An investigation into the use of combined linear and neural network models for time series data

AS Kruger

11954086

Dissertation submitted in partial fulfillment of the requirements for the degree Master of Science at the Vaal Triangle campus of the North-West University

Supervisor: Prof P.D. Pretorius

August 2009

Vaal Triangle campus

ABSTRACT

Time series forecasting is an important area of forecasting in which past observations of the same variable are collected and analyzed to develop a model describing the underlying relationship. The model is then used to extrapolate the time series into the future. This modeling approach is particularly useful when little knowledge is available on the underlying data generating process or when there is no satisfactory explanatory model that relates the prediction variable to other explanatory variables. Time series can be modeled in a variety of ways e.g. using exponential smoothing techniques, regression models, autoregressive (AR) techniques, moving averages (MA) etc. Recent research activities in forecasting also suggested that artificial neural networks can be used as an alternative to traditional linear forecasting models. This study will, along the lines of an existing study in the literature, investigate the use of a hybrid approach to time series forecasting using both linear and neural network models. The proposed methodology consists of two basic steps. In the first step, a linear model is used to analyze the linear part of the problem and in the second step a neural network model is developed to model the residuals from the linear model. The results from the neural network can then be used to predict the error terms for the linear model. This means that the combined forecast of the time series will depend on both models. Following an overview of the models, empirical tests on real world data will be performed to determine the forecasting performance of such a hybrid model. Results have indicated that depending on the forecasting period, it might be worthwhile to consider the use of a hybrid model.

Keywords: Time series, forecasting, linear models, neural networks, hybrid models.

OPSOMMING

Die voorspelling van tydreekse is 'n belangrike aspek waar waarnemings ten opsigte van dieselfde veranderlike oor 'n tyd versamel en ontleed word om sodoende 'n model te kan ontwikkel wat die onderliggende verwantskap kan beskryf. Die model word dan gebruik om die tydreeks te ekstrapoleer in die toekoms. Hierdie benadering is veral nuttig wanneer min inligting oor die data genereringsproses beskikbaar is of wanneer geen bevredigende model bestaan wat die verwantskap tussen 'n afhanklike en onafhanklike veranderlikes aandui nie. Die modellering van tydreekse kan op verskillende maniere gedoen word, byvoorbeeld, deur die gebruik van 'n eksponensiële gladmaking proses, regressie modelle, bewegende gemiddeldes ens. Onlangse navorsing toon ook aan dat kunsmatige neurale netwerke as alternatief vir die tradisionele lineêre modelle kan dien. Hierdie studie gaan aan die hand van 'n bestaande studie in die literatuur, ondersoek instel na die gebruik van 'n hibriede benadering tot tydreekse waar beide lineêre en neurale netwerke gebruik word vir voorspellings. Die voorgestelde metodologie bestaan uit twee stappe. In die eerste stap word 'n lineêre model gebruik om die lineêre gedeelte van die probleem aan te spreek terwyl in die tweede stap 'n neurale netwerk gebruik word om die residue van die lineêre model te modelleer. Die resultate van die neurale netwerk kan dan gebruik word om die foute van die lineêre model te voorspel. Dit beteken dat die gekombineerde voorspelling van die tydreeks afhanklik van beide modelle is. 'n Oorsig van die modelle sal aangebied word asook empiriese toetse op regte data sodat die prestasie van so 'n hibriede model geëvalueer kan word. Resultate het aangedui dat, afhangende van die voorspellingstydperk, die gebruik van 'n hibriede model oorweeg behoort te word.

Sleutelwoorde: Tydreeks, voorspelling, lineêre modelle, neurale netwerke, hibriede modelle.

Acknowledgements

- All the glory to God who gave me wisdom and strength to complete this work
- Thank you to Prof Philip Pretorius, my supervisor, for his support, comments and willingness to share his knowledge.

CONTENTS

CHAPTER 1
INTRODUCTION AND PROBLEM STATEMENT

1.1 Introduction

Predictions of future events and conditions are called forecasts, and the act of making such predictions is called forecasting (Bowerman *et al*, 2005). Forecasting models form an integral part of any business' decision-making process and examples of where business forecasts are needed can be found in areas such as marketing, finance, human resources, production scheduling, process control etc. Forecasting models help to set targets and goals for future performance and may assist with determining staffing requirements, raw materials, capital and equipment needs.

