Artificial neural networks

and their application to modelling South African market returns

By

Matthew Lee Smith

Submitted in partial fulfilment of the requirements for the degree of

Master of Science (Actuarial Science)

In the Faculty of Natural and Agricultural Sciences

University of Pretoria

Pretoria

March 2014

Abstract

The modelling technique known as Artificial Neural Networks (ANNs) is investigated. ANNs have the ability to detect and project non-linear relationships between variables. Further, they can adapt in dynamically changing environments while providing accurate results. A method of constructing ANNs in order to form a forecasting system is presented here. Further, in many of the applications studies, ANNs are fitted using crude guesses as to the efficient input parameters. In this study detailed investigations into parameter estimates are performed. In addition, ANNs and traditional models (ARIMA, seasonal smoothing, geometric Brownian motion, etc.) are constructed to forecast monthly inflation and the average monthly return on the money, bond and equity markets in South Africa from 1975 to 2010. The ANNs constructed are done through an integrated and isolated approach. The performance of the traditional and ANN models are compared. No general conclusion, as to which model is superior for all the applications considered, can be made. This suggests that ANNs perform as well as traditional models when forecasting financial markets. Further, it is found that the money market and inflation are forecast efficiently through all the models, over a single month. As the forecast period extends to three months the money market favours the traditional model. However, a forecast period of twelve months leads to the preference of ANNs in the case of the money market. Neither technique can forecast the equity or bond market accurately, as these require additional explanatory variables to those considered. As the forecast period increased, the forecast accuracy decreased for all the models. The integrated ANNs, which allow interaction between the markets, do not lead to improved forecasts which indicates that the relationships between the markets have a limited effect on the future values of the markets. Hybrid models are constructed, trained and tested for the money market and inflation. They are found to add value to traditional models when forecasting inflation but not the money market. The sensitivity of the performance of ANNs and the traditional model to different subsets of the inflation data is tested. No statistical difference between the models is found. The implementation advantages of ANNs are also described.

Key words—ANNs, ARIMA, Financial application of ANNs, Financial forecasting, Money market, Bond market, Equity market, Inflation

Supervisor	**:**	Dr FJC Beyers
Co-Supervisor	**:**	Dr JP de Villiers
Department	**:**	Insurance and Actuarial Science
Degree	**:**	Master of Science

Declaration

I, Matthew Lee Smith declare that the thesis/dissertation, which I hereby submit for the degree Master of Science (Actuarial Science) 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.

Signature:… …….

Date: 14/04/2014

Acknowledgements

There are several people without whom this study would not have been possible. First I would like to thank ABSA and the NRF for their support over the last two years. Dr Beyers and Dr de Villiers whom provided guidance and additional investigations when I required the former, not the latter. My mother who always listened to the issues I had with this work. It must have been boring for you but your perseverance and regular donations of coffee made it possible. My father whom continually convinced me that if it was easy then everyone would do it, thank you for setting the challenge. The staff of the Insurance and Actuarial Science department who attended and supported every long winded, repetitive presentation I gave on Artificial Neural Networks. Dominique for providing undeserved support, coffee breaks and information on how things should be done after they were done. My brother for believing in the work done here and commenting on thousands of graphs depicting the same information. Deon for the reading and motivation without hesitation. Alex for providing guidance and helping when dealing with computing challenges. Samuel and Francois for your reading and suggestions. Fleckie, Charlie and Tommy for understanding, supporting and all the late night companionship.

Artificial neural networks

and their application to modelling South African market returns

MSc Actuarial Science

2013/2014

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Matthew Lee Smith

Supervised by

Dr FJC Beyers

Dr JP de Villiers

Table of Contents

List of Tables

List of Figures

List of Equations

1 Introduction

Curiosity of the human mind began with the ancient philosophers about 2 300 years ago. Prized on their ability to 'understand' intelligence they attempted to define it. This was the beginning of several different philosophical movements. Philosophy of the mind diverged into various directions, ranging from Rationalism to Dualism. These philosophers had a dream that eventually human intelligence would be replicated. However, this dream began to fade as the definitions of intelligence began to diverge. It was not until 1950, on a paper by Alan Turing, that the definition of intelligence[1] was solidified. This definition allowed scientists, logicians, biologists and mathematician to begin a lifelong pursuit. This pursuit is termed Artificial Intelligence. Turing's definition of intelligence has been the inspiration of a wealth of unique and innovative mathematical and engineering concepts; the most promising being Support Vector Machines and Artificial Neural Networks. This study focuses firmly on Artificial Neural Networks (or ANNs for short).

An Artificial Neural Network (ANN) is a dynamic modelling technique. This modelling technique is part of soft computing techniques, which make up a considerable portion of modern artificial intelligence [1]. The name 'Artificial Neural Network' is derived from the origin of the model, namely, that the model attempts to replicate the functioning of the human brain [2]. However, the brain is extremely complex and virtually impossible to replicate. To be more accurate an ANN is limited to replicating the workings of the human brain for a very specific task [3]. Further, ANNs are rational agents. This indicates that they make the 'best' decision based on the available information [1].

[1] ***Turing's definition of intelligence:*** *Let there be three parties; party A, party B and party C. Suppose A and B are humans and C is a machine. Assume A is interviewing B and C over the phone. The aim of the interview by A is to determine which of B or C is the machine and which is the human. The machine is said to be intelligent if, during the interview process, A cannot differentiate between the human and the machine [4].*

1.1 Why Artificial Neural Networks?

1.1.1 A Brief History of Artificial Intelligence

In 1943 neuroscientist McCulloch and logician Pitts published the workings of the first artificial neuron [1]. This spurred Minsky and Edmonds on to create the first neural network computer in 1950 [1]. This lead to what is seen as the birth of Artificial Intelligence in 1956 when 10 researchers were brought together for two months to study and define the field of Artificial Intelligence [1]. Rosenblatt further developed McCulloch and Pitts' work and defined the perceptron in 1958 [5]. Their concept was improved two years later by Widrow and his student Hoff. Widrow and Hoff included a bias term that represents the threshold value [5]. In the period 1952-1969 there were great expectations for the field of Artificial Intelligence; however, the growth of the field did not meet expectations. Funding diminished from 1966 onwards as the field became constrained. A breakthrough came in 1982 when the first successful commercial expert system (the R 1) was implemented by the Digital Equipment Corporation [1]. This system configured orders for new computer systems, and by 1986 was saving the company $40 million per year [1]. This lead to an expansion in popularity as hundreds of these 'expert' systems were implemented. 1986 is seen as the return of the neural network as Rumelhart, Hinton and Williams published the paper '*Learning representations by back-propogation errors'* [6]. This paper provided key information to overcome previous challenges faced by neural netwoks. Since then there have been various implementations of, and developments in the field of Artificial Intelligence and, more importantly, Artificial Neural Networks.

1.1.2 Background on Artificial Neural Networks

The world is moving into a new paradigm of problems and improved solutions are required. In this new world it is no longer possible to ignore what is not understood. This new paradigm presents a unique problem, the problem of rapid change. The world can no longer be modelled using a static approach as real time data and big data sets are becoming increasingly relevant. So where do Artificial Neural Networks (ANNs) fit in? Currently, statistical models and methods require detailed knowledge of the underlying workings and restrictions for application [7]. ANNs can be applied to a range of problems without requiring the same level of knowledge of the underlying workings. Further, ANNs capture and project non-linear relationships, which occur when several points cannot be separated by a linear equation. In traditional time series forecasting models, linear relationships (through linear correlations) is predominately considered. ANNs are on the other side of the coin to statistical methods. ANNs identify and use non-linear relationships within data. This is appropriate as real-world problems are non-linear in nature [2].

ANNs have often been associated with 'black box' models. This occurs because it is difficult to fully interpret the internal workings present in ANNs. This makes the interpretation of the fitted model difficult, and requires a significant amount of time to be spent on analysing the results of ANNs. Another disadvantage is the computing time required for ANNs to 'learn' large data sets compared to traditional statistical models. However, as computing power increases, this disadvantage will fade. Further, as the complexity of ANNs increase, the number of associated parameters increase which leads to increased complexity with parameter estimation. However, this complexity is not experienced by the user, as the training algorithm associate with ANNs perform parameter optimisation automatically.

The results obtained from ANNs support the proposition that in some cases they have increased accuracy above traditional statistical models [8]. Further, this technique is still immature and there is room for discovery and improvement.

Another aspect of ANNs that make them unique is their ability to learn and evolve with experience. They are models that can be applied in dynamic environments without the need of regular redesign [7]. This makes them a prime candidate for this new dynamic world. Finally, ANNs deal well with vaguely defined problems, unfinished data, inaccuracies and uncertainty [9].

1.1.3 Implementation Considerations

Several aspects of Artificial Neural Networks (ANNs) must be considered when deciding on ANNs over other models. These aspects are described in this section.

1.1.3.1 Advantages of ANNs

Developing traditional models can be a more complex process than training ANNs. In this study ANNs are explained in depth and have complex workings, similar to other forecasting models. However, once established, ANNs can easily be used and updated by people who have little knowledge of the underlying workings. This makes ANNs attractive for people, both in and outside the fields of Actuarial Science and Statistics, who do not possess or cannot afford the required knowledge to build traditionally used models. Further, most problems in the financial sector contain a significant amount of noise or random variation. ANNs are able to deal well with random variation, as seen from the inflation application in chapter 4. The versatility of ANNs stretch beyond that of most traditional methods. The same ANN can be used to solve a range of problems, whereas, for each application a unique traditional model must be developed. This is supported by the number of traditional models developed in chapter 3 for each application. This versatility spreads through both classification and regression problems as users are able to effectively solve both types of problems with the same skill set. In addition, updating traditional models takes a significant amount of time, and if the trends associated with the data are continually evolving, may require redevelopment. ANNs are easy to update and retrain as additional trends emerge in the data. The retraining process does not require expert intervention or parameter re-estimation, making ANNs low in model maintenance costs. Further, as additional inputs are introduced in traditional models, the number of parameters that require estimation increase in proportion. Large traditional systems require a significant number of parameters to be estimated, and if little data exists, will lead to poor parameter estimations and a poor model. The number of parameters that require estimation by the user for ANNs remains constant for any number of inputs. ANNs can accommodate many input variables without an increase in complexity to the user. This is seen from the construction of the integrated ANNs in chapter 4, which contains up to 152 inputs. ANNs are fairly robust models. The optimal ANN is difficult to obtain due to the lack of developed optimisation techniques. However, an efficient ANN is not difficult to obtain (as demonstrated in chapter 4), which results in similar accuracy to that of traditional models. Thus, not obtaining the optimal ANN structure does not necessarily lead to poor model accuracy. Complex statistical programs are not required for ANNs to be used as they consist of relatively simple mathematics. Finally, ANNs can easily be combined with traditional models through the analysis of residuals. To clarify, traditional models can be used to strip the data of any linear relationships and ANNs can capture the remaining non-linear relationships in the residuals. This combination process is simple as the residuals only need be presented to the ANN for analysis and forecasting. This allows ANNs to improve existing models without redevelopment of the system. The hybrid models in chapter 5 are an example of ANNs as ‘Add-on’ models.

1.1.3.2 Disadvantages of ANNs

The main disadvantage of ANNs is their classification as ‘black box’ techniques. This implies it is difficult to interpret the results and optimised parameters of the fitted model. Thus, for processes where known relationships are modelled, traditional models will be favoured as these relationships can explicitly be modelled by the user. Further, the problem

of over fitting arises with ANNs. Over fitting is the instance where, due to the many internal parameter, the ANN fits the training set of data exactly and reduces the generalisation ability of the model. This disadvantage is discussed in detail further in this study. For large data sets the computing time and requirements are significant for ANNs to learn trends in the data, compared to traditional models. This disadvantage will fade as computing power increases. Finally, there is little study done on the determination of the optimal ANN structure. This study aims to perform investigations into a method that may be used to determine the optimal structure.

1.2 Aim of this Study

There are several reasons why this study is performed – each is described below.

1.2.1 Method of fitting ANNs

Little study has been done on the fitting process of ANNs to time series problems. The aim of this study is to develop and test a methodology of fitting ANNs to time series problems. Further, this methodology aims to determine more accurate parameter estimates for the ANNs than past applications.

1.2.2 Insight into Artificial Neural Networks

For application purposes, it is important to develop an introductory theoretical understanding of Artificial Neural Networks (ANNs). This is presented in this study. Further, past applications of ANNs in the financial sector are investigated. This provides a well-rounded understanding of the current ANN literature and indicates areas in the Actuarial field where ANNs can be applied. Particularly interesting is the forecasting of indices relating to different asset markets and inflation in South Africa.

1.2.3 Study of Inflation and the Money, Bond and Equity Markets

This study attempts to forecast indices relating to the monthly return on the money, bond and equity markets, as well as the monthly inflation rate from 1975 to 2010. These indices have a significant effect on the decision making process of actuaries, as they affect the current and future financial environments. Further, these indices directly affect many areas in which actuaries are involved. These areas include setting and evaluating assumptions underlying financial models, determining investment strategies of firms, forecasting sale numbers and volumes, estimating profits, designing products, estimating lapse and withdrawal rates, performing annual valuations and setting capital requirements.

The forecasting performed in this study uses only past observations of each case, which forms a closed forecasting system. It is important to analyse each case in detail before constructing forecasting models. The results of this analysis are presented in this study.

1.2.4 Forecasts of the Markets and Inflation

The level of accuracy associated with the forecasts, in each application, is analysed to determine whether markets and inflation in South Africa can be forecast using a closed time series forecasting system.

1.2.5 Integrated and Isolated ANNs

Theoretically, there are significant relationships between the different markets, each other and inflation. An aim of this study is to determine the extent and effect of these relationships on forecasting accuracy. It is expected that the

model allowing interaction between the markets and inflation (the integrated ANN) will result in more accurate forecasts than the model that does not allow any interaction (the isolated ANN).

1.2.6 ANNs and Traditional Models

New innovative models need to be tested against traditionally used models in order to determine their performance and effectiveness. The traditional models and ANNs constructed are used for identical forecasting purposes. The forecasting accuracy is compared to determine which technique is most effective at forecasting returns on the markets and inflation in South Africa.

Different forecast periods are considered when comparing traditional models to ANNs. This is done because financial decisions made by actuaries take account of both the current and future financial environments. Thus, to make informed decisions it is necessary to forecast financial variables over several time periods.

1.2.7 Hybrid Models

To implement ANNs in practise it is important to determine their performance when integrated with current models. Hybrid models are constructed and compared to traditional models in order to determine whether ANNs add value to traditional models or not.

1.3 Research Questions and Hypothesis

1.3.1 Research Questions

The questions this study aims to answer are:

1. How to build and optimise ANN structures for modelling financial data.
2. Can monthly inflation or return in the money, bond or equity markets in South Africa be forecast accurately using a pure time series approach?
3. Does the accuracy associated with forecasts decrease as the forecast period expands?
4. How do ANNs compare to traditional models when forecasting time series?
5. Are the inter-market relationships significant when forecasting inflation and/or the money, bond and equity markets over a one and three-month forecast period?
6. Does the combination of ANNs and traditional models (hybrid models) add value to traditional models?

1.3.2 Hypothesis

Monthly inflation and return on the money market can be forecast accurately over both a single and three-month period. The bond market can be forecast but less accurately over the same periods but the equity market cannot be forecast using a time series approach. As the forecast period expands the accuracy of the forecasts will decrease for all applications. ANNs possess a similar forecasting accuracy to that of traditional models. ANNs may outperform traditional models as the forecast period extends. The inter-market relationships are significant for all applications and will aid in the forecasting process. Hybrid models will outperform all the other models considered as they combine the advantages of both traditional models and ANNs.

2 Theory of Artificial Neural Networks

2.1 Aim of Chapter

This chapter presents the theory behind Artificial Neural Networks (ANNs). Further, the types of ANNs are investigated. The structures are explained in detail. Learning algorithms are described and intuitive reasoning is presented.

2.2 Background

The field of Artificial Intelligence consists of many categories, one of which being Artificial Neural Networks (ANNs). Other categories consist of mathematical modelling techniques such as Support Vector Machines, Self-organising maps and Bayesian Statistics to mention a few. This study is concerned with the most widely implemented ANNs. First the Multi-layered Perceptron (MLP) trained through Back Propagation is considered. Next, the Bridged MLP (BMLP) and Fully Connected Cascade (FCC) networks are described.

All ANNs have similar structures consisting of input, hidden and output layers. The neurons in each layer are interconnected with those of the adjacent layers, resulting in parallel processing of information. The different layers are described below:

- Input layer – the network is provided with information from a data set.
- Hidden layers – the network detects and projects the relationships within the data.
- Output layer – the output of the model is produced.

Figure 1 provides an illustration of a MLP with 3 input neurons, 2 hidden neurons and a single output neuron.

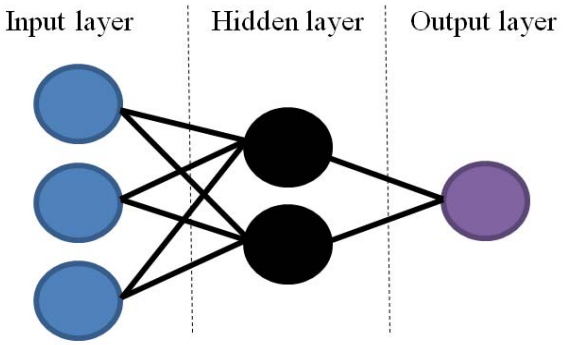

Figure 1: Multi-layered Perceptron (MLP) ANN

There are several unique aspects of ANNs, all of which are covered here. The most important is the ability to 'learn' from experience. This allows the system to be applied in dynamic environments without requiring constant redesign, leading to one of the advantages ANNs possess over traditional statistical methods [7]. Further, real world problems tend to be non-linear in nature [2]. This implies the relationships between variables underlying the data are non-linear. Traditional statistical models assume linear relationships between the underlying variables [2]. Hence, accuracy is lost in some applications when traditional statistical techniques are used. ANNs do not make this assumption and it is possible for them to model these non-linear relationships accurately. Thus, ANNs have increased accuracy above

traditional statistical models in some applications [8]. Finally, ANNs deal well with vaguely defined problems, unfinished data, inaccuracy and uncertainty [9]. These aspects have resulted in ANNs being the most popular soft computing model, but only make up 9% of the models currently being used [9].

ANNs are models that generalise based on past data through the manipulation of weights [1]. They are different from traditional models as they require a small number of underlying assumptions. This makes ANNs far more implementable by other disciplines besides those associated with statistics. Due to ANNs' universal generalisation ability there are a vast range of areas in which they have been applied [2]. These applications are discussed in section 2.9.

2.3 Neural Networks

The origin of Artificial Neural Networks is briefly described in this section. It is followed by an in depth analysis of the Artificial Neuron.

2.3.1 Biological Neural Networks (BNN)

All animals currently on the planet are here because they have adapted, in the best possible way, to their ever-changing environment. As Darwin wrote, the species that is most adaptable to change is the one that is most able to survive. This adaption is able to occur due to a unique and common trait amongst these animals, 'the ability to learn'. Without this ability, it is likely that the earth would be barren, as life would not exist [10].

The ability to learn emanates from the brain of every living creature. Furthermore, the art of learning is performed through a highly specialised network of cells called neurons, which are located in the brain. It is also worth noting the strength and size of these networks determine the extent to which the brain can learn, thus making the human brain far more capable than a "less intelligent" species.

Neurons themselves are the 'building blocks' of the central nervous system. In more practical terms, these neurons can be seen as performing similar tasks to processors in computers. There are, however, differences between the two:

- Processors perform tasks based on simple mathematics while neurons rely on chemical reactions, and,
- Processor speeds are measured in nanoseconds (10^{-9} seconds) while the speed of neurons is measured in milliseconds (10^{-3} seconds).

The neuron is complex in design and contains all the structures of an animal's cell. In order to aid in the understanding of a neuron, an illustration of a single neuron is given in Figure 2. Additional labels have been provided, as the medical terminology is not required to understand the functioning of a neuron.

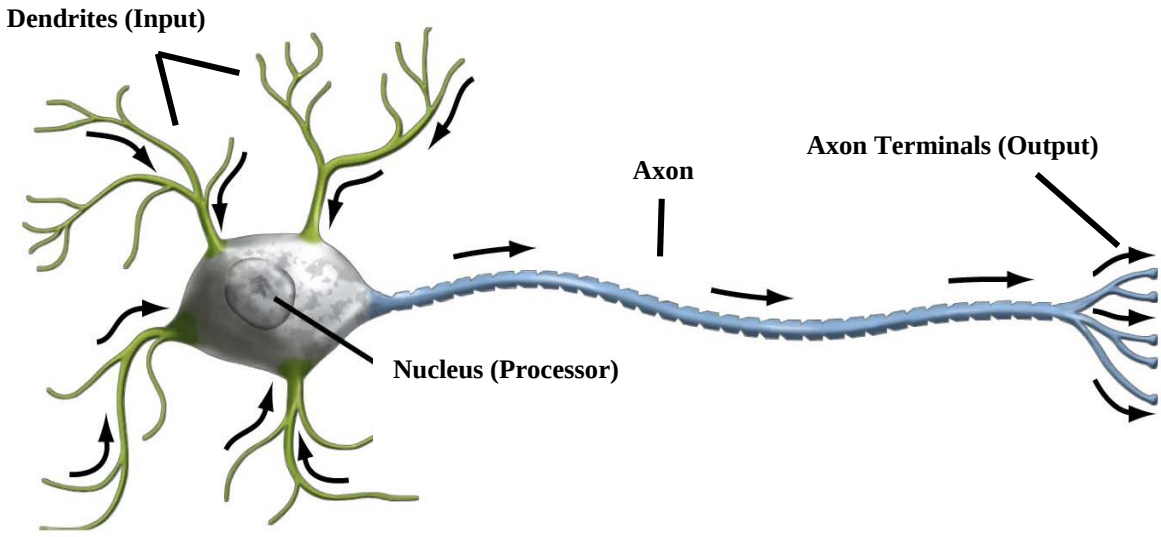

Figure 2: Diagram of a biological neuron

2.3.1.1 Explanation of Biological Neuron

The functioning of the neurons, described below, is aided by Figure 2.

The 'Dendrites' receive signals from other neurons and receptors, and then transmit these signals to the 'Nucleus'. The strength of the signals received by the 'Nucleus' will be communicated to the 'Nucleus' by the synaptic weights accompanying the signals. Synaptic weights are representations of the strength of the signals.

All the signals converge at the 'Nucleus'. The 'Nucleus' emits a single signal that travels along the 'Axon' to the 'Axon Terminals' which are connected to other neurons at 'Synaptic points'. However, if the synaptic weights are low, the neuron will not emit a signal along its 'Axon' and is said to be inactive. The activation of a network or system of neurons will depend on the number of neurons that are active at a certain point in time.

This process is continually repeated throughout the whole brain, when a single task is being performed by an individual. The human brain consists of over 10 billion neurons and over 60 trillion interconnections between these neurons. The magnitude of this network provides the brain with a processing power far beyond that of modern-day computers.

2.3.1.2 Characteristics of Biological Neural Networks

There are several characteristics that have contributed to the brain being such a powerful processing machine, namely its non-linearity as all the neurons are interconnected and are non-linear in nature, hence the inputs received by the brain are not directly proportional to the eventual output. This non-linearity is distributed throughout the brain. Its input-output mapping as the brain can take input from sensory receptors, analyses it and produce output. This allows the brain to learn and develop over time. Its adaptability as it can adapt to different situations and be confronted with unknown scenarios, analyse them and produce relevant output. Its evidential response because every decision made is done with a certain amount of confidence. This is a result of the brain associating an amount of certainty with each

output provided. Its fault tolerance as the failure of a single neuron does not cause a failure of the whole brain. In fact there will be no visible effect to the individual if a few neurons fail. Finally, its large integrated circuit as every neuron is interlinked a massively parallel system is created which results in a large amount of computing power.

2.3.2 Design of an Artificial Neuron

ANNs replicate the rational functioning of the human brain. This replication is done on a small scale. Figure 3 is a diagram of a single artificial neuron which is referred to in the explanation of ANNs.

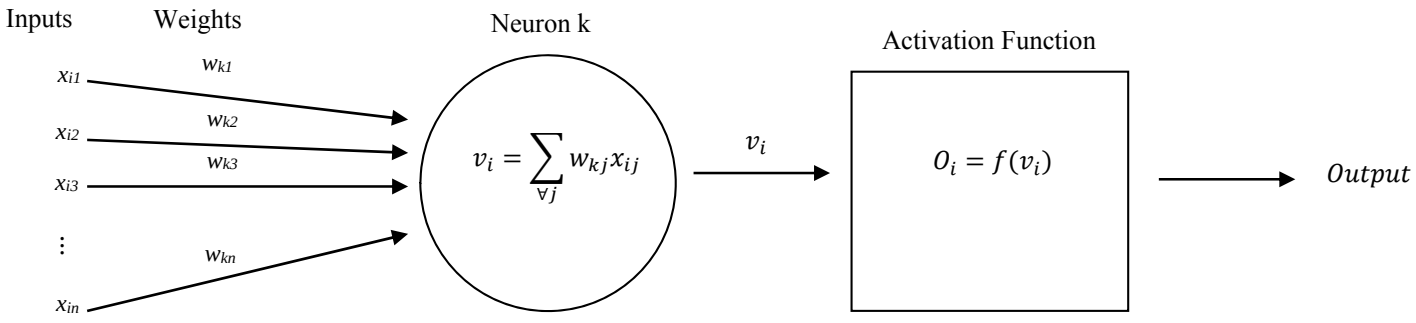

Figure 3: An artificial neuron

Inputs

Inputs allow data into the system. These data are from the original data set or are outputs of preceding neurons. Each input variable (x_{ij}) represents a variable from the data set or a single output from a preceding neuron.

Weights

Every layer is fully connected to both its preceding and succeeding layer. Each connection has a level of strength. This strength is the 'weight' of the connection. Weights are conventionally denoted by (w_{ij}), where '*i*' is the neuron in the preceding layer and '*j*' is the neuron in the current layer. ANNs learn and adapt in dynamic environments through the manipulation of these weights.

Neuron

The energy of a neuron is determined in the neuron through Equation 1. The energy is the sum of all the inputs multiplied by their associated weights, and the bias term. Mathematically the energy is calculated as

$$Energy_j = v_j = b_j + \sum_{i=1}^{n} x_{ij} \times w_{ij}$$

Equation 1: Activation energy in neuron j

where *i* is the input variable reference and b_j is the bias for neuron *j*. Each neuron must be referenced clearly as an ANN may consist of several hundred neurons.

Activation Function

The energy generated in the neuron forms an output through the use of an activation function. Any function can be used as an activation function. Though, it is common for a 'sigmoid' function to be used. The logistic function $\left(f(y) = \frac{1}{1+e^{-\lambda y}} \right)$ or hyperbolic tangent $\left(f(y) = \frac{2}{1+e^{-2\lambda y}} - 1 \right)$ are common choices [11]. The two activation functions mentioned have output restricted to (0:1) and (-1:1), respectively. Scaling of input and output data is required when solving regression problems involving values outside the above bounds.

Scaling

Regression problems commonly have output ranges that exceeds (0;1) or (-1;1). As noted, the activation functions typically used have a range of output that fall in a specified interval. To extend beyond this interval it is necessary to scale the output and occasionally the inputs.

Let $\overline{x} = \langle x_1 \; x_2 \; \ldots \; x_n \; y \rangle$ be an input vector. The (x_i) represent the input information and (y) represents the associated known output. This easily extends to multiple outputs, i.e. (y_j), without the loss of generality. The vector $\bar{x}$ must be scaled before the system can begin analysis. If not scaled properly the activation function becomes saturated and the network becomes erratic. Common scaling methods are discussed below [2].

Along Channel Normalisation

Each input variable (x_i) is scaled independently. Identical scaling is used for the input variable common to all input vectors. For example, (x_1) will be scaled using the same numeric values for every input vector.

Across Channel Normalisation

Each input vector is scaled independently of its predecessor. For example, only one input vector is considered at a time.

Mixed Channel Normalisation

The above two methods are combined.

External Normalisation

All the input vectors are scaled to ensure that they exist in a specific range.

Common Scaling Formulae

Common scaling functions used are as follows: (x_i^*) represents the scaled variable [2].

- $x_i^* = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}$ hence $x_i^* \in [0,1]$
- $x_i^* = \frac{x_i}{x_{\max}}$, so $x_i^* \leq 1$ if $x_{\max} > 0$

- $x_i^* = \frac{(x_i - \bar{x})}{s}$ where $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$ and $s = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$

And [12], [13]

- $x_i^* = b\frac{x_i - x_{\min}}{x_{\max} - x_{\min}} + a$ hence $x_i^* \in [a, b]$
- $x_i^* = \frac{x_i}{\sqrt{\sum_{\forall i} x_i^2}}$

Output

The output represents the result of the system. This consists of either a single value or a vector of values. In the case of regression problems that required scaling of inputs the scaling process must be reversed to obtain a relevant output. This is done before the output is finalised.

Epochs

Epochs or training epochs are the number of iterations the ANN can utilise to learn the trends underlying the data. The maximum number of epochs is specified by the user. If this maximum is too few, the ANN will not learn the trends underlying the data. If this maximum is too many, the ANN will learn random variation in the data (if random variation is present).

2.4 Network Structures

Three main categories of ANN structures exist. Each of which have advantages and disadvantages relative to the others. Different structures outperform others in specific applications, making the optimal choice of structure problem specific. The three categories are described below.

2.4.1 Multilayer Perceptron (MLP) [14]

This is the oldest structure of Artificial Neural Networks. It consists of an input layer, possibly several hidden layers and an output layer. Each layer is interconnected with the preceding and succeeding layer. There are no connections that bypass a layer, i.e. there are no direct connections from the input to the output layer. Further, the number of hidden layers and the neurons within those layers are specified in advance. This kind network is rigid as it cannot adapt its structure automatically. This results in large, unnecessary amounts of computing time. Also, there is no theoretical method of determining the correct number of layers and neurons within those layers. This makes MLP the least powerful of the ANN structures described in this section. However, this structure is the simplest and is most often used in practice. Figure 4 provides a diagram of the network with two hidden layers.

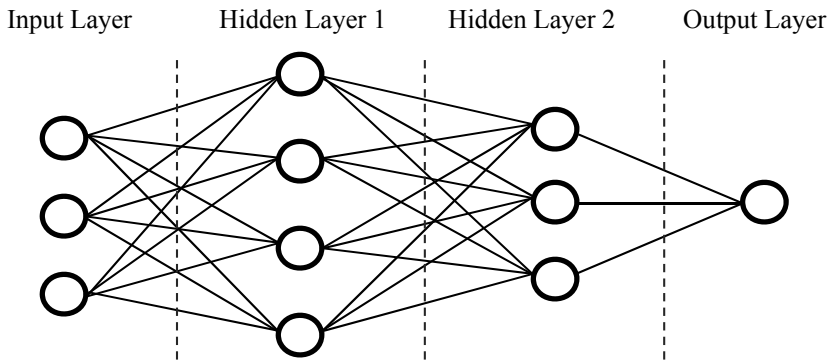

Figure 4: MLP (3-4-3-1)

2.4.2 Bridged Multilayer Perceptron (BMLP) [14]

The BMLP is similar to the MLP. However, this ANN structure allows connections to bypass layers. For example, the input and output layers can be connected directly, bypassing the hidden layers. As in the MLP, the number of hidden layers and neurons within those layers are specified in advance and are not susceptible to change during learning. This results, again, in unnecessary amounts of computing time. Subjectivity enters the design through the choice of hidden layers and neurons as the structure is chosen on a trial-and-error basis. Thus, the optimal structure is often missed. However, this ANN structure requires fewer hidden layers and neurons than the MLP. Thus, computing time is reduced in comparison to the MLP.

Figure 5 illustrates the structure of the network. The dotted lines indicate connections that bypass layers.

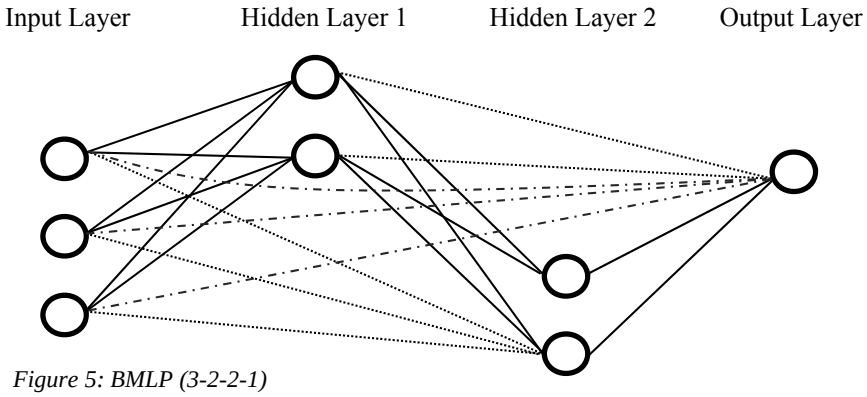

Figure 5: BMLP (3-2-2-1)

2.4.3 Fully Connected Cascade (FCC) [14]

The most powerful and most complex ANN structure is the FCC. Connections are allowed to bypass layers, as in the BMLP. However, the structure of the network is not predetermined. This topology allows the real world problem to construct the network that is most effective. Initially, the network consists of one hidden layer with one hidden neuron. As the system learns the error decreases. If the system achieves a pre-specified error acceptance level, the system stops learning. If this does not happen before the maximum number of epochs is achieved, the system adds another hidden layer with one neuron and the training begins again. This process is performed until the pre-specified error acceptance level is obtained. Thus, the network begins small and grows over time. The final network uses minimal amounts of computing time to train and forecast variables, as every neuron is significant. The simplest network possible is also

obtained, resulting in the greatest generalisation power. However, this system is the most complex to implement. Figure 6 illustrates a possible network structure.

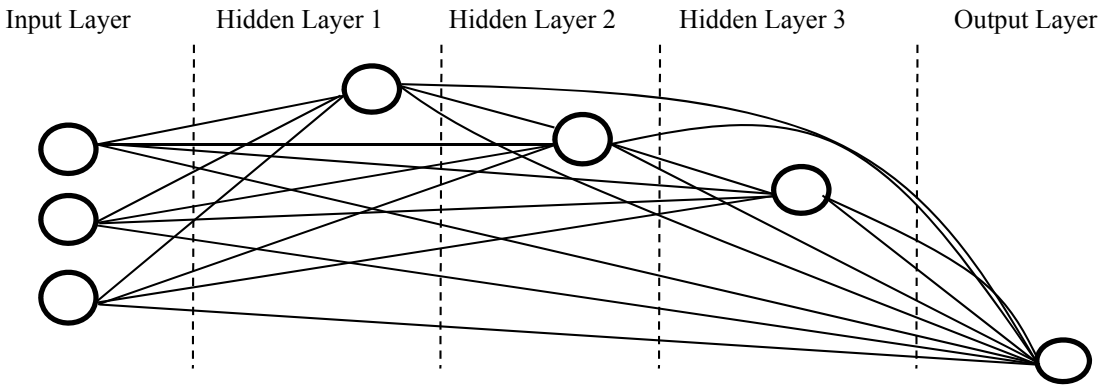

Figure 6: FCC (3-1-1-1-1)

2.5 Types of Artificial Neural Networks

There are two main types of Artificial Neural Networks, namely, Feed Forward and Recurrent ANNs.

2.5.1 Feed Forward [5]

This is the most common type of ANN. It allows a unidirectional flow of information, i.e. from input to output. Information is not allowed to flow backwards through the network, i.e. from output to input. This type of ANN is simpler than the Recurrent ANN. However, it uses more computing power and time than the alternative. Figure 7 depicts the information flow through the connections of a Feed Forward MLP.

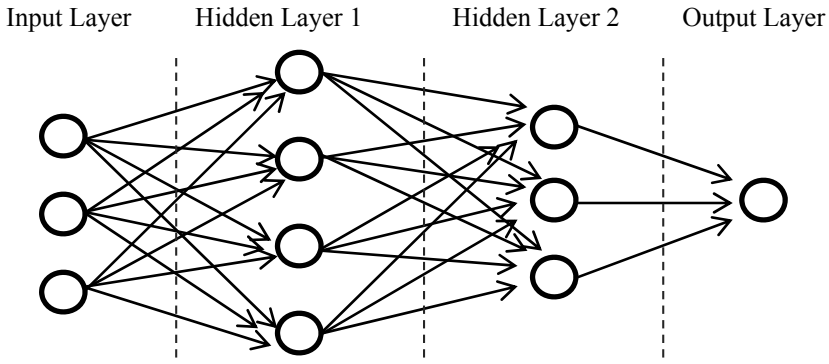

Figure 7: Feed Forward MLP (3-4-3-1)

2.5.2 Recurrent ANNs [5]

Recurrent networks allow bidirectional connections between neurons. For example, the outputs previously determined by the system are connected to the system as inputs. This results in a more complex system than Feed Forward networks. Figure 8 depicts such a network.

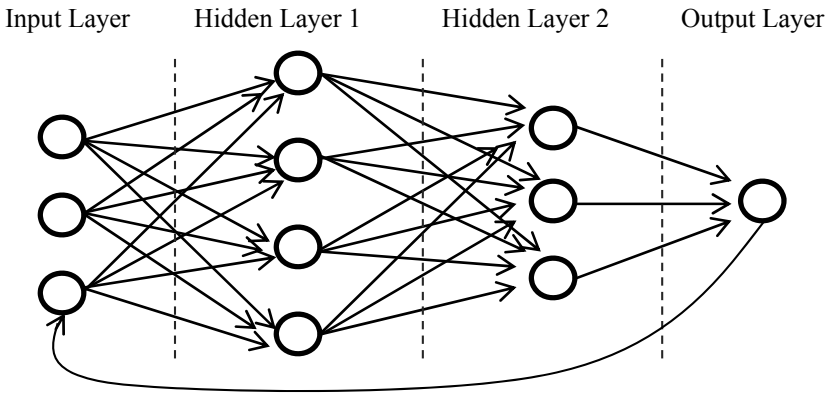

Figure 8: Recurrent ANN (3-4-3-1)

2.6 Learning in ANNs

2.6.1 Biological Learning

It is important to have a basic understanding of how learning is performed on the biological level before it is replicated on a machine level. In this section biological learning is described, followed by the machine representation of different learning mechanisms. Limited knowledge is available to provide an exact explanation as to how people learn. The information provided below is derived from current understanding and may change over time.

Human beings have five ways of learning. They are:

- Error Correction learning,
- Memory learning,
- Hebbian learning,
- Boltzmann learning, and
- Competitive learning.

These are divided into the categories of supervised and unsupervised learning. There are several overlaps between the two categories. This overlap forms a category termed semi-supervised learning. Due to the complexity of each learning category only supervised learning is considered in this study. Further, only detailed investigations of error correction learning are presented. Figure 9 provides an illustration of the learning areas.

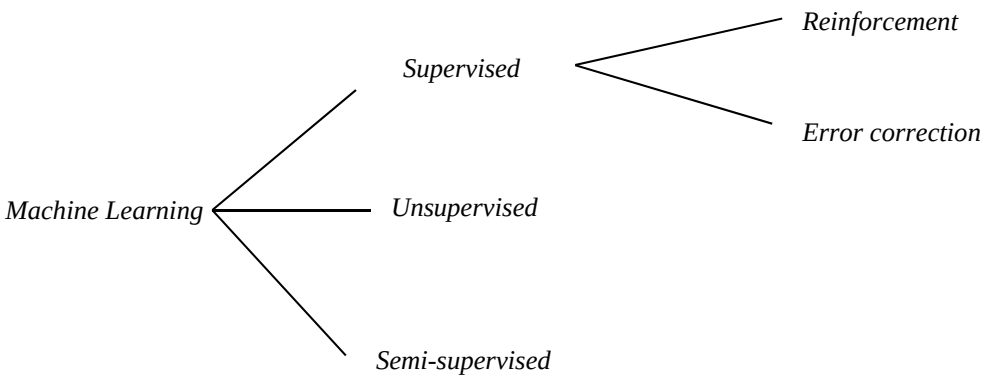

Figure 9: Learning types and categories

Supervised and unsupervised learning are discussed next. This is followed by each supervised learning mechanism both in the biological case and the machine case.

2.6.2 Supervised Learning

Supervised machine learning occurs when training samples, consisting of known inputs and desired outputs, are presented to the system. The training samples, or patterns, are used by the ANN to minimise the error between the forecast and desired outputs. This is done by manipulating each connection's weight.

Supervised learning is used to solve classification and regression problems [15]. This learning can be subdivided into the following [16]:

2.6.2.1 **Reinforcement Learning** – The success of the forecast output is known in each instance, i.e. if the output is identical to the desired result or not. However, the magnitude of the error is not known. Thus, only the input vector is used in weight manipulation. This approach is mainly used to solve classification problems.

2.6.2.2 **Error Correction Learning** – The magnitude of the error is known and is combined with the input vector when manipulating the weights. Error correction learning is used for regression or classification problems.

2.6.3 Unsupervised Learning

Unsupervised machine learning is used when the desired results are unknown. The aim of this learning mechanism is to determine the unconditional distribution of the input data [15]. Further, it is used to identify clusters of the input data [17]. This is important in solving control and optimisation problems [15].

2.6.4 Semi-supervised Learning

Practical data are likely to have partially completed inputs and/or outputs. Semi-supervised learning is a combination of supervised and unsupervised learning. The system learns from complete training examples and extends to the incomplete inputs/outputs. This allows machines to learn in environments less 'crisp' than those that are ideal [1].

In this study, only regression problems are considered. Further, only error correction learning is investigated in detail.

2.7 Supervised Learning Mechanisms in ANNs

A high level description of biological learning is presented in this section. The method of mathematical replication is also provided. Error Correction learning is explained in detail in the next section (section 2.8), but an overview is provided below.

2.7.1 Error Correction Learning

2.7.1.1 Biological Error Correction learning [18]

This is the most widely understood method of learning. It aims to minimise the error an individual can make when coming to a decision based on past experience. The following example will clarify this learning mechanism.

Directly after birth a child is unable to differentiate between a dog and a cat. Over time, the child learns to differentiate between these two animals. Error correction learning results in the ability to recognise these two animals without ever having seen the specific animal before.

This learning process is as follows:

1. The child observes a big dog for the first time and is told that it is a dog. Specific characteristics to what constitutes a dog are assigned in the child's brain. For example, the dog may be furry, panting, large, friendly and be wagging its tail.
2. Later the child observes a small cat but automatically assumes it is a dog because the child has nothing else to classify it as. However, this animal is a cat.
3. The teacher or parents of the child corrects him/her.
4. Now the child has characteristics that are specific to a dog and a cat.
5. The child then observes a small dog. The child has only seen a large dog and a small cat and decides the small dog is a cat (assuming the decision is entirely based on the size of the animal).
6. The teacher or parents corrects the child.
7. The child establishes that dogs can vary in size and assigns a lighter significance weighting to the size factor of the animal when deciding on what type of animal is being observed.
8. This error correction learning continues throughout childhood. The child eventually establishes a large source of past experience. Over time the weights assigned to each factor will be finely tuned and will result in accurate decisions being made.

In conclusion, past experience will be used to adjust the synaptic weights (or weights) so that the error in any decision is minimised.

2.7.1.1.1 Analytical Expression of Biological Error Correction Learning

Assume a binary representation for each input variable and output value.

The following input vector consisting of the animal's characteristics is considered:

$$\bar{x} = \text{<Size, Wet nose, Energetic>}$$

- Size: 0 – small, 1 – large
- Wet nose: 0 – no, 1 – yes
- Energetic: 0 – no, 1 – yes

Consider the two inputs:

- <1,1,1> which is a dog
- <0,0,0> which is a cat

Assume that after learning, the weights associated with the inputs are as follows:

- $W_{size} = 0.1$
- $W_{Wet\ nose} = 1$
- $W_{Energetic} = 2$

And that the decision function is as follows:

$$f(\bar{x}) = \begin{cases} 0 \ \ if\ w_{size} * Size + w_{Wet\ nose} * Wet\ nose + w_{Energetic} * Energetic < 1.5 \\ 1 \ \ if\ w_{size} * Size + w_{Wet\ nose} * Wet\ nose + w_{Energetic} * Energetic \geq 1.5 \end{cases}$$

Equation 2: Dog/cat decision function

Note: '1' is the decision that the animal is a dog and '0' is the decision that the animal is a cat.

The vector which represents a dog will have the value of 3.1 and the vector which represents the cat will have the value of 0. These values are the sum defined in the decision function above (Equation 2). The decision function will conclude correctly that the cat is a cat and the dog is a dog.

The decision function is illustrated in Figure 10.

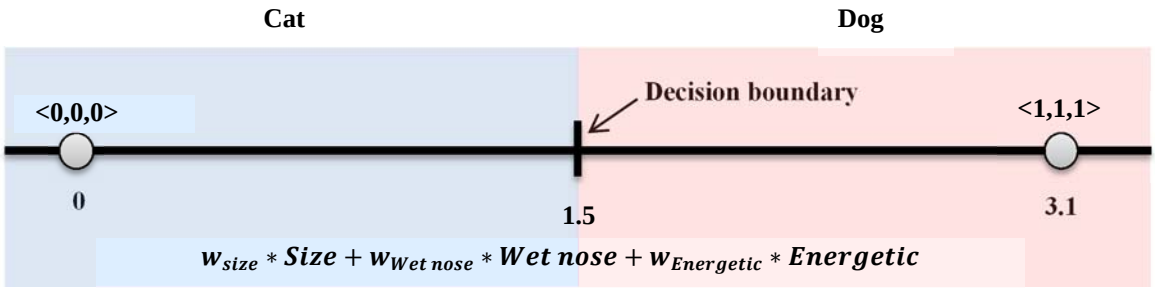

Figure 10: An example of a decision function

Efficient learning results in the optimal decision boundary being obtained. This is the boundary which differentiates between the decision of the animal being a dog or a cat.

2.7.1.2 Introduction to Machine Error Correction Learning

Error Correction learning is the manipulation of weights between neurons in order to minimise the error between the forecast and actual outputs over the training set of data. The training set must be a supervised data set. For additional information on Error Correction learning refer section 2.8.

2.7.2 Memory Learning

2.7.2.1 Introduction to Memory Learning [18]

Memory learning is a process of pattern recognition. For example, it is responsible for the ability to recognise people based on their features. To clarify this consider the following example.

When two people meet, their brains assign 'a name to the face'. Their brains store all the characteristics associated with a specific individual. On meeting later the people's brains receive the same characteristics as in the first encounter and assign the same output, in other words they recognise each other.

The difference between error correction learning and memory learning is the generalisation ability possessed by the former. Error correction learning forms an educated guess based on past experience when forecasting an output. Memory learning determines an identical match to the input from previously experienced samples. Once found the input is assigned the output of the similar vector from past experience. The mechanism of Memory learning is formalised below.

2.7.2.2 The Mechanism of Memory Learning [18]

Memory learning is based on the recognition of patterns and is the easiest to implement but has the poorest generalisation ability. Further, it requires significant computing time for large sets of data.

The learning process is described below.

1. Define the training data set input vectors and the corresponding outputs.

$$\bar{X}_i = \langle x_{i1}\ x_{i2}\ x_{i3}\ \dots x_{in} \rangle \sim x_{ij}\ is\ the\ input\ j\ from\ the\ vector\ i$$

   where x_{ij} is the input *j* from the vector *i*. Further, y_i is the actual output for each corresponding input vector *i*. y_i can be a vector or a scalar. For simplicity it is assumed to be a scalar in this case.

2. The training data is represented as follows (Equation 3), assuming there are *m* different input vectors.

$$\{\bar{X}_i\ y_i\}_{i=1}^{m}$$

Equation 3: Training data set – Memory learning

3. Assume there is a new input vector presented to the system and an output must be assigned. Let the new input vector and the assigned output be represented by $\bar{X}_T$ and y_T where $\bar{X}_T$ is the new input vector and y_T is the expected output for the new input vector.

   To assign an output it is necessary to determine the closest match of $\bar{X}_T$ with the $\bar{X}_i's$. To achieve this the Euclidean distances between the new input vector and all the training input vectors are calculated. The new input vector is assigned the output of the training vector which has the shortest distance from it.

   The distance between the new vector ($\bar{X}_T$) and the training vectors ($\bar{X}_i$) is calculated for each training vector, using the Euclidean distance formula (Equation 4).

$$dist(\bar{X}_i, \bar{X}_T) = \sqrt{\sum_{j=1}^{n} (x_{ij} - x_{Tj})^2}$$

Equation 4: Euclidean distance formula

The minimum distance is determined and the output associated with the nearest training vector is assigned to the new vector.

$$y_T = y_i$$

2.7.2.2.1 Analytical Expression of Memory Learning

To determine the forecast output of a new input vector the mathematical process below is followed.

Let $\overline{X}_T$ represent the new input vector and $\overline{X}_i$ represent previous input vector i.

Let y_T be the forecast output of $\overline{X}_T$.

The vector $\overline{X}_k \in \{\overline{X}_i\}_{i=1}^{n}$ is defined such that:

$$Dist(\overline{X}_T, \overline{X}_k) = \min_{\forall i} \left[Dist(\overline{X}_T, \overline{X}_i) \right]$$

Then $y_T = y_k$

2.7.3 Boltzmann learning

2.7.3.1 The Boltzmann Machine [19, 20]

Boltzmann machine learning constructs a Boltzmann distribution whereby the input vectors presented to it have a high probability of occurring.

> "*Boltzmann distribution is a certain distribution function or probability measure for the distribution of the states of a system"[21]*

The Boltzmann neural network consists of interconnected visible and hidden neurons. The visible neurons receive the input vectors and the hidden layer trains the distribution through manipulating the weights connecting the neurons.

This is a stochastic learning type, which is derived from statistical mechanics and is the most complex mechanism of all five learning mechanisms. Further, Boltzmann learning is still mainly a theoretical learning mechanism.

Boltzmann machine learning is not practical because [21]:

1. The complexity of the system increases exponentially with the expansion of the system. Thus, for any nontrivial problem the system is too complex to use.
2. The time required to run the system until equilibrium increases exponentially as the system becomes more complex.
3. If activation probabilities are intermediate between zero and one, the associated noise in the system causes the weights to random walk.

2.7.3.2 Introduction to the Restricted Boltzmann Machine (RBM) [22]

RBM was developed to deal with the limitations of the Boltzmann machine. The RBM avoids the same limitations by reducing the interconnections between the hidden neurons. Further, complexity is reduced by training the hidden neurons separately.

The RBM and Boltzmann Machine have similar aims. The RBM determines underlying characteristics of the inputs, which can then be used for statistical analysis. To clarify this an example is presented below.

Suppose John enjoys the films Harry Potter and Avatar. The RBM is able to indicate that John enjoys fantasy films. Or, given that John enjoys fantasy films will recommend Harry Potter and/or Avatar. The main difference between the RBM and other learning mechanisms enters when John might not enjoy Harry Potter or Avatar but, in general, enjoys fantasy films. This scenario is known as the 'messy' real world and is captured through the stochastic nature of the network. The RBM makes it possible to be confident (which entails a certain probability) that John enjoys Harry Potter or Avatar given that he enjoys fantasy films and vice versa.

2.7.3.3 Structure of the RBM

The neurons have binary representations in this network. The binary output is used to indicate whether a neuron is active at a certain point in time or not.

- 1 indicates the neuron is 'on' or active, and
- 0 indicates the neuron is 'off' or inactive.

Three types of neurons exist in the network, namely:

- Visible neurons that receive the input vectors,
- Hidden neurons that learn relationships in the data, and
- Bias neurons which adjusts for any bias.

All the visible and hidden neurons are interconnected and have bi-directional connections. Further, every neuron is connected to the bias neurons and each connection has a weight.

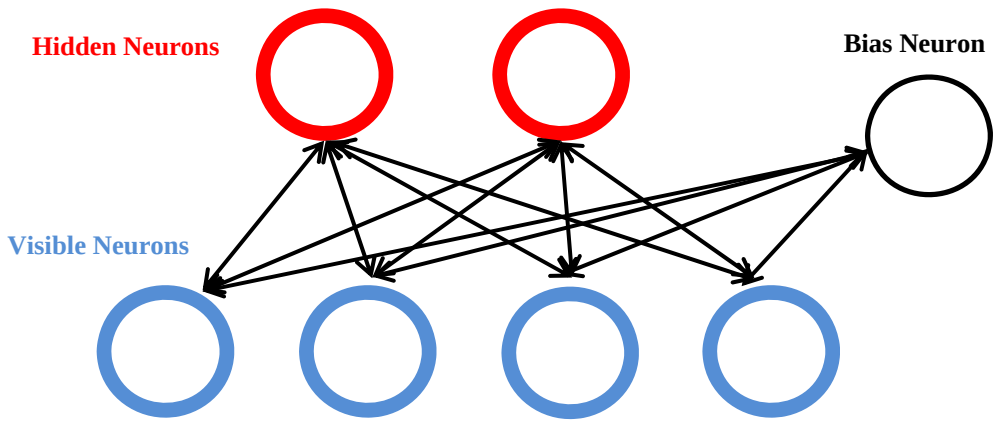

Figure 11: An illustration of the Restricted Boltzmann Machine ANN

2.7.3.4 How the States of the Neurons Change [19, 22]

Consider the weight w_{ij} where i indicates the number of the visible neuron and j represents the number of the hidden neuron. The state of each hidden neuron is updated by computing the 'activation energy'. This energy is defined by:

$$E_j = \sum_{\forall i} w_{ij} x_i$$

Equation 5: Activation energy of hidden neuron j

Where j is the hidden neuron, x_i are the input values (0 or 1) for visible neuron i and w_{ij} is the weight connecting visible neuron i and hidden neuron j. The probability the hidden layer is active is determined through the following process.

Let p_j be the probability the state is 1 at hidden neuron j

Then

$$p_j = \frac{1}{1 + e^{-E_j}}$$

Equation 6: Probability of hidden neuron j being active

A logistic function is chosen in Equation 6 because high activation energies result in a probability close to 1 (activation of the neuron) and close to 0 (neuron is inactive) for low activation energies. Intuitively, the more neurons that are active the greater the likelihood that this neuron is active. Referring to the John's favourite films example (section 2.7.3.2), the more information obtained that suggests he enjoys fantasy films, the more likely the conclusion that he enjoys fantasy films will be made.

The activation function p_j is defined as:

$$p_j = \begin{cases} \text{neuron } j = 1 \text{ with probability } \dfrac{1}{1 + e^{-E_j}} \\ \text{neuron } j = 0 \text{ with probability } 1 - \dfrac{1}{1 + e^{-E_j}} \end{cases}$$

Equation 7: Hidden neurons activation probability for hidden neurons

2.7.3.5 Learning in the Restricted Boltzmann Machine [22]

The learning mechanism in the Restricted Boltzmann Machine replicates the distribution of input vectors by creating a Boltzmann distribution. The similarity between the actual and Boltzmann distribution is measured by the 'Kullback-Leibler divergence'. This measure and dynamics behind it are not discussed in this study.

A supervised data set must be used during the training of the RBM.

2.7.3.5.1 The RBM Learning Mechanism

The learning mechanism present in the RBM is described below.

- Let the input variables in each training vector be denoted by x_i (i=1,2,3,…) and the associated output at the hidden neuron j be y_j.
- Each training sample is considered in turn.

The process:

1. Consider a training vector $\bar{x}$. Let it be replicated by the visible neurons, i.e. visible neuron i will have value x_i assigned to it.
2. The value of the hidden neuron is determined using the logistic activation rule (Equation 7). Let the hidden neuron have the value y_j. The activation energy used is then determined using Equation 5. Initially, the weights (w_{ij}) are randomised.
3. For each connection between two neurons (c_{ij}) Equation 8 is determined which indicates whether both neurons are active or if at least one is inactive.

$$Positive(c_{ij}) \text{ or } P(c_{ij}) = x_i \times y_j$$

Equation 8: Measure of association between visible and hidden neurons

4. The process is reconstructed in reverse order.
5. The value of the visible neurons are determined using the activation probability function (Equation 10). Where the activation energy is determined using Equation 9. This may result in the recalculated vector being different from the original input vector.

$$E_i = \sum_{\forall j} w_{ij} y_j$$

Equation 9: Activation energy for visible neurons

$$p_i = \begin{cases} \text{neuron } i = 1 \text{ with probability } \dfrac{1}{1+e^{-E_i}} \\ \text{neuron } i = 0 \text{ with probability } 1 - \dfrac{1}{1+e^{-E_i}} \end{cases}$$

Equation 10: Activation probability function for visible neurons

6. The hidden neurons are updated and the activation energy E_j (Equation 5) is recalculated. This is done as some of the input values may have changed in the update, resulting in a different activation energy at the hidden neurons.
7. For each connection between two neurons (c_{ij}) Equation 11 is calculated.

$$Negative(c_{ij}) \text{ or } N(c_{ij}) = x_i \times y_j$$

Equation 11: Measure of association between hidden and visible neurons

8. The weights of the connections are updated using Equation 12.

$$w_{ij}^{new} = w_{ij}^{old} + \eta \times \left(P\left(c_{ij}\right) - N\left(c_{ij}\right) \right)$$

Equation 12: Weight update algorithm – RBM

where w_{ij} is the weight between visible neuron j and hidden neuron i and η is the learning rate.

9. This is repeated for all input vectors from the training data set.

The process is continued until the network converges, or the error between the actual and Boltzmann distribution is sufficiently small.

Note:

- $Positive(c_{ij})$ or $P(c_{ij})$ is a measure of the association between the expected and actual output. For example, given the system has the correct inputs does it give the correct output?
- $Negative(c_{ij})$ or $N(c_{ij})$ is the system's estimates given the associated hidden neurons' values. For example, given the system has the hidden neurons' values does it come up with the correct input vector?

2.8 Error Correction Learning

Error Correction learning is performed through the use of the error between forecast and actual outputs. The error measure used is generally the Sum of Squared Error (Equation 13) or Mean Squared Error (Equation 14). The weights of the connections between neurons are manipulated using the error of the system over the training set of data. The learning methodology is described later. The different data sets are described in the next section.

$$SSE = \sum_{t=1}^{n} \varepsilon_t^2$$

Equation 13: Sum of squared errors (SSE)

$$MSE = \sqrt{\frac{1}{n} SSE}$$

Equation 14: Mean squared error (MSE)

where $\varepsilon_t = (\hat{X}_t - X_t)$, $\hat{X}_t$ is the forecast at time *t*, X_t is the actual value at time *t* and *n* is the number of forecasts.

2.8.1 Training, Testing and Validation Data Sets

Artificial Neural Networks (ANNs) are capable of identifying and projecting patterns present in data. However, interest is placed on the generalisation ability of ANNs. If all the patterns present in a noisy data set are learned to an arbitrarily small error the generalisation ability of the ANN becomes poor. This phenomenon is termed *over fitting*. Over fitting occurs when the ANN has learned patterns in a specific data set overly accurately and begins to capture the random variation within the data. This reduces the generalisation ability of the ANN. To decrease the probability of over fitting, the data is divided into several data sets.

- Training set – the training of the network is done on this set. The majority of data is contained in this data set.
- Testing set – the trained network is tested on this set. The testing error determines if the system needs additional or reduced training. This ensures a satisfactory generalisation ability.
- Validation set – the fully trained network is evaluated further on this set and the error obtained is viewed as an independent error of the system.

To avoid over fitting, the error of the system over the training and testing data sets are considered after each epoch. If the error over the training set decreases and the error over the testing set increases there is strong evidence of over fitting. The occurrence of over fitting indicates the training process must halt.

2.8.2 Historic Error Correction Learning

Error correction learning was initially applied to ANNs with no hidden layers. Figure 12 provides an illustration of such an ANN.

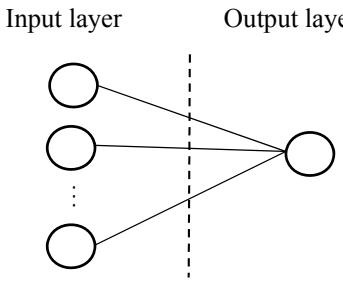

Figure 12: N-1 ANN

This learning mechanism is centred on the principle of gradient descent, which is described in the next section.

2.8.3 Gradient Descent

Gradient descent is an iterative process used to find the minimum of a function [15]. The gradient of the function in combination with several iterative steps is used to find this minimum. The gradient is the first derivative of the error function with regard to the explanatory variable. The principles of gradient descent are explained below.

Initially, a guess is made with respect to the minimum of a function. The gradient of the function at this point is calculated. The guess is altered in the opposite direction to the gradient of the function at this point. The intuitive reasoning for this is that the guess will descend the slope of the function until it achieves the function's global minimum. To clarify this process an example is provided.

2.8.3.1 Gradient Descent – Example

Consider the function in Equation 15:

$$f(x) = x^2$$

where $x \in [-5; 5]$

Equation 15: Example of a function for gradient descent

Gradient descent is used to find the minimum of this function. Let $x_1 = 5$ be the initial estimate of the minimum. The function value at this point is 25. The function (Equation 15) is graphed in Figure 13.

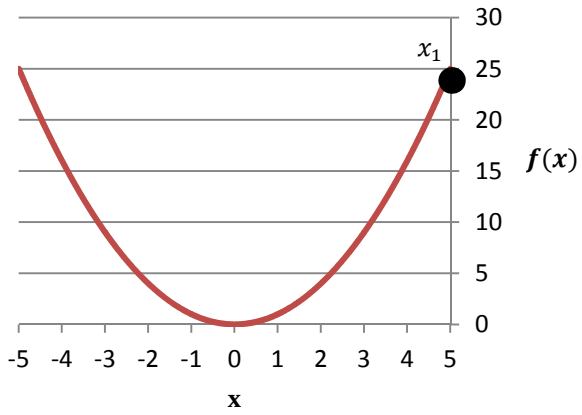

Figure 13: $f(x)$ error surface

According to gradient descent the derivative of Equation 15 at point x_1 is determined.

$$f'(x) = 2x$$

$$f'(x = 5) = 10$$

Equation 16: Derivative of function in Equation 15

The estimate is reduced by a fraction of this derivative (Equation 16). Assume this fraction is 0.2. The adjustment made to point x_1 is as follows.

$$Adjustment = (-1) \times 10 \times 0.2 = -2$$

$$x_1^{new} = x_1 + Adjustment = 5 + (-2) = 3$$

$$f(3) = 9 < 25 = f(5)$$

The function value has decreased and the adjustment has progressed towards the minimum of the function. In Figure 14, x_1^{new} is included in the graph of the original function (Equation 15).

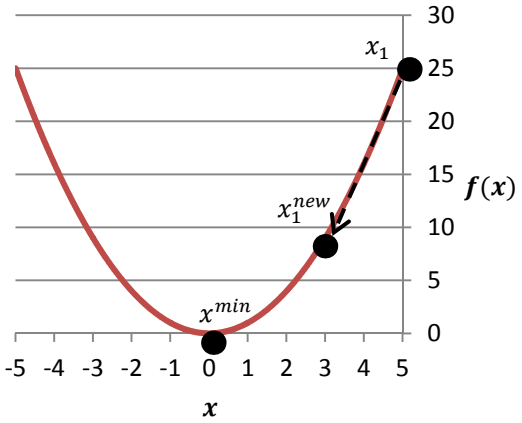

Figure 14: Change in error after a single learning iteration

This process is repeated until the minimum of the function is achieved. This minimum is represented by x^{min} in Figure 14.

2.8.4 Gradient Descent in ANNs with No Hidden Layers

In this section learning through gradient descent is applied to ANNs with no hidden layers. Several simplifying assumptions are made for illustrative purposes. These assumptions are listed below.

1. The weights and bias are initialized randomly between 0 and 1.
2. All activation functions are logistic functions with a parameter of 1.
3. The error measure used is the sum of squared errors.
4. There are n inputs in each input vector.
5. There is a single output neuron.
6. Weights and bias are adjusted after each entry of the data set (input vector) has been processed.

2.8.4.1 Analytical Method

Common notation for this section is provided here. w_{ij} is the weight of the connection between input neuron i and output neuron j. n is the number of inputs. x is the input value. i is the input index. j is the output neuron index. $Desired_i$ is the actual result obtained from the data. k is the number of outputs (one in this case). η is the learning rate (For more information of the learning rate see section 2.8.6.5).

Process:

1. The output of the system is calculated.

$$Output_j = \frac{1}{1 - e^{-\sum_{i=1}^{n} w_{ij} x_i}}$$

Equation 17: Output at neuron j using a logistic activation function

2. The error is calculated.

$$SSE = \frac{1}{2} \sum_{j=1}^{k} (Desired_j - Output_j)^2$$

Equation 18: Sum of squared errors function for ANN with no hidden layers

Note: The $\frac{1}{2}$ in Equation 18 aids with simplification of the weight update term and has no significant effect on the learning in the ANN.

3. The derivative with respect to each weight is calculated.

$$\frac{dSSE}{dw_{ij}} = -Output_j(1 - Output_j)(Desired_j - Output_j)x_i$$

4. Each weight is updated.

$$w_{ij}^{new} = w_{ij}^{old} + \eta \times \frac{dSSE}{dw_{ij}} \times (-1)$$

This process is repeated for every training sample in the data set. Further, it continues for the number of cycles or training epochs specified by the user. This process optimises the system over a number of iterations.

2.8.5 Problems with Historic Error Correction Learning

There are several disadvantages of an ANN structure that does not consist of any hidden layers. The main one lies with the usefulness of the ANN. ANNs with no hidden layers can only classify patterns, in $\mathbb{R}^2$, that are linearly separable. Any non-linear pattern cannot be classified accurately. To demonstrate this, two problems are discussed below. An ANN with no hidden layers is used to identify and replicate the OR logic function (linearly separable) and the Exclusive OR logic function (not linearly separable).

2.8.5.1 OR Logic Operator

For simplicity, it is assumed there are two input variables. The OR logic function assigns 1 as an output if either one or both inputs are 1, and 0 otherwise. The data set is tabulated below (Table 1).

X_1	X_2	Result
1	1	1
1	0	1
0	1	1
0	0	0

Table 1: OR data set

The ANN with one output, one bias and two input neurons used to identify and replicate this function is illustrated in Figure 15.

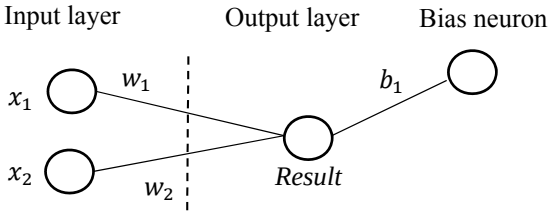

Figure 15: 2-1 ANN

The ANN is trained using the data provided in Table 1. The resulting weights, bias and Mean Squared Error (MSE) is provided in Table 2.

Weight 1 (w_1)	Weight 2 (w_2)	Bias (b_1)	MSE
6.2936	6.2936	-2.9016	0.0012

Table 2: Results of *trained* network – *OR*

The ANN is tested by presenting the same input patterns (Table 1) and recording the assigned outputs. The results are presented in Table 3.

Input 1	Input 2	Output	Actual	Error
0	1	0.96745	1	0.03255
0	0	0.05207	0	0.05207
1	1	0.99994	1	0.00006
1	0	0.96745	1	0.03255

Table 3: Testing of trained network – OR

A goodness of fit parameter of 0.995 out of a possible 1 is achieved by the ANN, which indicates the ANN has successfully identified and replicated the function underlying the data. This demonstrates that ANNs can perform linear classification, in $\mathbb{R}^2$, efficiently using no hidden layers. The *Exclusive OR* problem is considered next.

2.8.5.2 Exclusive OR (XOR)

The XOR logic function is non-linear in $\mathbb{R}^2$. It assigns 1 as an output if and only if one of the two inputs is 1, else 0 is assigned. The data set is tabulated below (Table 4).

X_1	X_2	Result
1	1	0
1	0	1
0	1	1
0	0	0

Table 4: XOR data set

Using and training an ANN with no hidden layers, illustrated in Figure 15, on the XOR data set results in the weights, bias and MSE presented in Table 5.

Weight 1	Weight 2	Bias	MSE
-0.0025	-0.0013	0.0013	0.2503

Table 5: Results of trained network – XOR

The MSE of the system approaches 0.25, but does not decrease below this. An explanation for this emerges on the consideration of each data point independently. These results are provided in Table 6.

Input 1	Input 2	Output	Actual	Error
0	1	0.5	1	0.5
0	0	0.5003	0	0.5003
1	1	0.4994	0	0.4994
1	0	0.4997	1	0.5003

Table 6: Testing trained network – XOR

Each output of the network is approximately 0.5 which results in the error being minimised. This ANN is no more accurate than determining the output by flipping a coin and does not solve the problem efficiently.

To understand the limit of the error function, the linear separability of the data is considered. The data is not linearly separable in $\mathbb{R}^2$, which causes large errors and poor system performance. This ANN structure cannot solve non-linear classification problems.

ANNs with hidden layers are able to deal with non-linear patterns. However, there is an obstacle in using multi-layered networks. The problem involves weight estimation as gradient descent theory can only be applied to networks where errors are present at each neuron. As there are no calculable errors at the hidden neurons it is not possible to update the weights connecting the input and hidden layers. This is a significant problem as most real world problems contain non-linear characteristics. In 1986 a paper was published that solved this problem through the introduction of a learning mechanism termed Back Propagation [23].

2.8.6 Back Propagation Learning (BP)

This learning mechanism also uses the principle of gradient descent. The main factor which differentiates this mechanism from that described previously is that it can be used to train networks with hidden layers. The idea underlying Back Propagation is summarised below.

> *The error present at the output neuron is a combination of the errors at all the hidden neurons of all the hidden layers.*

A detailed description of BP is provided in the following section.

2.8.6.1 Back Propagation Learning Process

Consider a network with '*n*' input neurons, '*m*' hidden neurons, a single hidden layer and '*k*' output neurons. This network is illustrated in Figure 16. For learning to take place it is necessary to determine the gradient of the error function at each hidden and output neuron.

For the purpose of the demonstration several assumptions are made.

1. The Sum of Squared Errors (SSE) function is used as the error measure;
2. The weights are adapted after each entry of the data set has been processed; and
3. All activation functions are logistic functions with a parameter of 1.

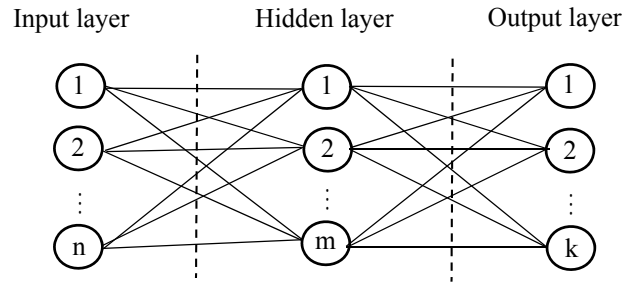

Figure 16: n-m-k ANN

Notation used in the learning process is given below.

Variable	Description
w_{hj}	Weight between hidden neuron *h* and output neuron *j*.
w_{ih}	Weight between input neuron *i* and hidden neuron *h*.
h_h	Value at hidden neuron *h*.
o_j	Value of output neuron *j*.
d_j	Desired result at output neuron *j*.
x_i	Input neuron *i*.
E	SSE of input vector.
η	Learning rate.
n	Number of input neurons.
m	Number of hidden neurons.

Notation 1: Notation used in BP learning explanation

The process of Back Propagation (BP) learning is described below.

1. The values of the hidden neurons are determined using (Equation 19).

$$h_h = \frac{1}{1 + e^{-\sum_{i=1}^{n} w_{ih} x_i}}$$

Equation 19: Value of hidden neuron h

2. The values of the output neurons are determined using (Equation 20).

$$o_j = \frac{1}{1 + e^{-\sum_{h=1}^{m} w_{hj} h_h}}$$

Equation 20: Value of output neuron j

3. The error (SSE) at the output layer is calculated.

$$E = \frac{1}{2} \sum_{j=1}^{k} (d_j - o_j)^2$$

Equation 21: SSE of n-m-k ANN for the BP example

4. The partial derivatives of the error function (Equation 21) with respect to the weights connecting the hidden and output layers are calculated.

$$\frac{dE}{dw_{hj}} = \frac{dE}{do_j} \times \frac{do_j}{dw_{hj}}$$

where $o_j = \frac{1}{1 + e^{-\sum_{h=1}^{m} w_{hj} h_h}}$

The partial derivative simplifies to:

$$\frac{dE}{dw_{hj}} = o_j(1 - o_j)(d_j - o_j)h_h = \delta_j h_h$$

Equation 22: Simplified partial derivative of SSE with regard to each weight (hidden to output layer)

5. The weights of the connections between the hidden and output layers are updated. This weight update is derived from the principle of gradient descent, which uses the partial derivatives of the error measure with regard to the weights connecting the hidden and output layers (Equation 22).

$$w_{hj}^{new} = w_{hj}^{old} + \eta \times \frac{dE}{dw_{hj}}$$

6. The weights connecting the input and hidden layers are updated. This requires the partial derivative of the error measure with respect to each weight connecting the input and hidden layer (Equation 23). The partial derivatives of all output neurons are considered because the weights connecting the input and hidden layers affect all the output neurons and thus the overall error of the system [24].

$$\begin{aligned}
\frac{dE}{dw_{ih}} &= \sum_{j=1}^{k} \frac{dE}{do_j} \times \frac{do_j}{dh_h} \times \frac{dh_h}{dw_{ih}} \\
&= \sum_{j=1}^{k} o_j(1 - o_j)(d_j - o_j) w_{hj} h_h (1 - h_h) x_i \\
&= h_h(1 - h_h) x_i \left(\sum_{j=1}^{k} o_j(1 - o_j)(d_j - o_j) \right) \\
&= h_h(1 - h_h) x_i \sum_{j=1}^{k} \delta_j w_{hj}
\end{aligned}$$

Equation 23: Simplified partial derivative of SSE with regard to each weight (input to hidden layer)

7. The weights connecting the input and hidden neurons are updated. This uses the weight update determined in Equation 23.

$$w_{ih}^{new} = w_{ih}^{old} + (-1) \times \eta \times \frac{dE}{dw_{ih}}$$

8. This process is performed for all the input vectors in the data.

These steps are repeated until the maximum number of training epochs is achieved or the error of the system is sufficiently small. This update allows networks with multiple layers to learn. Learning in Feed Forward Multi-layered Perceptron Artificial Neural Networks (FFMLPANNs) is described in the following section.

2.8.6.2 Learning in the Multi-layered Perceptron

The learning process for networks consisting of several layers is similar to that of a single layer, described in section 2.8.6.1. An explanation of learning in the Feed Forward Multi-layered Perception ANN is presented below. It assumes the weights are updated after each data entry has been processed. The notation used is similar to that in the previous section (Notation 1) with two additions presented in Notation 2.

Variable	Description
b_j	Bias neuron for output neuron j.
b_h	Bias neuron for hidden neuron h.

Notation 2: Additional notation for bias neurons

Learning process:

1. The output at each hidden neuron is calculated.

$$h_h = \frac{1}{1+e^{-\left(b_h+\sum_{i=1}^{n} w_{ih}x_i\right)}}$$

Equation 24: Value of hidden neuron h, including a bias term

2. The output at each output neuron is calculated.

$$o_j = \frac{1}{1+e^{-\left(b_j+\sum_{h=1}^{m} w_{hj}h_h\right)}}$$

Equation 25: Value of output neuron j, including a bias term

3. The Sum of Squared Errors (SSE) of the ANN is calculated using Equation 21. If the SSE is below the threshold level specified by the user the system stops learning as the weights are optimised. If the SSE is above the acceptable error level, step 4 is performed.

4. The partial derivatives of the SSE with respect to the weights connecting the hidden and output neurons are determined using Equation 22.

5. The partial derivatives of the error measure with respect to the weights connecting the input and hidden layers are calculated using Equation 23.

6. The weights connecting the input and hidden layers are updated using the value of the partial derivatives determined in step 5.

$$w_{ih}^{new} = w_{ih}^{old} + \eta \times \frac{dE}{dw_{ih}} \times (-1)$$

$$= w_{ih}^{old} + (-1) \times \eta \times h_h(1-h_h)x_i \sum_{j=1}^{k} \delta_j w_{hj}$$

7. The bias at each hidden neuron is updated.

$$b_{ih}^{new} = b_{ih}^{old} + (-1) \times \eta \times \frac{dE}{dw_{ih}}$$

$$= b_{ih}^{old} + (-1) \times \eta \times h_h(1-h_h) \sum_{j=1}^{k} \delta_j w_{hj}$$

In this case $x_i = 1$ because it is assumed that the value of the bias neuron is always 1. In this study the connection between the bias neuron and the neuron associated with it is referred to as the bias.

8. The weights connecting the hidden and output layers are updated using the partial derivatives determined in step 4.

$$w_{hj}^{new} = w_{hj}^{old} + (-1) \times \eta \times \frac{dE}{dw_{hj}}$$

$$= w_{hj}^{old} + (-1) \times \eta \times o_j(1-o_j)(d_j - o_j)h_h$$

9. The bias at each output neuron is updated.

$$b_{hj}^{new} = b_{hj}^{old} + (-1) \times \eta \times \frac{dE}{dw_{hj}}$$

$$= b_{hj}^{old} + (-1) \times \eta \times o_j(1-o_j)(d_j - o_j)$$

This process is repeated for all the entries of the data set. Further, it is repeated for a specified number of training epochs or until an acceptable overall error of the system is achieved.

2.8.6.3 Exclusive OR (XOR) – Revisited

The Exclusive OR (XOR) problem described earlier (section 2.8.5.2) is re-discussed here in light of the Back Propagation (BP) learning mechanism. To reiterate, the XOR logic function assigns a 1 if and only if one of the two inputs is 1. Table 7 provides the input values of each variable with the assigned output.

X_1	X_2	Result
1	1	0
1	0	1
0	1	1
0	0	0

Table 7: XOR data set

This problem is solved by considering an ANN with two input neurons, one hidden layer, two hidden neurons and one output neuron (2-2-1). This network is depicted in Figure 17.

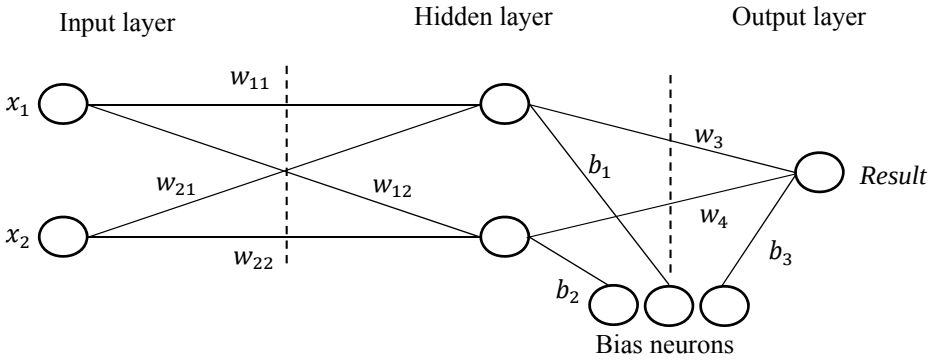

Figure 17: 2-2-1 ANN

The ANN is trained on the data set presented in Table 7. The resulting weights, bias and MSE after training are presented in Table 8.

Weight 1-1 (w_{11})	Weight 1-2 (w_{12})	Weight 2-1 (w_{21})	Weight 2-2 (w_{22})	Weight 3 (w_3)	Weight 4 (w_4)
-7.4181	-4.8643	-8.1629	-4.8447	-8.6117	8.0824

Bias 1 (b_1)	Bias 2 (b_2)	Bias 3 (b_3)	MSE
2.5277	7.0968	-3.7611	0.001

Table 8: Results of trained network – XOR revisited

The ANN is tested by presenting the same patterns as those on which it is trained and noting the error. The results are provided in Table 9.

Input 1	Input 2	Output	Actual	Error	R-squared
0	1	0.038842	0	0.038842	0.996
0	0	0.969942	1	0.03006	
1	1	0.971293	1	0.02871	
1	0	0.025079	0	0.025079	

Table 9: Testing trained network – XOR revisited

A coefficient of determination (R-squared statistic) of 0.996 indicates that ANNs can perform non-linear classifications efficiently using a single hidden layer. Thus, Back Propagation learning enables ANNs to deal with non-linear patterns. This is a crucial aspect of ANNs as they are otherwise not efficient in non-linear real world problems.

2.8.6.4 Online and Batch Learning

The Back Propagation learning mechanism described in the previous section adjusts the weights of the connections after a single entry from the data has been processed. This results in a significant number of updates being determined and applied within a single cycle through the training data. There are two disadvantages of this method [25].

- Over or under fitting may occur as it is difficult to determine the optimal number of epochs.
- For large sets of data the computing time required for convergence and minimisation of the system's error may be great.

In light of these disadvantages, 'Batch' learning was developed. This learning uses the same techniques as the learning described earlier (online learning) but has several differences. Batch learning updates the weights of the connections after a batch, several data points, has been processed. Online learning adjusts the weights after a single data point has been processed. Further, batch learning uses the average error over the batch of data to update the weights of the connections. This reduces the required computing time and improves the generalisation ability of the system. The size of the batch of data can vary from several entries to the entire data set. There is no theoretical method of determining the optimal size of each batch. For batch learning, the Back Propagation (BP) learning mechanism of a FF MLP ANN with a single hidden layer is presented below.

Back Propagation Learning in Batch Form

The notation differs slight from that used to demonstrate online learning (section 2.8.6.2). The notation used in this section is given in Notation 3. Further, the Mean Squared Error (MSE) measure is used to measure the error of the system.

Variable	Description
w_{ih}	Weight connecting input neuron *i* and hidden neuron *h*.
w_{hj}	Weight connecting hidden neuron *j* and output neuron *j*.
x_{il}	Input neuron *i* for input vector *l*.
h_{lh}	Value of hidden neuron *h* for input vector *l*.
o_{lj}	Value of output neuron *j* for input vector *l*.
d_{lj}	Desired value of output neuron *j* for input vector *l*.
b_h	Bias neuron for hidden neuron *h*.
b_j	Bias neuron for output neuron *j*.
η	Learning rate.
n	Number of input neurons.
m	Number of hidden neurons.
l	Data entry (input vector) index.
k	Number of output neurons.
j	Output neuron index.
N	Number of data entries (input vectors) in batch.

Notation 3: Notation used in demonstration of BP learning in FFMLPANN

Learning process:

1. The value of each hidden neuron is calculated using Equation 26.

$$h_{lh} = \frac{1}{1 + e^{-\left(b_h + \sum_{i=1}^{n} w_{ih} x_{il}\right)}}$$

Equation 26: Value at hidden neuron h for data entry l

2. The value of each output neuron is calculated using Equation 27.

$$o_{lj} = \frac{1}{1+e^{-\left(b_j + \sum_{h=1}^{m} w_{hj} h_{lh}\right)}}$$

Equation 27: Value at output neuron j for data entry l

3. The MSE of the system is calculated using Equation 28.

$$MSE = \frac{1}{N}\sum_{l=1}^{N} E_l = \frac{1}{2N}\sum_{l=1}^{N}\sum_{j=1}^{k}(d_{lj} - o_{lj})^2$$

Equation 28: MSE error over a batch of data

If the MSE is below the threshold level specified by the user the system stops learning and the weights will be viewed as optimised. If it is above the acceptable error level step 4 is performed.

4. The partial derivatives of the error function with respect to each weight connecting the hidden and output layers are calculated using Equation 29.

$$\frac{dE_l}{dw_{hj}} = o_{lj}(1-o_{lj})(d_{lj} - o_{lj})h_{lh} = \delta_{lj} h_{lh}$$

Equation 29: Partial derivative of MSE with respect to weights connecting the hidden and output layers

The partial derivative is calculated for every entry in the batch. Once all the derivatives have been determined the average is calculated (Equation 30).

$$\frac{d\overline{E}}{dw_{hj}} = \frac{1}{N}\sum_{l=1}^{N}\frac{dE_l}{dw_{hj}} = \frac{1}{N}\left(\sum_{l=1}^{N}\delta_{lj} h_{lh}\right)$$

Equation 30: Average partial derivative of MSE with respect to weights connecting the hidden and output layers

5. The partial derivatives of the error function with respect to each weight connecting the input and hidden layers are calculated using Equation 31.

$$\begin{aligned}\frac{dE_l}{dw_{ih}} &= h_{lh}(1-h_{lh})x_{il}\left(\sum_{j=1}^{k} o_{lj}(1-o_{lj})(d_{lj}-o_{lj})w_{hj}\right)\\ &= h_{lh}(1-h_{lh})x_{il}\sum_{j=1}^{k}\delta_{lj} w_{hj}\end{aligned}$$

Equation 31: Partial derivative of MSE with respect to weights connecting the input and hidden layers

The partial derivative is calculated for every entry in the batch. Once all the derivatives have been determined the average is calculated (Equation 32).

$$\frac{d\overline{E}}{dw_{ih}} = \frac{1}{N}\sum_{l=1}^{N}\frac{dE_l}{dw_{ih}} = \frac{1}{N}\left(\sum_{l=1}^{N}\left(h_{lh}(1-h_{lh})x_{il}\sum_{j=1}^{k}\delta_{lj} w_{hj}\right)\right)$$

Equation 32: Average partial derivative of MSE with respect to weights connecting the input and hidden layers

6. The weights connecting the input and hidden layers are updated using Equation 32.

$$w_{ih}^{new} = w_{ih}^{old} + (-1) \times \eta \times \frac{d\overline{E}}{dw_{ih}}$$
$$= w_{ih}^{old} + (-1) \times \eta \times \frac{1}{N}\left(\sum_{l=1}^{N}\left(h_{lh}(1-h_{lh})x_{il}\sum_{j=1}^{k}\delta_{lj}w_{hj}\right)\right)$$

7. The bias at each hidden neuron is updated.

$$b_{ih}^{new} = b_{ih}^{old} + (-1) \times \eta \times \frac{d\overline{E}}{dw_{ih}}$$
$$= b_{ih}^{old} + (-1) \times \eta \times \frac{1}{N}\left(\sum_{l=1}^{N}\left(h_{lh}(1-h_{lh})\sum_{j=1}^{k}\delta_{lj}w_{hj}\right)\right)$$

   In this case $x_{il} = 1$ because it is assumed that the value of the bias neuron is always 1.

8. The weights connecting the hidden and output layers are updated using Equation 30.

$$w_{hj}^{new} = w_{hj}^{old} + (-1) \times \eta \times \frac{d\overline{E}}{dw_{hj}}$$
$$= w_{hj}^{old} + (-1) \times \eta \times \frac{1}{N}\left(\sum_{l=1}^{N}\delta_{lj}h_{lh}\right)$$

9. The bias at each output neuron is updated.

$$b_{hj}^{new} = b_{hj}^{old} + (-1) \times \eta \times \frac{d\overline{E}}{dw_{hj}}$$
$$= b_{hj}^{old} + (-1) \times \eta \times \frac{1}{N}\left(\sum_{l=1}^{N}\delta_{lj}\right)$$

This process is repeated for all the entries of the data set. Further, it is repeated for a specified number of training epochs or until an acceptable overall error of the system is achieved.

2.8.6.5 Shortcomings of Back Propagation learning

Back Propagation (BP) learning assumes the error surface is simple in nature. In real world problems the error surface is likely to have level plains, deep local minima and varying ranges. This leads to several shortcomings associated with BP learning. The differences between the theoretical and real world problem error surfaces are illustrated in Figure 18. Each major shortcoming of Back Propagation learning is described.

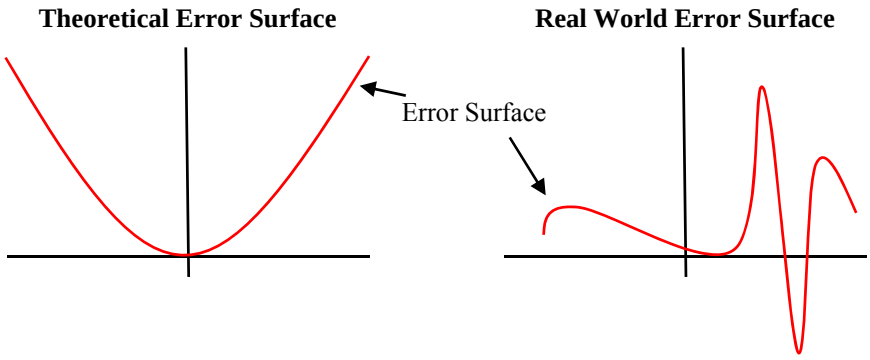

Figure 18: Error surfaces

Learning Rate

The learning rate indicates the proportion of the system's error that is used to update the weights after each epoch. This parameter is specified by the user and is unique for each problem. There are no theoretical methods of determining the optimal learning rate in practical cases. Specifying excessively large or small learning rates present problems.

Large Learning Rates

If the learning rate chosen is far greater than the optimal rate the learning process begins to oscillate. This occurs as it is not possible for the system to reach a minimum of the error surface because the weight updates are too large. This oscillation around a minimum is depicted in Figure 19, which shows the error oscillating between points 'A' and 'B' as it attempts to reach point 'C' (the minimum).

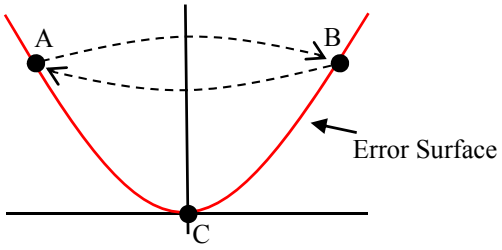

Figure 19: Oscillations due to large learning rate

Small Learning Rates

If the learning rate is far smaller than the optimal rate, the computing time required for the training process increases significantly. Small learning rates indicate the weight updates are small and may have little or no effect on the overall error. Figure 20 depicts the change in the error after several thousand training epochs have been performed with an excessively small learning rate. This system will require a significant amount of training time before the minimum (point C) is reached. For large data sets the required computing power will be substantial.

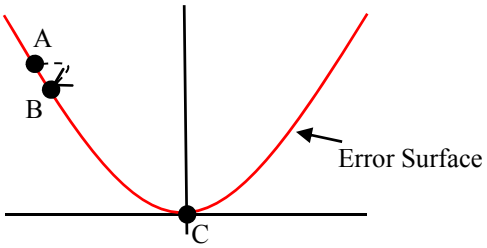

Figure 20: Small change in error due to small learning rate

Local Minimums

The error surfaces in real world problems are not simple and are likely to contain several local minimums. Learning through BP can result in the error getting 'stuck' in these local minimums, causing the global minimum to be missed. This results in an inefficient system. Figure 21 illustrates a scenario where the error starts at point 'A' and gets trapped in a local minimum at point 'B' during the learning process. In this case the system has missed the global minimum (point C) and is not optimal.

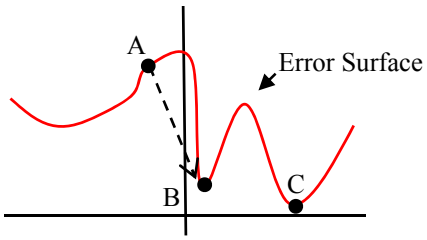

Figure 21: Learning getting trapped in local minima

Weight Update Speed

The weight of each connection is generally in the range (-1; 1). Weights larger than this generally lead to the saturation of the network. The error of the system is used by the learning algorithm to update the weights of the connections between the final hidden and output layers. This error is then propagated backwards through the hidden layers to the input layer, adjusting the weights as it progresses backwards through the layers. As the error is propagated backwards it is reduced in absolute size, because it is multiplied by weights in the range (-1:1). The result is that the update size and speed decrease as the error is propagated backwards. This leads to a greater learning speed between the final hidden and output layers than between the input and first hidden layers. For large networks with many hidden layers this causes slow convergence of the system and significant computational requirements.

2.8.6.6 Improvements to Back Propagation Learning

There have been several improvements to the BP learning method in order to address its shortcomings. The most efficient are described below.

Stopping Criterion [26]

The system is programmed to stop learning when the error of the system is smaller than an acceptable value (which indicates sufficient training), or the error is larger than a specified value (which indicates network saturation). This reduces the probability of excessive learning and the computational requirements associated with it.

Momentum [5]

Momentum is a proportion of the weight update from the previous training epoch and is used as an additional term added onto the weight update. The aim of this is to increase the learning rate and to avoid local minima during training. It is described below.

$$\text{Momentum} = \alpha \Delta w(t-1)$$

Momentum is incorporated into the weight update as follows:

$$\begin{aligned}\Delta w(t) &= -\eta \frac{dE(t)}{dw(t)} + \text{Momentum} \\ &= -\eta \frac{dE(t)}{dw(t)} + \alpha \Delta w(t-1)\end{aligned}$$

where $E(t)$ is the error of the system at time t, $\Delta w(t)$ is the weight update at time t, η is the learning rate and α is the momentum coefficient ($\alpha \in (0,1)$).

Intuitive Reasoning

The idea behind Momentum is explained through the use of a landscape (which represents the error surface of the system) and a boulder (which represents the current error of the system) analogy. When the system is initialised, the error is random. This represents the boulder being placed randomly on the landscape. The aim of this exercise is to 'roll' this boulder so that it reaches the lowest point in the landscape. This 'rolling' is done by the learning algorithm. There is a risk of two things happening as we do not know the exact structure of the landscape.

Firstly, if the landscape is shallow the boulder will take a significant amount of time to reach the minimum point, which represents an increase in training time for the ANN. Momentum allows the increase in velocity expected in the boulder, to be represented in the learning algorithm though an increase in weight update size. On shallow error surfaces, Momentum increases the learning speed.

Secondly, there may be many local minimums in the error surface. Momentum allows the boulder to 'roll out' of these local minimums. This increases the likelihood that the boulder will come to rest at the global minimum of the error surface/landscape.

Shortcomings of Momentum

1. Momentum requires the estimation of the Momentum Coefficient parameter (α) for which there are no theoretical optimisation methods.
2. The introduction of Momentum into the training algorithm requires an additional parameter to be estimated (learning rate and Momentum Coefficient) leading to increased complexity.
3. The Momentum Coefficient is unique for each practical application. The estimation of the parameters is done through trial and error which increases the time required to tailor the ANN to a specific task.
4. In several practical cases, Momentum did not decrease the time required for the system to converge but only increased the difficulty of optimisation [27].

2.8.7 Resilient Propagation (RPROP)

RPROP is an improvement on the BP learning algorithm and addresses several shortcomings of BP. RPROP outperforms most first order training algorithms in both training time and accuracy; these algorithms include Back Propagation, Quick Propagation and Super SAB [28].

RPROP differs from BP in several areas.

