configuration.read(config_file)

    # Test for 'creation' section
    creation_error = True
    if configuration.has_section("creation"):
        # Test for section options
        if configuration.has_option("creation", "number_input"):
            if configuration.has_option("creation", "number_neurons_hidden"):
                if configuration.has_option("creation", "number_output"):
                    if configuration.has_option("creation", "number_layers"):
                        creation_error = False
                    else:
                        error_message = "Missing option: [number_layers]"
                else:
                    error_message = "Missing option: [number_output]"
            else:
                error_message = "Missing option: [number_neurons_hidden]"
        else:
            error_message = "Missing option: [number_input]"
    else:
        error_message = "Missing section: [creation]"

    if creation_error:
        logging.error(error_message)

    # Test for 'training' section
    training_error = True
    if configuration.has_section("training"):
        # Test for section options
        if configuration.has_option("training", "desired_error"):
            if configuration.has_option("training", "maximum_epochs"):
                if configuration.has_option("training", \
                    "epochs_between_reports"):
                    training_error = False
```

```
55                  else:
56                      error_message = "Missing option: [epochs_between_reports]"
57              else:
58                  error_message = "Missing option: [maximum_epochs]"
59          else:
60              error_message = "Missing option: [desired_error]"
61      else:
62          error_message = "Missing section: [training]"
63
64      if training_error:
65          logging.error(error_message)
66
67      files_error = True
68      # Test for 'files' section
69      if configuration.has_section("files"):
70          # Test for section options
71          if configuration.has_option("files", "training_file"):
72              if configuration.has_option("files", "ann_file"):
73                  if configuration.has_option("files", "testing_file"):
74                      if configuration.has_option("files", "export_file"):
75                          files_error = False
76                      else:
77                          error_message = "Missing option: [export_file]"
78                  else:
79                      error_message = "Missing option: [testing_file]"
80              else:
81                  error_message = "Missing option: [ann_file]"
82          else:
83              error_message = "Missing option: [training_file]"
84      else:
85          error_message = "Missing section: [files]"
86
87      if files_error:
88          logging.error(error_message)
89
90      if creation_error or training_error or files_error:
91          sys.exit("Configuration file error.")
92
93      return configuration
```

This document was prepared using LaTeX 2_ε.