To perform a forecast, past data is analyzed to identify a pattern that can be used to describe it. This pattern can then be used to prepare a forecast for the future – such an approach is based on the assumption that the identified pattern will continue in future. Quantitative forecasts can be developed for cross-sectional data (values observed at one point in time) or for time-ordered or time series data. A time series is defined as a chronological sequence of observations on a particular variable (Bowerman *et al*, 2005) and will be the data type used in this study.

The purpose of this chapter is to guide the reader into the research project by explaining the problem statement, objectives of the study and the methodology that will be followed. A layout of the study, explaining the purpose of each chapter, is also presented.

1.2 Problem statement

Time series forecasting is an important area of forecasting in which past observations of the same variable are collected and analyzed to develop a model describing the underlying relationship. The model is then used to extrapolate the time series into the future. This modeling approach is particularly useful when little knowledge is available

on the underlying data generating process or when there is no satisfactory explanatory model that relates the prediction variable to other explanatory variables.

Time series can be modeled in a variety of ways e.g. using exponential smoothing techniques. regression models, autoregressive (AR) techniques, moving averages (MA) etc. One of the most important and widely used time series models is the autoregressive integrated moving average (ARIMA) model. The popularity of the ARIMA model is due to its statistical properties as well as the well-known Box–Jenkins methodology in the model building process – see for example Bowerman *et al* (2005) and Zhang (2003). Zhang (2003) noted that there is however a major limitation to these types of models – the pre-assumed linear form of the models. That means a linear correlation structure is assumed among the time series values and therefore, no nonlinear patterns can be captured by, for example, the ARIMA model. The approximation of linear models to complex real-world problem is therefore not always satisfactory.

Recent research activities in forecasting suggested that artificial neural networks can be used as an alternative to traditional linear forecasting models. The major advantage of neural networks is their flexible nonlinear modeling capability and the use of such artificial neural networks have been extensively studied and used in time series forecasting. See for example Gareta, Romeo and Gil (2006) and Bodyanskiya and Popov (2006). The major advantage of neural networks is their flexible nonlinear modeling capability. The combination of different modeling techniques has also become a popular way of trying to improve forecasts – specifically the use of linear and neural network models seems to have received attention from researchers. Examples of work being carried out in this area can be found in Ho, Xie and Goh (2002), Ince and Trafalis (2005), Pai and Lin (2005), Prybutok and Mitchell (2002) and Tseng, Yu and Tzeng (2002). A brief overview of additional examples will be given in chapter 5.

This study will, along the lines of the Zhang study (2003), investigate the use of a hybrid approach to time series forecasting using both linear and neural network models. The proposed methodology consists of two basic steps. In the first step, a linear model is used

to analyze the linear part of the problem and in the second step a neural network model is developed to model the residuals from the linear model. The results from the neural network can then be used to predict the error terms for the linear model. This means that the combined forecast of the time series will depend on both models. Chapter 5 will present details on the combination of the two techniques but for the purpose of the problem statement, it is very briefly mentioned below.

It may be reasonable to consider a time series to be composed of a linear autocorrelation structure and a nonlinear component.

That is,

$$y_t = L_t + N_t \tag{1.1}$$

where L_t denotes the linear component and N_t denotes the nonlinear component.

These two components have to be estimated from the data. First, a linear model is used to model the linear component, and then the residuals from the linear model will contain only the nonlinear relationship. Let e_t denote the residual at time t from the linear model, then

$$e_t = y_t - \hat{L}_t \tag{1.2}$$

where $\hat{L}_t$ is the forecast value for time t from the estimated relationship.