- Only the direction of the gradient of the error measure (sign of the partial derivative) is used to manipulate the weights. RPROP does not use the magnitude of the error to update the weights, as in BP. Thus, all areas of the system are trained evenly and more efficiently.
- Each connection is assigned a unique learning rate. This localises learning and allows weights to be updated in accordance with their contribution to the system error.
- The vagaries caused by a single generalised learning rate are removed.
- The learning rate is directly and solely responsible for the size of the weight update.

2.8.7.1 Functioning of RPROP [26]

Each weight is assigned a random learning rate when the system is initialised. These rates increase or decrease exponentially in order to minimise the error of the system. Whether the rates increase or decrease is determined by the gradient of the error function at the previous and current updates. The gradient of the error function is determined by its partial derivative with respect to a specific weight.

If there is a change in sign of the partial derivative of the error measure (gradient direction of the error surface) with respect to the weight being considered, the optimal minimum has been missed. This results in a decrease of the learning rate. If the sign of the partial derivative remains the same, the learning rate is increased. This technique requires that there be bounds on the learning rates. Traditionally, these limits have been 0.000001 and 50.

RPROP requires two parameters to be estimated, an exponential learning rate increase and decrease. These are consider in sections 2.8.7.2 and 2.8.7.3. The mathematical interpretation of RPROP is presented in section 2.8.7.4.

2.8.7.2 Learning Rate Increase

An increase in the learning rate reduces the time until convergence of the ANN. This is achieved by increasing the size of the learning step if the sign of the gradient remains constant over two consecutive weight updates. Two consecutive identical signs indicate the system is approaching a minimum error. By increasing the step size the minimum error will be approached with an increased speed. This is illustrated in Figure 22.

The optimal learning rate increase parameter is difficult to determine as there is little theoretical study available on it. There is a danger that the learning rate increase is significantly larger or smaller than the optimal parameter. An excessively small learning rate increase requires a large number of training epochs for the ANN to converge. A large learning rate increase causes the system to ‘overshoot’ the minimum error, resulting in a significant number of oscillations and increased training time.

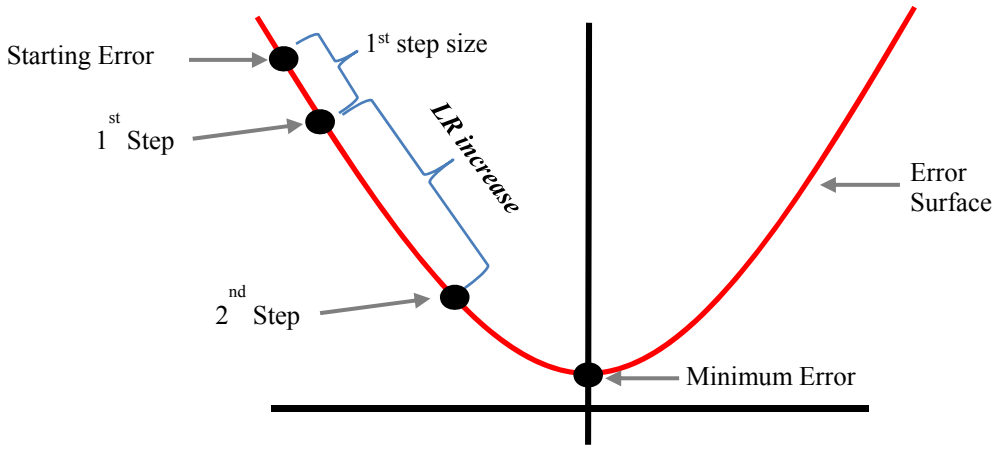

Figure 22: Learning rate increase

2.8.7.3 Learning Rate Decrease

The decrease in learning rate is responsible for the reduction in the learning step size if the system 'overshoots' the minimum error as shown in Figure 23. If the learning rate decrease is not present, the system will begin to oscillate and the error will become trapped at a certain point (as depicted by the dotted blue line in Figure 23).

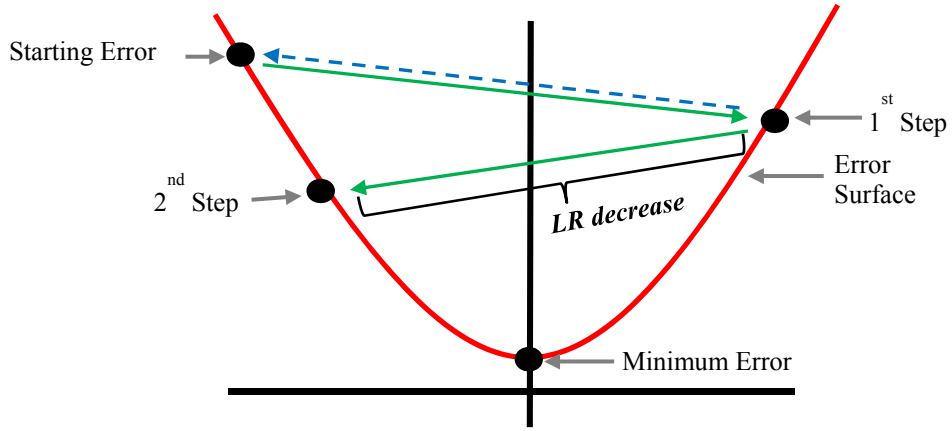

Figure 23: Learning rate decrease

The efficient learning rate decrease parameter is difficult to determine as there is little theoretical study available on it. There is a danger that the learning rate decrease is significantly larger or smaller than the optimal parameter. An excessively small learning rate decrease results in a significant number of oscillations before the minimum is obtained. A large learning rate decrease requires a large amount of training epochs and computing power to achieve convergence of the system.

2.8.7.4 RPROP (Online Learning)

The notation used in this explanation is similar to Notation 1 but several additional variables are introduced which are presented in Notation 4.

Variable	Description
$\eta_{pq}(t)$	Learning rate at epoch t for connection between neuron p and q.
$\eta_{\min}$	Minimum learning rate.
$\eta_{\max}$	Maximum learning rate.

Notation 4: Additional notation for RPROP online learning

The RPROP learning process is presented below.

1. Specify the learning rates. Initially, these values are randomized using a statistical distribution with reasonable parameters. For example, a normal distribution with mean 0 and standard deviation 0.001 can be used because the weight updates are expected to be small in magnitude.

$$\text{Let } \eta_{pq}(t) \in (\eta_{\min} : \eta_{\max}),\ a \in (1 : \infty),\ b \in (0 : 1)$$

2. The learning rate is updated for epoch t using Equation 33.

$$\eta_{pq}(t) = \begin{cases} \min(a \times \eta_{pq}(t-1), \eta_{\max}) & \text{if} \quad \frac{dE}{dw_{pq}}(t) \times \frac{dE}{dw_{pq}}(t-1) > 0 \\ \max(\eta_{\min}, b \times \eta_{pq}(t-1)) & \text{if} \quad \frac{dE}{dw_{pq}}(t) \times \frac{dE}{dw_{pq}}(t-1) < 0 \\ \eta_{ij}(t-1) & \text{elsewhere} \end{cases}$$

Equation 33: Learning rate update rule for RPROP online learning

3. Determine the weight update.

$$\Delta w_{pq}(t) = \begin{cases} -\eta_{pq}(t) & \text{if} \quad \frac{dE}{dw_{pq}}(t) > 0 \\ \eta_{pq}(t) & \text{if} \quad \frac{dE}{dw_{pq}}(t) < 0 \\ 0 & \text{elsewhere} \end{cases}$$

Equation 34: Weight update rule for RPROP online learning

Note: This section presents the RPROP learning algorithm as an online version. The steps presented here are used to determine superior weight updates to BP. Steps 1 – 3 are not an isolated process. They must be incorporated into the learning process presented in section 2.8.6.2.

2.8.7.5 Batch Learning

This section presents the batch learning RPROP process. The notation used in this section is that from section 2.8.6.4.1 (Notation 3) with the additions made in section 2.8.7.4 (Notation 4).

Note: *Steps 2 – 6 are identical to the learning process presented in section 2.8.6.4.1 and are presented here for completeness.*

1. Specify the learning rates. Initially, these values are randomized using a statistical distribution with the reasonable parameters. For example, a normal distribution with mean 0 and standard deviation 0.001 can be used because the weight updates are expected to be small in magnitude.

$$\text{Let } \eta_{pq}(t) \in (\eta_{\min} : \eta_{\max}),\ a \in (1 : \infty),\ b \in (0 : 1)$$

2. The value of each hidden neuron is calculated using Equation 26.
3. The value of each output neuron is calculated using Equation 27.
4. The MSE of the system is calculated using Equation 28. If the MSE is below the threshold level specified by the user the system stops learning and the weights will be viewed as optimised. If it is above the acceptable error level step 5 is performed.
5. The partial derivatives of the error function with respect to each weight connecting the hidden and output layers are calculated using Equation 29. The partial derivative is calculated for every entry in the batch. Once all the derivatives have been determined the average is calculated (Equation 30).
6. The partial derivatives of the error function with respect to each weight connecting the input and hidden layers are calculated using Equation 31. The partial derivative is calculated for every entry in the batch. Once all the derivatives have been determined the average is calculated (Equation 32).

Note: Steps 7 – 10 are performed for every weight in the ANN.

7. The partial derivatives, calculated above, are evaluated in the current training epoch (time *t*). These are compared to the partial derivatives from the previous training epoch (time *t-1*) through Equation 35. For the first epoch the partial derivatives are assumed to be 0.

$$\frac{d\overline{E}}{dw_{pq}}(t) \times \frac{d\overline{E}}{dw_{pq}}(t-1)$$

Equation 35: Evaluation of successive partial derivatives for RPROP

8. The learning rates are updated using Equation 36.

$$\eta_{pq}(t) = \begin{cases} \min(a \times \eta_{pq}(t-1), \eta_{\max}) & \text{if } \frac{d\overline{E}}{dw_{pq}}(t) \times \frac{d\overline{E}}{dw_{pq}}(t-1) > 0 \\ \max(b \times \eta_{pq}(t-1), \eta_{\min}) & \text{if } \frac{d\overline{E}}{dw_{pq}}(t) \times \frac{d\overline{E}}{dw_{pq}}(t-1) < 0 \\ \eta_{pq}(t-1) & \text{if } \frac{d\overline{E}}{dw_{pq}}(t) \times \frac{d\overline{E}}{dw_{pq}}(t-1) = 0 \end{cases}$$

Equation 36: Learning rate update rule for RPROP batch learning

9. The weight update is calculated using Equation 37.

$$\Delta w_{pq}(t) = \begin{cases} -\eta_{pq}(t) & \text{if } \frac{d\overline{E}}{dw_{pq}}(t) > 0 \\ +\eta_{pq}(t) & \text{if } \frac{d\overline{E}}{dw_{pq}}(t) < 0 \\ 0 & \text{if } \frac{d\overline{E}}{dw_{pq}}(t) = 0 \end{cases}$$

Equation 37: Weight update rule for RPROP batch learning

10. The weights are updated.

$$w_{pq}(t) = w_{pq}(t-1) + \Delta w_{pq}(t)$$

This process is repeated for all the entries of the data set. Further, it is repeated for a specified number of training epochs or until an acceptable overall error of the system is achieved.

Note: There is no explicit mention of the bias in the above process. In this case it has been included as a weight. Further, the input and hidden neurons have not been considered separately when implementing the RPROP learning algorithm, as done in the Back Propagation explanation.

2.9 Application of ANNs

ANNs have been applied to several financial problems in the past. This section aims to describe several applications and discuss their shortfalls. The applications considered are listed below.

1. Forecasting forward interest rates.
2. Estimating general insurance reserves.
3. Credit risk evaluation.
4. Detecting credit card fraud.
5. Forecasting stock exchange movements.
6. Forecasting commodity prices.
7. Forecasting foreign exchange rates.
8. Predicting distress in credit unions and bankruptcy.

2.9.1 Forecasting Forward Interest Rates [29]

The aim of the study was to compare the performance of Recurrent Artificial Neural Networks (RANNs – section 2.5.2) to a recursive method, which had provided good forecasts in the past. This comparison was performed using forward interest rates. The RANN structure allowed the forecast rates to be used in future forecasts, which permitted the past forecast values to have an influence on future forecasts.

To generate a set of forward rates at time t, the price, the coupon payment of a specific bond at time t, and the forward rate at time t-1 were used. These were combined through Monte Carlo simulation to achieve the required approach.

For the ANN, the Quick Propagation learning algorithm was implemented. Several different structures were attempted, with different combinations of momentum term and learning rate. The maximum number of epochs used

was 10 000. The hidden and output layers used sigmoid and linear activation functions, respectively. The data set consisted of 2 000 observations, which were divided into a training (1333 observations) and a testing (667 observations) set of data.

The parameters used are provided in Table 10.

Parameters	Values
Input Neurons	3
Hidden Neurons	3
Momentum Term	0.9
Learning Rate	0.5

Table 10: Parameters used in study – forward rates

As the Hetroskedasticity within the data increased, the RANN outperformed the recursive method. This is explained by the ANN's ability to deal with random variation within the data set and the increased non-linearity in the data.

Shortcomings and Possible Improvements

1 Previous forecasts are used to generate additional forecasts. This leads to a compounding of errors associated with the forecasts, as the forecast period extends. This can be avoided by not allowing past forecasts to influence future forecasts. Alternatively, an additional error term that represents the error of past forecasts can be incorporated into the model.
2 The structures and parameters are not extensively examined and are likely inefficient. When constructing an ANN it is important to determine an efficient structure, as a poor structure will lead to poor forecasts.

2.9.2 Estimating General Insurance Reserves [13]

Establishing reserves are essential in any insurance industry. The accuracy of these reserves results in the failure or success of an insurer. In this study ANNs were used to estimate these reserves through a run-off triangle approach. Initially, the ANNs separately estimated the expected claims by development year and year of origin. These networks were then combined through a linking process, leading to an increase in forecasting accuracy. It was concluded that ANNs estimate general insurance reserves accurately and a linked ANN outperforms isolated ANNs.

The structure of the Feed-Forward Multi-Layered Perceptron ANN had five input, ten hidden and a single output neuron.

Shortcomings and Possible Improvements

1 The ANN structure is not investigated, which implies an inefficient structure is likely used. Further investigation to determine the efficient ANN structure must be performed.
2 The ANNs are only compared to the Chain Ladder Method. Addition run-off triangle calculation methods must be considered.

2.9.3 Credit Risk Evaluation [30]

The aim of this experiment was to determine if ANNs are capable of classifying high and low risk housing mortgage loans. The data used consisted of 240 data points obtained from a commercial bank. The application was a simplification of the real world scenario.

The variables considered in this model are presented in Table 11.

Related to Enterprise	*Related to Individual*
Earnings per share	Yearly income
Net assets per share	Education received
Net asset earnings ratio	Monthly payment
Earnings per share deductions	
Current ratio	
Liability ratio	
Gross profit	
Net profit to sales ratio	
Inventory turnover	
Fixed-assets turnover	
Assets turnover	
Net asset ratio	
Fixed asset ratio	

Table 11: Explanatory variables used in models - CRE

Through statistical analysis it was determined that 16 of the variables were significant. This lead to a structure consisting of 16 input neurons. The number of hidden neurons was determined by using a rule of thumb presented in Equation 38.

$$HN = \sqrt{IN + ON} + a$$

Equation 38: Rule of thumb for determining number of hidden neurons

Where:

$HN \rightarrow$ Hidden Neurons

$IN \rightarrow$ Input Neurons

$ON \rightarrow$ Output Neurons

$a \rightarrow$ constant $\in$ 1,2,...,10

Experiments were performed using different values of *a* in Equation 38. A structure using two hidden layers with seven hidden neurons in each proved efficient. There were two output neurons in the structure. One indicated default and the other non-default. This lead to a structure consisting of:

- 2 hidden layers,
- 7 hidden neurons in each hidden layer,
- 16 input neurons, and
- 2 output neurons.

To train the network, the Newton algorithm was implemented which made use of a Hessian matrix. This matrix was difficult to calculate and thus an estimate had to be made.

The ANN performed well and classified the relative data points correctly. However, this application was a simplification of the real world problem.

Shortcomings and Possible Improvements

1 A rule of thumb is used to determine the number of hidden neurons. Limited consideration beyond this rule is given. Additional investigations to determine the efficient number of hidden layers and hidden neurons in each layer must be done.
2 It is assumed that each loan had the same exposure and term of repayment. The actual group considered is not necessarily homogenous which may skew results.
3 More study must be performed into the choice of variables used to explain credit risk.
4 Estimation of the Hessian matrix in the learning algorithm increased the error associated with the training process.

2.9.4 Detecting Credit Card Fraud [31]

This study compared the effectiveness of Artificial Neural Networks at detecting online fraud to logistic regression. The ANN structure used was the Feed Forward Multi-layered Perceptron. Further, simple Back Propagation was used as the learning algorithm with logistic activation functions.

The ratio of fraudulent to non-fraudulent transactions in the data was approximately 1:22 500. This made it difficult to identify any trends associated with credit card fraud, because of the dilution caused by non-fraudulent transactions. To solve this problem stratified sampling was implemented. This divided the data into three sub-data sets. The first set had a ratio of 1:1, fraudulent to non-fraudulent transactions. The next had a ratio of 1:4 and the final set had a ratio of 1:9.

For each model a binary output was assigned. The output of 1 flagged the transaction as fraudulent and the output of 0 indicated a non-fraudulent transaction. Two measures of error were considered in the models. The first was the Mean Squared Error (MSE) between the actual and output result. This determined the accuracy of the system. The second measure was the number of fraudulent transactions the system identified correctly, which determined the percentage of fraudulent transactions captured.

In total 13 ANNs were built; all of which outperformed the logistic regression models over the testing set of data.

Shortcomings and Possible Improvements

1 The models built use traditional Back Propagation. Their performance will improve through the use of more complex and efficient learning algorithms.
2 The data sets considered are not a true representation of reality as the percentage of fraudulent to non-fraudulent transactions is manipulated. Training the models on the manipulated data sets and testing them on a data set that contains the actual ratio of fraudulent to non-fraudulent transactions will lead to more practical results.
3 More complex structures must be considered. These will improve the performance of the models but will lead to increased model risk.
4 The models implemented are of the classification type. Introducing regression models is important as the possible size of fraudulent transactions must be estimated.

2.9.5 Stock Exchange Movements [32, 33]

Stock market forecasts are important in the financial sector. There have been several attempts to forecast the movements of the stock market. Two of them are described below.

The ANNs constructed in these studies consist of Feed Forward Multi-layered Perceptrons trained using Back Propagation. One of the networks incorporates the Leven-Marquardt learning algorithm which increases the speed of learning. Both the networks have a fixed learning rate and momentum term. The studies indicate that the movement of stock prices (increase/decrease) can be forecast to some degree through the use of ANNs.

Shortcomings and Possible Improvements

1 The models mentioned above are of a classification type. They determine if the stock price is expected to rise or fall. The usefulness of the models will be improved through the introduction of a regression element that estimates the size of the forecast movement.
2 Variable momentum and learning rates will allow the system to deal with the problem of local minima more efficiently.
3 Additional investigations on the choice of parameters must be done to ensure efficient parameter estimates are used.

2.9.6 Forecasting Commodity Prices [34]

A comparison of ANNs and ARIMA models was performed in respect to forecasting commodity prices. The ANNs outperformed the ARIMA models. However, the combination of the ANN and ARIMA model into a hybrid model outperformed both ANNs and ARIMA models.

Shortcomings and Possible Improvements

1 The concept of a hybrid model is promising. A future application of allowing the ARIMA model to deal with all the linear relationships in the data should be attempted. All the non-linear relationships in the residuals can then be modelled using ANNs.
2 More advanced traditional models, such as GARCH models, must be implemented and compared to ANNs.

2.9.7 Forecasting Foreign Exchange Rates [7]

A Feed Forward Multi-layered Perceptron ANN was constructed and used to make trading decisions regarding foreign exchange rates. The model provided a signal whether to 'buy', 'sell' or 'do nothing'. The output neuron's value was continuous in nature. This output value was presented to a simple filter which represented the risk profile of the investor. A constant learning rate and momentum term was used. The model resulted in appropriate trading decisions.

Shortcomings and Possible Improvements

1 The introduction of a variable learning rate and momentum term will improve effectiveness.
2 The structure of the network should be made more dynamic in order to improve generalisation ability.

2.9.8 Predicting Distress in Credit Unions and Bankruptcy

ANNs were created to classify credit unions or firms into low or high default risk categories. They took several financial ratios as input and had a binary output. These models proved to be accurate in predicting financial distress and bankruptcy on the validation and testing data sets. For more information on these applications see [7, 9, 11, 35, 36].

Possible improvements

1 It will be helpful if the models could provide a probability of ruin. This is the probability that the credit union or firm will become bankrupt over a future period. To achieve this, a regression element will need to be introduced.
2 A method must be developed to determine borderline credit unions or firms so action can be taken before a transition from low to high risk occurs.

2.10 Structures Investigated

This study focuses on the regression ability of Feed Forward Multi-layered Perceptron Artificial Neural Networks (FFMLPANN). Associated with this, a supervised learning approach is taken and the Resilient Propagation (RPROP) learning algorithm used. Further, RPROP is used in a batch form with the entire training set forming a single batch. This type of ANN is applied to four different data sets which demonstrate the various abilities of the ANNs.

2.10.1 Foreseen Problems in Application of ANNs

ANN Structure

There are no theoretical methods of determining the optimal number of input or hidden neurons. Further, the optimal number of hidden layers is difficult to determine. ANNs can have infinitely many structures which leads to the difficulty of determining the optimal structure. This difficulty is exacerbated by the knowledge that each ANN structure will be unique for each application.

Learning in ANNs

RPROP requires the specification of a learning rate increase and decrease parameter. The optimal parameters are difficult to determine as there is little literature available with regard to their optimisation. Further, the number of training epochs required for efficient learning is difficult to determine as it is unique for each application. Also, the randomisation of the initial structure will add increased difficulty in determining this parameter. Finally, the division

of the training and testing sets is subjective as there are best practise principles but no theoretical way of determining the optimal split.

Over Fitting in ANNs

An ANN has the ability to precisely replicate the data set presented to it. Over fitting is a term associated with ANNs when the system no longer captures the underlying trends but captures the random variation in the data. This problem is addressed by dividing the data into two independent sets. A training data set which consists of the majority of data and a testing data set which consists of the remaining data points. By considering the error of the system during training on both data sets it is possible to determine when the system begins to over fit. This point is indicated by an increase in the error over the testing data while the error over the training set continues to decrease.

3 Traditional Time Series Forecasting Models

The aim of this chapter is to construct traditional time series forecasting models for the monthly inflation rate, as well as the monthly return on the money, bond and equity markets in South Africa. The models constructed forecast the required index over a one and three-month forecast period. These models are compared to ANNs in chapter 6.

3.1 Time Series Forecasting

Time Series Forecasting is a mathematical modelling technique which uses past observations of a specific variable to forecast values of that variable at specific times in the future. It assumes a significant amount of the variation of a variable can be explained by its observed values. In most cases this is untrue, however, generally there is correlation between the past and future values of a variable. These correlations may be weak or strong, depending on the specific variable under consideration. Incorporating time series forecasting with an allowance for influential, external factors generally leads to better explanation of a variable at future points in time, than performing either process in isolation.

To reiterate, time series models are centred on the assumption that a significant amount of the variation associated with a variable can be explained by its past observations. Further, the relationships between the current and past observations are assumed to remain constant over time. The construction of the traditional models in this chapter is achieved through the use of the statistics program SAS.

The forecast periods of each market and inflation are one and three months. These periods are considered as comparing the models on an increased time horizon allows for better and more general conclusions as to the performance of the models. Further, forecasting some of the markets a single month ahead may not be particularly useful for actuaries. When considering the money market, which is stable in the short term, the forecast period is extended to twelve months. Forecasts of the money market a year from now is of more use to actuaries than one or three-month forecasts.

The definition of a three-month forecast period is as follows. The three-month forecast of inflation or return is the monthly inflation rate or return over the third month, two months from the moment the forecast is generated. It is the forecast of inflation or return from month $t+2$ to $t+3$, where t is the month in which the forecast is generated.

3.2 Data Analysis

The data used to fit the time series models was collected from the capital markets of South Africa and relates to the money, bond and equity markets, as well as inflation. The data is obtained from two capital market history studies [37, 38], which have identical inflation, money market and bond market returns but differing average equity returns that date back to 1925. The difference in the average equity returns is due to differing focus of the two studies. Study 1 [37] focuses on returns driven by commercial and industrial equities. Study 2 [38] focuses on the inclusion of mining equities. From 1960, the returns on equity are common. The data available on the money market, bond market and inflation dates back to 1945, 1929 and 1939, respectively. All returns recorded are inclusive of re-invested dividends or interest. The data used are average monthly readings of each market and inflation from 1975 to 2010. This is done to maintain consistency across all the markets considered.

3.2.1 Inflation Data [37]

The Consumer Price Index (CPI) of South Africa is used to measure inflation. Readings are produced by the Central Statistical Service. CPI is widely used as a measure of inflation since it relates to the change in price over time of a basket of consumer goods. This basket is constructed using a sample of 30 000 household buying patterns and

represents the average basket of goods bought by a household. The data set is constructed using CPI readings on a monthly basis over a range of 36 years (or 432 months) from 1975 to 2010.

3.2.1.1 Analysis of Inflation Data

Table 12 provides summary statistics on the inflation data set considered.

Statistics	
Minimum	-1.149 %
Maximum	4.315 %
Mean	0.802 %
Standard Deviation	0.603 %
Range	5.463 %

Table 12: Inflation data characteristics

The line chart in Figure 24 illustrates inflation from 1975 to 2010 on a monthly basis. Further, the mean, maximum, minimum and one standard deviation from the mean are superimposed on the chart.

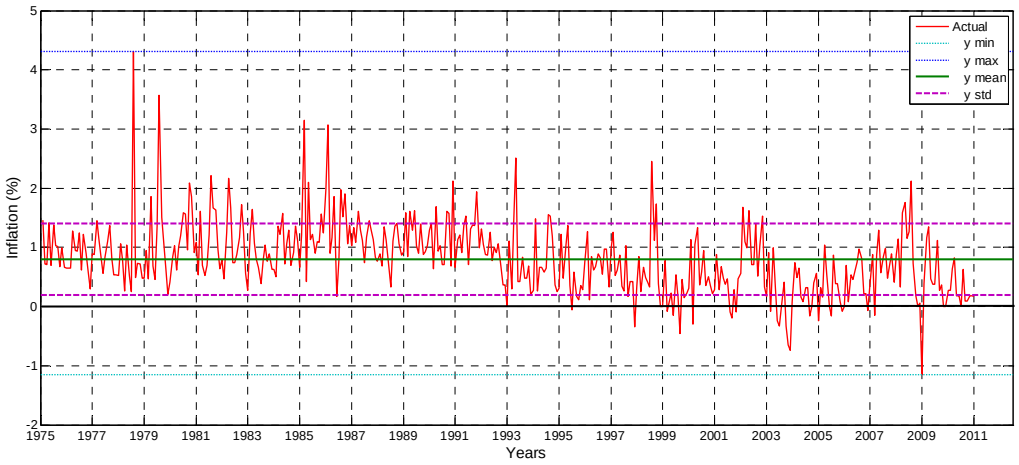

Figure 24: Actual Inflation from 1975 – 2010, including mean, max, min, and one standard deviation from the mean

The column chart in Figure 25 plots the number of observations in each inflation category against its corresponding inflation category. This provides a graphical estimation of the density function associated with inflation. The bell curve , superimposed on the chart, graphically represents the estimated density function.

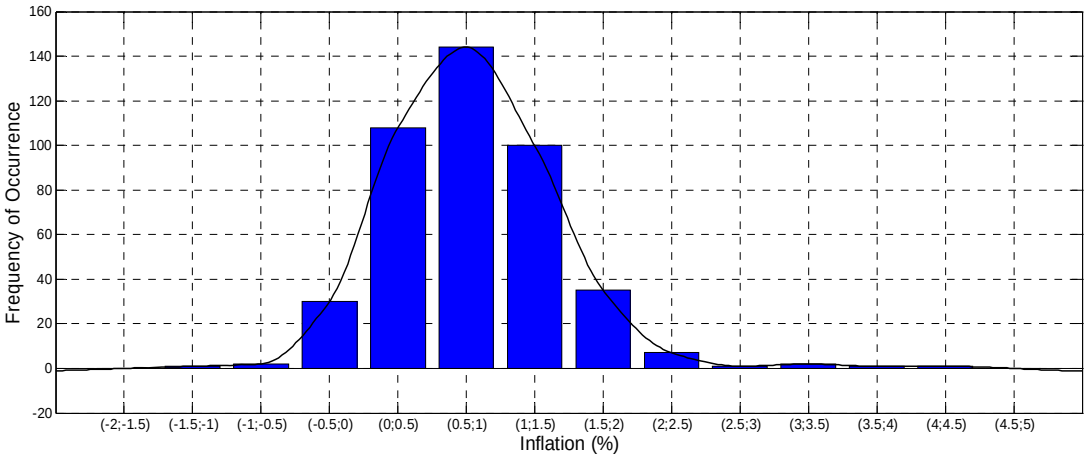

Figure 25: Distribution of inflation with a graphically estimated density function

3.2.1.2 Discussion of Inflation Data

The average monthly inflation is approximately 0.8% (or 10% per year) over the period considered. There are instances of negative monthly inflation in South Africa, the most recent being in 2009. No visible patterns are detected in the monthly inflation rate, as can be seen from Figure 24. Random variation is present in the data set, as indicated by the infrequent spikes and volatile rates in the data. This random variation is further supported by the large standard deviation (0.603%) compared to the mean (0.802%). Figure 25 indicates inflation was bound within -1.5% and 4.5% per month. The majority of the inflation readings are situated between -0.5% and 2.5%. Seasonality is detected in the data set, with a period of 12 months.

3.2.2 Money Market Data [37]

There were and still are no official indices tracking the overall performance of the money market. In this case, an index has to be chosen which represents the average return from the money market. There are several possible instruments that can be used, ranging from Bankers' Acceptances and Treasury Bills to Negotiable Certificates of Deposit (NCD). The instrument chosen is NCDs since their rates were not distorted by the various investment requirements implemented in different industries from 1960 to 2010. Further, during the 1990s, 40% to 50% of the money market's capitalisation consisted of NCDs. Finally, NCDs were issued for varying durations, up to one year, providing greater information on the yield curve of the money market. Other instruments were only issued for periods of three months. The observations used in this investigation range over 36 years (432 months) from 1975 to 2010.

3.2.2.1 Analysis of Return on the Money Market

Table 13 provides summary statistics on the money market data set considered.

Statistics	
Minimum	0.371%
Maximum	1.935%
Mean	1.014%
Standard Deviation	0.348%
Range	1.563%

Table 13: Money market data characteristics

The line chart in Figure 26 plots the actual monthly return on the money market from 1975 to 2010. Further, the mean, maximum, minimum and one standard deviation from the mean are superimposed on the chart.

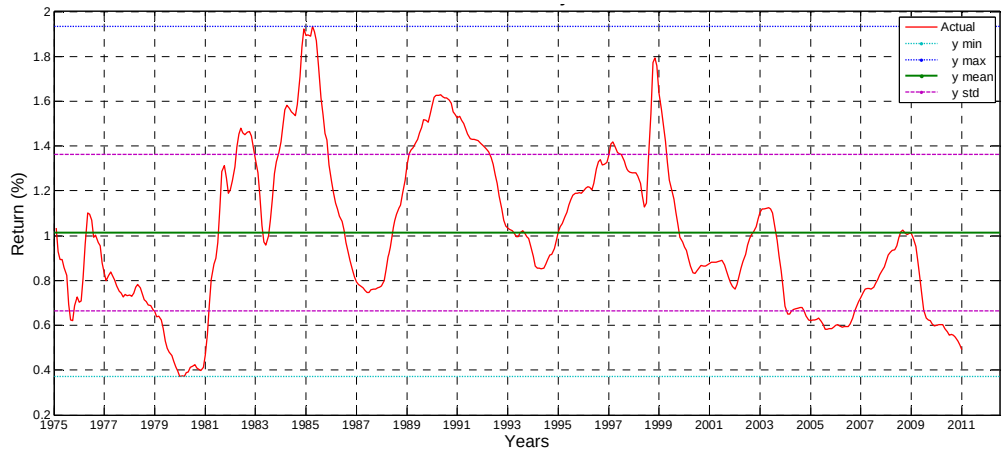

Figure 26: Actual monthly returns on the money market 1975 – 2010, including mean, max, min, and one standard deviation from the mean

The column chart in Figure 27 plots the number of observations in each return category against its corresponding return category to provide a graphical estimation of the density function associated with money market returns. The line, superimposed on the chart, aids in the graphical representation of the estimated density function.

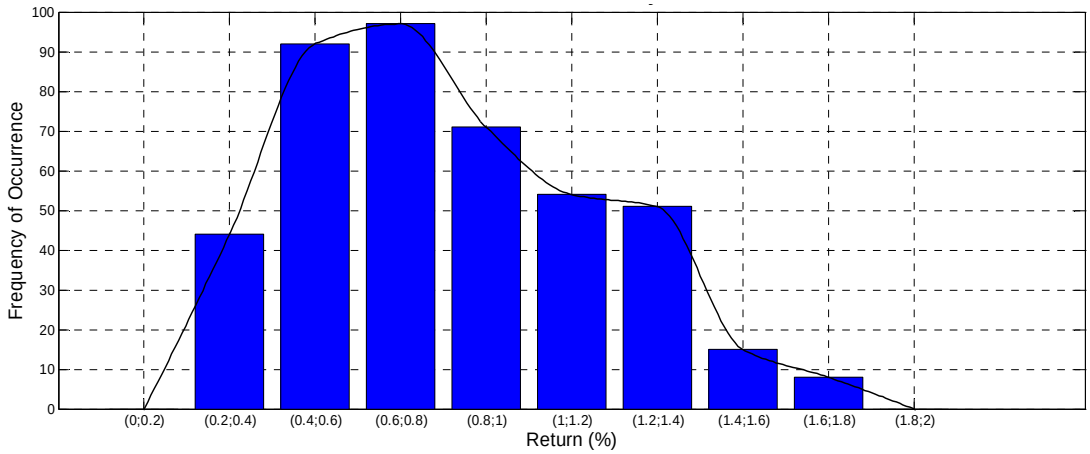

Figure 27: Distribution of returns on the money market with a graphically estimated density function

3.2.2.2 Discussion of Money Market Data

The average return on the money market is approximately 1% per month (or 12.7% per year) over the period considered. No negative returns from the money market over the same period are observed. Figure 26 suggests there are some relationships in the data and little random variation. This is supported by the small standard deviation (0.348%) compared to the mean (1.014%). Figure 27 indicates the distribution of returns is skewed to the right and the returns are bound between 0% and 2% per month. No seasonality is detected.

3.2.3 Bond Market Data [37]

The period considered (1975 – 2010) includes the construction of the JSE-Actuaries All Bond Index in 1986. The JSE-Actuaries All Bond Index is used to evaluate the performance of the bond market from 1986 onwards. For the period 1975 – 1986 the performance on long term bonds are used as a proxy of the bond market's performance. Due to the construction of the JSE-Actuaries All Bond Index in 1986, inconsistencies in the data are expected.

The All Bond Index has two constituents, a capital section and an income section. These sections are widely known as the Price and the Yield Index. The Price Index is the market capitalisation of a portfolio of bonds. The volume of each bond is weighted in proportion to the amount held privately. The Yield Index is constructed by considering the closing yield on the JSE Gilt floor, which provides a daily index. The constituents comprising the portfolio of bonds are reviewed quarterly. The income accrued on the bonds is assumed to be smooth and the actual discrete income payments are ignored. This ensures a smooth progression of indices over a single year. The monthly yield and price are calculated on the last day of each month. This implies they are date dependent and may be sensitive to random fluctuations.

3.2.3.1 Analysis of Return on the Bond Market

Table 14 provides summary statistics on the bond market data set considered.

Statistics	
Minimum	-14.457%
Maximum	11.407%
Mean	1.072%
Standard Deviation	2.443%
Range	25.864%

Table 14: Bond market data characteristics

The line chart in Figure 28 plots the return on the bond market from 1975 to 2010 on a monthly basis. Further, the mean, maximum, minimum and one standard deviation from the mean are superimposed on the chart.

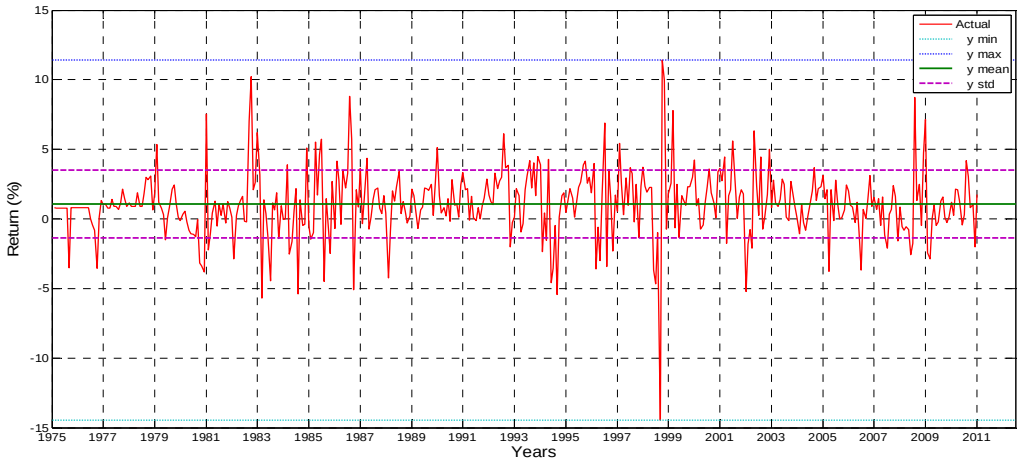

Figure 28: Actual monthly returns on the bond market from 1975 – 2010, including mean, max, min, and one standard deviation from the mean

The column chart in Figure 29 plots the number of observations in each return category against the return category, to provide a graphical estimation of the density function of the bond market returns. The curve, superimposed on the chart, aids in the graphical representation of the estimated density function.

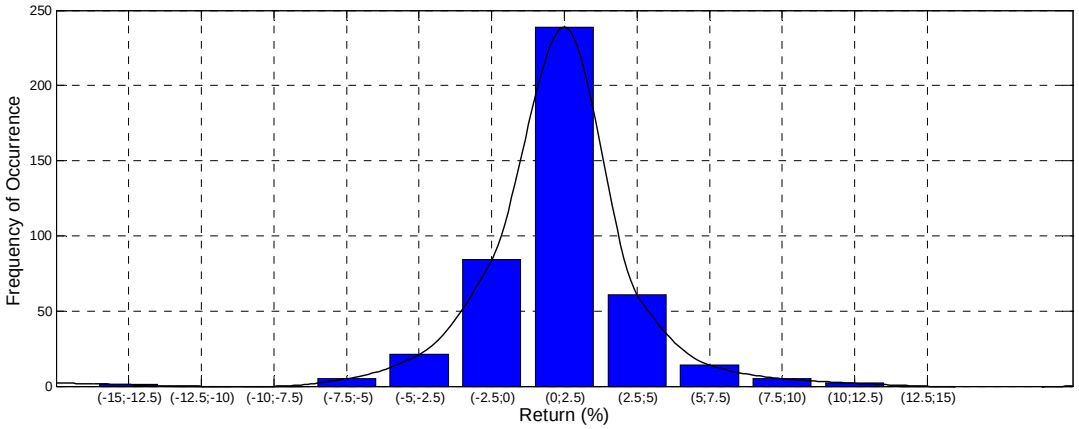

Figure 29: Distribution of returns on the bond market with a graphically estimated density function

3.2.3.2 Discussion of Bond Market Data

The average monthly return on the bond market is approximately 1% (or 12.7% per year) over the period considered. Several negative returns are observed over the period considered, with the greatest being in 1998. The largest positive return is also observed in 1998. This suggests the time series exhibits hetroskedasticity, but is not explicitly tested for. Figure 29 indicates the distribution of returns is fairly symmetrical with thin tails. This is unexpected as returns on the bond market are believed to have a Leptokurtic distribution with thick tails. The observations are bound by a maximum of 11.5% and a minimum of -14.5% per month. No seasonality is detected in the data. There is significant volatility present in the data as indicated by the standard deviation of 2.443%, being more than twice the mean, 1.072%.

3.2.4 Equity Market Data [37]

The data used to determine returns from the equity market is based on the available JSE-Actuaries All Share Index, from 1978 to 2010. Prior to this, the Rand Daily Mail Industrial Index (RDM100) is used. The RDM100 constituents were changed infrequently by the editors of the Rand Daily Mail. Further, limited information is available on the methodology behind the calculation of the RDM100. The methodology behind the calculation of these constituents improved in 1995 and now covers the full range of equity investments.

The index consists of a price and income index. The price index is based on a market capitalization weighted average of its components. Readings were taken every two minutes while trading was possible. The income portion of the index is the dividends yield on the constituents. It is assumes that dividends are received evenly throughout the year, which does not hold true. However, this is a reasonable assumption based on a study conducted which investigated the timing of dividend payments during the year [37].

The data is converted to monthly returns based on the price index. The index is assumed to be purchased at the beginning of the month and sold at the end. The month's dividend is assumed to be received half-way through the month and is reinvested in the index at its average price over that specific month. The return on the index is determined by dividing the end of month value (including reinvestment income) by the investment at the beginning of the month.

3.2.4.1 Analysis of Return on the Equity Market

Table 15 provides summary statistics on the equity market data set considered.

Statistics	
Minimum	-29.706%
Maximum	17.846%
Mean	1.660%
Standard Deviation	6.254%
Range	47.552%

Table 15: Equity market data characteristics

The line chart in Figure 30 plots the return on the equity market from 1975 to 2010, on a monthly basis. Further, the mean, maximum, minimum and one standard deviation from the mean are superimposed on the chart.

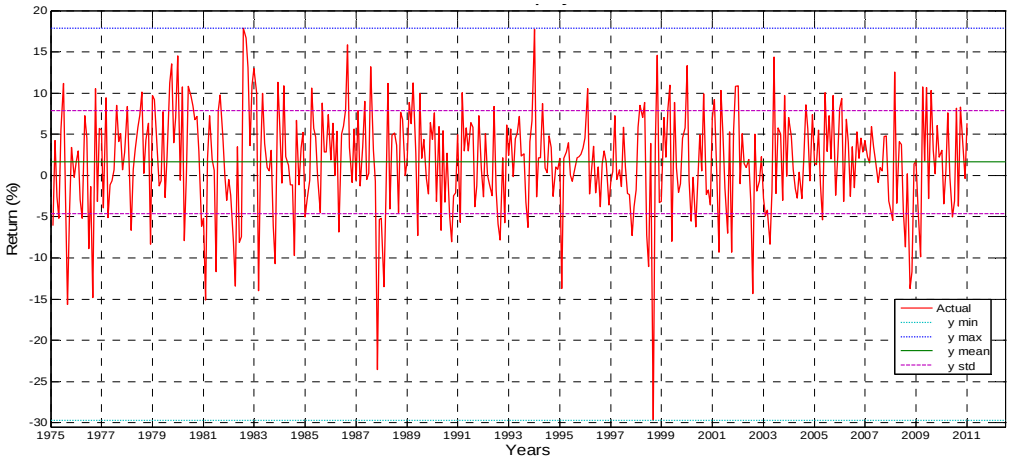

Figure 30: Actual monthly returns on the equity market from 1975 – 2010, including mean, max, min, and one standard deviation from the mean

The column chart in Figure 31 plots the number of observations in each return category against its corresponding return category, to provide a graphical estimation of the density function associated with equity market returns. The line, superimposed on the chart, provides graphical representation of the estimated density function.

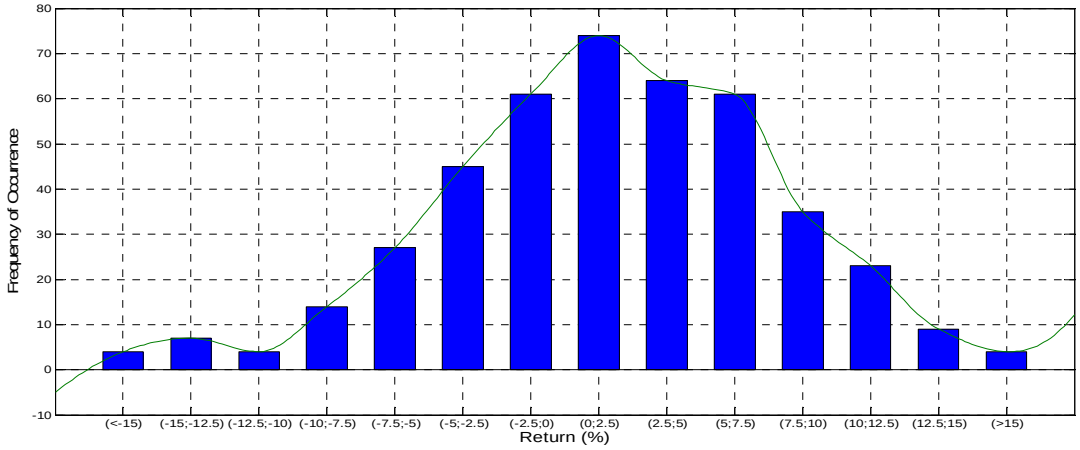

Figure 31: Distribution of returns on the equity market with a graphically estimated density function

3.2.4.2 Discussion of Equity Market Data

The average monthly return on the equity market is approximately 1.7% (or 22% per year). The returns have a large standard deviation (6.25%) in comparison to the mean (1.66%). This indicates significant volatility associated with the equity market. There are high negative and positive returns on this market. The index was not influenced as significantly by the global financial crisis in 2008, as expected. Further, high negative returns are followed by high positive returns, which suggest the data is hetroskedastic and the presence of volatility clustering. This is expected, but further investigations to confirm this are not performed.

Figure 31 indicates the majority of observations are between -7.5% and 12.5%, with most observations between 0% and 2.5%. The observations are limited to a maximum of 17.8% and a minimum of -29.7%. No seasonality is detected in the data. The distribution is non-symmetrical and has a large range.

3.2.5 Correlation between Data

Relationships between the data sets are determined through the use of two measures of correlation. Pearson's Rho is used to calculate the linear correlation in the data while Spearman's Rho is used to determine the rank correlation. The correlation matrices are captured in Table 16 and Table 17.

Pearson's Rho - Correlation Matrix				
	Inflation	Money	Bond	Equity
Inflation	1.00	0.20	-0.07	0.02
Money	0.20	1.00	0.14	-0.05
Bond	-0.07	0.14	1.00	0.30
Equity	0.02	-0.05	0.30	1.00

Table 16: Pearson's Rho correlation matrix

Spearman's Rho - Correlation Matrix				
	Inflation	Money	Bond	Equity
Inflation	1.00	0.24	-0.13	0.02
Money	0.24	1.00	0.15	-0.07
Bond	-0.13	0.15	1.00	0.21
Equity	0.02	-0.07	0.21	1.00

Table 17: Spearman's Rho correlation matrix

The correlation matrices indicate that money market return has an unavoidable relationship with inflation. The correlation between money market returns and inflation is approximately 0.2. There is negative correlation between inflation and return on the bond market. These are close to zero and are not expected to have a significant effect on the individual markets. It is noted that the rank correlation suggests a greater relationship between the two variables, than the linear correlation. This is interesting as it indicates significant non-linear relationships between inflation and the return on the bond market. There is no relationship between inflation and the return on the equity market. This is expected as the equity market is driven by supply and demand, as well as factors other than inflation. The correlation measures indicate little correlation between returns on the money and bond market. This is also the case for returns on the money and equity market. Significant relationships exist between the returns on the bond and equity market, as the linear correlation measure is 0.3. This is the greatest correlation throughout all the data sets. The correlation is

high in relative terms but not in absolute terms, indicating that important external explanatory variables are not included in the data, as expected.

3.2.6 Source of Data

The data analysed is obtained from the following studies:

- Firer, C and Staunton, M., 2002. 102 Years of South African financial market history. Investment Analysts Journal, 56, 57-65; and
- Firer, C. and McLeod, H., 1999. Equities, Bonds, Cash and Inflation: Historical performance in South Africa, 1925 - 1998. Investment Analysts Journal, 50, 7-28.

3.3 Construction Process

To determine an efficient traditional time series model the following process is used.

1. The data is collected, analysed and amended for the application.
2. Two mutually exclusive data sets are constructed using the collected data. The sets are name 'Training' set and 'Testing' set.
3. Statistical tests are performed on the data. The statistical tests conducted use a 5% level of significance.
4. Autocorrelation and partial autocorrelation plots are constructed and analysed.
5. Several best fit models are determined using the 'Time Series Forecasting System' in SAS[2]
6. Considering the Root Mean Squared Error (RMSE) and the R-squared value, over the testing data, the model which has the smallest RMSE and greatest R-squared value is selected as the best performing model.
7. The RMSE (Equation 40) and R-squared value (Equation 39) over both sets of data is determined for the one and three-month forecast periods.

This process results in the construction of a traditional model for the money, bond and equity markets, as well as inflation.

3.3.1 Data Sets

The data is divided into two data sets for fitting and comparing the traditional models.

Training Data Set

This set ranges from January 1975 to August 2002 and contains the majority of observations, 332. It is used to fit the traditional models and train the ANNs.

Testing Data Set

This set ranges from September 2002 to December 2010 and consists of observations not contained within the training data set, 100 observations. It is important to ensure the training and testing sets are mutually exclusive.

[2] *The models selected are those auto fitted through the use of the Time Series Forecasting System in SAS. The chosen models are aligned with common methods used for solving time-series forecasting problems. These are consistent with the traditional models used in past comparisons of ANNs to traditional models.*

3.3.2 R-squared value (Coefficient of determination)

A description of the R-squared value, used in the selection of the traditional models, is given below.

$$\text{Coefficient of Determination} = R^2 \text{, and}$$

$$R^2 \in (-\infty; 1]$$

$$R^2 = 1 - \frac{SSE}{SST}$$

Equation 39: Coefficient of determination (R-Squared value)

where $SSE = \sum_{i=1}^{n} (\hat{x}_i - x_i)^2$, $SST = \sum_{i=1}^{n} (x_i - \bar{x})^2$, $\bar{x} = \sum_{i=1}^{n} \frac{1}{n} x_i$, n is the number of data points, $\hat{x}_i$ is the forecast value of data point i and x_i is the actual value of data point i.

If $R^2 = 1$, the regression line perfectly fits the data and the model is a perfect fit. If $R^2 \leq 0$, the regression line does not fit the data well and the mode is a poor fit. In this case the mean of the data set used for fitting will provide a better fit than the model. Alternatively, if $0 < R^2 < 1$ the goodness of fit depends on the value of the measure. Values close to 1 indicate a good fit and values close to 0 indicate a poor fit.

3.3.3 Root Mean Squared Error (RMSE)

A description of the Root Mean Squared Error (RMSE), used to supplement the selection process of traditional models and ANNs, is given below.

$$RMSE = \sqrt{\frac{1}{n} \sum_{t=1}^{n} \varepsilon_t^2} \text{, therefore}$$

Equation 40: Root Mean Squared Error function

$$\therefore RMSE \in [0, \infty)$$

where $\varepsilon_t = (\hat{X}_t - X_t)$, $\hat{X}_t$ is the forecast value at time t, X_t is the actual value at time t and n is the number of forecasts. The RMSE is interpreted as follows. If RMSE = 0, the observed and forecast variables are identical. If RMSE > 0, there is some error between the forecast and observed values of the variable. In particular, if RMSE far greater than 0, there is a large error between the forecast and observed values of the variable. The size of the error determines the reliability of the model.

3.4 Inflation Application

An efficient traditional inflation forecasting model is developed in this section. The model uses past observations of inflation to forecast future inflation over a one and three-month period.

3.4.1 Time Series Characteristics – Inflation

To construct forecasting models it is important to determine the characteristics underlying the data set. Tests to determine if the process is stationary and/or white noise (random) are performed. The autocorrelations and partial autocorrelation of the data are then analysed.

3.4.1.1 Stationarity and Dependence

The Ljung-Box Chi-Squared Statistic indicates the process is not white noise[3]. This implies there are significant relationships between successive observations of inflation. The Dickey-Fuller Single Mean Test indicates the series is stationary. The results of the conducted tests are illustrated in Table 18.

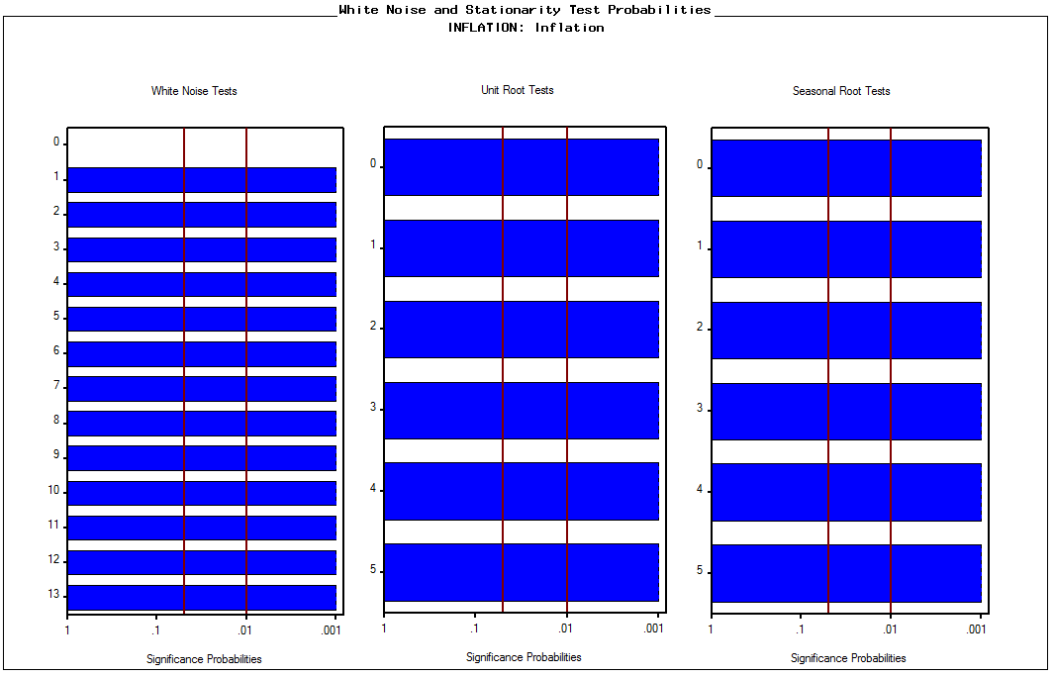

Table 18: Result from testing for white noise and stationarity – inflation

3.4.1.2 Autocorrelations and Partial Autocorrelations

The autocorrelation plot of inflation indicates there are significant relationships between successive lags, as expected. The significance of the lag decreases as number of the lags increase. An exception occurs at a period of 12 months, which suggests a seasonal pattern with a period of 12 months exists.

The autocorrelation and partial autocorrelation plots of the inflation data once the seasonality has been removed, Table 19, indicates the autocorrelations decrease as the number of lags increase. The partial autocorrelations indicate the three most recent observations have a significant effect on the current value of inflation.

[3] *White noise processes imply there are no significant correlations between the observations at different times.*

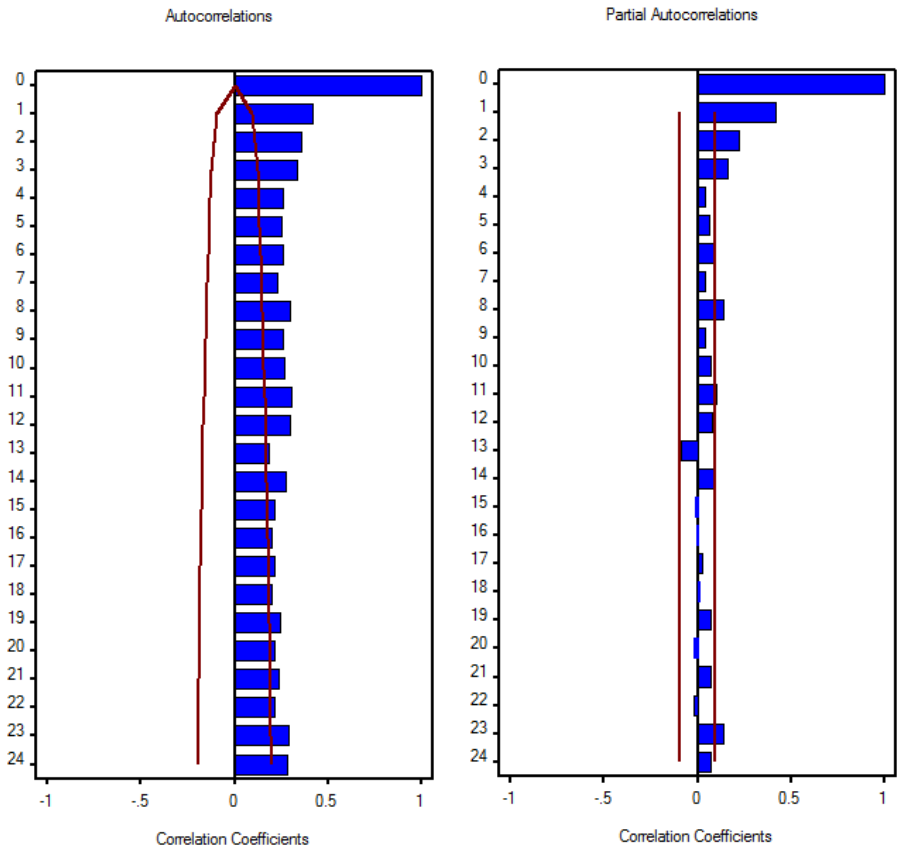

Table 19: Autocorrelation and partial autocorrelation plots of inflation after seasonality is removed

3.4.2 Best Fit Models – Inflation

Several models are fitted to the time series and are presented in Table 20.

Model Number	Model Type	RMSE (Testing Set)	R-Squared (Testing Set)
1	Seasonal Exponential Smoothing	0.004631	0.220
2	ARIMA(0,1,1)	0.005128	0.043
3	Seasonal Dummy	0.006687	-0.627
4	Seasonal Dummy + AR(2)	0.004993	0.093
5	Seasonal Dummy + ARMA(2,1)	0.004492	0.266
6	Seasonal Dummy + AR(3)	0.004799	0.162

Table 20: Best fit traditional models – model errors – inflation

3.4.3 Efficient Model – Inflation

Model 5 has the smallest RMSE and greatest R-squared value of the models constructed, which results in it being the efficient model. The model consists of 12 seasonal dummies with two autoregressive variables and a single moving average term. The formula for the model is given in Equation 41.

$$Y_t = SD_t + X_t, \text{ and}$$

$$X_t = a + \Phi_1 X_{t-1} + \Phi_2 X_{t-2} + \varepsilon_t + \theta_1 \varepsilon_{t-1}$$

Equation 41: Hybrid seasonal and ARMA(2,1) model for inflation

where Y_t is inflation at time *t*, X_t is inflation less the seasonal dummy at time *t*, a is the intercept, Φ_1 is the coefficient of the first lagged observation, Φ_2 is the coefficient of the second lagged observation, ε_t is the error term (the expected value of ε_t is zero), θ_1 is the coefficient of the first lagged residual and SD_t is the seasonal dummy at time *t*.

The parameter estimates for this model are given in Table 21.

Parameters	Estimate	Parameters	Estimate
Intercept	0	Seasonal dummy 5	0.0002228
Lag 1 coefficient (Φ_1) - AR	1.10301	Seasonal dummy 6	0.0009143
Lag 2 coefficient (Φ_2) - AR	-0.11357	Seasonal dummy 7	0.00616
Lag 1 error term (θ_1) - MA	0.93189	Seasonal dummy 8	0.00266
Seasonal dummy 1	0.00464	Seasonal dummy 9	0.00346
Seasonal dummy 2	0.00158	Seasonal dummy 10	0.00144
Seasonal dummy 3	0.00353	Seasonal dummy 11	-0.0006362
Seasonal dummy 4	0.00467	Seasonal dummy 12	0

Table 21: Efficient inflation model – parameters

3.4.3.1 Seasonal Dummies

Seasonal dummies are additional variables used to capture seasonal trends in the data. Each seasonal dummy is activated when a specific period is considered. Only one seasonal dummy can be active at any point in time. As there is a seasonal trend with a period of 12 months, there are 12 seasonal dummies. Each dummy is active in a specific month.

3.4.4 Model Error – Inflation

The RMSE over the training and training set, for one and three-month forecast periods, are tabulated in Table 22.

Data Set and Forecast Period	RMSE
One month forecast	
Training data set	0.004922
Testing data set	0.004492
Three-month forecast	
Training data set	0.005032
Testing data set	0.004921

Table 22: Efficient inflation model - errors

3.4.5 Forecast and Actual Inflation

The line chart below, Figure 32, plots the forecast one and three-month, and actual inflation from 1975 to 2010.

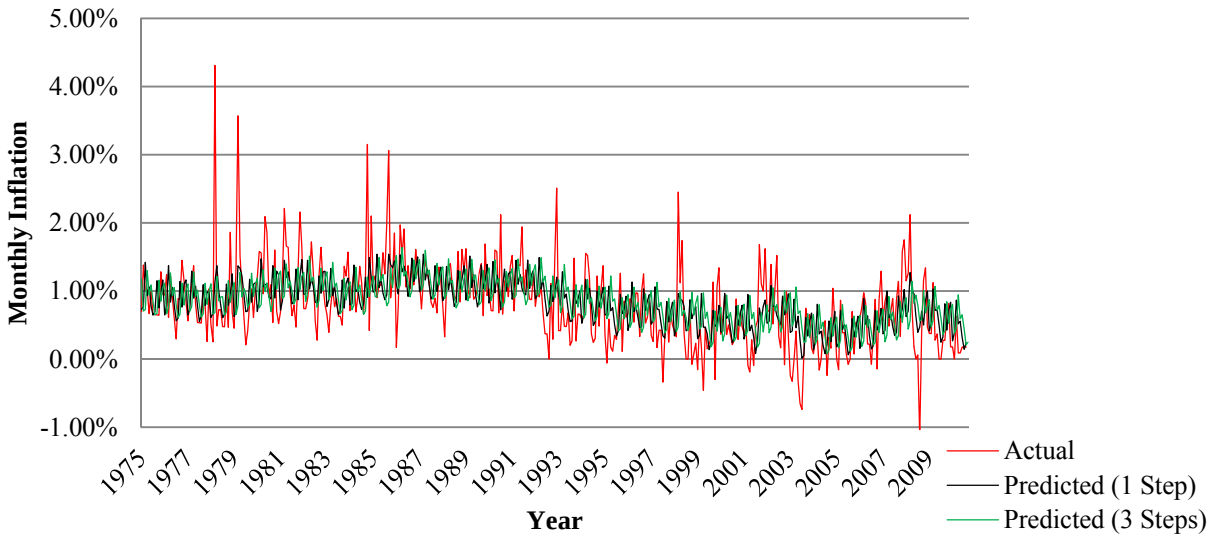

Figure 32: Forecast and actual inflation from 1975 to 2010 – traditional time series forecasting models

The one-month forecasts are efficient estimates of inflation over the period considered. There are several outlier in the data that the model does not foresee, as indicated by the infrequent spikes in that actual inflation values. The three-month forecasts are less accurate than the one month forecasts. This is expected as accuracy decreases as the forecast period increases.

3.5 Money Market Application

The traditional model responsible for forecasting the return on the money market over a one and three-month period is constructed in this section. Only past observations of return are used to forecast future returns on the money market.

3.5.1 Time Series Characteristics – Money Market

To construct forecasting models it is important to determine the characteristics underlying the data set. Tests to determine if the process is stationary and/or white noise are performed. The autocorrelations and partial autocorrelations of the data are then analysed.

3.5.1.1 Stationarity and Dependence

The Ljung-Box Chi-Squared Statistic indicates the process is not white noise. This implies there are significant relationships between successive observations of return on the money market. The Dickey-Fuller Single Mean Test indicates the series is not stationary. The simple difference of the series is calculated and tested, which proves to be stationary. The results of these tests are provided in Table 23.

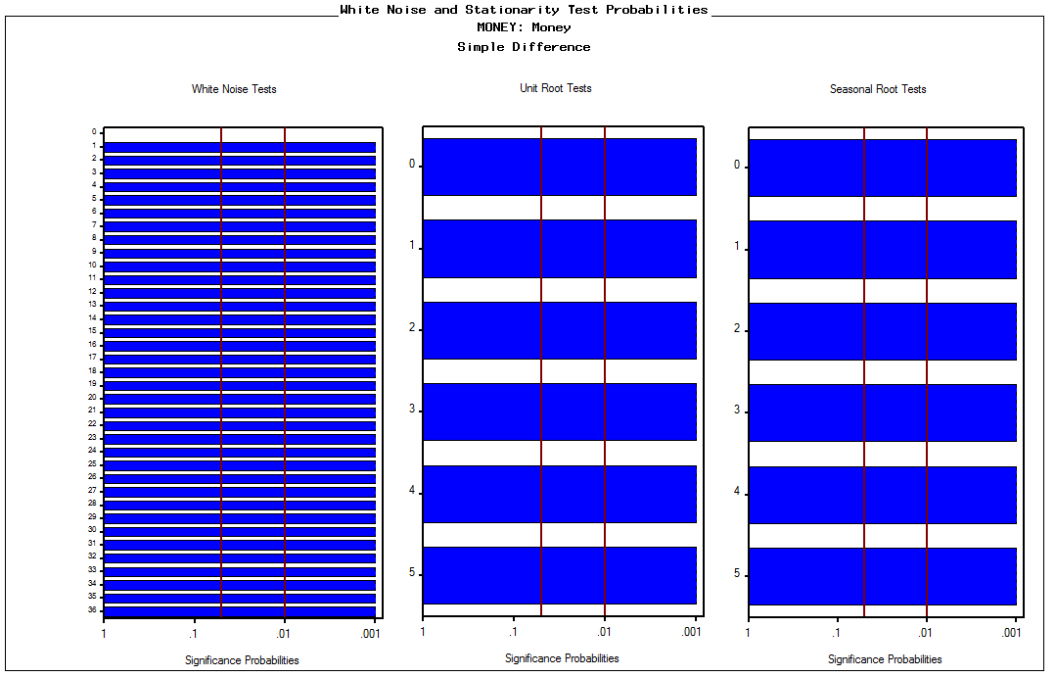

Table 23: Results from testing for white noise and stationarity – simple difference – money market

3.5.1.2 Autocorrelations and Partial Autocorrelations

The autocorrelation plot of the simple difference series, Table 24, indicates there are significant relationships between successive lags, as expected. This significance decreases as the number of lags increase. The two most recent observed returns have significant correlation with the current return, as indicated by the autocorrelation plot in Table 24. The results from the partial autocorrelation plots are less clear.

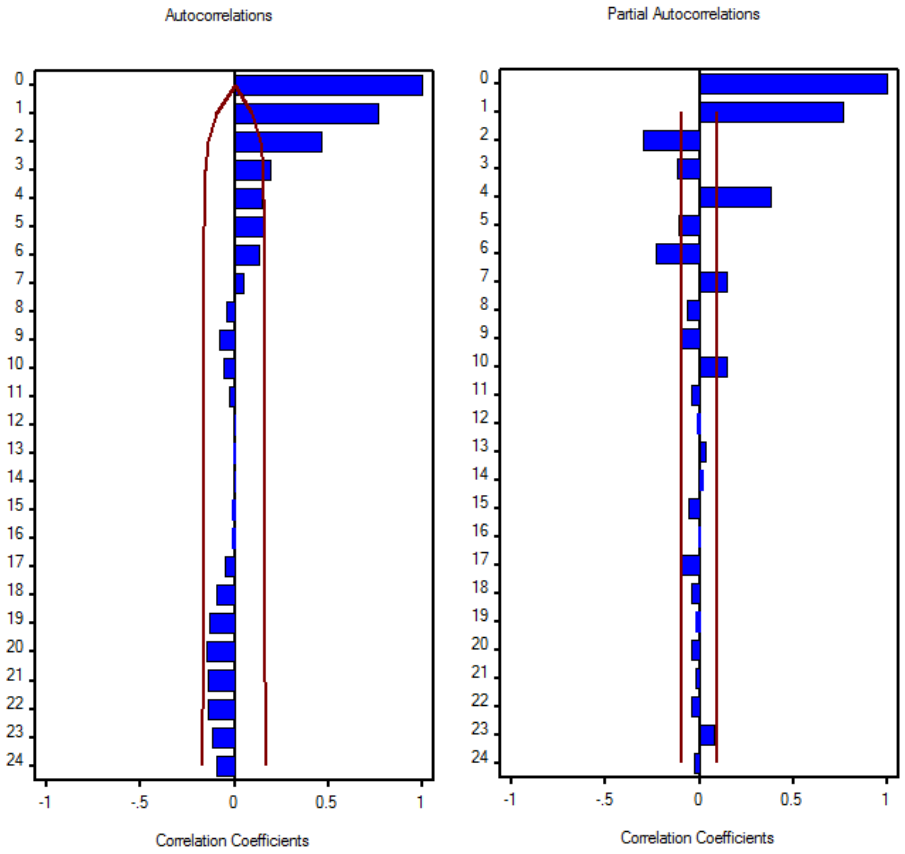

Table 24: Autocorrelation and partial autocorrelation plots of returns on the money market after simple difference

3.5.2 Best Fit Models – Money Market

Several models are fitted to the money market time series. Each is presented in Table 25.

Model Number	Model Type	RMSE (Testing Set)	R-Squared (Testing Set)
1	Simple Exponential Smoothing	0.0002674	0.978
2	AR(2)	0.0001417	0.994
3	ARI (2,1)	0.0001162	0.996
4	Log Simple Exponential Smoothing	0.0002674	0.978
5	Mean	0.0038147	-3.398
6	Log Mean	0.0032911	-2.274

Table 25: Best fit traditional models – model errors – money market

3.5.3 Efficient Model – Money Market

Model 3 has the smallest RMSE and greatest R-squared value of the models constructed, which result in it being chosen as the efficient model. The model consists of a simple difference with two autoregressive terms. The formula of the model is presented in Equation 42.

$$\Delta X_t = \Phi_1 \Delta X_{t-1} + \Phi_2 \Delta X_{t-2} + \varepsilon_t$$

Equation 42: Money market model with 2 autoregressive variables and a simple difference

where $\Delta X_t = X_t - X_{t-1}$, X_t is the return at time t, Φ_1 is the coefficient of the first lagged observation, Φ_2 is the coefficient of the second lagged observation and ε_t is the error term.

The parameter estimates for this model are given in Table 26.

Parameters	Estimate
Lag 1 coefficient (Φ_1) - AR	1.00405
Lag 2 coefficient (Φ_2) - AR	-0.31480

Table 26: Efficient model for returns on the money market - parameters

3.5.4 Model Error – Money Market

The RMSE over the training and testing sets are tabulated below (Table 27). The RMSE is calculated for a one and three-month forecast period.

Data Set and Forecast Period	RMSE
One month forecast	
Training data set	0.000305
Testing data set	0.000116
Three-month forecast	
Training data set	0.001073
Testing data set	0.000525

Table 27: Efficient money market model – errors

3.5.5 Forecast and Actual Returns – Money Market

The line chart in Figure 33 illustrates the forecast one and three-month, and actual returns on the money market from 1975 to 2010.

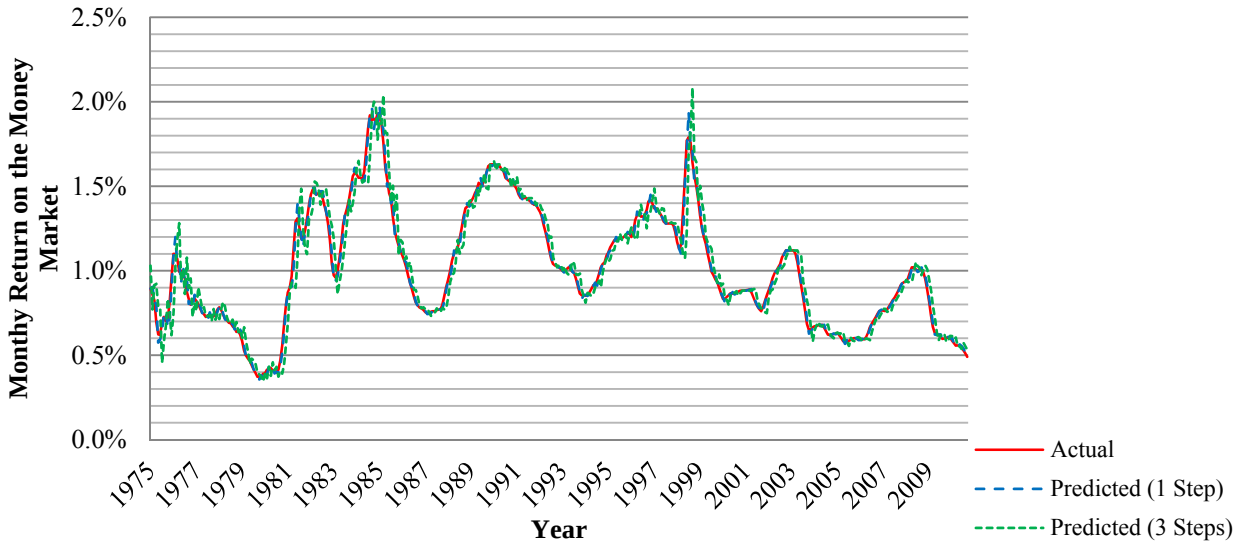

Figure 33: Forecast and actual returns on the money market from 1975 to 2010 – traditional time series forecasting models

The one month forecasts are an accurate estimate of the actual returns on this market. This can be explained through the high predictability of this market in the short term. The three-month forecasts are less representative of the actual returns. As expected, the error associated with forecasts increase as the length of the forecast period increases.

3.6 Bond Market Application

The traditional model responsible for forecasting the return on the bond market over a one and three-month period is constructed in this section. Only past observations of return are used to forecast future returns on the bond market.

3.6.1 Time Series Characteristics – Bond Market

To construct forecasting models it is important to determine the characteristics underlying the data set. Tests to determine if the process is stationary and/or white noise are performed. The autocorrelations and partial autocorrelations of the data are then analysed.

3.6.1.1 Stationarity and Dependence

The Ljung-Box Chi-Squared Statistic indicates the process is not white noise, but is not as comprehensive as in the other cases. If a 1% level of significance is considered, the conclusion that this process is white noise is made. The Dickey-Fuller Single Mean Test indicates the series is stationary. The results of the conducted tests are provided in Table 28.

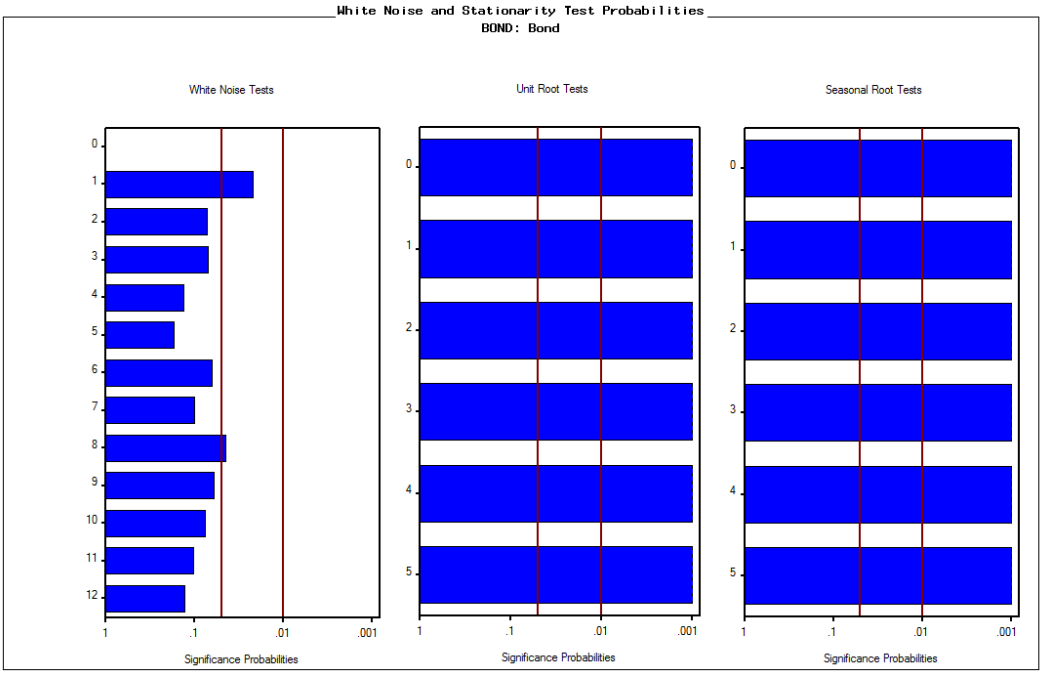

Table 28: Result from testing for white noise and stationarity – bond market

3.6.1.2 Autocorrelations and Partial Autocorrelations

The autocorrelation plot indicates there is significant correlation between the current and most recent observation. Further lags do not have statistically significant relationships with the current observation. The partial autocorrelation plot, similar to the autocorrelation plot, suggests the most recent return on the bond market has a significant effect on the observed value. The autocorrelations and partial autocorrelations of the return on the bond market time series are plotted in Table 29.

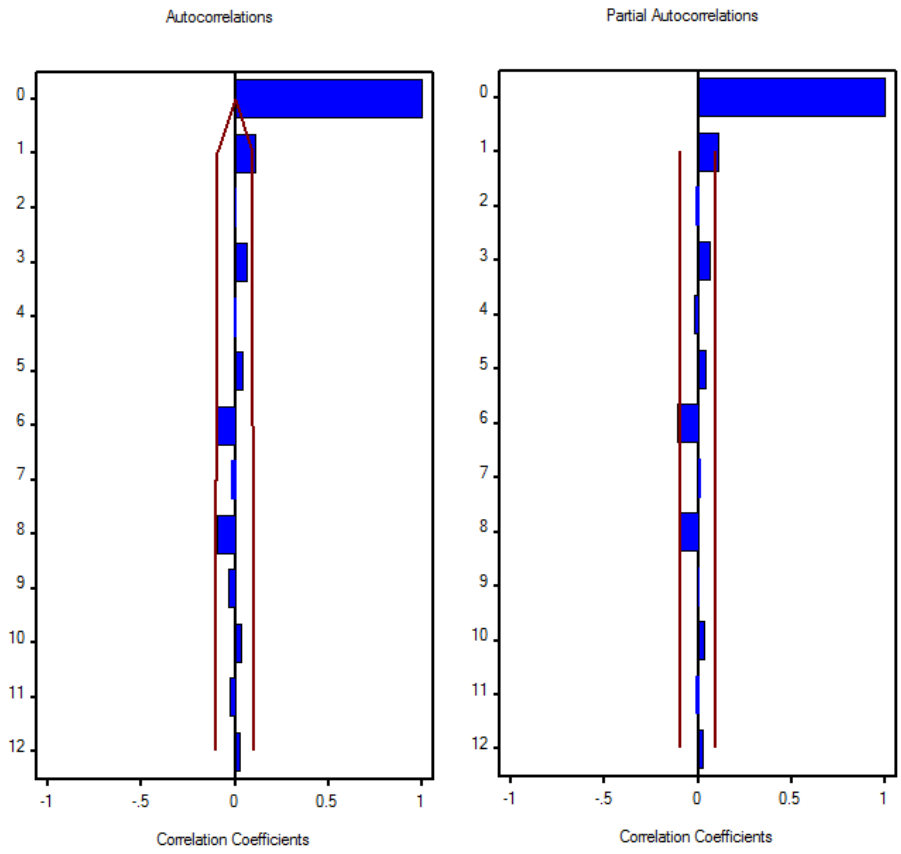

Table 29: Autocorrelation and partial autocorrelation plots of returns on the bond market

3.6.2 Best Fit Models – Bond Market

Three models are fitted to the time series. Each is presented in Table 30.

Model Number	Model Type	RMSE (Testing Set)	R-Squared (Testing Set)
1	Simple Exponential Smoothing	0.01937	-0.013
2	AR(1)	0.01906	-0.025
3	Mean	0.01925	0.007

Table 30: Best fit traditional models – model errors – bond market

3.6.3 Efficient Model – Bond Market

The model which has the smallest RMSE and greatest R-squared value is chosen as the efficient model. In this case the autoregressive model with a single lagged term, model 2, is selected. The formula of the model is presented in Equation 43.

$$X_t = a + \Phi_1 X_{t-1} + \varepsilon_t$$

Equation 43: Bond market model with 1 autoregressive variable

where a is the intercept, X_t is the return at time t, Φ_1 is the coefficient of the first lagged observation and ε_t is the error term.

The parameter estimates for this model are given in Table 31.

Parameters	Estimate
Intercept	0.01004
Lag 1 coefficient (Φ_1) - AR	0.10419

Table 31: Efficient model for returns on the bond market - parameters

3.6.4 Model Error – Bond Market

The RMSE over the training and testing sets are tabulated below (Table 32). It is calculated for a one and three-month forecast period.

Data Set and Forecast Period	RMSE
One month forecast	
Training data set	0.02566
Testing data set	0.01906
Three-month forecast	
Training data set	0.02586
Testing data set	0.01925

Table 32: Efficient bond market model - errors

3.6.5 Forecast and Actual Returns – Bond Market

The line chart in Figure 34 plots the forecast one and three month, and actual returns on the bond market from 1975 to 2010.

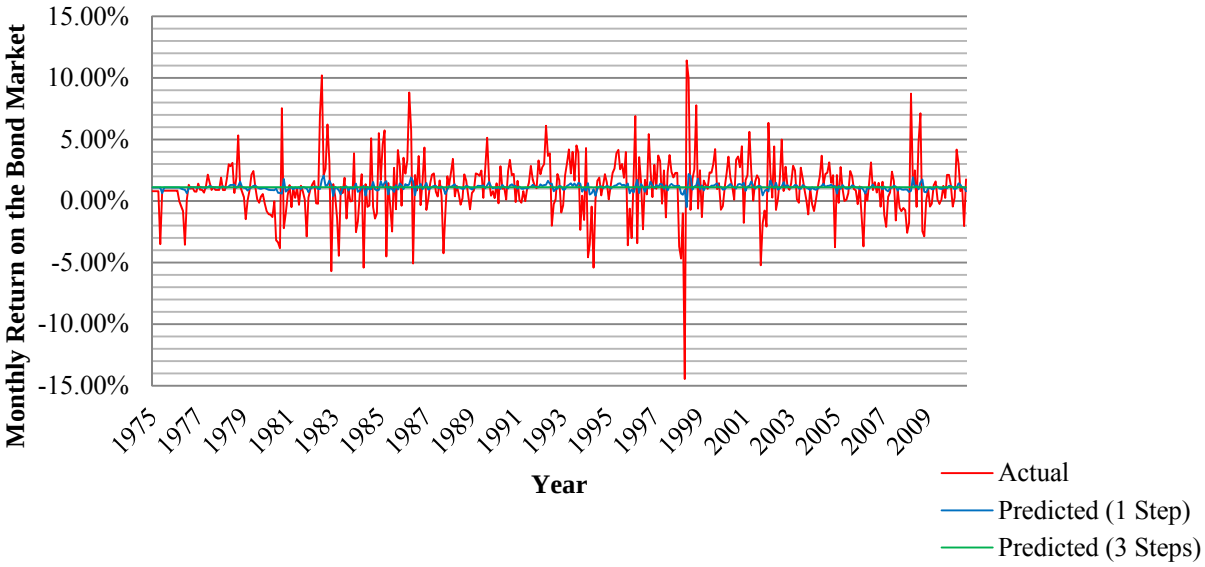

Figure 34: Forecast and actual returns on the bond market from 1975 to 2010 – traditional time series forecasting models

The one and three-month forecasts are poor representations of the actual returns on the bond market over the period considered. This is due to the large effect external factors have on the returns from this market. The forecasts generated by this model are not suitable for practical purposes, due to their high level of inaccuracy.

3.7 Equity Market Application

The traditional model responsible for forecasting the return on the equity market over a one and three-month period is constructed in this section. Only past observations of return are used to forecast future returns on the equity market.

3.7.1 Time Series Characteristics – Equity Market

To construct forecasting models it is important to determine the characteristics underlying the data set. Tests to determine if the process is stationary and/or white noise are performed. The autocorrelations and partial autocorrelations of the data are then analysed.

3.7.1.1 Stationarity and Dependence

The Ljung-Box Chi-Squared Statistic indicates the process is white noise. This implies there are no significant relationships between successive returns on the equity market. The Dickey-Fuller Single Mean Test indicates the series is stationary. The results of the conducted tests are provided in Table 33.

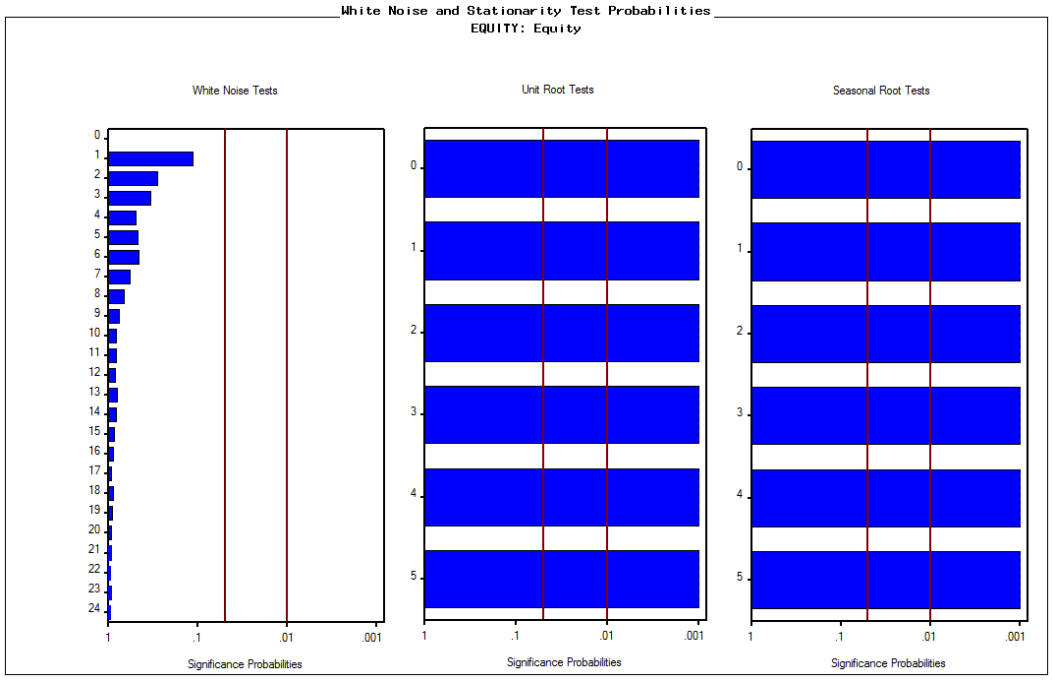

Table 33: Result from testing for white noise and stationarity – equity market

3.7.1.2 Autocorrelations and Partial Autocorrelations

The autocorrelation plot indicates there are no significant correlations between the current and any previous returns. The partial autocorrelation plot confirms there are no significant relationships between returns. Further, no seasonality is detected. The autocorrelations and partial autocorrelations of the return on the equity market time series are plotted in Table 34.

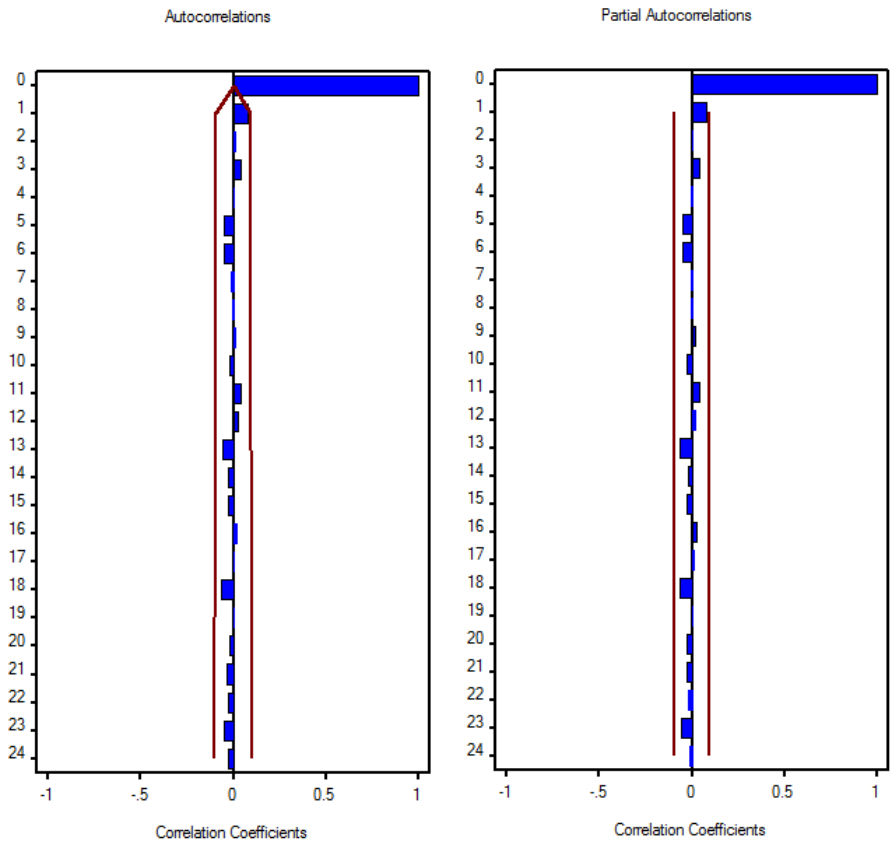

Table 34: Autocorrelation and partial autocorrelation plots of returns on the equity market

3.7.2 Best Fit Models – Equity Market

Several models are fitted to the time series. Each is presented in Table 35.

Model Number	Model Type	RMSE (Testing Set)	R-Squared (Testing Set)
1	Mean	0.05278	0
2	AR(1)	0.05303	-0.006
3	Simple Exponential Smoothing	0.05292	-0.010

Table 35: Best fit traditional models – model errors – equity market

3.7.3 Efficient Model – Equity Market

Model 1 has the smallest RMSE and greatest R-squared value of the models constructed, which result in it being chosen as the efficient model. The model consists of only an intercept term. The formula for the model is presented in Equation 44.

$$X_t = a + \varepsilon_t$$

Equation 44: Equity market mean model

where a is the intercept, X_t is the return at time *t* and ε_t is the error term.

The parameter estimate for this model is given in Table 36.

Parameter	Estimate
Intercept	0.01684

Table 36: Efficient model for returns on the equity market - parameter

3.7.4 Model Error – Equity Market

The RMSE over the training and testing sets are tabulated below (Table 37). They are calculated using a one and three-month forecast period.

Data Set and Forecast Period	RMSE
One month forecast	
Training data set	0.06510
Testing data set	0.05278
Three-month forecast	
Training data set	0.06522
Testing data set	0.05278

Table 37: Efficient equity market model - errors

3.7.5 Forecast and Actual Returns – Equity Market

The line chart in Figure 35 illustrates the forecast one and three-month, and actual returns on the equity market from 1975 to 2010.

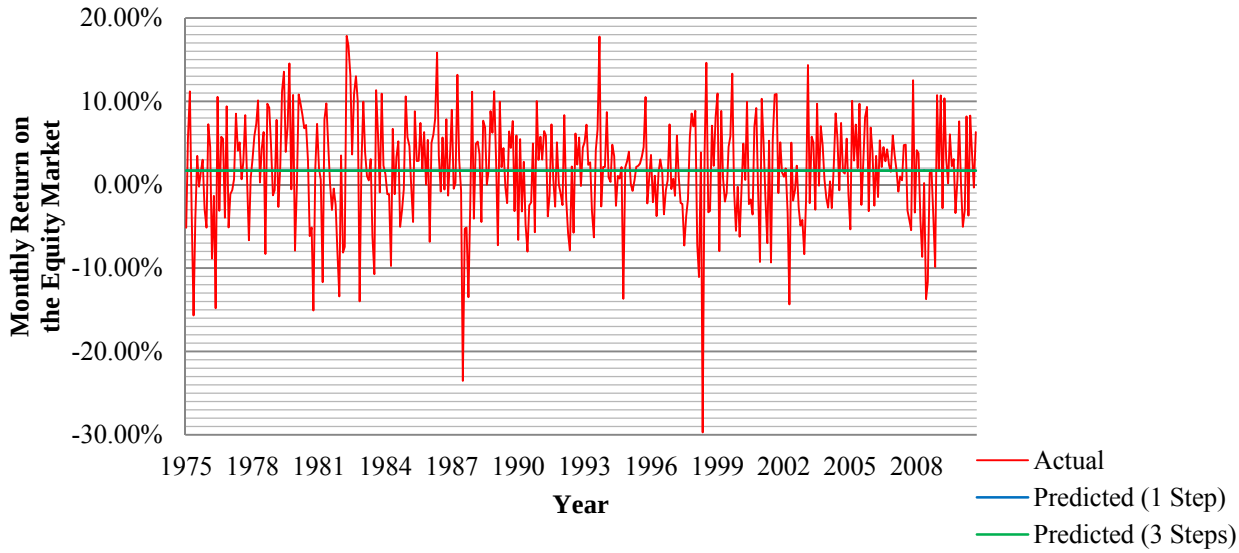

Figure 35: Forecast and actual returns on the equity market from 1975 to 2010 – traditional time series forecasting models

The return on the equity market is unpredictable using only past observations. This is expected as the return on this market is sensitive to external factors, which are not captured here. The model constructed has poor forecasting accuracy and should not be used in practise. It is important to include additional explanatory variables in the model before it is applied to forecast these returns.

3.7.6 Geometric Brownian Motion – Equity Market

The traditional models considered in the previous section to forecast the equity market did not result in any relationships being detected. The best model obtained is a simple mean model. Geometric Brownian motion (GBM) is considered in this section, as it is expected to capture some relationships in the data and outperform the simple mean model. GBM is only used to forecast the return on the equity market three steps ahead.