By modeling residuals using a neural network, nonlinear relationships can be discovered. With n input nodes, the neural network model for the residuals will be

$$e_t = f(e_{t-1}; e_{t-2}; \ldots; e_{t-n}) + \varepsilon_t \tag{1.3}$$

where f is a nonlinear function determined by the neural network and ε_t is a random error. Denote the forecast from the neural network as $\hat{N}_t$, and the combined forecast will then be

$$\hat{y}_t = \hat{L}_t + \hat{N}_t \tag{1.4}$$

Following an overview of the models, empirical tests on real world data will be performed to determine the forecasting performance of such a hybrid model.

1.3 Objectives of the study

The primary objective of this research project is to investigate the use of a combined linear and neural network model to determine the forecasting performance of such a hybrid model. This will be accomplished by addressing the following secondary research objectives.

- Gain a clear understanding of and present an introductory overview of time series analysis and different forecasting methods;
- Gain a clear understanding of and present an introductory overview of neural networks;
- Gain a clear understanding of and present a brief introduction to the well-known Box-Jenkins approach;
- Investigate, describe and formulate a combined linear and neural network model; and
- Investigate the performance and forecasting accuracy of the combined model by applying it to real world time series data.

1.4 Methodology

The project will start with a general literature survey that will be used to give an overview of time series analysis and the different forecasting methods and then, to present the necessary background to neural networks. A more focused literature survey will be carried out to investigate the use of combining time series forecasting techniques and neural networks. Finally, empirical work will be conducted to test and present the results of a combined model. Actual time series data will be used for the empirical tests.

1.5 Layout of study

The project is documented through a set of chapters and this section explains the purpose of each chapter and how it is structured.

Chapter 2 will present an overview of time series forecasting techniques. Main techniques such as exponential smoothing, autoregressive and moving averages will be

discussed. Chapter 3 will be devoted to an explanation of artificial neural networks while chapter 4 will focus on the Box-Jenkins approach to nonseasonal time series forecasting. In chapter 5 the use of linear and neural networks in the forecasting of time series data, as well as an overview of using the two techniques as a hybrid model will be given. Chapter 6 will present the results of empirical tests performed to determine the forecasting performance of a hybrid model. The last chapter, chapter 7, will then summarize the goals set forth for the study and how they were achieved. Opportunities for further studies, identified during the research project, will also be pointed out.

The abovementioned chapters are supplemented by a set of appendices which contains details of work related to the study.

1.6 Conclusion

Chapter 1 served as an introduction and guided the reader into the research project by explaining the problem statement, objectives of the study and the methodology that will be followed. A layout of the study, explaining the purpose of each chapter, was also presented. In the next chapter an overview, from the literature, of time series forecasting techniques will be presented.

CHAPTER 2
TIME SERIES FORECASTING MODELS

2.1 Introduction

The two main areas of study that will be involved in this research project are time series and neural networks. To provide sufficient background and to have a good understanding of these two areas, this chapter presents an introductory overview of the first area, time series forecasting techniques. Following a brief introduction, the components of a time series are presented, as well as two methods to perform trend analysis. Next, an introductory discussion on seasonal analysis is given. Measures of forecast accuracy and the exponential smoothing procedure, known as the exponentially weighted moving average, forms the content of the second half of the chapter. Aspects such as tracking signals, as well as other methods, e.g. the Holt-Winters method, will also briefly be reviewed.

2.2 Time series forecasting models

2.2.1 Introduction

There are numerous ways to classify forecasting models – one classification that is often used is the distinction between quantitative and qualitative forecasting techniques (Moore and Weatherford, 2001).

Qualitative forecasting techniques attempt to incorporate judgmental or subjective factors into the forecasting model (Render *et al*, 2006). Opinions by experts, individual experiences and judgments and other subjective factors may be considered. Examples of qualitative models include the Delphi method (iterative group process), jury of executive opinion (opinions of a small group of high-level managers), sales force composite (combined estimates of people – salespersons – at different levels) and consumer market surveys (input from customers or potential customers regarding their future purchasing plans).