The parameter estimates required for GBM are determined using the training set of data. The resulting estimates are presented in Table 38

Parameter	Estimate
Drift (μ)	0.0165
Volatility (σ)	0.0631

Table 38: Geometric Brownian motion parameters

Three step forecasts are generated using Equation 45 [39].

$$P(t) = P(0)e^{\left(\mu-\frac{1}{2}\sigma^2\right)t+\sigma B_t}$$

$$\mathrm{c}(t) = \frac{P(t)}{P(0)}$$

$$r(t+2;t+3\,|\,t) = \frac{\mathrm{c}(t+3)}{\mathrm{c}(t+2)} - 1 = \frac{e^{3\left(\mu-\frac{1}{2}\sigma^2\right)+\sigma B_3}}{e^{2\left(\mu-\frac{1}{2}\sigma^2\right)+\sigma B_2}} - 1$$

Equation 45: Three step forecasts for equity market using GBM

where $P(t)$ is the price of the equity market index at time t, B_t is the Brownian Motion random variable at time t, $\mathrm{c}(t)$ is the proportion change in value over period (0:t) and $r(t+2;t+3\,|\,t)$ is the return over period (t+2;t+3) given that the forecasts are generated at time t.

The RMSE of the forecasts over the testing set is 0.0551 which is greater than the RMSE using the simple mean model. This model underperforms expectations and is not considered further in this study. Figure 36 plots the forecast and actual returns on the equity market using the GBM model.

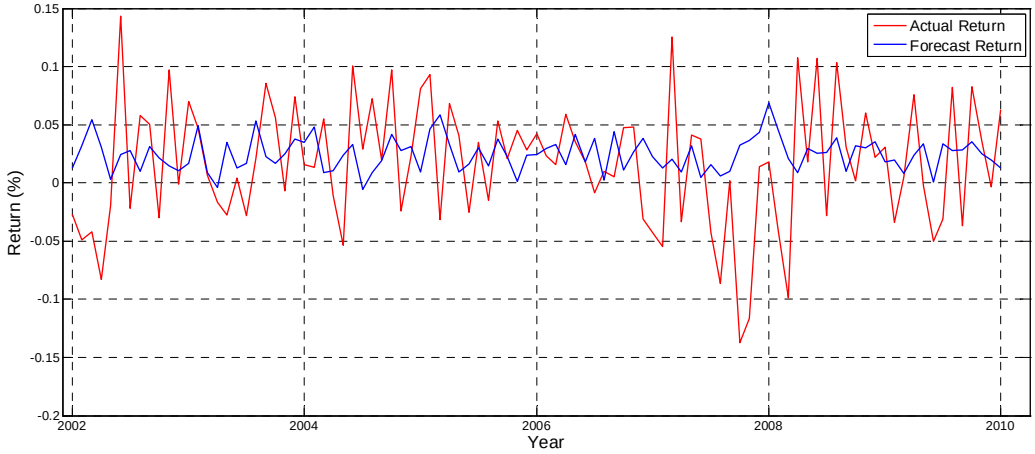

Figure 36: Forecast and actual returns on the equity market from 2002 to 2010 using Geometric Brownian motion – traditional time series forecasting models

3.8 Summary of Traditional Models

Table 39 summarises the traditional models constructed and is used for comparison with ANNs in chapter 6.

Summary of Traditional Models					
		1 Step		***3 Steps***	
Application	**Model Type**	**RMSE (Training)**	**RMSE (Testing)**	**RMSE (Training)**	**RMSE (Testing)**
Inflation	Seasonal Dummies + ARMA(2,1)	0.004922	0.004492	0.005032	0.004921
Money Market	ARI(2,1)	0.000305	0.000116	0.001073	0.000525
Bond Market	AR(1)	0.02566	0.01906	0.02586	0.01925
Equity Market	Mean	0.0651	0.05278	0.06522	0.05278

Table 39: Summary of experiments – traditional time series forecasting models

4 Application of ANNs to Time Series Forecasting

The application presented in this chapter uses Artificial Neural Networks (ANNs) to produce a forecasting system. It aims to forecast inflation and returns on the money, bond and equity markets one and three (and twelve for the money market) months ahead. The forecasting system is first constructed to estimate each market's return in isolation, allowing no interaction between the markets. This is expected to result in a poor explanation of the equity and bond market returns, as a great portion of their variation is a result of external factors (such as the oil price and exchange rate).

Following the construction of these networks, they are combined into a single network. This combination allows interaction between the markets. This is expected to improve the accuracy of the forecasts as more information is used in generating them. All the ANNs are coded in the program Matlab R2011a.

ANNs have been chosen for this application because of their ability to detect and project non-linear relationships between variables. This should lead to an improvement in prediction accuracy over the traditional models constructed in chapter 3. Further, the relationships within and between the markets are expected to change regularly. Machine learning, in the form of ANNs, allow any new trends to be detected and projected as they become relevant. Finally, the markets considered have a large amount of random variation. ANNs are able to deal well with random variation making them ideal for this application.

In chapter 6, the ANNs constructed in this chapter are compared to traditional models. This comparison is performed in order to determine which approach has superior forecasting accuracy.

4.1 Relevant Research Questions

This chapter assists in answering five of the six research questions stated at the beginning of this study (section 1.3.1).

1. How to build and optimise ANN structures for modelling financial data.
2. Can monthly inflation or return in the money, bond or equity markets in South Africa be forecast accurately using a pure time series approach?
3. To what extent does the accuracy associated with forecasts decrease as the forecast period expands?
4. How do ANNs compare to traditional models when forecasting time series?
5. Are the inter-market relationships significant when forecasting inflation and/or the money, bond and equity markets over a one and three-month forecast period?

The definition of a three-month forecast period is as follows. The three-month forecast of inflation or return is the monthly inflation rate or return over the third month, two months from the moment the forecast is generated. It is the forecast of inflation or return from month t+2 to t+3, where t is the month in which the forecast is generated.

4.1.1 Method

ANNs are constructed to generate forecasts of inflation and/or returns by considering a number of past observations of that specific index. This is achieved through the following process.

1. Possible learning rate changes and structures are investigated.
2. The most efficient structure is selected and improved.
3. The forecasts and associated errors are analysed.
4. The constructed ANNs are compared to traditional models and conclusions are drawn (Chapter 6).

4.1.2 Forecast Periods

The experiments are performed using a one and three-month forecast period. For the twelve-month forecast of the money market refer to section 5.5.

4.2 Data Sets

The data used is identical to that discussed in section 3.2. To reiterate, it spans from 1975 to 2010 and is the monthly returns on the money, bond and equity markets, as well as inflation. The characteristics are not given here, as they are discussed in section 3.2.

As in chapter 3, the data is divided into two mutually exclusive data sets.

- Training data set (332 observations), and
- Testing data set (100 observations).

As the ANNs developed in this chapter are compared to the traditional models, it is important to maintain consistency between the data sets used for the construction and testing of the various models. This ensures no model can benefit from having additional information during the construction process.

4.3 Parameters for Experiments

A brief description of the parameters used in the experiments is given in Table 40.

Parameter	Description
Type of Network	A Feed Forward Multi-layered Perceptron Neural Network is used as it is the simplest to construct and is applicable to the task.
Activation Function	A hyperbolic tangent is used as the activation function. This function has y-axis asymptotes of -1 and 1. This is chosen as the outputs could either be positive or negative.
Error Measure	The Mean Square Error (MSE) is used as the error measure. This is done because the learning algorithm is derived from the MSE.
Normalization	The data is normalized by dividing each data point by the largest absolute observation. This ensures the observations are scaled into the interval [-1; 1].
Data Sets	Training Data Set – Sub-data set consisting of the majority of observations. This is used in the training process of the ANN. Testing Data Set – Sub-data set consisting of the remaining observations. This is used to test the system.

Number of initialized structures / Number of times the structure is initialized	The weights and learning rates of the system are initialized randomly. This is the number of times the ANN is initialized and subsequently trained.
Learning Algorithm	Resilient Propagation (RPROP) is used to update the weights and minimize the error of the system.
Learning Increase/Decrease	The speed of learning. High rates increase speed, however, may lead to larger errors. Low rates increase accuracy but increase computing time.
Training Epochs (Epochs)	The number of times the system is trained. High values reduce the error of the ANN over the training set of data, however, lead to over fitting in some cases.
Hidden Layers	A single hidden layer is used because there is a belief that a single hidden layer can be used to solve most practical problems.
Input Neurons / Lagged Observations (lags)	The number of lagged observations used in the forecast process.
Hidden Neurons	The number of neurons used for processing information. These are found in the hidden layers.
Output Neurons	The forecast value. A single output neuron is used in each ANN.

Table 40: Parameters for ANN applications

4.4 Artificial Neural Network Designs

Designs of the ANNs used in the experiments are provided in figures Figure 37, Figure 38 and Figure 39.

4.4.1 Structure of Isolated ANNs

The ANNs constructed use observed values of a specific index to forecast the future value of that index. The indices considered are inflation, return on the money, bond and equity markets. The isolated ANN allows no interaction between the indices (markets in this case). For example, future inflation can only be forecast using observed inflation values.

The diagram in Figure 37 depicts the structure of the ANN used to forecast inflation.

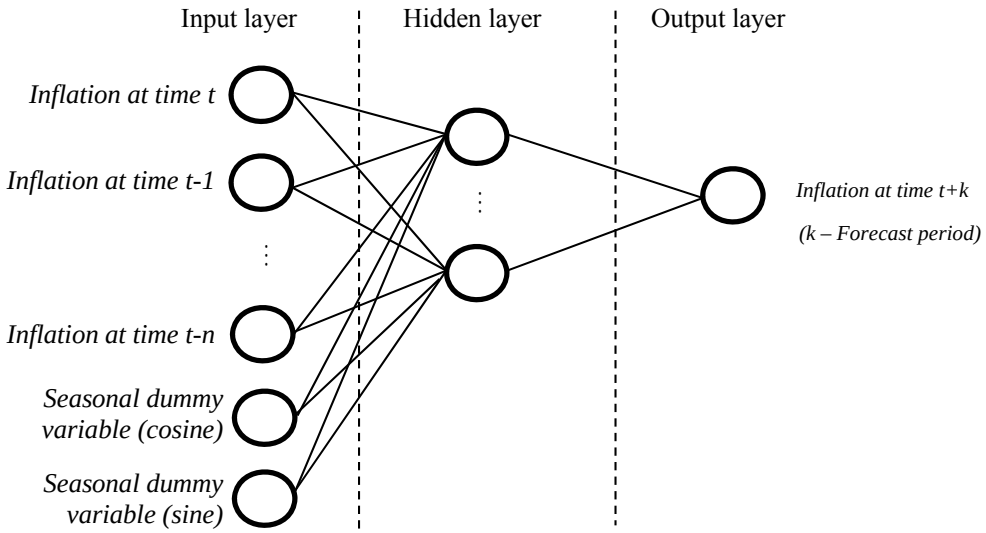

Figure 37: Design of inflation forecasting isolated ANN

Figure 38 is a diagram of the ANN used to forecast the return on each market.

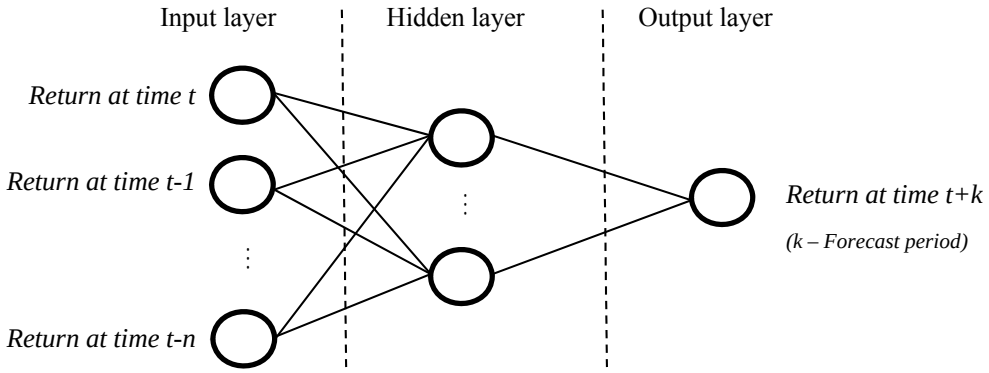

Figure 38: Design of return forecasting isolated ANN

4.4.2 Structure of the Integrated ANN

The integrated ANNs constructed use the observed values of several indices to forecast the future value of a single index. The indices considered are again inflation, return on the money, bond and equity markets. This integrated ANN allows full interaction between the indices (markets in this case). For example, future inflation can be forecast using observed returns from the money, bond and equity markets, as well as inflation.

Figure 39 is a diagram of the integrated ANN used in this experiment.

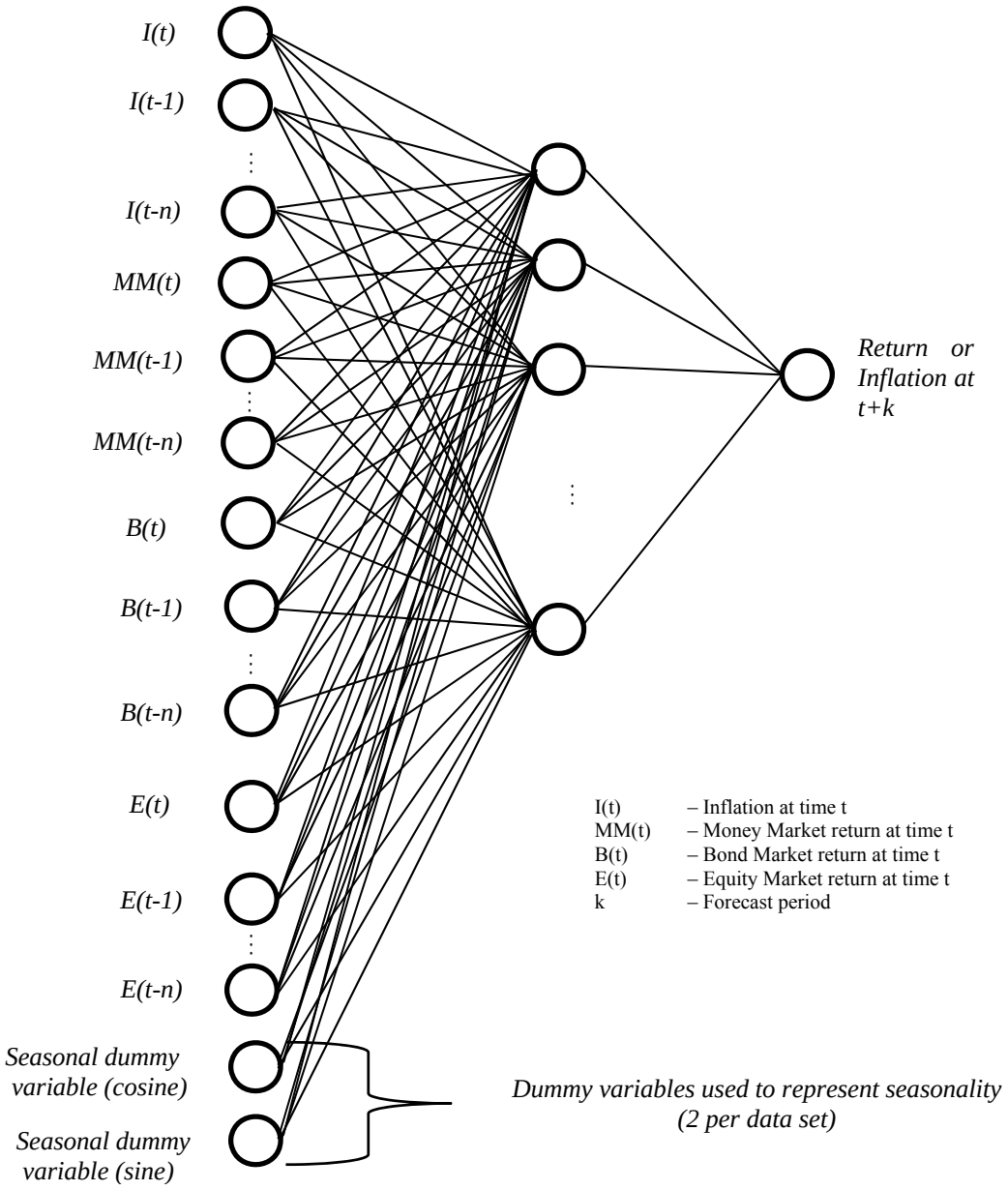

Figure 39: Design of return and inflation forecasting integrated ANN

4.5 Overview of Experiments

A high level description of the experiments is given in this section.

4.5.1 Training and Testing Data Sets

The data set is divided into two independent sets.

- A testing set which consists of the 100 latest observations (September 2002 – December 2010). This set is not used in the training process.
- A training set which has the remaining 332 observations (January 1975 – August 2002).

4.5.2 Experiment 1 – Determining an Efficient Structure

There are no theoretical processes in place to determine the optimal structure for Artificial Neural Networks (ANNs). However, effectiveness of the ANN in each case depends on the efficiency[4] of the structure chosen. An overly simplified structure will result in a reduced ability to learn trends in the data. An overly complex structure will increase computing requirements and result in random variation in the forecasts.

Several parameters need to be specified in order for an ANN to function effectively. There are no theoretical methods of determining the optimal parameter in any of the cases. The aim of experiment 1 is to determine an efficient estimation of the parameters. A brief description of the parameters is given in Table 41.

Parameter	Description
Learning Increase/Decrease	The speed of learning. High rates increase speed, however, may lead to larger errors. Low rates increase accuracy but increase computing time.
Training Epochs (Epochs)	The number of times the system is trained. High values reduce the error of the system over the training set of data, however, may lead to over fitting in some cases.
Input Neurons / Lagged Observations (lags)	The number of lagged observations used in the forecast process.
Hidden Neurons	The number of neurons used for processing information. These are found in the hidden layers.

Table 41: Parameters that need estimation – application of ANNs to time series forecasting

There are several practical problems associated with obtaining efficient estimations of these parameters. The method discussed below results in efficient estimations; however, several simplifying assumptions are made for practical purposes.

[4] *An efficient ANN structure is one that has input parameters (input neurons, hidden neurons, learning increase, learning rate decrease and number of training epochs) which have been determined to be superior to other estimates through an optimisation process, as described here.*

4.5.3 Method – Experiment 1

ANNs are constructed to forecast inflation or return one and three months ahead, using a certain number of past observations. In each case various ANNs are used to estimate the required parameters before an efficient ANN structure is specified. The parameter estimations are done over several experiments. Each experiment is explained below.

4.5.4 Part 1 – Determining an Efficient Number of Epochs

The aim of this experiment is to determine the maximum number of training epochs required for all the systems to reach their respective minimums. To determine the number of epochs, the most complex system with the slowest learning rates is considered and trained. The point where the Root Mean Square Error (RMSE) begins to increase over the testing set of data indicates the system has reached a minimum error and begins to over fit. This point is determined for each application and is necessary for a consistent comparison between different ANN structures.

4.5.5 Part 2 – Determining Efficient Parameters

In this section an efficient learning rate increase and decrease is determined. Thereafter, an efficient number of input and hidden neurons are estimated.

4.5.5.1 Process – Parameter Estimation

Different combinations of the parameters are considered and tested. The number of combinations considered is limited due to time constraints. The limiting of these combinations is justified, as this study proposes a possible method of parameter estimation. Increasing the number of combinations will increase the training times significantly. For example, if four different learning rate changes are considered the number of structures analysed will increase from 324 to 576. Further, as each structure is initialized 5 times, the number of individual ANNs trained will increase from 1 620 to 2 880. Before applying these ANNs in a practical setting, it is advised to thoroughly investigate which structure is optimal by testing more combinations than done here.

The value of each parameter considered and tested is given in Table 42.

Parameters	
Input Neurons	1,3,6,12,24,36
Hidden Neurons	1,4,8,16,32,64
Learning rate increase	1.005, 1.01 1.1
Learning rate decrease	0.8, 0.5, 0.2

Table 42: Parameter combinations considered – application of ANNs to time series forecasting

For each parameter combination, the Mean Squared Error (MSE) over the training process is determined for both the training and testing sets. This results in the consideration and testing of 324 different structures. Further, each structure is considered over several initialization cycles.

The efficient ANN is the ANN with the lowest MSE over the training and testing data sets, and is analysed in detail. However, the minimum MSE over the training and testing data sets is generally achieved with different structures. A decision is made in each case to choose a structure which minimises the increase in error over both of the data sets

with an increased weighting attached to the error over the testing set. This decision has subjective components which are noted in each case.

Only two of the parameters are analysed in each experiment. The learning rate changes are considered first, followed by the number of input and hidden neurons.

4.5.5.2 Learning Rate Increase/Decrease

To estimate efficient learning rate changes the average MSE over different combinations of the input and hidden neurons is considered. This is performed for every learning rate increase and decrease combination (9 combinations). These errors are tabulated and the trends analysed.

It is important to note that the average MSE does not explicitly allow for outliers in the data and may be skewed. From the investigations no outliers are found, but extensive tests to verify this remark are not performed. Further investigation into the possibility and effect of outliers must be done.

4.5.5.3 Input and Hidden Neurons

To determine an efficient combination of input and hidden neurons the MSE for every combination of input and hidden neurons is calculated over both the training and testing sets. This is done using the learning rate increase and decrease chosen in the previous section (section 4.5.5.2). The MSE surface over the training and testing data sets is plotted and analysed. The minimum MSE over the data sets is generally achieved using different ANN structures. In each case an ANN structure that minimises the increase in error over both data sets is chosen, with an increased weighting attached to the MSE over the testing set. This decision has subjective components which are noted in each case.

The selection of each parameter above considers the average MSE over several ANN initialization cycles.

4.5.6 Experiment 2: Analysing the Efficient Model

Experiment 1, section 4.5.2 to section 4.5.5, results in an efficient ANN structure. Experiment 2 analyses the efficient ANN in detail. The ANN is trained over an increased number of epochs which determines the minimum MSE over the testing and training sets of data. The error over the training set is considered, however, is of less importance due to possible over fitting. Analysis is done and conclusions drawn from the results. The final efficient ANN has the lowest summed MSE over the testing set. This ensures the error over the testing set is stable and over fitting is avoided. The ANNs are compared to traditional models in chapter 6.

4.5.7 Assumptions

The following assumptions are made in this experiment.

4.5.7.1 Input Neurons

Autocorrelation plots suggest observations pre-dating a 3-year period will have a negligible influence on the forecast. A maximum of 36 input neurons is considered, which corresponds to a 3-year period. The number of input neurons considered and tested increases in an exponential manner. The increase is exponential because as the ANN structure increases in complexity, the addition of a single neuron will make a smaller and smaller difference.

4.5.7.2 Hidden Neurons

It is assumed that a maximum of 64 hidden neurons is sufficient to capture all the relationships in the data. The number of hidden neurons is varied in an exponential manner and relies on the same argument as in the input neurons case.

4.5.7.3 Learning Rate Increase/Decrease

Three different increases are considered, which are 1.005, 1.05 and 1.1 (0.5%, 5% and 10% increase). These are chosen because small learning rates will ensure the minimum error is obtained, however, it will require significantly more training epochs. This is similarly the case for the learning rate decreases, which are 0.2, 0.5 and 0.8 and correspond to an 80%, 50% and 20% decrease, respectively.

4.5.7.4 Weight and Learning Rate Initialization

The weights and learning rates are initialized using a normal distribution with a mean of 0 and a standard deviation of 0.001. This is done as all the weights are expected to be small. Additional research must be conducted to determine the effect of the weight and learning rate initialization on training time and system accuracy.

4.5.8 Seasonal Variables

Two additional variables are used to explain seasonality in the data sets. The only data set containing seasonality is inflation, which contains annual seasonality. The variables used to explain seasonality are determined using the following process.

1. Each month is given a number ranging from 1 to 12 (1 = January, 2 = February…).
2. Each number is divided by 12. This gives the proportion of 12 each period represents.
3. This number is presented to a cosine and sine function to produce two variables used to capture any seasonal trends in the data.
4. These variables are added to the data set as additional explanatory variables.

Only the inflation forecasting is provided with seasonal explanatory variables in the isolated ANN. All the data sets are provided with seasonal explanatory variables when considering the integrated ANN.

4.5.9 Experiment Shortcomings

The above experiment determines an efficient structure through a trial and error basis which may not determine the optimal structure. Ideally, every possible structure should be considered with every variation of learning rate change. This is not done due to time and computing constraints. The structure determined is efficient but likely requires additional computational time over the optimal structure.

Theoretically, there may be a global minimum of the 5 dimensional surface consisting of MSE, input neurons, hidden neurons, learning rate increase and learning rate decrease. This must be investigated further but is not done here as it is not the aim of this study. To read further on shortcomings and suggested future work refer to chapter 7.

4.5.10 Terminology used in Experiments

Several terms are used interchangeably during the experiments. It is important for the reader to keep this in mind when developing an understanding of the results. The following terms are used interchangeably:

- Input neurons, lagged observations, lags and past observations.
- Upper and lower limits are referred to as a confidence interval (or CI), due to the method of construction.
- Random variation, noise and random fluctuations are used interchangeably.
- The monthly return is referred to through the use of the word return, unless otherwise stated.
- Epoch and training epochs.
- Return on markets and/or inflation may be referred to as indices in specific cases.
- With numbers using scientific notation, 10^{-n} is equivalent to 1e-00n or 1E-0n.
- Peaks and valleys on the MSE surface are marked in different colours. The scale below, Figure 40, provides an interpretation of the colours.

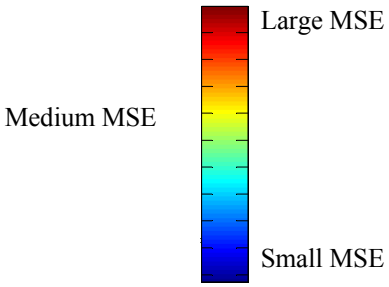

Figure 40: MSE surface scale

4.6 Isolated Inflation ANN – One-Month Forecast

This section aims to construct an ANN that efficiently forecasts inflation a period of one month in advance, and allows no interaction between inflation and the three markets.

4.6.1 Determining an Efficient Structure

An efficient estimate of each parameter is obtained in this experiment.

4.6.1.1 Part 1 – Efficient Number of Training Epochs

The parameters used for this experiment are tabulated below (Table 43).

Parameter	Value
Network structure	36-64-1
Training set size	295
Testing set size	100
Dummy variables (for seasonality)	2
Number of initialized structures	5
Epochs	10 000
Training algorithm	RPROP
Learning rate change:	
Increase	0.5% (1.005)
Decrease	20% (0.8)
Maximum	100
Minimum	0.000000001

Table 43: Parameters for experiment - Part 1 – inflation one step isolated ANN

Observations – Part 1

Figure 41 indicates the minimum Root Mean Squared Error (RMSE) of 0.005 over the testing set is achieved within the first 250 epochs, after which the RMSE begins to increase. The increase indicates the system is over fitting the ANN to the data. The minimum RMSE, of 0.0032, over the training set is achieved at approximately the 10 000 epoch mark. This is expected as over fitting results in a highly accurate representation of the training data by the ANN, but leads to poor generalisation. The poor generalisation ability is reflected through the greater error over the testing set of data.

The presence of over fitting suggests there is random variation in the data. Past observations of inflation cannot explain this random variation. This supports theory, stipulating that inflation is affected by external factors, such as the country's monetary policy.

		Training Epochs				
	Min RMSE	**500**	**1 000**	**10 000**	**Δ in RMSE – 500 to 1 000**	**Δ in RMSE – 1 000 to 10 000**
RMSE – Testing Set	0.0050	0.0050	0.0052	0.0072	-0.0002	-0.002
RMSE – Training Set	0.0032	0.0050	0.0048	0.0032	0.0002	0.0016

Table 44: RMSE after different numbers of epochs – part 1 – inflation one step isolated ANN

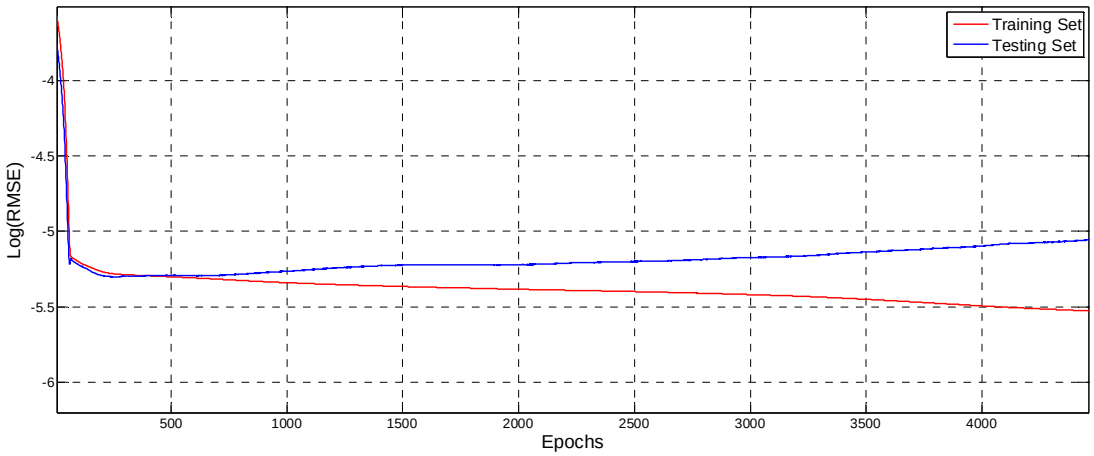

Figure 41: Natural Logarithm of RMSE over the training process for training and testing data sets – Part 1 – inflation one step isolated ANN

Conclusion – Part 1

The number of epochs used to compare different ANN structures is 1 000. Over fitting occurs within 1 000 epochs as indicated by the previous experiment. Over fitting in this case is desired as it ensures the minimum error over the testing data set is surpassed and captured. This decision allows consistent comparison of the different ANN structures. Over the testing set a minimum RMSE of 0.005 is achieved, which is higher than that over the training data set of 0.0032. The minimum RMSE over the testing set is obtained within the first 1 000 epochs. This inference is supported by Figure 41 through the trough created by the natural logarithm of the RMSE over the testing data set at approximately 250 epochs.

The ANN considered here is the most complex and has the slowest learning parameters. Simpler ANNs are expected to reach their minimum RMSE within the 1 000 epochs as their learning is faster. The decision of an efficient number of learning epochs is primarily based on the results obtained over the testing data set. Purely considering the error over the training set will cause the maximum number of epochs to be selected, as over fitting in complex ANNs is common. This reasoning holds for the following three applications (section 4.7 – section 4.9) and is not repeated in each instance.

4.6.1.2 Part 2 – Determining Efficient Parameters

Observations – Learning Rate Increase/Decrease

The learning rate increase of 1.05 (5% increase) and decrease of 0.8 (20% decrease) results in the minimum Mean Squared Error (MSE) over the training data set. This is seen from Table 45. The increase and decrease of 1.005 (0.5% increase) and 0.8 (20% decrease), respectively, results in the minimum MSE over the testing data set. The minimum MSE over the training and testing sets of 2.60E-05 and 2.56E-05 respectively are marked in green on Table 45. The MSEs associated with the efficient learning rate change combination are marked in orange.

The efficient combination of learning rate changes achieves the minimum MSE over the testing data set and 2.63E-05 over the training set. This leads to an increase of 0.03E-05 and 0 in MSE over the training and testing sets, from their respective minima.

Training Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	2.76E-05	2.83E-05	2.90E-05
	0.5	2.72E-05	2.77E-05	2.87E-05
	0.8	**2.63E-05**	2.60E-05	2.70E-05
	min	2.60E-05		
Testing Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	2.61E-05	2.72E-05	2.81E-05
	0.5	2.61E-05	2.66E-05	2.75E-05
	0.8	**2.56E-05**	2.59E-05	2.65E-05
	min	2.56E-05		
	Minimum			
	Efficient			

Table 45: MSE for learning rate changes – inflation one step isolated ANN

Conclusion – Learning Rate Increase/Decrease

The following decisions are made:

- The learning rate decrease of 0.8 is to be used, i.e. a 20% decrease in learning rate if the error of the system increases during training.
- The learning rate increase of 1.005 is to be used, i.e. a 0.5% increase in learning rate if the error of the system decreases during training.

Input and Hidden Neurons – Training Set's MSE Surface

The MSE surface with upper and lower limits, Figure 42, indicates a general decrease as the complexity of the system rises. This is expected since the increasing complexity of the system will raise the likelihood of over fitting in the ANN, causing the error over the training set to decrease. Further, the upper and lower limits become volatile as the complexity of the system grows. This phenomenon is unexpected and is likely due to random fluctuations from the ANN initialization stage. Investigations to confirm this reasoning must be done.

Considering the best estimate MSE surface, Figure 43, the MSE of the system decreases as the complexity of the system increases. The minimum of 2.07E-05 is obtained using 36 input and 32 hidden neurons. This general trend is expected; however, the minimum MSE is not obtained using the most complex structure (36 input and 64 hidden neurons). This is likely due to fluctuation caused by the randomly initialized structures.

The upper and lower surfaces represent the upper and lower limits of the MSE over the training set of data. These are determined by increasing/decreasing the best estimate surface by two standard deviations of the MSE. The upper and lower limits of the MSE surface over the data set require assumptions in order for statistical significance to hold. There is insufficient information available to make these assumptions. These bounds, however, provide a good estimate of the limits for the MSE. The construction process of the upper and lower surfaces, described here, is performed for both the training and testing sets. Further, it is repeated for all the ANN applications considered in this chapter. However, in the interest of space the above reasoning is not repeated for each application.

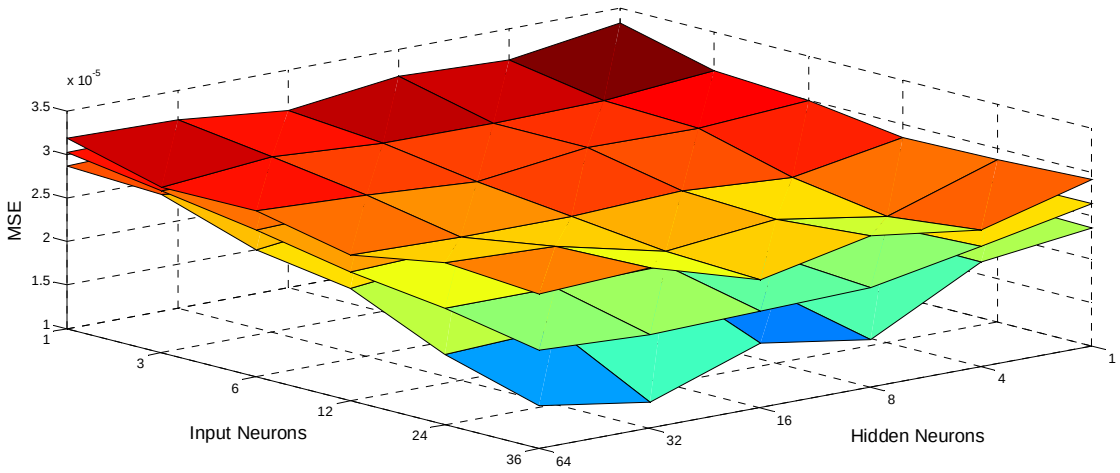

Figure 42: MSE surface with upper and lower limits – training data set – inflation one step isolated ANN

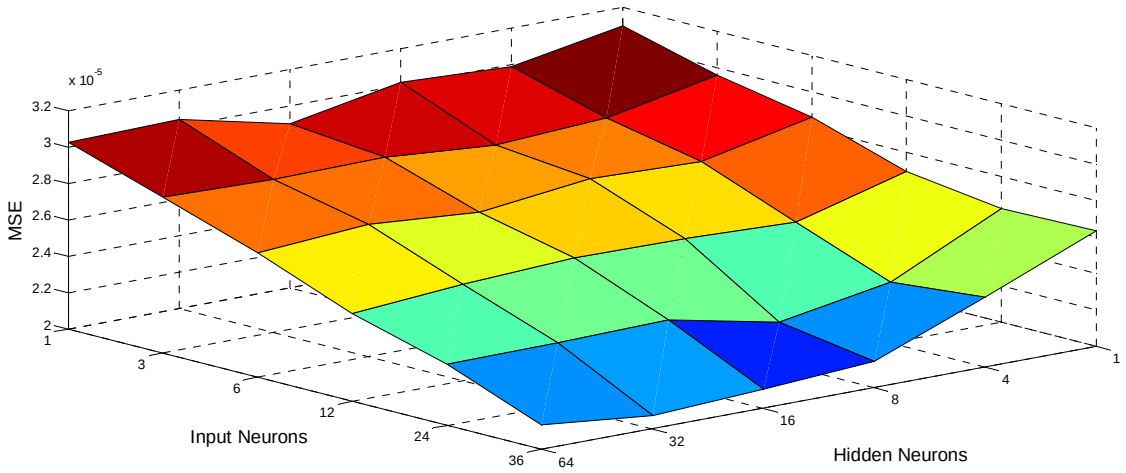

Figure 43: Best estimate MSE surface – training data set – inflation one step isolated ANN

Input and Hidden Neurons – Testing Set's MSE Surface

The MSE surface with upper and lower limits, Figure 44, is constructed using the testing set of data. The surface indicates a rough surface with no visible trends. The best estimate MSE surface, Figure 45, suggests a general decrease in MSE as the number of input neurons increase. However, signs of over fitting are present when more than 24 input neurons are considered. As the number of hidden neurons increases the MSE decreases. When considering 24 input and 32 hidden neurons the minimum MSE, of 2.25E-05, is achieved.

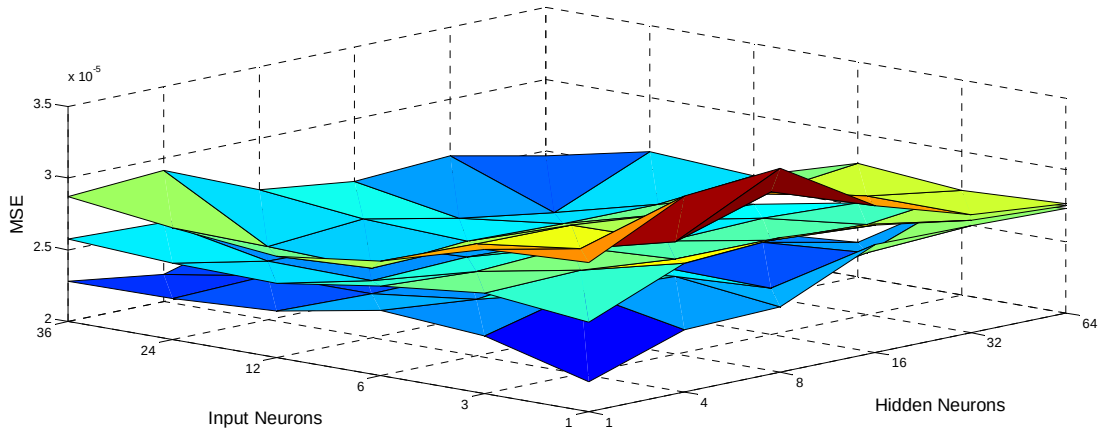

Figure 44: MSE surface with upper and lower limits – testing data set – inflation one step isolated ANN

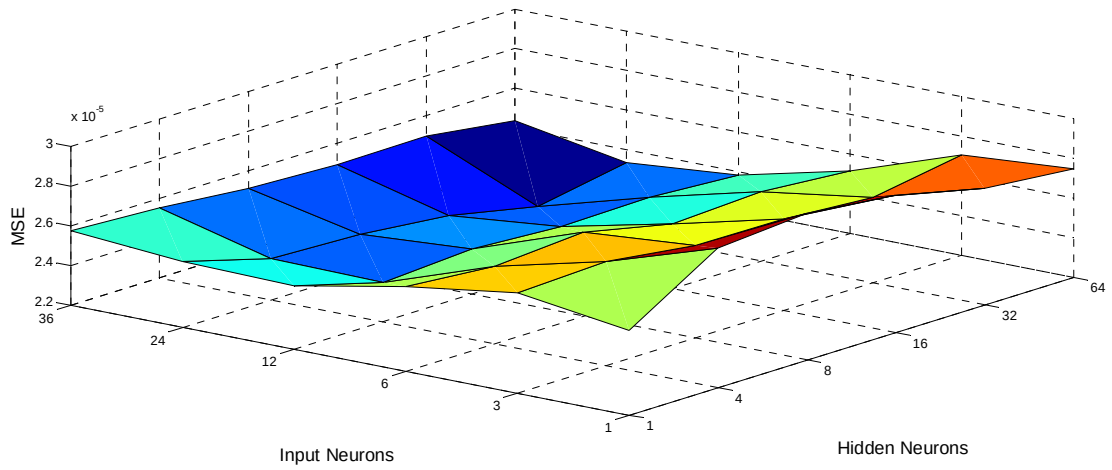

Figure 45: Best estimate MSE surface – testing data set – inflation one step isolated ANN

Conclusion – Input and Hidden Neurons

A parsimonious ANN structure is chosen which balances the errors over both data sets. This structure consists of 24 input and 32 hidden neurons. The MSE obtains its minimum over the testing set and increases by 0.27E-05 over the training set.

The structure chosen aims to minimize the MSE over both data sets. However, compensation is needed as the minimum MSE does not occur using the same ANN structure over both data sets. In this decision the results from the testing set have a heavier weighting than those from the training set, which ensures the limited effect of over fitting.

The choice made has subjective components. It is important not to base the decision purely on the results obtained from the training set as this will lead to over fitting. Similarly, basing the decision purely on the results over the testing set will cause the ANN to only fit the testing set. Both the scenarios described above will reduce the generalization ability of the ANN, leading to poor forecasting accuracy.

Conclusion of Determining an Efficient Structure

Efficient parameter estimates are:

- 24 input neurons,
- 2 seasonal input neurons,
- 32 hidden neurons,
- A learning rate increase of 1.005 (0.5% increase), and
- A learning rate decrease of 0.8 (20% decrease).

4.6.2 Analysing the Efficient ANN

The ANN is trained using the following parameters (Table 46).

Training set size	307
Testing set size	100
Dummy variables (for seasonality)	2
Input/lagged neurons	24
Hidden neurons	32
Number of initialized structures	10
Epochs	10 000
Training algorithm	RPROP
Learning rate:	
Increase	1.005 (0.5% increase)
Decrease	0.8 (20% decrease)
Maximum	100
Minimum	0.000000001

Table 46: Parameters for the efficient ANN structure – inflation one step forecasting isolated ANN

Observations of System Error

After approximately 3 000 training epochs the ANN begins to learn the randomness in the data, which indicates over fitting. This is supported by the increase in the RMSE over the testing set after 3 000 epochs and is depicted in Figure 46. The minimum RMSE over the testing set, of 0.0047, is obtained after approximately 3 000 training epochs. The errors over the different data sets intersect after approximately 3 500 training epochs, which is used as the efficient number of epochs for this structure. This balances the error over the training and testing sets. Numerical error values at points of interest are provided in Table 47.

Error Readings	Testing Set	Training Set
Minimum RMSE	0.0047	0.0045
Minimum MSE	2.2144e-005	2.0483e-005
Final RMSE after training *(3 500 epochs)*	0.0048	0.0048
Final MSE after training *(3 500 epochs)*	2.2913e-005	2.3392e-005

Table 47: Results of efficient model after training – inflation one step isolated ANN

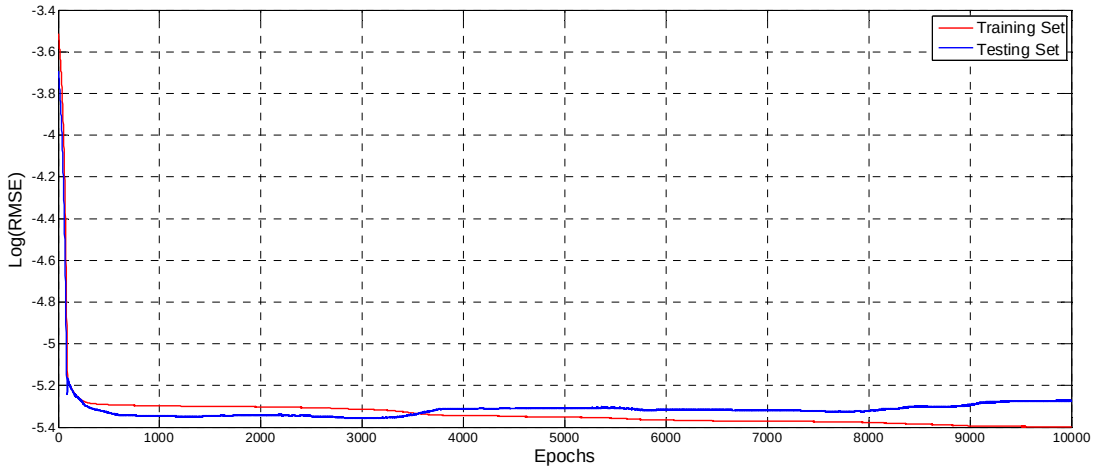

Figure 46: Natural logarithm of RMSE over training process – efficient ANN – inflation one step isolated ANN

Observations of Forecast and Actual Inflation

The ANN captures several underlying trends in the data, as expected. There are several occasions where the ANN either under or overestimates inflation. This is true for both the training and testing data sets. These instances are the result of external factors influencing inflation and cannot be explained using past observations of inflation. The best estimate forecasts from 1977 to 2010 are illustrated in Figure 47 by the blue line. The actual values of inflation are represented by the red line.

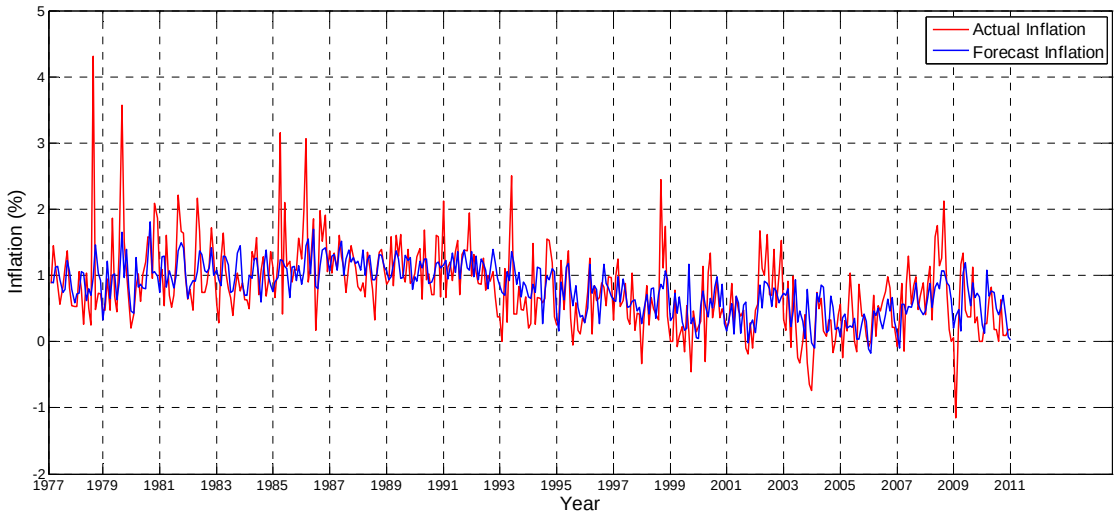

Figure 47: Actual and forecast inflation from 1977 to 2010 – inflation one step isolated ANN

Conclusion - Analysing the Efficient ANN

The ANN captures several trends underlying the data. The efficient ANN structure has an MSE over the training and testing sets of 2.3392e-005 and 2.2913e-005, respectively. This is obtained after 3 500 training epochs, which is viewed as the efficient training period. The ANN performs similarly over both sets of data. The level of effectiveness is determined by comparing these results to traditional models in chapter 6. The minimum RMSE over the testing set, of 0.0047, is obtained after approximately 3 000 epochs. The RMSE over the training set decreases as 3 000 epochs are surpassed and continues to decrease until the maximum number is achieved. This is expected as ANNs have the ability to replicate a data set. This replication is the product of over fitting and reduces the generalization ability of the ANN. To avoid this, and ensure the underlying relationships in the data are captured, decisions relating to efficiency are made with reference to the performance of the ANN over both the testing and training data sets, with heavier weighting given to the results over the testing set.

4.6.3 Summary of Experiment

Table 48 provides a summary of the results from this experiment.

Structure of ANN	24-32-1
Training epoch required for optimal training	3 500
Minimum Root Mean Squared Error:	
Training Data Set	0.0045
Testing Data Set	0.0047
Final Root Mean Squared Error (3 500 epochs):	
Training Data Set	0.0048
Testing Data Set	0.0048
Parameters for RPROP:	
Learning Rate Increase	0.5% (1.005)
Learning Rate Decrease	20% (0.8)
Data Sets:	
Training Set	307 observations (1977 – 2002)
Testing Set	100 observations (2002 – 2010)
Times the structure is initialised	10

Table 48: Summary of experiment's result – inflation one step isolated ANN

4.7 Isolated Money Market ANN – Three-Month Forecast

This section aims to construct an ANN that efficiently forecasts return on the money market three months in advance, and allows no interaction between inflation and the three markets.

4.7.1 Determining an Efficient Structure

An efficient estimate of each parameter is obtained in this experiment.

4.7.1.1 Part 1 – Efficient Number of Training Epochs

The parameters used in this experiment are tabulated below (Table 49).

Parameter	Value
Network Structure	36-64-1
Training set size	293
Testing set size	100
Number of initialized structures	5
Epochs	50 000
Training algorithm	RPROP
Learning rate change:	
Increase	0.5% (1.005)
Decrease	20% (0.8)
Maximum	100
Minimum	0.000000001

Table 49: Parameters for experiment – Part 1 – money market three step isolated ANN

Observations – Part 1

The error of the ANN, over both data sets, minimise after approximately 10 000 epochs. This is supported by Figure 48 which illustrates the natural logarithm of the RMSE over the learning process for both data sets. The minimum RMSE over the training and testing sets are 0.001 and 0.0011, respectively.

Three epoch intervals are specified, and the change of RMSE within these intervals is calculated and analysed in Table 50. The RMSE over both data sets decrease throughout the training process and stabilise at the 2 000 epoch mark. The decrease in RMSE after 2 000 epochs is minimal. There are no signs of over fitting within the ANN, which is unexpected. This anomaly may be explained through consideration of the nature of the money market, which is highly predictable in the short term.

		Training Epochs				
	Min RMSE	**100**	**2 000**	**10 000**	**Δ in RMSE – 100 to 2 000**	**Δ in RMSE – 2 000 to 100 000**
RMSE – Testing Set	0.0011	0.0023	0.0012	0.0012	0.0011	0
RMSE – Training Set	0.0010	0.0028	0.0011	0.0011	0.0017	0

Table 50: RMSE after different numbers of epochs – Part 1 – money market three step isolated ANN

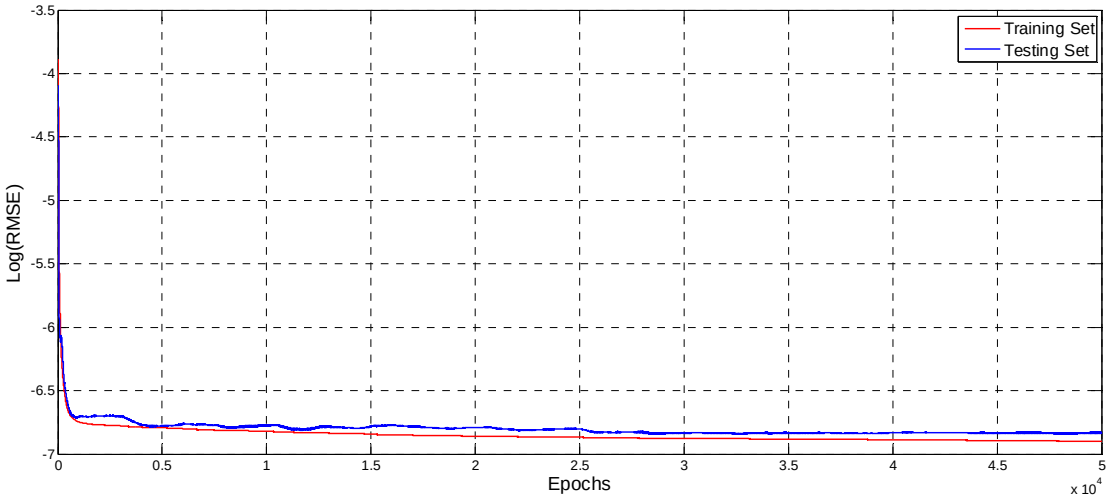

Figure 48: Natural Logarithm of RMSE over the training process for training and testing data set – Part 1 – money market three step isolated ANN

Conclusion – Part 1

The decision to use 2 000 training epochs in the comparison of ANN structures, is made. This point balances accuracy and computing requirements. Ideally a larger number of epochs should be used, however, this will increase computing time considerably and is not feasible in this study.

4.7.1.2 Part 2 – Determining Efficient Parameters

Observations – Learning Rate Increase/Decrease

The learning rate increase of 1.005 (0.5% increase) and decrease of 0.8 (20% decrease) result in the minimum MSE over the training and testing data sets. The MSEs obtained from different combinations of learn rate changes are tabulated below (Table 51). The efficient combination achieves the minimum MSE over the training and testing data sets and is marked in green.

Training Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	2.38E-06	3.60E-06	4.44E-06
	0.5	2.10E-06	3.06E-06	4.10E-06
	0.8	**1.88E-06**	2.76E-06	3.72E-06
	min	1.88E-06		
Testing Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	1.52E-06	2.00E-06	2.28E-06
	0.5	1.35E-06	1.90E-06	2.10E-06
	0.8	**1.22E-06**	1.76E-06	1.96E-06
	min	1.22E-06		
	Minimum			
	Efficient			

Table 51: MSE for learning rate changes – money market three step isolated ANN

Conclusion – Learning Rate Increase/Decrease

The following decisions are made:

- The learning rate decrease of 0.8 is to be used, i.e. a 20% decrease in learning rate if the error of the system increases during training.
- The learning rate increase of 1.005 is to be used, i.e. a 0.5% increase in learning rate if the error of the system decreases during training.

Input and Hidden Neurons – Training Set's MSE Surface

Considering four or more hidden neurons, the MSE is relatively stable for any combination of input neurons over the training set of data. This implies that the MSE is insensitive to different combinations of input neurons when four or more hidden neurons are considered. This is unexpected, as the MSE should decrease significantly over the training set as the ANN increases in complexity. When only a single hidden neuron is considered, the MSE is significantly greater than in other cases. These inferences are supported by the flattening and peaks of the MSE surface with upper and lower limits, in Figure 49.

The best estimate MSE surface, Figure 50, suggests a relationship within the hidden neurons. As the number of hidden neurons increases the MSE decreases for different numbers of input neurons. Further, a minimum valley emerges when 6 input neurons are considered. This valley minimises at 64 hidden neurons.

The minimum MSE, of 1.33E-06, is achieved using 36 input and 64 hidden neurons. The maximum, of 4.94E-06, is achieved using 12 input and a single hidden neuron. This maximum point is unexpected, as it should be obtained when a single input and hidden neuron are considered.

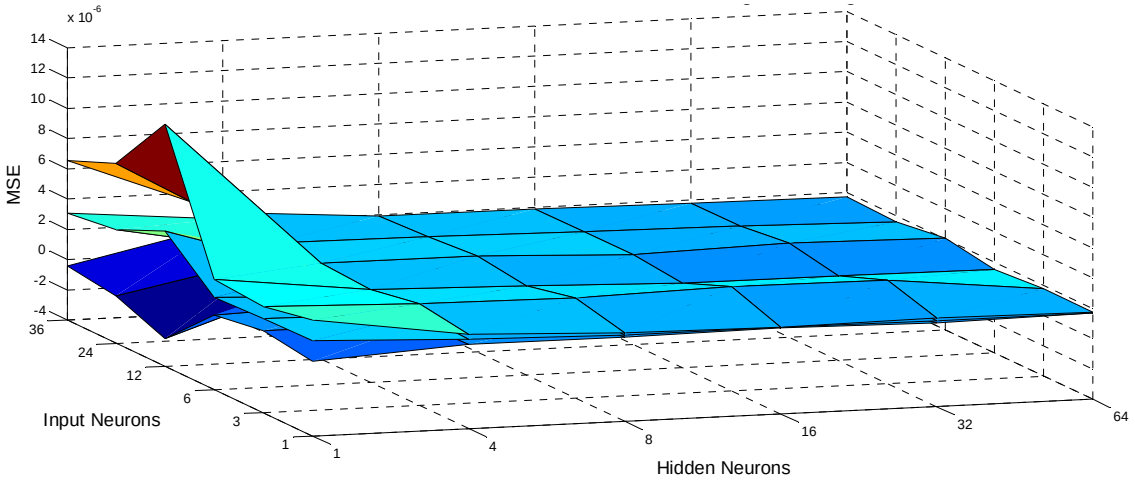

Figure 49: MSE surface with upper and lower limits – training data set – money market three step isolated ANN

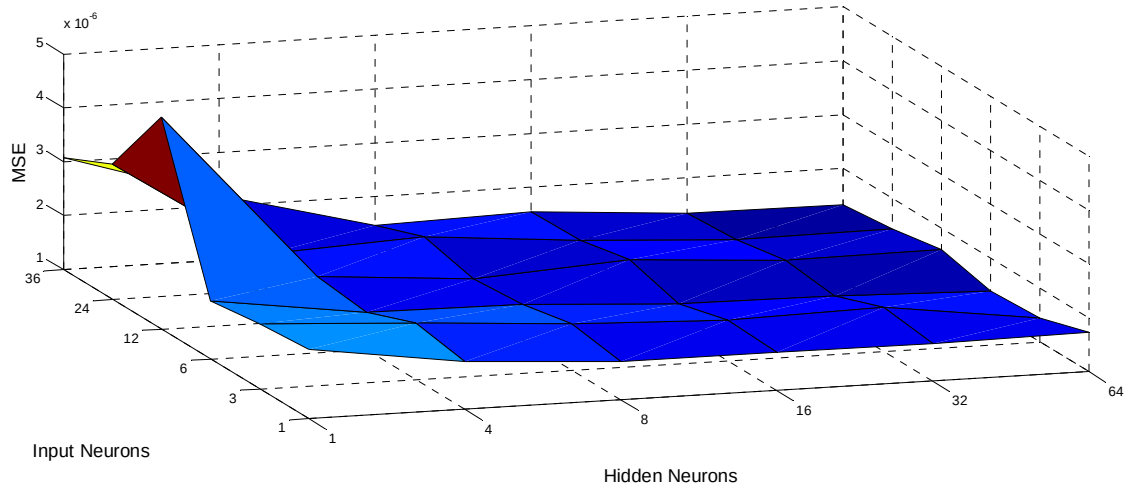

Figure 50: Best estimate MSE surface – training data set – money market three step isolated ANN

Input and Hidden Neurons – Testing Set's MSE Surface

The MSE surface with upper and lower limits over the testing set, Figure 51, indicates a rough surface with a general decrease as the number of input neurons decreases and hidden neurons increases. Considering less than 6 input neurons and more than 4 hidden neurons leads to a stabilisation of the MSE surface. This inference is supported by the best estimate MSE surface, Figure 52. The minimum, of 5.60E-07, is obtained using 3 input and 64 hidden neurons.

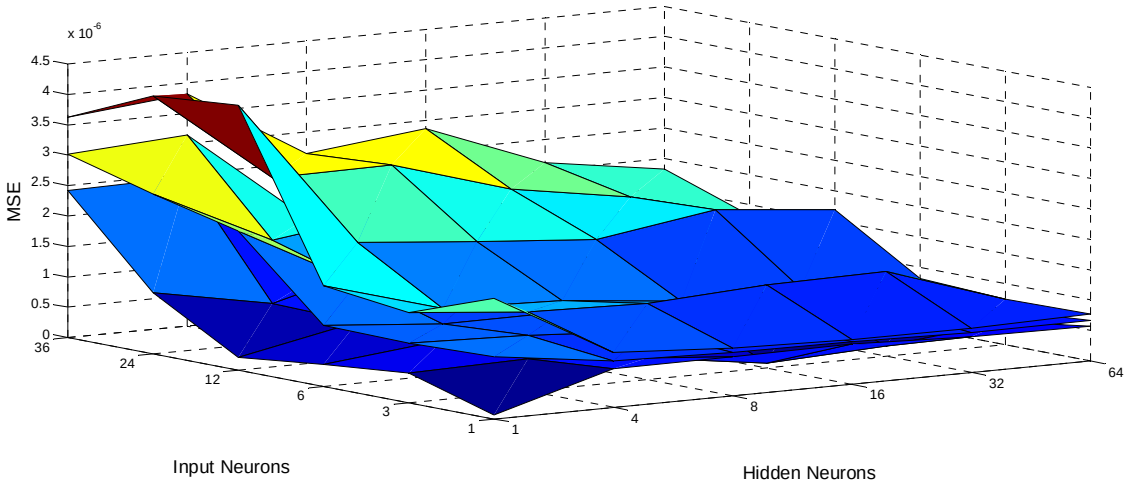

Figure 51: MSE surface with upper and lower limits – testing data set – money market three step isolated ANN

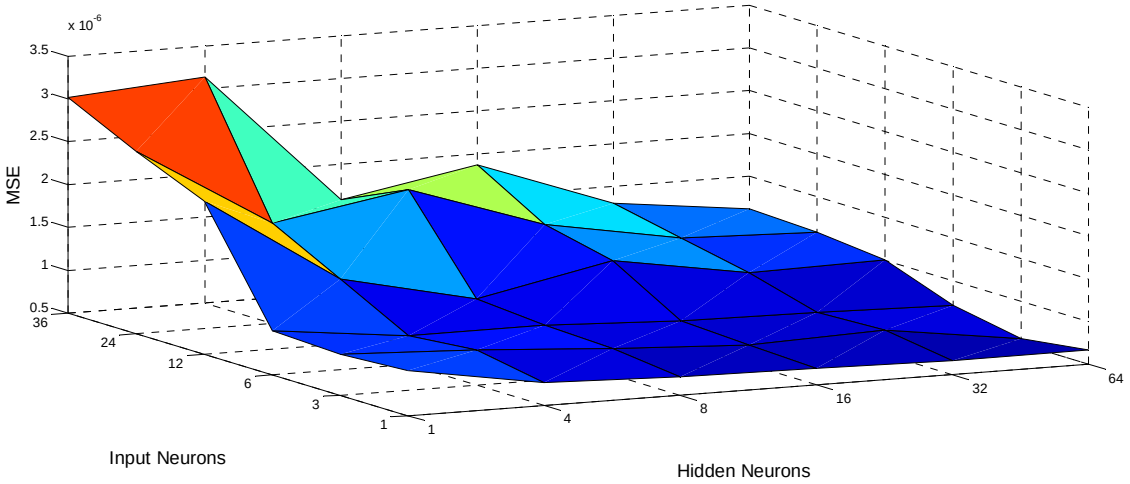

Figure 52: Best estimate MSE surface – testing data set – money market three step isolated ANN

Conclusion – Input and Hidden Neurons

The efficient ANN structure consists of 6 input and 64 hidden neurons. This combination results in a small and stable MSE over the best estimate MSE surface for the training and testing sets, and is supported by Figure 50 and Figure 52. Further, parsimony of the model and accuracy over both data sets are balanced in this decision.

The choice made has subjective components. It is important not to base the decision purely on the results obtained from the training set as this will lead to over fitting. Similarly, basing the decision purely on the results over the testing set will cause the ANN to only fit the testing set. Both the scenarios described above will reduce the generalization ability of the ANN, leading to poor forecasting accuracy.

The results from this experiment suggest there is little randomness in the data due to the limited over fitting present. This observed characteristic is noted.

Conclusion of Determining an Efficient Structure

Efficient parameter estimates are:

- 6 input neurons,
- 64 hidden neurons,
- The learning rate increase of 1.005 (0.5% increase), and
- The learning rate decrease of 0.8 (20% decrease).

4.7.2 Analysing the Efficient ANN

The ANN is trained using the following parameters (Table 52).

Training set size	323
Testing set size	100
Input/lagged neurons	6
Hidden neurons	64
Number of initialized structures	10
Epochs	50 000
Training algorithm	RPROP
Learning rate:	
Increase	1.005 (0.5% increase)
Decrease	0.8 (20% decrease)
Maximum	100
Minimum	0.000000001

Table 52: Parameters for the efficient ANN structure – efficient ANN – money market three step isolated ANN

Observations of System Error

The RMSE over the testing set begins large and decreases substantially thereafter as indicated by the drop in RMSE over the first 1 000 training epochs in the natural logarithm of the RMSE over the training process, Figure 53. After the initial decrease the RMSE over the testing set begins to oscillate as training continues. There is no increase in RMSE during this oscillation period. Over fitting is not prevalent in the ANN, as seen from the stable error over the testing data set.

The minimum RMSE of 6.4518e-004 over the testing set is achieved after approximately 40 000 epochs. The RMSE over the training set decreases over the training process and reaches a minimum after approximately 50 000 epochs.

The efficient number of epochs chosen to train the system is 49 999, as the oscillation of the RMSE over the testing set is at its minimum. Further, the RMSE over the training set is near its minimum at this point. This number of epochs balances both the RMSE over the training and testing sets with computing power. The numerical error values at points of interest are provided in Table 53.

Error Readings	Testing Set	Training Set
Minimum RMSE	6.4518e-004	0.0011
Minimum MSE	4.1625e-007	1.1676e-006
Final RMSE after training *(49 999 epochs)*	6.4518e-004	0.0011
Final MSE after training *(49 999 epochs)*	4.1625e-007	1.1676e-006

Table 53: Results of efficient model after training – money market three step isolated ANN

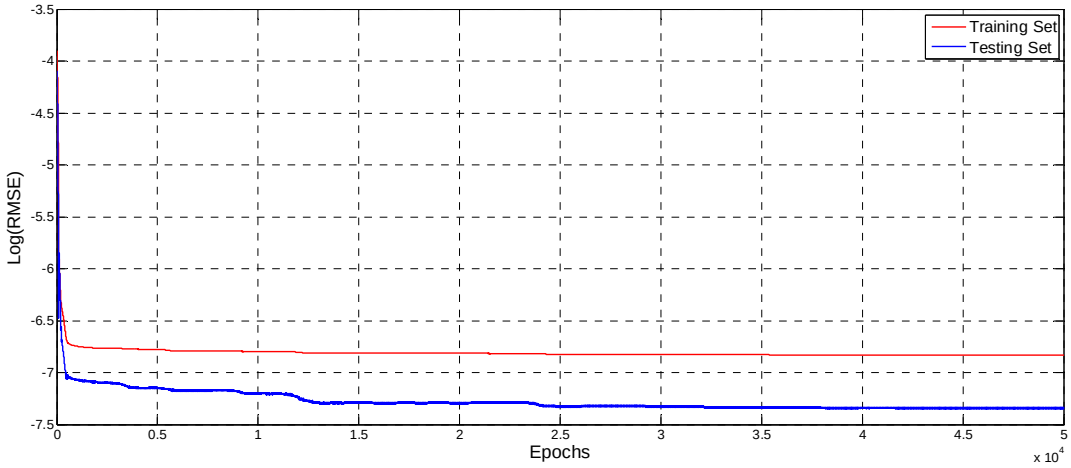

Figure 53: Natural logarithm of RMSE over the training process – money market three step isolated ANN

Observations of Forecast and Actual Returns

The ANN captures several underlying trends in the data. The actual returns from the money market, represented in Figure 54 through the red line, indicate the increased predictability of the money market. This explains the significantly lower error values than in the other applications.

There are certain periods where the system over or underestimates the actual returns. This is explained by the effect of external influences on the money market, such as South Africa's monetary policy. The best estimate forecasts are represented through the blue line in Figure 54.

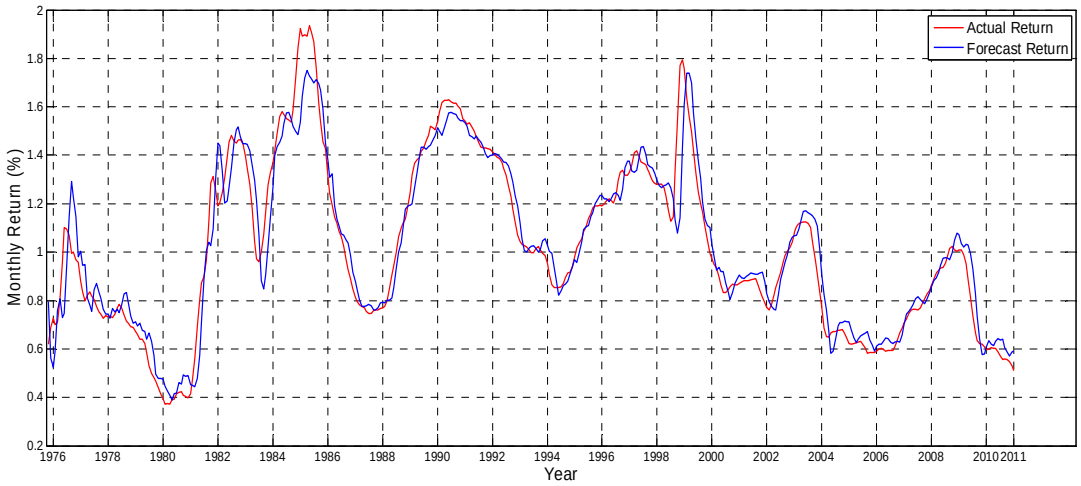

Figure 54: Actual and forecast return on the money market from 1975 to 2010 – money market three step isolated ANN

Conclusion - Analysing the Efficient ANN

The ANN captures several trends underlying the money market. The efficient ANN structure has an MSE over the training and testing sets of 1.1676e-006 and 4.1625e-007, respectively. The ANN performs similarly over both data sets. The level of effectiveness is determined by comparing these results to traditional models in chapter 6. The minimum RMSE of 6.4518e-004 over the testing set and of 0.0011 over the training set are obtained after approximately the maximum number of epochs.

The RMSEs over the training and testing sets continue to decrease as the training process approaches its maximum, of 50 000 epochs. This is unexpected for the testing set, as it indicates no over fitting is present. Theoretically, this can only occur if little random variation is present in the data and the underlying trends remain stable.

4.7.3 Summary of Experiment

Table 54 provides a summary of the results from this experiment.

Structure of ANN	6-64-1
Training epoch required for optimal training	49 999
Minimum Root Mean Squared Error:	
Training Data Set	0.0011
Testing Data Set	6.4518e-004
Final Root Mean Squared Error (49 999 epochs):	
Training Data Set	0.0011
Testing Data Set	6.4518e-004
Parameters for RPROP:	
Learning Rate Increase	0.5% (1.005)
Learning Rate Decrease	20% (0.8)
Data Sets:	
Training Set	323 observations (1975 – 2002)
Testing Set	100 observations (2002 – 2010)
Times the structure is initialised	10

Table 54: Summary of experiment's result – money market three step isolated ANN

4.8 Integrated Bond Market ANN – One-Month Forecast

This section aims to construct an ANN that efficiently forecasts return on the bond market one month in advance, and allows full interaction between inflation and the three markets.

4.8.1 Determining an Efficient Structure

An efficient estimate of each parameter is obtained in this experiment.

4.8.1.1 Part 1 – Efficient Number of Training Epochs

The parameters used for this experiment are tabulated below (Table 55).

Parameter	Value
Network Structure (input neurons per data set × number of data sets) – hidden neurons – output neurons	144 (36x4)-64-1
Training set size	295
Testing set size	100
Dummy variables (for seasonality) (variables per data set x number of data sets)	8 (2x4)
Number of initialized structures	5
Epochs	1 000
Training algorithm	RPROP
Learning rate change:	
Increase	0.5% (1.005)
Decrease	20% (0.8)
Maximum	100
Minimum	0.000000001

Table 55: Parameters for experiment – Part 1 – bond market one step integrated ANN

Observations – Part 1

The plot of the natural logarithm of the RMSE over the training process, Figure 55, indicates that after approximately 330 epochs the RMSE over the testing set achieves its minimum of 0.022. An increase in the error occurs thereafter, which indicates the presence of over fitting in the ANN. The minimum RMSE over the training set, of 0.0026, is achieved at the 1 000 epoch mark, as expected.

To confirm the minimum RMSE over the testing set is achieved within 1 000 epochs, the RMSE is calculated at three points which are tabulated below (Table 56). The increase in RMSE over the testing set, from 500 to 1 000 epochs, confirms the minimum RMSE has been surpassed and captured.

		Training Epochs				
	Min RMSE	**100**	**500**	**1 000**	**Δ in RMSE – 100 to 500**	**Δ in RMSE – 500 to 1 000**
RMSE – Testing Set	0.0220	0.1371	0.0277	0.0330	0.1094	-0.0053
RMSE – Training Set	0.0026	0.1353	0.0086	0.0026	0.1267	0.006

Table 56: RMSE after different numbers of epochs – Part 1 – bond market one step forecasting integrated ANN

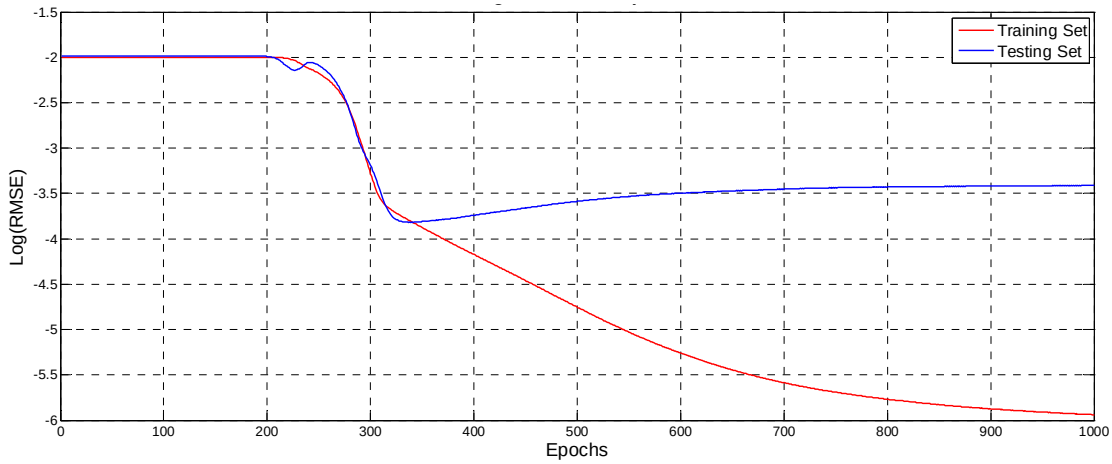

Figure 55: Natural logarithm of RMSE over the training process for training and testing data set – Part 1 – bond market one step integrated ANN

Conclusion – Part 1

Different ANN structures are compared using 1 000 epochs. This number of epochs results in over fitting. It is important for the ANN to reach an over fitted stage, as it indicates the minimum error over the testing set has been surpassed and captured. The increase in RMSE over the testing set, after 400 epochs, indicates the presence of over fitting which is supported by Figure 55. These observations indicate that the minimum RMSE over the testing set is surpassed and captured within 1 000 epochs.

4.8.1.2 Part 2 – Determining Efficient Parameters

Observations – Learning Rate Increase/Decrease

The learning rate increase of 1.05 (5% increase) and decrease of 0.8 (20% decrease) result in the minimum MSE of 3.57E-04 over the training set. The minimum MSE of 3.69E-04 over the testing set is achieved using the learning rate increase of 1.005 (0.5% increase) and decrease of 0.5 (50% decrease). These observations indicate different efficient learning rate increase and decrease combinations, for the different data sets. The observations are given in Table 57, with the minimum MSEs marked in green.

The shape of the MSE surface over the different learning rate changes is considered for both data sets. The range of the surface of 2.23E-04 over the training set is significant in comparison to 0.19E-04 over the testing set.

The efficient learning rate changes are determined by minimising the decrease in accuracy over both sets of data. The decision to use an increase of 0.5% and a decrease of 20% is made. This results in an increase in MSE over the training set, of 0.05E-04, and testing set, of 0.02E-04 from their respective minimums. The MSE associated with the chosen combination is marked in orange in Table 57.

Training Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	5.80E-04	4.83E-04	4.97E-04
	0.5	4.86E-04	4.27E-04	4.36E-04
	0.8	4.02E-04	3.57E-04	3.89E-04
	min	3.57E-04		
Testing Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	3.74E-04	3.77E-04	3.88E-04
	0.5	3.69E-04	3.75E-04	3.81E-04
	0.8	3.71E-04	3.85E-04	3.86E-04
	min	3.69E-04		
	Minimum			
	Efficient			

Table 57: MSE for learning rate changes – bond market one step integrated ANN

Conclusion – Learning Rate Increase/Decrease

The following decisions are made:

- The learning rate decrease of 0.8 is to be used, i.e. a 20% decrease in learning rate if the error of the system increases during training.
- The learning rate increase of 1.005 is to be used, i.e. a 0.5% increase in learning rate if the error of the system decreases during training.

Input and Hidden Neurons – Training Set's MSE Surface

The MSE surface with upper and lower limits over the training set, Figure 56, indicates a decrease in MSE as the ANN structure increases in complexity. As expected, this is due to the presence of over fitting in complex ANN structures. The best estimate MSE surface, Figure 57, indicates a similar trend as in Figure 56. The minimum MSE of 3.55E-06 is achieved using the most complex ANN structure, 36 input and 64 hidden neurons.

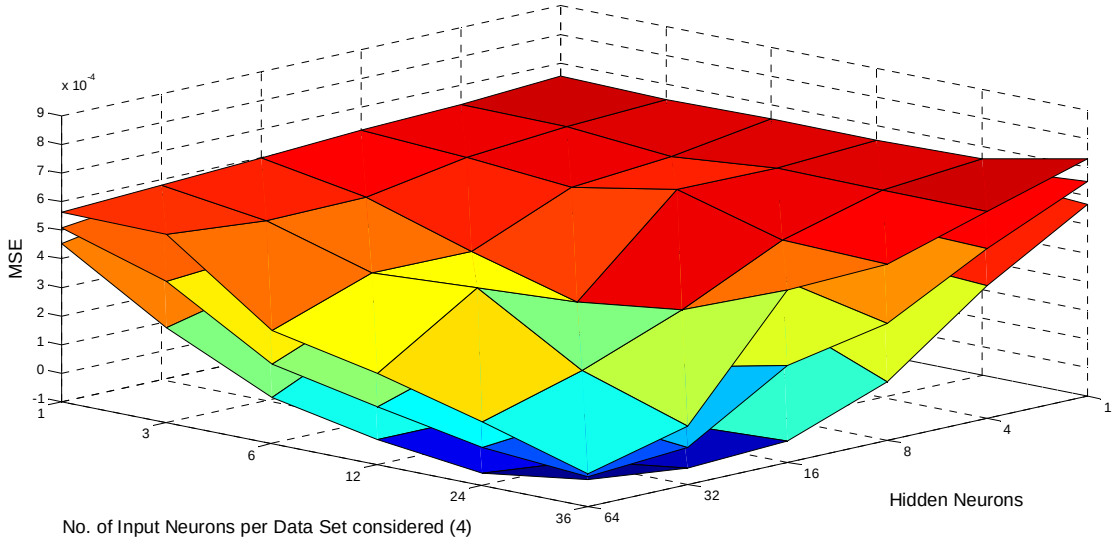

Figure 56: MSE surface with upper and lower limits – training data set – bond market one step integrated ANN

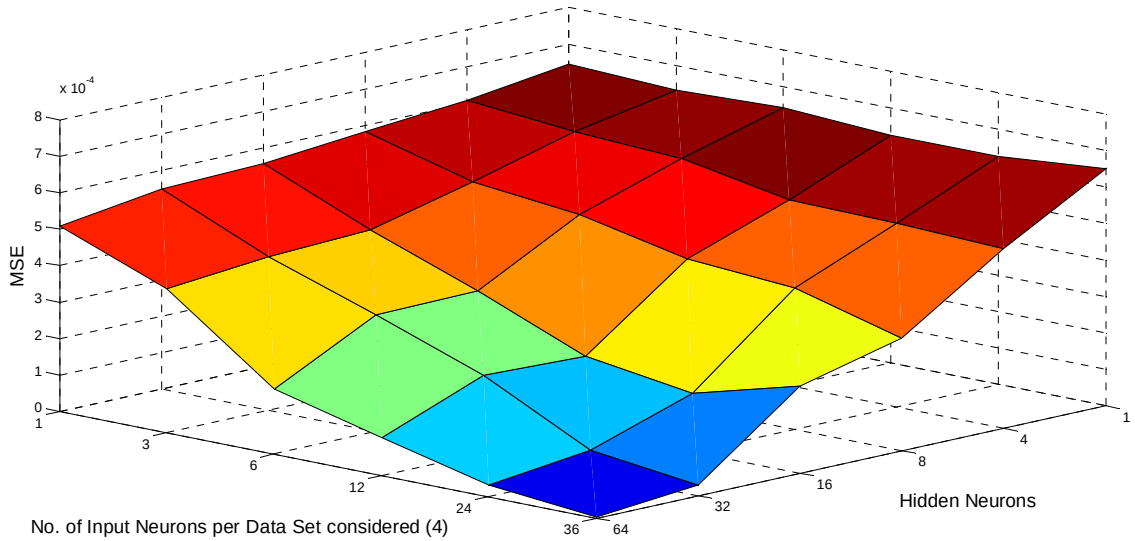

Figure 57: Best estimate MSE surface – training data set – bond market one step integrated ANN

Input and Hidden Neurons – Testing Set's MSE Surface

The MSE surface with upper and lower limits over the testing data set, Figure 58, indicates a rough surface with no visible trends. A significant peak is observed when 36 input and 64 hidden neurons are considered. Further, the limits increase in volatility as the structure becomes more complex.

The best estimate MSE surface, Figure 59, indicates two valleys in relation to the input neurons. One is found between a single and 3 input neurons per data set. The other is between 12 and 24 input neurons per data set. Consideration of the hidden neurons indicates a similar characteristic when 32 neurons are used. The minimum MSE, of 3.45E-04, is achieved using a single input neuron per data set and 4 hidden neurons.

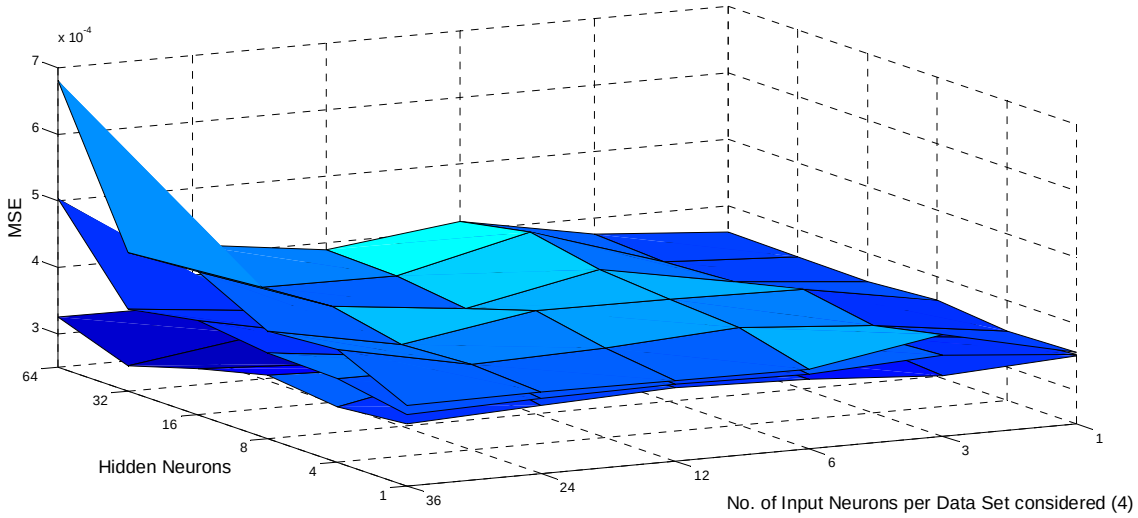

Figure 58: MSE surface with upper and lower limits – testing data set – bond market one step integrated ANN

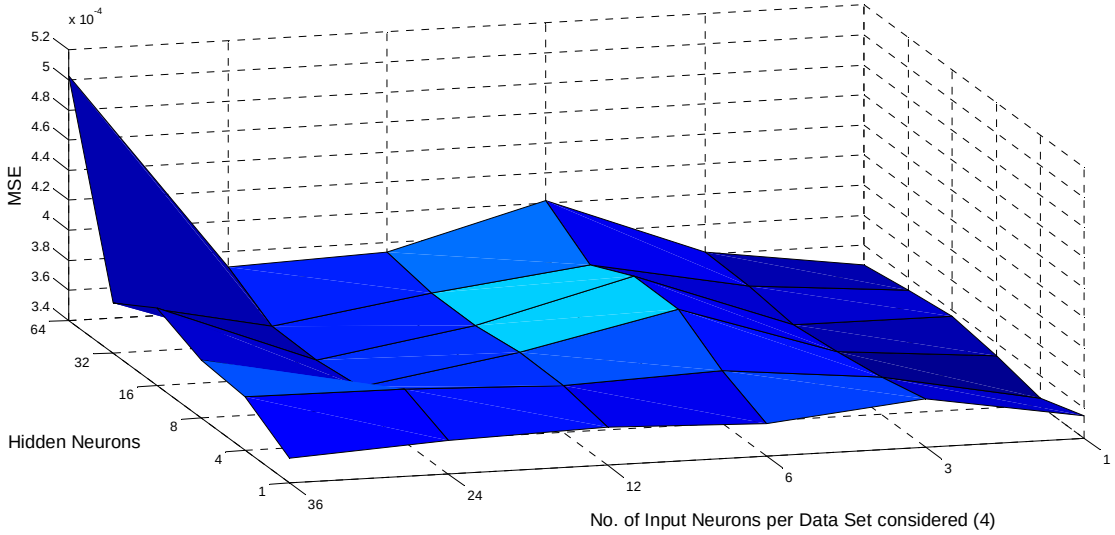

Figure 59: Best estimate MSE surface – testing data set – bond market one step integrated ANN

Conclusion – Input and Hidden Neurons

Parameters that minimize and balance the errors over the training and testing sets, are chosen. This structure consists of 3 input neurons per data set (12 input neurons in total) and 32 hidden neurons. The increase in MSEs over the training and testing sets, from their respective minima are 4.18E-04 and 0.16E-04, respectively.

The choice made has subjective components. It is important not to base the decision purely on the results obtained from the training set as this will lead to over fitting. Similarly, basing the decision purely on the results over the testing set will cause the ANN to only fit the testing set. Both the scenarios described above will reduce the generalization ability of the ANN, leading to poor forecasting accuracy.

Conclusion of Determining an Efficient Structure

Efficient parameter estimates are:

- 3 input neurons per data set (12 input neurons in total),
- 8 seasonal input neurons,
- 32 hidden neurons,
- A learning rate increase of 1.005 (0.5% increase), and
- A learning rate decrease of 0.8 (20% decrease).

4.8.2 Analysing the Efficient ANN

The ANN is trained using the following parameters (Table 58).

Network Structure (input neurons per data set x number of data sets) – hidden neurons – output neurons	12 (3x4)-32-1
Training set size	325
Testing set size	100
Dummy variables (for seasonality) (variables per data set x number of data sets)	8 (2x4)
Number of initialized structures	10
Epochs	3 000
Training algorithm	RPROP
Learning rate change:	
Increase	0.5% (1.005)
Decrease	20% (0.8)
Maximum	100
Minimum	0.000000001

Table 58: Parameters for the efficient ANN structure – bond market one step integrated ANN

Observations of System Error

Over fitting in the ANN occurs after approximately 250 training epochs, as the RMSE over the testing set begins to increase after this point. This is illustrated in Figure 60.

The minimum RMSE of 0.0188 over the testing set is obtained after approximately 250 training epochs. Intersection of the errors does not occur within the first 3 000 epochs. To avoid over fitting, the efficient number of training epochs is chosen to be 250. Computing requirements are minimised and both errors fairly balanced with this decision.

Error Readings	Testing Set	Training Set
Minimum RMSE	0.0188	0.0215
Minimum MSE	3.5279e-004	4.6091e-004
Final RMSE after training *(250 epochs)*	0.0189	0.0252
Final MSE after training *(250 epochs)*	3.5677e-004	6.3347e-004

Table 59: Results of efficient model after training – bond market one step integrated ANN

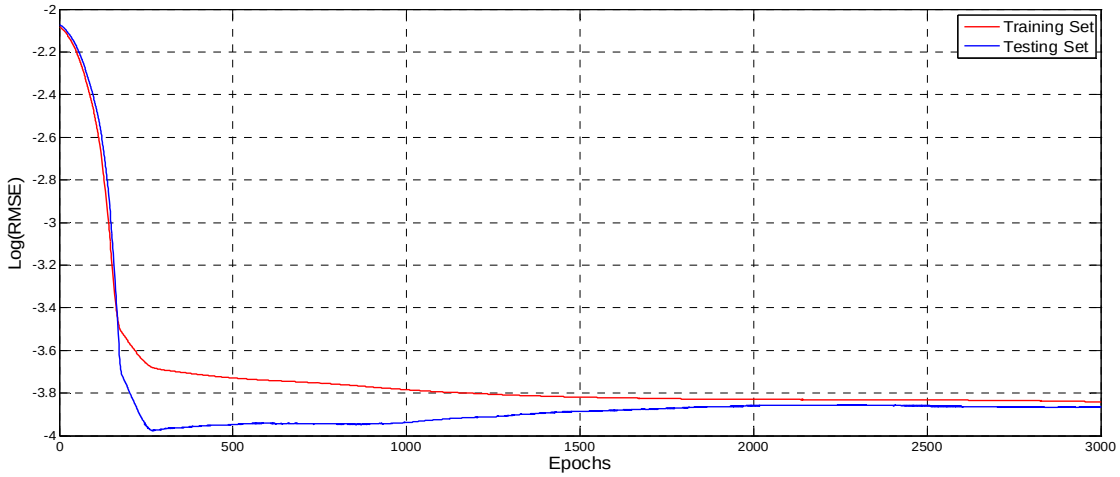

Figure 60: Natural logarithm of RMSE over the training process – bond market one step integrated ANN

Observations of Forecast and Actual Returns

The ANN captures a limited number of trends underlying the bond market. There are many occasions where the ANN either under or overestimates the return on the bond market. These instances are the result of external factors on the market and cannot be explained using past observations of the market. The best estimate forecasts over the period considered are represented by the blue line in Figure 61.

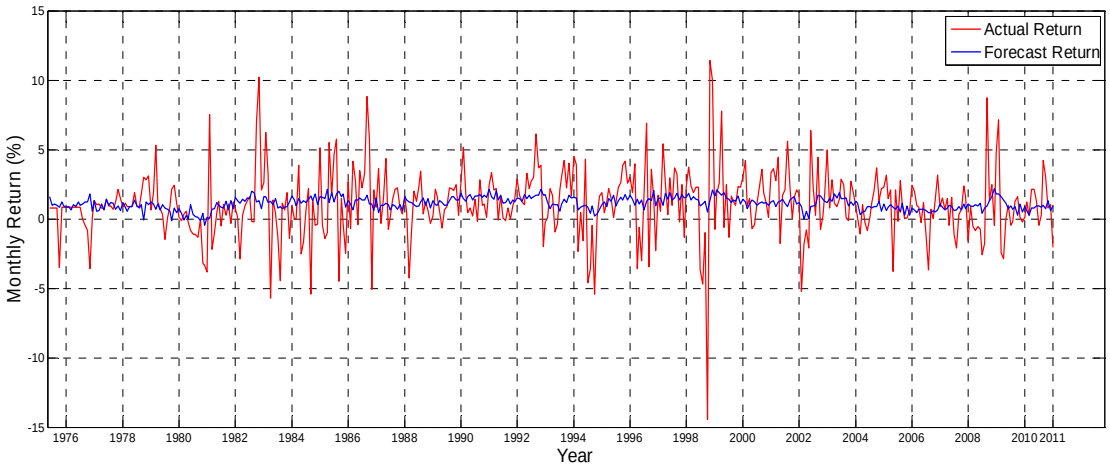

Figure 61: Actual and forecast return on the bond market from 1975 to 2010 – bond market one step integrated ANN

Conclusion - Analysing the Efficient ANN

The ANN captures a limited number of trends underlying the data. MSEs of 6.3347e-004, over the training set, and 3.5677e-004, over the testing set, are achieved using the efficient ANN structure. The performance of the ANN over both sets of data is similar. The effectiveness of the ANN is determined in chapter 6.

The minimum RMSE of 0.0188 over the testing set and 0.0215 over the training set are achieved after approximately 250 and 3 000 epochs, respectively. The RMSE over the training set decreases as the training progresses past 250 epochs and continues to decrease until the maximum number of epochs is obtained.

After 250 epochs of training, which is the efficient training period, RMSEs of 0.0189 and 0.0252 are achieved over the testing and training data sets respectively. The efficient number of training epochs is chosen through consideration of errors over both data sets and the computing equipment available.

The RMSEs are compared to those from other applications and are found to be larger. This indicates the limited relationship between future and past returns on the bond market. This is supported by theory, which suggests returns on the bond market are largely driven by demand and supply factors that do not depend heavily on past returns.

4.8.3 Summary of Experiment

Table 60 provides a summary of the results from this experiment.

Network Structure (input neurons per data set x number of data sets) – hidden neurons – output neurons	12 (3x4)-32-1
Training epoch required for optimal training	250
Minimum Root Mean Squared Error: Training Data Set Testing Data Set	 0.0215 0.0188
Final Root Mean Squared Error (250 epochs): Training Data Set Testing Data Set	 0.0252 0.0189
Parameters for RPROP: Learning Rate Increase Learning Rate Decrease	 0.5% (1.005) 20% (0.8)
Data Sets: Training Set Testing Set	 328 observations (1975 – 2002) 100 observations (2002 – 2010)
Times the structure is initialised	10

Table 60: Summary of experiment's result – bond market one step integrated ANN

4.9 Integrated Equity Market ANN – Three-Month Forecast

This section aims to construct an ANN that efficiently forecasts return on the equity market three months in advance, and allows full interaction between inflation and the three markets.

4.9.1 Determining an Efficient Structure

An efficient estimate of each parameter is obtained in this experiment.

4.9.1.1 Part 1 – Efficient Number of Training Epochs

The parameters used for this experiment are tabulated below (Table 61).

Parameter	Value
Network Structure (input neurons per data set x number of data sets) – hidden neurons – output neurons	144 (36x4)-64-1
Training set size	293
Testing set size	100
Dummy variables (for seasonality) (variables per data set x number of data sets)	8 (2x4)
Number of initialized structures	5
Epochs	1 000
Training algorithm	RPROP
Learning rate change:	
Increase	0.5% (1.005)
Decrease	20% (0.8)
Maximum	100
Minimum	0.000000001

Table 61: Parameters for experiment – Part 1 – equity market three step integrated ANN

Observations – Part 1

The minimum RMSE of 0.0624 over the testing set is achieved after approximately 330 training epochs. Over fitting occurs after this point, as indicated in Figure 62 by the general increase of the RMSE over the testing set after this point.

The RMSE is calculated, at different intervals, over the training process which is presented in Table 62. An increase in RMSE, over the testing set, occurs between 500 and 1 000 epochs. This suggests over fitting is present during this interval. The RMSE over the training set continues to decrease throughout the training process, as expected.

		Training Epochs				
	Min RMSE	**100**	**500**	**1 000**	**Δ in RMSE – 100 to 500**	**Δ in RMSE – 500 to 1 000**
RMSE – Testing Set	0.0624	0.2863	0.0762	0.0883	0.2101	-0.0121
RMSE – Training Set	0.0073	0.2861	0.0249	0.0073	0.2612	0.0176

Table 62: RMSE after different numbers of epochs – Part 1 – equity market one step integrated ANN

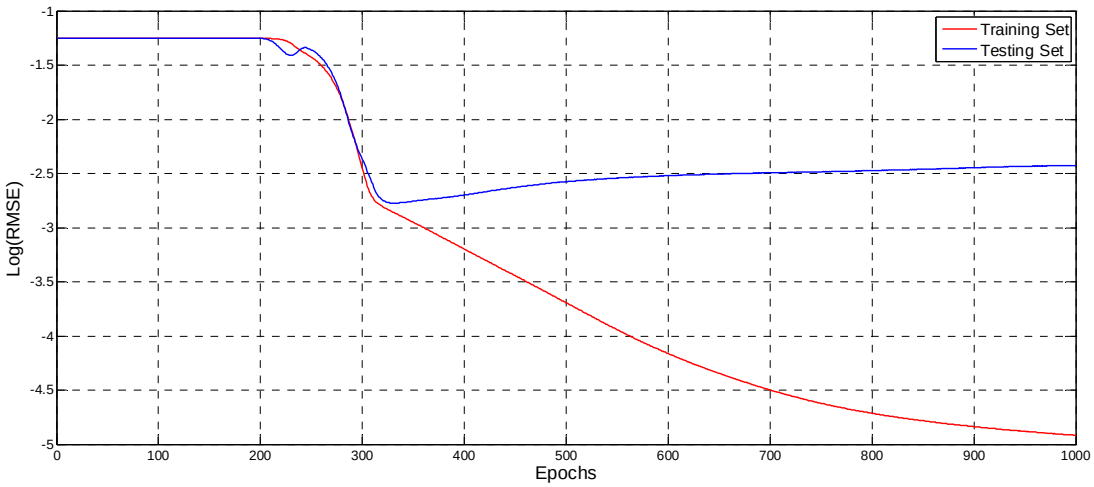

Figure 62: Natural logarithm of RMSE over the training process for training and testing data set – Part 1 – equity market three step integrated ANN

Conclusion – Part 1

The number of epochs used to compare different ANN structures is 1 000, as over fitting occurs within this period. This implies the minimum RMSE over the testing set has been surpassed and captured. The increase in RMSE over the testing set, from 500 to 1 000 epochs, supports that the minimum of 0.0624 is achieved within 1 000 epochs. Figure 62 further supports this inference.

4.9.1.2 Part 2 – Determining Efficient Parameters

Observations – Learning Rate Increase/Decrease

The learning rate increase of 1.05 (5% increase) and decrease of 0.8 (20% decrease) result in the minimum MSE of 2.40E-03 over the training set. The minimum MSE of 2.95E-03 over the testing set is achieved using an increase of 1.005 (0.5% increase) and a decrease of 0.2 (80% decrease). The observations over each data set do not support the same combination of increase and decrease.

The range of the MSE surface of 1.26E-03 over the training set is significant in comparison to that of 0.08E-03 for the testing set.

An efficient learning rate change combination, which minimize computing requirement and balance the MSE over both data sets, is an increase of 5% and a decrease of 50%. This combination leads to an increase in MSE of 0.42E-03, over the training set, and 0.07E-03, over the testing set, from their respective minima.

The MSEs for each combination of learning rate, over each data set, is tabulated below (Table 63). The minimum MSEs are marked in green and those associated with the efficient combination are marked in orange.

Training Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	3.66E-03	3.11E-03	3.18E-03
	0.5	3.13E-03	**2.82E-03**	2.82E-03
	0.8	2.67E-03	2.40E-03	2.55E-03
	min	2.40E-03		
Testing Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	2.95E-03	2.96E-03	2.99E-03
	0.5	3.01E-03	**3.02E-03**	3.02E-03
	0.8	3.02E-03	3.03E-03	3.02E-03
	min	2.95E-03		
	Minimum			
	Efficient			

Table 63: MSE for learning rate changes – equity market three step integrated ANN

Conclusion – Learning Rate Increase/Decrease

The following decisions are made:

- The learning rate decrease of 0.5 is to be used, i.e. a 50% decrease in learning rate if the error of the system increases during training.
- The learning rate increase of 1.05 is to be used, i.e. a 5% increase in learning rate if the error of the system decreases during training.

Input and Hidden Neurons – Training Set's MSE Surface

As the ANN structure increases in complexity, the MSE decreases. This is supported by the Figure 63, which depicts the MSE surface with upper and lower limits over the training set. Increasingly complex structures are more vulnerable to over fitting than simpler structures, which causes the general decrease in MSE.

Similarly, the best estimate MSE surface, Figure 64, indicates a general decrease as the complexity of the ANN structure increases. This surface minimises, at an MSE of 1.39E-04, when 36 input neurons per data set and 64 hidden neurons are considered.

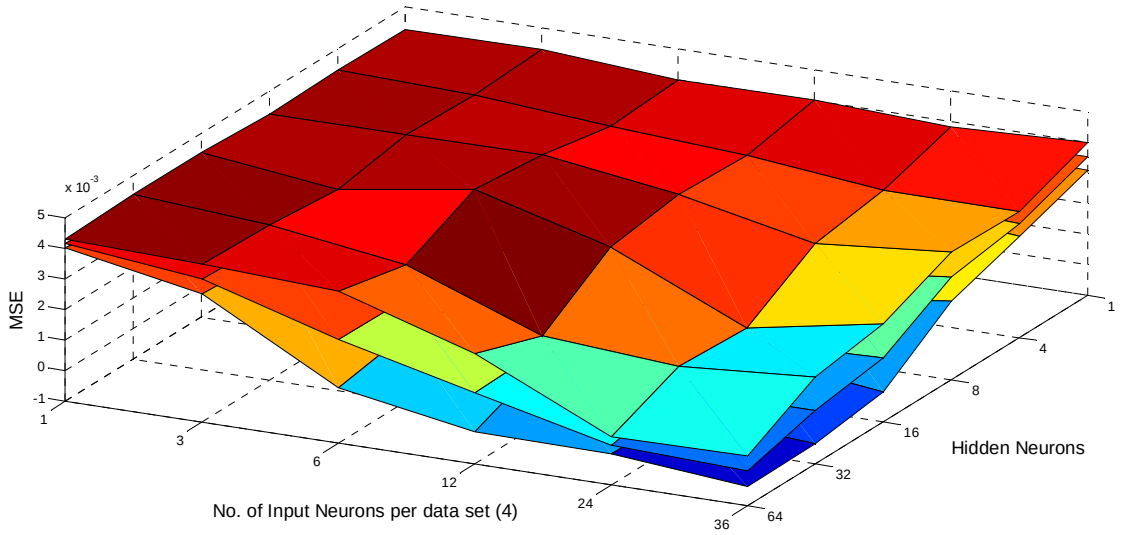

Figure 63: MSE surface with upper and lower limits – training data set – equity market three step integrated ANN

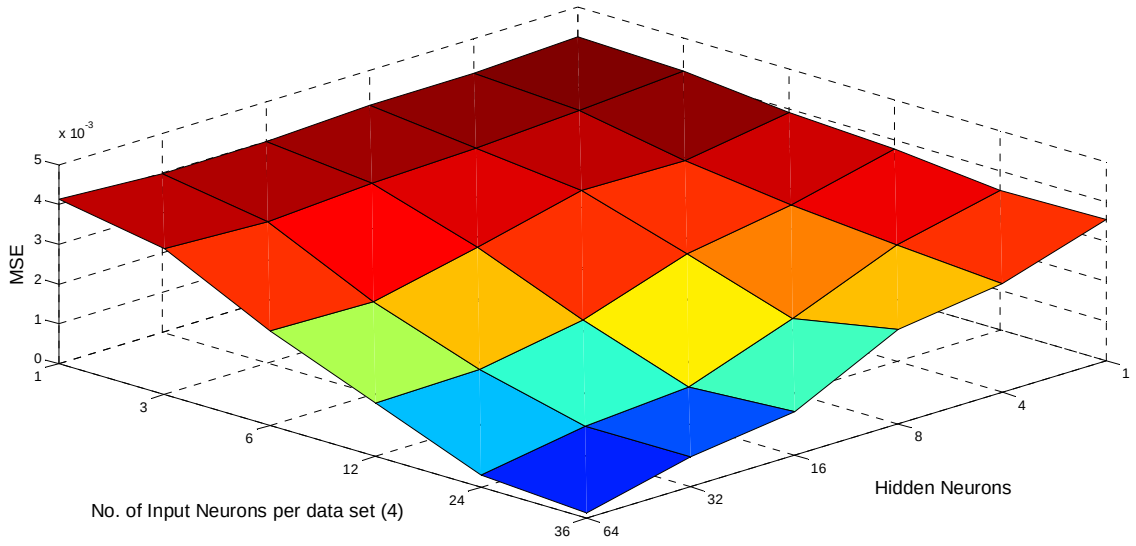

Figure 64: Best estimate MSE surface – training data set – equity market three step integrated ANN

Input and Hidden Neurons – Testing Set's MSE Surface

The MSE surface with upper and lower limits over the testing set, Figure 65, suggests a relatively stable surface, with volatility increasing when the system increases in complexity.

The best estimate MSE surface over the testing set, Figure 66, indicates a general increase in MSE as the number of hidden neurons increases. A similar trend is indicated by the input neurons. This trend is exacerbated as the numbers of hidden and input neurons increases simultaneously. Over fitting is likely the cause of these trends, as complex structures and over fitting result in larger errors over the testing set of data. The minimum of 2.59E-03 of the best estimate MSE surface is achieved using 36 input neurons per data set (144 input neurons in total) and a single hidden neuron.

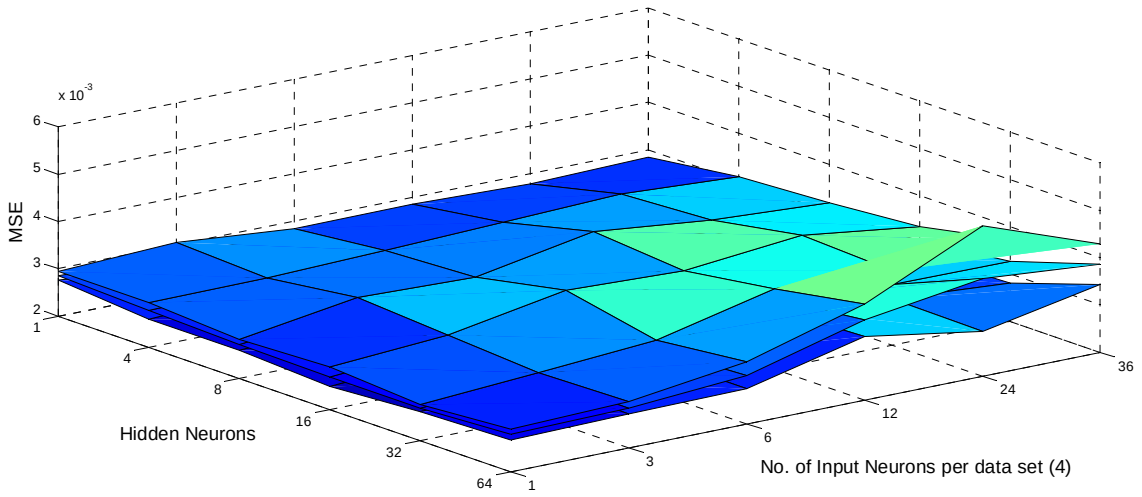

Figure 65: MSE surface with upper and lower limits – testing data set – equity market three step integrated ANN

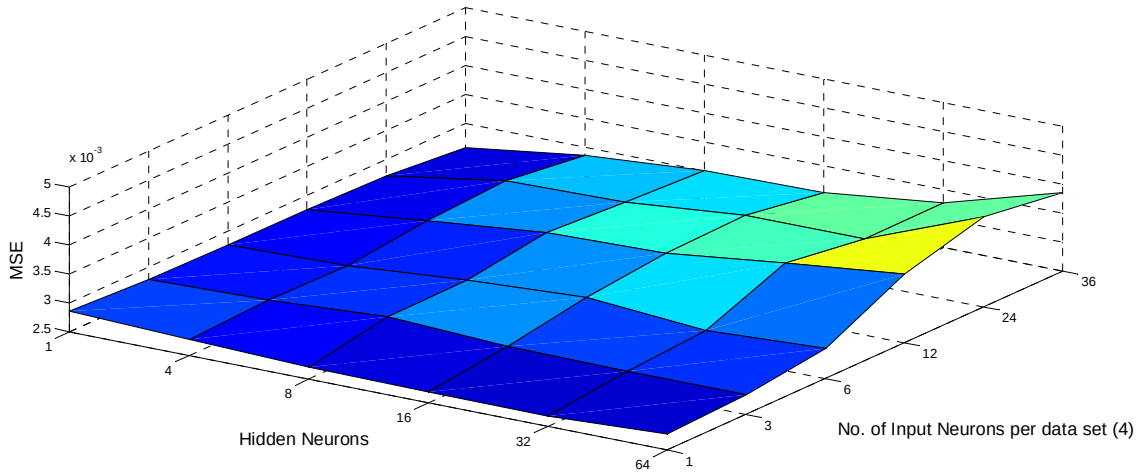

Figure 66: Best estimate MSE surface – testing data set – equity market three step integrated ANN

Conclusion – Input and Hidden Neurons

A parsimonious structure is chosen which balances the errors over both data sets. The structure chosen consists of 3 input neurons per data set (12 input neurons in total) and 16 hidden neurons. The MSEs over the training and testing data sets associated with this structure are 3.74E-03 and 2.86E-03, respectively.

The choice made has subjective components. It is important not to base the decision purely on the results obtained from the training set as this will lead to over fitting. Similarly, basing the decision purely on the results over the testing set will cause the ANN to only fit the testing set. Both the scenarios described above will reduce the generalization ability of the ANN, leading to poor forecasting accuracy.

Conclusion of Determining an Efficient Structure

Efficient parameter estimates are:

- 3 input neurons per data set (12 input neurons in total),
- 8 seasonal input neurons,
- 16 hidden neurons,
- A learning rate increase of 1.05 (5% increase), and
- A learning rate decrease of 0.5 (50% decrease).

4.9.2 Analysing the Efficient ANN

The ANN is trained with the following parameters (Table 64).

Network Structure (input neurons per data set x number of data sets) – hidden neurons – output neurons	12 (3x4)-16-1
Training set size	326
Testing set size	100
Dummy variables (for seasonality) (variables per data set x number of data sets)	8 (2x4)
Number of initialized structures	10
Epochs	5 000
Training algorithm	RPROP
Learning rate change:	
Increase	5% (1.05)
Decrease	50% (0.5)
Maximum	100
Minimum	0.000000001

Table 64: Parameters for the efficient ANN structure – equity market three step integrated ANN

Observations of System Error

Over fitting begins after approximately 50 training epochs, indicated by the increase in the RMSE over the testing set in the plot of the natural logarithm of the RMSE over the training process (Figure 67). The simple ANN structure limits the effect of over fitting in the system, as indicated by the stable RMSE over both sets of data.

The minimum RMSE of 0.052 over the testing set is obtained after approximately 50 epochs. There is no intersection of the errors over the training process. The error over the training set continues to decrease over the training process.

The efficient number of training epochs is chosen to be 50, as this balances the errors over both data sets, ensures a stable error over both sets and avoids over fitting. The numerical error values at points of interest are provided in Table 65.

Error Readings	Testing Set	Training Set
Minimum RMSE	0.0520	0.0619
Minimum MSE	0.0027	0.0038
Final RMSE after training *(50 epochs)*	0.0522	0.0655
Final MSE after training *(50 epochs)*	0.0027	0.0043

Table 65: Results of efficient model after training – equity market three step integrated ANN

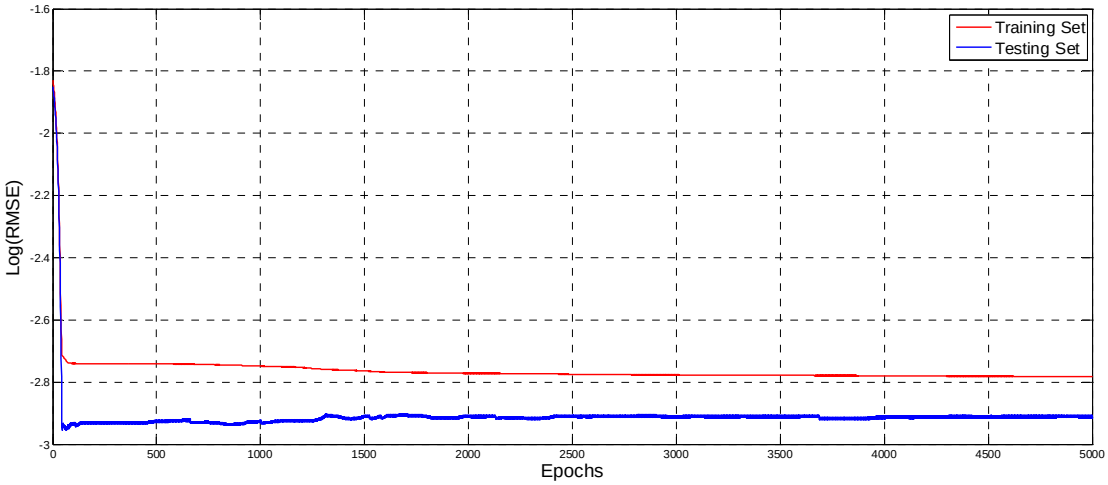

Figure 67: Natural logarithm of RMSE over the training process – equity market three step integrated ANN

Observations of Forecast and Actual Returns

The ANN captures few underlying trends in the data. There is significant under and overestimation of returns on the equity market. This is expected, because the equity market is largely affected by external factors. Excluding these factors lead to poor forecasting accuracy. The best estimate forecast returns are indicated by the blue line in Figure 68, while the actual returns over the same period are represented by the red line.

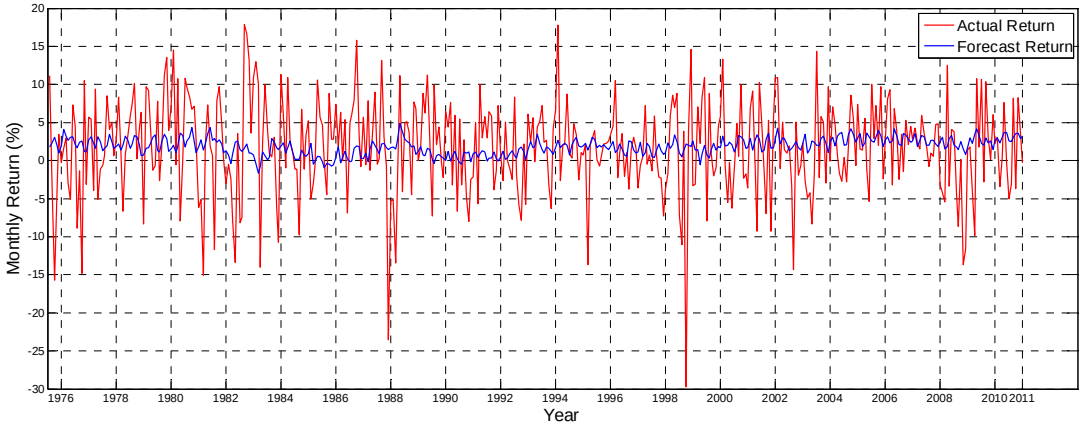

Figure 68: Actual and forecast return on the equity market from 1975 to 2010 – equity market three step integrated ANN

Conclusion - Analysing the Efficient ANN

The ANN captures few trends underlying the data. To effectively model the returns on the equity marked it is necessary to incorporate relevant external factors. The minimum RMSEs of 0.052 over the testing set and of 0.0619 over the training set are obtained after approximately 50 and 5 000 epochs, respectively. The RMSE over the training data set continues to decrease over the entire training process.

The efficient structure is trained over 50 epochs, which results in RMSEs of 0.0655 and 0.0522 over the training and testing data sets respectively. Limited over fitting is present in the efficient structure, due to the simple structure chosen.

The RMSE of 0.0522 over the testing set is the largest in comparison to the other applications. This indicates returns on the equity market are most difficult to forecast, as expected. Further, it suggests there are few relationships between successive returns on the equity market over the period considered.

4.9.3 Summary of Experiment

Table 66 provides a summary of the results from this experiment.

Network Structure (input neurons per data set x number of data sets) – hidden neurons – output neurons	12 (3x4)-16-1
Training epoch required for optimal training	50
Minimum Root Mean Squared Error: Training Data Set Testing Data Set	 0.0619 0.0520
Final Root Mean Squared Error (50 epochs): Training Data Set Testing Data Set	 0.0655 0.0522
Parameters for RPROP: Learning Rate Increase Learning Rate Decrease	 5% (1.05) 50% (0.5)
Data Sets: Training Set Testing Set	 326 observations (1975 – 2002) 100 observations (2002 – 2010)
Times the structure is initialised	10

Table 66: Summary of experiment's result – equity market three step integrated ANN

4.10 Results of Experiments

Table 67 and Table 68 provide the results of 16 experiments conducted. Sections 4.6 – 4.9 provide detailed descriptions of 4 out of the 16 experiments. Appendix A through L provide the description of the remaining 12 experiments.

Summary of Experiment's Results - Parameters				
	Structure	**Efficient Epochs**	**Learning Rate Increase**	**Learning Rate Decrease**
1 Step (Isolated)				
Inflation	24-32-1	3,500	0.50%	20%
Money	6-32-1	100,000	0.50%	20%
Bond	6-32-1	2,000	5.00%	20%
Equity	6-16-1	49	5.00%	20%
3 Step (Isolated)				
Inflation	12-32-1	4,999	5.00%	20%
Money	6-64-1	49,999	0.50%	20%
Bond	12-32-1	59	5.00%	20%
Equity	6-32-1	50	5.00%	20%
1 Step (Integrated)				
Inflation	24(6x4)-16-1	850	0.50%	20%
Money	24(6x4)-16-1	10,000	0.50%	20%
Bond	12(3x4)-32-1	250	0.50%	20%
Equity	12(3x4)-32-1	2,200	5.00%	20%
3 Step (Integrated)				
Inflation	48(12x4)-16-1	4,000	0.50%	20%
Money	12(3x4)-8-1	4,200	0.50%	20%
Bond	48(12x4)-8-1	50	5.00%	20%
Equity	12(3x4)-16-1	50	5.00%	50%

Table 67: Parameter estimations from experiments – Application of ANNs to Time Series Forecasting

Summary of Experiment's Results - Errors		
	RMSE (Training)	**RMSE (Testing)**
1 Step (Isolated)		
Inflation	0.0048	0.0048
Money	0.000474	0.000217
Bond	0.0233	0.0188
Equity	0.0652	0.0525
3 Step (Isolated)		
Inflation	0.0049	0.0052
Money	0.0011	0.000645
Bond	0.0260	0.0193
Equity	0.0647	0.0525
1 Step (Integrated)		
Inflation	0.0045	0.0045
Money	0.000496	0.000484
Bond	0.0252	0.0189
Equity	0.0514	0.0520
3 Step (Integrated)		
Inflation	0.0047	0.0052
Money	0.000908	0.000736
Bond	0.0248	0.0189
Equity	0.0655	0.0522

Table 68: RMSE from experiments – Application of ANNs to Time Series Forecasting

Warning: *As the ANN structures are randomly initialized it is unlikely that identical results will be obtained if the experiments are repeated; however, the general trends should hold every time the experiments are repeated.*

5 Hybrid Models, Cross Validation and Extended Forecasts

This chapter has three aims.

1. Investigate the performance of hybrid models compared to ANNs and traditional models.
2. Determine the general forecasting ability of ANNs compared to traditional models. This is done by varying the data sets used to fit and evaluate the models.
3. Consider a longer forecast period with respect to the money market.

The hybrid models are constructed and evaluated in sections 5.1-5.3 below. This is followed by cross validation of ANNs and traditional models in section 5.4. Finally, in section 5.5, the money market model and the associated forecasts are considered for a forecast period of twelve months.

5.1 Hybrid Models

The aim of this section is to combine the traditional models and ANNs when forecasting inflation and the money market[5]. Only these two indices are considered in this case as neither model type, from chapter 3 and 4, detects significant trends for the bond and equity markets. The combined or 'hybrid' models are compared to the traditional models and ANNs in chapter 6.

5.1.1 Traditional Models

The traditional models used are those constructed in Chapter 3. The error between the forecast and actual index values are determined for the models. The set of errors is used as the residual data set.

5.1.2 Artificial Neural Networks

The residual data set is presented to the ANNs for analysis. This ensures the non-linear relationships in the data are captured in the hybrid model.

The ANN structures used are simpler than those developed in chapter 4. The smallest learning rate increase and decrease are used, as these are favoured by the ANNs constructed in chapter 4. A simpler ANN structure is used because fewer relationships remain in the data after the traditional model removes a significant number of them. This approach of parameter estimation reduces the efficiency of the ANNs; however, the aim of this section is to determine whether the hybrid models improve or reduce forecasting accuracy in comparison to the other models. Hence, if an inefficient ANN adds value to the traditional models the efficient ANN will add increased value.

Both the integrated and isolated ANNs are integrated into the traditional model. This is performed over both a single and three-month forecast period.

[5] *For an application of similar hybrid models refer to section 2.9.6*

5.1.3 Calculation of Forecasts

The forecasts are calculated by using all the past forecasts generated by the traditional models. The past forecasts are compared to the actual values of each index and residuals are determined. ANNs are trained on these residuals. Forecasts of the residuals are generated by the ANNs and combined with the forecasts from the traditional models. The forecasts which include the forecast values and forecast residuals are compared to the actual values of each index. This allows for the calculation of the RMSE for each application considered.

5.1.4 Expectation from Hybrid Models

The hybrid models are able to detect both linear and non-linear patterns in the data. They are expected to outperform both the traditional models and ANNs, and at worse perform as well as the best of either the traditional models or ANNs.

5.1.5 Definition of Forecast Period

The definition of a three-month forecast period is as follows. The three-month forecast of inflation or return is the monthly inflation rate or return over the third month, two months from the moment the forecast is generated. It is the forecast of inflation or return from month $t+2$ to $t+3$, where t is the month in which the forecast is generated.

5.2 Inflation Hybrid Model

The hybrid models, used to forecast inflation, are constructed with an isolated or integrated ANN. Further, forecasts are generated for a one and three-month forecast period.

5.2.1 One Month Forecast Period – Inflation Hybrid

First the hybrid model with an isolated ANN is considered, followed by the hybrid model with an integrated ANN. In this case a one month forecast period is considered.

5.2.1.1 Hybrid with Isolated ANN

The ANN parameters used in the hybrid model are given in Table 69.

Network Structure	3-8-1
Training epoch required for optimal training	500
Parameters for RPROP:	
Learning Rate Increase	0.5% (1.005)
Learning Rate Decrease	20% (0.8)
Times the structure is initialised	25

Table 69: Inflation hybrid model with isolated ANN parameters – one month forecasts

The results of the hybrid model after the ANN is efficiently trained are provided in Table 70.

Data Set	RMSE
Training	0.004926
Testing	0.004436

Table 70: RMSEs – inflation hybrid (isolated ANN, one step)

The error over the training set is greater than over the testing set. This is similar to the results from the traditional models and ANNs constructed in chapter 3 and chapter 4, respectively.

The one step forecasts of inflation, from 1975 to 2010, generated by the hybrid model with an isolated ANN are plotted in the line graph, Figure 69.

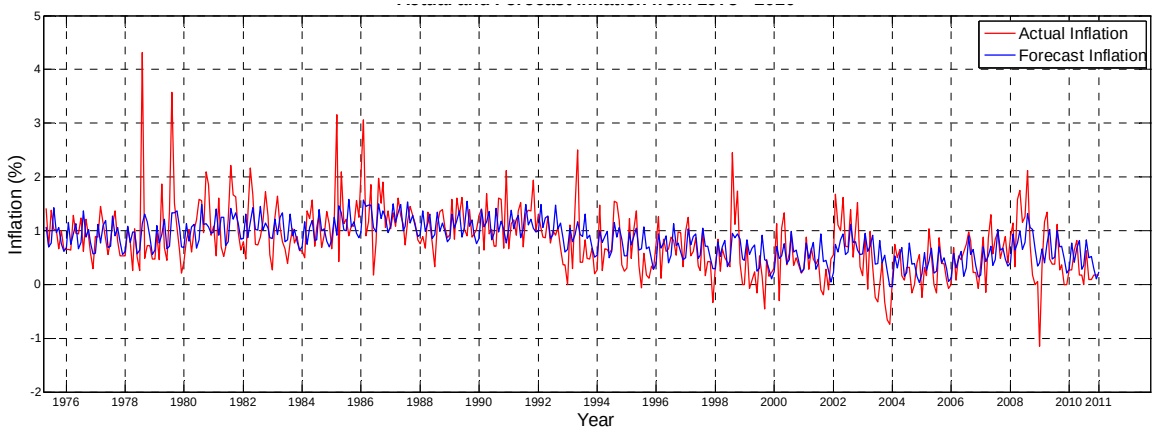

Figure 69: Actual and forecast inflation generated by isolated hybrid model for 1975 to 2010 – one month forecasts

5.2.1.2 Hybrid with Integrated ANN

The ANN parameters used in the hybrid model with an integrated ANN are given in Table 71.

Network Structure	12(3x4)-4-1
Training epoch required for optimal training	200
Parameters for RPROP:	
Learning Rate Increase	0.5% (1.005)
Learning Rate Decrease	20% (0.8)
Times the structure is initialised	20

Table 71: Inflation hybrid model parameters with integrated ANN – one month forecasts

The results of the hybrid model after the ANN is efficiently trained are provided in Table 72.

Data Set	RMSE
Training	0.005145
Testing	0.00469

Table 72: RMSEs– inflation hybrid (integrated ANN, one step)

The one step forecasts of inflation, from 1975 to 2010, generated by the hybrid model with an integrated ANN are plotted in the line graph below, Figure 70.

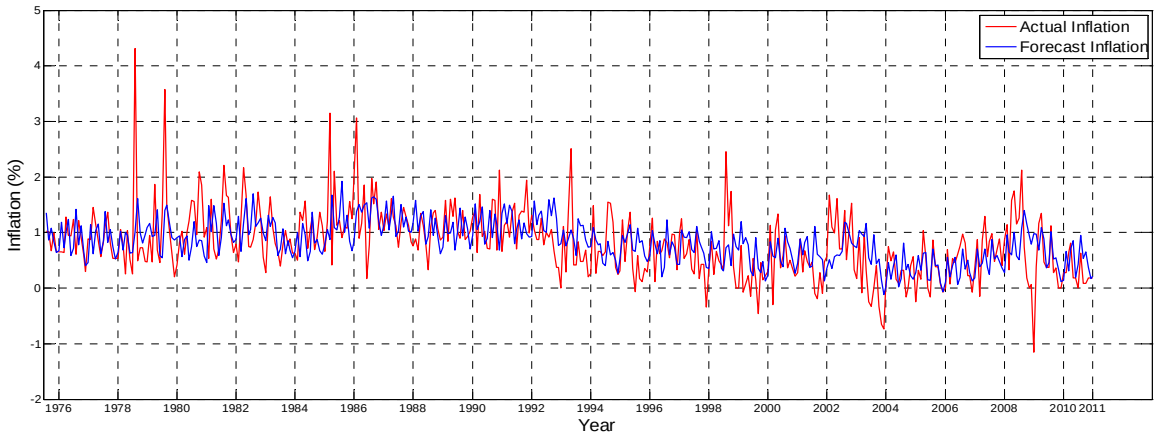

Figure 70: Actual and forecast inflation generated by integrated hybrid model from 1975 to 2010 – one step forecasts

The above process is repeated for the three-step inflation case. Further, it is repeated for the one and three-step case for the money market. To view details on these experiments refer to appendix M.

5.3 Summary of Experiment – Hybrid Models

The results from the hybrid models is provided in Table 73. In chapter 6 these results are analysed and used to compare hybrid models to traditional models and ANNs.

Hybrid Models - Errors		
	RMSE	
Inflation	***Training Set***	***Testing Set***
1 Step		
Isolated	0.004926	0.004436
Integrated	0.005145	0.00469
3 Steps		
Isolated	0.005316	0.004807
Integrated	0.005194	0.004916
Money Market		
1 Step		
Isolated	0.000301	0.000117
Integrated	0.000313	0.000116
3 Steps		
Isolated	0.001077	0.000523
Integrated	0.001146	0.000516

Table 73: Summary of hybrid models

5.4 Cross Validation of Models

This section aims to determine the general forecasting accuracy of ANNs, both in isolation and when integrated, and the traditional models. The results are used to compare ANNs to traditional models. To allow this comparison the performance of the two approaches is evaluated over different intervals of the data. This reduces the effect random variation can have on the comparison of the two approaches.

5.4.1 Practical Applications

For this analysis, only the one month forecast of inflation is considered. Both the isolated and integrated ANNs are investigated. In future studies the scope must be expanded to include the money market, hybrid models and a forecast period of three months.

5.4.2 Data Sets

The data is divided into five sets, each with 100 observations. Further, each set is divided into a training and a testing set. The training set consists of the first 80 observations and the testing set consists of the remaining 20. Table 74 provides details on the five sets.

Data Sets				
Set	**Beginning of Training Set**	**End of Training Set**	**Beginning of Testing Set**	**End of Testing Set**
1	Jan-75	Aug-81	Sep-81	Apr-83
2	Sep-81	Apr-88	May-88	Dec-89
3	Jan-90	Aug-96	Sep-96	Apr-98
4	May-98	Dec-04	Jan-05	Aug-06
5	Sep-02	Apr-09	May-09	Dec-10

Table 74: Details of data sets for evaluation of general forecasting ability of ANN and traditional model

5.4.3 Model Construction

The models, both traditional and ANNs, consist of the same structure as those constructed in chapter 3 and 4 for the purpose of forecasting inflation one month ahead. In the ANN case, the efficient number of training epochs is re-estimated over each data set. For the traditional models, the parameters are re-estimated over each data set. This implies that the parameter values of the traditional model and epochs used in the ANN change over each data set; however, the model structure remains constant.

5.4.4 Results for Experiment – Inflation Application

Table 75 presents the RMSE over the testing set for the five data sets considered. This includes the results of the traditional model, isolated ANN and integrated ANN.

RMSE over Testing Set for Five Independent Data Sets			
Data Sets	**Traditional Model**	**Isolated ANN**	**Integrated ANN**
1	5.44E-03	5.26E-03	5.02E-03
2	3.47E-03	3.40E-03	3.16E-03
3	3.58E-03	3.14E-03	4.48E-03
4	2.94E-03	3.16E-03	3.99E-03
5	3.01E-03	2.59E-03	2.86E-03

Table 75: RMSE over testing set for five independent data sets

The histogram in Figure 71 depicts the RMSE over the testing set for each of the five data sets. This is done for the traditional model, isolated ANN and integrated ANN.

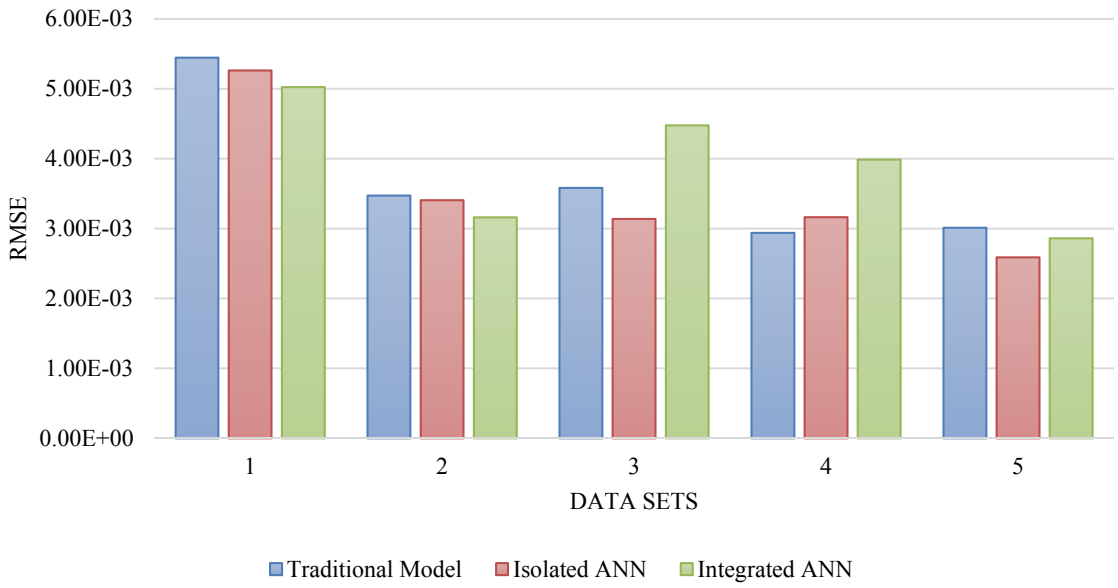

Figure 71: Column chart of RMSEs over testing set for models over five different sub data sets – Inflation one step models.

The RMSE of the isolated ANN is smaller than that of the traditional model for four of the five data sets considered. This suggests that on average the isolated ANN performs more efficiently than the traditional model; however, statistical analysis on this conclusion is performed in chapter 6.

For data sets 1 and 2 the traditional model has the greatest RMSE, the isolated ANN has a smaller RMSE than the traditional model and the integrated ANN has the smallest RMSE. This is expected, as the integrated ANN can use data from other markets to explain the variation within inflation.

For data sets 3 and 5 the isolated ANN has the smallest RMSE. The integrated ANN does not outperform the isolated ANN, as it captures a significant amount of random variation in these data sets. The traditional model outperforms the

integrated ANN when data set 3 is considered. This is due to the integrated ANN capturing the increased random variation within data set 3.

Data set 4 is of interest because the traditional model outperforms the isolated ANN, which in turn outperforms the integrated ANN. This suggests that the traditional model captures more relationships in the data than the ANNs over the period considered. It is important that research be done to verify and explain this phenomenon.

The relationships in the data have changed over the period January 1975 to December 2010. This is indicated by the change in RMSE for the different data sets. If the trends remain constant the RMSE will be the same for each data set considered, because the sizes of the data sets are identical.

5.5 Money Market Twelve-Month Forecasts

This section determines the twelve-month forecast accuracy of the traditional model and ANNs constructed in chapters 3 and 4, respectively. The additional forecast period of twelve months is considered for the money market as it is stable in the short-term. Further, short-term forecasts have a limited value with regard to actuarial analysis. Considering a twelve-month forecast period allows the difference between the traditional model and ANNs to be accentuated and analysed with increased clarity.

The ANNs use the parameters associated with the three-month forecasting ANNs (Table 67) and are not re-estimated in this application. The RMSE over the training and testing sets associated with the traditional model, integrated ANN and isolated ANN are presented in Table 76.

Twelve Step Forecasts – Money Market		
	Training set	**Testing Set**
Traditional model	0.003163	0.002126
Isolated ANN	0.00434	0.001834
Integrated ANN	0.004725	0.001967

Table 76: money market twelve-month forecast RMSEs

Table 76 suggests the ANNs outperform the traditional model over the testing set when forecasting twelve months into the future. This observation and the significance thereof is analysed and clarified in chapter 6.

.

6 Results and Conclusions

6.1 Aim of this Chapter

Several aims of the study are achieved in this chapter.

1. The traditional models and ANNs constructed in chapters 3 and 4 are analysed and compared.
2. The general one month forecasting ability of the ANNs and traditional model for inflation is analysed and compared.
3. The performance of hybrid models is analysed and compared.
4. The research questions are answered.

When discussing the performance of one and three-step forecasts, it is important to note that a step is identical to a month. Further, the definition of a three-month forecast period is as follows. The three-month forecast of inflation or return is the monthly inflation rate or return over the third month, two months from the moment the forecast is generated. It is the forecast of inflation or return from month $t+2$ to $t+3$, where t is the month in which the forecast is generated.

6.2 Effectiveness of ANNs and Traditional Models

This section discusses the effectiveness of the traditional models and ANNs at forecasting the markets and inflation. The models considered are those constructed in chapters 3 and 4. Their effectiveness is measured using the coefficient of determination, R-squared (R^2) value defined in section 3.3.2. This value provides the goodness of fit a model achieves to a data set and is a good measure of performance. Further, only the results from the testing data set are considered when evaluating the performance of the ANNs. This decision is made because ANNs are capable of having arbitrarily small errors over the training data set while being a poor model. Thus, considering the ANNs over the training set will lead to skewed conclusions.

6.2.1 Measure of Effectiveness

The coefficient of determination (R^2 value) is used to measure effectiveness of the models, as mentioned previously. The value of this measure exists in the interval (-∞; 1]. The interpretation of important values of this measure is as follows. If $R^2 = 1$, the regression line perfectly fits the data and the model is a perfect fit. If $R^2 \leq 0$, the regression line does not fit the data well and the mode is a poor fit. In this case the mean of the data set used for fitting will provide a better fit than the model. Alternatively, if $0 < R^2 < 1$ the goodness of fit depends on the value of the measure. Values close to 1 indicate a good fit and values close to 0 indicate a poor fit.

Conventionally, the adjusted R^2 value is used in the evaluation of a model, as it accounts for the parsimony of the model. This is done by incorporating the number of parameters required by the model into the calculation of the measure. In most cases this provides a good balance between accuracy and economy of a model. In the case of ANNs the number of parameters does not necessarily increase as the system becomes increasingly complex. Hence, using the adjusted R^2 measure will skew results as only the traditional models will be kept parsimonious. For this reason the R^2 measure is used.

6.2.2 Threshold Value for Effectiveness

An R^2 threshold of 0.3 is used when evaluating the effectiveness of the models. This value is chosen for illustrative purposes. The R^2 value of 0.3 results in the model explaining a fair amount of variation in the data. More research should be done to determine an optimal threshold value.

6.2.3 Results of Experiments

Table 77 and Table 78 present the R^2 values obtained over the testing set from the traditional models and ANNs constructed in chapter 3 and 4, respectively.

1 Step - R-Squared - Testing Set			
	1 Step (Isolated)	**1 Step (Integrated)**	**1 Step (Traditional)**
Inflation	0.161	0.262	0.265
Money Market	0.986	0.929	0.996
Bond Market	0.034	0.024	0.007
Equity Market	0.010	0.029	-0.001

Table 77: R^2 of one step forecasting models

3 Step - R-Squared - Testing Set			
	3 Step (Isolated)	**3 Step (Integrated)**	**3 Step (Traditional)**
Inflation	0.015	0.015	0.118
Money Market	0.874	0.836	0.917
Bond Market	-0.018	0.024	-0.013
Equity Market	0.010	0.021	-0.001

Table 78: R^2 of three step forecasting models

The histogram in Figure 72 plots the R^2 value of the isolated ANNs, integrated ANNs and traditional models used to forecast the return on the money, bond and equity markets, as well as inflation, a period of one and three months into the future. The threshold value of 0.3 is included in the histogram to aid with analysis.

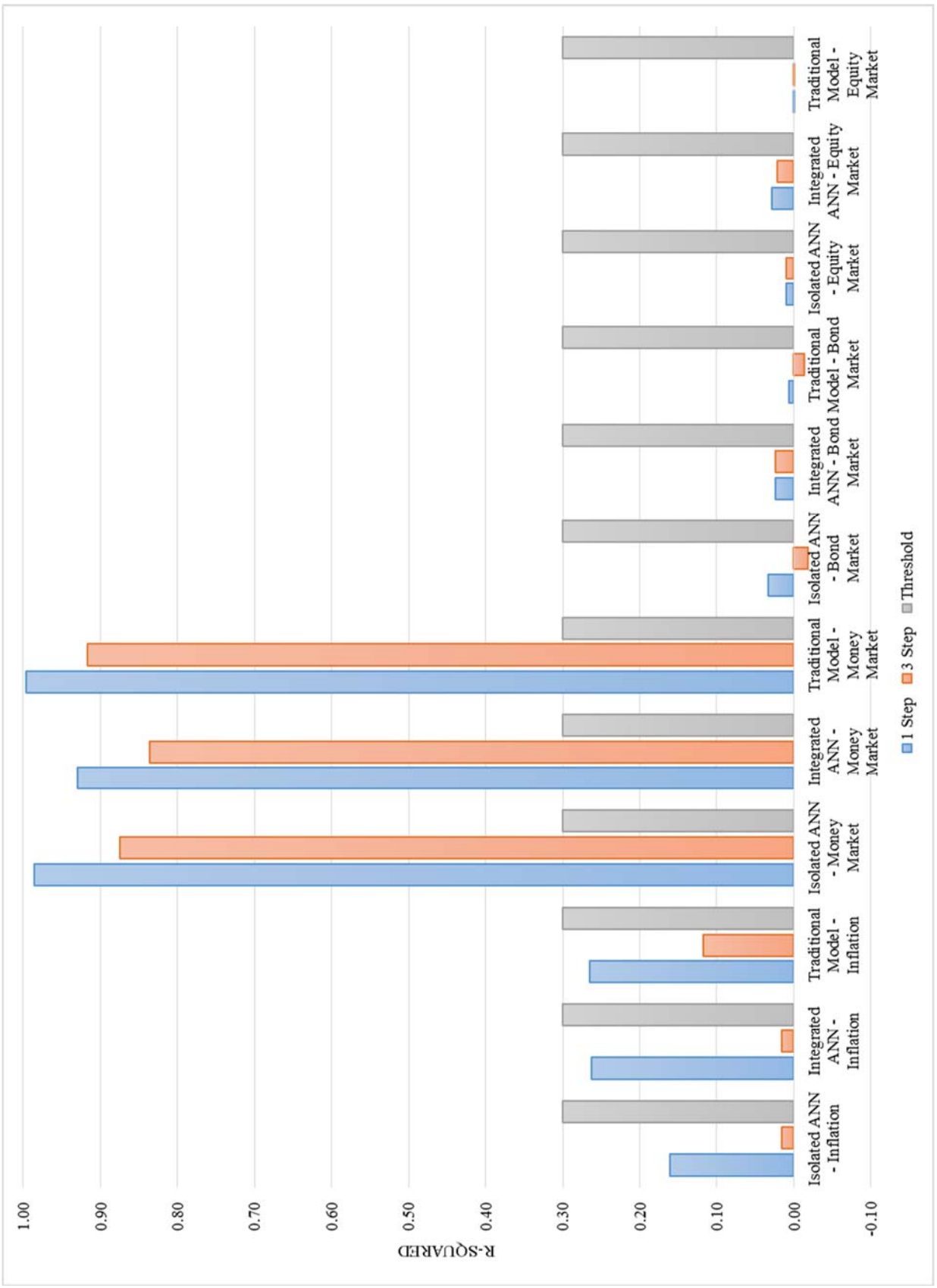

Figure 72: R^2 values of forecasting models – testing data set

6.2.4 Analysis of Inflation Models

Figure 72 indicates that for all the models of inflation the R^2 value is below the threshold for both the one and three-month forecast period. The integrated ANN and traditional model explain approximately 25% of the variation within the inflation data when a single month forecast period is considered. The isolated ANN explains slightly over 15%. This indicates that all the models explain a significant amount of variation within the inflation data when a single month forecast period is considered.

The three-step ANNs produce significantly smaller R^2 values, indicating they are poor at forecasting inflation three months ahead. The traditional model explains slightly over 10% of the variation in the data. The significantly higher R^2 value for the traditional model can be explained by better capturing of the seasonal trend.

Overall, the R^2 values are below the threshold. The ANNs and traditional model explain a fair amount of variation in the inflation data when a single month forecast period is considered. As the forecast period increases the R^2 value decreases considerably for the ANNs and traditional models. Only the traditional model explains a fair amount of variation when a three-month forecast period is considered.

6.2.5 Analysis of the Money Market Models

Figure 72 indicates that the R^2 values of all the money market models are above the threshold for both the one and three-month forecasts. The models explain over 80% of the variation within the money market, which suggests this market can be modelled as a strict time series in the short term. Thus, incorporating external variables into the models will not have a significant effect on the accuracy of the short term forecasts. The high predictability of this market suggests there is little random variation within the data, which is expected as the money market is stable over the short term. Of all the markets considered, the forecasts of the money market have the greatest accuracy.

When considering a one month forecast period all the money market models possess an R^2 value greater than 0.9. This indicates that the models forecast the money market accurately over this period. Further, this suggests the trends within the money market do not change regularly and are not volatile. All the models constructed to forecast the money market, both one and three months into the future, can be used in practise.

6.2.6 Analysis of the Bond Market Models

Figure 72 indicates the R^2 values are significantly smaller than the threshold value in the case of the bond market models. This is true for both the one and three-month forecasts. Negative R^2 values are observed for the isolated ANN and traditional models when the three-month forecasts are considered, which suggests the mean return would have provided a better fit that the models constructed. All the models explain less than 5% of the variation within the bond market. This indicates that none of the models are effective at forecasting the return on the bond market, either one or three months into the future.

These results suggest that the bond market is largely driven by external variables and not previous observations. The accuracy of the models will improve by incorporating these explanatory variables. Without the inclusion of these variables the models developed must not be used in practise.

6.2.7 Analysis of the Equity Market Models

Figure 72 indicates all the equity market models explain less than 5% of the variation within the equity market, which is significantly lower than the threshold of 30%. This observation is similar to that from the bond market models. This poor performance is expected as the equity market is highly volatile and heavily dependent on external factors which are not included in the models. Incorporating these factors will significantly increase the accuracy of the forecasts.

The traditional model explains almost zero variation within the equity market indicating that no relationships between successive observations are detected. The integrated ANN has the greatest R^2 value which indicates there are relationships between the equity and other markets.

None of the models constructed must be used in practise without the inclusion of additional explanatory variables. This applies for both the one and three-month forecast periods.

6.2.8 Conclusion – Effectiveness of Models

The models (both ANNs and traditional models) used to forecast the money market perform well and can be implemented in practise. This holds for both the one and three-month forecast periods. The models used to forecast inflation over a one month period explain a fair amount of variation within the inflation data. These models perform poorly when the forecast period is extended to three months. The models used to forecast the bond and equity market perform poorly over both the one and three-month forecast periods.

6.3 Change in Error from One to Three-Step Forecasts

This section aims to determine if the increase in forecast period causes the forecast accuracy of the models to decline radically. It is expected that as the period extends the error will increase. This is expected because the distant future is less certain than the near future.

To investigate this, the RMSE and the change thereof, over the testing set for the isolated ANNs, integrated ANNs and traditional models is analysed. These investigations are performed for the money, bond and equity markets, as well as inflation.

6.3.1 Results of Experiments

Table 79 and Table 80 provide the RMSE of the traditional models and ANNs constructed in chapters 3 and 4, respectively.

1 Step Models - RMSE - Testing Set			
	1 Step (Isolated)	**1 Step (Integrated)**	**1 Step (Traditional)**
Inflation	4.80E-03	4.50E-03	4.49E-03
Money Market	2.17E-04	4.84E-04	1.17E-04
Bond Market	1.88E-02	1.89E-02	1.91E-02
Equity Market	5.25E-02	5.20E-02	5.28E-02

Table 79: RMSE of one step forecasting models

3 Step Models - RMSE - Testing Set			
	3 Step (Isolated)	**3 Step (Integrated)**	**3 Step (Traditional)**
Inflation	5.20E-03	5.20E-03	4.92E-03
Money Market	6.45E-04	7.36E-04	5.25E-04
Bond Market	1.93E-02	1.89E-02	1.93E-02
Equity Market	5.25E-02	5.22E-02	5.28E-02

Table 80: RMSE of three step forecasting models

The column chart in Figure 73 plots the RMSE of the isolated ANNs, integrated ANNs and traditional models used to forecast the return on the money, bond and equity markets, as well as inflation, a period of one and three months into the future.

6.3.2 Analysis of Inflation Models

Figure 73 indicates that for all the inflation models the RMSE of the forecasts increases as the forecast period expands from one to three months. This is expected as the distant future is less certain than the near future when considering inflation. The greatest increase in RMSE occurs when the integrated ANN is considered and can be explained by the ANN capturing random variation, due to slight over fitting, when the forecast period extends.

The isolated ANN has the smallest increase in RMSE as the forecast period extends. However, this ANN has the greatest RMSE over the one month period and increases to the same level as the integrated ANN over the three-month period. This does not imply the isolated ANN has superior performance over the one or three-month forecast period.

The traditional model handles the extension of the forecast period most efficiently, as it has the smallest RMSE over the three-month period. This is unexpected as the errors in the forecasts associated with the traditional model are expected to compound as the forecast period increases, leading to poor forecasting accuracy. The ANNs do not possess this shortcoming as they are trained to forecast either one or three-months in advance.

6.3.3 Analysis of the Money Market Models

Figure 73 indicates there is a general increase in RMSE as the forecast period extends for the money market models. The increase is significant for all models. The isolated ANN has the greatest increase and the integrated ANN the smallest. The small increase in RMSE shown by the integrated ANN does not imply it is the most stable model because it has the greatest RMSE over both forecast periods. This implies the integrated ANN captures random variation when both periods are considered. The traditional model has a significant increase in RMSE as the forecast period increases; however, it has the smallest RMSE over both the one and three-month forecast periods.

6.3.4 Analysis of the Bond Market

Figure 73 indicates the RMSE of the forecasts associated with the bond market increases for both the isolated ANN and traditional models as the forecast period extends. The increase is lower than expected and is explained by the poor forecasting accuracy of the models associated with the bond market. No increase occurs when the integrated ANN is considered, which is unexpected. Similar errors over the one and three-month periods imply the bond market increases in predictability as the forecast period expands. This is an unlikely result and is likely explained through a combination of poor forecasting accuracy and random variation.

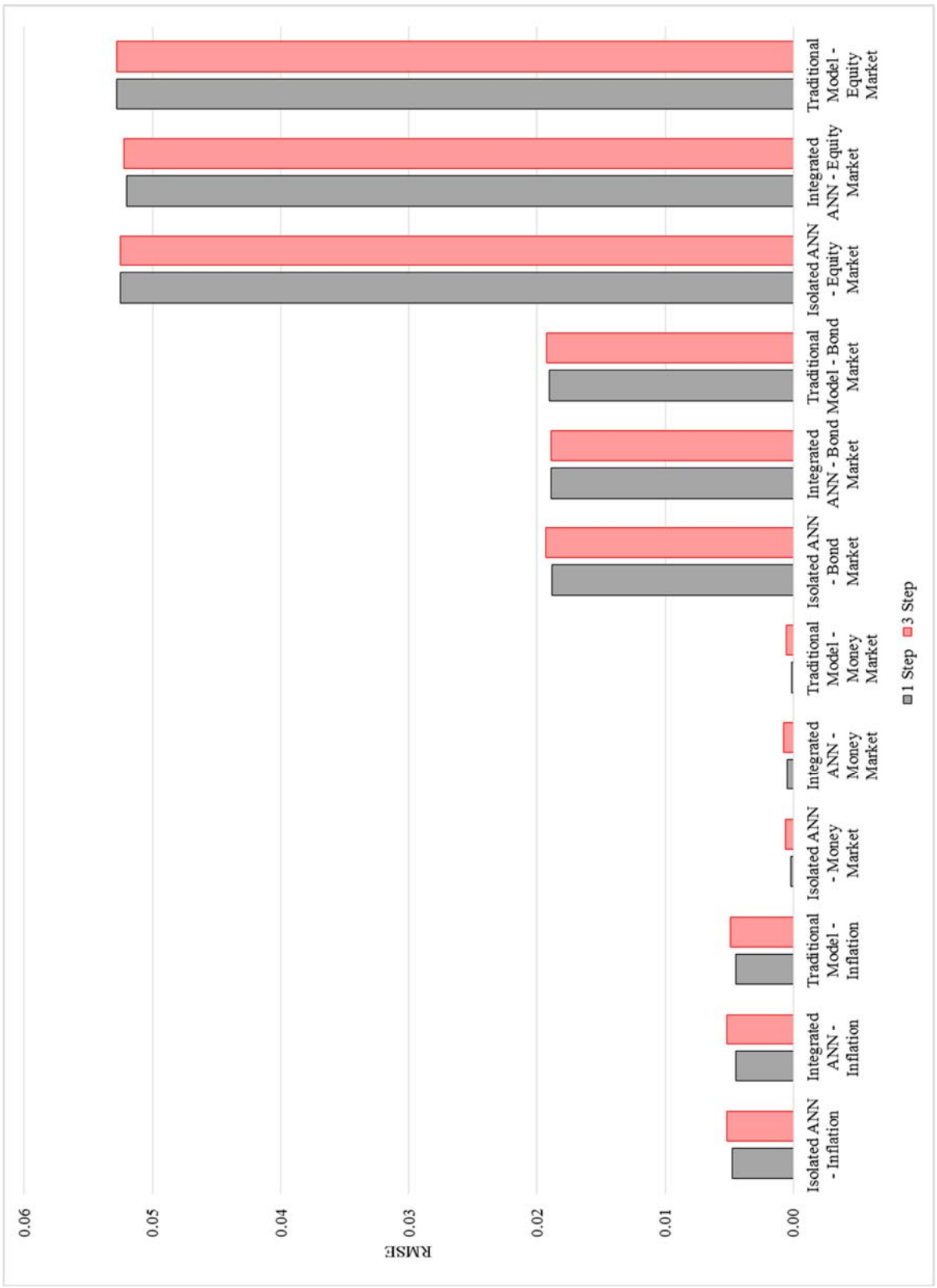

Figure 73: RMSE of forecasting models – testing data set

6.3.5 Analysis of the Equity Market Models

Figure 73 indicates no increase in the RMSE of the forecasts, associated with the equity market, for the isolated ANN and traditional model. Further, the increase in RMSE for the integrated ANN is slight. This implies the equity market can be forecast equally as accurately over one month as it can over three. This is unlikely as the equity market becomes increasingly volatile as the forecast period extends. The observed anomaly can be explained through the poor forecasting accuracy of the models over both one and three-month periods.

6.3.6 Conclusions – Change in Error from One to Three-Step Forecasts

In general, the error of each model increases as the forecast period extends from one to three months. This is expected as the near future is more certain than the distant future. Anomalies arise when the bond and equity markets are considered. These are explained through the poor forecasting accuracy of the models used.

6.4 Isolated and Integrated ANNs

The aim of this section is to compare the performance of the isolated and integrated ANNs constructed for each application. This comparison uses the RMSE as the measure of performance. Further, only the results over the testing set are considered, because possible over fitting will skew results over the training set. The integrated ANNs are expected to perform better than the isolated ANNs as they have access to additional explanatory variables, i.e. the other markets and inflation.

6.4.1 One Step Forecasts

The column chart in Figure 74 plots the RMSE of the forecasts generated by the isolated and integrated ANNs. Forecasts are generated for the return on the money, bond and equity markets as well as inflation, over a forecast period of one month.

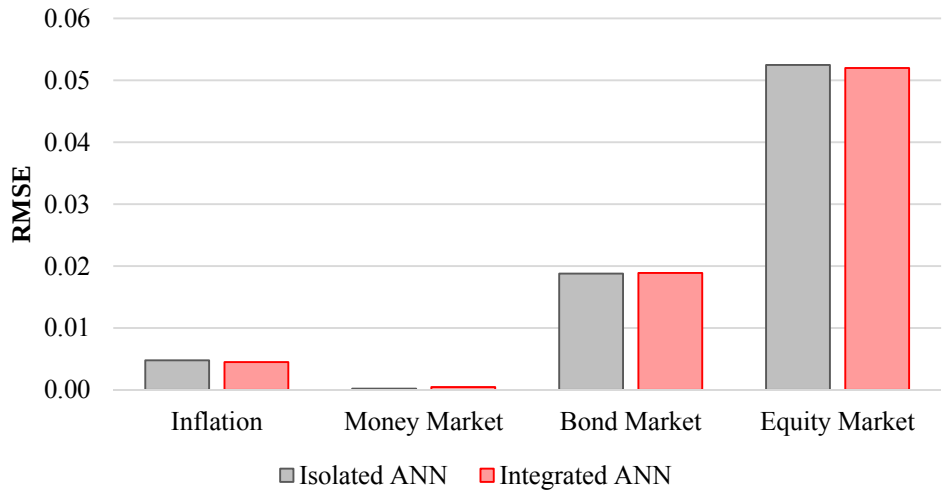

Figure 74: RMSE of isolated and integrated ANNs – one step forecasts

In the case of inflation, the integrated ANN generates one month forecasts that are more accurate than those generated by the isolated ANN. This implies there are relationships between inflation and the markets that can be used to improve the explanation of the variation within the inflation data.

The integrated ANN generates one month forecasts of the money market that are less accurate than those generated by the isolated ANN. This is unexpected as the integrated ANN has additional data that can be used to explain the money market. The complex structure of the integrated ANN may explain this phenomenon as it is more susceptible to over fitting. This results in random variation within the data being captured, leading to a decrease in forecast accuracy.

The RMSE of the forecasts, associated with the bond market, is similar for both the isolated and integrated ANNs. This suggests the relationships between the bond and the other markets do not have a significant impact on the bond market when forecasting one month into the future.

In the case of the equity market, the RMSE of the forecasts generated by the integrated ANN is slightly lower than that generated by the isolated ANN. This is expected as the equity market is affected by supply and demand factors within other markets. However, the effect of these inter-market factors is smaller than expected, which implies the equity market is largely driven by external factors.

6.4.2 Three-Step Forecasts

The column chart in Figure 75 plots the RMSE of the forecasts generated by the isolated and integrated ANNs. Forecasts are generated for the return on the money, bond and equity markets as well as inflation, over a forecast period of three months.

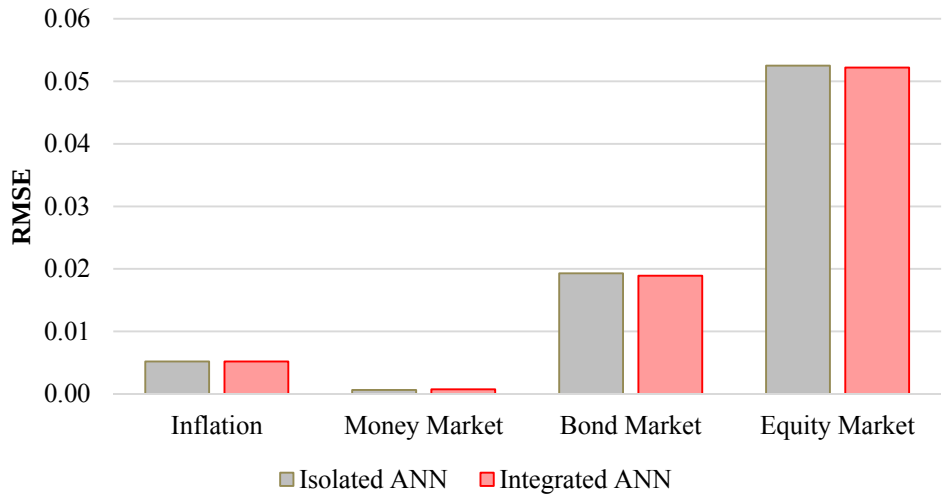

Figure 75: RMSE of isolated and integrated ANNs – three step forecasts

The RMSEs of the inflation forecasting ANNs are identical when forecasting three months ahead. This suggests incorporating relationships between the markets and inflation into the forecasts of inflation add no value when forecasting three months ahead. Further, this implies the relationships between inflation and the markets fade as the forecast period increases.

The RMSE of the money market forecasts generated by the integrated ANN is greater than that of the isolated ANN. This difference is not significant but supports that the return on the money market should be modelled as a strict time series process, and introducing additional variables will not improve short term forecasting accuracy.

The integrated ANN performs better than the isolated ANN when forecasting returns on the bond market three months in advance. This suggests the relationships between the bond and other markets become increasingly important as the forecast period expands.

The equity market forecasts generated by the integrated ANN are more accurate than those generated by the isolated ANN. This implies the relationships between the equity and other markets continue to have an effect on the equity market as the forecast period increases. However, this impact is again smaller than expected and is explained by the equity market being largely driven by external factors.

6.4.3 Conclusion – Isolated and Integrated ANNs

In the case of the one month forecasts, the integrated ANN only generates more accurate forecasts when inflation and the equity market are considered. This is unexpected as the integrated ANN allows interaction between the markets and inflation which should lead to an increase in forecasting accuracy. These observations imply the relationships between the markets and inflation have a smaller effect on forecasting accuracy over a one and three-month period than expected.

In the case of the three-month forecasts, the integrated ANNs increase forecasting accuracy when the bond and equity markets are considered. However, the relationships between the markets and inflation do not add value to the forecasts in the case of inflation and the money market.

In conclusion, the integrated ANNs do not produce significantly more accurate forecasts for each application. This suggests the markets have a smaller influence on each other than expected. The relationships between the markets and inflation have an impact when considering inflation one month ahead, but as the forecast period increases these relationships fade. This is not true for the bond and equity markets as the relationships between the markets become increasingly prominent as the forecast period expands from one to three months.

6.5 ANNs compared to Traditional Models

The aim of this section is to compare the performance of traditional models to ANNs at forecasting returns on the money, bond and equity markets as well as inflation one and three months in advance. The traditional models and ANNs compared are those constructed in chapters 3 and 4, respectively.

It is important that random variation is accounted for when comparing models, as it may skew results. To account for it both the RMSE of the forecasts generated by each model and the resulting residual series are considered. This is done over the testing set of data as over fitting, present in the training set, will skew results.

Comparison intervals are constructed by increasing/decreasing the RMSE of each model by a single standard deviation of the residuals of the forecasts generated by the model. This is explained in detail below.

Let ε_t be the residual of forecast *t*. Then

$$\varepsilon_t = (\hat{X}_t - X_t)$$

where $\hat{X}_t$ is the forecast at time *t* and X_t is the actual value at time *t*. A residual series (ε) is created by the forecasts

$$\varepsilon = (\varepsilon_1, \varepsilon_2, ..., \varepsilon_{100}) .$$

The standard deviation ($\sigma_{residuals}$) of this series is determined by

$$\sigma_{residuals} = \sqrt{\frac{1}{99}\sum_{t=1}^{100}(\varepsilon_t - \bar{\varepsilon})^2}$$

where $\bar{\varepsilon} = \frac{1}{100}\sum_{t=1}^{100}\varepsilon_t$. The comparison interval is calculated as follows

$$[\max(RMSE - \sigma_{residuals}, 0); RMSE + \sigma_{residuals}].$$

Equation 46: Comparison interval for traditional models and ANNs

The lower limit of this interval has a minimum of zero, because the RMSE cannot be smaller than zero. These intervals are determined for all the models considered. An indication that one model outperforms another would be that there is a relatively small overlap between their comparison intervals. This method of comparison takes account of the RMSE of the forecasts and the volatility within the associated residuals.

Rationale for use of comparison intervals

The above method of comparison is used because the underlying distributions of the forecasts, RMSE and forecast residuals are not known with any certainty. Thus, making the assumptions required for the use of more elegant statistical tests, inappropriate. Further, the standard deviation of the residuals is used to increase/decrease the RMSE of the forecasts, instead of the mean error of the residuals, because the mean error of the residuals is close to zero for all cases. This result is a by-product of the model fitting process. It will be necessary to use additional tests in instances where the test described above indicates a significant difference in the performance of the models. If no such instance is found, it is likely that additional tests will not produce different conclusions.

6.5.1 ANNs and Traditional Model Analysis – Inflation

The analysis is first done of the models that forecast inflation one month ahead. This is followed by analysis of the models forecasting inflation three months ahead.

6.5.1.1 One Step Forecasts – Inflation Models

The chart in Figure 76 plots the comparison intervals of the isolated ANN, integrated ANN and traditional model used to forecast inflation one month into the future. On the chart the horizontal line connects the RMSE of each model. The vertical line illustrates the comparison interval for each model. This interpretation holds for similar charts throughout chapter 6.

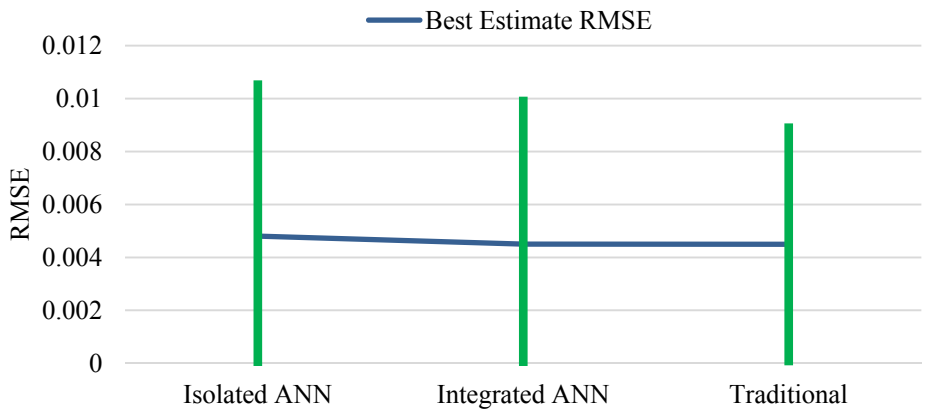

Figure 76: Best estimate and comparison interval for inflation models – one step forecasts

Neither the isolated ANN, nor the integrated ANN, nor the traditional model, is superior when the one month forecasts of inflation are considered. The RMSE of the integrated ANN and traditional model is similar. The isolated ANN achieves the greatest RMSE. Further, the residuals of the isolated ANN contain the greatest volatility, which is not desirable. The traditional model has the smallest volatility within its residuals, as indicated by the smallest comparison interval.

6.5.1.2 Three-Step Forecasts – Inflation Models

No significant difference is found between the performance of the isolated ANN, integrated ANN and traditional model when considering the three-month forecast period.

6.5.2 ANNs and Traditional Model Analysis – Money Market

The models constructed to forecast the money market are compared in this section.

6.5.2.1 One Step Forecasts – Money Market Models

The chart in Figure 77 plots the comparison intervals of the isolated ANN, integrated ANN and traditional model used to forecast return on the money market one month into the future.

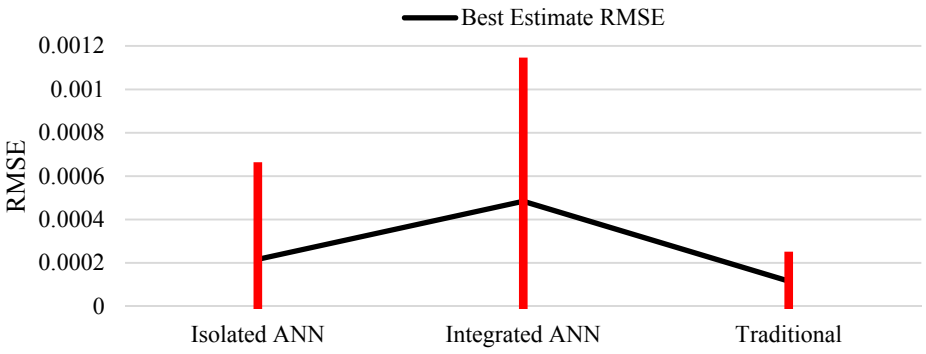

Figure 77: Best estimate and comparison interval for money market models – one step forecasts

The traditional model has both the smallest RMSE and volatility within its residuals. The volatility found within the traditional model's residuals is significantly smaller than those of the isolated and integrated ANN. The integrated ANN has both the greatest RMSE and volatility within its residuals. This suggests the integrated ANN captures the limited random variation present in the data. The isolated ANN has an intermediate RMSE and volatility within its residuals. None of the models can be classified as superior; however, the traditional model results in the smallest RMSE and volatility within its residuals and will be the favoured model when forecasting the money market one month in advance.

6.5.2.2 Three-Step Forecasts – Money Market Models

The chart in Figure 78 plots the comparison intervals of the isolated ANN, integrated ANN and traditional model used to forecast return on the money market three months into the future.

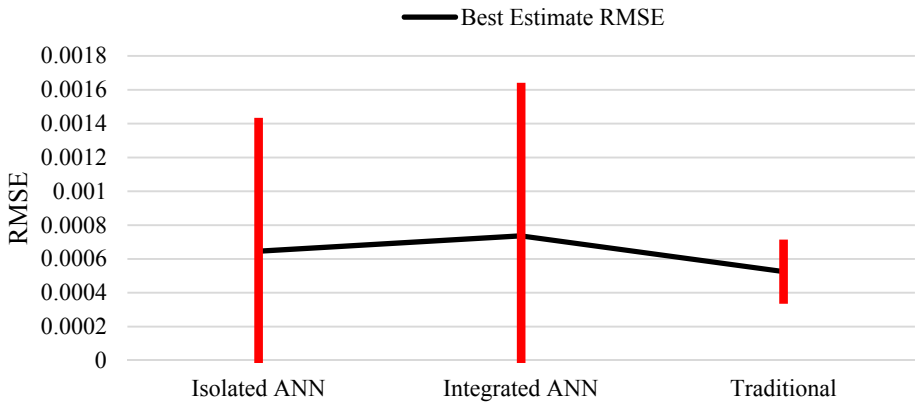

Figure 78: Best estimate and comparison interval for money market models – three step forecasts

The traditional model again achieves the smallest RMSE and volatility within its residuals. The RMSE of the forecasts and volatility within their residuals have increased from the one step forecasts, as expected. The RMSE of the forecasts generated by the isolated ANN is not significantly greater than those by the traditional model. Again, the integrated ANN has both the greatest RMSE and volatility within its residuals. The traditional model is preferred to the ANNs in this case, similar to the one step forecast conclusion.

6.5.2.3 Twelve-Step Forecasts – Money Market Models

The chart in Figure 79 plots the comparison intervals of the isolated ANN, integrated ANN and traditional model used to forecast return on the money market twelve months into the future.

In the case of a twelve-month forecast period the isolated and integrated ANNs have a smaller RMSE than the traditional model. Further, the residuals of the forecasts generated by the ANNs are less volatile than those associated with the traditional model. This suggests that as the forecast period extends the compounding of the errors associated with the forecasts from the traditional model has a significant effect on the model's accuracy. Further, as all the comparison intervals overlap it is not possible to conclude that any model performs significantly better than another. However, the isolated ANN is favoured as it results in the smallest RMSE and least volatility within its residuals.

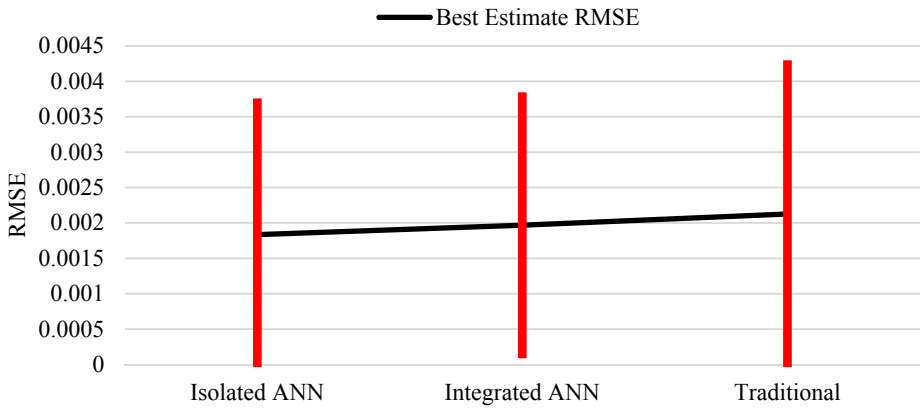

Figure 79: Best estimate and comparison interval for money market models – twelve step forecasts

6.5.3 ANNs and Traditional Model Analysis – Bond Market

The models constructed to forecast the bond market are compared in this section.

6.5.3.1 One Step Forecasts – Bond Market Models

No significant difference is found between the performance of the isolated ANN, integrated ANN and traditional model when considering the one month forecast period.

6.5.3.2 Three-Step Forecasts – Bond Market Models

The chart in Figure 80 plots the comparison intervals of the isolated ANN, integrated ANN and traditional model used to forecast return on the bond market three months into the future.

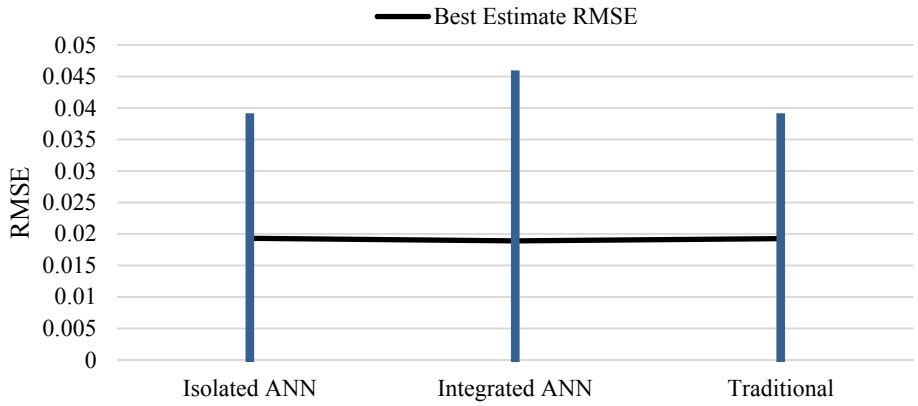

Figure 80: Best estimate and comparison interval for bond market models – three step forecasts

The forecasts generated by the integrated ANN have the smallest RMSE of all three models; however, the residuals of this model have the greatest volatility. The isolated ANN and traditional model have a similar RMSE and volatility within their residuals. Further, these residuals are less volatile than those of the integrated ANN. No model is significantly superior to the others when forecasting the returns on the bond market three months in advance.

6.5.4 ANNs and Traditional Model Analysis – Equity Market

The models constructed to forecast the equity market are compared in this section.

6.5.4.1 One Step Forecasts – Equity Market Models

The chart in Figure 81 plots the comparison intervals of the isolated ANN, integrated ANN and traditional model used to forecast return on the equity market a single month into the future.

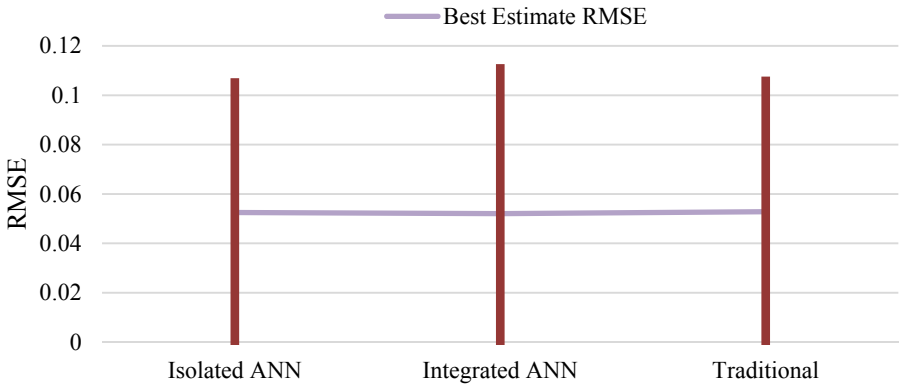

Figure 81: Best estimate and comparison interval for equity market models – one step forecasts

The forecasts generated by the integrated ANN achieve the smallest RMSE. The improvement of the RMSE over the other models is not significant as all models explain less than 5% of the variation within the equity market. The residuals of the forecasts generated by the integrated ANN contain the greatest volatility of all three models. The volatility within the residuals of the isolated ANN and traditional model is similar. None of the models significantly outperforms the others when forecasting the equity market one month in advance.

6.5.4.2 Three-Step Forecasts – Equity Market Models

No significant difference is found between the performance of the isolated ANN, integrated ANN and traditional model when considering the three-month forecast period.

6.5.5 ANNs and Traditional Model Analysis – Conclusions

In general no model proves significantly superior to the others. The expectation of significantly improved performance by the integrated ANNs due to the allowance of inter-market relationships is not realised. There are slight effects on forecasting accuracy from the relationships between the markets and inflation. However, these effects are smaller than expected implying relationships between the markets and inflation are not large drivers of any market or inflation. When forecasting the money market over a period of one and three months the traditional model is preferred over the ANNs because of the smaller RMSE and volatility within the traditional model's residuals. The opposite is true when forecasting the money market twelve months ahead. This occurs because the forecasting error associated with the traditional model begins to compound as the forecast period extends. This leads to reduced accuracy and increased volatility within the traditional model's residuals when a forecast period of twelve months is considered. It can be concluded that ANNs, both isolated and integrated, perform as well as traditional models when forecasting the return on the money, bond and equity markets as well as inflation.

6.6 Training and Testing Data Set Error Anomaly

It is found, in general, that the Root Mean Squared Error (RMSE) over the testing set is smaller than over the training set. This result implies the forecasts are more accurate over the testing set than the set used to build the models, which is unexpected. This anomaly occurs for both the traditional models and ANNs.

The observed anomaly is explained through the increased variation contained in the training set. Removing this variation, from the data set, leads to the error over the training set being smaller than over the testing set. This is in line with expectations. An example of this anomaly and explanation is presented below.

6.6.1 One Step Isolated Inflation ANN

The error over the training and testing sets, during the training process, is plotted below in Figure 82. In this case, outliers and the resulting increased variation are present in the training set. Figure 83 is an identical plot to that of Figure 82, however, the outlier within the training set have been removed.

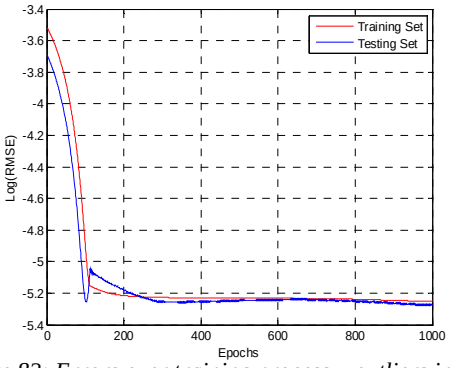

Figure 82: Errors over training process – outliers included in training data

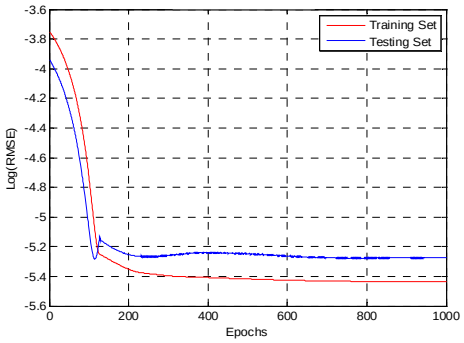

Figure 83: Errors over training process – outliers removed from training data

The error over the training set is generally higher than over the testing set when outliers are present in the data, as indicated by Figure 82. Once several outliers are removed from the training set, and the variation within the training and testing set is aligned, the error over the testing set is generally greater than that over the training set, as indicated by Figure 83. Hence, the error anomaly is explained through the additional variation present in the training data set.

6.7 Cross Validation of Models

To compare the general forecasting ability of ANNs and traditional models it is important to consider several different data sets. This allows the average forecasting ability of each model to be determined and compared. In this section the one month inflation forecast models are considered. For additional details of this experiment refer to chapter 5.

To be statistically certain that a model outperforms another it is necessary to construct a confidence interval. These intervals are compared and a model is regarded as statistically superior to another if the upper limit of its confidence interval is smaller than another model's lower limit. The confidence intervals are constructed using the RMSE of each model over five different sub-divisions of the inflation data set. The mean and standard deviation of the RMSE are calculated. The intervals are determined by increasing/decreasing the mean RMSE by the standard deviation of the RMSE of each model.

The interval construction process is described by

$$\overline{RMSE}_i = \frac{1}{5}\sum_{t=1}^{5} RMSE_i(t)$$

where $RMSE_i(t)$ is the RMSE of model *i* for data set *t*. The standard deviation of the RMSE for model *i* is determined by

$$\sigma_{RMSE_i} = \sqrt{\frac{1}{4}\sum_{t=1}^{5}(RMSE_i(t) - \overline{RMSE}_i)^2}\ .$$

The confidence interval is given as

$$[\overline{RMSE}_i - \sigma_{RMSE_i}; \overline{RMSE}_i + \sigma_{RMSE_i}].$$

Equation 47: Confidence interval for RMSE

6.7.1 Analysis of Cross Validation of Inflation Models – 1 Step Forecasts

The chart in Figure 84 plots the confidence interval (Equation 47) for the isolated ANN, integrated ANN and traditional model used to forecast inflation one month in advance. On the chart, the horizontal line connects the average RMSE and helps with analysis. The vertical lines represent the confidence intervals.

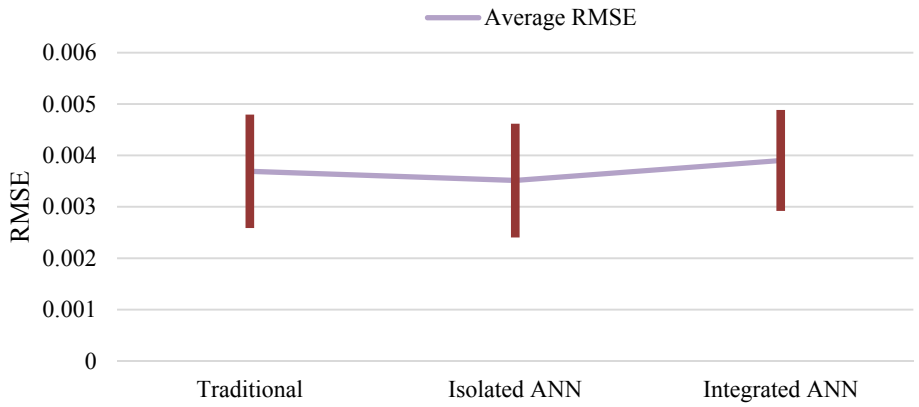

Figure 84: Confidence intervals of RMSEs of models over five different data sets – one step inflation forecast models

The isolated ANN generates forecasts which on average are more accurate than those generated by any other model. The forecasts of the integrated ANN produce the greatest RMSE, on average over the five data sets considered. As all the confidence intervals overlap no model can be concluded to be statistically superior in the case of forecasting inflation one month in advance.

6.7.2 Conclusions of Cross Validation of Inflation Models – One Step Forecasts

No model is statistically superior to the others in the case of forecasting inflation a single month in advance. The integrated ANN is expected to outperform the other models as it allows interaction between the markets and inflation. These interactions are less influential than expected and cause additional random variation to be captured by the ANN. The isolated ANN performs on average better than the other models. The variation of the RMSE is similar for all three models. These conclusions are based on the results from five different training and testing data sets. For the conclusions to hold increased statistical credibility it is necessary to consider more than five sets of data.

6.8 Hybrid Models

This section aims to determine whether ANNs add value to traditional models when combined. This is done through the use of hybrid models, which are constructed in chapter 5. These models are constructed for inflation and the money market over forecast periods of one and three months. The hybrid models are compared to traditional models through the use of comparison intervals. Refer to section 6.5 for the method behind the determination of these intervals. The charts below possess an identical structure to those analysed in section 6.5 and the details are not repeated here.

This section is important as industry currently uses traditional models. Thus, ANNs will need to be incorporated into these models when applied in practical settings.

6.8.1 Inflation Hybrid Models

The hybrid and traditional models constructed to forecast inflation are compared in this section.

6.8.1.1 Inflation Hybrid Models – One Step Forecasts

The chart in Figure 85 plots the comparison interval of the hybrid model with an isolated ANN, hybrid model with an integrated ANN and traditional model used to forecast inflation one month into the future.

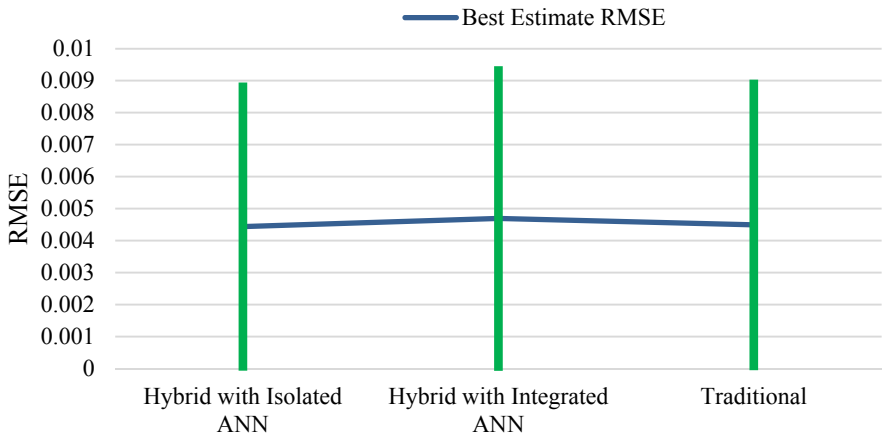

Figure 85: Best estimate and comparison interval of RMSE for hybrid inflation models – one step forecasts

There is a slight difference between the RMSE and volatility within the residuals of the models considered. The smallest RMSE is achieved by the hybrid model with an isolated ANN. This model also has slightly less volatility within its residuals than the other models. The hybrid model with an integrated ANN has the greatest RMSE and volatility within its residuals, which indicates it captures random variation within the data. No model significantly outperforms the others in this application. However, the lower RMSE achieved using the hybrid model with an isolated ANN suggests that ANNs can add value to traditional models when forecasting inflation a period of one month ahead.

6.8.1.2 Inflation Hybrid Models – Three-Step Forecasts

The chart in Figure 86 plots the comparison interval of the hybrid model with an isolated ANN, hybrid model with an integrated ANN and traditional model used to forecast inflation three months into the future.

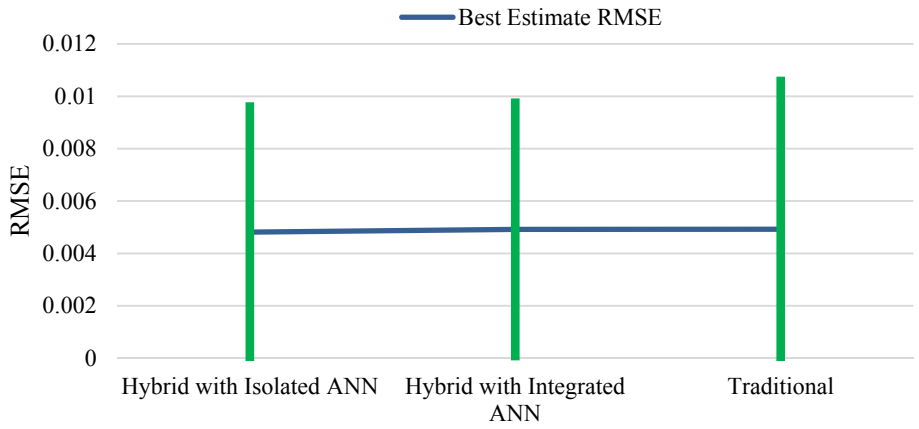

Figure 86: Best estimate and comparison interval of RMSE for hybrid inflation models – three step forecasts

The models have a similar RMSE when forecasting inflation three months ahead. The volatility within the residuals of the traditional model is the greatest. The hybrid models have similar volatility in their residuals. This suggests the hybrid models result in less volatile forecasts when forecasting inflation three months in advance. For this analysis it is not possible to conclude that any model is statistically more accurate than another at forecasting inflation three months in advance. However, since the volatility of the residuals from the hybrid models is smaller than the traditional model, the ANNs are likely to add value to the traditional models.

6.8.2 Money Market Hybrid Models

The hybrid and traditional models constructed to forecast the return on the money market are compared in this section.

6.8.2.1 Money Market Hybrid Models – One Step Forecasts

No significant difference is found between the performance of the hybrid models and traditional model when considering the one month forecast period.

6.8.2.2 Money Market Hybrid Models – Three-Step Forecasts

The chart in Figure 87 plots the comparison interval of the hybrid model with an isolated ANN, hybrid model with an integrated ANN and traditional model used to forecast the return on the money market three months into the future.

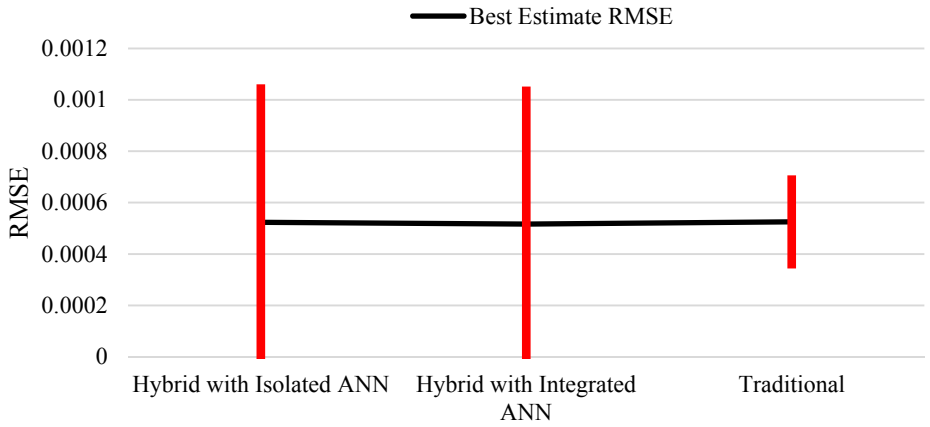

Figure 87: Best estimate and comparison interval for hybrid money market models – three step forecasts

The RMSE of all three models are similar. The hybrid model with an integrated ANN achieves a slightly lower RMSE than the other models. However, the residuals from the traditional model have the least volatility of all the models considered.

No model significantly outperformed the others when forecasting the return on the money market three months into the future. However, the traditional model contains the least volatility within its residuals while maintaining a similar RMSE as the other models, making it the favoured approach.

6.8.3 Conclusions – Hybrid Models

No model type can be classified as the statistically superior type. The hybrid model is favoured when forecasting inflation, because of the reduction in volatility within the residuals of the forecasts generated by this model. It can be concluded that ANNs are likely to add value to traditional models, through hybrid models, when forecasting inflation. The traditional model is preferred when forecasting the return on the money market as the volatility within its residuals is the smallest.

6.9 Concluding Remarks

An efficient method for building forecasting systems using ANNs is presented. Further, an efficient method of parameter estimation is explained. This methodology leads to efficient ANN forecasting systems used to forecast monthly inflation, as well as the monthly return on the money, bond and equity markets both one and three months in advance.