Quantitative forecasting models can be categorized as causal models and time series models. Causal models, also called explanatory forecasting, assume a cause and effect relationship between the inputs to a system, and its output. If y denotes the true value for some variable of interest, and $\hat{y}$ denotes a predicted or forecast value for that variable, then, in a causal model the following is true $\hat{y} = f(x_1, x_2, \ldots, x_n)$ where f is a forecasting rule, or function, and $x_1, x_2, \ldots, x_n$ is a set of variables. The x_i variables are often called independent variables, whereas $\hat{y}$ is the dependent or response variable. The notion is that the independent variables are known and that they are used to forecast the dependent variable. A common approach to create causal forecasting models is curve fitting e.g. least squares fits.

The second class of quantitative forecasting models is the time series forecasting models. These models produce forecasts by extrapolating the historical behavior of the values of a particular single variable of interest (Moore and Weatherford, 2001). A time series is defined by Wegner (1993) as a set of observations of a random variable arranged in chronological (time) order and the purpose of time series analysis is stated by him as 'to identify any recurring patterns which could be useful in estimating future values of the time series'. An important assumption in time series analysis is the continuation of past patterns into the future – i.e. the environment in which the time series occurs is reasonably stable. Another assumption is that there are four underlying components that individually and collectively determine the variable's value in a time series. The next section presents the four components of a time series.

2.2.2 Components and decomposition of a time series

A time series typically has four components (Wegner, 1993):

- Trend (T)
- Cyclical variations (C)
- Seasonal variations (S)
- Irregular variations (I)

Trend is the gradual upward or downward movement of data over time. It describes the effect that long-term factors may have on the series. The long-term factors usually tend to operate fairly gradually and in one direction for a considerable period of time. A statistical technique, trend analysis, can be used to isolate the underlying long-term movement and will be introduced in the next section.

Cycles are medium to long-term deviations from the trend and reflect periods of relative expansion and contraction. Cycles can be caused by certain actions of bodies such as governments (e.g. change in fiscal or monetary policy, sanctions etc.), trade unions, world organizations etc. These actions can induce levels of pessimism or optimism into an economy which are then reflected in time series data. Cycles can vary greatly in both duration and amplitude and are therefore difficult to measure statistically – their use in statistically forecasting is limited.

Seasonal variations are fluctuations that are repeated periodically and at regular intervals. Events such as climatic conditions, special occurring events (e.g. shows), and religious, public and school holidays are examples of causes of seasonal fluctuations. Due to the high degree of regularity it can be readily isolated through statistical analyses. Seasonal indices are used to measure the regular pattern of seasonal fluctuations and will also be addressed in the next section.

Irregular variations or random fluctuations in a time series are attributed to unpredictable occurrences and follow no discernible pattern. They are generally caused by once-off events such as natural disasters (floods, droughts, fires etc.) or man-made disasters (strikes, boycotts, accidents, acts of violence such as war, riots etc.) – because they are so unpredictable with no specific pattern, they are not really incorporated into statistical forecasts.

Time series analysis aims to isolate the influence of each of the four components on the actual series. Decomposition models, where the idea is to decompose the time series into the four factors, are used in an effort to reach this goal. Two decomposition models exist that can be used (Bowerman *et al*, 2005):

- A *multiplicative* decomposition model which has been found useful when modeling time series that display increasing or decreasing seasonal variation and which is defined as

$$y = T \times C \times S \times I \quad (2.1)$$

- An *additive* decomposition model which can be employed when modeling time series that exhibit constant seasonal variation. This model is defined as

$$y = T + C + S + I \quad (2.2)$$

Comprehensive discussions and examples on the decomposition models can be found in Bowerman *et al* (2005).

Statistical analysis can be used to effectively isolate the trend (T) and the seasonal (S) components, but is of less value in quantifying the cyclical movements, and of no value in isolating the irregular component (Wegner, 1993). Sections 2.2.3 and 2.2.4 will therefore examine statistical approaches to quantify T and S.

2.2.3 Trend analysis

The trend in a time series can be identified by averaging out the short term fluctuations in the series. Two methods for trend isolation can be used – they are

- Moving average method
- Regression analysis

2.2.3.1 The moving average method

A moving average removes the short term fluctuations in a time series by taking successive averages of groups of observations. Each time period's value is replaced by the average of observations which surround it. This is known as smoothing a time series (Wegner, 1993).