When considering the one month forecast period, only the forecasts generated for inflation and the money market can be considered for practical use. In the case of the three-month forecast period, only the forecasts generated for the money market can be used in practise. The bond and equity market forecasts generated by all the models are poor. This is expected as important factors that drive the bond and equity market are not included in the models.

In general, the forecasting accuracy associated with each market decreases as the forecast period expands. As seen in the bond and equity markets, this general expectation can be distorted by random variation.

The integrated ANN results in a slightly smaller RMSE and volatility within the forecast's residuals than the isolated ANNs when forecasting inflation and the return on the equity market. The opposite is true when the money and bond

markets are considered. The difference in performance of these models is slight and cannot be generalised. As the integrated ANNs do not significantly outperform the other models, the relationships between the markets and inflation have a smaller impact on the markets (or inflation) than expected.

When comparing the models, it is not possible to conclude that any model significantly outperforms the others. Only the money market results in a preference of the traditional model when one and three months are forecast. When considering the twelve-month forecasts of the money market the ANNs are favoured. This suggests that ANNs perform as well as traditional models when forecasting inflation and the return on the bond and equity markets in South Africa over a period of one and three months. Traditional models are preferred when forecasting short term returns on the money market, but as the forecast period extends ANNs become favoured.

The general performance of ANNs is not significantly worse or better than traditional models when forecasting inflation over a single month. This is suggested by the results obtained from applying the models to different subsets of inflation data. Further, ANNs add value to traditional models (through hybrid models) when forecasting inflation. This is not necessarily true when forecasting the return on the money market.

There are several implementation considerations of ANNs which make them advantageous over traditional models. These qualities are the:

1. Ease of implementation and integration into existing models.
2. Ability to deal with random variation within the data sets.
3. Versatility and ability to be applied to different types of problems.
4. Ease of updating.
5. Ability to deal with a large number of inputs without leading to complex parameter estimations.
6. Robustness in terms of forecasting accuracy.
7. Ability to be applied in any computing platform.

In addition to the advantages, ANNs possess disadvantages. The main areas being:

1. The requirement for significant computing time for large data sets,
2. The lack of developed theoretical optimisation techniques,
3. The possibility of over fitting,
4. A lack of understanding of the internal workings of ANNs,
5. The difficulty of understanding the relationship between the inputs and outputs, and
6. The requirement of significant computational power for large data sets.

6.10 Answer to Research Questions

1. *How to build and optimise ANN structures for modelling financial data.*

   This study provides a general methodology for constructing ANNs to model financial data. The methodology developed is applicable in various circumstances. However, many modelling questions still remain (refer to chapter 7).

2. *Can monthly inflation or return in the money, bond or equity markets in South Africa be forecast accurately using a pure time series approach?*

   Inflation and the money market can be forecast efficiently over a one month period. As the period extends to three months only the money market can be forecast accurately. Neither the bond nor equity markets can be forecast without the required external explanatory variables.

3. *Does the accuracy associated with forecasts decrease as the forecast period expands?*

   In general, the forecasting accuracy decreases as the forecast period expands. However, there are suggestions from the twelve-month forecasts of the money market that as the forecast period continues to extend, the accuracy of the ANNs increases relative to that of traditional models.

4. *How do ANNs compare to traditional models when forecasting time series?*

   On the comparison of ANNs to traditional models no general conclusion, as to which model is superior for all the applications considered, can be made. This implies that ANNs perform as well as traditional models when forecasting financial markets in South Africa. The preference of ANNs or traditional models will be determined by the specific problem faced by the user. Further, ANNs possess a range of implementation advantages that must be considered when deciding between models.

5. *Are the inter-market relationships significant when forecasting inflation and/or the money, bond and equity markets over a one and three-month forecast period?*

   The integrated ANNs indicate the existence of relationships between markets over a one and three-month forecast period. However they suggest that the relationships between markets have a limited effect on the future values of the market under consideration. When forecasting inflation the relationships between inflation and the markets fade as the forecast period expands. This is not true when the bond and equity markets are considered.

6. *Does the combination of ANNs and traditional models (hybrid models) add value to traditional models?*

   ANNs add value to traditional models, through hybrid models, when forecasting inflation. This is not necessarily true when considering the money market. Additional investigations must be carried out to increase the clarity of the effect of hybrid models on forecasting accuracy.

7 Future Work

This chapter discusses model shortfalls, possible future work and future investigations.

7.1 Hidden Layers

This study only considers Multi-layered Perceptron Artificial Neural Networks with a single hidden layer. Introducing additional hidden layers will improve the efficiency of the networks and reduce training time. Further, additional hidden layers will result in simplified ANN structures, leading to an improvement in generalisation ability.

However, introducing additional layers will increase the difficulty of estimating the required parameters. Further, the optimal number of hidden layers will be difficult to determine. If the parameters are estimated in the same manner as in this study, adding layers will increase computing time and requirements. Model risk will also be increased by introducing additional layers, due to increased coding complexity.

The optimal number of hidden layers can be estimated using the same process as in this study. The number of hidden layers tested can be varied in an exponential manner. This will produce an interval of hidden layers, in which the optimal number exists. Further simulations can be run considering only the number of layers within the determined interval. This will reduce the size of the interval, until only the optimal number of hidden layers remains.

The optimal number of hidden layers will be small in most cases. A recommended beginning interval of hidden layers is [1; 10] as no applications found exceed this maximum.

Each hidden layer requires an estimation of hidden neurons for that layer. This will lead to an increase in parameter estimation complexity. Fixing the number of hidden neurons per layer to one and varying the number of hidden layers is a possible method to reduce the complexity. This may result in the number of hidden layers extending beyond the interval specified above. Making this decision will simplify the estimation process as the hidden neuron parameters will not require estimation. It is important that tests be done to determine whether this decision will affect the accuracy of the ANN.

7.2 Learning Algorithms

It is important to determine the most effective learning algorithm, as it will minimise training time and increase system accuracy. Traditional Back Propagation may result in the minimum model error not being achieved.

More advanced algorithms will increase training speed and accuracy. These will, however, be more complex to implement and may require large amounts of data to be stored. For example, Jacobian matrices may be required for some second order algorithms, which restricts the number of inputs the ANN can deal efficiently with.

Further, most learning algorithms do not extend to more complex structures, such as the Fully Connected Cascade ANNs. Alternative algorithms must be implemented for these more efficient structures, such as neuron by neuron learning.

A problem associated with most learning algorithms is the possibility of the system's error settling in a local minimum. This risk is reduced by training several ANNs which have randomly initialized weights. Considering many ANNs, initialized at different points on the error surface, increases the probability of selecting the ANN that achieves the global minimum error. This method is used in the applications presented in this study. However, a larger number of initializations should be considered to ensure the global minimum is achieved.

7.3 Data Improvements

7.3.1 Additional Data

Forecasting accuracy will be improved by collecting additional data pertaining to external explanatory variables and incorporating it into the models. This is especially true for the bond and equity market models.

Three years of data (2011, 2012 and 2013) are outstanding and should be collected. The forecasts over this period can be evaluated against actual readings. This data can be used as a validation set, which will produce an independent error measure for use in the model evaluation process.

Alternatively, this data can be incorporated into the training and testing data sets to produce forecasts which are ready for practical use.

Forecasts can be generated by the constructed models for this outstanding period. This would allow comparison between the forecast and actual indices, which would indicate whether underlying trends in the data have changed from 2010 to 2013.

Overseas market indices data can be obtained and presented to the models, resulting in a global forecasting model. This can be used to generate forecasts for a specified index in a specific country. Further, market data from developing countries can be presented to the models, resulting in a forecasting system for emerging markets.

Readings from different indices can be collected and included in the data. This may provide a better overall forecast of the markets.

There may be recording errors in the data as there is no information to validate the data against. Independent validation may be required.

7.3.2 Training and Testing Data Set Size

There is little study found on the optimal split of data between the training and testing sets. In the applications, the training set consists of the first 322 observations and the testing set consists of latest 100 observations. This is in line with previous applications of ANNs.

If the data is split heavily in favour of the training set it will be difficult to determine when the trends underlying the data are captured, because the testing set will contain a small number of the trends in the data. This will lead to inefficient training and a poor forecasting system. Alternatively, favouring the testing set heavily will result in the training set containing insufficient trends underlying the data. This will cause the ANN to inefficiently learn the trends in the data, resulting in poor forecasting accuracy.

Splitting the data sets using a stochastic method will result in various sizes of training and testing sets being presented to the ANN, which allows the optimal size to be determined. Unfortunately, it is likely that each practical application will have a different optimal split and a general rule may not exist.

The training and testing sets can be constructed through a random selection process. The data points in each set can be a random combination of data points from the original data set. This will allow trends present in the latest observations to be captured by the ANN through the training data set, resulting in increased forecast accuracy. However, the data points will not be independent in each set and may skew the results.

Random blocks of independent observations can be used to construct the training and testing data sets. This will allow the most recent observations to be included in the training set and remain independent from the testing set.

For practical use it will be necessary to reduce the testing set substantially as the maximum number of trends must be captured by the ANN through the training set. Large testing sets remove significant amounts of data from the training set. This reduces the number of trends captured by the ANNs.

7.3.3 Change in Training Data Set Size with Lags

For each application the number of observed readings of each index is 432. The testing set consists of 100 data points and the training set consists of the remaining points.

The ANNs and traditional models use a certain number of past observations in the forecasting process. As the number of past observations, used in the models, increases the number of data points in the training data set decreases.

For example, if an ANN consists of 36 input neurons, the number of data points the system can use during training for a one month forecast period is 295. Models that are different in structure have a different number of data points in the training set. This creates an inconsistency when training or fitting the models and must be addressed.

7.3.4 Scaling the Data

The scaling method used divides all the observations by the maximum absolute observation. This scales all the observations to the interval [-1; 1]. Further, the activation functions used in the ANN limit the output of the ANN to the interval (-1; 1) which leads to a shortcoming. The forecasts generated by the ANN in each application are restricted to the interval [-absolute(maximum); absolute(maximum)], which implies that no forecast value can be greater than the greatest absolute observed value.

An appropriate scaling technique must be devised which does not limit the output in the same way as above. Alternatively, activation functions which are not restricted can be used.

7.3.5 Validation Data Set

Only training and testing data sets are used during the construction and testing of the models. This ensures consistency between the different development processes. An additional data set must be constructed for validation of the models.

Ideally the training, testing and validation data sets must be independent and created stochastically. This will generate a multivariate distribution of errors, which can be used to determine whether ANNs are statistically superior to traditional models over each set of data.

7.3.6 Testing and Training Data Set Error Anomaly

A point of interest in the results of the experiments is that the RMSE over the training set is generally higher than over the testing set. This is unexpected, but is explained through a higher proportion of outliers present in the training set (refer to section 6.6). Removing the outliers result in a larger error over the testing set than over the training set.

A decision as to the inclusion or exclusion of outliers in the training and testing sets must be made. Their removal will likely increase the accuracy of the models; however, it may not give an accurate representation of reality. It is possible to remove the outliers from the training set, in order to allow the ANNs to easily determine the trends in the data. Retaining the outliers in the testing set will allow the comparison of the forecasts with actual experience.

Further, if outliers are to be removed, a clear definition of outliers must be determined. This will be challenging as the nature of outliers change over time. For example, an outlier over a certain period may not be classified as an outlier over another period.

7.3.7 Period of Data

The period over which data is considered must be tested. To forecast markets over a specific period it is necessary to use data from similar financial environments in the past, as the underlying relationships will be similar in common scenarios.

For current short term trends to be determined and forecast, a data set that consists of recent short term observations must be used. However, it is likely there will be few observations in both the training and testing data sets. This will lead to a limited number of underlying trends being represented in the data. Further, not including data that contain long term trends will result in long term trends not being forecast by the models.

Considering a large amount of past data will result in a good representation of long term trends. However, excess volumes of past data will lead to the detection of long term trends that are no longer valid.

7.4 Classification Problems

ANNs should be applied and tested on classification problems to determine the extent of their classification ability. Possible applications include:

1. Detecting fraudulent transactions or claims.
2. Determining customers with a high probability of purchasing a specific good.
3. Determining efficient employees.

The classification problem can be extended to determine whether a data point belongs to one of several categories. This can be achieved by equating the number of output neurons to the number of categories. Hence, each output neuron will represent a specific category. As a data point is presented to the system it will be classified in one of the possible categories. For example, if a data point belongs to category B a 1 will be output at the output neuron which represents this category. The other categories will have an output of 0.

Continuous activation functions will be used in this discrete classification application, which presents an inconsistency. A threshold value, above which the output is 1 and below which the output it 0, will need to be determined. This will require estimation of another parameter, which can be done by varying the threshold until the system's error is minimised.

A further possible problem is that two categories can result in an output above the chosen threshold value, causing more than a single output neuron to be active. This possibility will be problem specific and can be solved by specifying selection rules in the ANN.

7.5 Weight Initialization

The weights are initialized using a normal distribution with a mean of 0 and a standard deviation of 0.001. This is chosen as the weights in large ANNs are expected to be small. The weights are initialized randomly to reduce the risk of the system's error settling in a local minimum.

Weight initialization has a significant effect on the training time and optimal solution of the ANN. To ensure the optimal solution is obtained within the shortest learning period the weights must be initialized near the optimal ones. This is difficult to achieve as the error surface of the system is unknown.

Little study is found on the optimal initial weights. To investigate this, it will be necessary to initialize the structure several thousand times and train it to an efficient level. Once the distribution of the error has been determined for several practical problems, it can be analysed and any trends in the weight initialization identified. This can be used to determine optimal initial weights. The process will be time consuming, however, once the code is vectorised (see section 7.25) and a more efficient learning algorithm is implemented, will be feasible.

7.6 Initial Learning Rates

Similarly, to the weight initialization case, the learning rates are initialized randomly when using the RPROP learning algorithm. The optimal initial learning rates will result in faster and more efficient learning. As the learning rates may consist of any real number, a theoretical method must be developed to determine the optimal initial values. A simulation approach will provide efficient initial learning rates; however, it will not likely be optimal unless a vast number of simulations are used.

7.7 Increase the Number of Initializations

As the ANNs are randomly initialized on the error surface, an increased number of initializations will aid the model in finding the global minimum. However, increasing the number of initializations of a large ANN will lead to significantly increased computing requirements and training time. If learning is made more efficient, increasing the number of initializations will be feasible.

7.8 Online vs Batch Learning

The sensitivity of an ANN's results to online and batch learning must be determined. This can be done by training the ANN using an online and batch learning method. The resultant errors and training time must be compared, which will indicate whether batch or online learning is more efficient. It is likely that online learning will result in greater forecasting accuracy but require significantly more training time.

It is important to determine the sensitivity of the ANN's results to different batch sizes, during batch learning. Ideally, batches must be created stochastically, which will result in the ANN being exposed to all the possible sub combination of batches. The resulting distribution can be analysed and the optimal number of batches determined. Increasing the number of batches will likely increase the forecasting accuracy and generalisation ability of the ANNs, but will also increase training time.

7.9 Different ANN Structures

The Multi-layered Perceptron with a single hidden layer is the simplest ANN structure. Other structures such as the Fully Connected Cascade ANN, mentioned in chapter 2, are more complex but result in simpler and more efficient models. However, the learning algorithms of these structures become increasingly complex and increase model risk.

To improve the performance of ANNs it is important that these structures are investigated.

7.10 Explanatory Variables

To improve forecasting accuracy, significant external explanatory variables should be added to the models. These factors can be identified by collecting data on factors that are intuitively linked to each market and testing for the significance of the correlation between them and the markets. These must then be factored into the models.

7.11 More Combinations of Hidden and Input Neurons

Only six combinations of input and hidden neurons are considered in the applications. Ideally, every possible combination of input and hidden neurons should be tested. This is not practically achievable as there are infinitely many possible combinations.

A method of determining the optimal combination of input and hidden neurons is through successive simulations on each application. It is important to determine an upper limit of each parameter. This will then create an interval, within which the optimal parameter estimate exists. A simulation approach can be used, as done in this study, by varying the parameter in an exponential manner, to determine a smaller interval in which the optimal parameter exists. Another simulation can be used to further reduce this interval. This process can be repeated until the interval contains only the optimal parameter estimate. For large ANNs this process will take a significant amount of time.

A timesaving approach would be a theoretical one. Each practical problem has an error surface which has a global minimum. A method must be developed to determine the global minimum point in terms of the input and hidden neurons. This is a challenging approach to develop as each practical problem has a unique error surface.

Further, additional hidden layers will increase the complexity of the parameter estimation process.

7.12 More combinations of Learning Rate Changes

Only three different learning rate increases and decreases are considered in the experiments. It is necessary to consider far more combinations in order to determine the optimal combination of the changes.

Further, learning rate changes should adapt to the specific application to which they are applied. This can be done by linking the changes to the size of the model error, and the change thereof, during training.

7.13 Integrated ANN Input Neurons

The integrated ANNs use the same number of lagged observations from each data set. This is done in order to reduce complexity when estimating the parameters. The integrated ANNs assume the same number of past observations from each market is equally significant, which is unlikely as each market will influence the other to a greater or lesser extent.

If the number of input neurons from each data set is allowed to vary, the parameter estimation process becomes complex and requires significant computational power. A possible solution to this will be to determine the correlation of each past observation to the considered index. This will give a good indication of the required number of lagged observations from each data set. These can be fixed and the number of hidden neurons and/or layers varied.

7.14 Forecast Period

The applications in this study only consider two forecast periods. It will be beneficial to extend the number of forecast periods beyond three months. As the period increases the ANNs are expected to become more effective in comparison to traditional models. This is due to the increased compounding of forecast errors found in traditional models.

7.15 Monthly vs Annual Forecasts

It will be useful to consider other periods of forecast to that of a single month. For example, a twelve month period can be forecast. This can then be compared to the combination of twelve, one month forecasts to determine which method better captures the underlying trends in the data. Less noise will be associated with the annual returns than the monthly returns.

Further, varying the periods will allow the user to specify whether short or long term trends are to be forecast by the ANN. Long forecast periods will result in the ANN capturing and projecting long term trends. Short forecast periods will project short term trends.

7.16 Criteria of Efficient Models

In each case the efficient ANN is chosen from several different initializations by considering the sum of the MSEs from the testing set, over the training process. There may be an ANN which achieve a lower minimum error. However, due to later increases and over fitting they are not selected as the optimal models. Further, this method of selection results in the fitting process not being independent of the testing data set.

Selecting the ANN which results in strictly the minimum error over the testing set will result in the ANN being fitted to the testing set. Further, a selection criterion which considers a single point of the error will expose the selection process to the risk of random fluctuations, influencing decisions.

If only the MSE over the training sets is considered, the system will be over fitted to the data, resulting in poor generalisation and forecasting accuracy.

Investigations into the most efficient selection criteria must be performed as this is currently subjective in nature.

7.17 Error Measures

ANNs are fitted by considering the error over both training and testing data sets, which leads to subjectivity as to when the training process must end. It is necessary to determine an error measure which combines the errors over the data sets. This will improve the consistency and efficiency of the training process. It is important that the error over the testing set is weighted heavier than over the training set in order to avoid over fitting.

Other measures of error must be considered. Examples of these are provided below.

1. *Direction of index movement accuracy*. The forecast increase or decrease of each index can be compared to the actual movements. This will provide a measure as to the accuracy of the ANNs at forecasting the movement of each index.
2. *Total return over testing set period*. The actual total return over the period which is spanned by the testing set can be compared to the forecast return over the same period. This will determine the efficiency of the forecasts over the entire period.
3. *Trading efficiency*. A theoretical amount can be invested at the beginning of the testing set period. Training algorithms based on the expected movement of the forecasts can be developed and implemented. For example, if the index is forecast to increase/decrease, a command to buy/sell the index can be given. The final value obtained from this active investment management can be compared to the passive position over the same period. If the amount earned from the active position is greater than from the passive position, the ANN is effective.
4. *Validation set error*. The error over a validation data set must be considered, as it will be an error independent of the training and testing data sets.

7.18 Parsimonious Selection Criteria

A numerical measurement of parsimony must be incorporated into the selection criteria of the optimal ANN structure. The parsimony of the ANN is taken into account when deciding on the efficient ANN structure; however, this is done subjectively and a consistent measure must be established. Using the adjusted R-squared measure may result in increased parsimonious ANN structures. However, the number of parameters does not increase with an increase in system complexity, which will skew the results.

7.19 Advanced Traditional Models

The traditional models constructed in the applications may be improved by using more advanced models. For example GARCH models, which are designed to represent the hetroskedacity (changing variance with time) in the data, can be implemented. The forecasting accuracy of these more advanced models can be compared to that of the traditional models constructed and more complex ANNs.

7.20 Combining Specific Markets

Some indices may only be affected by specific markets. Thus, combining all the markets in a single ANN may result in increased complexity and computing requirements without improved forecasting accuracy. This possibility must be investigated by comparing the ANNs which combine all the markets and inflation to those which only combine markets with significant correlation to the index being forecast.

7.21 Including in Existing Models

For practical implementation ANNs will have to be combined with existing models. Combining ANNs with any model is relatively simple, as only the data presented to the ANNs will change.

For example, an ARIMA process can be used to strip the data of all the linear relationships. The residuals can then be presented to the ANNs which strips them of the non-linear relationships. This results in all the relationships in the data being fairly represented in the model, and the final residuals consisting of only white noise. An application of this is introduced in chapter 5 through the hybrid models.

7.22 Inflation Application

The inflation data presented to the ANNs is not stripped of seasonality. The ANNs must be trained on the data once the seasonality has been removed. They can then be compared to the constructed traditional model, which is a combination of both a seasonal and ARMA model. This will determine whether the seasonality is fully captured by the ANNs or if they require the removal of these relationships from the data before training.

7.23 Money Market Application

The money market time series requires a simple difference in order to be stationary. The traditional model is built on this difference, whereas the ANNs are built on the original data. The effect of this inconsistency on the forecasting accuracy of the ANNs must be determined. This can be done by training the ANNs on the data after a simple difference has been taken and comparing them to the traditional model.

7.24 Noisy and Quiet Data Sets

From the applications considered, the ANNs generally perform better when the data set included a significant amount of noise. This trend needs to be investigated through the consideration of many different data sets, of which some are noisy and others are not. ANNs must be constructed and compared over all the data sets. This will determine if the suggested trend extends as a general rule or not.

7.25 Vectorisation

The code of the ANNs is not fully vectorised. This causes an increase in training time, as matlab deals with matrices and vectors better than value by value processing.

Ensuring the code is vectorised will be important when large data sets are considered.

7.26 Comparison Intervals

The use of comparison intervals holds little proven statistical significance. In future applications it is important that more elegant statistical tests be used when comparing models. Several tests exist for possible use, however they require the residuals to be analysed and a distribution to be determined. This process must be included in future training and testing of ANNs.

7.27 Traditional Model Selection

The traditional models, from chapter 3, are selected based on the RMSE over the testing set of data. In future a confidence interval of the performance of each model must be constructed and a statistically significant decision made as to the superior traditional model.

7.28 Robustness of Models

The ANNs and traditional models must be tested on different data sets which exhibit various characteristics. This can be done by applying the models to simulated data and then changing the underlying characteristics of the data. The performance of the models over each data set must be analysed which will indicate the characteristics of the data each model deals more efficiently with.

7.29 Validation of Models

It is important that the ANNs and traditional models are validated. This can be done through the use of simulation data sets that have known characteristics. Analysing the results of the models fitted in these sets should yield the results already known to the user, allowing any coding or other fitting issues to be identified and corrected. For the ANNs build, data sets with simple functions (increasing, sine) were used to determine if the model was performing as expected. The models replicated these functions which supported that the models were functioning correctly. However, more complex tests were not performed. Further investigation into determining the performance of these models on more complex simulations must be done. It may also be easier to compare traditional models to ANNs based on simulated data sets, however, this approach would have reduced the validity of the parameter estimation method, developed here, for ANNs applied to problems in the financial sector.

8 References

1. Russel, S. & Norvig, P., *Artificial Intelligence - A Modern Approach*. Third ed2010: Pearson.
2. Zhang, G., Patuwo, B. E. & Hu, M. Y., *Forecasting with artificial neural networks: The state of the art.* International Journal of Forecasting, 1998. **14**(1): p. 35-62.
3. Department of Electronics and Electrical Communication Engineering, I.K., *Lec-1* in *Introduction to artificial neural networks*, I.K. Department of Electronics and Electrical Communication Engineering, Editor 2009: Department of Electronics and Electrical Communication Engineering, IIT Kharagpur. p. 58 min.
4. *Turing test*. [cited 2014 09/04/2014]; Available from: http://en.wikipedia.org/wiki/Turing_test#The_Imitation_Game.
5. Huang, Y., *Advances in Artificial Neural Networks - Methodological Development and Application.* Open Access algorithms, 2009.
6. Plaut, D., Nowlan, S. & Hinton, G.E., *Learning representations by back-propogation errors*, in *Technical Report CMU-CS-86-126*1986, Carnegie-Mellon University: Carnegie-Mellon University.
7. Tan, C.N.W., *An Artificial Neural Networks Primer with Financial Applications Examples in Financial Distress Predictions and Foreign Exchange Hybrid Trading System '*.
8. Huang, G.-B., Zhu, Q.-Y. & Siew, C.-K., *Extreme learning machine: Theory and applications.* Neurocomputing, 2006. **70**(1–3): p. 489-501.
9. Korol, T., *Early warning models against bankruptcy risk for Central European and Latin American enterprises.* Economic Modelling, 2013. **31**(0): p. 22-30.
10. Schmidt, A., *Biological Neural Networks*, in *16:45:34 CEST 2000*2000: Mit.
11. Lacher, R.C., Coats, P.K., Sharma, S.C. & Fant, L.F., *A neural network for classifying the financial health of a firm.* European Journal of Operational Research, 1995. **85**(1): p. 53-65.
12. Özkan, F., *Comparing the forecasting performance of neural network and purchasing power parity: The case of Turkey.* Economic Modelling, 2013. **31**(0): p. 752-758.
13. Braun, H. & Lai, L.L., *A Neural Network Linking Process for Insurance Claims*. in *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*. 2005.
14. Wilamowski, B.M., *Neural network architectures and learning algorithms.* Journal of Machine Learning Research 2009. **12**: p. 2493-2537.
15. Jordand, M.I. & Bishop, C.M., *Neural Networks.* CRC Handbook of Computer Science, 1996.
16. Rojas, R., *Neural Networks: A Systematic Introduction*1996: Springer.
17. Krose, B. & Smagt, P.v.d., *An Introduction to Neural Networks*1996. 135.
18. Department of Electronics and Electrical Communication Engineering, I.K., *Lec-4 Nonlinear Activation Units and Learning Mechanisms*, in *Introduction to artificial neural networks*, I.K. Department of Electronics and Electrical Communication Engineering, Editor 2009: Department of Electronics and Electrical Communication Engineering, IIT Kharagpur. p. 58 min.
19. Department of Electronics and Electrical Communication Engineering, I.K., *Lec-5 Learning Mechanisms-Hebbian,Competitive,Boltzmann*, in *Introduction to artificial neural networks*, I.K. Department of Electronics and Electrical Communication Engineering, Editor 2009: Department of Electronics and Electrical Communication Engineering, IIT Kharagpur. p. 57 min.
20. Hinton, G.E., Scholarpedia. *Boltzmann machine*. 2007 [cited 2012 12/10/2012]; Boltzmann machine learning]. Available from: http://www.scholarpedia.org/article/Boltzmann_machine.
21. Landau, L.D.a.L., Evgeny Mikhailovich *Statistical Physics*, (1980) [1976].

22. Chen, E., *Introduction to Restricted Boltzmann Machines*, in *Edwin Chen's Blog*2011. p. Introduction to Restricted Boltzmann Machines.
23. Rumelhart, D.E., Hinton, G.E. & Williams, R.J., *Learning representations by back-propagating errors.* Nature, 1986. **323**: p. 533-536.
24. Noriega, L., *Multilayer Perceptron Tutorial*. 2005.
25. Wilamowski, B.M., *Advantages and problems of soft computing*. in *Industrial Informatics (INDIN), 2011 9th IEEE International Conference on*. 2011.
26. Allard, R. & Faubert, J., *Neural Networks: Different problems require different learning rate adaptive methods.* Proceedings of SPIE - The International Society for Optical Engineering 5298, 2004: p. 516-527.
27. Møller, M.F., *A scaled conjugate gradient algorithm for fast supervised learning.* Neural Networks, 1993. **6**(4): p. 525-533.
28. Riedmiller, M. & H. Braun., *A direct adaptive method for faster backpropagation learning: the RPROP algorithm*. in *Neural Networks, 1993., IEEE International Conference on*. 1993.
29. Bouqata, B., Bensaid, A., & Palliam, R., *Recurrent artificial neural networks for forecasting of forward interest rates*. in *Neural Networks, 1999. IJCNN '99. International Joint Conference on*. 1999.
30. Fei, W. & Zhigang, S., *Research on the Credit Risk Evaluation and Forecast of Housing Mortgage Loans*. in *Intelligent Information Technology Application Workshops, 2008. IITAW '08. International Symposium on*. 2008.
31. Sahin, Y. & Duman, E., *Detecting credit card fraud by ANN and logistic regression*. in *Innovations in Intelligent Systems and Applications (INISTA), 2011 International Symposium on*. 2011.
32. Abhishek, K., *A stock market prediction model using Artificial Neural Network*. in *Computing Communication & Networking Technologies (ICCCNT), 2012 Third International Conference on*. 2012.
33. O'Connor, N. & Madden, M.G., *A neural network approach to predicting stock exchange movements using external factors.* Knowledge-Based Systems, 2006. **19**(5): p. 371-378.
34. Kohzadi, N., Boyd, M.S., Kermanshahi, B. & Kaastra, I., *A comparison of artificial neural network and time series models for forecasting commodity prices.* Neurocomputing, 1996. **10**(2): p. 169-181.
35. Tsai, C.-F. & Wu, J.-W., *Using neural network ensembles for bankruptcy prediction and credit scoring.* Expert Systems with Applications, 2008. **34**(4): p. 2639-2649.
36. Saha, S.P., *On Small Sample Prediction of Financial Crisis*. in *Advances in Pattern Recognition, 2009. ICAPR '09. Seventh International Conference on*. 2009.
37. Firer, C., & McLeod, H., *Equities, Bonds, Cash and Inflation: Historical performance in South Africa.* Investment Analysts, 1999. **50**: p. 7-28.
38. Firer, C., & Staunton, M., *102 Years of South African financial market history.* Investment Analysts, 2002. **56**: p. 57-65.
39. Dodson, J.A., *Risk & Asset Allocation - Homework for Week 4*, 2013.

Appendix A – Isolated Money Market ANN - One Month Forecast Results

A1 Determining an Efficient Structure

A1.1 Part 1 – Efficient number of training epochs

The parameters used for this experiment were:

Parameter	Value
Network Structure	36-64-1
Training set size	295
Testing set size	100
Number of initialized structures	5
Epochs	100 000
Training algorithm	RPROP
Learning rate change: Increase Decrease Maximum Minimum	 0.5% (1.005) 20% (0.8) 100 0.000000001

Table A1: Parameters for experiment - Part 1

A1.1.1 Observations – Part 1

Figure A1 indicated the error of the system minimised after 60 000 epochs with a minimum Root Mean Squared Error (RSME) of 5.3219e-004 and 3.9192e-004 over the training and testing sets, respectively. The difference in RMSE between different epoch numbers was calculated and analysed. The decrease in error after 2 000 epochs (2.2412e-004 over testing set and 1.6616e-004 over training set) was minimal and did not warrant the increase in computing time required for more.

The RMSE in both the training and testing sets case continued to decrease over a vast number of training epochs. There were no signs of over fitting in this system, which was unexpected. This anomaly may be explained through consideration of the nature of the money market, which is highly predictable. Further, this indicated there was low levels of noise in the data set.

		Training Epochs				
	Min RMSE	**100**	**2 000**	**100 000**	**Δ in RMSE – 100 to 2 000**	**Δ in RMSE – 2 000 to 100 000**
RMSE – Testing Set	3.92e-004	0.0021	6.25e-004	4.01e-004	4.01e-004	2.24e-004
RMSE – Training Set	5.32e-004	0.0027	6.99e-004	5.32e-004	5.32e-004	1.66e-004

Table A2: RMSE after different numbers of epochs

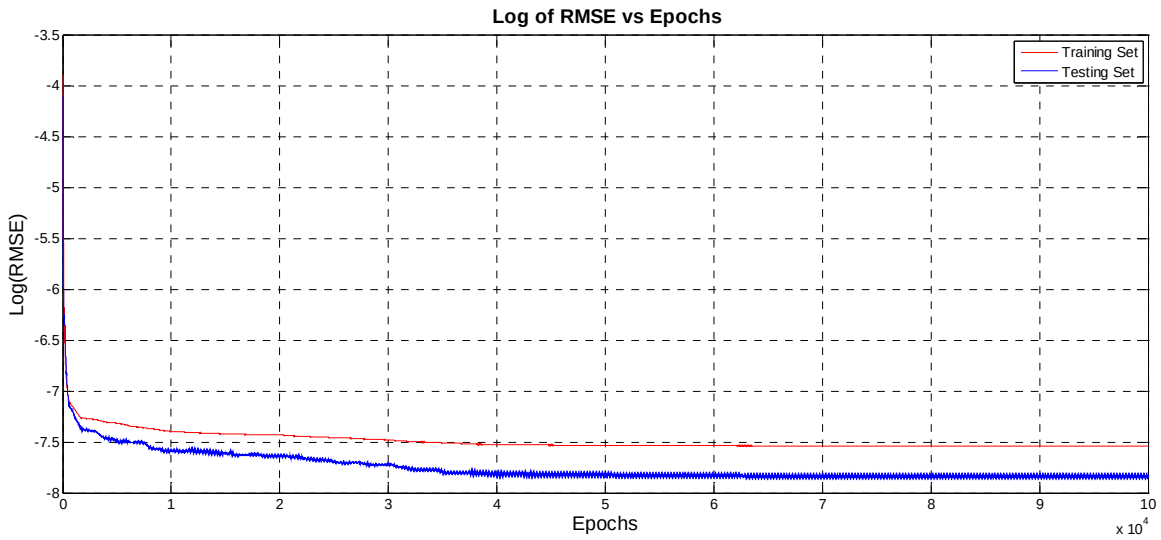

Figure A1: Log of RMSE over training process

A1.1.2 Conclusion – Part 1:

From the results obtained, the number of epochs used to compare different structures was 2000. This was decided as it was the point which balanced accuracy and computing requirements. Ideally, a larger number of epochs should be used, however, this would have increased the computing time considerably and was not feasible in this case. This decision was primarily based on the results over the testing set, as the error over the training set would continue to decrease indefinitely, leading to over fitting.

A1.2 Part 2 – Determining efficient parameters

A1.2.1 Observations – Learning Rate Increase/Decrease

From table A3 it is clear the learning rate increase of 1.005 (0.5% increase) and decrease of 0.8 (20% decrease) resulted in the smallest MSE being achieved over both the training and testing data sets. The minimum MSEs were highlighted in green, with the efficient combination being highlighted in orange. In this case the efficient combination was that

which achieved the minimum MSE. Further, the minimum MSE achieved was 8.61E-07 and 4.8E-0.7 over the training and testing sets, respectively.

It is important to note the values used to decide on the learning rate changes were the arithmetic average MSE over different combinations of input and hidden neurons. This average did not explicitly allow for the event of outliers in the data and may be skewed. From the investigations there were no outliers found but extensive tests to verify this remark were not performed. Further investigation into the possibility of outliers can be done.

Training Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	1.36E-06	2.56E-06	3.50E-06
	0.5	1.07E-06	2.00E-06	3.09E-06
	0.8	**8.61E-07**	1.65E-06	2.72E-06
	min	8.61E-07		
Testing Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	7.70E-07	1.31E-06	1.59E-06
	0.5	5.89E-07	1.16E-06	1.38E-06
	0.8	**4.80E-07**	9.63E-07	1.29E-06
	min	4.80E-07		
	Minimum			
	Efficient			

Table A3: MSE for Learning Rate Changes

A1.2.2 Conclusion – Learning Rate Increase/Decrease

Base on the results obtained the following decisions were made:

- A learning rate decrease of 0.8 was to be used, i.e. a 20% decrease in learning rate if the error of the system increases during training.
- A learning rate increase of 1.005 was to be used, i.e. a 0.5% increase in learning rate if the error of the system decreases during training.

A1.2.3 Observations – Input and Hidden Neurons – Training Set's MSE surface

The MSE over the training set of data minimised after 4 hidden neurons were considered, as shown by the flattening of the surface in both figure A2 and A3. When 4 or more hidden neurons were considered the MSE minimised for any number of input neurons. This was unexpected, as the MSE of the system should decrease over the training set with the increase in complexity. The MSE did decrease as the structure became more complex, however, this reduction was of a lower magnitude than expected. The MSE was insensitive to the change in input neurons for the case of 4 or more hidden neurons. When a single hidden neuron was considered, the MSE was larger. The minimum MSE of 4.24E-07 was obtained using 36 input and 32 hidden neurons. The maximum MSE of 3.94E-06 was obtained using

12 input and a single hidden neurons. The maximum point was suspicious, since the largest MSE was expected to be obtained when a single input and hidden neuron was considered.

The upper and lower limit of the MSE surface over the training set requires assumptions in order to hold statistical significance. There was insufficient information available to make these assumptions. These limits, however, did provide a good estimate of the MSE limits for the randomly initialized structures.

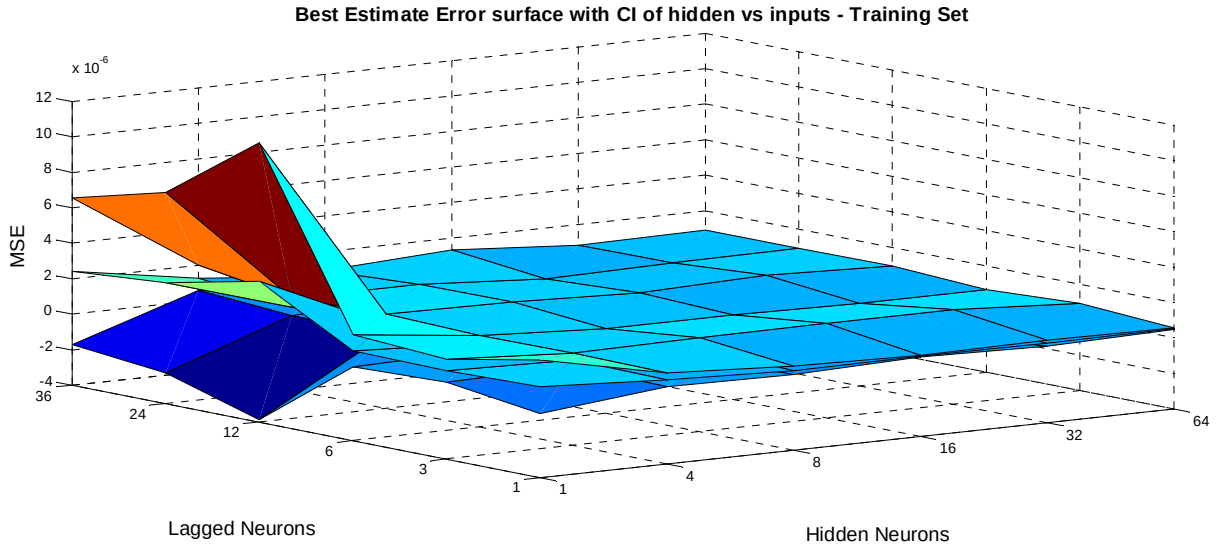

Figure A2: MSE with CI surface – Training data set

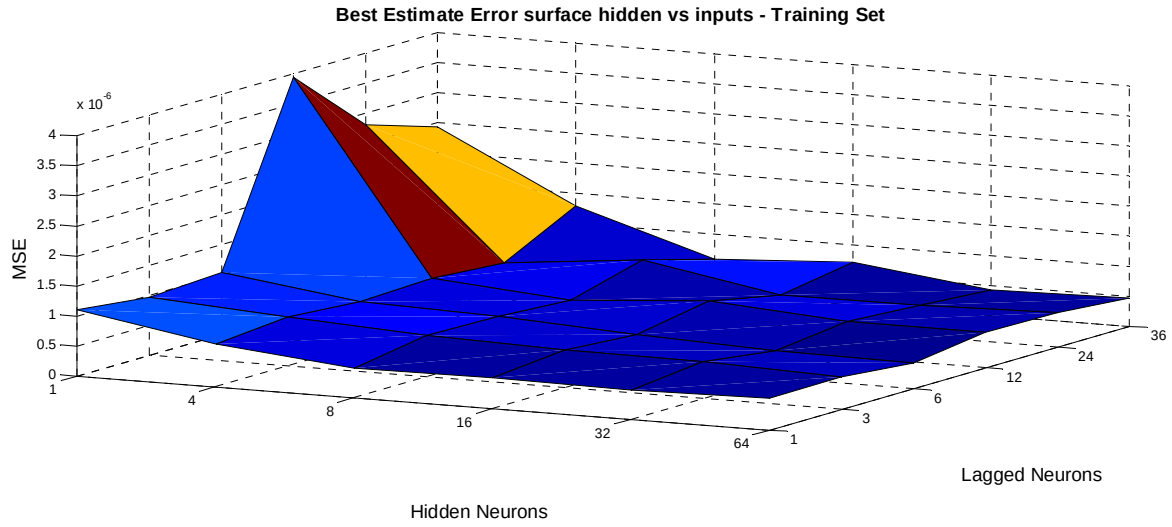

Figure A3: MSE surface – Training data set

A1.2.4 Observations – Input and Hidden Neurons – Testing Set's MSE surface

The testing set MSE surface (figure A4 and A5) minimized after 6 input and 4 hidden neurons. This was similar to the MSE surface over the training set of data (figure A2 and A3). Figure 4.31 suggested there was little over fitting present in the system for any combination of input and hidden neurons. Figure A5 indicated the maximum MSE on the best estimate MSE surface of 2.03E-06 was achieve using 36 input and 4 hidden neurons. The MSE decreased

rapidly when more complex systems were considered. An accurate and parsimonious model with 6 input and 4 hidden neurons was suggested by figure A5. The minimum MSE on the best estimate MSE surface of 1.31E-07 was obtained using a single input and 64 hidden neurons. This was unexpected as it suggests only a lagged period of a single month had an effect on the return on the money market. This may be explained by considering the nature of the money market. The money market has a low level of volatility and is unlikely to fluctuate significantly over a single month.

The upper and lower limit of the MSE surface over the testing set requires assumptions in order to hold statistical significance. There was insufficient information available to make these assumptions. These limits, however, did provide a good estimate of the MSE limits for the randomly initialized structures.

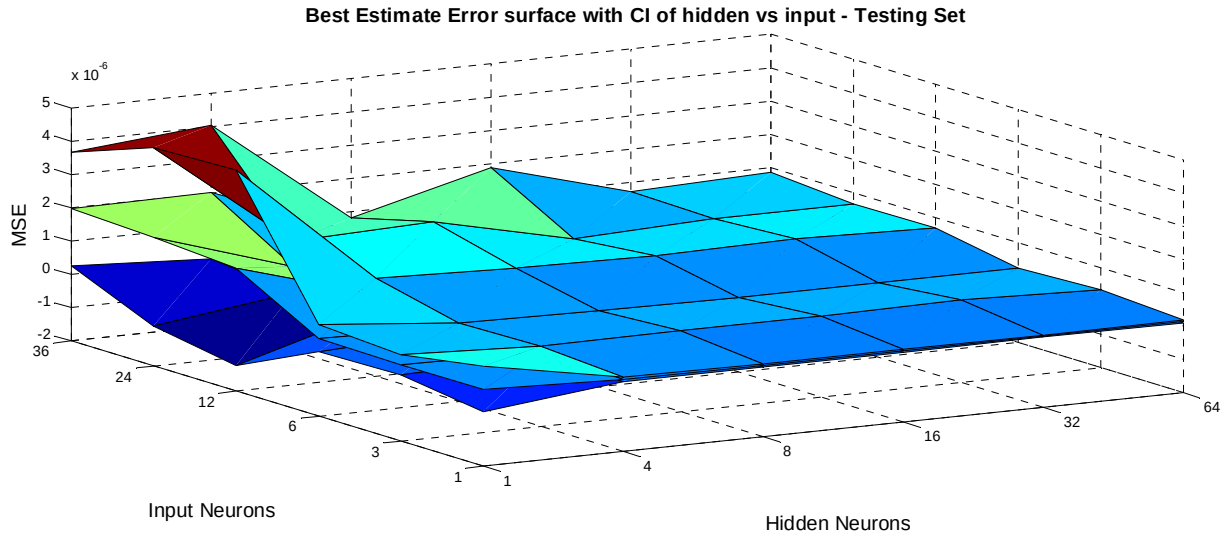

Figure A4: MSE with CI surface – Testing data set

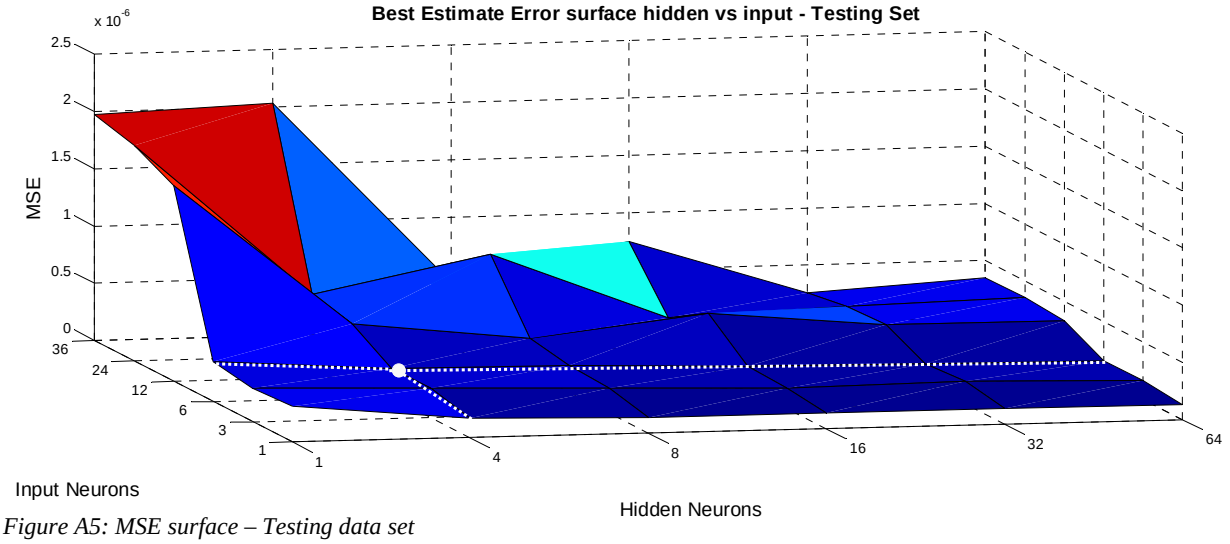

Figure A5: MSE surface – Testing data set

A1.2.5 Numerical Representation of MSE surfaces

The tables below provide the numerical average MSE values used to construct the error surfaces in figures A2, A3, A4 and A5.

Training Sets							
		LR increase		1.005			
		LR decrease		0.8	(20% decrease)		
	Average MSE vs Input and Hidden						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	1.11E-06	7.14E-07	4.92E-07	5.09E-07	4.90E-07	5.32E-07
	3	9.71E-07	8.14E-07	6.91E-07	6.24E-07	6.09E-07	5.41E-07
	6	1.04E-06	7.25E-07	6.65E-07	5.35E-07	**4.53E-07**	4.28E-07
	12	3.94E-06	7.65E-07	5.88E-07	7.50E-07	5.87E-07	5.84E-07
	24	2.80E-06	6.80E-07	9.06E-07	4.99E-07	5.09E-07	5.69E-07
	36	2.42E-06	1.28E-06	5.71E-07	7.10E-07	4.24E-07	4.73E-07
average	8.61E-07						
min	4.24E-07						
	Upper limit MSE vs Input and Hidden (+2 STD Dev)						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	2.63E-06	1.10E-06	7.32E-07	5.56E-07	6.51E-07	5.81E-07
	3	1.60E-06	1.18E-06	1.27E-06	1.35E-06	1.36E-06	9.30E-07
	6	1.94E-06	1.13E-06	7.79E-07	7.63E-07	**6.61E-07**	6.50E-07
	12	1.17E-05	1.33E-06	1.23E-06	1.00E-06	1.20E-06	9.73E-07
	24	7.89E-06	1.17E-06	1.20E-06	7.16E-07	8.45E-07	9.16E-07
	36	6.54E-06	2.00E-06	9.16E-07	1.30E-06	7.92E-07	9.06E-07
average	1.74E-06						
Efficient Structure							
Minimum Error							

Table A4: MSE of different combinations of input and hidden neurons – Training data set

Testing Sets							
		LR increase		1.005			
		LR decrease		0.8	(20% decrease)		
	Average MSE vs Input and Hidden						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	3.16E-07	1.67E-07	1.38E-07	1.36E-07	1.33E-07	1.31E-07
	3	2.95E-07	2.53E-07	2.14E-07	1.76E-07	1.94E-07	1.66E-07
	6	3.47E-07	2.46E-07	2.35E-07	1.87E-07	1.71E-07	1.57E-07
	12	1.71E-06	4.58E-07	2.99E-07	4.73E-07	3.34E-07	3.29E-07
	24	1.88E-06	5.41E-07	8.52E-07	2.60E-07	3.16E-07	3.59E-07
	36	1.97E-06	2.03E-06	4.32E-07	7.46E-07	2.58E-07	3.52E-07
average	4.80E-07						
min	1.31E-07						
	Upper limit MSE vs Input and Hidden (+2 STD Dev)						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	9.99E-07	2.41E-07	1.81E-07	1.80E-07	1.54E-07	1.80E-07
	3	5.95E-07	4.06E-07	4.31E-07	4.26E-07	4.88E-07	3.31E-07
	6	7.51E-07	3.59E-07	2.95E-07	2.38E-07	2.38E-07	2.31E-07
	12	4.65E-06	9.42E-07	8.05E-07	7.40E-07	8.14E-07	6.78E-07
	24	4.56E-06	1.32E-06	1.45E-06	4.92E-07	6.56E-07	6.48E-07
	36	3.68E-06	4.04E-06	8.13E-07	1.87E-06	7.12E-07	8.44E-07
average	1.01E-06						
Efficient Structure							
Minimum Error							

Table A5: MSE of different combinations of input and hidden neurons – Testing data set

A1.2.6 Conclusion – Input and Hidden Neurons

The observations over the training data set indicated an efficient structure of 6 input and 32 hidden neurons, which was similar to that suggested over the testing data set. Following this it was decided that an efficient structure of 6 input and 32 hidden neurons was to be used. Further, the results from this experiment suggested there was little randomness in the data set as there was little signs of over fitting. This observed characteristic was noted.

The choice made here had subjective components. It was important not to base the decision purely on the results over the training set as this would lead to over fitting. Similarly, basing the decision purely on the results over the testing set would lead to the fitting of the ANN to the testing set. Both the scenarios described above would have reduced the generalization ability of the ANN.

A1.2.7 Conclusion of Determining an Efficient Structure

Efficient parameters were:

- 6 input neurons,
- 32 hidden neurons,
- A learning rate increase of 1.005 (0.5% increase), and
- A learning rate decrease of 0.8 (20% decrease).

A2 Analysing the Efficient ANN

The ANN was trained with the following parameters.

Training set size	325
Testing set size	100
Input/lagged neurons	6
Hidden neurons	32
Number of initialized structures	10
Epochs	100 000
Training algorithm	RPROP
Learning rate:	
Increase	1.005 (0.5% increase)
Decrease	0.8 (20% decrease)
Maximum	100
Minimum	0.000000001

Table A6: Parameters for the efficient ANN structure

A2.1 Observations of System Error

Figures A6 indicated the error of the system decreased over the training epochs. This was true for both the training and testing sets. The continual decrease over the testing set suggested this system was not capable of over fitting the data. This could only be true if there was little (if any) random variation in the data set. It was concluded that an efficient number of training epochs was 100 000 as this resulted in the minimum MSE over the training set (2.2433e-007) and the testing set (4.6892e-008). The numerical error values at points of interest are provided in table A7.

Error Readings	**Testing Set**	**Training Set**
Minimum RMSE	2.1655e-004	4.7364e-004
Minimum MSE	4.6892e-008	2.2433e-007
Final RMSE after training	2.1655e-004	4.7364e-004

(100 000 epochs)		
Final MSE after training *(100 000 epochs)*	4.6892e-008	2.2433e-007

Table A7: Results of efficient model after training

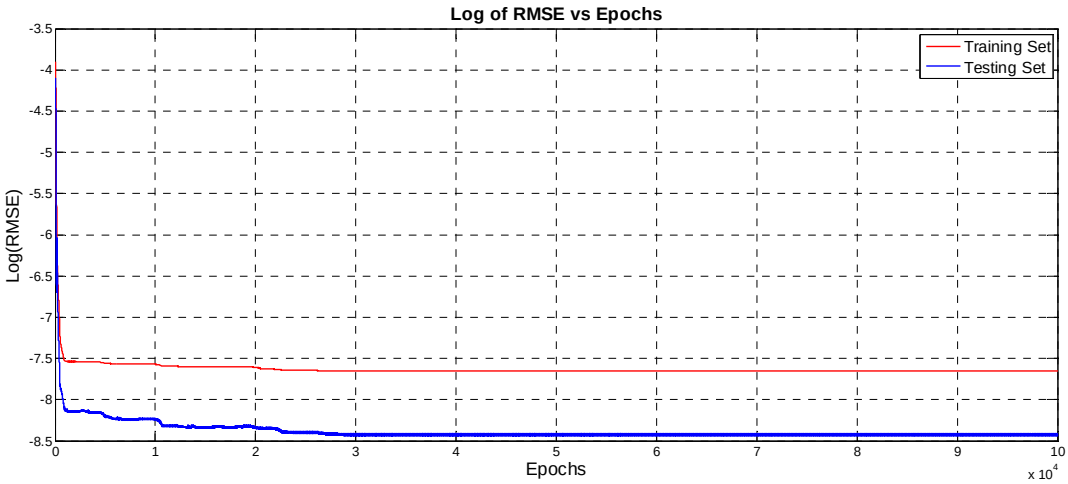

Figure A6: Log of RMSE after 100 000 training epochs

A2.2 Observations of Forecasts and Actuals

The ANN captured the underlying trends in the data set as expected. There were certain periods where the system underestimated the actual values, this could be explained by the effect of external influences on the money market. The best estimate over the training and testing sets was represented by the blue line in figure A7.

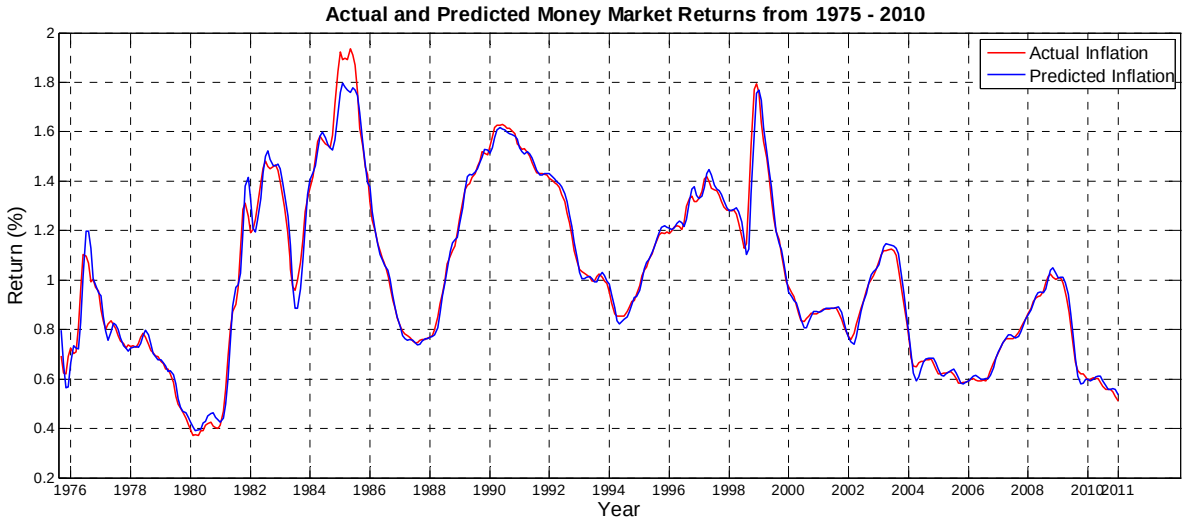

Figure A7: Actual and Predicted – Training and Testing Data Sets

A2.3 Conclusion - Analysing the Efficient ANN

The ANN captured the trends underlying the data. The efficient ANN structure had an MSE over the training and testing sets of 2.2433e-007 and 4.6892e-008, respectively. The ANN performed similarly over the testing and training data set. The level of effectiveness was determined by comparing these results to traditional models in a following chapter (Chapter 5). The minimum RMSE over the testing set, of 2.1655e-004, and over the training set, of 4.7364e-004, was obtained after the maximum number of epoch (100 000) were completed.

The RMSE over the training and testing sets continued to decrease as the training approached 100 000 epochs. This was not expected for the testing set, as it indicated that the system could not over fit. Theoretically, this could only happen when there is little random variation in the data and the underlying trends remain stable.

The error was larger over the training than the testing set. This was expected as there are more unexplainable events over the first 332 observations than over the following 100, as seen from the large spikes of the actual values in figure A7. The training data set contained more unexpected events than the testing set and hence had the greater error of the two. Further, investigation could be done into the reason for this occurrence. This was not done here due to time constraints.

A3 Summary of Experiment

The following table provides a summary of the results from this experiment.

Structure of ANN	6-32-1
Training epoch required for optimal training	100 000
Minimum Root Mean Squared Error:	
Training Data Set	4.7364e-004
Testing Data Set	2.1655e-004
Final Root Mean Squared Error (100 000 epochs):	
Training Data Set	4.7364e-004
Testing Data Set	2.1655e-004
Parameters for RPROP:	
Learning Rate Increase	0.5% (1.005)
Learning Rate Decrease	20% (0.8)
Data Sets:	
Training set	325 observations (1975 – 2002)
Testing set	100 observations (2002 – 2010)
Times the structure was initialized	10

Table A8: Summary of experiment's result

Appendix B – Isolated Bond Market ANN - One Month Forecast Results

B1 Determining an Efficient Structure

B1.1 Part 1 – Efficient number of training epochs

The parameters used for this experiment were:

Parameter	Value
Network Structure	36-64-1
Training set size	295
Testing set size	100
Number of initialized structures	5
Epochs	100 000
Training algorithm	RPROP
Learning rate change: Increase Decrease Maximum Minimum	 0.5% (1.005) 20% (0.8) 100 0.000000001

Table B1: Parameters for experiment - Part 1

B1.1.1 Observations – Part 1

Figure B1 indicated the minimum RMSE (0.0193) over the testing set was achieved within the first 2 000 epochs. The RMSE increased after this point which indicated the system was over fitting the ANN to this particular data set. The minimum RMSE (0.0028) over the training set was achieved at the 100 000 epoch mark, as expected. The RMSE of 0.0193 over the testing set indicated that historically there were limited relationships between the successive return on the bond market. It was expected that external factors had a significant impact on the observations.

		Training Epochs				
	Min RMSE	**100**	**2 000**	**100 000**	**Δ in RMSE – 100 to 2 000**	**Δ in RMSE – 2 000 to 100 000**
RMSE – Testing Set	0.0193	0.0193	0.0245	0.0272	-0.0052	-0.0027
RMSE – Training Set	0.0028	0.0267	0.0130	0.0028	0.0137	0.0102

Table B2: RMSE after different numbers of epochs

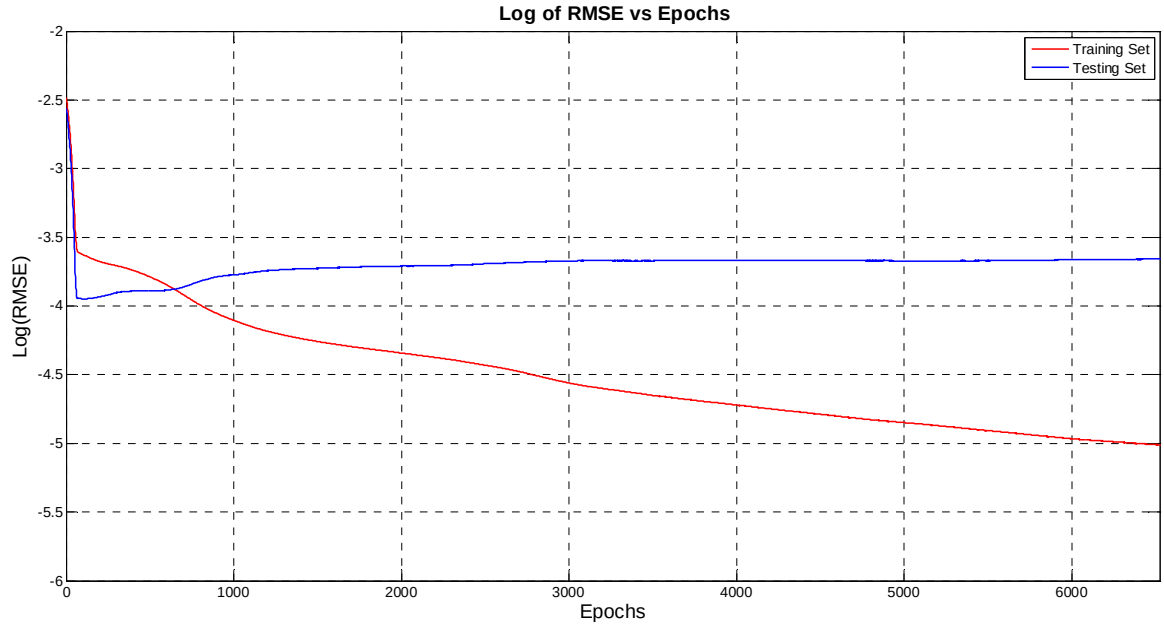

Figure B1: Log of RMSE over training process

B1.1.2 Conclusion – Part 1:

From the results obtained, the number of epochs used to compare different structures was 2000. This was chosen because it was important to allow the system to reach an over fitted stage, which implied the minimum error over the testing set of data was surpassed and captured. Since the minimum RMSE over the testing set was 0.0193, and the RMSE after 2 000 epochs was 0.0245, the minimum RMSE was obtain within the first 2 000 epochs. As this system was the largest with the slowest learning parameters, the other simpler structures would reach their minimum within this 2 000 epochs. This decision was primarily based on the results over the testing set, as the error over the training set would continue to decrease indefinitely, leading to over fitting.

B1.2 Part 2 – Determining efficient parameters

B1.2.1 Observations – Learning Rate Increase/Decrease

From table B3 it is clear the minimum MSE over the training set (4.77E-04) was achieved with a learning rate increase of 1.05 (5% increase) and a decrease of 0.8 (20% decrease). The minimum MSE over the testing set (3.66E-04) was achieved with an increase of 1.1 (10% increase) and a decrease of 0.2 (80% decrease). These observations did not support the same conclusion.

The shape of the average MSE surface over the training set was considered. It was observed the range over the training set surface was significant (1.37E-04) in comparison to the range of the average MSE surface over the testing set (0.04E-04). The efficient learning rate changes were chosen by determining the smallest trade off in error over both sets of data.

The efficient learning rate increase and decrease (represented by orange in table B3) were 1.05 and 0.8 as this resulted in the minimum error over the training data set (represented by green in table B3) and an increase in error by 0.01E-04 over the testing data set.

It is important to note the values used to decide on the learning rate changes were the arithmetic average MSE over different combinations of input and hidden neurons. This average did not explicitly allow for the event of outliers in the data and may be skewed. From the investigations there were no outliers found but extensive tests to verify this remark were not performed. Further investigation into the possibility of outliers can be done.

Training Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	6.14E-04	5.36E-04	5.53E-04
	0.5	5.86E-04	5.18E-04	5.23E-04
	0.8	5.26E-04	**4.77E-04**	4.90E-04
	min	4.77E-04		
Testing Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	3.70E-04	3.67E-04	3.66E-04
	0.5	3.69E-04	3.67E-04	3.67E-04
	0.8	3.70E-04	**3.67E-04**	3.68E-04
	min	3.66E-04		
	Minimum			
	Efficient			

Table B3: MSE for Learning Rate Changes

B1.2.2 Conclusion – Learning Rate Increase/Decrease

Base on the results above the following decisions were made:

- A learning rate decrease of 0.8 was to be used, i.e. a 20% decrease in learning rate if the error of the system increases during training.
- A learning rate increase of 1.05 was to be used, i.e. a 5% increase in learning rate if the error of the system decreases during training.

B1.2.3 Observations – Input and Hidden Neurons – Training Set's MSE surface

The error surface in figure B2 indicated the MSE of the system decreased as the number of input and hidden neurons increased. This was expected since the increased complexity of the system would increase the accuracy over the training data set. The upper and lower surface represents the upper and lower limit of the MSE. The limits were determined by increasing/decreasing the best estimate surface by two standard deviations of the MSE. Considering the best estimate error surface, figure B3, it was clear the MSE of the system decreased as the complexity of the system increased. The surface obtained the minimum at 36 input and 64 hidden neurons (5.15E-05).

The upper and lower limit of the MSE surface over the training set requires assumptions in order to hold statistical significance. There was insufficient information available to make these assumptions. These limits, however, did provide a good estimate of the MSE limits for the randomly initialized structures.

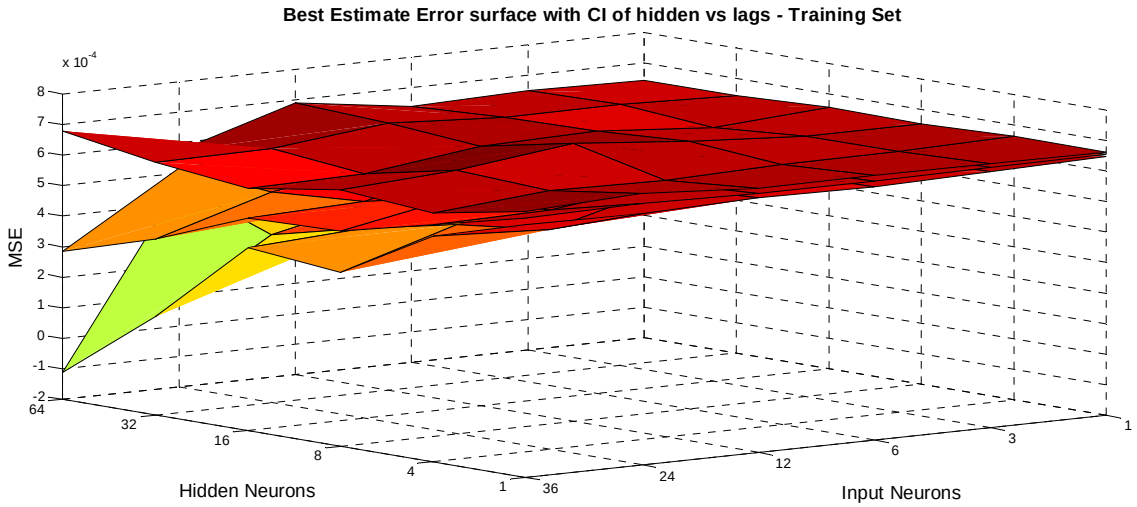

Figure B2: MSE with CI surface – Training data set

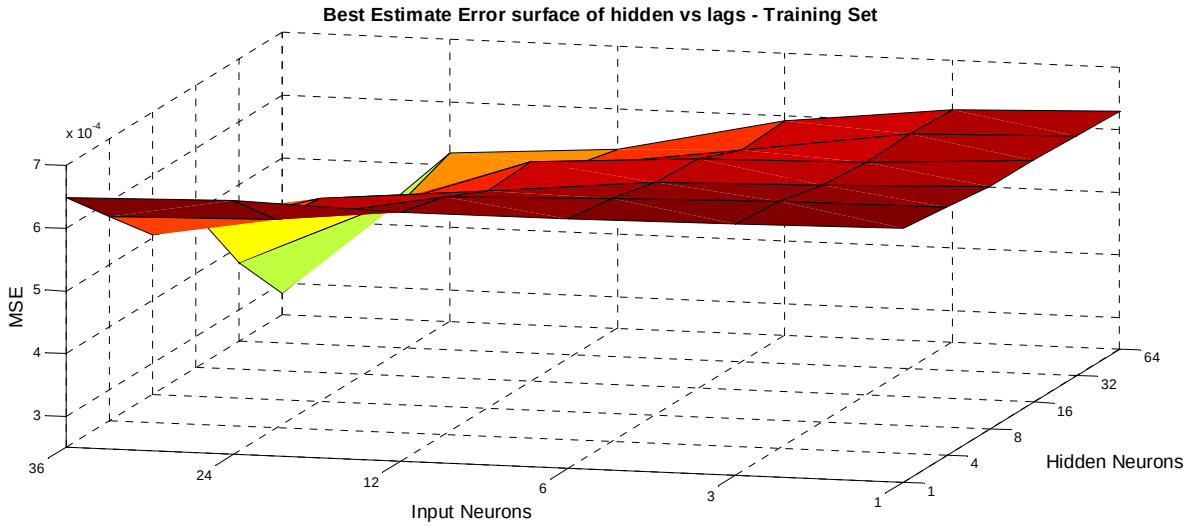

Figure B3: MSE surface – Training data set

B1.2.4 Observations – Input and Hidden Neurons – Testing Set's MSE surface

The testing set surface (figure B4 and B5) indicated a rough surface with no visible trend. The minimum MSE on the best estimate surface (figure B5) was achieved using 1 input and 32 hidden neurons (3.6E-04). The MSE surface over the testing set was structurally different to that of the training set.

The upper and lower limit of the MSE surface over the testing set requires assumptions in order to hold statistical significance. There was insufficient information available to make these assumptions. These limits, however, did provide a good estimate of the MSE limits for the randomly initialized structures.

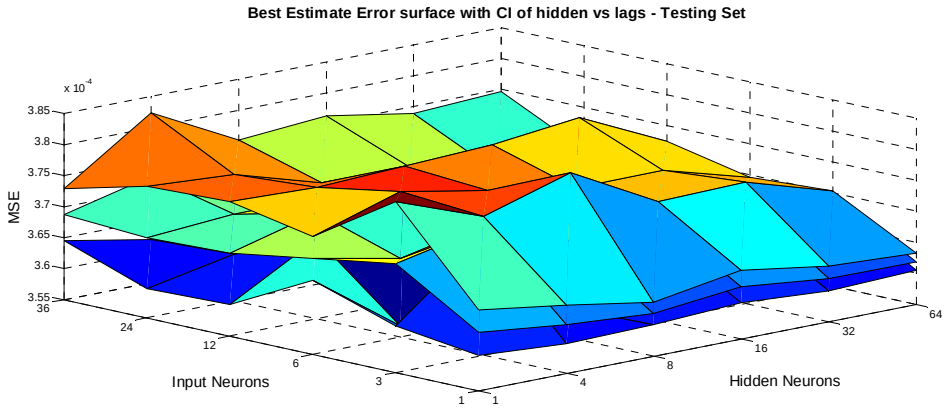

Figure B4: MSE with CI surface – Testing data set

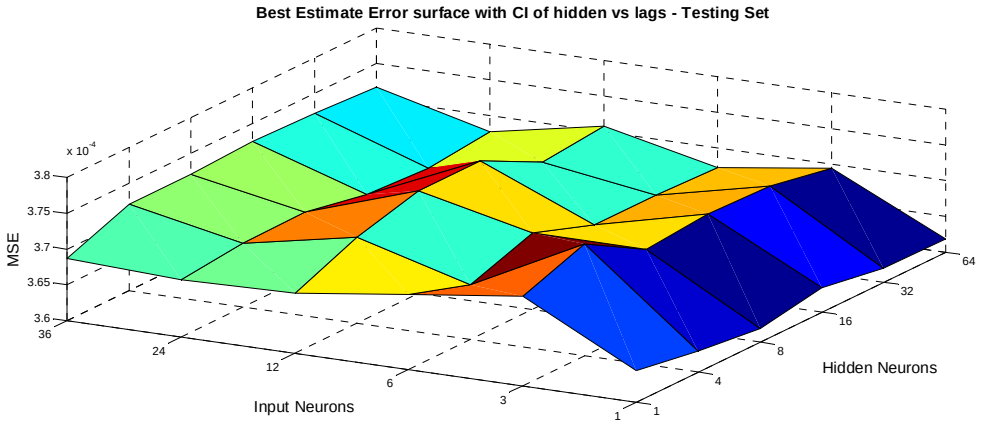

Figure B5: MSE surface – Testing data set

4.8.1.2.5 Numerical Representation of MSE surfaces

The tables below provide the numerical average MSE values used to construct the error surfaces in figures B2, B3, B4 and B5.

Training Sets							
		LR increase		1.05			
		LR decrease		0.8	(20% decrease)		
	Average MSE vs Input and Hidden						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	6.57E-04	6.38E-04	6.33E-04	6.25E-04	6.20E-04	6.21E-04
	3	6.52E-04	6.28E-04	6.13E-04	5.88E-04	5.81E-04	5.56E-04
	6	6.53E-04	6.04E-04	5.48E-04	5.46E-04	**4.47E-04**	3.95E-04
	12	6.49E-04	5.40E-04	4.71E-04	4.26E-04	3.48E-04	3.01E-04
	24	6.34E-04	5.06E-04	3.78E-04	2.66E-04	2.15E-04	1.00E-04
	36	6.15E-04	4.02E-04	3.04E-04	2.10E-04	1.41E-04	5.15E-05
average	4.77E-04						
min	5.15E-05						
	Upper limit MSE vs Input and Hidden (+2 STD Dev)						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	6.65E-04	6.60E-04	6.42E-04	6.30E-04	6.29E-04	6.25E-04
	3	6.64E-04	6.80E-04	6.57E-04	6.67E-04	6.29E-04	5.79E-04
	6	6.65E-04	6.54E-04	6.10E-04	5.76E-04	**5.12E-04**	4.73E-04
	12	6.58E-04	6.47E-04	5.70E-04	4.74E-04	5.17E-04	4.23E-04
	24	6.44E-04	5.65E-04	5.51E-04	4.21E-04	3.23E-04	1.77E-04
	36	6.80E-04	5.69E-04	4.04E-04	3.50E-04	2.05E-04	8.71E-05
average	5.41E-04						
Efficient Structure							
Minimum Error							

Table B4: MSE of different combinations of input and hidden neurons – Training data set

Testing Sets							
		LR increase		1.05			
		LR decrease		0.8	(20% decrease)		
	Average MSE vs Input and Hidden						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	3.64E-04	3.62E-04	3.60E-04	3.60E-04	3.60E-04	3.60E-04
	3	3.71E-04	3.70E-04	3.67E-04	3.67E-04	3.69E-04	3.70E-04
	6	3.72E-04	3.61E-04	3.62E-04	3.63E-04	3.64E-04	3.67E-04
	12	3.69E-04	3.68E-04	3.68E-04	3.66E-04	3.72E-04	3.71E-04
	24	3.70E-04	3.67E-04	3.65E-04	3.66E-04	3.65E-04	3.67E-04
	36	3.69E-04	3.79E-04	3.74E-04	3.74E-04	3.72E-04	3.70E-04
average	3.67E-04						
min	3.60E-04						
	Upper limit MSE vs Input and Hidden (+2 STD Dev)						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	3.73E-04	3.65E-04	3.65E-04	3.62E-04	3.62E-04	3.61E-04
	3	3.85E-04	3.81E-04	3.74E-04	3.71E-04	3.72E-04	3.71E-04
	6	3.93E-04	3.69E-04	3.67E-04	3.68E-04	3.70E-04	3.72E-04
	12	3.78E-04	3.78E-04	3.75E-04	3.76E-04	3.74E-04	3.74E-04
	24	4.07E-04	3.71E-04	3.68E-04	3.68E-04	3.69E-04	3.68E-04
	36	3.73E-04	3.85E-04	3.78E-04	3.86E-04	3.80E-04	3.72E-04
average	3.74E-04						
Efficient Structure							
Minimum Error							

Table B5: MSE of different combinations of input and hidden neurons – Testing data set

B1.2.6 Conclusion – Input and Hidden Neurons

In order to balance the error between the structures over the training and testing data sets, a structure was chosen which minimized the reduction of accuracy. The structure chosen had 6 input and 32 hidden neurons. The increase in MSE over the training and testing data sets were 4.185E-04 and 6E-06, respectively. The increase over the training set seemed significant, however, on comparison with the second lowest minimum (1E-04) was far less significant.

The choice made here had subjective components. It was important not to base the decision purely on the results over the training set as this would lead to over fitting. Similarly, basing the decision purely on the results over the testing set would lead to the fitting of the ANN to the testing set. Both the scenarios described above would have reduced the generalization ability of the ANN.

B1.2.7 Conclusion of Determining an Efficient Structure

Efficient parameters were:

- 6 input neurons,
- 32 hidden neurons,
- A learning rate increase of 1.05 (5% increase), and
- A learning rate decrease of 0.8 (20% decrease).

B2 Analysing the Efficient ANN

The ANN was trained with the following parameters.

Training set size	325
Testing set size	100
Input/lagged neurons	6
Hidden neurons	32
Number of initialized structures	10
Epochs	10 000
Training algorithm	RPROP
Learning rate:	
Increase	1.05 (5% increase)
Decrease	0.8 (20% decrease)
Maximum	100
Minimum	0.000000001

Table B6: Parameters for the efficient ANN structure

B2.1 Observations of System Error

There was little over fitting in the system, as indicated by the stable error over the testing set in figure B6. The minimum RMSE of the system (0.0187) over the testing set was obtained after approximately 2 000 training epochs. The error over the testing and training set did not intersect, however there were signs of a possible convergence of the errors after the 10 000 epoch mark. To avoid over fitting the efficient number of epochs used to train the efficient model was 2 000. This number balanced both the error over the training and testing sets. Figure B6 illustrates the log of the Root Mean Squared Error (Log (RMSE)) of the system over 10 000 training epochs. The numerical error value at points of interest are provided in table B7.

Error Readings	Testing Set	Training Set
Minimum RMSE	0.0187	0.0219
Minimum MSE	3.4936e-004	4.7772e-004
Final RMSE after training *(2 000 epochs)*	0.0188	0.0233
Final MSE after training *(2 000 epochs)*	3.5303e-004	5.4206e-004

Table B7: Results of efficient model after training

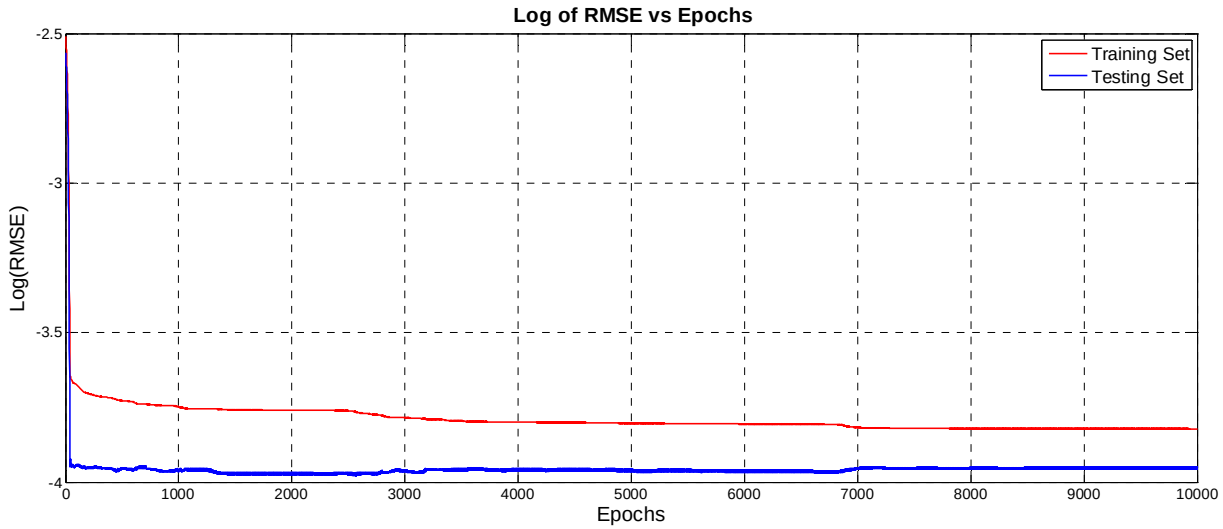

Figure B6: Log of RMSE after training epochs

B2.2 Observations of Forecasts and Actuals

The ANN captured the underlying trends in the data set as expected. The predictions were less representative of the data as the other cases considered. This was due external factors having a greater effect, than previous observations, on the return. The best estimate over the training and testing sets was the blue line in each figure B7.

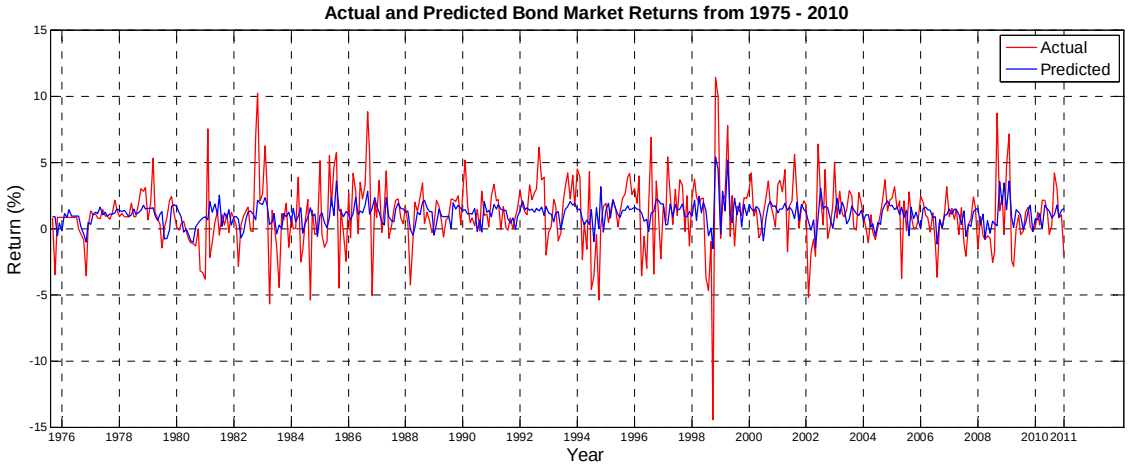

Figure B7: Actual and Predicted – Training and Testing Data Sets

B2.3 Conclusion - Analysing the Efficient ANN

The ANN captured the trends underlying the data. The efficient ANN structure (after 2 000 training epochs) had an MSE over the training and testing sets of 3.5303e-004 and 5.4206e-004, respectively. The ANN performed similarly over the testing and training data set. The level of effectiveness was determined by comparing these results to traditional models in a following chapter (Chapter 5). The minimum RMSE over the testing set, of 0.0187, and over the training set, of 0.0219, was obtained after approximately 2 000 and 10 000 epochs, respectively. The RMSE of the training set decreased as the training went over 2 000 epochs and continued to decrease until the maximum number of epochs were obtained. Theoretically, this was expected as ANNs have the ability to mimic a data set precisely. This mimicking leads to over fitting and reduces the systems generalization ability. To avoid over fitting but to ensure the underlying relationships in the data were captured, decisions relating to efficiency were made with reference to the performance of the ANN over both the testing and training data set.

B3 Summary of Experiment

The following table provides a summary of the results from this experiment.

Structure of ANN	6-32-1
Training epoch required for optimal training	2 000
Minimum Root Mean Squared Error:	
Training Data Set	0.0219
Testing Data Set	0.0187
Final Root Mean Squared Error (2 000 epochs):	
Training Data Set	0.0233
Testing Data Set	0.0188
Parameters for RPROP:	
Learning Rate Increase	5% (1.05)
Learning Rate Decrease	20% (0.8)
Data Sets:	
Training set	325 observations (1975 – 2002)
Testing set	100 observations (2002 – 2010)
Times the structure was initialized	10

Table B8: Summary of experiment's result

Appendix C – Isolated Equity Market ANN - One Month Forecast Results

C1 Determining an Efficient Structure

C1.1 Part 1 – Efficient number of training epochs

The parameters used for this experiment were:

Parameter	Value
Network Structure	36-64-1
Training set size	295
Testing set size	100
Number of initialized structures	5
Epochs	10 000
Training algorithm	RPROP
Learning rate change: Increase Decrease Maximum Minimum	 0.5% (1.005) 20% (0.8) 100 0.000000001

Tables C1: Parameters for experiment - Part 1

C1.1.1 Observations – Part 1

Figure C1 indicated the minimum RMSE (0.0533) over the testing set was achieved within the first 2 000 epochs. The RMSE increased after the minimum was achieved, until a plateau was constructed after approximately 2 000 epochs. This increase indicated the presence of over fitting and thus random variation in the data set. The RMSE over the training set continued to decrease until the maximum number of epochs was achieved. This was theoretically as expected.

The RMSE of 0.0533 over the testing set indicated that historically there were limited relationships between the successive return on the equity market. This was expected as it is believed that external factors have a significant impact on the behaviour of the equity market.

		Training Epochs				
	Min RMSE	**100**	**2 000**	**10 000**	**Δ in RMSE – 100 to 2 000**	**Δ in RMSE – 2 000 to 10 000**
RMSE – Testing Set	0.0533	0.0537	0.0732	0.0750	-0.0195	-0.0018
RMSE – Training Set	0.0064	0.0644	0.0152	0.0064	0.0492	0.0088

Tables C2: RMSE after different numbers of epochs

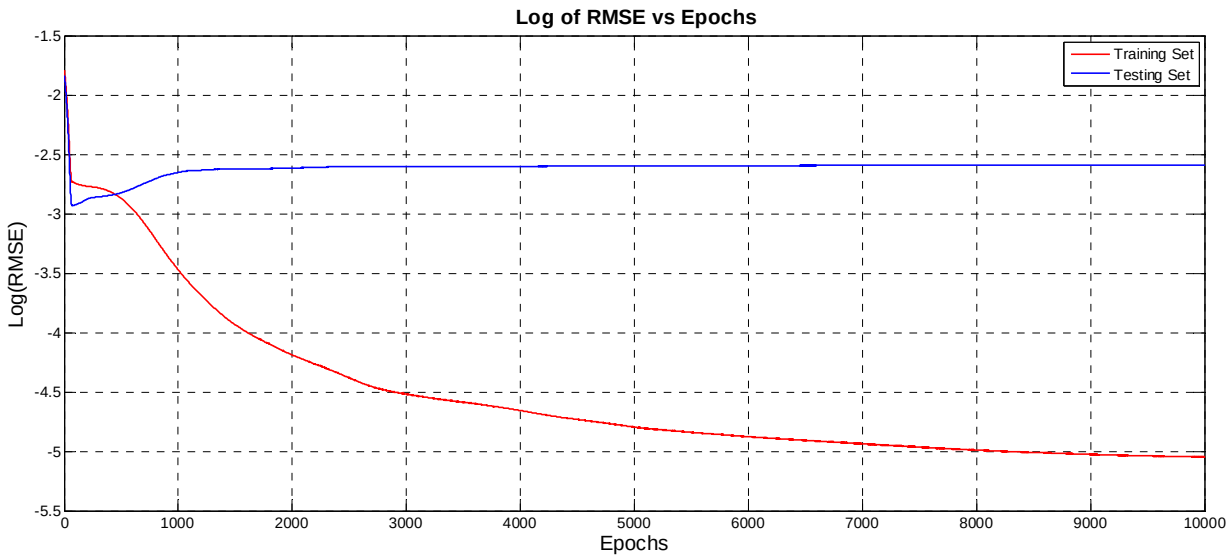

Figure C1: Log of RMSE after different numbers of epochs

C1.1.2 Conclusion – Part 1:

From the results obtained, the number of epochs used to compare different structures was 2000. This was chosen because it was important to allow the system to reach an over fitted stage, which implied the minimum error over the testing set of data was surpassed and captured. Since the minimum RMSE over the testing set was 0.0533, and the RMSE after 2 000 epochs was 0.0732, the minimum RMSE was obtain within the first 2 000 epochs. As this system was the largest with the slowest learning parameters, the other simpler structures would reach their minimum within this 2 000 epochs. This decision was primarily based on the results over the testing set, as the error over the training set would continue to decrease indefinitely, leading to over fitting.

C1.2 Part 2 – Determining efficient parameters

C1.2.1 Observations – Learning Rate Increase/Decrease

From table C3 it is clear the minimum MSE over the training set (2.85E-03) was achieved with a learning rate increase of 1.05 (5% increase) and a decrease of 0.8 (20% decrease). The minimum MSE over the testing set (2.79E-03) was achieved with an increase of 1.005 (0.5% increase) and a decrease of 0.2 (80% decrease). These observations did not support the same conclusion.

The shape of the average MSE surface over the training set was considered. It was observed the range over the training set surface was significant (1.05E-03) in comparison to the range of the average MSE surface over the testing set (0.03E-03). The efficient learning rate changes were determined by calculating the smallest trade off in error over both sets of data.

The efficient learning rate increase and decrease (represented by orange in table C3) were 1.05 and 0.8 as this resulted in the minimum error over the training data set (represented by green in table C3) and an increase in error by 0.03E-03 over the testing data set.

It is important to note the values used to decide on the learning rate changes were the arithmetic average MSE over different combinations of input and hidden neurons. This average did not explicitly allow for the event of outliers in the data and may be skewed. From the investigations there were no outliers found but extensive tests to verify this remark were not performed. Further investigation into the possibility of outliers can be done.

Training Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	3.90E-03	3.24E-03	3.29E-03
	0.5	3.58E-03	3.09E-03	3.13E-03
	0.8	3.10E-03	**2.85E-03**	2.95E-03
	min	2.85E-03		
Testing Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	2.79E-03	2.80E-03	2.81E-03
	0.5	2.80E-03	2.81E-03	2.81E-03
	0.8	2.81E-03	**2.82E-03**	2.82E-03
	min	2.79E-03		
	Minimum			
	Efficient			

Tables C3: Parameters for experiment - Part 1

C1.2.2 Conclusion – Learning Rate Increase/Decrease

The following decisions were made:

- A learning rate decrease of 0.8 was to be used, i.e. a 20% decrease in learning rate if the error of the system increases during training.
- A learning rate increase of 1.05 was to be used, i.e. a 5% increase in learning rate if the error of the system decreases during training.

C1.2.3 Observations – Input and Hidden Neurons – Training Set's MSE surface

The error surface in figure C2 indicated the MSE of the system decreased as the number of input and hidden neurons increased. This was expected since the increased complexity of the system would increase the accuracy over the training data set. The upper and lower surface represents the upper and lower limit of the MSE. The limits were determined by increasing/decreasing the best estimate surface by two standard deviations of the MSE. Considering the best estimate error surface, figure C3, it was clear the MSE of the system decreased as the complexity of the system increased. The surface obtained the minimum at 36 input and 64 hidden neurons (2.26E-04).

The upper and lower limit of the MSE surface over the training set requires assumptions in order to hold statistical significance. There was insufficient information available to make these assumptions. These limits, however, did provide a good estimate of the MSE limits for the randomly initialized structures.

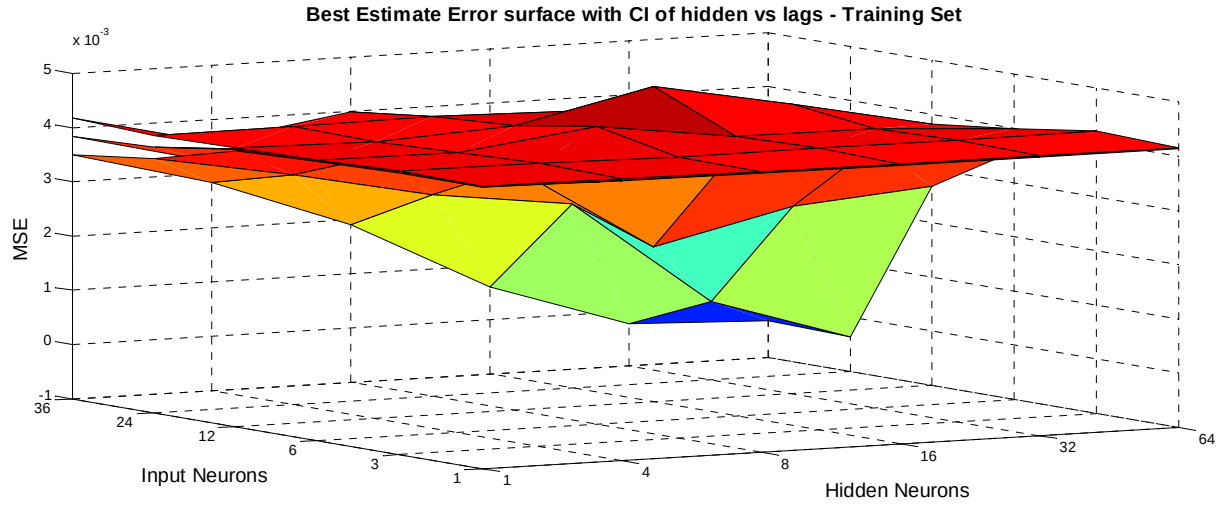

Figure C2: MSE with CI surface – Training data set

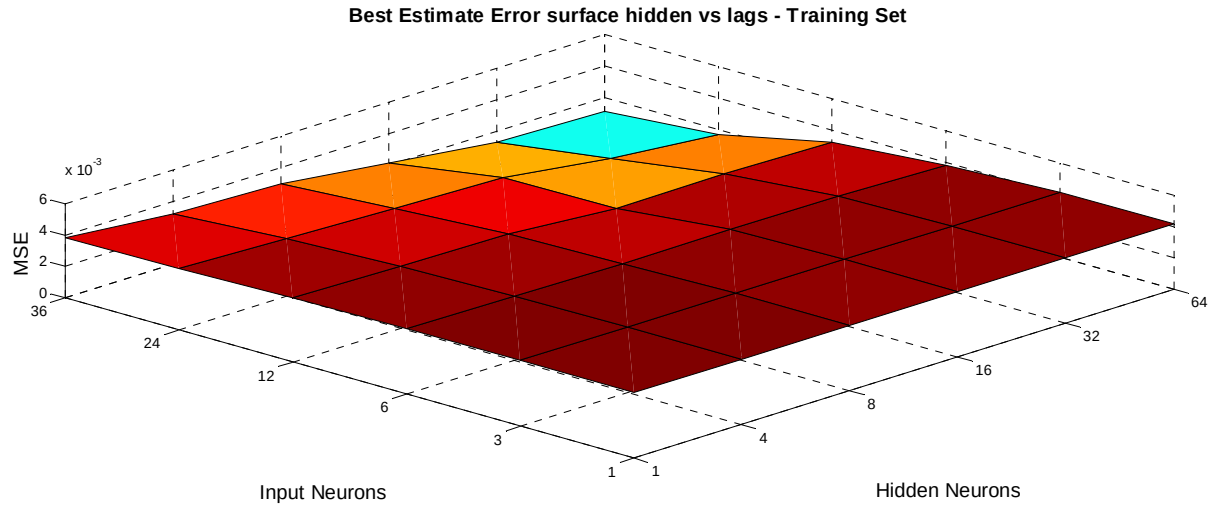

Figure C3: MSE surface – Training data set

C1.2.4 Observations – Input and Hidden Neurons – Testing Set's MSE surface

The testing set surface (figure C4 and C5) indicated a minimum plain from 3 to 6 input neurons and 4 to 16 hidden neurons, after which it increased. This suggested over fitting was prevalent in this system and that complex structures were likely to over fit the data. The minimum MSE (2.74E-03) on the best estimate surface (figure C5) was achieved using 3 input and 16 hidden neurons. The MSE surface over the testing set was structurally different to that of the training set.

The upper and lower limit of the MSE surface over the testing set requires assumptions in order to hold statistical significance. There was insufficient information available to make these assumptions. These limits, however, did provide a good estimate of the MSE limits for the randomly initialized structures.

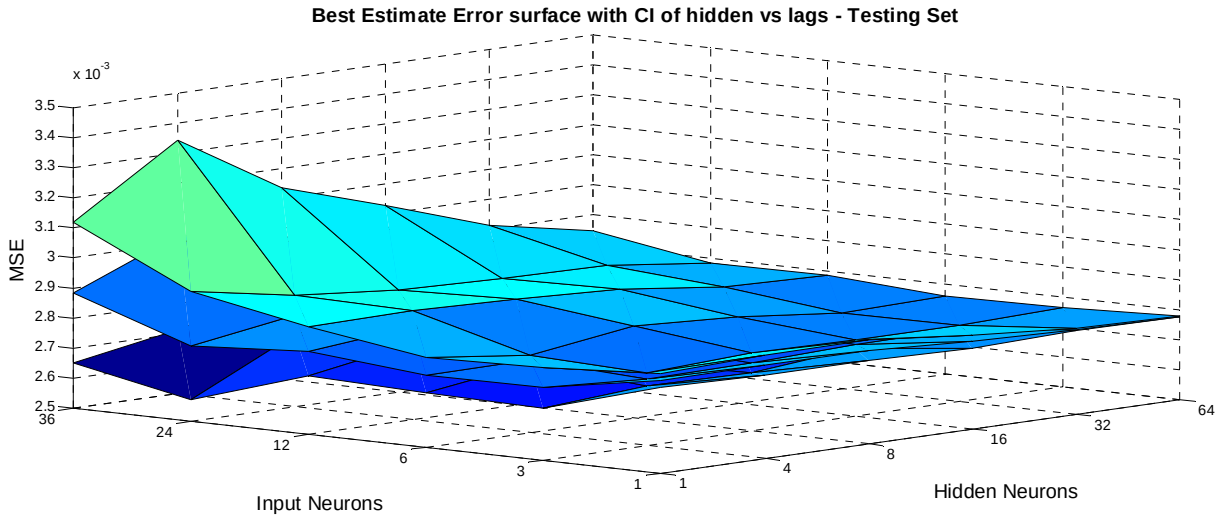

Figure C4: MSE with CI surface – Testing data set

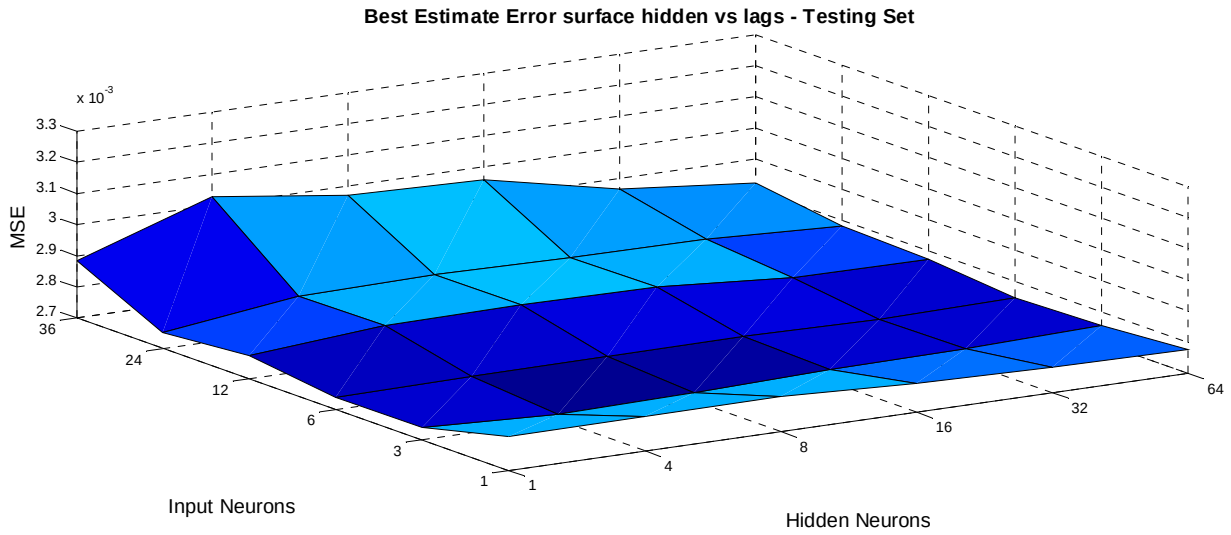

Figure C5: MSE surface – Testing data set

C1.2.5 Numerical Representation of MSE surfaces

The tables below provide the numerical average MSE values used to construct the error surfaces in figures C2, C3, C4 and C5.

Training Sets							
		LR increase		1.05			
		LR decrease		0.8	(20% decrease)		
	Average MSE vs Input and Hidden						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	4.20E-03	4.17E-03	4.16E-03	4.14E-03	4.13E-03	4.11E-03
	3	4.20E-03	4.12E-03	3.97E-03	3.77E-03	3.83E-03	3.70E-03
	6	4.13E-03	3.88E-03	3.59E-03	**2.98E-03**	3.04E-03	2.43E-03
	12	3.99E-03	3.28E-03	2.63E-03	2.21E-03	1.76E-03	8.24E-04
	24	3.95E-03	2.66E-03	1.74E-03	1.10E-03	8.46E-04	4.31E-04
	36	3.56E-03	2.38E-03	1.22E-03	7.61E-04	5.06E-04	2.26E-04
average	2.85E-03						
min	2.26E-04						
	Upper limit MSE vs Input and Hidden (+2 STD Dev)						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	4.22E-03	4.21E-03	4.18E-03	4.17E-03	4.16E-03	4.13E-03
	3	4.23E-03	4.27E-03	4.14E-03	4.27E-03	4.00E-03	3.78E-03
	6	4.24E-03	4.13E-03	3.78E-03	**3.39E-03**	3.25E-03	3.07E-03
	12	4.20E-03	3.45E-03	3.05E-03	2.59E-03	2.17E-03	1.74E-03
	24	4.06E-03	3.07E-03	2.22E-03	1.33E-03	1.55E-03	6.90E-04
	36	3.89E-03	3.03E-03	1.66E-03	9.78E-04	6.07E-04	3.30E-04
average	3.12E-03						
Efficient Structure							
Minimum Error							

Table C4: MSE of different combinations of input and hidden neurons – Training data set

Testing Sets							
		LR increase		1.05			
		LR decrease		0.8	(20% decrease)		
	Average MSE vs Input and Hidden						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	2.80E-03	2.80E-03	2.80E-03	2.79E-03	2.78E-03	2.78E-03
	3	2.76E-03	0.002757	2.75E-03	2.74E-03	2.74E-03	2.75E-03
	6	2.76E-03	2.75E-03	2.74E-03	**2.76E-03**	2.74E-03	2.75E-03
	12	2.77E-03	2.84E-03	2.82E-03	2.81E-03	2.79E-03	2.78E-03
	24	2.84E-03	2.81E-03	2.84E-03	2.84E-03	2.82E-03	2.79E-03
	36	2.97E-03	3.02E-03	3.05E-03	3.04E-03	2.93E-03	2.85E-03
average	2.82E-03						
min	2.74E-03						
	Upper limit MSE vs Input and Hidden (+2 STD Dev)						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	2.85E-03	2.82E-03	2.82E-03	2.81E-03	2.79E-03	2.78E-03
	3	2.80E-03	0.002793	2.79E-03	2.75E-03	2.75E-03	2.76E-03
	6	2.80E-03	2.80E-03	2.77E-03	**2.79E-03**	2.76E-03	2.76E-03
	12	2.96E-03	2.94E-03	2.88E-03	2.83E-03	2.82E-03	2.79E-03
	24	3.02E-03	2.84E-03	2.88E-03	2.88E-03	2.85E-03	2.80E-03
	36	3.36E-03	3.18E-03	3.30E-03	3.20E-03	2.97E-03	2.88E-03
average	2.88E-03						
Efficient Structure							
Minimum Error							

Table C5: MSE of different combinations of input and hidden neurons – Testing data set

C1.2.6 Conclusion – Input and Hidden Neurons

In order to balance the error between the structures over the training and testing data sets, a structure was chosen which minimized the reduction of accuracy. The structure chosen had 6 input and 16 hidden neurons. The increase in MSE from its minimum over the training and testing data sets were 2.754E-03 and 0.002E-03, respectively.

The choice made here had subjective components. It was important not to base the decision purely on the results over the training set as this would lead to over fitting. Similarly, basing the decision purely on the results over the testing set would lead to the fitting of the ANN to the testing set. Both the scenarios described above would have reduced the generalization ability of the ANN.

C1.2.7 Conclusion of Determining an Efficient Structure

Efficient parameters were:

- 6 input neurons,
- 16 hidden neurons,
- A learning rate increase of 1.05 (5% increase), and
- A learning rate decrease of 0.8 (20% decrease).

C2 Analysing the Efficient ANN

The ANN was trained with the following parameters.

Training set size	325
Testing set size	100
Input/lagged neurons	6
Hidden neurons	16
Number of initialized structures	10
Epochs	10 000
Training algorithm	RPROP
Learning rate:	
Increase	1.05 (5% increase)
Decrease	0.8 (20% decrease)
Maximum	100
Minimum	0.000000001

Table C6: Parameters for the efficient ANN structure

C2.1 Observations of System Error

Figure C6 indicated the system was resilient to over fitting as shown by the small increase of the error over the testing set throughout the training process. It was decided the efficient number of training epochs would consider the error over both the testing and training sets. To avoid any over fitting 49 epochs was chosen as the efficient training period. Further, this number provided a good balance of error reduction over both sets of data while not requiring additional computational power.

Error Readings	Testing Set	Training Set
Minimum RMSE	0.0525	0.0614
Minimum MSE	0.0028	0.0038
Final RMSE after training *(49 epochs)*	0.0525	0.0652
Final MSE after training *(49 epochs)*	0.0028	0.0042

Table C7: Results of efficient model after training

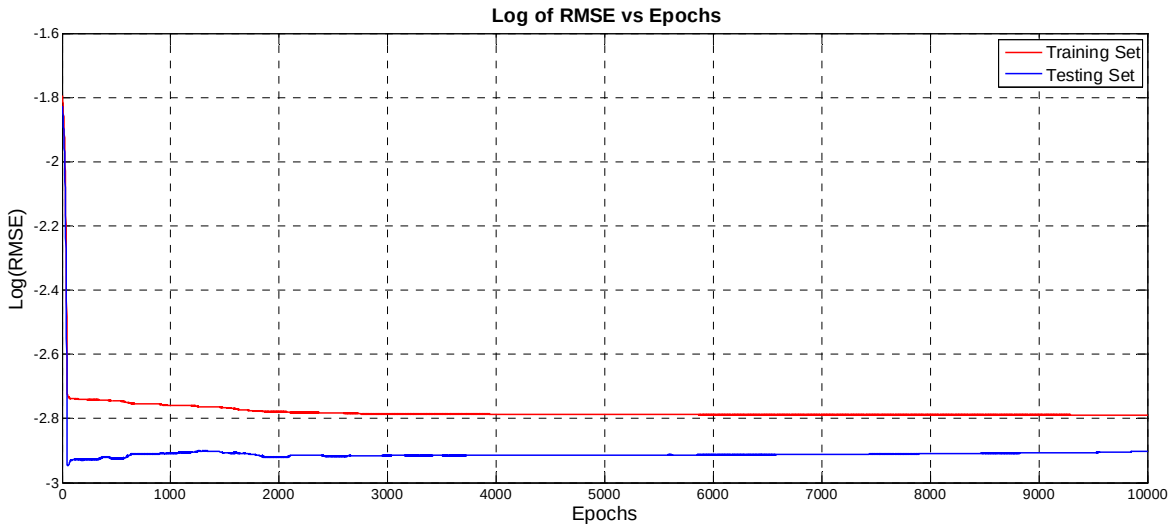

Figure C6: Log of RMSE after 10 000 training epochs

C2.2 Observations of Forecasts and Actuals

The ANN captured a limited number of underlying trends in the data set as expected. The predictions were less representative of the data as the other cases considered. This was due external factors having a greater effect, than previous observations, on the return. It was possible to over fit the ANN to this data set due to these external factors. The best estimate over the training and testing sets was the blue line in figure C7

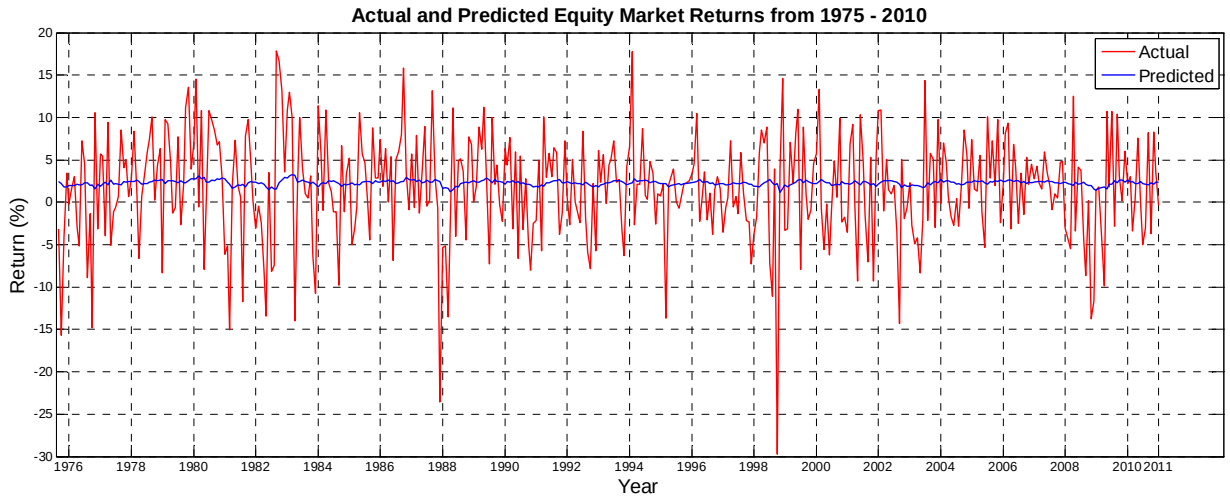

Figure C7: Actual and Predicted – Training and Testing Data Sets

C2.3 Conclusion - Analysing the Efficient ANN

The ANN captured a limited number of trends underlying the data. The efficient ANN structure (after 49 training epochs) had an MSE over the training and testing sets of 0.0042 and 0.0028, respectively. The ANN performed similarly over the testing and training data set. The level of effectiveness was determined by comparing these results to traditional models in a following chapter (Chapter 5). The minimum RMSE over the testing set, of 0.0525, and over the training set, of 0.0614, was obtained after approximately 49 and 10 000 epochs, respectively. The RMSE of the training set decreased as the training went over 49 epochs and continued to decrease until the maximum number of epochs were obtained. Theoretically, this was expected as ANNs have the ability to mimic a data set precisely. This mimicking leads to over fitting and reduces the system's generalization ability. To avoid over fitting but to ensure the underlying relationships in the data were captured, decisions relating to efficiency were made with reference to the performance of the ANN over both the testing and training data set.

C3 Summary of Experiment

The following table provides a summary of the results from this experiment.

Structure of ANN	6-16-1
Training epoch required for efficient training	49
Minimum Root Mean Squared Error:	
Training Data Set	0.0614
Testing Data Set	0.0525
Final Root Mean Squared Error (49 epochs):	
Training Data Set	0.0652
Testing Data Set	0.0525
Parameters for RPROP:	
Learning Rate Increase	5% (1.05)
Learning Rate Decrease	20% (0.8)
Data Sets:	
Training set	325 observations (1975 – 2002)
Testing set	100 observations (2002 – 2010)
Times the structure was initialized	10

Table C8: Summary of experiment's result

Appendix D – Isolated Inflation ANN – Three-Month Forecast Results

D1 Determining an Efficient Structure

D1.1 Part 1 – Efficient number of training epochs

The parameters used for this experiment were:

Parameter	Value
Network Structure	36-64-1
Training set size	293
Testing set size	100
Dummy variables (for seasonality)	2
Number of initialized structures	5
Epochs	5 000
Training algorithm	RPROP
Learning rate change: Increase Decrease Maximum Minimum	 0.5% (1.005) 20% (0.8) 100 0.000000001

Table D1: Parameters for experiment - Part 1

D1.1.1 Observations – Part 1

Figure D1 indicated the minimum RMSE (0.0054) over the testing set was achieve within the first 1 000 epochs. The RMSE increased after this point which indicated the system was over fitting the ANN to this particular data set. The minimum RMSE (0. 0039) over the training set was achieve at the 5 000 epoch mark, as expected.

Since the ANN over fitted the data, it indicated there was noise in this data set. This randomness was not explainable through past observations of inflation. This supported theory, stipulating that inflation is largely affected by external factors, such the country's monetary policy. Further, it is expected that as the forecast period increases the accuracy associated with forecasts will decrease, since the distant future is less predictable than the near future.

		Training Epochs				
	Min RMSE	**500**	**1 000**	**5 000**	**Δ in RMSE – 500 to 1 000**	**Δ in RMSE – 1 000 to 5 000**
RMSE – Testing Set	0.0054	0.0056	0.0057	0.0069	-0.0001	-0.0012
RMSE – Training Set	0.0039	0.0051	0.0049	0.0039	0.0012	0.001

Table D2: RMSE after different numbers of epochs

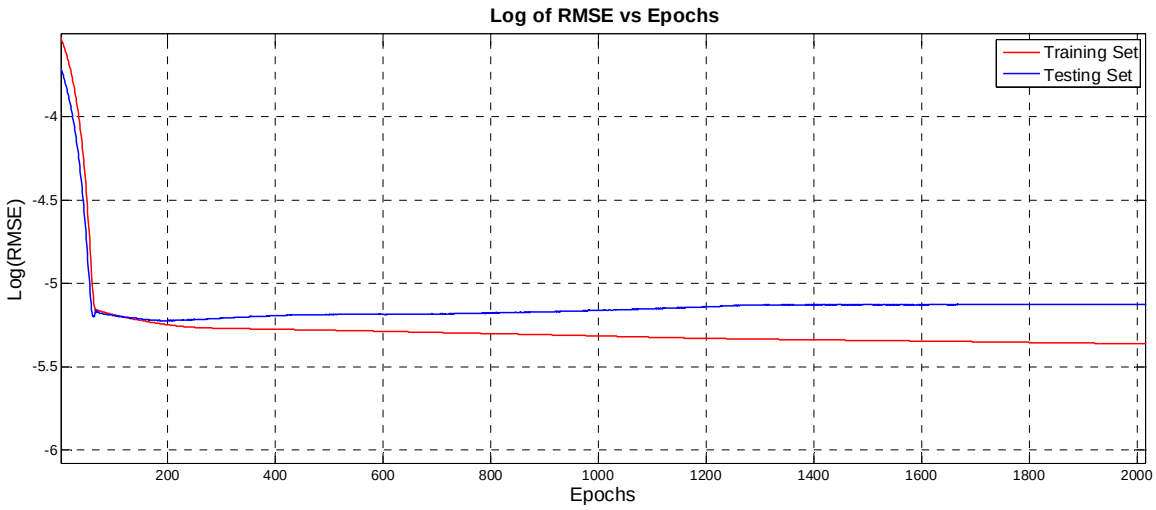

Figure D1: Log of RMSE after different numbers of epochs

D1.1.2 Conclusion – Part 1:

From the results obtained, the number of epochs used to compare different structures was 1 000. This was chosen because it was important to allow the system to reach an over fitted stage, which implied the minimum error over the testing set of data was surpassed and captured. Since the minimum RMSE over the testing set was 0.0054, and the RMSE after 1 000 epochs was 0.0057, the minimum RMSE was obtain within the first 1 000 epochs. This is supported by figure D1. As this system was the largest with slowest learning parameters, the other simpler structures would reach their minimum within 1 000 epochs. This conclusion was primarily based on the results over the testing set, as the error over the training set would continue to decrease indefinitely, leading to over fitting.

D1.2 Part 2 – Determining efficient parameters

D1.2.1 Observations – Learning Rate Increase/Decrease

From table D3 it is clear the minimum MSE over the training set (2.71E-05) was achieved with a learning rate increase of 1.05 (5% increase) and a decrease of 0.8 (20% decrease). The minimum MSE over the testing set (2.92E-05) was

achieved with an increase of 1.005 (0.5% increase) and a decrease of 0.2 (80% decrease). These observations did not support the same conclusion.

The shape of the average MSE surface over the training set was considered. It was observed that the range over the surface was significant (0.31E-05) in comparison to the range of the average MSE surface over the testing set (0.06E-5). The efficient learning rate changes were chosen by determining the smallest trade off in error over both sets of data.

The efficient learning rate increase and decrease (represented by orange in table D3) were 1.05 and 0.8 as this resulted in an increase in the minimum MSE (represented by green in table D3) over the training set and testing set of 0 and 0.02E-05, respectively.

It is important to note the values used to decide on the learning rate changes were the arithmetic average MSE over different combinations of input and hidden neurons. This average did not explicitly allow for the event of outliers in the data and may be skewed. From the investigations there were no outliers found but extensive tests to verify this remark were not performed. Further investigation into the possibility of outliers can be done.

Training Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	2.89E-05	2.96E-05	3.02E-05
	0.5	2.86E-05	2.89E-05	2.94E-05
	0.8	2.79E-05	**2.71E-05**	2.78E-05
	min	2.71E-05		
Testing Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	2.96E-05	2.95E-05	2.97E-05
	0.5	2.94E-05	2.94E-05	2.98E-05
	0.8	2.92E-05	**2.94E-05**	2.96E-05
	min	2.92E-05		
	Minimum			
	Efficient			

Tables D3: MSE for Learning Rate Changes

D1.2.2 Conclusion – Learning Rate Increase/Decrease

The following decisions were made:

- A learning rate decrease of 0.8 was to be used, i.e. a 20% decrease in learning rate if the error of the system increases during training.
- A learning rate increase of 1.05 was to be used, i.e. a 5% increase in learning rate if the error of the system decreases during training.

D1.2.3 Observations – Input and Hidden Neurons – Training Set's MSE surface

The error surface in figure D2 indicated the MSE of the system decreased as the number of input and hidden neurons increased. This was expected since the increased complexity of the system would increase the accuracy over the training data set.

The upper and lower surface represents the upper and lower limit of the MSE. The limits were determined by increasing/decreasing the best estimate surface by two standard deviations of the MSE.

Considering the best estimate error surface, figure D3, it was clear the MSE of the system decreased as the complexity of the system increased. The surface obtained the minimum at 36 input and 64 hidden neurons (1.59E-05). This general trend was as expected.

The upper and lower limit of the MSE surface over the training set requires assumptions in order to hold statistical significance. There was insufficient information available to make these assumptions. These limits, however, did provide a good estimate of the MSE limits for the randomly initialized structures.

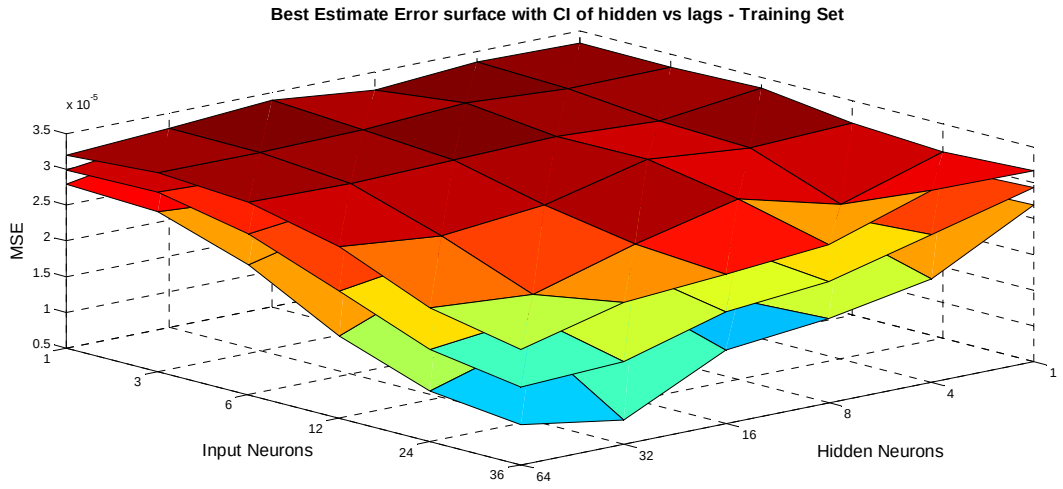

Figure D2: MSE with CI surface – Training data set

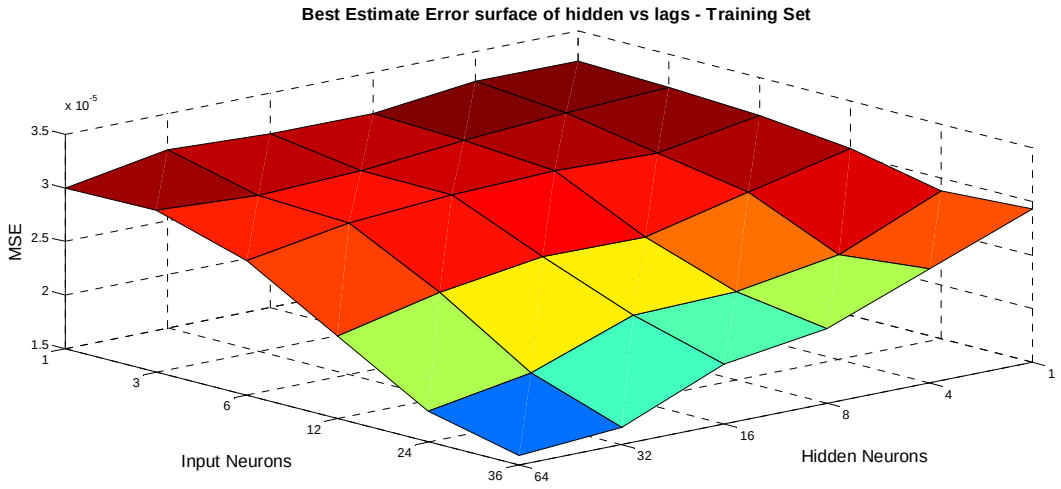

Figure D3: MSE surface – Training data set

D1.2.4 Observations – Input and Hidden Neurons – Testing Set's MSE surface

The testing set surface (figure D4 and D5) indicated a levelling off of the MSE after 32 hidden neurons and all number of input neurons. A valley in the surface was detected when 12 input neurons were used. The minimum MSE on the best estimate surface (figure D5) was achieved using 12 input and 64 hidden neurons (2.72E-05). The decrease in error was minimal from 12 input and 32 hidden neurons to 12 input to 64 hidden neurons (0.05E-05).

The upper and lower limit of the MSE surface over the training set requires assumptions in order to hold statistical significance. There was insufficient information available to make these assumptions. These limits, however, did provide a good estimate of the MSE limits for the randomly initialized structures.

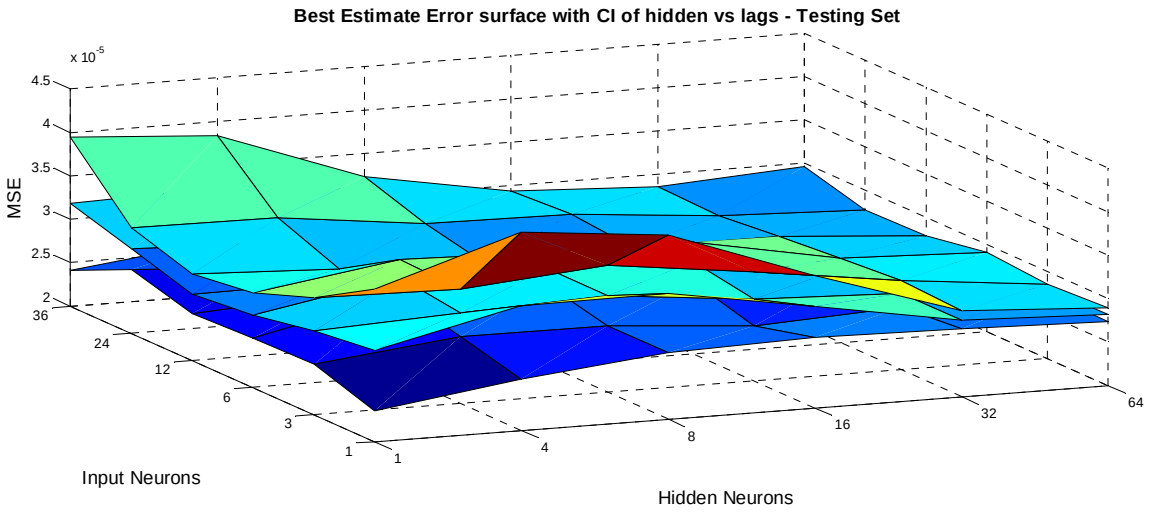

Figure D4: MSE with CI surface – Testing data set

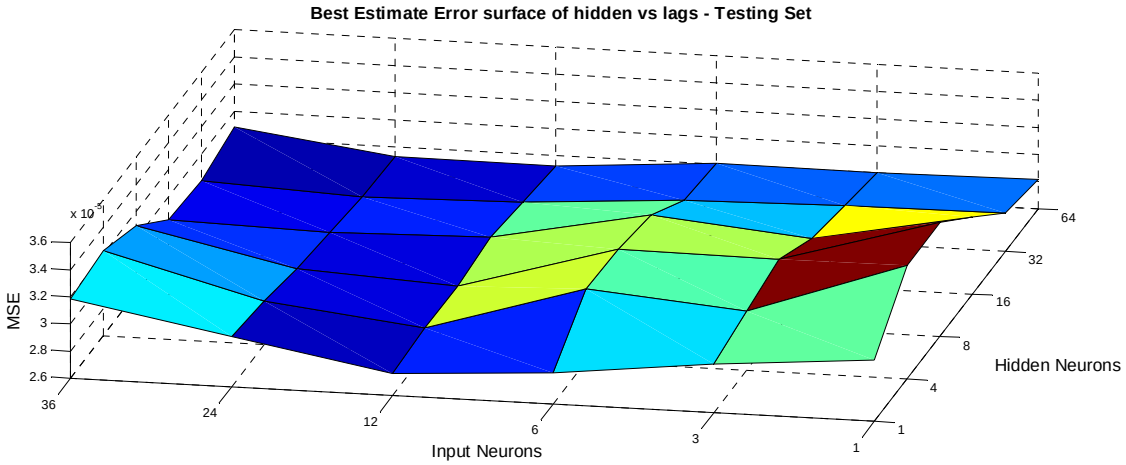

Figure D5: MSE surface – Testing data set

D1.2.5 Numerical Representation of MSE surfaces

The tables below provide the numerical average MSE values used to construct the error surfaces in figures D2, D3, D4 and D5.

Training Sets							
		LR increase		1.05			
		LR decrease		0.8	(20% decrease)		
	Average MSE vs Input and Hidden						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	3.22E-05	3.23E-05	3.12E-05	3.12E-05	3.16E-05	3.00E-05
	3	3.19E-05	3.15E-05	3.08E-05	2.99E-05	2.96E-05	3.01E-05
	6	3.15E-05	2.99E-05	3.02E-05	2.98E-05	2.91E-05	2.76E-05
	12	3.06E-05	2.84E-05	2.61E-05	2.63E-05	**2.48E-05**	2.27E-05
	24	2.88E-05	2.48E-05	2.32E-05	2.30E-05	1.95E-05	1.79E-05
	36	2.93E-05	2.56E-05	2.20E-05	2.06E-05	1.66E-05	1.59E-05
average	2.71E-05						
min	1.59E-05						
	Upper limit MSE vs Input and Hidden (+2 STD Dev)						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	3.33E-05	3.35E-05	3.25E-05	3.39E-05	3.29E-05	3.20E-05
	3	3.31E-05	3.30E-05	3.39E-05	3.31E-05	3.27E-05	3.28E-05
	6	3.35E-05	3.17E-05	3.25E-05	3.21E-05	3.16E-05	3.19E-05
	12	3.18E-05	3.13E-05	3.26E-05	2.90E-05	**2.76E-05**	2.90E-05
	24	3.11E-05	2.66E-05	3.03E-05	2.69E-05	2.27E-05	2.37E-05
	36	3.17E-05	3.19E-05	2.72E-05	2.59E-05	2.49E-05	2.12E-05
average	3.04E-05						
Efficient Structure							
Minimum Error							

Table D4: MSE of different combinations of input and hidden neurons – Training data set

Testing Sets							
		LR increase		1.05			
		LR decrease		0.8	(20% decrease)		
	Average MSE vs Input and Hidden						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	3.05E-05	3.44E-05	3.44E-05	3.17E-05	2.88E-05	2.82E-05
	3	2.96E-05	3.04E-05	3.11E-05	2.94E-05	2.87E-05	2.80E-05
	6	2.83E-05	3.14E-05	3.12E-05	3.06E-05	2.85E-05	2.81E-05
	12	2.76E-05	2.78E-05	2.78E-05	2.83E-05	2.77E-05	2.72E-05
	24	2.97E-05	2.92E-05	2.85E-05	2.80E-05	2.75E-05	2.73E-05
	36	3.18E-05	3.23E-05	3.09E-05	2.83E-05	2.80E-05	2.88E-05
average	2.94E-05						
min	2.72E-05						
	Upper limit MSE vs Input and Hidden (+2 STD Dev)						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	3.76E-05	4.28E-05	4.12E-05	3.52E-05	2.99E-05	2.90E-05
	3	3.34E-05	3.31E-05	3.46E-05	3.25E-05	3.03E-05	2.86E-05
	6	3.09E-05	3.34E-05	3.30E-05	3.28E-05	2.97E-05	2.91E-05
	12	2.99E-05	2.93E-05	2.85E-05	2.92E-05	2.91E-05	2.79E-05
	24	3.21E-05	3.20E-05	3.01E-05	2.98E-05	2.84E-05	2.77E-05
	36	3.95E-05	3.84E-05	3.23E-05	2.94E-05	2.86E-05	2.96E-05
average	3.19E-05						
Efficient Structure							
Minimum Error							

Table D5: MSE of different combinations of input and hidden neurons – Testing data set

D1.2.6 Conclusion – Input and Hidden Neurons

The observations over the testing and training set suggested two conflicting structures. To satisfy both data sets an average structure was chosen. This required compensation in accuracy over both the training and testing data sets. The structure chosen was 12 input and 32 hidden neurons, which had an MSE of 2.48E-05 over the training set and 2.77E-05 over the testing set.

The choice made here had subjective components. It was important not to base the decision purely on the results over the training set as this would lead to over fitting. Similarly, basing the decision purely on the results over the testing set would lead to the fitting of the ANN to the testing set. Both the scenarios described above would have reduced the generalization ability of the ANN. Thus the subjective approach followed was most appropriate.

D1.2.7 Conclusion of Determining an Efficient Structure

Efficient parameters were:

- 12 input neurons,
- 32 hidden neurons,
- A learning rate increase of 1.05 (5% increase), and
- A learning rate decrease of 0.8 (20% decrease).

D2 Analysing the Efficient ANN

The ANN was trained with the following parameters.

Training set size	317
Testing set size	100
Dummy variables (for seasonality)	2
Input/lagged neurons	12
Hidden neurons	32
Number of initialized structures	10
Epochs	5 000
Training algorithm	RPROP
Learning rate:	
Increase	1.05 (5% increase)
Decrease	0.8 (20% decrease)
Maximum	100
Minimum	0.000000001

Table D6: Parameters for the efficient ANN structure

D2.1 Observations of System Error

The error over the testing set began off large and decreased substantially thereafter as indicated in the plot of the error over the training process, figure D6. After the initial decreases the error over the testing set began to oscillate as the training continued, but there was no increase in error. Over fitting was not prevalent in the ANN as seen from the stable error over the testing set. The minimum RMSE (0.0052) over the testing set was achieved after approximately 2 000 epochs. The error over the training set continued to decrease over the training process and reached the minimum after 5 000 epochs. The number of epochs chosen to train the system was 4 999 as the oscillation of the error over the testing set was at its minimum. Further, the error over the training set was at its minimum. This number of epochs balanced both the error over the training and testing set. The numerical error value at points of interest are provided in table D7.

Error Readings	Testing Set	Training Set
Minimum RMSE	0.0052	0.0049
Minimum MSE	2.6912e-005	2.4141e-005
Final RMSE after training *(4 999 epochs)*	0.0052	0.0049
Final MSE after training *(4 999 epochs)*	2.7368e-005	2.4141e-005

Table D7: Results of efficient model after training

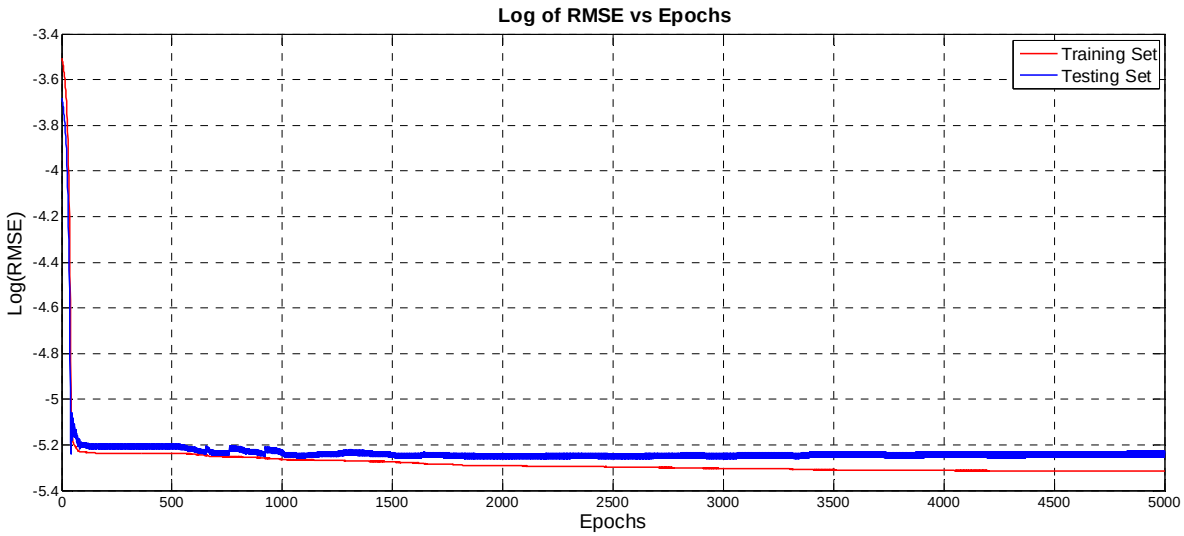

Figure D6: Log of RMSE after 5 000 training epochs

D2.2 Observations of Forecasts and Actuals

The ANN captured the underlying trends in the data set as expected. There were certain periods where the system underestimated or overestimated inflation. This was explained by the effect of external influences on inflation. The best estimate over the training and testing sets was represented by the blue line in figure D7.

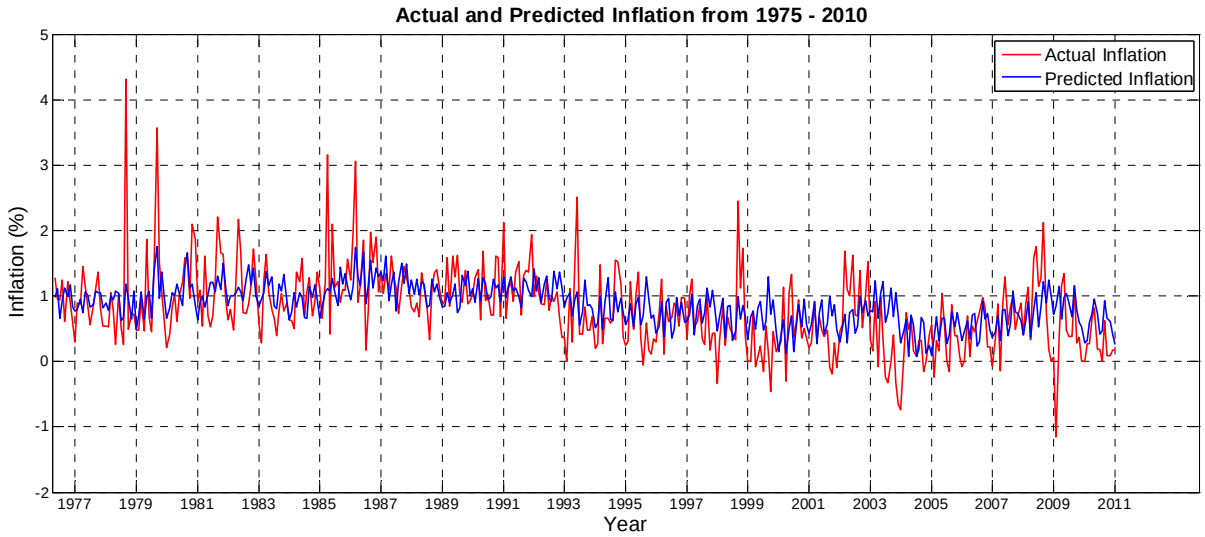

Figure D7: Actual and Predicted – Training and Testing Data Sets

D2.3 Conclusion - Analysing the Efficient ANN

The ANN captured the trends underlying the data. The efficient ANN structure (after 4 999 training epochs) had an MSE over the training and testing sets of 2.4141e-005 and 2.7368e-005, respectively. This was obtained after 4 999 training epochs, which was viewed as an efficient training period. The ANN performed similarly over the testing and training data set. The level of effectiveness was determined by comparing these results to traditional models in a following chapter (Chapter 5). The minimum RMSE (0.0052) over the testing set was obtained after approximately 2 000 training epochs. The RMSE over the training set decreased as the training went over 2 000 epoch and continued to decrease until the maximum number of epochs were obtained. Theoretically, this was expected as ANNs have the ability to mimic a data set precisely. This mimicking leads to over fitting and reduces the systems generalization ability. To avoid over fitting but to ensure the underlying relationships in the data were captured, decisions relating to efficiency were made with reference to the performance of the ANN over both the testing and training data set.

D3 Summary of Experiment

The following table provides a summary of the results from this experiment.

Structure of ANN	12-32-1
Training epoch required for optimal training	4 999
Minimum Root Mean Squared Error:	
Training Data Set	0.0049
Testing Data Set	0.0052
Final Root Mean Squared Error (4 999 epochs):	
Training Data Set	0.0049
Testing Data Set	0.0052
Parameters for RPROP:	
Learning Rate Increase	5% (1.05)
Learning Rate Decrease	20% (0.8)
Data Sets:	
Training set	317 observations (1976 – 2002)
Testing set	100 observations (2002 – 2010)
Times the structure was initialized	10

Table D8: Summary of experiment's result

Appendix E – Isolated Bond Market ANN – Three-Month Forecast Results

E1 Determining an Efficient Structure

E1.1 Part 1 – Efficient number of training epochs

The parameters used for this experiment were:

Parameter	Value
Network Structure	36-64-1
Training set size	293
Testing set size	100
Number of initialized structures	5
Epochs	50 000
Training algorithm	RPROP
Learning rate change: Increase Decrease Maximum Minimum	 0.5% (1.005) 20% (0.8) 100 0.000000001

Table E1: Parameters for experiment - Part 1

E1.1.1 Observations – Part 1

Figure E1 indicated the minimum RMSE (0.0193) over the testing set was achieved within the first 2 000 epochs. The RMSE increased after this point which indicated the system was over fitting the ANN to this particular data set. The minimum RMSE (0.0034) over the training set was achieved at the 50 000 epoch mark, as expected. The RMSE of 0.0193 over the testing set indicated that historically there were limited relationships between the successive return on the bond market. This was expected as external factors had a significant impact on the observations. As the minimum over the testing set was obtained over the first 2 000 epochs, it was decided to use this number when different ANN structures were compared. This would allow every structure to have reached a minimum MSE before they were compared.

		Training Epochs				
	Min RMSE	**100**	**2 000**	**50 000**	**Δ in RMSE – 100 to 2 000**	**Δ in RMSE – 2 000 to 50 000**
RMSE – Testing Set	0.0193	0.0194	0.0233	0.0255	0.0264	-0.0039
RMSE – Training Set	0.0034	0.0268	0.0135	0.0055	0.0034	0.0133

Table E2: RMSE after different numbers of epochs

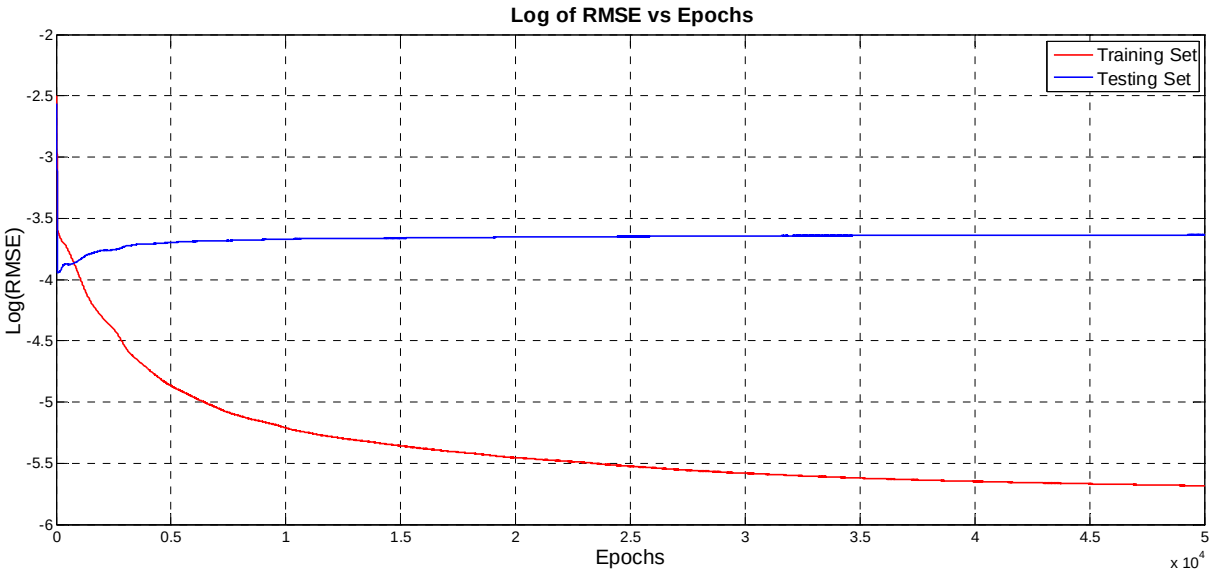

Figure E1: Log of RMSE after different numbers of epochs

E1.1.2 Conclusion – Part 1:

From the results obtained, the number of epochs used to compare different structures was 2 000. This was chosen because it was important to allow the system to reach an over fitted stage, which implied the minimum error over the testing set of data was surpassed and captured. Since the minimum RMSE over the testing set was 0.0193, and the RMSE after 2 000 epochs was 0.0233, the minimum RMSE was obtain within the first 2 000 epochs. This was supported by figure E1. As this system was the largest with the slowest learning parameters, the other simpler structures were expected to reach their minimums within the 2 000 epochs. This conclusion was primarily based on the results over the testing set, as the error over the training set would continue to decrease indefinitely, leading to over fitting.

E1.2 Part 2 – Determining efficient parameters

E1.2.1 Observations – Learning Rate Increase/Decrease

From table E3 it is clear the minimum MSE over the training set (5.25E-04) was achieved with a learning rate increase of 1.05 (5% increase) and a decrease of 0.8 (20% decrease). The minimum MSE over the testing set (3.72E-04) was achieved with an increase of 1.1 (10% increase) and a decrease of 0.2 (80% decrease). These observations did not support the same conclusion.

The shape of the average MSE surface over the training set was considered. It was observed the range over the training set surface was significant (1.37E-04) in comparison to the range of the average MSE surface over the testing set (0.02E-04). The efficient learning rate changes were chosen by determining the smallest trade off in error over both sets of data.

The efficient learning rate increase and decrease (represented by orange in table E3) were 1.05 and 0.8 as this resulted in the minimum error over the training data set (represented by green in table E3) and an increase in error by 0.01E-04 over the testing data set.

It is important to note the values used to decide on the learning rate changes were the arithmetic average MSE over different combinations of input and hidden neurons. This average did not explicitly allow for the event of outliers in the data and may be skewed. From the investigations there were no outliers found but extensive tests to verify this remark were not performed. Further investigation into the possibility of outliers can be done.

Training Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	6.40E-04	5.81E-04	5.91E-04
	0.5	6.20E-04	5.68E-04	5.68E-04
	0.8	5.84E-04	**5.25E-04**	5.41E-04
	min	5.25E-04		
Testing Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	3.73E-04	3.72E-04	3.72E-04
	0.5	3.73E-04	3.73E-04	3.72E-04
	0.8	3.74E-04	**3.73E-04**	3.73E-04
	min	3.72E-04		
	Minimum			
	Efficient			

Table E3: MSE for Learning Rate Changes

E1.2.2 Conclusion – Learning Rate Increase/Decrease

The following decisions were made:

- A learning rate decrease of 0.8 was to be used, i.e. a 20% decrease in learning rate if the error of the system increases during training.
- A learning rate increase of 1.05 was to be used, i.e. a 5% increase in learning rate if the error of the system decreases during training.

E1.2.3 Observations – Input and Hidden Neurons – Training Set's MSE surface

The error surface in figure E2 indicated the MSE of the system decreased as the number of input and hidden neurons increased. This was expected since the increased complexity of the system would increase the accuracy over the training data set. The upper and lower surface represents the upper and lower limit of the MSE. The limits were determined by increasing/decreasing the best estimate surface by two standard deviations of the MSE.

Considering the best estimate error surface, figure E3, it was clear the MSE of the system decreased as the complexity of the system increased. The surface obtained the minimum at 36 input and 32 hidden neurons (4.06E-04). The minimum was expected to be achieve when 36 input and 64 hidden neurons were considered. The reason for this not being realised was likely due to random fluctuations in the initialization of the network's weights.

The upper and lower limit of the MSE surface over the training set requires assumptions in order to hold statistical significance. There was insufficient information available to make these assumptions. These limits, however, did provide a good estimate of the MSE limits for the randomly initialized structures.

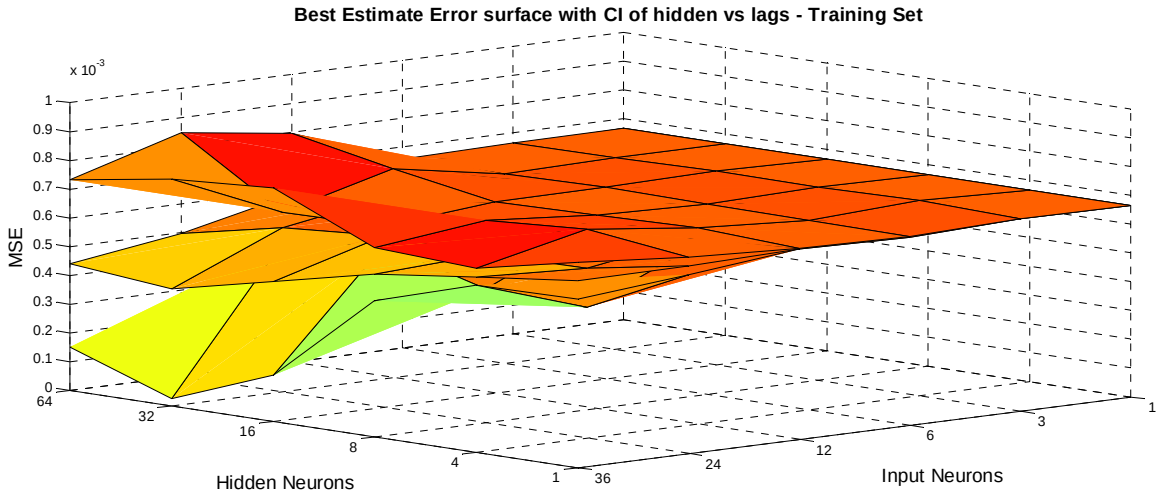

Figure E2: MSE with CI surface – Training data set

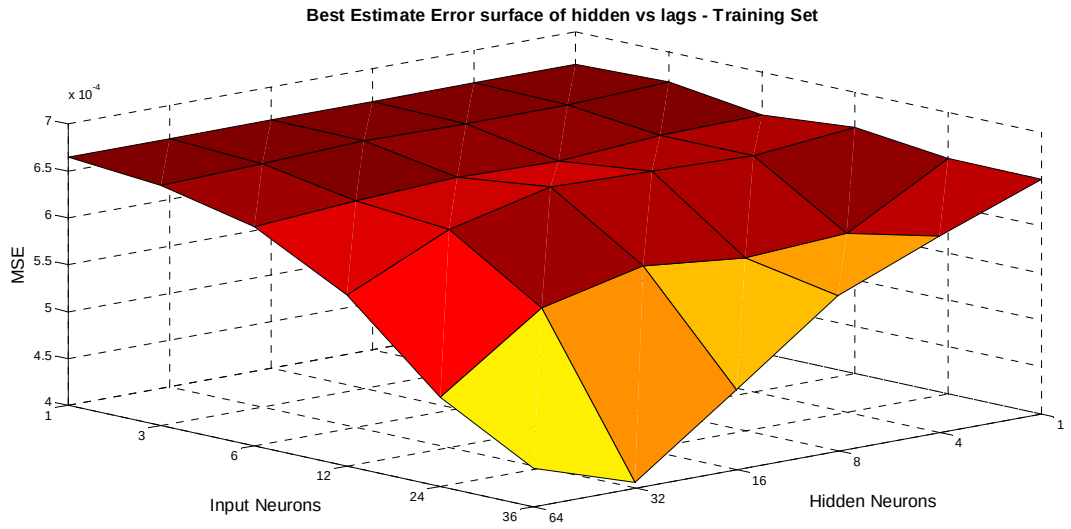

Figure E3: MSE surface – Training data set

E1.2.4 Observations – Input and Hidden Neurons – Testing Set's MSE surface

The MSE surface over the testing set with limits, figure E4, indicated a rough surface with no visible trends.

The minimum MSE on the best estimate surface, figure E5, was achieved using 24 input and 64 hidden neurons (3.67E-04). The MSE surface over the testing set was structurally different to that of the training set. Further, the best estimate MSE surface indicated two possible minimum troughs, these being when considering 3 or 24 input neurons and varying the hidden neurons.

The upper and lower limit of the MSE surface over the testing set requires assumptions in order to hold statistical significance. There was insufficient information available to make these assumptions. These limits, however, did provide a good estimate of the MSE limits for the randomly initialized structures.

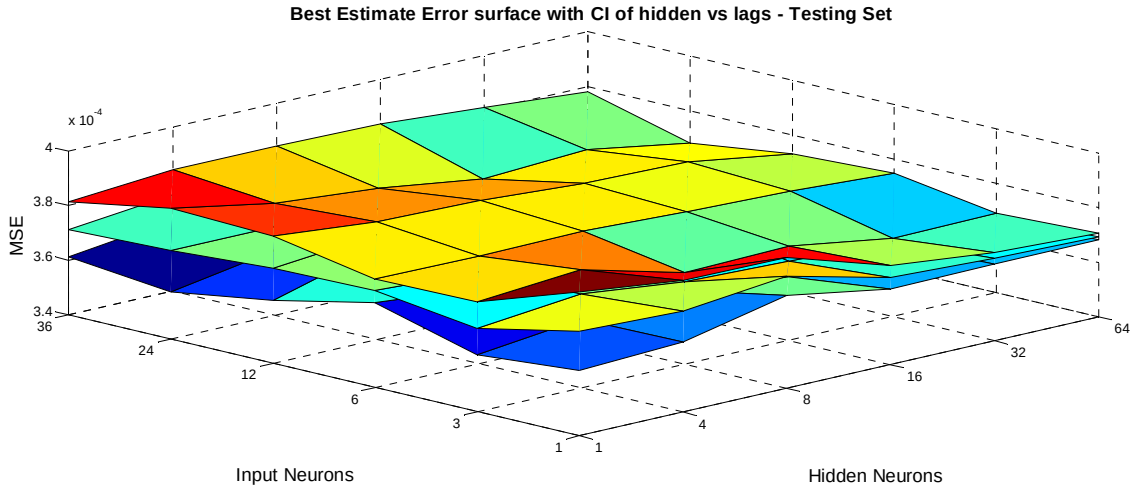

Figure E4: MSE with CI surface – Testing data set

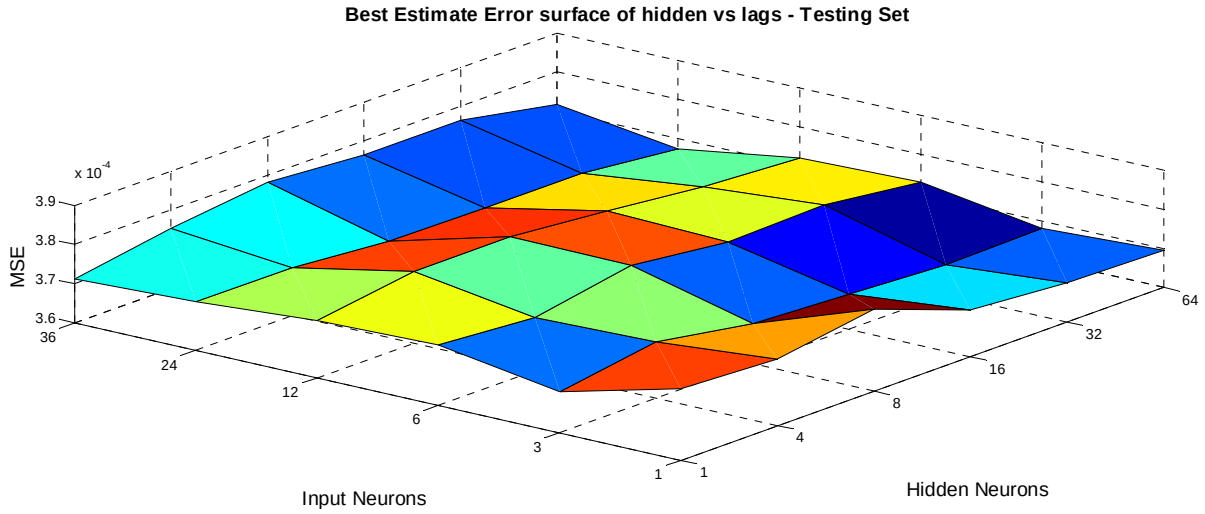

Figure E5: MSE surface – Testing data set

E1.2.5 Numerical Representation of MSE surfaces

The tables below provide the numerical average MSE values used to construct the error surfaces in figures E2, E3, E4 and E5.

Training Sets							
		LR increase		1.05			
		LR decrease		0.8	(20% decrease)		
	Average MSE vs Input and Hidden						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	6.65E-04	6.65E-04	6.65E-04	6.65E-04	6.65E-04	6.65E-04
	3	6.68E-04	6.63E-04	6.62E-04	6.64E-04	6.59E-04	6.56E-04
	6	6.54E-04	6.52E-04	6.44E-04	6.47E-04	6.41E-04	6.33E-04
	12	6.63E-04	6.52E-04	6.56E-04	6.59E-04	**6.33E-04**	5.82E-04
	24	6.51E-04	5.91E-04	5.84E-04	5.95E-04	5.71E-04	4.95E-04
	36	6.50E-04	6.10E-04	5.65E-04	4.85E-04	4.06E-04	4.41E-04
average	6.20E-04						
min	4.06E-04						
	Upper limit MSE vs Input and Hidden (+2 STD Dev)						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	6.68E-04	6.65E-04	6.65E-04	6.65E-04	6.65E-04	6.65E-04
	3	6.74E-04	6.70E-04	6.69E-04	6.91E-04	7.51E-04	6.24E-04
	6	6.71E-04	6.95E-04	6.87E-04	7.03E-04	6.60E-04	5.16E-04
	12	6.76E-04	6.87E-04	7.25E-04	7.23E-04	**6.08E-04**	5.45E-04
	24	6.45E-04	6.30E-04	7.10E-04	6.01E-04	5.38E-04	2.30E-04
	36	6.72E-04	6.74E-04	4.44E-04	4.01E-04	2.99E-04	9.61E-05
average	6.09E-04						
Efficient Structure							
Minimum Error							

Table E4: MSE of different combinations of input and hidden neurons – Training data set

Testing Sets							
		LR increase		1.05			
		LR decrease		0.8	(20% decrease)		
	Average MSE vs Input and Hidden						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	3.78E-04	3.77E-04	3.81E-04	3.72E-04	3.70E-04	3.69E-04
	3	3.70E-04	3.74E-04	3.70E-04	3.69E-04	3.68E-04	3.68E-04
	6	3.75E-04	3.74E-04	3.78E-04	3.75E-04	3.76E-04	3.73E-04
	12	3.75E-04	3.78E-04	3.78E-04	3.76E-04	**3.74E-04**	3.72E-04
	24	3.73E-04	3.72E-04	3.70E-04	3.70E-04	3.70E-04	3.67E-04
	36	3.71E-04	3.75E-04	3.78E-04	3.76E-04	3.76E-04	3.72E-04
average	3.73E-04						
min	3.67E-04						
	Upper limit MSE vs Input and Hidden (+2 STD Dev)						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	3.79E-04	3.84E-04	3.84E-04	3.76E-04	3.74E-04	3.71E-04
	3	3.82E-04	3.79E-04	3.77E-04	3.73E-04	3.72E-04	3.70E-04
	6	3.74E-04	3.80E-04	3.75E-04	3.77E-04	3.79E-04	3.75E-04
	12	3.79E-04	3.79E-04	3.78E-04	3.76E-04	**3.80E-04**	3.74E-04
	24	3.74E-04	3.84E-04	3.77E-04	3.73E-04	3.69E-04	3.70E-04
	36	4.45E-04	4.12E-04	3.86E-04	3.92E-04	3.76E-04	3.74E-04
average	3.80E-04						
Efficient Structure							
Minimum Error							

Table E5: MSE of different combinations of input and hidden neurons – Testing data set

E1.2.6 Conclusion – Input and Hidden Neurons

In order to balance the error between the structures over the training and testing data sets, a structure was chosen which minimized the reduction of accuracy while remaining parsimonious. The structure chosen had 12 input and 32 hidden neurons. The increase in MSE over the training and testing data sets from their minimums were 2.27E-04 and 0.07E-04, respectively.

The choice made here had subjective components. It was important not to base the decision purely on the results over the training set as this would lead to over fitting. Similarly, basing the decision purely on the results over the testing set would lead to the fitting of the ANN to the testing set. Both the scenarios described above would have reduced the generalization ability of the ANN.

E1.2.7 Conclusion of Determining an Efficient Structure

Efficient parameters were:

- 12 input neurons,
- 32 hidden neurons,
- A learning rate increase of 1.05 (5% increase), and
- A learning rate decrease of 0.8 (20% decrease).

E2 Analysing the Efficient ANN

The ANN was trained with the following parameters.

Training set size	317
Testing set size	100
Input/lagged neurons	12
Hidden neurons	32
Number of initialized structures	10
Epochs	1 000
Training algorithm	RPROP
Learning rate:	
Increase	1.05 (5% increase)
Decrease	0.8 (20% decrease)
Maximum	100
Minimum	0.000000001

Table E6: Parameters for the efficient ANN structure

E2.1 Observations of System Error

The error over the testing set began off large and decreased substantially thereafter as indicated in the plot of the error over the training process, figure E6. After the initial decreases the error over the testing set began to oscillate as the training continued, but there was no increase in error. Over fitting was not prevalent in the ANN as seen from the stable error over the testing set. The minimum RMSE (0.0193) over the testing set was achieved after approximately 60 epochs. The error over the training set continued to decrease over the training process and reached the minimum after 1000 epochs. The number of epochs chosen to train the system was 59 as this ensured any over fitting was avoided. This point balanced both the error over the training and testing sets. The numerical error value at points of interest are provided in table E7.

Error Readings	Testing Set	Training Set
Minimum RMSE	0.0193	0.0251
Minimum MSE	3.7354e-004	6.2859e-004
Final RMSE after training *(59 epochs)*	0.0193	0.0260
Final MSE after training *(59 epochs)*	3.7401e-004	6.7580e-004

Table E7: Results of efficient model after training

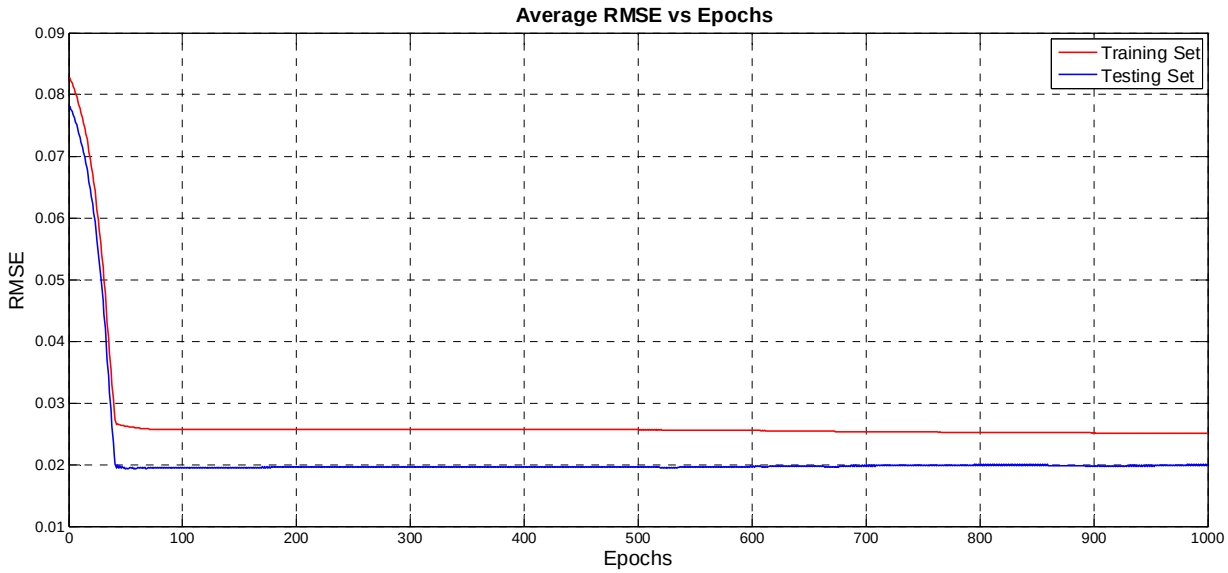

Figure E6: Log of RMSE over 1 000 training epochs

E2.2 Observations of Forecasts and Actuals

The ANN captured a limited number of trends underlying the data set, as expected. There were certain periods where the system underestimated or overestimated returns on the bond market. This was explained by the effect of external influences on the market. The best estimate over the training and testing sets was represented by the blue line in figure E7.

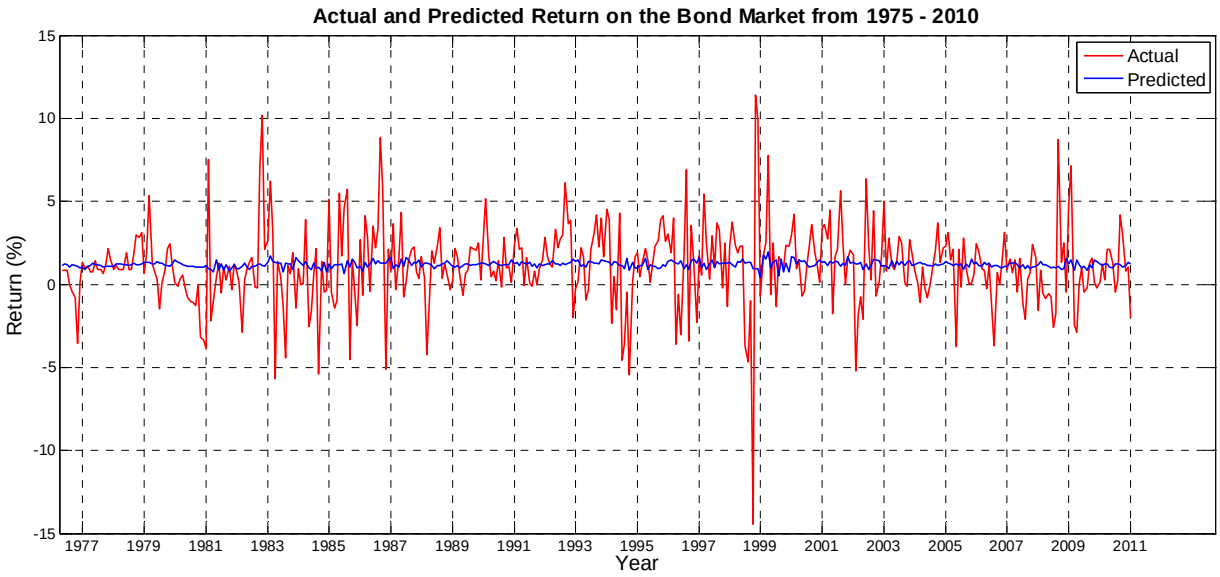

Figure E7: Actual and Predicted – Training and Testing Data Sets

E2.3 Conclusion - Analysing the Efficient ANN

The ANN captured a limited number of trends underlying the data. The efficient ANN structure (after 59 training epochs) had an MSE over the training and testing sets of 6.7580e-004 and 3.7401e-004, respectively. This 59 training epochs was viewed as an efficient training period. The ANN performed similarly over the testing and training data set. The level of effectiveness was determined by comparing these results to traditional models in a following chapter (Chapter 5). The minimum RMSE (0.0193) over the testing set was obtained after approximately 60 training epochs. The RMSE over the training set decreased as the training went over 60 epoch and continued to decrease until the maximum number of epochs were obtained. Theoretically, this was expected as ANNs have the ability to mimic a data set precisely. This mimicking leads to over fitting and reduces the system's generalization ability. To avoid over fitting but to ensure the underlying relationships in the data were captured, decisions relating to efficiency were made with reference to the performance of the ANN over both the testing and training data set.

E3 Summary of Experiment

The following table provides a summary of the results from this experiment.

Structure of ANN	12-32-1
Training epoch required for optimal training	59
Minimum Root Mean Squared Error:	
Training Data Set	0.0251
Testing Data Set	0.0193
Final Root Mean Squared Error (59 epochs):	
Training Data Set	0.0260
Testing Data Set	0.0193
Parameters for RPROP:	
Learning Rate Increase	5% (1.05)
Learning Rate Decrease	20% (0.8)
Data Sets:	
Training set	319 observations (1976 – 2002)
Testing set	100 observations (2002 – 2010)
Times the structure was initialized	10

Table E8: Summary of experiment's result

Appendix F – Isolated Equity Market ANN – Three-Month Forecast Results

F1 Determining an Efficient Structure

F1.1 Part 1 – Efficient number of training epochs

The parameters used for this experiment were:

Parameter	Value
Network Structure	36-64-1
Training set size	293
Testing set size	100
Number of initialized structures	5
Epochs	50 000
Training algorithm	RPROP
Learning rate change: Increase Decrease Maximum Minimum	 0.5% (1.005) 20% (0.8) 100 0.000000001

Table F1: Parameters for experiment - Part 1

F1.1.1 Observations – Part 1

The error of the system minimised over the training set after 50 000 epochs and over the testing set within the first 2 000 epochs, as illustrated by figure F1. This indicated over fitting was prevalent in this instance. The RMSE minimised at 0.0056 and 0.0536 over the training and testing sets, respectively. The large RMSE of 0.0536 over the testing set indicated what is generally expected, the current return on the equity market has a low correlation with past returns.

As the minimum over the testing set was obtained over the first 2 000 epochs, it was decided to use 2 000 epochs in the following experiment. This ensure every structure would obtain the minimum error of the system.

		Training Epochs				
	Min RMSE	**100**	**2 000**	**50 000**	**Δ in RMSE – 100 to 2 000**	**Δ in RMSE – 2 000 to 50 000**
RMSE – Testing Set	0.0536	0.0536	0.0695	0.0715	-0.0159	-0.002
RMSE – Training Set	0.0056	0.0647	0.0169	0.0056	0.0478	0.0113

Table F2: RMSE after different numbers of epochs

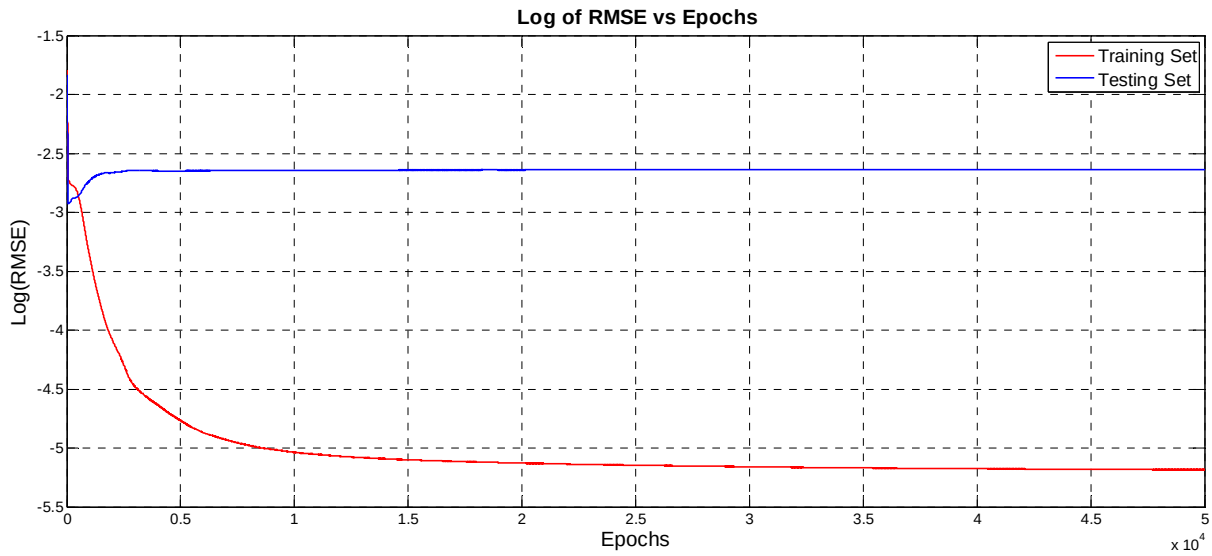

Figure F1: Log of RMSE after different numbers of epochs

F1.1.2 Conclusion – Part 1:

To compare different ANN structures and parameters, a maximum of 2 000 training epochs was used. This was more than required, but ensured every structure was fairly represented.

F1.2 Part 2 – Determining efficient parameters

F1.2.1 Observations – Learning Rate Increase/Decrease

From table F3 it is clear the minimum average MSE over the training set (3.03E-03) was achieved with a learning rate increase of 1.05 (5% increase) and a decrease of 0.8 (20% decrease). The minimum average MSE over the testing set (2.79E-03) was achieved with an increase of 1.005 (0.5% increase) and a decrease of 0.5 (50% decrease). These observations did not support the same conclusion.

The shape of the average MSE surface over the training set was considered. It was observed the range over the surface was significant (0.97E-03) in comparison to the range of the average MSE surface over the testing set (0.02E-03). The efficient learning rate changes were determined by calculating the smallest trade off in error over both sets of data.

The efficient learning rate increase and decrease chosen (represented by orange in table F3) were 1.05 and 0.8 as this resulted in the minimum error over the training data set (represented by green in table F3) and an increase in average MSE by 0.01E-03 over the testing data set.

It is important to note the values used to decide on the learning rate changes were the arithmetic average MSE over different combinations of input and hidden neurons. This average did not explicitly allow for the event of outliers in the data and may be skewed. From the investigations there were no outliers found but extensive tests to verify this remark were not performed. Further investigation into the possibility of outliers can be done.

Training Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	4.00E-03	3.46E-03	3.47E-03
	0.5	3.79E-03	3.31E-03	3.37E-03
	0.8	3.42E-03	**3.03E-03**	3.16E-03
	min	3.03E-03		
Testing Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	2.79E-03	2.79E-03	2.80E-03
	0.5	2.79E-03	2.80E-03	2.80E-03
	0.8	2.79E-03	**2.80E-03**	2.81E-03
	min	2.79E-03		
	Minimum			
	Efficient			

Tables F3: MSE for Learning Rate Changes

F1.2.2 Conclusion – Learning Rate Increase/Decrease

The following decisions were made:

- A learning rate decrease of 0.8 was to be used, i.e. a 20% decrease in learning rate if the error of the system increases during training.
- A learning rate increase of 1.05 was to be used, i.e. a 5% increase in learning rate if the error of the system decreases during training.

F1.2.3 Observations – Input and Hidden Neurons – Training Set's MSE surface

The MSE surface with limits, figure F2, suggested the MSE of the system decreased on average as the number of input and hidden neurons increased. This was expected since the increasing complexity of the system would increase the accuracy over the training data set. The upper and lower surface represents the upper and lower limit of the MSE.

The limits were determined by increasing/decreasing the best estimate surface by two standard deviations of the MSE. It was unexpected that these limits would become increasingly volatile as the structure increased in complexity. This was likely due to the increased number and thus random variation of the weights when the network was initialised.

Considering the best estimate error surface, figure F3, the MSE of the system decreased as the complexity of the system increased. The surface obtained the minimum MSE at 36 input and 64 hidden neurons (1.87E-04).

The upper and lower limit of the MSE surface over the training set requires assumptions in order to hold statistical significance. There was insufficient information available to make these assumptions. These limits, however, did provide a good estimate of the MSE limits for the randomly initialized structures.

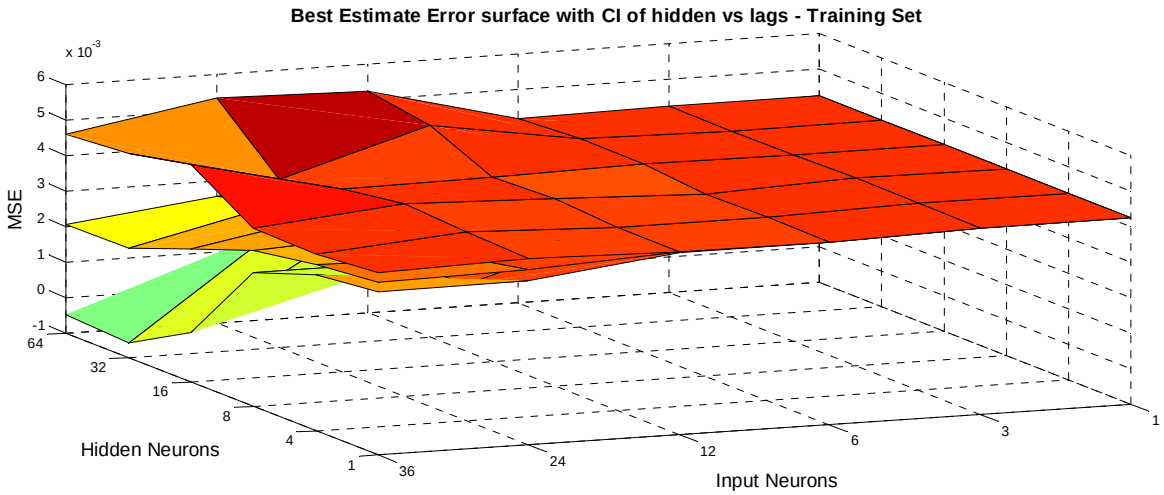

Figure F2: MSE with CI surface – Training data set

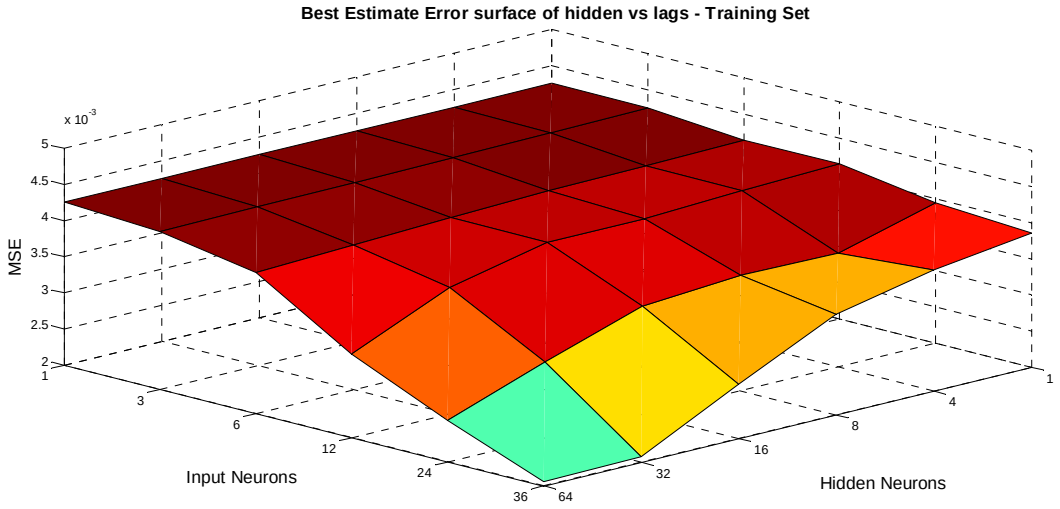

Figure F3: MSE surface – Training data set

F1.2.4 Observations – Input and Hidden Neurons – Testing Set's MSE surface

The MSE surface with limits over the testing set, figure F4, indicated a minimum plain from 1 to 6 input neurons and over all hidden neurons. There was a general increase in MSE as the number of input neurons increased. This may be due to the increased ability of the system to over fit the data given additional inputs. Complex structures were likely to over fit the data. The minimum MSE (2.73E-03) on the best estimate MSE surface was achieved using a single input and 8 hidden neurons. The MSE surface over the testing set was structurally different to that of the training set.

The upper and lower limit of the MSE surface over the training set requires assumptions in order to hold statistical significance. There was insufficient information available to make these assumptions. These limits, however, did provide a good estimate of the MSE limits for the randomly initialized structures.

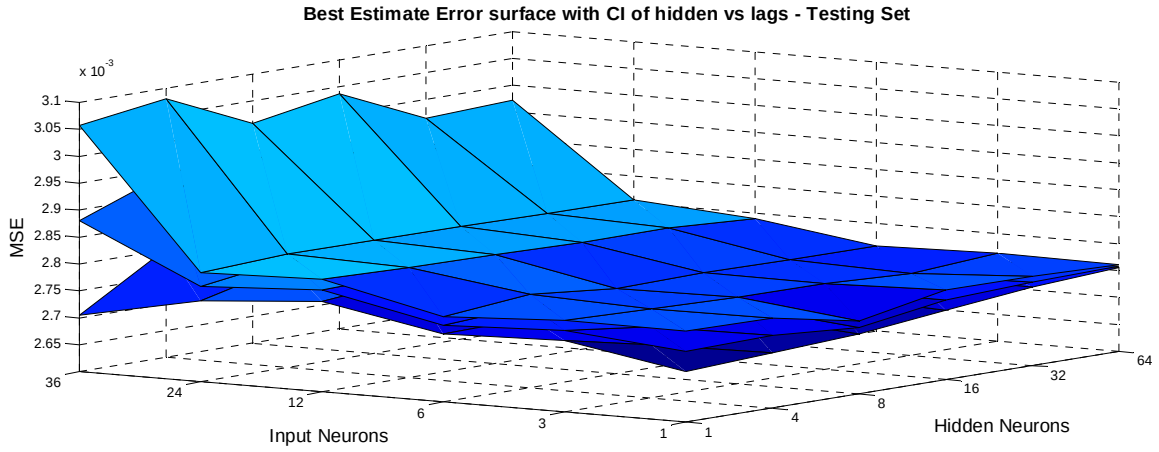

Figure F4: MSE with CI surface – Testing data set

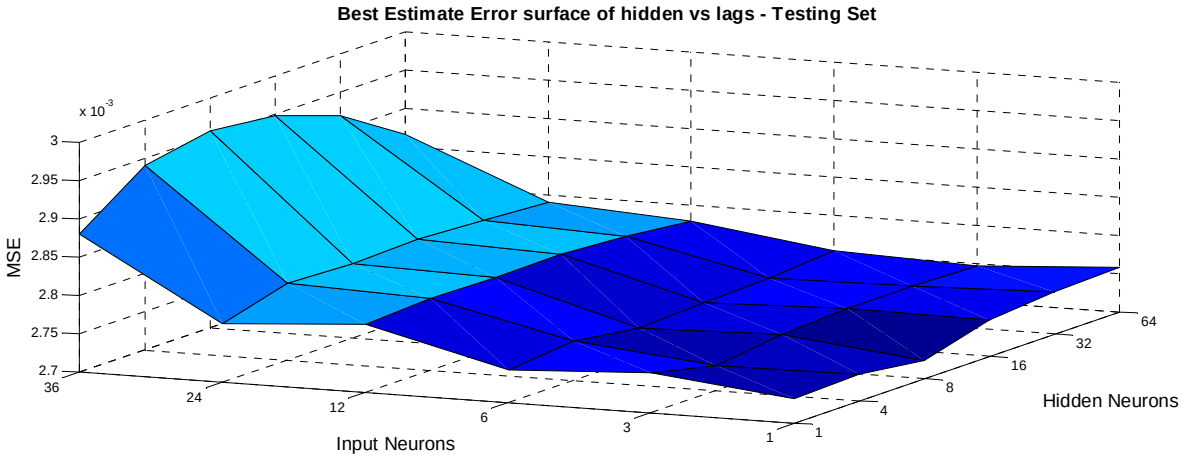

Figure F5: MSE surface – Testing data set

F1.2.5 Numerical Representation of MSE surfaces

The tables below provide the numerical average MSE values used to construct the error surfaces in figures F2, F3, F4 and F5.

Training Sets							
		LR increase		1.05			
		LR decrease		0.8	(20% decrease)		
	Average MSE vs Input and Hidden						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	4.26E-03	4.26E-03	4.26E-03	4.26E-03	4.26E-03	4.20E-03
	3	4.25E-03	4.21E-03	4.10E-03	3.88E-03	3.76E-03	3.64E-03
	6	4.13E-03	3.92E-03	3.75E-03	3.45E-03	**3.05E-03**	2.46E-03
	12	4.15E-03	3.80E-03	3.15E-03	2.92E-03	1.75E-03	1.23E-03
	24	3.91E-03	2.79E-03	2.52E-03	1.83E-03	1.02E-03	4.87E-04
	36	3.92E-03	2.33E-03	1.70E-03	9.09E-04	4.66E-04	1.87E-04
average	3.03E-03						
min	1.87E-04						
	Upper limit MSE vs Input and Hidden (+2 STD Dev)						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	4.27E-03	4.26E-03	4.26E-03	4.26E-03	4.26E-03	4.36E-03
	3	4.26E-03	4.30E-03	4.44E-03	4.03E-03	4.19E-03	3.86E-03
	6	4.18E-03	4.31E-03	4.02E-03	4.09E-03	**3.65E-03**	3.16E-03
	12	4.18E-03	4.75E-03	4.10E-03	3.84E-03	2.17E-03	2.05E-03
	24	4.40E-03	3.19E-03	3.37E-03	2.79E-03	1.90E-03	8.17E-04
	36	4.26E-03	3.34E-03	2.37E-03	1.30E-03	9.37E-04	2.98E-04
average	3.45E-03						
Efficient Structure							
Minimum Error							

Table F4: MSE of different combinations of input and hidden neurons – Training data set

Testing Sets							
		LR increase		1.05			
		LR decrease		0.8	(20% decrease)		
	Average MSE vs Input and Hidden						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	2.75E-03	2.74E-03	2.73E-03	2.75E-03	2.75E-03	2.75E-03
	3	2.75E-03	2.75E-03	2.75E-03	2.75E-03	2.75E-03	2.76E-03
	6	2.76E-03	2.74E-03	2.75E-03	2.74E-03	2.75E-03	2.75E-03
	12	2.79E-03	2.80E-03	2.79E-03	2.80E-03	2.78E-03	2.78E-03
	24	2.79E-03	2.82E-03	2.81E-03	2.81E-03	2.80E-03	2.79E-03
	36	3.03E-03	3.03E-03	3.03E-03	3.01E-03	2.95E-03	2.87E-03
average	2.80E-03						
min	2.73E-03						
	Upper limit MSE vs Input and Hidden (+2 STD Dev)						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	2.78E-03	2.77E-03	2.74E-03	2.76E-03	2.76E-03	2.76E-03
	3	2.78E-03	2.76E-03	2.77E-03	2.75E-03	2.76E-03	2.76E-03
	6	2.78E-03	2.76E-03	2.76E-03	2.76E-03	2.76E-03	2.76E-03
	12	2.85E-03	2.81E-03	2.82E-03	2.80E-03	2.79E-03	2.79E-03
	24	2.84E-03	2.88E-03	2.83E-03	2.82E-03	2.81E-03	2.80E-03
	36	3.51E-03	3.16E-03	3.12E-03	3.10E-03	3.01E-03	2.93E-03
average	2.85E-03						
Efficient Structure							
Minimum Error							

Table F5: MSE of different combinations of input and hidden neurons – Testing data set

F1.2.6 Conclusion – Input and Hidden Neurons

In order to balance the error between the structures over the training and testing data sets, a structure was chosen which minimized the reduction of accuracy while still remaining parsimonious. The structure chosen had 6 input and 32 hidden neurons. The increase in MSE over the training and testing data sets were 2.86E-03 and 0.02E-03, respectively.

The choice made here had subjective components. It was important not to base the decision purely on the results over the training set as this would lead to over fitting. Similarly, basing the decision purely on the results over the testing set would lead to the fitting of the ANN to the testing set. Both the scenarios described above would have reduced the generalization ability of the ANN.

F1.2.7 Conclusion of Determining an Efficient Structure

Efficient parameters were:

- 6 input neurons,
- 32 hidden neurons,
- A learning rate increase of 1.05 (5% increase), and
- A learning rate decrease of 0.8 (20% decrease).

F2 Analysing the Efficient ANN

The ANN was trained with the following parameters.

Training set size	323
Testing set size	100
Input/lagged neurons	6
Hidden neurons	32
Number of initialized structures	50
Epochs	10 000
Training algorithm	RPROP
Learning rate:	
Increase	1.05 (5% increase)
Decrease	0.8 (20% decrease)
Maximum	100
Minimum	0.000000001

Table F6: Parameters for the efficient ANN structure

F2.1 Observations of System Error

After approximately 50 training epochs the system began to over fit, as indicated by figure F6. This process was slow as could be seen from the gradual increase in the error over the testing set. It was decided the efficient number of training epochs would consider the error over both the testing and training sets. In this case the point at which the error over the testing set minimised, 50 epochs, was chosen. This point provided a good balance of error reduction over both sets of data while not requiring additional computational power.

Error Readings	Testing Set	Training Set
Minimum RMSE	0.0524	0.0578
Minimum MSE	0.0027	0.0033
Final RMSE after training *(50 epochs)*	0.0525	0.0647
Final MSE after training *(50 epochs)*	0.0028	0.0042

Table F7: Results of efficient model after training

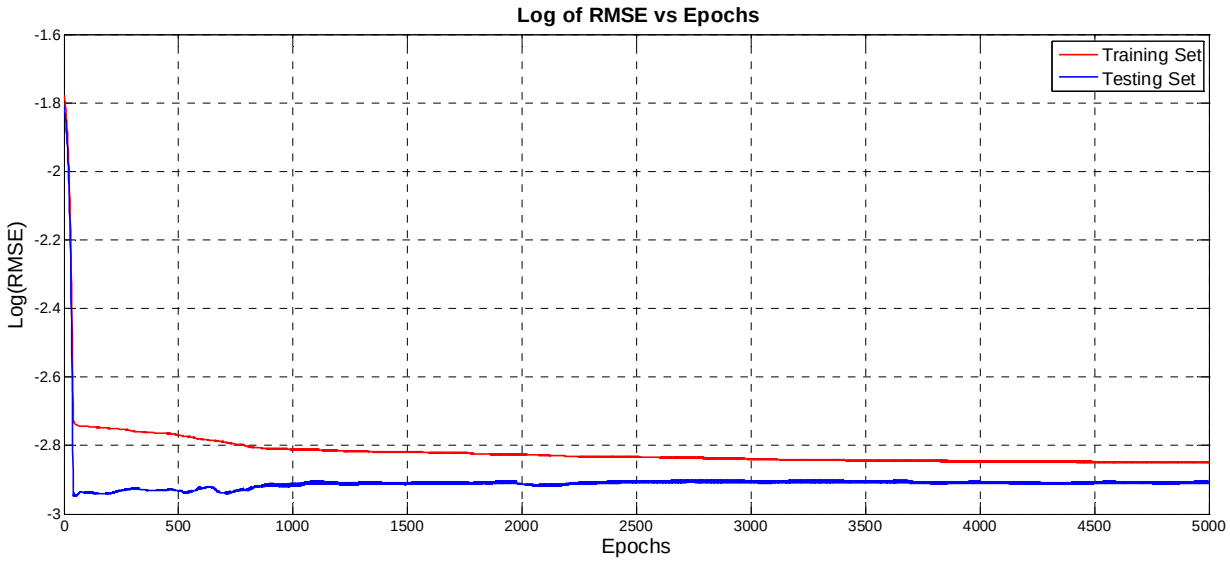

Figure F6: Log of RMSE after 5 000 training epochs

F2.2 Observations of Forecasts and Actuals

The ANN captured few underlying trends in the data. There were certain periods where the system underestimated or overestimated returns on the equity market. This was explained by the effect of external influences on the market. The best estimate over the training and testing sets was represented by the blue line in figure F7.

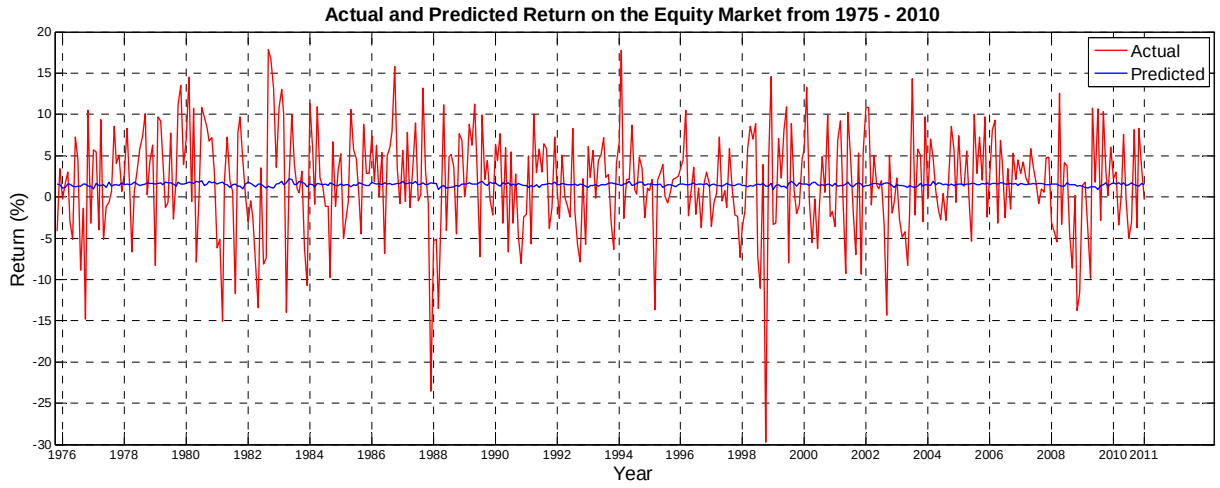

Figure F7: Actual and Predicted – Training and Testing Data Sets

F2.3 Conclusion - Analysing the Efficient ANN

The ANN captured a small number of the trends underlying the data. The efficient ANN structure had an MSE over the training and testing sets of 0.0042 and 0.0028, respectively. The ANN performed similarly over the testing and training data set. The level of effectiveness was determined by comparing these results to traditional models in a following chapter (Chapter 5). The minimum RMSE over the testing set, of 0.0524, and over the training set, of 0.0578, was obtained after approximately 50 and 5 000 epochs, respectively. The RMSE of the training set decreased as the training went over 50 epochs and continued to decrease until the maximum number of epochs were obtained. Theoretically, this was expected as ANNs have the ability to mimic a data set precisely. This mimicking leads to over fitting and reduces the system's generalization ability. To avoid over fitting but to ensure the underlying relationships in the data were captured, decisions relating to efficiency were made with reference to the performance of the ANN over both the testing and training data sets.

F3 Summary of Experiment

The following table provides a summary of the results from this experiment.

Structure of ANN	6-32-1
Training epoch required for optimal training	50
Minimum Root Mean Squared Error: Training Data Set Testing Data Set	 0.0578 0.0524
Final Root Mean Squared Error (50 epochs): Training Data Set Testing Data Set	 0.0647 0.0525
Parameters for RPROP: Learning Rate Increase Learning Rate Decrease	 1.05 (5% increase) 0.8 (20% decrease)
Data Sets: Training set Testing set	 323 observations (1975 – 2002) 100 observations (2002 – 2010)
Times the structure was initialized	50

Table F8: Summary of experiment's result

Appendix G – Integrated Inflation ANN - One Month Forecast Results

G1 Determining an Efficient

G1.1 Part 1 – Efficient number of training epochs

The parameters used for this experiment were:

Parameter	Value
Network Structure (input neurons per data set x number of data sets) – hidden neurons – output neurons	144 (36x4)-64-1
Training set size	295
Testing set size	100
Dummy variables (for seasonality) (variables per data set x number of data sets)	8 (2x4)
Number of initialized structures	5
Epochs	1 000
Training algorithm	RPROP
Learning rate change: Increase Decrease Maximum Minimum	 0.5% (1.005) 20% (0.8) 100 0.000000001

Tables G1: Parameters for experiment - Part 1

G1.1.1 Observations – Part 1

Figure G1 indicated the minimum RMSE (0.0059) over the testing set was achieve within the first 1 000 epochs. The error began to increase from this point onwards, represented by the trough of the blue line after approximately 340 training epochs. The increase of the RMSE indicated the system was over fitting to this particular data set. The minimum RMSE (0.00084) over the training set was achieve at the 1 000 epoch mark, as expected.

The occurrence of over fitting in the ANN indicated the presence of noise in the data. This randomness could not be explained by past observations of the money market. This supported the theory which stipulated that the money market

is affected by external factors, such the country's monetary policy and only a finite number of trends can be explained through past observations.

		Training Epochs				
	Min RMSE	**100**	**500**	**1 000**	**Δ in RMSE – 100 to 500**	**Δ in RMSE – 500 to 1 000**
RMSE – Testing Set	0.0059	0.0391	0.0069	0.0076	0.0322	-0.0007
RMSE – Training Set	8.4015e-004	0.0345	0.0019	8.4239e-004	0.0326	0.00106

Tables G2: RMSE after different numbers of epochs

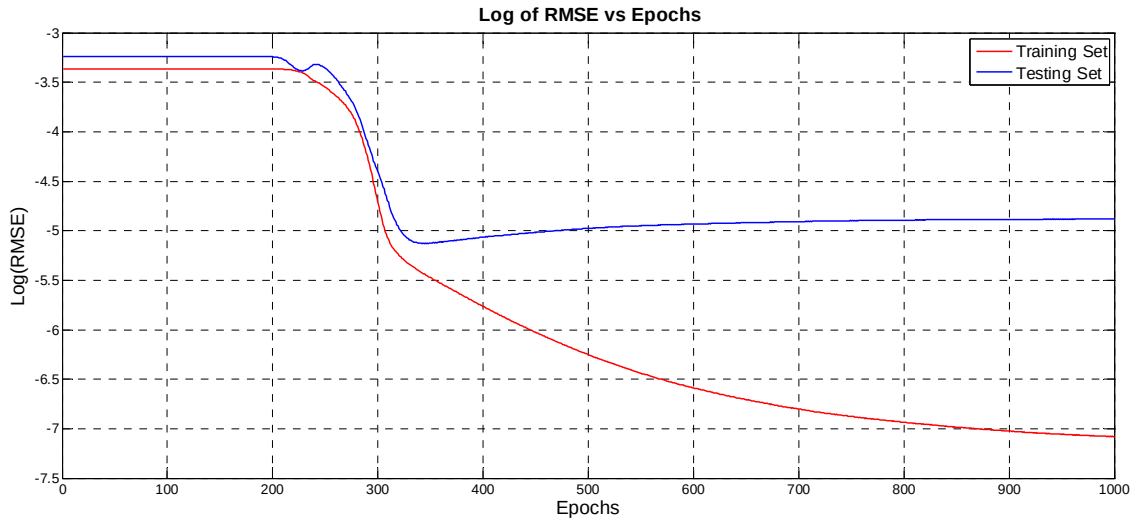

Figure G1: Log of RMSE over training process

G1.1.2 Conclusion – Part 1:

From the results, the number of epochs used to compare different ANN structures was 1 000. This was chosen to allow the system to over fit the data. Over fitting the data ensured the minimum error over the testing set was surpassed and captured. This process ensured different ANN structures could be compared consistently. The minimum RMSE over the testing set was 0.0059 and the RMSE after 1 000 epochs was 0.0076. This lead to the achievement of the minimum RMSE within the first 1 000 epochs. This inference was supported by figure G1 through the trough created by the blue line at approximately 330 epochs.

This ANN was the most complex and had the slowest learning parameters. Other simpler ANNs were expected to reach their minimum errors within the 1 000 epochs as the learning was faster. The conclusion of an efficient number of learning epochs was primarily based on the results obtained over the testing set. Primarily considering the error

over the training set would cause the most complex ANN to be selected due to the increased risk of over fitting in complex ANNs.

G1.2 Part 2 – Determining efficient parameters

G1.2.1 Observations – Learning Rate Increase/Decrease

From table G3 the minimum MSE over the training set (1.74E-05) was achieved with a learning rate increase of 1.05 (5% increase) and a decrease of 0.8 (20% decrease). The minimum MSE over the testing set (2.69E-05) was achieved with an increase of 1.005 (0.5% increase) and a decrease of 0.8 (20% decrease). These observations did not support a similar conclusion.

The shape of the average MSE surface over the training set was considered. It was observed the range over the training surface was significant (0.9E-05) in comparison to the range of the MSE surface over the testing set (0.39E-05). The efficient learning rate changes were chosen by determining the smallest trade off in error over both sets of data.

The efficient learning rate increase and decrease chosen were 1.005 and 0.8, respectively. They resulted in a fair trade off between the accuracy over the associated data sets. The increase in average MSE over the training and testing sets were 0.04E-05 and 0, respectively. The MSE associated with the chosen learning rate increase and decrease are marked in orange in table G3 and the minimum MSEs are marked in green.

It is important to note the values used to decide on the learning rate changes were the arithmetic average MSE over different combinations of input and hidden neurons. This average did not explicitly allow for the event of outliers in the data and may be skewed. From the investigations there were no outliers found but extensive tests to verify this remark were not performed. Further investigation into the possibility of outliers can be done.

Training Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	2.64E-05	2.25E-05	2.43E-05
	0.5	2.16E-05	2.01E-05	2.12E-05
	0.8	**1.78E-05**	1.74E-05	1.95E-05
	min	1.74E-05		
Testing Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	3.08E-05	2.99E-05	2.97E-05
	0.5	2.80E-05	2.80E-05	2.88E-05
	0.8	**2.69E-05**	2.78E-05	2.90E-05
	min	2.69E-05		
	Minimum			
	Efficient			

Tables G3: MSE for Learning Rate Changes

G1.2.2 Conclusion – Learning Rate Increase/Decrease

The following decisions were made:

- A learning rate decrease of 0.8 was to be used, i.e. a 20% decrease in learning rate if the error of the system increases during training.
- A learning rate increase of 1.005 was to be used, i.e. a 0.5% increase in learning rate if the error of the system decreases during training.

G1.2.3 Observations – Input and Hidden Neurons – Training Set's MSE surface

The MSE surface with limits, figure G2, indicated the MSE of the system decreased as the number of input and hidden neurons increased. This was expected since the increased complexity of the system would increase the probability of over fitting. Over fitting would lead to increased accuracy over the training data set. The upper and lower surface represents the upper and lower limit of the MSE. The limits were determined by increasing/decreasing the best estimate surface by two standard deviations of the MSE. Considering the best estimate error surface, figure G3, the MSE of the system decreased as the complexity of the system increased. The surface obtained the minimum at 36 input and 64 hidden neurons (4.06E-07). This was as expected.

The upper and lower limit of the MSE surface over the training set requires assumptions in order to hold statistical significance. There was insufficient information available to make these assumptions. These limits, however, did provide a good estimate of the MSE limits for the randomly initialized structures.

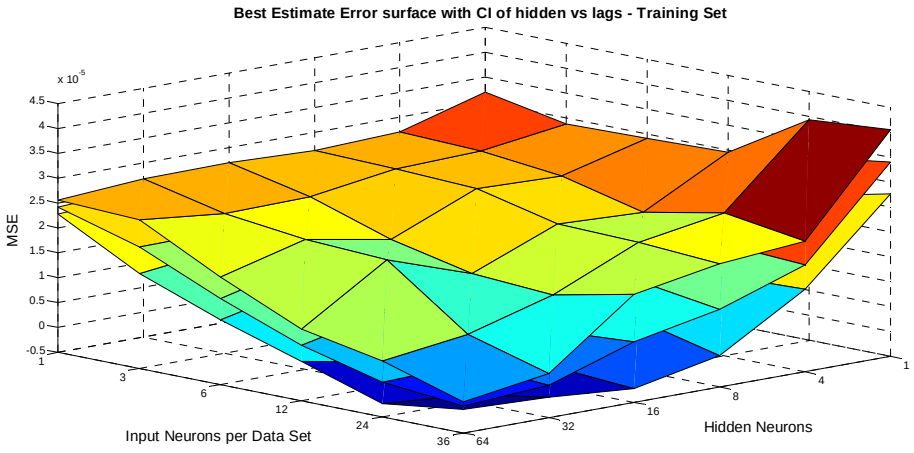

Figure G2: MSE with CI surface – Training data set

Figure G3: MSE surface – Training data set

G1.2.4 Observations – Input and Hidden Neurons – Testing Set's MSE surface

The MSE surface over the testing set with confidence intervals, displayed in figure G4, indicated a rough surface with a minimum located in a basin when 6 input and 16 hidden neurons were used.

The minimum MSE was indicated in the best estimate MSE surface, figure G5, by a basin when 6 input and 16 hidden neurons were considered. This was similar to that indicated by figure G4. The minimum MSE of the best estimate surface was achieved using 6 input and 16 hidden neurons (1.97E-05). The MSE surface over the testing set was structurally different to that of the training set.

The upper and lower limit of the MSE surface over the testing set requires assumptions in order to hold statistical significance. There was insufficient information available to make these assumptions. These limits, however, did provide a good estimate of the MSE limits for the randomly initialized structures.

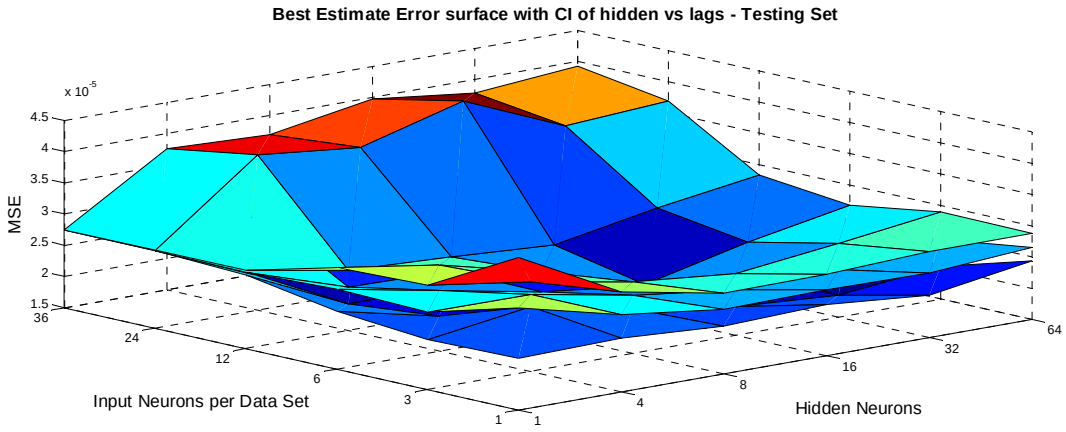

Figure G4: MSE with CI surface – Testing data set

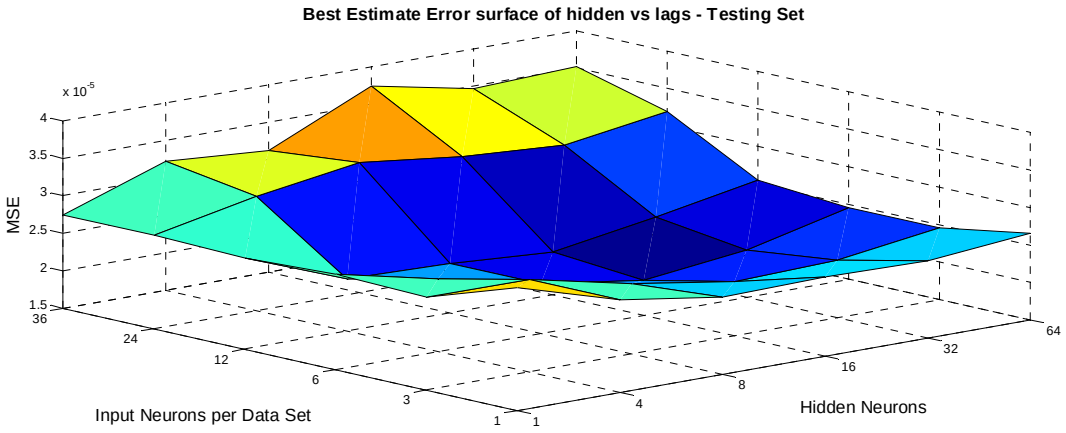

Figure G5: MSE surface – Testing data set

G1.2.5 Numerical Representation of MSE surfaces

The tables below provide the numerical average MSE values used to construct the error surfaces in figures G2, G3, G4 and G5.

Note: in the tables below, the Input Neurons refer to the number of input neurons per data set. Therefore, to determine the total number of input neurons this number was multiplied by 4.

Training Sets							
		LR increase		1.005			
		LR decrease		0.8	(20% decrease)		
	Average MSE vs Input and Hidden						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	3.02E-05	2.65E-05	2.52E-05	2.50E-05	2.52E-05	2.42E-05
	3	2.83E-05	2.57E-05	2.26E-05	2.05E-05	1.96E-05	1.94E-05
	6	2.74E-05	2.38E-05	2.20E-05	**1.51E-05**	1.64E-05	1.23E-05
	12	2.76E-05	1.96E-05	1.63E-05	9.08E-06	9.95E-06	6.13E-06
	24	3.17E-05	1.74E-05	1.30E-05	6.13E-06	2.41E-06	1.93E-06
	36	3.41E-05	1.64E-05	1.05E-05	7.20E-06	1.46E-06	4.06E-07
average	1.78E-05						
min	4.06E-07						
	Upper limit MSE vs Input and Hidden (+2 STD Dev)						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	3.19E-05	2.71E-05	2.63E-05	2.70E-05	2.68E-05	2.56E-05
	3	2.87E-05	2.65E-05	2.58E-05	2.34E-05	2.31E-05	2.49E-05
	6	2.91E-05	2.47E-05	2.51E-05	**1.80E-05**	2.10E-05	1.65E-05
	12	2.95E-05	2.06E-05	2.13E-05	1.44E-05	2.03E-05	9.37E-06
	24	3.93E-05	2.37E-05	2.08E-05	1.33E-05	8.47E-06	6.19E-06
	36	4.06E-05	2.12E-05	1.98E-05	1.66E-05	3.85E-06	1.16E-06
average	2.17E-05						
Efficient Structure							
Minimum Error							

Table 4.10: MSE of different combinations of input and hidden neurons – Training data set

Testing Sets							
		LR increase		1.005			
		LR decrease		0.8	(20% decrease)		
	Average MSE vs Input and Hidden						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	3.13E-05	2.73E-05	2.53E-05	2.56E-05	2.53E-05	2.66E-05
	3	2.74E-05	2.73E-05	2.43E-05	2.22E-05	2.26E-05	2.46E-05
	6	2.77E-05	2.47E-05	2.16E-05	**1.97E-05**	2.13E-05	2.45E-05
	12	2.72E-05	2.19E-05	2.16E-05	2.08E-05	2.30E-05	2.55E-05
	24	2.74E-05	3.02E-05	3.23E-05	3.07E-05	2.99E-05	3.19E-05
	36	2.74E-05	3.22E-05	3.12E-05	3.75E-05	3.47E-05	3.52E-05
average	2.69E-05						
min	1.97E-05						
	Upper limit MSE vs Input and Hidden (+2 STD Dev)						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	3.94E-05	3.10E-05	2.79E-05	2.80E-05	2.88E-05	2.88E-05
	3	3.17E-05	2.93E-05	2.65E-05	2.49E-05	2.68E-05	2.90E-05
	6	3.12E-05	2.87E-05	2.48E-05	**1.99E-05**	2.38E-05	2.67E-05
	12	2.76E-05	2.47E-05	2.39E-05	2.29E-05	2.60E-05	2.84E-05
	24	2.74E-05	3.99E-05	3.81E-05	4.27E-05	3.58E-05	3.69E-05
	36	2.74E-05	3.76E-05	3.69E-05	3.98E-05	3.78E-05	3.92E-05
average	3.06E-05						
Efficient Structure							
Minimum Error							

Table 4.11: MSE of different combinations of input and hidden neurons – Testing data set

G1.2.6 Conclusion – Input and Hidden Neurons

To balance the error between the varying ANN structures over the training and testing data sets, a structure was chosen which minimized the increase in MSE from the minimum in each data set. The structure chosen consisted of 6 input neurons per data set (24 in total) and 16 hidden neurons. The increase in MSE from the minimum over the training and testing data sets were 1.47E-05 and 0, respectively

The choice made here had subjective components. It was important not to base the decision purely on the results over the training set as this would lead to over fitting. Similarly, basing the decision purely on the results from the testing set would lead to fitting the ANN to the testing set. Both scenarios described above would have reduced the generalization ability of the ANN.

G1.2.7 Conclusion of Determining an Efficient Structure

Efficient parameters were:

- 6 input neurons per data set (24 input neurons in total),
- 8 seasonal input neurons,
- 16 hidden neurons,
- A learning rate increase of 1.005 (0.5% increase), and
- A learning rate decrease of 0.8 (20% decrease).

G2 Analysing the Efficient ANN

The ANN was trained with the following parameters.

Network Structure (input neurons per data set x number of data sets) – hidden neurons – output neurons	24 (6x4)-16-1
Training set size	325
Testing set size	100
Dummy variables (for seasonality) (variables per data set x number of data sets)	8 (2x4)
Number of initialized structures	10
Epochs	3 000
Training algorithm	RPROP
Learning rate change: Increase Decrease Maximum Minimum	 0.5% (1.005) 20% (0.8) 100 0.000000001

Table G6: Parameters for the efficient ANN structure

G2.1 Observations of System Error

Figure G6 indicated after 700 training epochs the system began to over fit and the ANN began to learn the random variation in the data set. This was represented by the RMSE over the testing set rising. The minimum RMSE of the system (0.0044) over the testing set was obtained after approximately 700 training epochs. The error over the testing and training set intersected after approximately 850 training epochs, which was used as the efficient number of epochs for this structure. This number of epochs balanced both the error over the training and testing sets. Figure G6 illustrates the log of the Root Mean Squared Error (Log (RMSE)) of the system over 3 000 training epochs. The numerical error value at points of interest are provided in table G7.

Error Readings	Testing Set	Training Set
Minimum RMSE	0.0044	0.0042
Minimum MSE	1.9127e-005	1.7851e-005
Final RMSE after training *(850 epochs)*	0.0045	0.0045
Final MSE after training *(850 epochs)*	2.0573e-005	2.0390e-005

Table G7: Results of efficient model after training

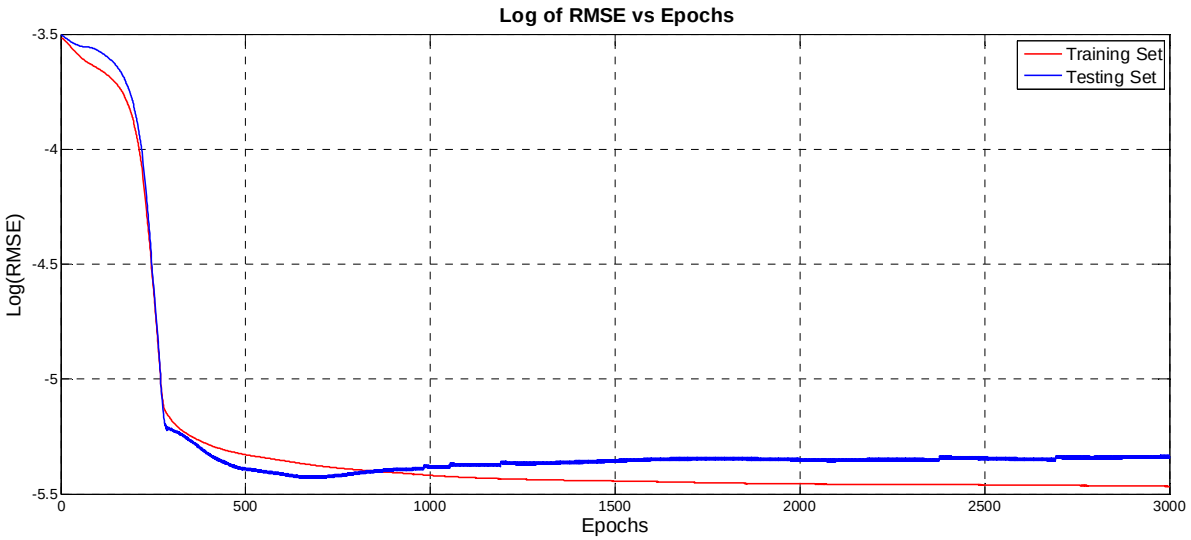

Figure G6: Log of RMSE after 3 000 training epochs

G2.2 Observations of Forecasts and Actuals

The ANN captured the underlying trends in the data set as expected. There were several occasions were the ANN either under or overestimated inflation. This was true for both the training and testing data sets. These instances were the result of external factors on inflation and could not be explained using only past observations of inflation. The best estimate over the training and testing sets was the blue line in figure G7.

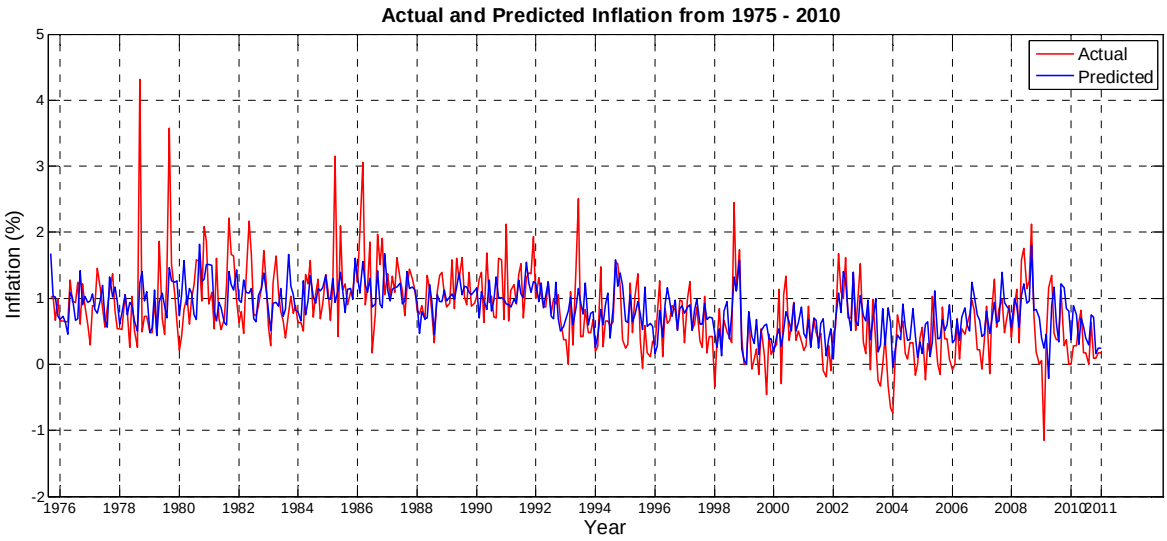

Figure G7: Actual and Predicted – Training and Testing Data Sets

G2.3 Conclusion - Analysing the Efficient ANN

The ANN captured several trends underlying the data. The efficient ANN structure had an MSE over the training and testing sets of 2.0390e-005 and 2.0573e-005, respectively. The ANN performed similarly over the testing and training data set. The level of effectiveness was determined by comparing these results to traditional models in a following chapter (Chapter 5). The minimum RMSE over the testing set, of 0.0044, and over the training set, of 0.0042, was obtained after approximately 700 and 3 000 epochs, respectively. The RMSE of the training set decreased as the training went over 700 epochs and continued to decrease until the maximum number of epochs were obtained. The efficient number of epochs used was 850 which resulted in a RMSE of 0.0045 over both the training and testing data sets.

G3 Summary of Experiment

The following table provides a summary of the results from this experiment

Network Structure (input neurons per data set x number of data sets) – hidden neurons – output neurons	24 (6x4)-16-1
Training epoch required for optimal training	850
Minimum Root Mean Squared Error: Training Data Set Testing Data Set	 0.0042 0.0044
Final Root Mean Squared Error (850 epochs): Training Data Set Testing Data Set	 0.0045 0.0045
Parameters for RPROP: Learning Rate Increase Learning Rate Decrease	 0.5% (1.005) 20% (0.8)
Data Sets: Training set Testing set	 325 observations (1975 – 2002) 100 observations (2002 – 2010)
Times the structure was initialized	10

Table G8: Summary of experiment's result

Appendix H – Integrated Money Market ANN - One Month Forecast Results

H1 Determining an Efficient Structure

H1.1 Part 1 – Efficient number of training epochs

The parameters used for this experiment were:

Parameter	Value
Network Structure (input neurons per data set x number of data sets) – hidden neurons – output neurons	144 (36x4)-64-1
Training set size	295
Testing set size	100
Dummy variables (for seasonality) (variables per data set x number of data sets)	8 (2x4)
Number of initialized structures	5
Epochs	1 000
Training algorithm	RPROP
Learning rate change: Increase Decrease Maximum Minimum	 0.5% (1.005) 20% (0.8) 100 0.000000001

Tables H1: Parameters for experiment - Part 1

H1.1.1 Observations – Part 1

The error over the training and testing sets was stable after 1 000 training epochs as indicated by figure H1. The error over the training set continued to decrease as the number of epochs approached its maximum. This was expected as ANNs are known to over fit the data set. The error over the testing set formed a plateau after approximately 500 epochs as illustrated by the horizontal portion of the blue line in figure H1. The minimum RMSE over the training and testing sets was 2.8288e-004 and 0.0026, respectively. The minimum error over the training set was obtained after the maximum number of epochs, 1 000 epochs in this case. The minimum error over the testing set was achieved after approximately 600 epochs after which there was a slight increase.

The RMSE over the testing set did not increase significantly over an increased number of training epochs. This indicated there were few signs of over fitting in this system, which was unexpected. This anomaly may be explained through consideration of the nature of the money market, which is highly predictable. Further, this indicated there was low levels of noise in the data set.

		Training Epochs				
	Min RMSE	**100**	**500**	**1 000**	**Δ in RMSE – 100 to 500**	**Δ in RMSE – 500 to 1 000**
RMSE – Testing Set	0.0026	0.0119	0.0026	0.0027	0.0093	-0.0001
RMSE – Training Set	2.8288e-004	0.0089	6.2521e-004	2.8288e-004	0.00827	3.424e-004

Tables H2: RMSE after different numbers of epochs

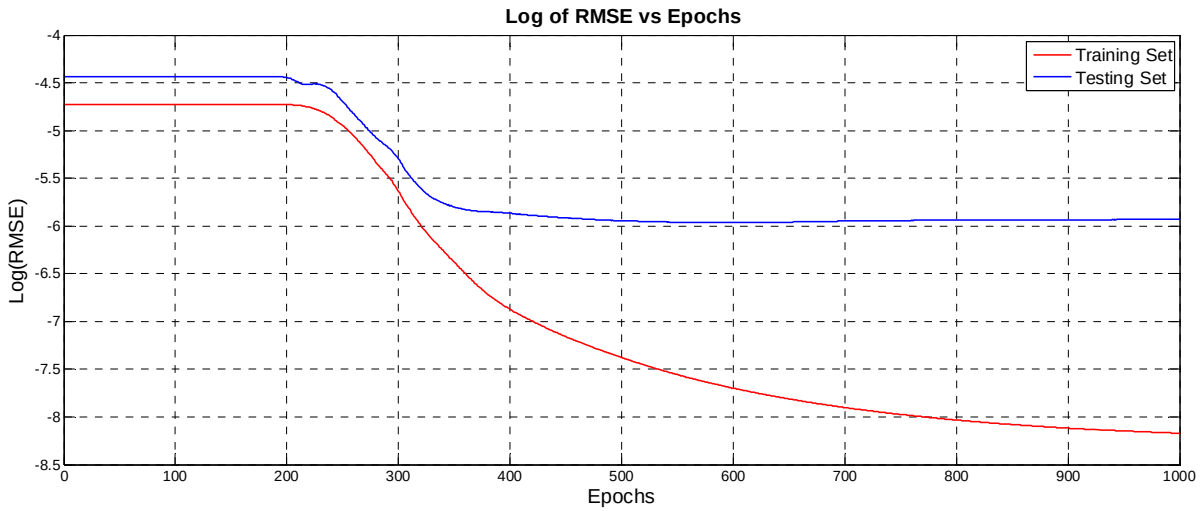

Figure H1: Log of RMSE over training process

H1.1.2 Conclusion – Part 1:

The results suggested the number of epochs to use to compare different ANN structures was 1000. After 1 000 epochs the system showed signs of slight over fitting. This was ideal for this experiment as this indicated the minimum error over the testing set was surpassed and captured. The minimum RMSE over the testing set was 0.0026, and the RMSE after 1 000 epochs was 0.0027. The increase in RMSE from the minimum to the 1 000 epoch mark confirmed the minimum error was achieved within the first 1 000 epochs. This was supported by the slight trough created by the blue line at approximately 600 epochs in figure H1.

This ANN was the most complex and had the slowest learning parameters. Other simpler ANNs were expected to reach their minimum errors within the 1 000 epochs as the learning progressed faster. The conclusion of an efficient

number of learning epochs was primarily based on the results obtained over the testing set. If the decision was made considering only the error over the training set the most complex ANN to be selected due to the increased risk of over fitting in complex ANNs. Ideally, a larger number of epochs should be used, however, this would have increased the computing time considerably and was not feasible in this case.

H1.2 Part 2 – Determining efficient parameters

H1.2.1 Observations – Learning Rate Increase/Decrease

The minimum MSE over the training set (8.24E-07) was achieved with a learning rate increase of 1.005 (0.5% increase) and a decrease of 0.8 (20% decrease), tabulated in table H3. The minimum MSE over the testing set (1.73E-06) was achieved with an increase of 1.005 (0.5% increase) and a decrease of 0.8 (20% decrease).

The efficient learning rate increase and decrease chosen were 1.005 and 0.8 as they resulted in the minimum MSE over both the training and testing data sets. The MSE associated with the chosen learning rate increase and decrease and the minimum MSE are marked in green in table H3.

It is important to note the values used to decide on the learning rate changes were the arithmetic average MSE over different combinations of input and hidden neurons. This average did not explicitly allow for the event of outliers in the data and may be skewed. From the investigations there were no outliers found but extensive tests to verify this remark were not performed. Further investigation into the possibility of outliers can be done.

Training Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	3.79E-06	2.78E-06	3.57E-06
	0.5	1.73E-06	1.99E-06	2.54E-06
	0.8	**8.24E-07**	1.30E-06	1.90E-06
	min	8.24E-07		
Testing Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	4.19E-06	3.56E-06	4.04E-06
	0.5	2.38E-06	2.86E-06	3.35E-06
	0.8	**1.73E-06**	2.75E-06	3.28E-06
	min	1.73E-06		
	Minimum			
	Efficient			

Tables H3: MSE for Learning Rate Changes

H1.2.2 Conclusion – Learning Rate Increase/Decrease

Base on the results obtained the following decisions were made:

- A learning rate decrease of 0.8 was to be used, i.e. a 20% decrease in learning rate if the error of the system increases during training.

- A learning rate increase of 1.005 was to be used, i.e. a 0.5% increase in learning rate if the error of the system decreases during training.

H1.2.3 Observations – Input and Hidden Neurons – Training Set's MSE surface

The MSE surface from the training set of data minimised after 4 hidden neurons were considered, as illustrated by the flattening of the MSE surface with and without limits, figure H2 and H3. When 4 or more hidden neurons were considered the MSE minimised for any number of input neurons. This was unexpected because the MSE of the system should have decreased over the training set as the structure increased in complexity. The MSE did decrease as the structure became more complex, however, this reduction was of a lower magnitude than expected. The MSE was insensitive to the change of input neurons when considering 4 or more hidden neurons. When a single hidden neuron was considered, the MSE was larger. The minimum MSE of 9.00E-08 was obtained using 36 input neurons per data set (144 input neurons in total) and 64 hidden neurons. The maximum MSE of 6.91E-06 was obtained using 36 input neurons per data set (144 in total) and a single hidden neurons. The maximum point was suspicious, since the largest MSE was expected to be obtained when a single input and hidden neuron was considered.

The upper and lower limit of the MSE surface over the training set were constructed by increasing/decreasing the average MSE by two standard deviations. Several assumptions are required for the limits to hold statistical significance. There was insufficient information available to make these assumptions. These limits, however, did provide a good estimate of the MSE limits for the randomly initialized structures.

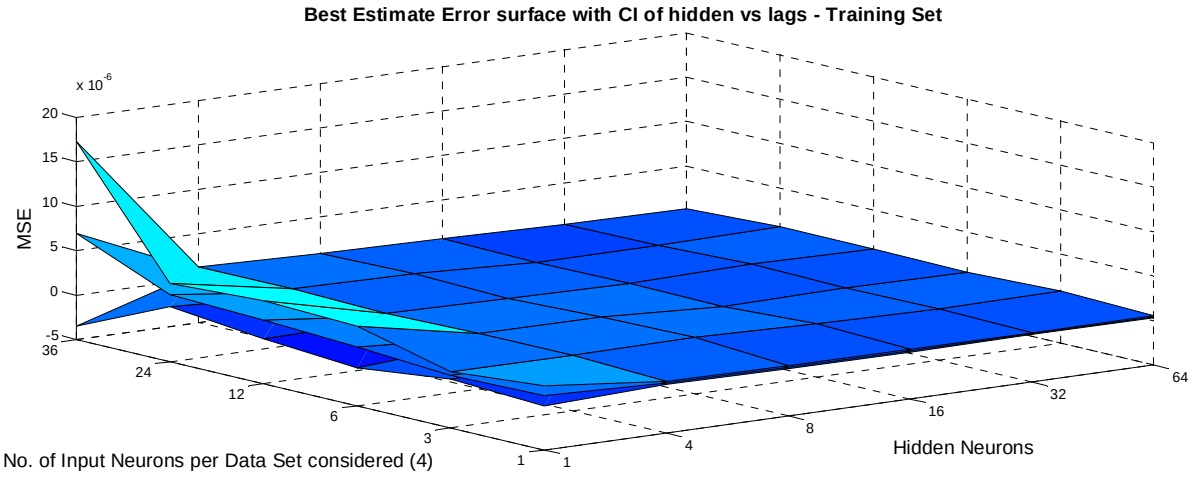

Figure H2: MSE with CI surface – Training data set

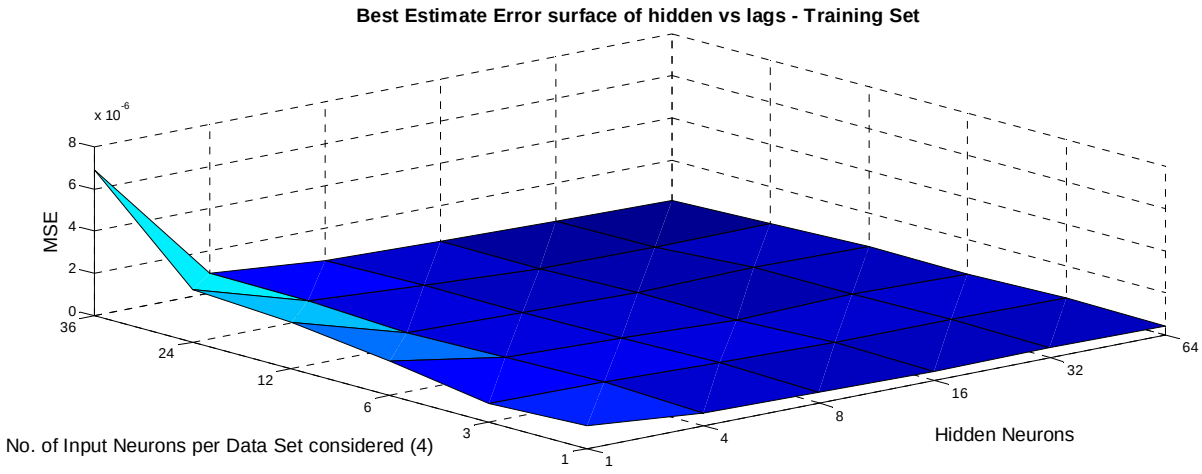

Figure H3: MSE surface – Training data set

H1.2.4 Observations – Input and Hidden Neurons – Testing Set's MSE surface

The MSE surface over the testing set with upper and lower limits, displayed in figure H4, minimized when between 1 and 12 input neurons per data set (4 – 48 input neurons in total) and more than 4 hidden neurons were considered. Figure H4 indicated over fitting was present in the system for large complex systems. Further, the volatility of the surface increased as the system increased in complexity. This was due to the increased number of weights associated with complex systems and their limited number of initializations.

The best estimate MSE surface, displayed in figure H5, indicated the maximum MSE of 7.63E-06 was achieve using 36 input neurons per data set (144 input neurons in total) and 64 hidden neurons.

The minimum MSE on the best estimate MSE surface of 1.36E-07 was obtained using a single input neuron per data set and 16 hidden neurons. This was unexpected as it suggests only a lagged period of a single month had an effect on the return on the money market. This may be explained by considering the nature of the money market. The money market has a low level of volatility and is unlikely to fluctuate significantly over a single month.

The upper and lower limit of the MSE surface over the training set were constructed by increasing/decreasing the average MSE by two standard deviations. Several assumptions are required for the limits to hold statistical significance. There was insufficient information available to make these assumptions. These limits, however, did provide a good estimate of the MSE limits for the randomly initialized structures.

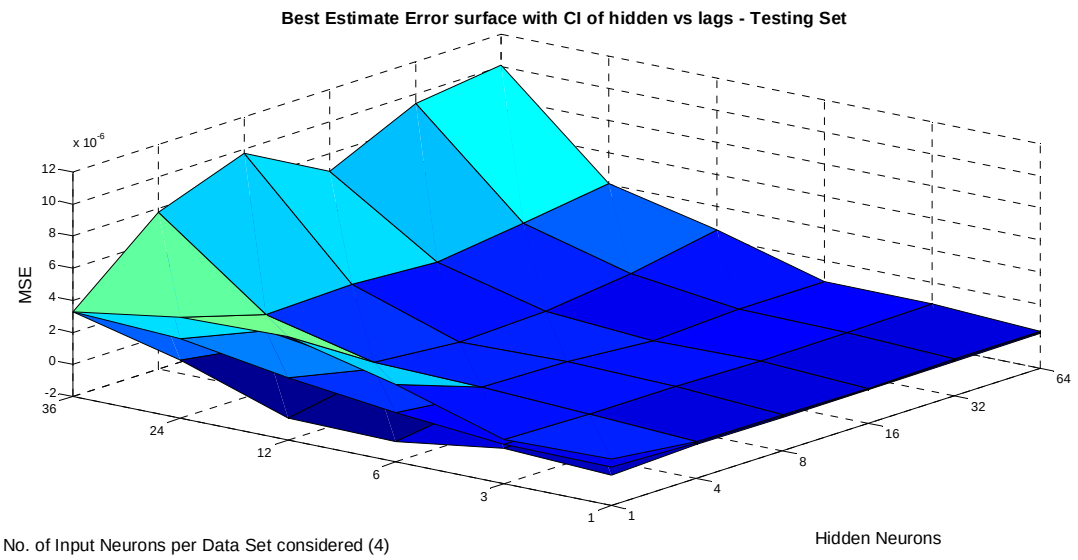

Figure H4: MSE with CI surface – Testing data set

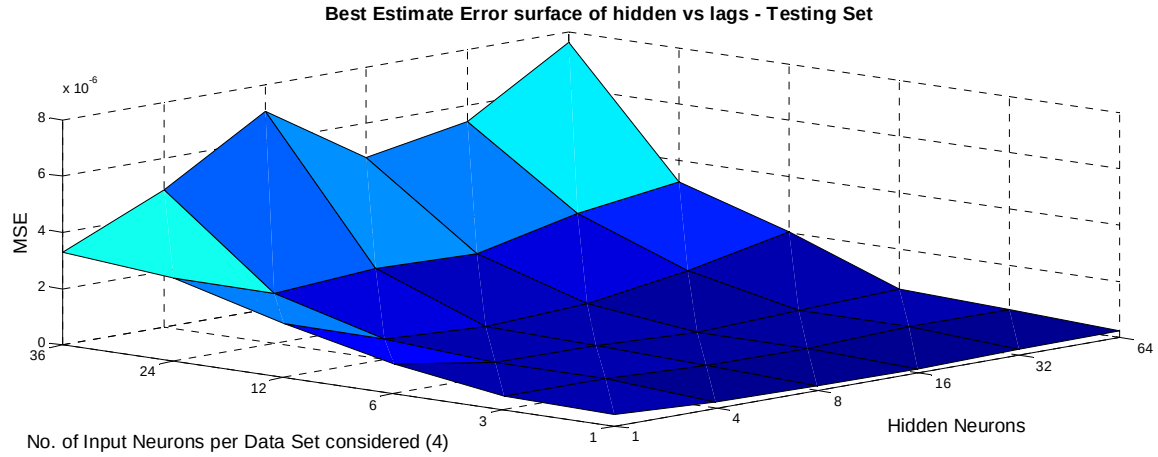

Figure H5: MSE surface – Testing data set

H1.2.5 Numerical Representation of MSE surfaces

The tables below provide the numerical average MSE values used to construct the error surfaces in figures H2, H3, H4 and H5.

Note: in the tables below, the Input Neurons refer to the number of input neurons per data set. Therefore, to determine the total number of input neurons this number was multiplied by 4.

Training Sets							
		LR increase		1.005			
		LR decrease		0.8	(20% decrease)		
	Average MSE vs Input and Hidden						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	1.07E-06	6.12E-07	5.06E-07	4.29E-07	4.80E-07	4.36E-07
	3	8.81E-07	8.20E-07	6.56E-07	5.54E-07	5.38E-07	5.17E-07
	6	1.61E-06	7.25E-07	5.79E-07	**3.38E-07**	4.77E-07	3.69E-07
	12	2.18E-06	6.29E-07	5.07E-07	3.87E-07	5.00E-07	3.99E-07
	24	2.49E-06	8.80E-07	5.66E-07	2.58E-07	1.86E-07	2.56E-07
	36	6.91E-06	9.38E-07	4.59E-07	2.79E-07	1.54E-07	9.00E-08
average	8.24E-07						
min	9.00E-08						
	Upper limit MSE vs Input and Hidden (+2 STD Dev)						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	2.20E-06	7.54E-07	6.74E-07	5.95E-07	5.47E-07	5.43E-07
	3	1.30E-06	1.21E-06	1.34E-06	1.01E-06	7.66E-07	8.61E-07
	6	3.99E-06	1.25E-06	1.17E-06	**4.57E-07**	6.63E-07	4.99E-07
	12	4.31E-06	1.01E-06	7.98E-07	5.42E-07	8.83E-07	5.94E-07
	24	3.82E-06	1.29E-06	1.04E-06	4.07E-07	4.48E-07	6.31E-07
	36	1.73E-05	1.23E-06	9.02E-07	6.33E-07	3.75E-07	2.36E-07
average	1.56E-06						
Efficient Structure							
Minimum Error							

Table H4: MSE of different combinations of input and hidden neurons – Training data set

Testing Sets							
		LR increase		1.005			
		LR decrease		0.8	(20% decrease)		
	Average MSE vs Input and Hidden						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	3.92E-07	2.20E-07	1.76E-07	1.36E-07	2.15E-07	2.55E-07
	3	4.84E-07	4.80E-07	3.22E-07	2.99E-07	4.40E-07	4.37E-07
	6	1.06E-06	4.87E-07	4.51E-07	2.84E-07	4.55E-07	5.73E-07
	12	1.91E-06	7.44E-07	5.62E-07	7.49E-07	1.28E-06	2.05E-06
	24	2.96E-06	1.77E-06	2.03E-06	1.93E-06	2.74E-06	3.25E-06
	36	3.30E-06	4.88E-06	7.05E-06	4.77E-06	5.43E-06	7.63E-06
average	1.73E-06						
min	1.36E-07						
	Upper limit MSE vs Input and Hidden (+2 STD Dev)						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	8.91E-07	2.99E-07	2.49E-07	1.94E-07	2.95E-07	3.15E-07
	3	7.67E-07	6.56E-07	6.50E-07	5.29E-07	5.79E-07	6.75E-07
	6	2.83E-06	9.42E-07	8.24E-07	4.17E-07	7.00E-07	6.98E-07
	12	4.43E-06	1.15E-06	6.76E-07	9.02E-07	1.55E-06	2.57E-06
	24	4.31E-06	2.75E-06	2.94E-06	2.61E-06	3.32E-06	4.08E-06
	36	3.30E-06	7.76E-06	9.72E-06	6.90E-06	9.40E-06	1.01E-05
average	2.53E-06						
Efficient Structure							
Minimum Error							

Table H5: MSE of different combinations of input and hidden neurons – Testing data set

H1.2.6 Conclusion – Input and Hidden Neurons

To balance the error between the varying ANN structures over the training and testing data sets, a structure was chosen which minimized the increase in MSE from the minimum in each data set. The structure chosen consisted of 6 input neurons per data set (24 in total) and 16 hidden neurons. The increase in MSE from the minimum over the training and testing data sets were 2.34E-07 and 1.48E-07, respectively

The choice made here had subjective components. It was important not to base the decision purely on the results over the training set as this would lead to over fitting. Similarly, basing the decision purely on the results from the testing set would lead to fitting the ANN to the testing set. Both scenarios described above would have reduced the generalization ability of the ANN.

H1.2.7 Conclusion of Determining an Efficient Structure

Efficient parameters were:

- 6 input neurons per data set (24 input neurons in total),
- 8 seasonal input neurons,
- 16 hidden neurons,
- A learning rate increase of 1.005 (0.5% increase), and
- A learning rate decrease of 0.8 (20% decrease).

H2 Analysing the Efficient ANN

The ANN was trained with the following parameters.

Network Structure (input neurons per data set x number of data sets) – hidden neurons – output neurons	24 (6x4)-16-1
Training set size	325
Testing set size	100
Dummy variables (for seasonality) (variables per data set x number of data sets)	8 (2x4)
Number of initialized structures	10
Epochs	10 000
Training algorithm	RPROP
Learning rate change: Increase Decrease Maximum Minimum	 0.5% (1.005) 20% (0.8) 100 0.000000001

Table H6: Parameters for the efficient ANN structure

H2.1 Observations of System Error

Figures H6 indicated the error of the system did not increase significantly over the training process. This was true for both the training and testing sets and suggested this system was not capable of over fitting the data. This could only be true if there was little (if any) random variation in the data set. It was concluded that an efficient number of training epochs was 10 000 as the error over both data sets were stable and close to the minimum at that point. The numerical error values at points of interest are provided in table H7.

Error Readings	Testing Set	Training Set
Minimum RMSE	4.7536e-004	4.9600e-004
Minimum MSE	2.2597e-007	2.4602e-007
Final RMSE after training *(10 000 epochs)*	4.8429e-004	4.9613e-004
Final MSE after training *(10 000 epochs)*	2.3454e-007	2.4614e-007

Table H7: Results of efficient model after training

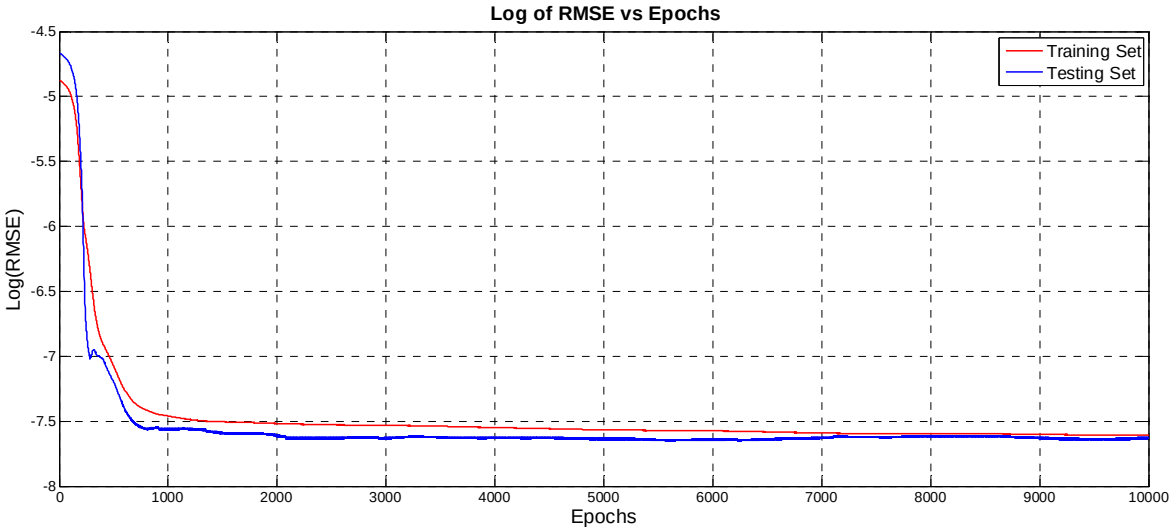

Figure H6: Log of RMSE after 10 000 training epochs

H2.2 Observations of Forecasts and Actuals

The ANN captured the underlying trends in the data set as expected. There were certain periods where the system underestimated the actual values (1984-1986, 1989-1992 and 1999), this could be explained by the effect of external influences on the money market. The best estimate over the training and testing sets was represented by the blue line in figure A7.

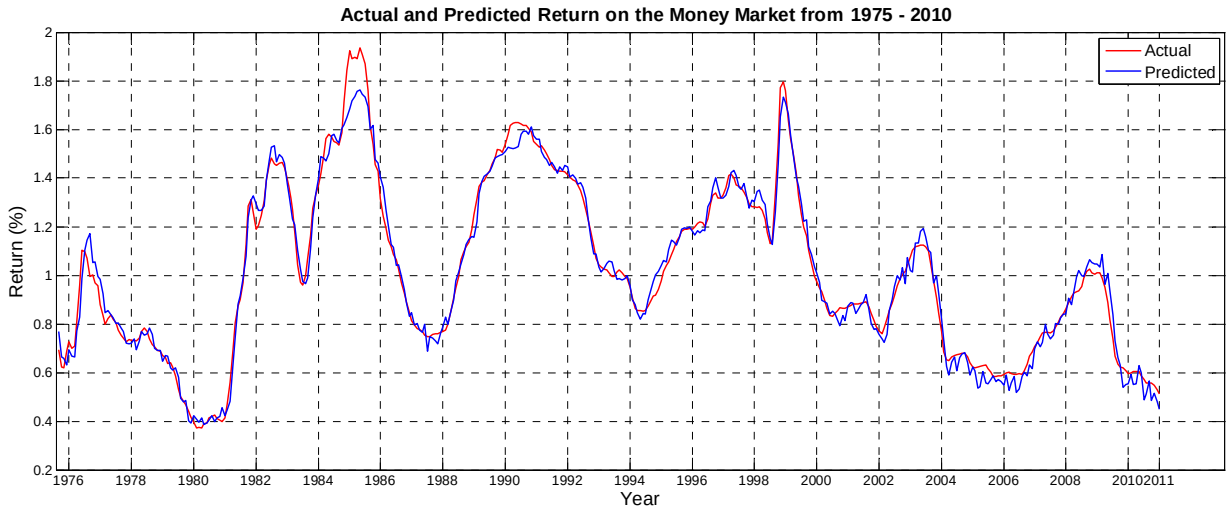

Figure 4.34: Actual and Predicted – Training and Testing Data Sets

H2.3 Conclusion - Analysing the Efficient ANN

The ANN captured the trends underlying the data. The efficient ANN structure had an MSE over the training and testing sets of 2.4614e-007 and 2.3454e-007, respectively. The ANN performed similarly over the testing and training data set. The level of effectiveness was determined by comparing these results to traditional models in a following chapter (Chapter 5). The minimum RMSE over the testing set, of 4.9600e-004, and over the training set, of 4.7536e-004, were obtained after approximately 2 500 and 10 000 epochs, respectively.

The RMSE over the testing set did not show signs of a significant increase as the training approached 10 000 epochs. This was not expected, as it indicated that the system was not susceptible to over fitting. Theoretically, this could only happen when there is little random variation in the data and the underlying trends remain stable.

The error was larger over the training than the testing set. This was expected as there are more unexplainable events over the first 332 observations than over the following 100. These outliers appeared from 1986 – 1988 and 1999, as seen from the large spikes of the actual values in figure 4.36. The training data set contained more unexpected events than the testing set and hence had the greater error of the two. Further, investigation could be done into the reason for this occurrence. This was not done here due to time constraints.

H3 Summary of Experiment

The following table provides a summary of the results from this experiment.

Network Structure (input neurons per data set x number of data sets) – hidden neurons – output neurons	24 (6x4)-16-1
Training epoch required for optimal training	10 000
Minimum Root Mean Squared Error: Training Data Set Testing Data Set	 4.9600e-004 4.7536e-004
Final Root Mean Squared Error (10 000 epochs): Training Data Set Testing Data Set	 4.9613e-004 4.8429e-004
Parameters for RPROP: Learning Rate Increase Learning Rate Decrease	 0.5% (1.005) 20% (0.8)
Data Sets: Training set Testing set	 325 observations (1975 – 2002) 100 observations (2002 – 2010)
Times the structure was initialized	10

Table H8: Summary of experiment's result

Appendix I – Integrated Equity Market ANN - One Month Forecast Results

I1 Determining an Efficient Structure

I1.1 Part 1 – Efficient number of training epochs

The parameters used for this experiment were:

Parameter	Value
Network Structure (input neurons per data set x number of data sets) – hidden neurons – output neurons	144 (36x4)-64-1
Training set size	295
Testing set size	100
Dummy variables (for seasonality) (variables per data set x number of data sets)	8 (2x4)
Number of initialized structures	5
Epochs	1 000
Training algorithm	RPROP
Learning rate change: Increase Decrease Maximum Minimum	 0.5% (1.005) 20% (0.8) 100 0.000000001

Tables I1: Parameters for experiment - Part 1

I1.1.1 Observations – Part 1

Figure I1 indicated the minimum RMSE (0.0632) over the testing set was achieved within the first 1 000 epochs. The RMSE increased after the minimum was achieved (at approximately 330 epochs), until a plateau was reached after approximately 600 epochs. This increase indicated the presence of over fitting and thus random variation in the data set. The RMSE over the training set continued to decrease until the maximum number of epochs was achieved. This was expected.

The RMSE of 0.0632 over the testing set was large when compared to other applications. This indicated historically there were limited relationships between successive return from the equity market. This was expected as it is believed that external factors are the main drivers of demand and supply in the equity market.

		Training Epochs				
	Min RMSE	**100**	**500**	**1 000**	**Δ in RMSE – 100 to 500**	**Δ in RMSE – 500 to 1 000**
RMSE – Testing Set	0.0632	0.2863	0.0717	0.0763	0.2146	-0.0046
RMSE – Training Set	0.0065	0.2864	0.0218	0.0065	0.2646	0.0153

Tables I2: RMSE after different numbers of epochs

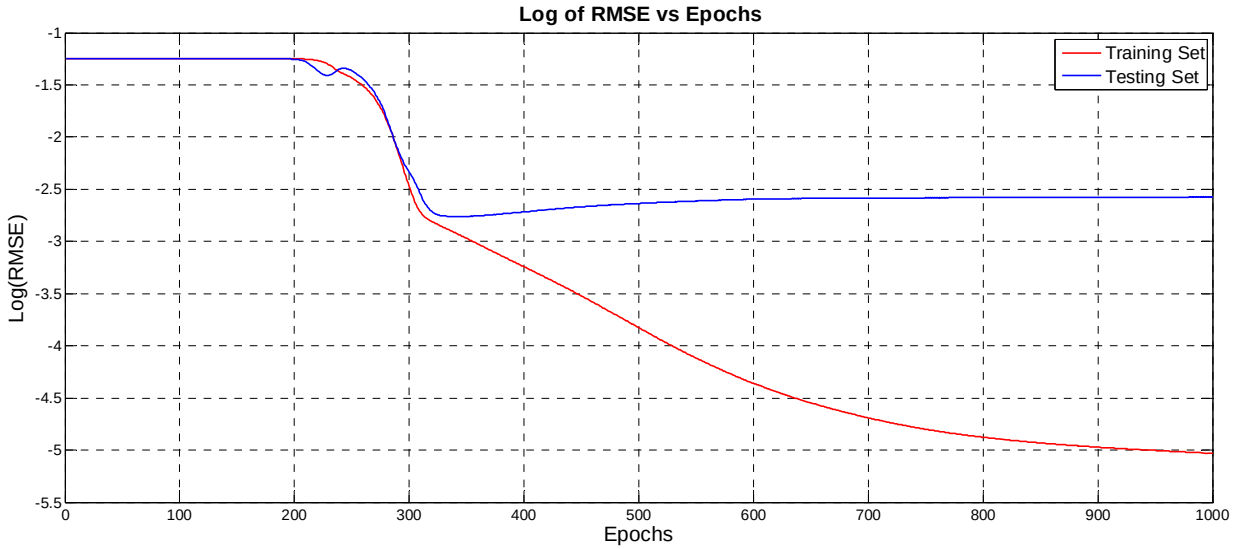

Figure I1: Log of RMSE after different numbers of epochs

I1.1.2 Conclusion – Part 1:

From the results obtained, the number of epochs used to compare different ANN structures was 1 000. This was chosen as it was important to allow the system to reach an over fitted stage, which implied the minimum error over the testing set of data was surpassed and captured. The minimum RMSE over the testing set was 0.0632, the RMSE after 1 000 epochs was 0.0763, and the RMSE after 100 epochs was 0.2863. This indicated the RMSE over the testing set decreased initially and then began to increase as over fitting began. This whole process occurred in the first 1 000 epochs and was supported by the line chart, figure I1, of the errors over each data set.

The ANN considered in this experiment was the most complex and had the slowest learning parameters. Other simpler ANNs were expected to reach their minimum errors within the 1 000 epochs as the learning progressed faster. The

conclusion of an efficient number of learning epochs was primarily based on the results obtained over the testing set. If the decision was made considering only the error over the training set the most complex ANN to be selected due to the increased risk of over fitting in complex ANNs.

I1.2 Part 2 – Determining efficient parameters

I1.2.1 Observations – Learning Rate Increase/Decrease

From the results of the experiment, tabulated in table I3, the minimum MSE over the training set (2.37E-03) was achieved with a learning rate increase of 1.05 (5% increase) and a decrease of 0.8 (20% decrease). The minimum MSE over the testing set (2.93E-03) was achieved with an increase of 1.005 (0.5% increase) and a decrease of 0.2 (80% decrease). These observations did not support the same conclusion.

The shape of the average MSE surface over the training set was considered. The range over the surface of the training set was significant (1.27E-03) in comparison to the range of the average MSE surface over the testing set (0.07E-03). The efficient learning rate changes were determined by calculating the smallest trade off in error over both sets of data.

The efficient learning rate increase and decrease (represented by orange in table I3) were 1.05 and 0.8 as this resulted in the minimum error over the training data set (represented by green in table I3) and an increase in error by 0.08E-03 over the testing data set.

It is important to note the values used to decide on the learning rate changes were the arithmetic average MSE over different combinations of input and hidden neurons. This average did not explicitly allow for the event of outliers in the data and may be skewed. From the investigations there were no outliers found but extensive tests to verify this remark were not performed. Further investigation into the possibility of outliers can be done.

Training Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	3.64E-03	3.10E-03	3.15E-03
	0.5	3.09E-03	2.78E-03	2.79E-03
	0.8	2.62E-03	**2.37E-03**	2.54E-03
	min	2.37E-03		
Testing Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	2.93E-03	2.94E-03	2.98E-03
	0.5	2.98E-03	2.99E-03	3.00E-03
	0.8	3.00E-03	**3.01E-03**	3.00E-03
	min	2.93E-03		
	Minimum			
	Efficient			

Tables I3: Parameters for experiment - Part 1

I1.2.2 Conclusion – Learning Rate Increase/Decrease

The following decisions were made:

- A learning rate decrease of 0.8 was to be used, i.e. a 20% decrease in learning rate if the error of the system increases during training.
- A learning rate increase of 1.05 was to be used, i.e. a 5% increase in learning rate if the error of the system decreases during training.

I1.2.3 Observations – Input and Hidden Neurons – Training Set's MSE surface

The error surface in figure I2 indicated the MSE of the system decreased as the number of input and hidden neurons increased. This was expected since the increased complexity of the system would increase the accuracy over the training data set due to over fitting. The upper and lower surface represents the upper and lower limit of the MSE. These limits were determined by increasing/decreasing the best estimate surface by two standard deviations of the MSE. Considering the best estimate error surface, figure I3, the MSE of the system decreased as the complexity of the system increased which was characteristic of over fitting. The surface obtained the minimum at 36 input and 64 hidden neurons (8.13E-05).

The upper and lower limit of the MSE surface over the training set requires assumptions in order to hold statistical significance. There was insufficient information available to make these assumptions. These limits, however, did provide a good estimate of the MSE limits for the randomly initialized structures.

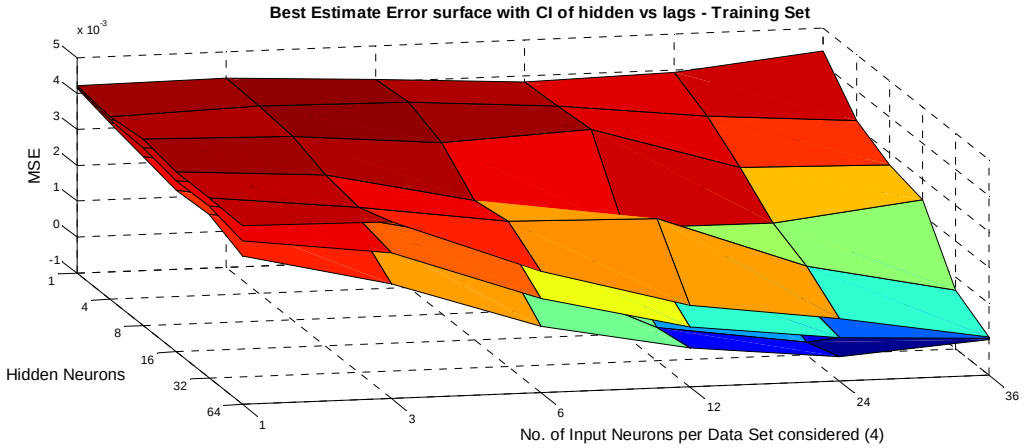

Figure I2: MSE with CI surface – Training data set

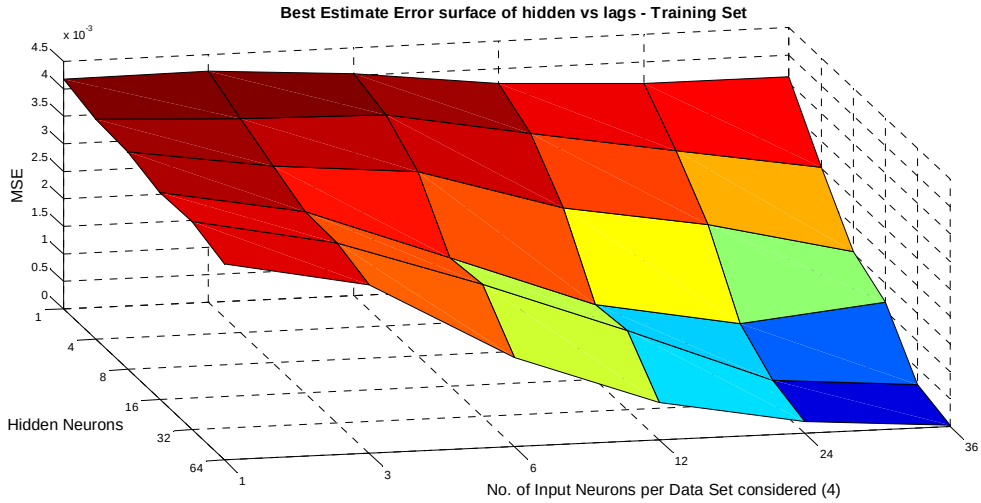

Figure I3: MSE surface – Training data set

I1.2.4 Observations – Input and Hidden Neurons – Testing Set's MSE surface

The MSE surface with limits over the testing set, figure I4, indicated a minimum plain from 1 to 3 input neurons and 1 to 64 hidden neurons, after which it increased. This suggested over fitting was prevalent in this system and that complex structures were likely to over fit the data. The minimum MSE (2.61E-03) on the best estimate MSE surface, figure I5, was achieved using 36 input neurons per data set (144 input neurons in total) and a single hidden neuron. The MSE surface over the testing set was structurally different to that of the training set.

The upper and lower limit of the MSE surface over the testing set requires assumptions in order to hold statistical significance. There was insufficient information available to make these assumptions. These limits, however, did provide a good estimate of the MSE limits for the randomly initialized structures.

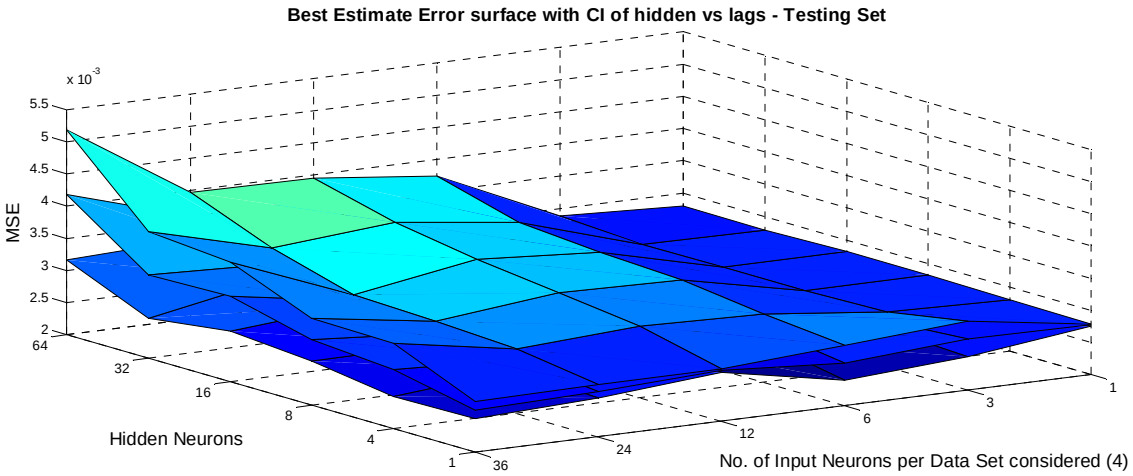

Figure I4: MSE with CI surface – Testing data set

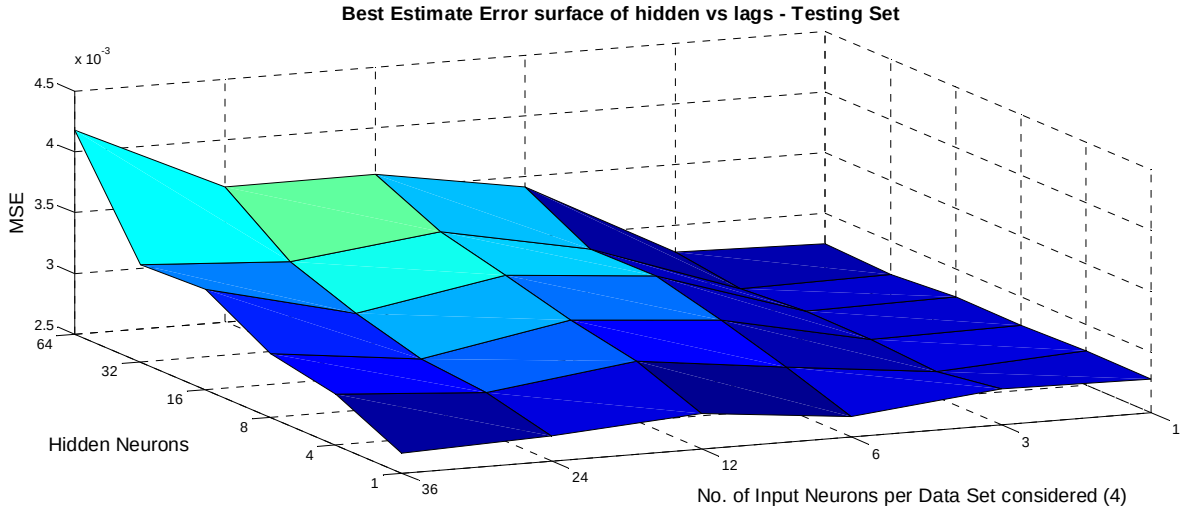

Figure I5: MSE surface – Testing data set

I1.2.5 Numerical Representation of MSE surfaces

The tables below provide the numerical average MSE values used to construct the error surfaces in figures I2, I3, I4 and I5.

Note: in the tables below, the Input Neurons refer to the number of input neurons per data set. Therefore, to determine the total number of input neurons this number was multiplied by 4.

Training Sets							
		LR increase		1.05			
		LR decrease		0.8	(20% decrease)		
	Average MSE vs Input and Hidden						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	4.22E-03	3.91E-03	3.88E-03	3.79E-03	3.41E-03	3.69E-03
	3	4.17E-03	3.63E-03	3.40E-03	2.80E-03	**2.66E-03**	2.85E-03
	6	3.88E-03	3.46E-03	2.67E-03	2.22E-03	1.93E-03	7.57E-04
	12	3.61E-03	3.14E-03	2.10E-03	1.56E-03	5.22E-04	2.92E-04
	24	3.29E-03	2.50E-03	1.54E-03	8.64E-04	2.22E-04	9.85E-05
	36	3.29E-03	2.40E-03	1.25E-03	7.89E-04	3.68E-04	8.13E-05
average	2.37E-03						
min	8.13E-05						
	Upper limit MSE vs Input and Hidden (+2 STD Dev)						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	4.24E-03	4.08E-03	4.31E-03	4.15E-03	3.92E-03	4.21E-03
	3	4.34E-03	3.84E-03	3.73E-03	3.62E-03	**4.09E-03**	3.98E-03
	6	3.98E-03	3.72E-03	3.68E-03	3.74E-03	4.53E-03	2.24E-03
	12	3.76E-03	3.35E-03	3.62E-03	2.74E-03	1.53E-03	7.64E-04
	24	3.50E-03	2.81E-03	2.70E-03	2.25E-03	7.05E-04	3.17E-04
	36	3.65E-03	2.68E-03	2.34E-03	2.42E-03	1.20E-03	2.34E-04
average	3.08E-03						
Efficient Structure							
Minimum Error							

Table I4: MSE of different combinations of input and hidden neurons – Training data set

Testing Sets							
		LR increase		1.05			
		LR decrease		0.8	(20% decrease)		
	Average MSE vs Input and Hidden						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	2.85E-03	2.83E-03	2.81E-03	2.80E-03	2.78E-03	2.75E-03
	3	2.75E-03	2.86E-03	2.86E-03	2.63E-03	2.74E-03	2.74E-03
	6	2.78E-03	2.88E-03	3.14E-03	3.02E-03	3.04E-03	3.14E-03
	12	2.73E-03	3.07E-03	3.19E-03	3.32E-03	3.41E-03	3.44E-03
	24	2.73E-03	2.91E-03	2.94E-03	3.24E-03	3.46E-03	3.79E-03
	36	2.61E-03	2.79E-03	2.98E-03	3.23E-03	3.41E-03	3.78E-03
average	3.01E-03						
min	2.61E-03						
	Upper limit MSE vs Input and Hidden (+2 STD Dev)						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	2.96E-03	2.95E-03	2.89E-03	2.84E-03	2.88E-03	2.85E-03
	3	3.16E-03	3.10E-03	2.94E-03	2.70E-03	2.96E-03	2.94E-03
	6	3.00E-03	3.19E-03	3.36E-03	3.31E-03	3.47E-03	3.40E-03
	12	2.82E-03	3.15E-03	3.55E-03	3.77E-03	3.67E-03	3.77E-03
	24	2.79E-03	3.03E-03	3.07E-03	3.58E-03	4.04E-03	4.78E-03
	36	2.71E-03	2.99E-03	3.26E-03	3.56E-03	4.18E-03	4.70E-03
average	3.29E-03						
Efficient Structure							
Minimum Error							

Table I5: MSE of different combinations of input and hidden neurons – Testing data set

I1.2.6 Conclusion – Input and Hidden Neurons

In order to balance the error between the structures over the training and testing data sets, a structure was chosen which minimized the reduction of accuracy. The structure chosen had 3 input neurons per data set (12 input neurons in total) and 32 hidden neurons. The increase in MSE over the training and testing data sets were 0.13E-03 and 2.58E-03, respectively.

The choice made here had subjective components. It was important not to base the decision purely on the results over the training set as this would lead to over fitting. Similarly, basing the decision purely on the results over the testing set would lead to the fitting of the ANN to the testing set. Both the scenarios described above would have reduced the generalization ability of the ANN.

I1.2.7 Conclusion of Determining an Efficient Structure

Efficient parameters were:

- 3 input neurons per data set (12 input neurons in total),
- 32 hidden neurons,
- A learning rate increase of 1.05 (5% increase), and
- A learning rate decrease of 0.8 (20% decrease).

I2 Analysing the Efficient ANN

The ANN was trained with the following parameters.

Network Structure (input neurons per data set x number of data sets) – hidden neurons – output neurons	12 (3x4)-32-1
Training set size	328
Testing set size	100
Dummy variables (for seasonality) (variables per data set x number of data sets)	8 (2x4)
Number of initialized structures	10
Epochs	3 000
Training algorithm	RPROP
Learning rate change: Increase Decrease Maximum Minimum	 5% (1.05) 20% (0.8) 100 0.000000001

Table I6: Parameters for the efficient ANN structure

I2.1 Observations of System Error

Figure I6 indicated after 500 training epochs the system began to over fit and the ANN began to learn the random variation in the data set. This was represented by the RMSE over the testing set rising. The minimum RMSE of the system (0.0498) over the testing set was obtained after approximately 500 training epochs. The error over the testing and training set intersected after approximately 2 200 training epochs, which was used as the efficient number of epochs for this structure. This number of epochs balanced both the error over the training and testing sets. Figure I6 illustrates the log of the Root Mean Squared Error (Log (RMSE)) of the system over 3 000 training epochs. The numerical error value at points of interest are provided in table I7.

Error Readings	Testing Set	Training Set
Minimum RMSE	0.0498	0.0498
Minimum MSE	0.0025	0.0025
Final RMSE after training *(2 200 epochs)*	0.0520	0.0514
Final MSE after training *(2 200 epochs)*	0.0027	0.0026

Table I7: Results of efficient model after training

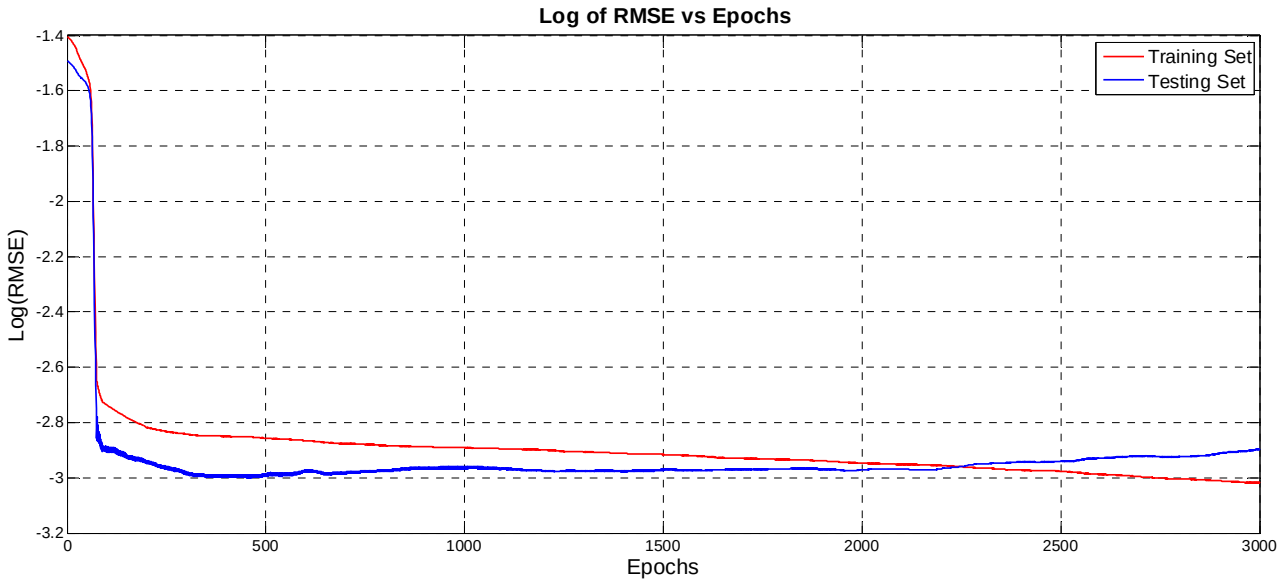

Figure I6: Log of RMSE after 3 000 training epochs

I2.2 Observations of Forecasts and Actuals

The ANN captured the underlying trends in the data set as expected. There were several occasions were the ANN either under or overestimated the return on the market. This was true for both the training and testing data sets. These instances were the result of external factors on the market and could not be explained using only past observations of the equity market. The best estimate over the training and testing sets was the blue line in figure I7.

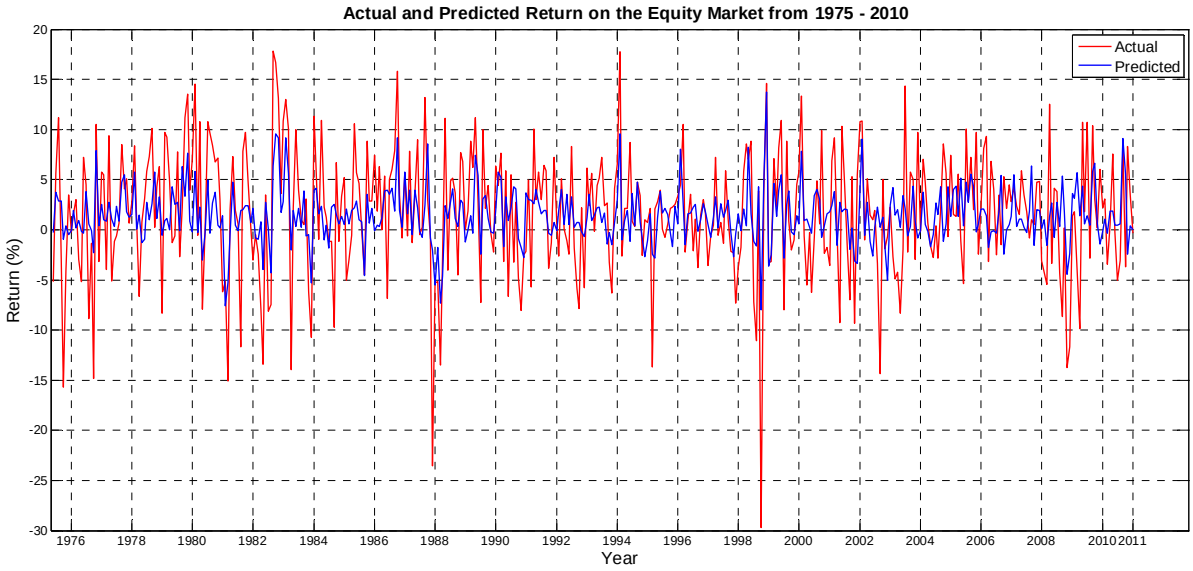

Figure I9: Actual and Predicted – Training and Testing Data Sets

I2.3 Conclusion - Analysing the Efficient ANN

The ANN captured several trends underlying the data. The efficient ANN structure had an MSE over the training and testing sets of 0.0026 and 0.0027, respectively. The ANN performed similarly over the testing and training data set. The level of effectiveness was determined by comparing these results to traditional models in a following chapter (Chapter 5). The minimum RMSE over the testing set, of 0.0498, and over the training set, of 0.0498, was obtained after approximately 500 and 3 000 epochs, respectively. The RMSE of the training set decreased as the training went over 500 epochs and continued to decrease until the maximum number of epochs were obtained. The efficient number of epochs used was 2 200 which resulted in a RMSE of 0.0514 over the training set and 0.0520 over the testing data set. Theoretically, this was expected as ANNs have the ability to mimic a data set precisely. This mimicking leads to over fitting and reduces the systems generalization ability. To avoid over fitting but to ensure the underlying relationships in the data were captured, decisions relating to efficiency were made with reference to the performance of the ANN over both the testing and training data set.

I3 Summary of Experiment

The following table provides a summary of the results from this experiment.

Network Structure (input neurons per data set x number of data sets) – hidden neurons – output neurons	12 (3x4)-32-1
Training epoch required for optimal training	2 200
Minimum Root Mean Squared Error: Training Data Set Testing Data Set	 0.0498 0.0498
Final Root Mean Squared Error (2 200 epochs): Training Data Set Testing Data Set	 0.0514 0.0520
Parameters for RPROP: Learning Rate Increase Learning Rate Decrease	 5% (1.05) 20% (0.8)
Data Sets: Training set Testing set	 328 observations (1975 – 2002) 100 observations (2002 – 2010)
Times the structure was initialized	10

Table I8: Summary of experiment's result

Appendix J – Integrated Inflation ANN – Three-Month Forecast Results

J1 Determining an Efficient

J1.1 Part 1 – Efficient number of training epochs

The parameters used for this experiment were:

Parameter	Value
Network Structure (input neurons per data set x number of data sets) – hidden neurons – output neurons	144 (36x4)-64-1
Training set size	293
Testing set size	100
Dummy variables (for seasonality) (variables per data set x number of data sets)	8 (2x4)
Number of initialized structures	5
Epochs	1 000
Training algorithm	RPROP
Learning rate change: Increase Decrease Maximum Minimum	 0.5% (1.005) 20% (0.8) 100 0.000000001

Tables J1: Parameters for experiment - Part 1

J1.1.1 Observations – Part 1

Figure J1 indicated the minimum RMSE (0.0062) over the testing set was achieve within the first 350 epochs. The error began to increase from this point onwards, represented by the trough of the blue line after approximately 350 training epochs. The increase of the RMSE indicated the system was over fitting to this particular data set. The minimum RMSE (8.9229e-004) over the training set was achieve at the 1 000 epoch mark, as expected.

The occurrence of over fitting in the ANN indicated the presence of noise in the data. This randomness could not be explained by past observations of inflation. This supported the theory which stipulated that inflation is largely affected

by external factors, such the country's monetary policy and only a finite number of trends can be explained through past observations.

		Training Epochs				
	Min RMSE	**100**	**500**	**1 000**	**Δ in RMSE – 100 to 500**	**Δ in RMSE – 500 to 1 000**
RMSE – Testing Set	0.0062	0.0391	0.0070	0.0075	0.0321	-0.0005
RMSE – Training Set	8.9229e-004	0.0345	0.0020	8.9229e-004	0.0325	0.0011

Tables J2: RMSE after different numbers of epochs

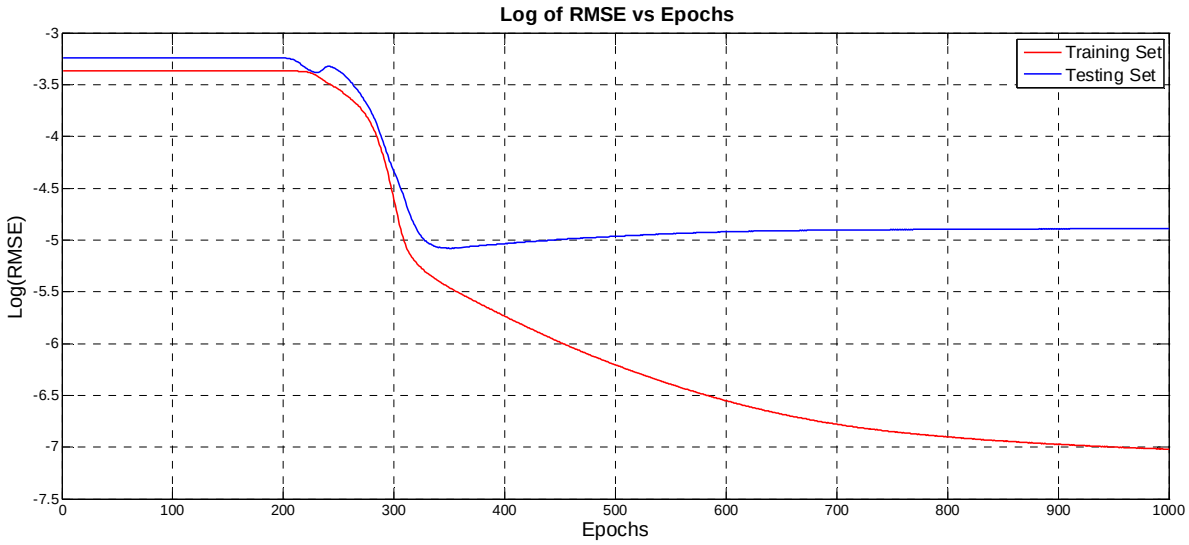

Figure J1: Log of RMSE over training process

J1.1.2 Conclusion – Part 1:

From the results, the number of epochs used to compare different ANN structures was 1 000. This was chosen to allow the system to over fit the data. Over fitting the data ensured the minimum error over the testing set was surpassed and captured. This process ensured different ANN structures could be compared consistently. The minimum RMSE over the testing set was 0.0062 and the RMSE after 1 000 epochs was 0.0075. This lead to the achievement of the minimum RMSE within the first 1 000 epochs. This inference was supported by figure J1 by the trough created by the blue line at approximately 350 epochs.

This ANN was the most complex and had the slowest learning parameters. Other simpler ANNs were expected to reach their minimum errors within the 1 000 epochs as the learning was faster. The conclusion of an efficient number of learning epochs was primarily based on the results obtained over the testing set. Primarily considering the error

over the training set would cause the most complex ANN to be selected due to the increased risk of over fitting in complex ANNs.

J1.2 Part 2 – Determining efficient parameters

J1.2.1 Observations – Learning Rate Increase/Decrease

From table J3 the minimum MSE over the training set (1.83E-05) was achieved with a learning rate increase of 1.05 (5% increase) and a decrease of 0.8 (20% decrease). The minimum MSE over the testing set (3.17E-05) was achieved with an increase of 1.005 (0.5% increase) and a decrease of 0.8 (20% decrease). These observations did not support a similar conclusion.

The shape of the average MSE surface over the training set was considered. It was observed the range over the training surface was significant (0.93E-05) in comparison to the range of the MSE surface over the testing set (0.28E-05). The efficient learning rate changes were chosen by determining the smallest trade off in error over both sets of data.

The efficient learning rate increase and decrease chosen was 1.005 and 0.8, respectively. They resulted in a fair trade off between the accuracy over the associated data sets. The increase in average MSE over the training and testing sets were 0.03E-05 and 0, respectively. The MSE associated with the chosen learning rate increase and decrease are marked in orange in table J3 and the minimum MSE are marked in green.

It is important to note the values used to decide on the learning rate changes were the arithmetic average MSE over different combinations of input and hidden neurons. This average did not explicitly allow for the event of outliers in the data and may be skewed. From the investigations there were no outliers found but extensive tests to verify this remark were not performed. Further investigation into the possibility of outliers can be done.

Training Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	2.76E-05	2.37E-05	2.53E-05
	0.5	2.29E-05	2.11E-05	2.19E-05
	0.8	**1.86E-05**	1.83E-05	2.03E-05
	min	1.83E-05		
Testing Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	3.45E-05	3.34E-05	3.35E-05
	0.5	3.27E-05	3.25E-05	3.28E-05
	0.8	**3.17E-05**	3.21E-05	3.31E-05
	min	3.17E-05		
	Minimum			
	Efficient			

Tables J3: MSE for Learning Rate Changes

J1.2.2 Conclusion – Learning Rate Increase/Decrease

The following decisions were made:

- A learning rate decrease of 0.8 was to be used, i.e. a 20% decrease in learning rate if the error of the system increases during training.
- A learning rate increase of 1.005 was to be used, i.e. a 0.5% increase in learning rate if the error of the system decreases during training.

J1.2.3 Observations – Input and Hidden Neurons – Training Set's MSE surface

The MSE surface with limits, figure J2, indicated the MSE of the system decreased as the number of input and hidden neurons increased. This was expected since the increased complexity of the system would increase the probability of over fitting. Over fitting would lead to increased accuracy over the training data set. The upper and lower surface represents the upper and lower limit of the MSE. The limits were determined by increasing/decreasing the best estimate surface by two standard deviations of the MSE.

Considering the best estimate error surface, figure J3, the MSE of the system decreased as the complexity of the system increased. The surface obtained the minimum at 36 input and 64 hidden neurons (4.1E-07). This was as expected.

The upper and lower limit of the MSE surface over the training set requires assumptions in order to hold statistical significance. There was insufficient information available to make these assumptions. These limits, however, did provide a good estimate of the MSE limits for the randomly initialized structures.

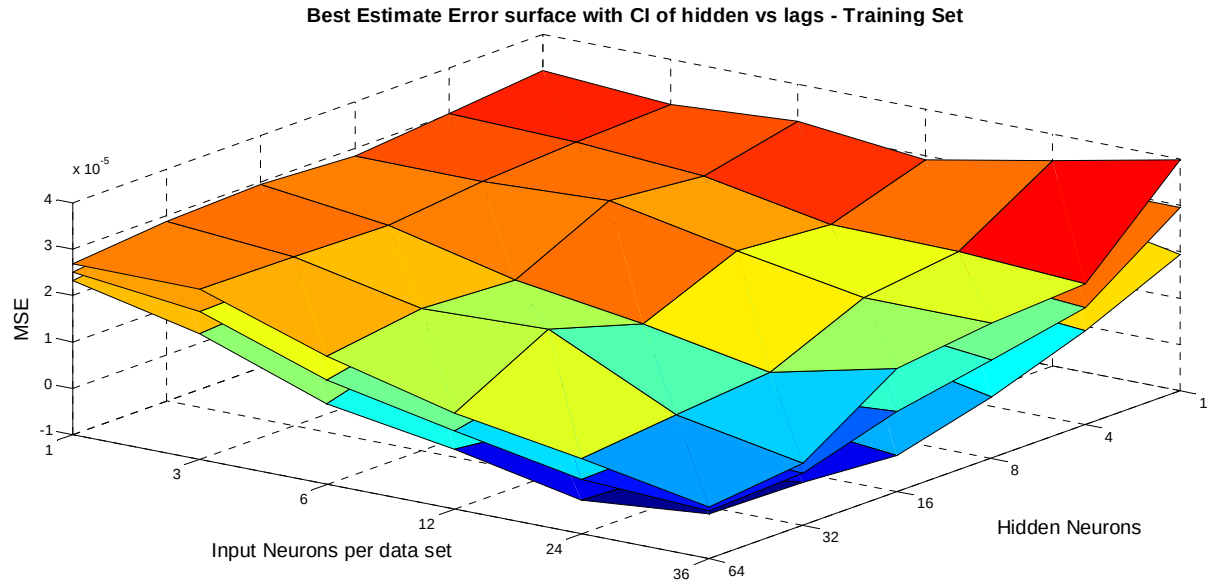

Figure J2: MSE with CI surface – Training data set

Figure J3: MSE surface – Training data set

J1.2.4 Observations – Input and Hidden Neurons – Testing Set's MSE surface

The MSE surface over the testing set with limits, displayed in figure J4, indicated a rough surface with a valley when 12 input neurons were considered. There were no trends visible with varying hidden neurons.

The best estimate MSE surface, figure J5, indicated a minimum basin when 12 input and 16 hidden neurons were considered. The minimum MSE of the best estimate surface was achieved using 12 input and 16 hidden neurons (2.62E-05). The MSE surface over the testing set was structurally different to that of the training set.

The upper and lower limit of the MSE surface over the testing set requires assumptions in order to hold statistical significance. There was insufficient information available to make these assumptions. These limits, however, did provide a good estimate of the MSE limits for the randomly initialized structures.

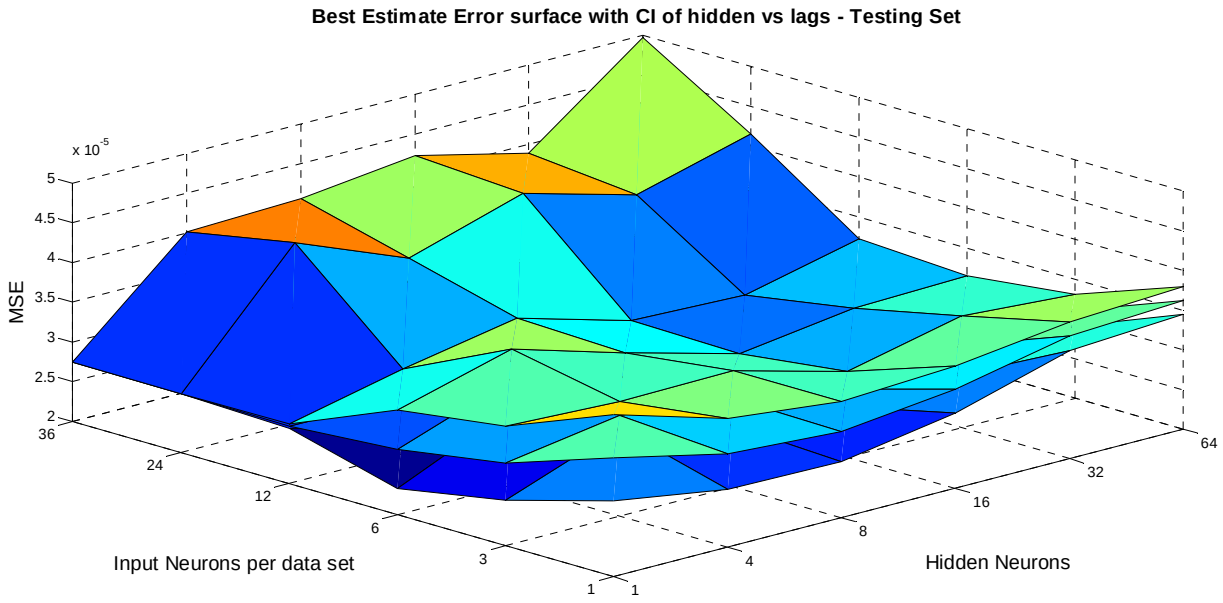

Figure J4: MSE with CI surface – Testing data set

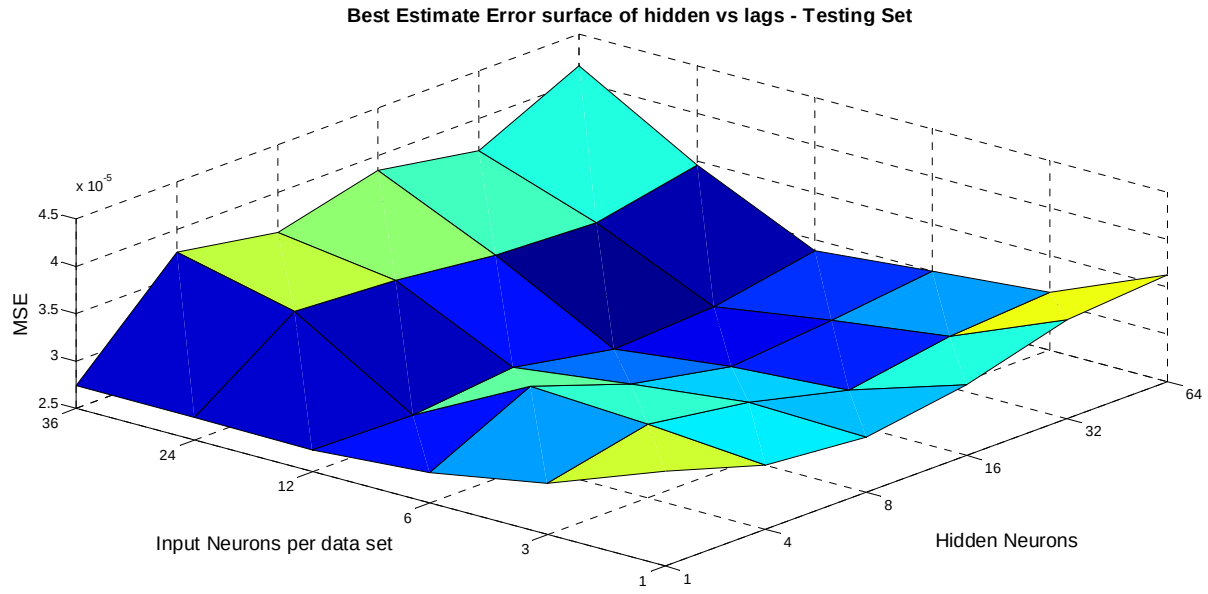

Figure J5: MSE surface – Testing data set

J1.2.5 Numerical Representation of MSE surfaces

The tables below provide the numerical average MSE values used to construct the error surfaces in figures J2, J3, J4 and J5.

Note: in the tables below, the Input Neurons refer to the number of input neurons per data set. Therefore, to determine the total number of input neurons this number was multiplied by 4.

Training Sets							
		LR increase		1.005			
		LR decrease		0.8	(20% decrease)		
	Average MSE vs Input and Hidden						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	3.10E-05	2.90E-05	2.75E-05	2.71E-05	2.73E-05	2.51E-05
	3	3.00E-05	2.78E-05	2.49E-05	2.25E-05	2.25E-05	2.19E-05
	6	3.02E-05	2.63E-05	2.39E-05	1.58E-05	1.69E-05	1.25E-05
	12	2.74E-05	2.07E-05	1.74E-05	**9.58E-06**	1.12E-05	6.83E-06
	24	2.96E-05	1.73E-05	1.40E-05	5.48E-06	2.36E-06	1.80E-06
	36	2.96E-05	1.53E-05	1.11E-05	7.25E-06	1.32E-06	4.10E-07
average	1.86E-05						
min	4.10E-07						
	Upper limit MSE vs Input and Hidden (+2 STD Dev)						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	3.24E-05	3.03E-05	2.83E-05	2.95E-05	2.87E-05	2.69E-05
	3	3.03E-05	2.95E-05	2.83E-05	2.61E-05	2.64E-05	2.67E-05
	6	3.21E-05	2.75E-05	2.95E-05	1.97E-05	2.06E-05	1.76E-05
	12	2.91E-05	2.24E-05	2.40E-05	**1.54E-05**	2.16E-05	1.08E-05
	24	3.43E-05	2.20E-05	1.92E-05	1.03E-05	8.43E-06	6.27E-06
	36	3.98E-05	2.04E-05	1.91E-05	1.66E-05	3.43E-06	1.15E-06
average	2.26E-05						
Efficient Structure							
Minimum Error							

Table J4: MSE of different combinations of input and hidden neurons – Training data set

Testing Sets							
		LR increase		1.005			
		LR decrease		0.8	(20% decrease)		
	Average MSE vs Input and Hidden						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	3.50E-05	3.18E-05	3.09E-05	3.25E-05	3.54E-05	3.62E-05
	3	3.04E-05	3.28E-05	3.12E-05	2.86E-05	3.04E-05	3.11E-05
	6	2.82E-05	3.34E-05	2.98E-05	2.77E-05	2.87E-05	3.00E-05
	12	2.73E-05	2.71E-05	2.82E-05	**2.62E-05**	2.68E-05	2.88E-05
	24	2.74E-05	3.47E-05	3.41E-05	3.28E-05	3.24E-05	3.45E-05
	36	2.74E-05	3.76E-05	3.58E-05	3.84E-05	3.66E-05	4.17E-05
average	3.17E-05						
min	2.62E-05						
	Upper limit MSE vs Input and Hidden (+2 STD Dev)						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	4.04E-05	3.63E-05	3.47E-05	3.54E-05	3.72E-05	3.80E-05
	3	3.51E-05	3.44E-05	3.46E-05	3.08E-05	3.41E-05	3.30E-05
	6	3.32E-05	3.72E-05	3.29E-05	2.91E-05	3.11E-05	3.15E-05
	12	2.75E-05	3.07E-05	3.34E-05	**2.94E-05**	2.88E-05	3.22E-05
	24	2.74E-05	4.27E-05	3.71E-05	4.15E-05	3.76E-05	4.15E-05
	36	2.74E-05	4.01E-05	4.06E-05	4.23E-05	3.89E-05	4.98E-05
average	3.52E-05						
Efficient Structure							
Minimum Error							

Table J5: MSE of different combinations of input and hidden neurons – Testing data set

J1.2.6 Conclusion – Input and Hidden Neurons

To balance the error between the varying ANN structures over the training and testing data sets, a structure was chosen which minimized the increase in MSE from the minimum in each data set while remaining parsimonious. The structure chosen consisted of 12 input neurons per data set (48 in total) and 16 hidden neurons. The increase in MSE from the minimum over the training and testing data sets were 0.917E-05 and 0, respectively

The choice made here had subjective components. It was important not to base the decision purely on the results over the training set as this would lead to over fitting. Similarly, basing the decision purely on the results from the testing set would lead to fitting the ANN to the testing set. Both scenarios described above would have reduced the generalization ability of the ANN.

J1.2.7 Conclusion of Determining an Efficient Structure

Efficient parameters were:

- 12 input neurons per data set (48 input neurons in total),
- 8 seasonal input neurons,
- 16 hidden neurons,
- A learning rate increase of 1.005 (0.5% increase), and
- A learning rate decrease of 0.8 (20% decrease).

J2 Analysing the Efficient ANN

The ANN was trained with the following parameters.

Network Structure (input neurons per data set x number of data sets) – hidden neurons – output neurons	48 (12x4)-16-1
Training set size	317
Testing set size	100
Dummy variables (for seasonality) (variables per data set x number of data sets)	8 (2x4)
Number of initialized structures	10
Epochs	5 000
Training algorithm	RPROP
Learning rate change: Increase Decrease Maximum Minimum	 0.5% (1.005) 20% (0.8) 100 0.000000001

Table J6: Parameters for the efficient ANN structure

J2.1 Observations of System Error

The error over the training and testing sets was stable after 500 training epochs as indicated by figure J6. The error over the training set continued to decrease as the number of epochs approached its maximum. This was expected as ANNs are known to over fit the data set. The error over the testing set reached a minimum after approximately 200 epochs. The error over the testing set increased slightly between 500 and 2 500 epochs. It decreased after the 2 500 epoch mark and reached a minimum after 4 000 epochs, after which it increased again.

The minimum RMSE over the training set was achieved after 5 000 epochs. The minimum RMSE over the training and testing sets was 0.0045 and 0.0052, respectively.

The efficient number of epoch chosen was 4 000 as it resulted in a RMSE near its minimum over both training and testing data sets. Further, the RMSE over both sets remained stable around this point.

Error Readings	**Testing Set**	**Training Set**
Minimum RMSE	0.0052	0.0045
Minimum MSE	2.7173e-005	2.0190e-005
Final RMSE after training *(4 000 epochs)*	0.0052	0.0047
Final MSE after training *(4 000 epochs)*	2.7240e-005	2.1797e-005

Table J7: Results of efficient model after training

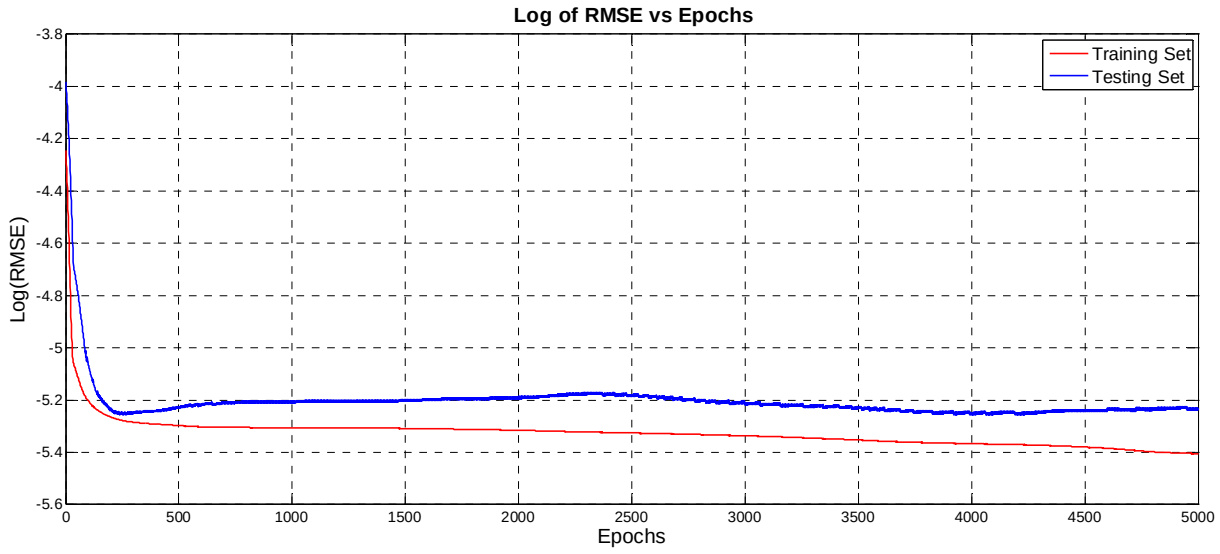

Figure J6: Log of RMSE after 5 000 training epochs

J2.2 Observations of Forecasts and Actuals

The ANN captured the underlying trends in the data set as expected. There were several occasions were the ANN either under or overestimated inflation. This was true for both the training and testing data sets. These instances were the result of external factors on inflation and could not be explained using only past observations of inflation. The best estimate over the training and testing sets is represented by the blue line in figure J7.

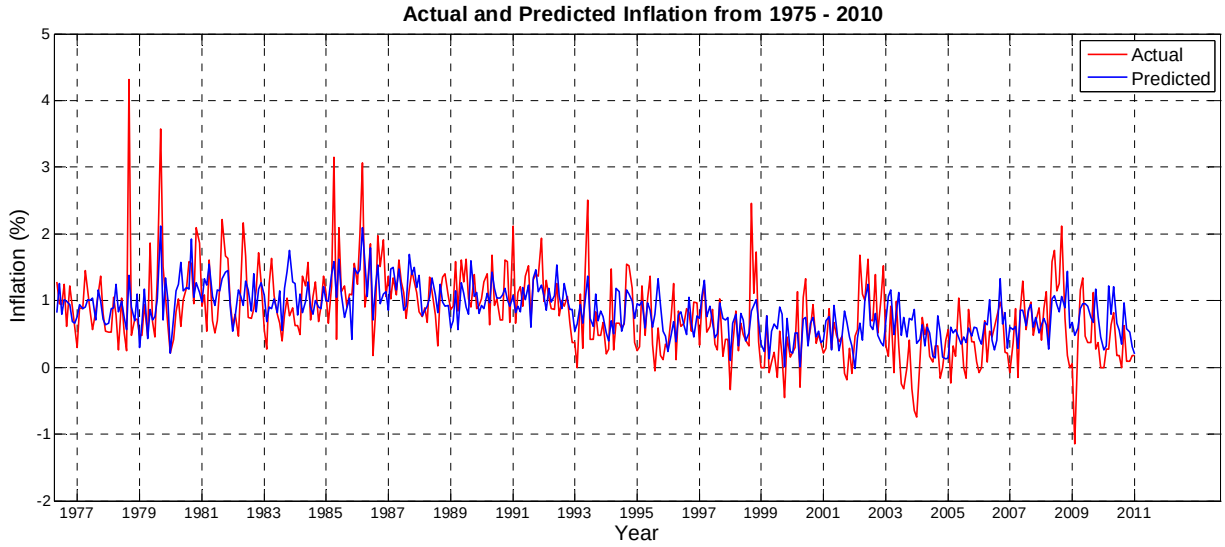

Figure J7: Actual and Predicted – Training and Testing Data Sets

J2.3 Conclusion - Analysing the Efficient ANN

The ANN captured several trends underlying the data. The efficient ANN structure had an MSE over the training and testing sets of 2.1797e-005 and 2.7240e-005, respectively. The ANN performed similarly over the testing and training data set. The level of effectiveness was determined by comparing these results to traditional models in a following chapter (Chapter 5). The minimum RMSE over the testing set, of 0.0052, and over the training set, of 0.0045, was obtained after approximately 200 and 5 000 epochs, respectively. The RMSE of the training set decreased as the training went over 200 epochs and continued to decrease until the maximum number of epochs were obtained. The efficient number of epochs used was 4 000 which resulted in a RMSE of 0.0047 over the training data set and 0.0052 over the testing data set. To avoid over fitting but to ensure the underlying relationships in the data were captured, decisions relating to efficiency were made with reference to the performance of the ANN over both the testing and training data set.

J3 Summary of Experiment

The following table provides a summary of the results from this experiment

Network Structure (input neurons per data set x number of data sets) – hidden neurons – output neurons	48 (12x4)-16-1
Training epoch required for optimal training	4 000
Minimum Root Mean Squared Error: Training Data Set Testing Data Set	 0.0045 0.0052
Final Root Mean Squared Error (4 000 epochs): Training Data Set Testing Data Set	 0.0047 0.0052
Parameters for RPROP: Learning Rate Increase Learning Rate Decrease	 0.5% (1.005) 20% (0.8)
Data Sets: Training set Testing set	 317 observations (1976 – 2002) 100 observations (2002 – 2010)
Times the structure was initialized	10

Table J8: Summary of experiment's result

Appendix K – Integrated Money Market ANN – Three-Month Forecast Results

K1 Determining an Efficient Structure

K1.1 Part 1 – Efficient number of training epochs

The parameters used for this experiment were:

Parameter	Value
Network Structure (input neurons per data set x number of data sets) – hidden neurons – output neurons	144 (36x4)-64-1
Training set size	293
Testing set size	100
Dummy variables (for seasonality) (variables per data set x number of data sets)	8 (2x4)
Number of initialized structures	5
Epochs	1 000
Training algorithm	RPROP
Learning rate change: Increase Decrease Maximum Minimum	 0.5% (1.005) 20% (0.8) 100 0.000000001

Tables K1: Parameters for experiment - Part 1

K1.1.1 Observations – Part 1

The error over the training and testing sets was stable after 1 000 training epochs as indicated by figure K1. The error over the training set continued to decrease as the number of epochs approached its maximum. This was expected as ANNs are known to over fit the data set. The error over the testing set formed a plateau after approximately 400 epochs as illustrated by the horizontal portion of the blue line in figure K1. The minimum RMSE over the training and testing sets was 3.6451e-004 and 0.0032, respectively. The minimum error over the training set was obtained after the maximum number of epochs, 1 000 epochs in this case. The minimum error over the testing set was achieved after approximately 500 epochs after which there was a slight increase.

The RMSE over the testing set did not increase significantly over an increased number of training epochs. This indicated there were no signs of over fitting in this system, which was unexpected. This anomaly may be explained through consideration of the nature of the money market, which is highly predictable. Further, this indicated there was low levels of noise in the data set.

		Training Epochs				
	Min RMSE	**100**	**500**	**1 000**	**Δ in RMSE – 100 to 500**	**Δ in RMSE – 500 to 1 000**
RMSE – Testing Set	0.0032	0.0119	0.0032	0.0034	0.0087	-0.0002
RMSE – Training Set	3.6451e-004	0.0088	7.2799e-004	3.6451e-004	0.00807	3.6e-004

Tables K2: RMSE after different numbers of epochs

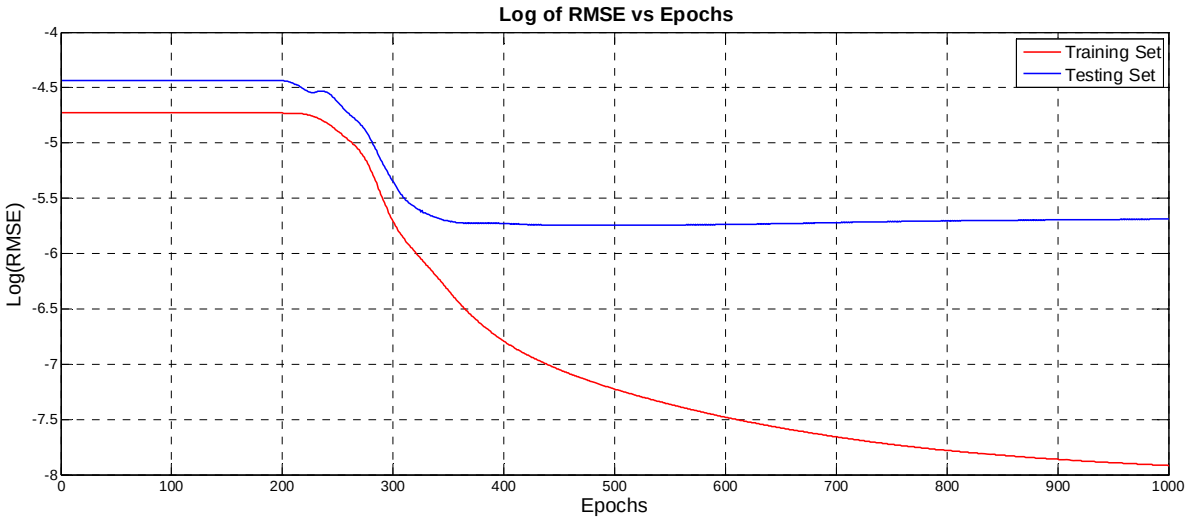

Figure K1: Log of RMSE over training process

K1.1.2 Conclusion – Part 1:

The results indicated 1 000 epochs should be used to compare different ANN structures. After 500 epochs the system showed signs of slight over fitting. This was ideal for this experiment as this indicated the minimum error over the testing set was surpassed and captured. The minimum RMSE over the testing set was 0.0032, and the RMSE after 1 000 epochs was 0.0034. The increase in RMSE from the minimum to the 1 000 epoch mark confirmed the minimum error was achieved within the first 1 000 epochs. This was supported by the slight trough created by the blue line at approximately 500 epochs in figure K1.

This ANN was the most complex and had the slowest learning parameters. Other simpler ANNs were expected to reach their minimum errors within the 1 000 epochs as the learning progressed faster. The conclusion of an efficient

number of learning epochs was primarily based on the results obtained over the testing set. If the decision was made considering only the error over the training set the most complex ANN to be selected due to the increased risk of over fitting in complex ANNs. Ideally, a larger number of epochs should be used, however, this would have increased the computing time considerably and was not feasible in this case.

K1.2 Part 2 – Determining efficient parameters

K1.2.1 Observations – Learning Rate Increase/Decrease

The minimum MSE over the training set (1.32E-06) was achieved with a learning rate increase of 1.005 (0.5% increase) and a decrease of 0.8 (20% decrease), tabulated in table K3. The minimum MSE over the testing set (2.72E-06) was achieved with an increase of 1.005 (0.5% increase) and a decrease of 0.8 (20% decrease).

The efficient learning rate increase and decrease chosen were 1.005 and 0.8 as they resulted in the minimum MSE over both the training and testing data sets. The MSE associated with the chosen learning rate increase and decrease and are marked in green in table K3.

It is important to note the values used to decide on the learning rate changes were the arithmetic average MSE over different combinations of input and hidden neurons. This average did not explicitly allow for the event of outliers in the data and may be skewed. From the investigations there were no outliers found but extensive tests to verify this remark were not performed. Further investigation into the possibility of outliers can be done.

Training Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	4.27E-06	3.41E-06	4.08E-06
	0.5	2.32E-06	2.60E-06	2.97E-06
	0.8	**1.32E-06**	1.91E-06	2.43E-06
	min	1.32E-06		
Testing Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	5.00E-06	4.41E-06	4.80E-06
	0.5	3.30E-06	3.84E-06	4.03E-06
	0.8	**2.72E-06**	3.79E-06	4.05E-06
	min	2.72E-06		
	Minimum			
	Efficient			

Tables K3: MSE for Learning Rate Changes

K1.2.2 Conclusion – Learning Rate Increase/Decrease

Base on the results obtained the following decisions were made:

- A learning rate decrease of 0.8 was to be used, i.e. a 20% decrease in learning rate if the error of the system increases during training.
- A learning rate increase of 1.005 was to be used, i.e. a 0.5% increase in learning rate if the error of the system decreases during training.

K1.2.3 Observations – Input and Hidden Neurons – Training Set's MSE surface

The MSE surface from the training set of data stabilised when more than 4 hidden neurons were considered, as illustrated by the flattening of the MSE surface with limits (figure K2).

When 4 or more hidden neurons were considered the MSE minimised for any number of input neurons. This was unexpected because the MSE of the system should have decreased over the training set as the structure increased in complexity. The MSE did decrease as the structure became more complex, however, this reduction was of a lower magnitude than expected. The MSE was insensitive to the change of input neurons when considering 4 or more hidden neurons. When a single hidden neuron was considered, the MSE was larger.

The best estimate MSE surface, figure K3, indicated the minimum MSE (7.88E-08) was obtained using 36 input neurons per data set (144 input neurons in total) and 64 hidden neurons. The maximum MSE of 6.47E-06 was obtained using 36 input neurons per data set (144 in total) and a single hidden neurons. The maximum point was suspicious, since the largest MSE was expected to be obtained when a single input and hidden neuron was considered.

The upper and lower limit of the MSE surface over the training set were constructed by increasing/decreasing the average MSE by two standard deviations. Several assumptions are required for the limits to hold statistical significance. There was insufficient information available to make these assumptions. These limits, however, did provide a good estimate of the MSE limits for the randomly initialized structures.

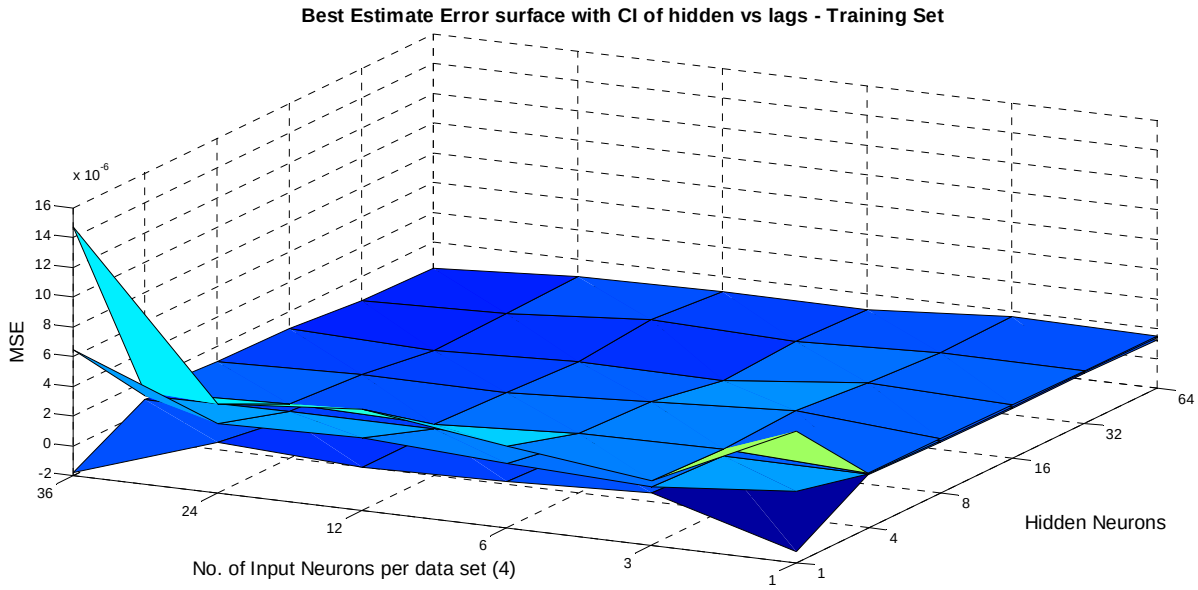

Figure K2: MSE with CI surface – Training data set

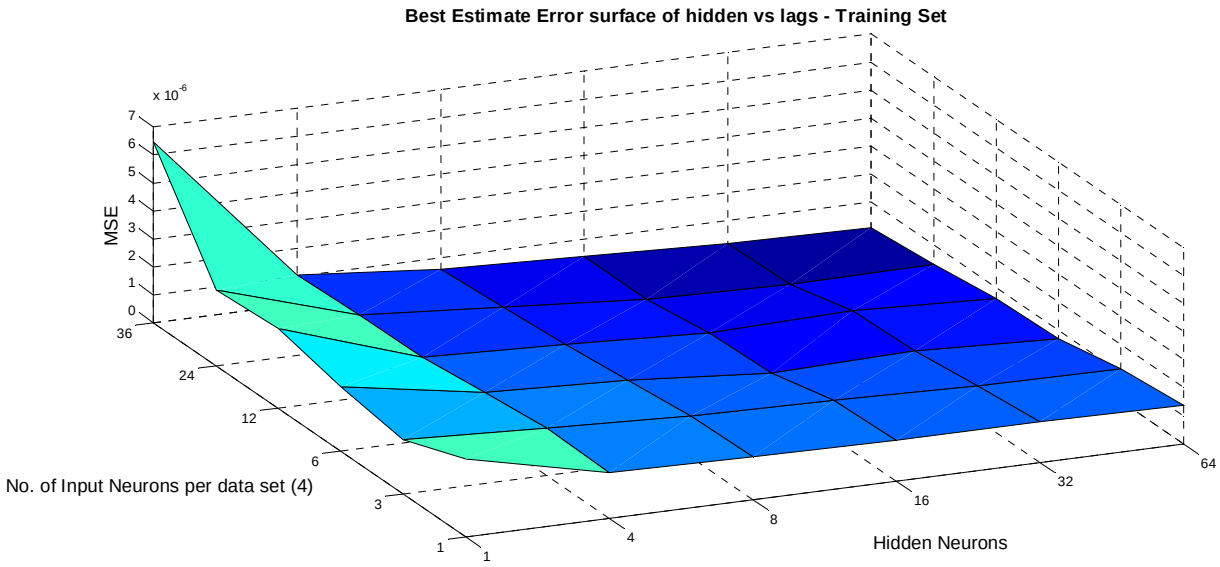

Figure K3: MSE surface – Training data set

K1.2.4 Observations – Input and Hidden Neurons – Testing Set's MSE surface

The MSE surface over the testing set with upper and lower limits, displayed in figure K4, minimized when between 1 and 12 input neurons per data set (4 – 48 input neurons in total) and more than 4 hidden neurons were considered. Figure K4 indicated over fitting was present in the system for large complex systems. Further, the volatility of the predictions increased as the system increased in complexity. This was due to the increased number of weights associated with complex systems and their limited number of initializations.

The best estimate MSE surface, displayed in figure K5, indicated the maximum MSE of 9.19E-06 was achieve using 36 input neurons per data set (144 input neurons in total) and 64 hidden neurons. The minimum MSE on the best estimate MSE surface of 7.78E-07 was obtained using a single input neuron per data set and 8 hidden neurons. This was unexpected as it suggests only a lagged period of a single month had an effect on the return on the money market. This may be explained by considering the nature of the money market. The money market has a low level of volatility and is unlikely to fluctuate significantly over a single month.

The upper and lower limits of the MSE surface over the testing set were constructed by increasing/decreasing the average MSE by two standard deviations. Several assumptions are required for the limits to hold statistical significance. There was insufficient information available to make these assumptions. These limits, however, did provide a good estimate of the MSE limits for the randomly initialized structures.

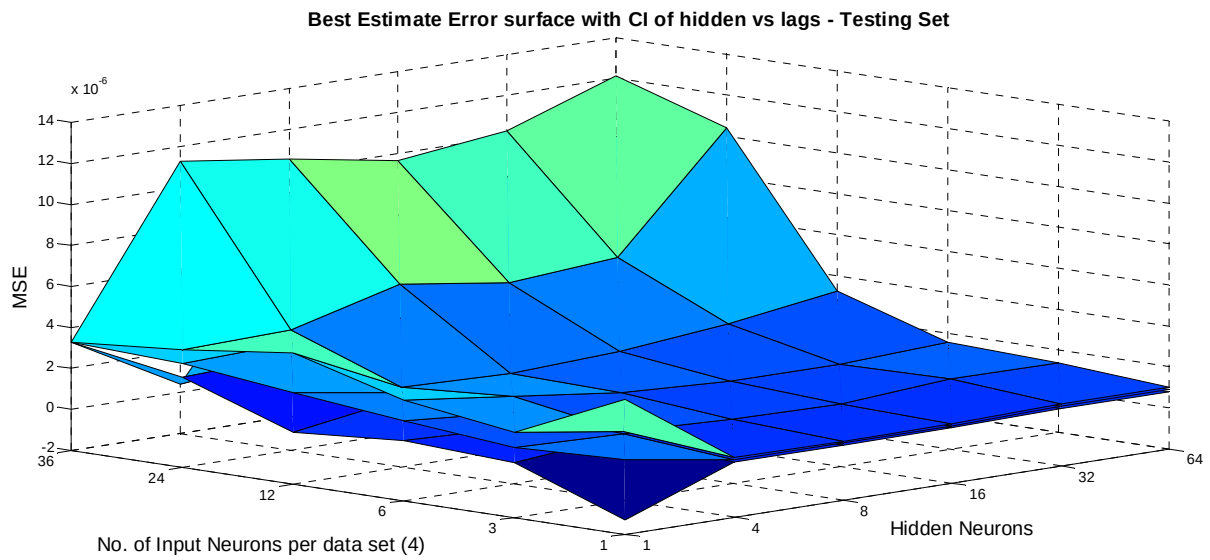

Figure K4: MSE with CI surface – Testing data set

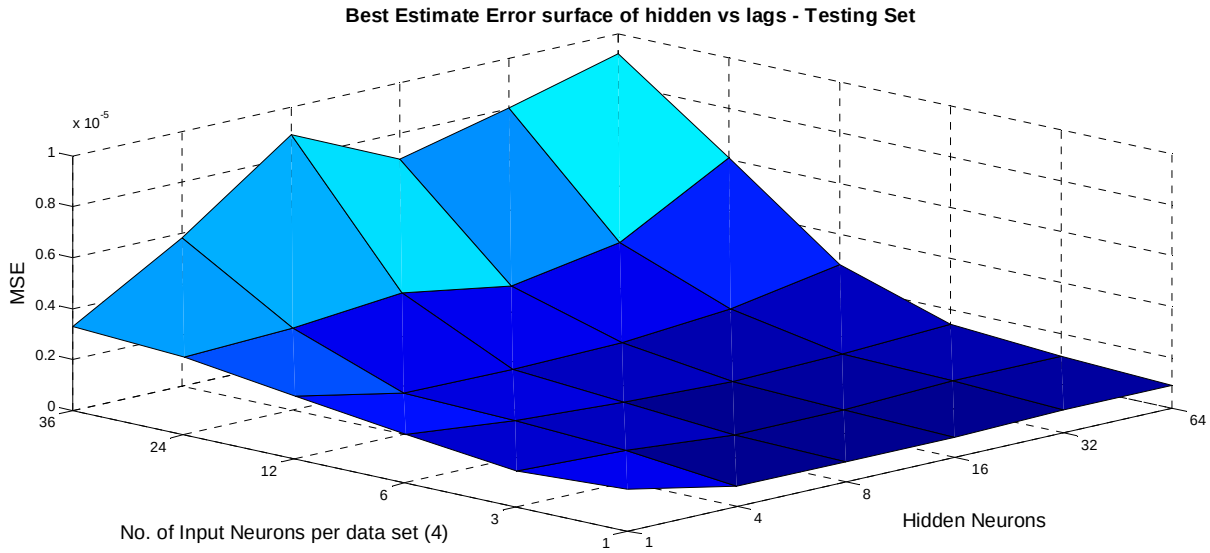

Figure K5: MSE surface – Testing data set

K1.2.5 Numerical Representation of MSE surfaces

The tables below provide the numerical average MSE values used to construct the error surfaces in figures K2, K3, K4 and K5.

Note: in the tables below, the Input Neurons refer to the number of input neurons per data set. Therefore, to determine the total number of input neurons this number was multiplied by 4.

Training Sets							
		LR increase		1.005			
		LR decrease		0.8	(20% decrease)		
	Average MSE vs Input and Hidden						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	2.78E-06	1.62E-06	1.52E-06	1.45E-06	1.45E-06	1.38E-06
	3	1.93E-06	1.67E-06	**1.46E-06**	1.36E-06	1.21E-06	1.14E-06
	6	2.30E-06	1.43E-06	1.21E-06	7.86E-07	9.21E-07	6.99E-07
	12	2.85E-06	1.16E-06	9.61E-07	6.98E-07	8.34E-07	6.06E-07
	24	2.70E-06	1.14E-06	7.64E-07	3.67E-07	2.48E-07	2.95E-07
	36	6.47E-06	1.04E-06	5.83E-07	3.75E-07	1.76E-07	7.88E-08
average	1.32E-06						
min	7.88E-08						
	Upper limit MSE vs Input and Hidden (+2 STD Dev)						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	6.85E-06	1.66E-06	1.64E-06	1.60E-06	1.53E-06	1.50E-06
	3	2.33E-06	2.13E-06	**2.37E-06**	1.86E-06	1.57E-06	1.66E-06
	6	3.49E-06	2.01E-06	1.79E-06	8.91E-07	1.17E-06	9.24E-07
	12	4.81E-06	1.43E-06	1.24E-06	9.92E-07	1.46E-06	9.89E-07
	24	3.98E-06	1.64E-06	1.29E-06	5.63E-07	6.88E-07	8.41E-07
	36	1.47E-05	1.29E-06	9.74E-07	8.58E-07	4.04E-07	2.07E-07
average	2.09E-06						
Efficient Structure							
Minimum Error							

Table K4: MSE of different combinations of input and hidden neurons – Training data set

Testing Sets							
		LR increase		1.005			
		LR decrease		0.8	(20% decrease)		
	Average MSE vs Input and Hidden						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	1.64E-06	8.05E-07	7.78E-07	7.90E-07	9.16E-07	8.96E-07
	3	1.42E-06	1.26E-06	**8.94E-07**	9.15E-07	1.12E-06	1.09E-06
	6	1.90E-06	1.48E-06	1.24E-06	1.09E-06	1.22E-06	1.42E-06
	12	2.47E-06	1.62E-06	1.60E-06	1.67E-06	2.05E-06	2.80E-06
	24	3.05E-06	3.24E-06	3.64E-06	2.96E-06	3.70E-06	6.06E-06
	36	3.30E-06	5.83E-06	8.90E-06	7.00E-06	8.05E-06	9.19E-06
average	2.72E-06						
min	7.78E-07	9.19E-06					
	Upper limit MSE vs Input and Hidden (+2 STD Dev)						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	4.57E-06	9.08E-07	8.63E-07	8.87E-07	1.03E-06	1.01E-06
	3	2.16E-06	1.36E-06	**1.12E-06**	1.04E-06	1.43E-06	1.40E-06
	6	2.89E-06	2.24E-06	1.61E-06	1.36E-06	1.34E-06	1.57E-06
	12	4.41E-06	1.90E-06	1.74E-06	1.99E-06	2.48E-06	3.25E-06
	24	3.72E-06	3.87E-06	5.27E-06	4.52E-06	4.92E-06	1.04E-05
	36	3.30E-06	1.13E-05	1.06E-05	9.64E-06	1.03E-05	1.21E-05
average	3.73E-06						
Efficient Structure							
Minimum Error							

Table K5: MSE of different combinations of input and hidden neurons – Testing data set

K1.2.6 Conclusion – Input and Hidden Neurons

To balance the error between the varying ANN structures over the training and testing data sets, a structure was chosen which limited the increase in MSE from the minimum in each data set. The structure chosen consisted of 3 input neurons per data set (12 in total) and 8 hidden neurons. The increase in MSE from the minimum over the training and testing data sets were 1.28E-06 and 1.16E-07, respectively

The choice made here had subjective components. It was important not to base the decision purely on the results over the training set as this would lead to over fitting. Similarly, basing the decision purely on the results from the testing set would lead to fitting the ANN to the testing set. Both scenarios described above would have reduced the generalization ability of the ANN. Thus, the subjective approach followed was most appropriate. Further, it was important to ensure a parsimonious model was selected.

K1.2.7 Conclusion of Determining an Efficient Structure

Efficient parameters were:

- 3 input neurons per data set (12 input neurons in total),
- 8 seasonal input neurons,
- 8 hidden neurons,
- A learning rate increase of 1.005 (0.5% increase), and
- A learning rate decrease of 0.8 (20% decrease).

K2 Analysing the Efficient ANN

The ANN was trained with the following parameters.

Network Structure (input neurons per data set x number of data sets) – hidden neurons – output neurons	12 (3x4)-8-1
Training set size	326
Testing set size	100
Dummy variables (for seasonality) (variables per data set x number of data sets)	8 (2x4)
Number of initialized structures	10
Epochs	5 000
Training algorithm	RPROP
Learning rate change: Increase Decrease Maximum Minimum	 0.5% (1.005) 20% (0.8) 100 0.000000001

Table K6: Parameters for the efficient ANN structure

K2.1 Observations of System Error

The error over the training and testing sets was stable after 1 000 training epochs as indicated by figure K6. The error over the training set continued to decrease as the number of epochs approached its maximum. This was expected as ANNs are known to over fit the data set. The error over the testing set reached a minimum after approximately 4 200 epochs.

The minimum RMSE over the training set was achieved after 5 000 epochs. The minimum RMSE over the training and testing sets was 9.037e-004 and 7.299e-004, respectively.

The efficient number of epoch chosen was 4 200 as it resulted in a RMSE near its minimum over both training and testing data sets. Further, the RMSE over both sets remained stable around this point.

Error Readings	Testing Set	Training Set
Minimum RMSE	7.2993e-004	9.0374e-004
Minimum MSE	5.3280e-007	8.1675e-007
Final RMSE after training *(4 200 epochs)*	7.3598e-004	9.0819e-004
Final MSE after training *(4 200 epochs)*	5.4167e-007	8.2481e-007

Table K7: Results of efficient model after training

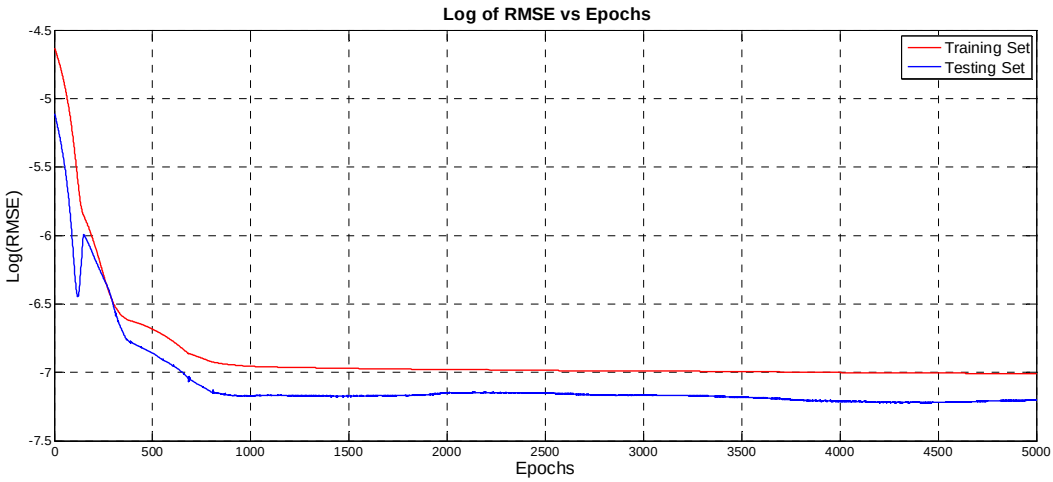

Figure K6: Log of RMSE after 5 000 training epochs

K2.2 Observations of Forecasts and Actuals

The ANN captured the underlying trends in the data set as expected. There were certain periods where the system underestimated the actual values, this could be explained by the effect of external influences on the money market. The best estimate over the training and testing sets was represented by the blue line in figure K7.

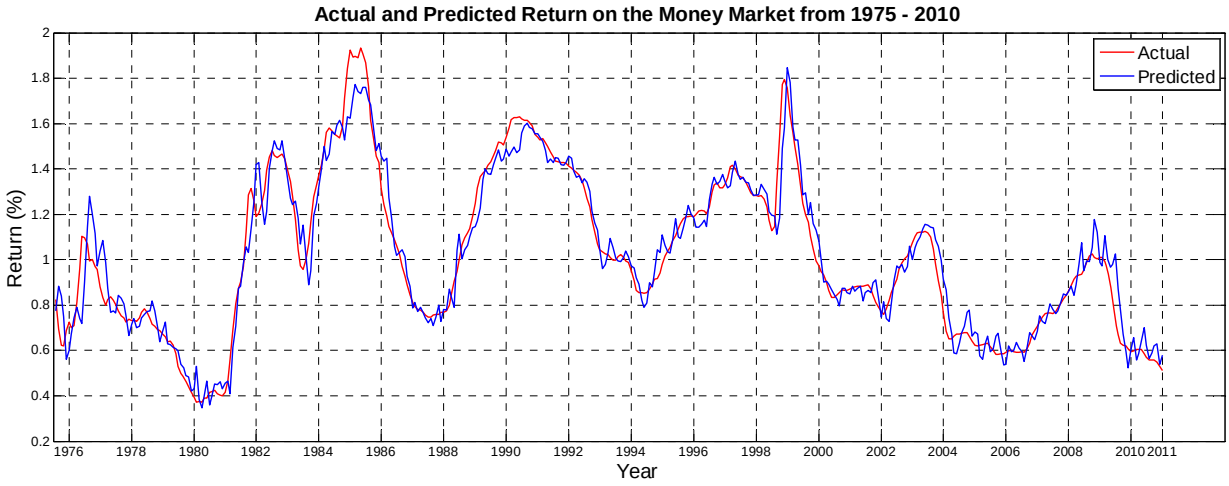

Figure K7: Actual and Predicted – Training and Testing Data Sets

K2.3 Conclusion - Analysing the Efficient ANN

The ANN captured the trends underlying the data. The efficient ANN structure had an MSE over the training and testing sets of 8.2481e-007 and 5.4167e-007, respectively. The ANN performed similarly over the testing and training data set. The level of effectiveness was determined by comparing these results to traditional models in a following chapter (Chapter 5). The minimum RMSE over the testing set, of 7.2993e-004, and over the training set, of 9.0374e-004, was obtained after approximately 4 200 and 5 000 epochs, respectively.

The RMSE over the testing set did not show signs of a significant increase as the training approached 5 000 epochs. This was not expected, as it indicated that the system was not susceptible to over fitting.

K3 Summary of Experiment

The following table provides a summary of the results from this experiment.

Network Structure (input neurons per data set x number of data sets) – hidden neurons – output neurons	12 (3x4)-16-1
Training epoch required for optimal training	5 000
Minimum Root Mean Squared Error: Training Data Set Testing Data Set	 9.0374e-004 7.2993e-004
Final Root Mean Squared Error (4 200 epochs): Training Data Set Testing Data Set	 9.0819e-004 7.3598e-004
Parameters for RPROP: Learning Rate Increase Learning Rate Decrease	 0.5% (1.005) 20% (0.8)
Data Sets: Training set Testing set	 326 observations (1975 – 2002) 100 observations (2002 – 2010)
Times the structure was initialized	10

Table K8: Summary of experiment's result

Appendix L – Integrated Bond Market ANN – Three-Month Forecast Results

L1 Determining an Efficient Structure

L1.1 Part 1 – Efficient number of training epochs

The parameters used for this experiment were:

Parameter	Value
Network Structure (input neurons per data set × number of data sets) – hidden neurons – output neurons	144 (36x4)-64-1
Training set size	293
Testing set size	100
Dummy variables (for seasonality) (variables per data set x number of data sets)	8 (2x4)
Number of initialized structures	5
Epochs	1 000
Training algorithm	RPROP
Learning rate change: Increase Decrease Maximum Minimum	 0.5% (1.005) 20% (0.8) 100 0.000000001

Tables L1: Parameters for experiment - Part 1

L1.1.1 Observations – Part 1

The minimum RMSE (0.0217) over the testing set was achieved within the first 1 000 epochs. This was supported by figure L1 which plotted the error over each data set for each epoch. The minimum error was represented by the minimum point of the trough created by the testing set error (blue line) after approximately 330 epochs. The RMSE over the testing set increased after this point which indicated over fitting in the ANN. The minimum RMSE (0.0032) over the training set was achieved at the 1 000 epoch mark, as expected. The RMSE at important points were tabulated in table L2.

The RMSE of 0.0217 over the testing set was compared to other applications. The RMSE was significantly larger than other applications which indicated historically there were limited relationships between the successive return on the bond market. This was expected as return on the bond market is largely driven by demand and supply factors that do not depend heavily on past returns.

		Training Epochs				
	Min RMSE	**100**	**500**	**1 000**	**Δ in RMSE – 100 to 500**	**Δ in RMSE – 500 to 1 000**
RMSE – Testing Set	0.0217	0.1371	0.0285	0.0351	0.1086	-0.0066
RMSE – Training Set	0.0032	0.1353	0.0095	0.0032	0.1258	0.0063

Tables L2: RMSE after different numbers of epochs

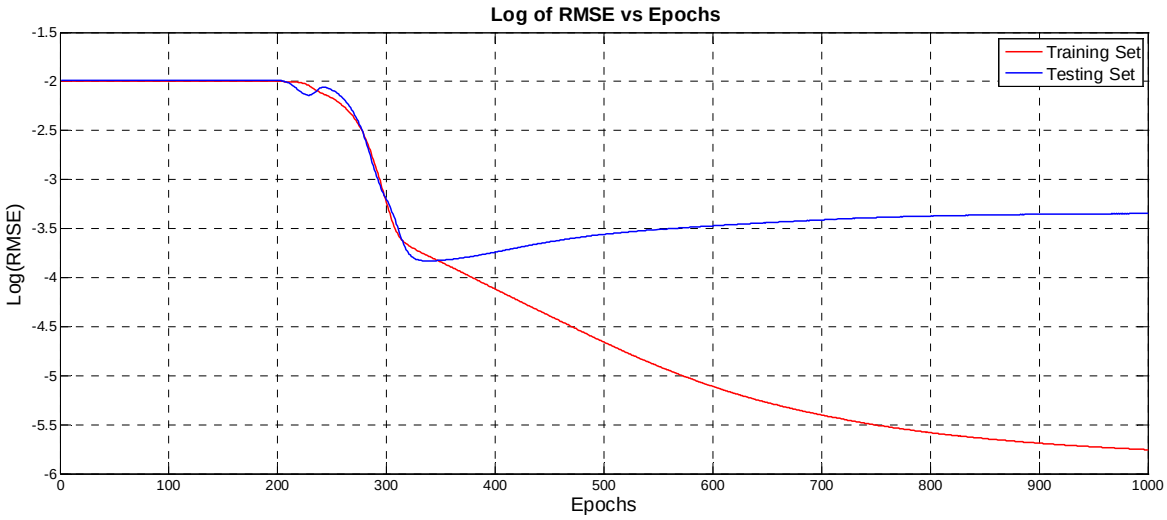

Figure L1: Log of RMSE over training process

L1.1.2 Conclusion – Part 1:

From the results obtained, the number of epochs used to compare different structures was 1000. This was chosen because it was important to allow the system to reach an over fitted stage, which implied the minimum error over the testing set of data was surpassed and captured. The points in table L2 indicated the RMSE over the testing set increased from 500 to 1 000 epochs. The minimum RMSE of 0.0217 was achieved between 100 and 1 000 epochs. By considering 1 000 epochs the minimum RMSE over the testing set would be captured. This is supported by figure L1.

The ANN considered in this experiment was the most complex and had the slowest learning parameters. Other simpler ANNs were expected to reach their minimum errors within the 1 000 epochs as the learning progressed faster. The conclusion of an efficient number of learning epochs was primarily based on the results obtained over the testing set.

Considering only the error over the training set in the decision would lead to the selection of the most complex ANN. This would increase the risk of over fitting associated with overly complex ANNs.

L1.2 Part 2 – Determining efficient parameters

L1.2.1 Observations – Learning Rate Increase/Decrease

The minimum MSE over the training set (3.81E-04) was achieved using a learning rate increase of 1.05 (5% increase) and a decrease of 0.8 (20% decrease). The minimum MSE over the testing set (3.74E-04) was achieved using an increase of 1.005 (0.5% increase) and a decrease of 0.5 (50% decrease). These observations did not support a similar conclusion and are tabulated in table L3.

The shape of the MSE surface over the training set was considered. The range of the MSE surface over the training set was significant (2.14E-04) in comparison to that of the testing set (0.21E-04). The efficient learning rate increase/decrease was chosen by determining the smallest trade off in error over both sets of data.

The efficient learning rate increase and decrease chosen were 1.05 and 0.8, respectively. They resulted in a fair trade off between the accuracy over the associated data sets. The increase in average MSE over the training and testing sets were 0 and 0.18E-4, respectively. The MSE associated with the chosen learning rate increase and decrease are marked in orange in table L3 and the minimum MSEs are marked in green.

It is important to note the values used to decide on the learning rate changes were the arithmetic average MSE over different combinations of input and hidden neurons. This average did not explicitly allow for the event of outliers in the data and may be skewed. From the investigations there were no outliers found but extensive tests to verify this remark were not performed. Further investigation into the possibility of outliers can be done.

Training Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	5.95E-04	5.02E-04	5.15E-04
	0.5	5.05E-04	4.49E-04	4.54E-04
	0.8	4.23E-04	**3.81E-04**	4.10E-04
	min	3.81E-04		
Testing Set				
		LR Increase		
LR decrease		1.005	1.05	1.1
	0.2	3.75E-04	3.82E-04	3.89E-04
	0.5	3.74E-04	3.79E-04	3.85E-04
	0.8	3.78E-04	**3.92E-04**	3.95E-04
	min	3.74E-04		
	Minimum			
	Efficient			

Tables L3: MSE for Learning Rate Changes

4.8.1.2.2 Conclusion – Learning Rate Increase/Decrease

The following decisions were made:

- A learning rate decrease of 0.8 was to be used, i.e. a 20% decrease in learning rate if the error of the system increases during training.
- A learning rate increase of 1.05 was to be used, i.e. a 5% increase in learning rate if the error of the system decreases during training.

L1.2.3 Observations – Input and Hidden Neurons – Training Set's MSE surface

The MSE surface with limits, figure L2, indicated the MSE of the system decreased as the number of input and hidden neurons increased. This was expected since the increased complexity of the system would increase the probability of over fitting. Over fitting would lead to increased accuracy over the training data set. The upper and lower surface represents the upper and lower limits of the MSE. The limits were determined by increasing/decreasing the best estimate surface by two standard deviations of the MSE.

Considering the best estimate MSE surface, figure L3, the MSE decreased as the complexity of the system increased. The surface obtained a minimum of 1.64E-05 at 24 input neurons per data set (95 input neurons in total) and 64 hidden neurons. The minimum was expected at 36 input neurons per data set and 64 hidden neurons. This anomaly was due to random variation in the initialized structure.

The upper and lower limit of the MSE surface over the training set requires assumptions in order to hold statistical significance. There was insufficient information available to make these assumptions. These limits, however, did provide a good estimate of the MSE limits for the randomly initialized structures.

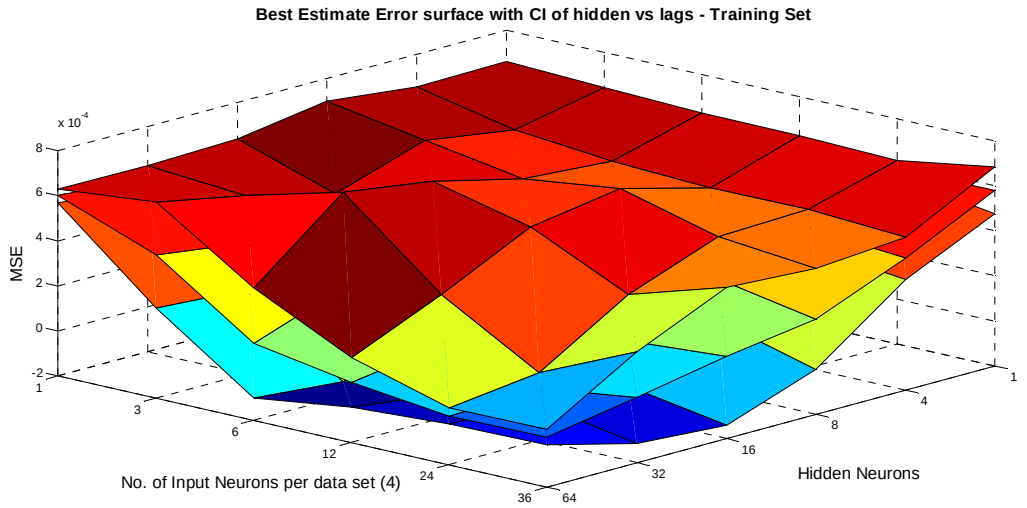

Figure L2: MSE with CI surface – Training data set

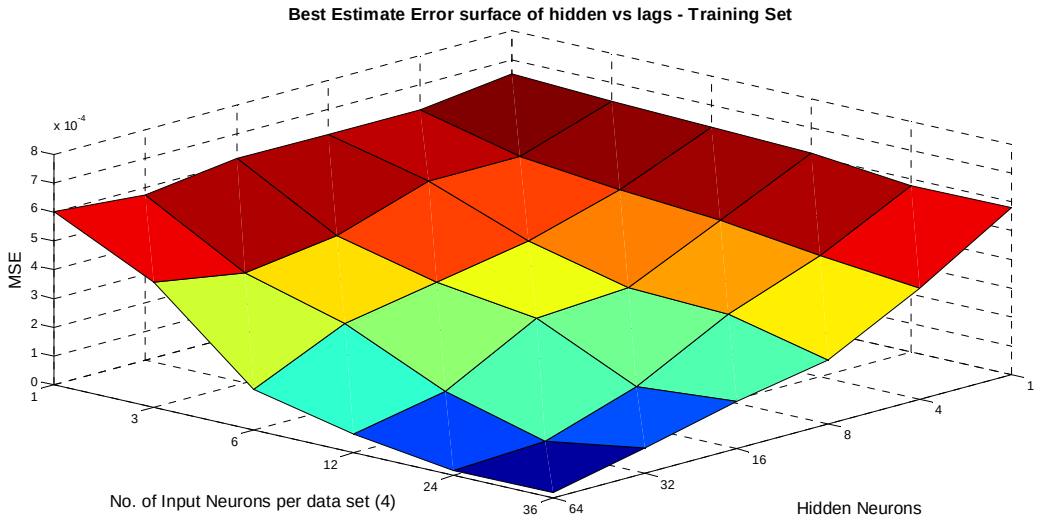

Figure L3: MSE surface – Training data set

L1.2.4 Observations – Input and Hidden Neurons – Testing Set's MSE surface

The MSE surface over the testing set with upper and lower limits, figure L4, indicated minimums when either a large number of input or hidden neurons were considered. Considering a large number of both input and hidden neurons resulted in a large MSE. The volatility of the surface increased as the structure became more complex. This was indicated by the increasingly large limits observed as the system increased in complexity.

The best estimate MSE surface, figure L5, indicated one ridge located when more than 12 input and 16 hidden neurons were considered. Minimums were located when either a large number of input or hidden neurons were present. The minimum MSE on the best estimate MSE surface, of 3.55E-04, was achieved using 36 input and a single hidden neurons. The MSE surface over the testing set was structurally different to that of the training set.

The upper and lower limit of the MSE surface over the testing set requires assumptions in order to hold statistical significance. There was insufficient information available to make these assumptions. These limits, however, did provide a good estimate of the MSE limits for the randomly initialized structures.

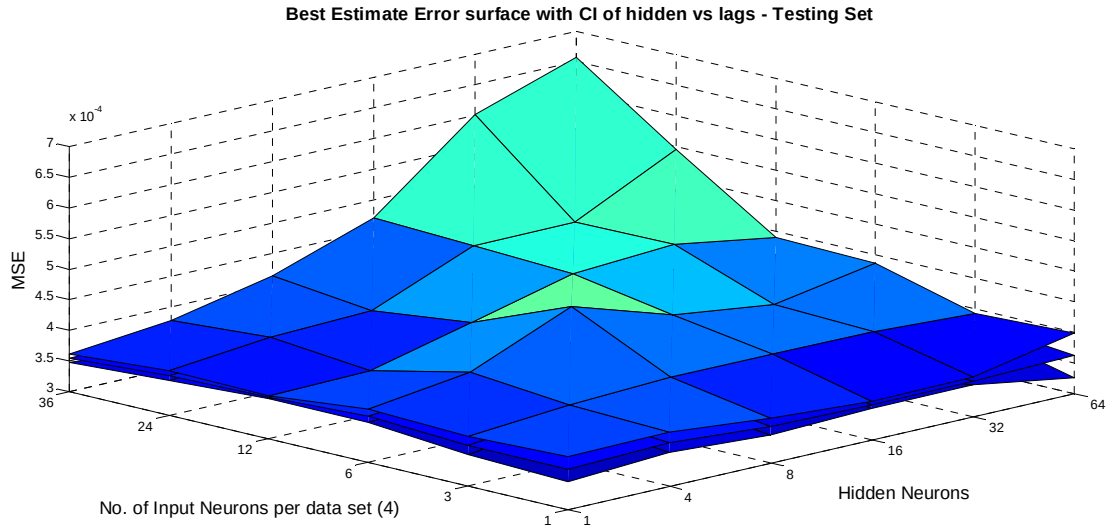

Figure L4: MSE with CI surface – Testing data set

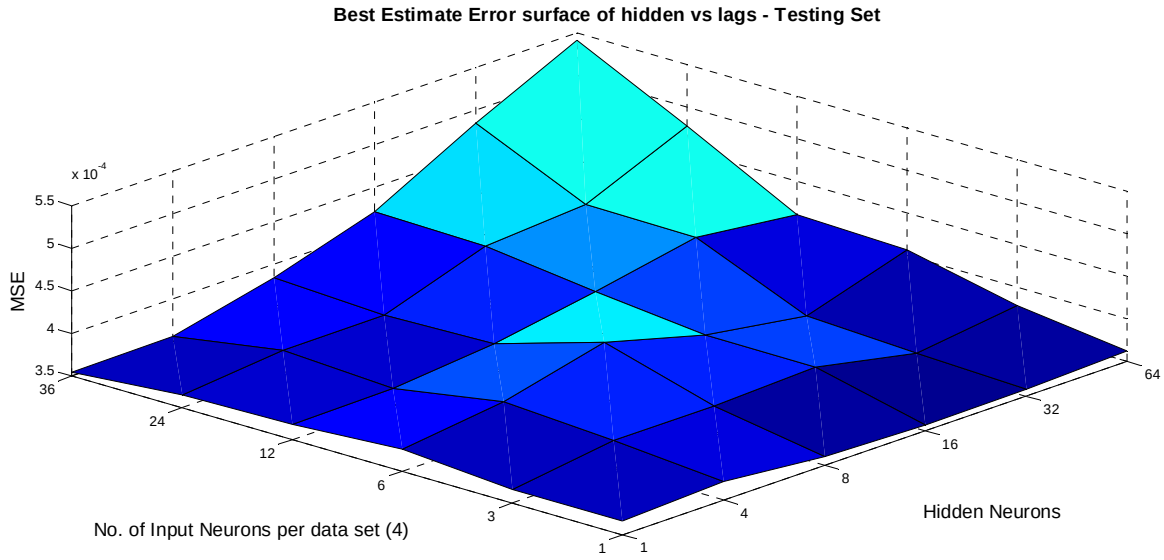

Figure L5: MSE surface – Testing data set

L1.2.5 Numerical Representation of MSE surfaces

The tables below provide the numerical average MSE values used to construct the error surfaces in figures L2, L3, L4 and L5.

Note: in the tables below, the Input Neurons refer to the number of input neurons per data set. Therefore, to determine the total number of input neurons this number was multiplied by 4.

Training Sets							
		LR increase		1.05			
		LR decrease		0.8	(20% decrease)		
	Average MSE vs Input and Hidden						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	6.52E-04	6.12E-04	6.13E-04	6.16E-04	5.73E-04	6.01E-04
	3	6.36E-04	5.30E-04	5.28E-04	4.26E-04	3.82E-04	4.36E-04
	6	6.23E-04	4.92E-04	4.00E-04	3.43E-04	2.84E-04	1.43E-04
	12	6.13E-04	4.64E-04	**3.17E-04**	2.95E-04	1.27E-04	6.66E-05
	24	5.78E-04	4.20E-04	3.04E-04	1.37E-04	3.31E-05	1.64E-05
	36	5.80E-04	3.85E-04	2.23E-04	1.65E-04	8.85E-05	2.06E-05
average	3.81E-04						
min	1.64E-05						
	Upper limit MSE vs Input and Hidden (+2 STD Dev)						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	6.59E-04	6.54E-04	6.98E-04	6.39E-04	6.25E-04	6.31E-04
	3	6.42E-04	5.63E-04	6.24E-04	4.93E-04	5.91E-04	6.71E-04
	6	6.29E-04	5.21E-04	5.51E-04	6.48E-04	7.05E-04	3.88E-04
	12	6.25E-04	5.03E-04	**6.06E-04**	5.43E-04	3.51E-04	1.77E-04
	24	6.13E-04	5.03E-04	4.90E-04	3.41E-04	1.03E-04	5.36E-05
	36	6.84E-04	4.78E-04	4.48E-04	4.72E-04	2.91E-04	5.48E-05
average	5.07E-04						
Efficient Structure							
Minimum Error							

Table L4: MSE of different combinations of input and hidden neurons – Training data set

Testing Sets							
		LR increase		1.05			
		LR decrease		0.8	(20% decrease)		
	Average MSE vs Input and Hidden						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	3.66E-04	3.72E-04	3.60E-04	3.57E-04	3.59E-04	3.62E-04
	3	3.66E-04	3.83E-04	3.83E-04	3.88E-04	3.63E-04	3.79E-04
	6	3.76E-04	3.91E-04	4.20E-04	3.88E-04	3.70E-04	4.07E-04
	12	3.68E-04	3.69E-04	**3.81E-04**	4.02E-04	4.25E-04	4.11E-04
	24	3.65E-04	3.77E-04	3.77E-04	4.18E-04	4.26E-04	4.78E-04
	36	3.55E-04	3.56E-04	3.84E-04	4.21E-04	4.84E-04	5.41E-04
average	3.92E-04						
min	3.55E-04						
	Upper limit MSE vs Input and Hidden (+2 STD Dev)						
	Hidden Neurons						
Input Neurons		1	4	8	16	32	64
	1	3.86E-04	3.88E-04	3.74E-04	3.62E-04	3.65E-04	3.99E-04
	3	3.81E-04	3.94E-04	4.02E-04	4.02E-04	4.01E-04	3.93E-04
	6	3.86E-04	4.10E-04	4.78E-04	4.27E-04	4.07E-04	4.36E-04
	12	3.70E-04	3.74E-04	**4.14E-04**	4.57E-04	4.66E-04	4.40E-04
	24	3.73E-04	3.89E-04	3.95E-04	4.64E-04	4.64E-04	5.45E-04
	36	3.62E-04	3.78E-04	4.14E-04	4.70E-04	6.01E-04	6.57E-04
average	4.23E-04						
Efficient Structure							
Minimum Error							

Table L5: MSE of different combinations of input and hidden neurons – Testing data set

L1.2.6 Conclusion – Input and Hidden Neurons

In order to balance the error between the structures over the training and testing data sets, a structure was chosen which minimized the reduction of accuracy. The structure chosen had 12 input neurons per data set (48 input neurons in total) and 8 hidden neurons. The increase in MSE from the minimum over the training and testing data sets was 3.01E-04 and 0.26E-04, respectively

The choice made here had subjective components. It was important not to base the decision purely on the results over the training set as this would lead to over fitting. Similarly, basing the decision purely on the results over the testing set would lead to the fitting of the ANN to the testing set. Both the scenarios described above would have reduced the generalization ability of the ANN.

L1.2.7 Conclusion of Determining an Efficient Structure

Efficient parameters were:

- 12 input neurons per data set (48 input neurons in total),
- 8 seasonal input neurons,
- 8 hidden neurons,
- A learning rate increase of 1.05 (5% increase), and
- A learning rate decrease of 0.8 (20% decrease).

L2 Analysing the Efficient ANN

The ANN was trained using the following parameters.

Network Structure (input neurons per data set x number of data sets) – hidden neurons – output neurons	48 (12x4)-8-1
Training set size	317
Testing set size	100
Dummy variables (for seasonality) (variables per data set x number of data sets)	8 (2x4)
Number of initialized structures	10
Epochs	5 000
Training algorithm	RPROP
Learning rate change: Increase Decrease Maximum Minimum	 5% (1.05) 20% (0.8) 100 0.000000001

Table L6: Parameters for the efficient ANN structure

L2.1 Observations of System Error

After approximately 50 training epochs the ANN began to over fit the data. This was supported by the rising of the error over the testing data set in the plot of error against learning epochs, figure L6. The minimum RMSE (0.0188) over the testing set was obtained after approximately 50 training epochs. The error over the testing and training set intersected after approximately 500 training epochs. The point where the MSE achieved a minimum over the testing set was chosen as the efficient number of training epochs for this ANN structure, 50 epochs in this case. Over fitting was completely avoided using this point. Figure L6 illustrates the log of the Root Mean Squared Error (Log (RMSE)) of the system over 5 000 training epochs. The numerical error value at points of interest are provided in table L7.

Error Readings	**Testing Set**	**Training Set**
Minimum RMSE	0.0188	0.0156
Minimum MSE	3.5231e-004	2.4438e-004
Final RMSE after training *(50 epochs)*	0.0189	0.0248
Final MSE after training *(50 epochs)*	3.5739e-004	6.1512e-004

Table L7: Results of efficient model after training

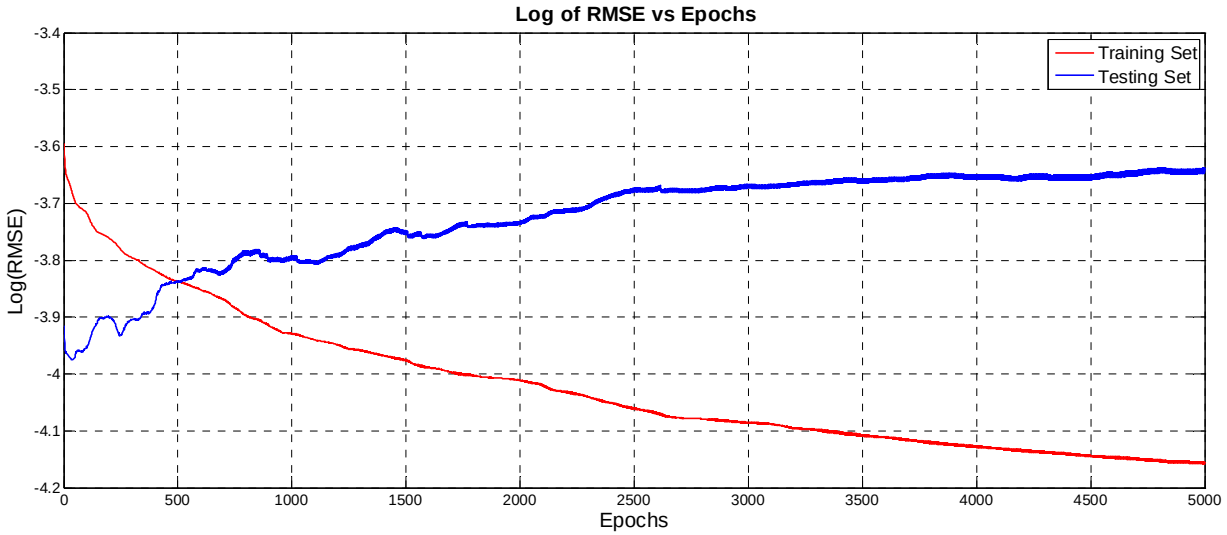

Figure L6: Log of RMSE over 5 000 training epochs

L2.2 Observations of Forecasts and Actuals

The ANN captured few underlying trends in the data set as expected. There were several occasions were the ANN either under or overestimated the return on the bond market. This was true for both the training and testing data sets. These instances were the result of external factors on the market and could not be explained using only past observations of the market. The best estimate over the training and testing sets are represented by the blue line in figure L7.

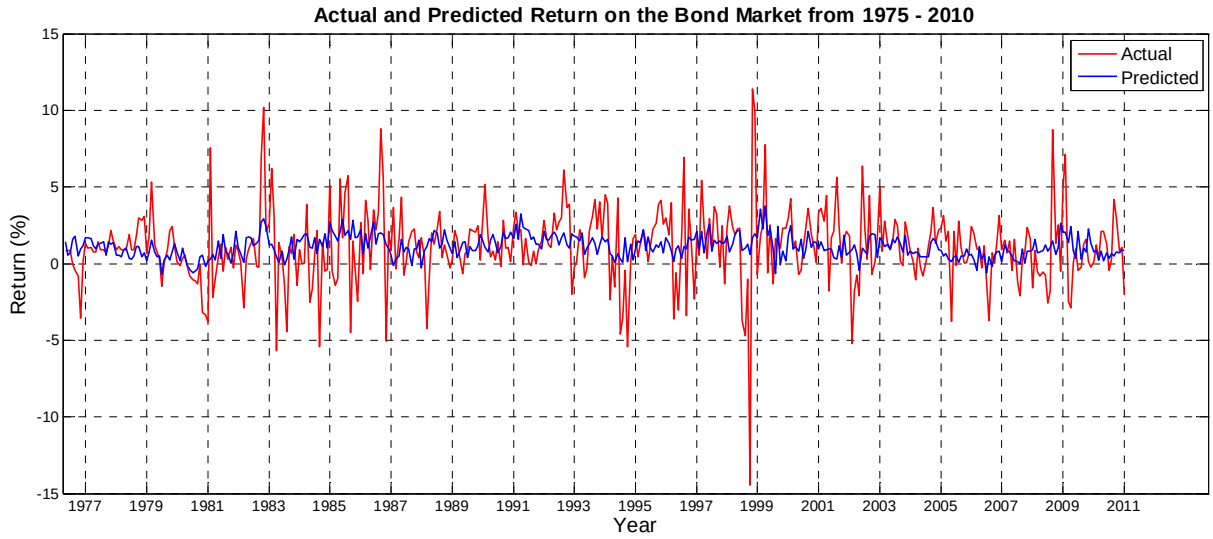

Figure L7: Actual and Predicted – Training and Testing Data Sets

L2.3 Conclusion - Analysing the Efficient ANN

The ANN captured few trends underlying the data. The efficient ANN structure had an MSE over the training and testing sets of 6.1512e-004 and 3.5739e-004, respectively. The ANN performed similarly over the testing and training data set. The level of effectiveness was determined by comparing these results to traditional models in a following chapter (Chapter 5). The minimum RMSE over the testing set, of 0.0188, and over the training set, of 0.0156, was obtained after approximately 50 and 5 000 epochs, respectively. The RMSE of the training set decreased as the training went over 50 epochs and continued to decrease until the maximum number of epochs was obtained. The efficient number of epochs used was 50 which resulted in a RMSE of 0.0189 over the testing set and 0.0248 over the training set. To avoid over fitting but to ensure the underlying relationships in the data were captured, decisions relating to efficiency were made with reference to the performance of the ANN over both the testing and training data set.

L3 Summary of Experiment

The following table provides a summary of the results from this experiment.

Network Structure (input neurons per data set x number of data sets) – hidden neurons – output neurons	48 (12x4)-8-1
Training epoch required for optimal training	50
Minimum Root Mean Squared Error: Training Data Set Testing Data Set	 0.0156 0.0188
Final Root Mean Squared Error (50 epochs): Training Data Set Testing Data Set	 0.0248 0.0189
Parameters for RPROP: Learning Rate Increase Learning Rate Decrease	 5% (1.05) 20% (0.8)
Data Sets: Training set Testing set	 317 observations (1976 – 2002) 100 observations (2002 – 2010)
Times the structure was initialized	10

Table L7: Summary of experiment's result

Appendix M – Hybrid Model construction

M1 Three-Month Forecast Period – Inflation

First the hybrid model with an isolated ANN is considered, followed by the hybrid model with an integrated ANN. In this case a three-month forecast period is considered.

M1.1 Hybrid with Isolated ANN

The ANN parameters used in the hybrid model are given in table M1.

Network Structure	3-4-1
Training epoch required for optimal training	170
Parameters for RPROP:	
Learning Rate Increase	5% (1.005)
Learning Rate Decrease	20% (0.8)
Times the structure is initialised	25

Table M1: Inflation hybrid model with isolated ANN – three-month forecasts

The results of the hybrid model after the ANN is efficiently trained are tabulated in table M2.

Data Set	RMSE
Training	0.005316
Testing	0.004807

Table M2: RMSEs– inflation hybrid (isolated ANN, three steps)

The three step forecasts of inflation, from 1975 to 2010, generated by the hybrid model with an isolated ANN are plotted in the line graph below, figure M1.

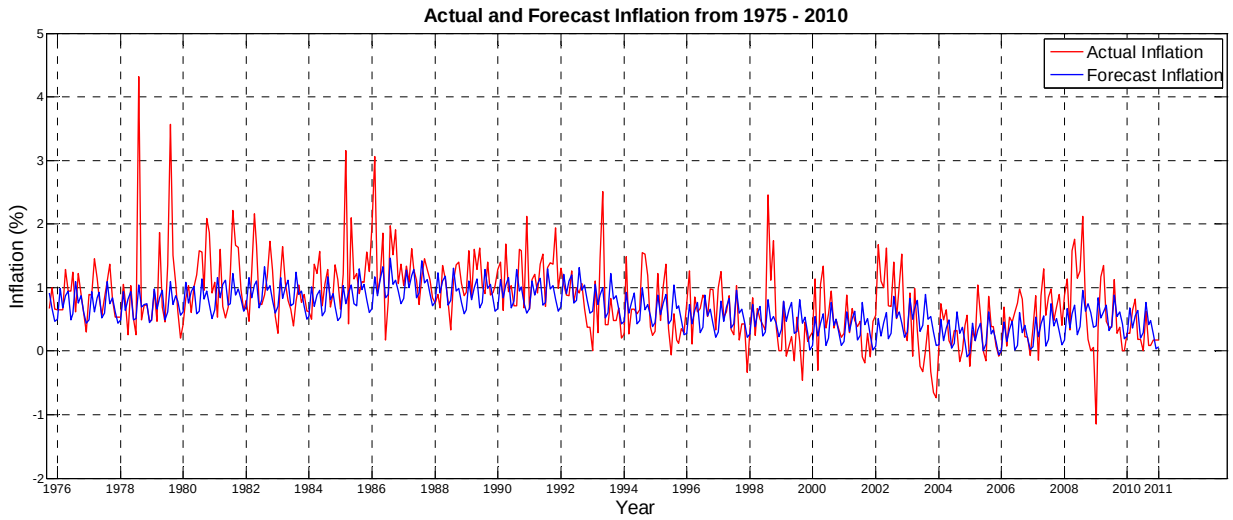

Figure M1: Actual and forecast inflation generated by isolated hybrid model – three steps

M1.2 Hybrid with Integrated ANN

The ANN parameters used in the hybrid model are given in table M3.

Network Structure	12(3x4)-4-1
Training epoch required for optimal training	100
Parameters for RPROP: Learning Rate Increase Learning Rate Decrease	 0.5% (1.005) 20% (0.8)
Times the structure is initialised	15

Table M3: Inflation hybrid model with integrated ANN – three-month forecasts

The results of the hybrid model after the ANN is efficiently trained are tabulated in table M4.

Data Set	RMSE
Training	0.005194
Testing	0.004916

Table M4: RMSEs– inflation hybrid (integrated ANN, three steps)

The three step forecasts of inflation, from 1975 to 2010, generated by the hybrid model with an integrated ANN are plotted in the line graph below, figure M2.

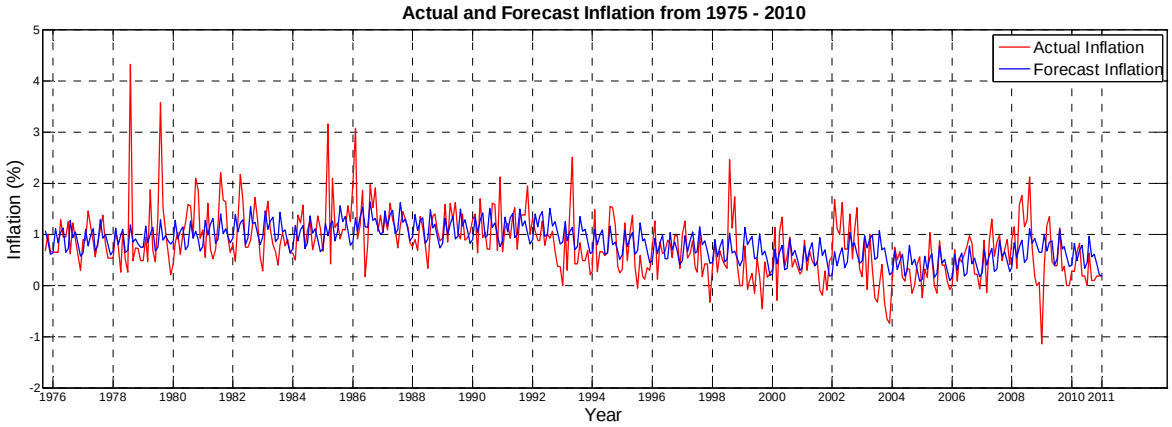

Figure M2: Actual and forecast inflation generated by integrated hybrid model – three steps

M2 Money Market Hybrid Model

The hybrid models, used to forecast return on the money market, are constructed with an isolated or integrated ANN. Further, forecasts are generated for a one and three-month forecast period.

M2.1 One Month Forecast Period – Money Market

First the hybrid model with an isolated ANN is considered, followed by the hybrid model with an integrated ANN. In this case a one month forecast period is considered.

M2.1.1 Hybrid with Isolated ANN

The ANN parameters used in the hybrid model are given in table M5.

Network Structure	3-4-1
Training epoch required for optimal training	200
Parameters for RPROP:	
Learning Rate Increase	0.5% (1.005)
Learning Rate Decrease	20% (0.8)
Times the structure is initialised	25

Table M5: Money market hybrid model with isolated ANN – one month forecasts

The results of the hybrid model after the ANN is efficiently trained are presented in table M6.

Data Set	RMSE
Training	0.000301
Testing	0.000117

Table M6: RMSEs – money market hybrid (isolated ANN, one step)

The error over the training set is greater than over the testing set. This is similar to the results from the traditional models and ANNs constructed in chapter 3 and chapter 4, respectively.

The one step forecasts of the money market, from 1975 to 2010, generated by the hybrid model with an isolated ANN are plotted in the line graph below, figure M3.

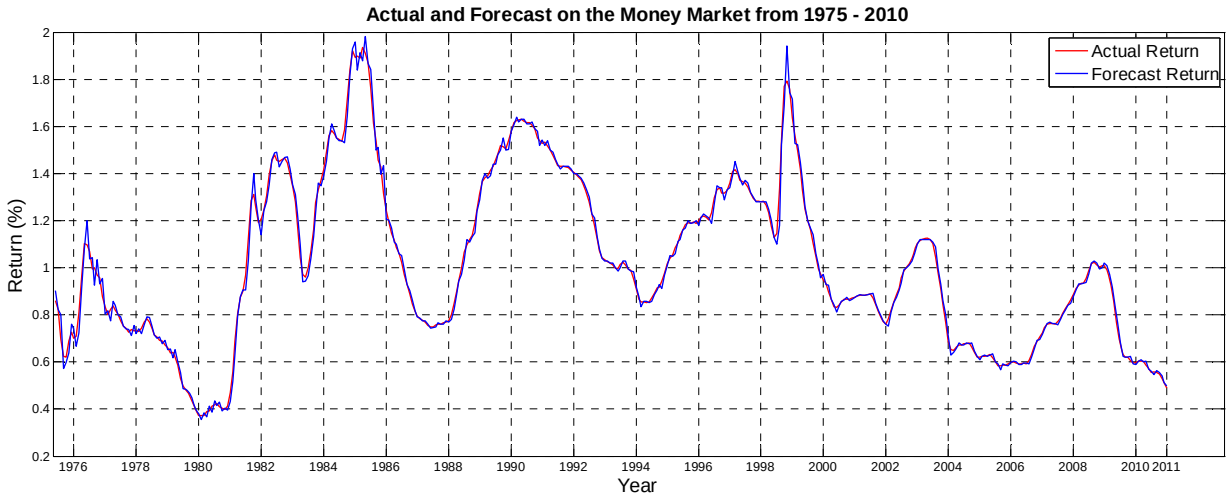

Figure M3: Actual and forecast return on the money market generated by isolated hybrid model – one step

M2.1.2 Hybrid with Integrated ANN

The ANN parameters used in the hybrid model are given in table M7.

Network Structure	12(3x4)-16-1
Training epoch required for optimal training	100
Parameters for RPROP:	
Learning Rate Increase	0.5% (1.005)
Learning Rate Decrease	20% (0.8)
Times the structure is initialised	10

Table M7: Money market hybrid model with integrated ANN – one month forecasts

The results of the hybrid model after the ANN is efficiently trained are presented in table M8.

Data Set	RMSE
Training	0.000313
Testing	0.000116

Table M8: money market hybrid (integrated ANN, one step)

The one step forecasts of the return on the money market, from 1975 to 2010, generated by the hybrid model with an integrated ANN are plotted in the line graph below, figure M4.

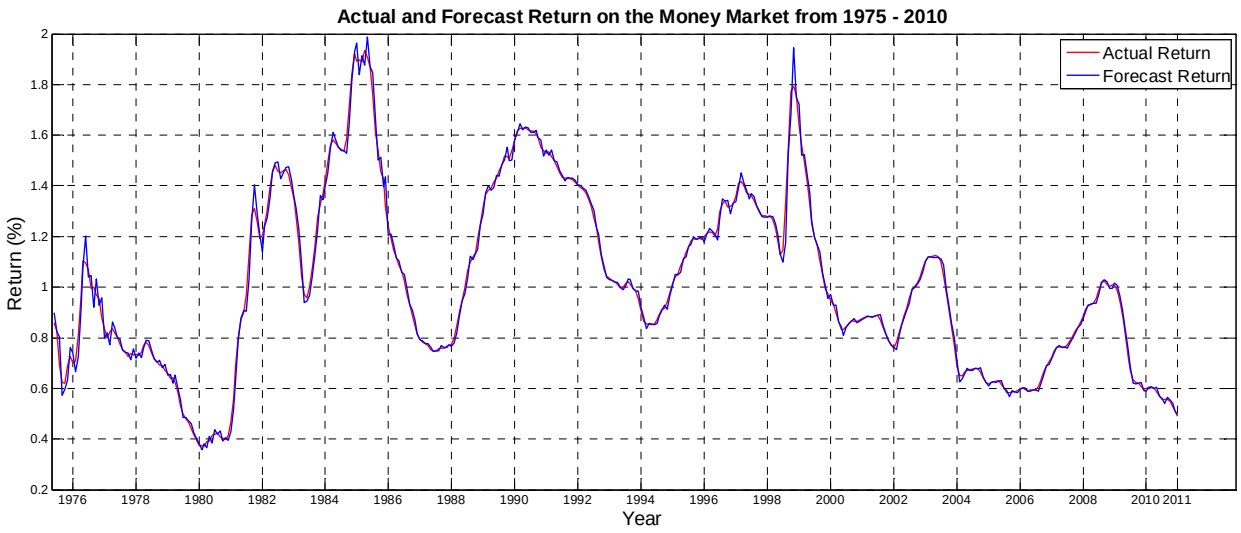

Figure M4: Actual and forecast return on the money market generated by integrated hybrid model – one step

M2.2 Three-Month Forecast Period – Money Market

First the hybrid model with an isolated ANN is considered, followed by the hybrid model with an integrated ANN. In this case a three-month forecast period is considered.

M2.2.1 Hybrid with Isolated ANN

The ANN parameters used in the hybrid model are given in table M9.

Network Structure	3-4-1
Training epoch required for optimal training	200
Parameters for RPROP:	
Learning Rate Increase	5% (1.005)
Learning Rate Decrease	20% (0.8)
Times the structure is initialised	25

Table M9: Money market hybrid model with isolated ANN – three-month forecasts

The results of the hybrid model after the ANN is efficiently trained are presented in table M10.

Data Set	RMSE
Training	0.001077
Testing	0.000523

Table M10: money market hybrid (isolated ANN, three steps)

The three step forecasts of the money market, from 1975 to 2010, generated by the hybrid model with an isolated ANN are plotted in the line graph below, figure M5.

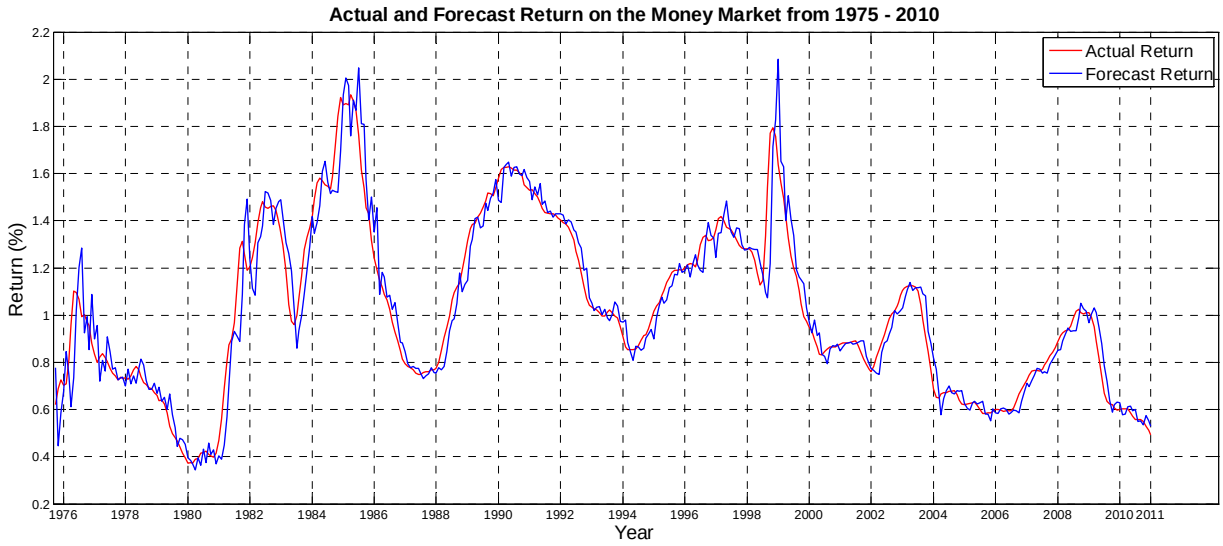

Figure M5: Actual and forecast return on the money market generated by isolated hybrid model – three steps

M2.2.2 Hybrid with Integrated ANN

The ANN parameters used in the hybrid model are given in table M11.

Network Structure	12(3x4)-4-1
Training epoch required for optimal training	150
Parameters for RPROP:	
Learning Rate Increase	0.5% (1.005)
Learning Rate Decrease	20% (0.8)
Times the structure is initialised	10

Table M11: Money market hybrid model with integrated ANN – three-month forecasts

The results of the hybrid model after the ANN is efficiently trained are tabulated in table M12.

Data Set	RMSE
Training	0.001146
Testing	0.000516

Table M12: money market hybrid (integrated ANN, three steps)

The three step forecasts of the money market, from 1975 to 2010, generated by the hybrid model with an integrated ANN are plotted in the line graph below, figure M6.

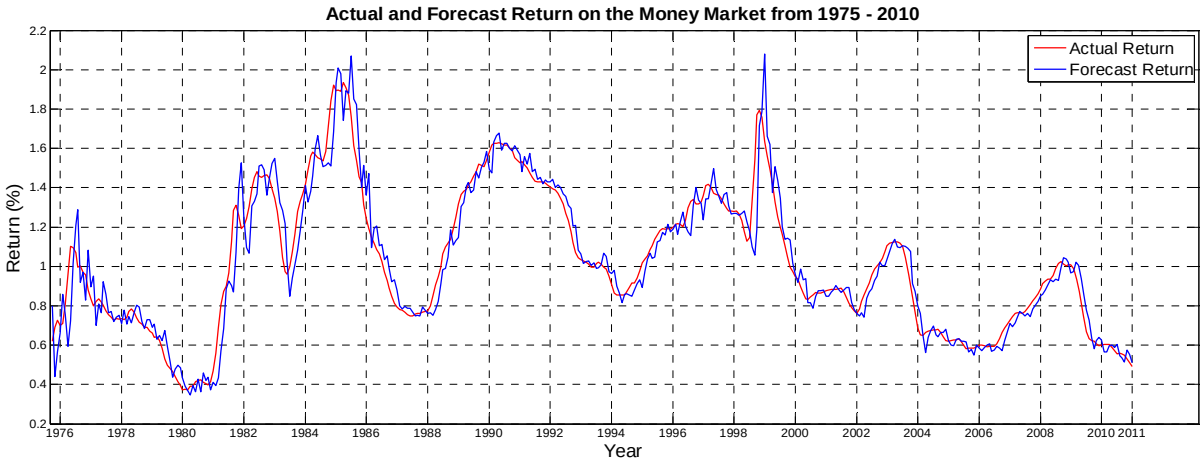

Figure M6: Actual and forecast return on the money market generated by integrated hybrid model – three steps