The simplest model in the moving average category is the simple n-period moving average. In this model the average of a fixed number (say, n) of the most recent observations is used as an estimate of the next value of a variable y and is defined as

$$\hat{y}_{t+1} = \frac{1}{n}(y_t + y_{t-1} + ... + y_{t-n+1}) \quad (2.3)$$

Moore and Weatherford (Moore and Weatherford, 2001) highlighted the fact that the simple moving average has two shortcomings. Firstly, when calculating a forecast, the most recent observation receives no more weight or importance than older observations. This is because each of the last n observations is assigned a weight of 1/n. This is in conflict with the general view that in many instances the more recent data should tell us more than the older data about the future. The second shortcoming is of an operational nature and concerns the storage of data. If n observations are to be included in the moving average, then n-1 pieces of past data must be brought forward to be combined with the nth observation. All this data must be stored in some way in order to calculate the forecast. This is not too much of a serious problem when taking into account the availability of current computing resources but it may become an issue when dealing with exceptionally large data sets and models.

To cater for the first shortcoming, recent data that are more important than older data, a weighted n-period moving average can be implemented. This is defined as

$$\hat{y}_{t+1} = \alpha_0 y_t + \alpha_1 y_{t-1} + ... + \alpha_n y_{t-n+1} \quad (2.4)$$

where the α's (which are called weights) are nonnegative numbers that are chosen so that smaller weights are assigned to older data and all the weights sum to 1. There are many ways of selecting a set of α's – one way to choose optimal weights is to make use of a linear program that minimizes the mean absolute deviation (this concept is defined in section 2.4) subject to the constraints $\sum \alpha_i = 1$; $\alpha_0 \geq \alpha_1 \geq ... \geq \alpha_n$; and $0 \leq \alpha_i \leq 1$. See Moore and Weatherford (Moore and Weatherford, 2001) for a worked example.

The major benefit of a moving average is the opportunity it affords a decision maker to focus more clearly on the long term trend movements by removing short term fluctuations (i.e. seasonal and irregular fluctuations) from the original observations – thereby isolating the long term trend. In symbol terms for the multiplicative model, this can be stated as

Moving average = (T x C x S x I) / (S x I)

$$= T \times C \qquad (2.5)$$

There are other extensions of moving averages e.g. a double moving average (i.e. a moving average of a moving average) and moving average combinations (i.e. a n-period moving average combined with a k-period moving average with $n \neq k$). Discussions and examples of these extensions can be found in Makridakis *et al* (1983).

2.2.3.2 Time series regression

Another method often employed for trend line isolation is the use of time series regression models. In these models the dependent variable, y_t, which is the actual time series is related to functions of time (independent variable). The model shows the general direction in which the series is moving and is represented by using polynomial functions of time. In this section the formulation of no trend, linear trend and quadratic trend will be shown. The concept autocorrelation and how to detect it will also be briefly mentioned.

A time series, y_t, can sometimes be described by using a trend model. Such a trend model is defined as follows (Bowerman *et al*, 2005):

$$y_t = TR_t + \varepsilon_t \qquad (2.6)$$

where

y_t = the value of the time series in period t

TR_t = the trend in time period t

ε_t = the error term in time period t

Bowerman *et al* (2005) also presents useful trends (TR) that are often encountered. These trends can be summarized as

- No trend. This is modeled as $TR_t = \beta_0$ and implies that there is no long-run growth or decline in the time series over time. See figure 2.1(a).
- Linear trend. This is modeled as $TR_t = \beta_0 + \beta_1 t$ and implies that there is a straight line long-run growth (if the slope $\beta_1 > 0$) or decline (if $\beta_1 < 0$) over time. See figures 2.1(b) and 2.1(c).

– Quadratic trend. This is modeled as $TR_t = \beta_0 + \beta_1 t + \beta_2 t^2$ and implies that there is a quadratic (or curvilinear) long-run change over time. This quadratic change can either be *growth* at an increasing or decreasing rate – see figures 2.1(d) and 2.1(e) – or *decline* at an increasing or decreasing rate – see figures 2.1(f) and 2.1(g).

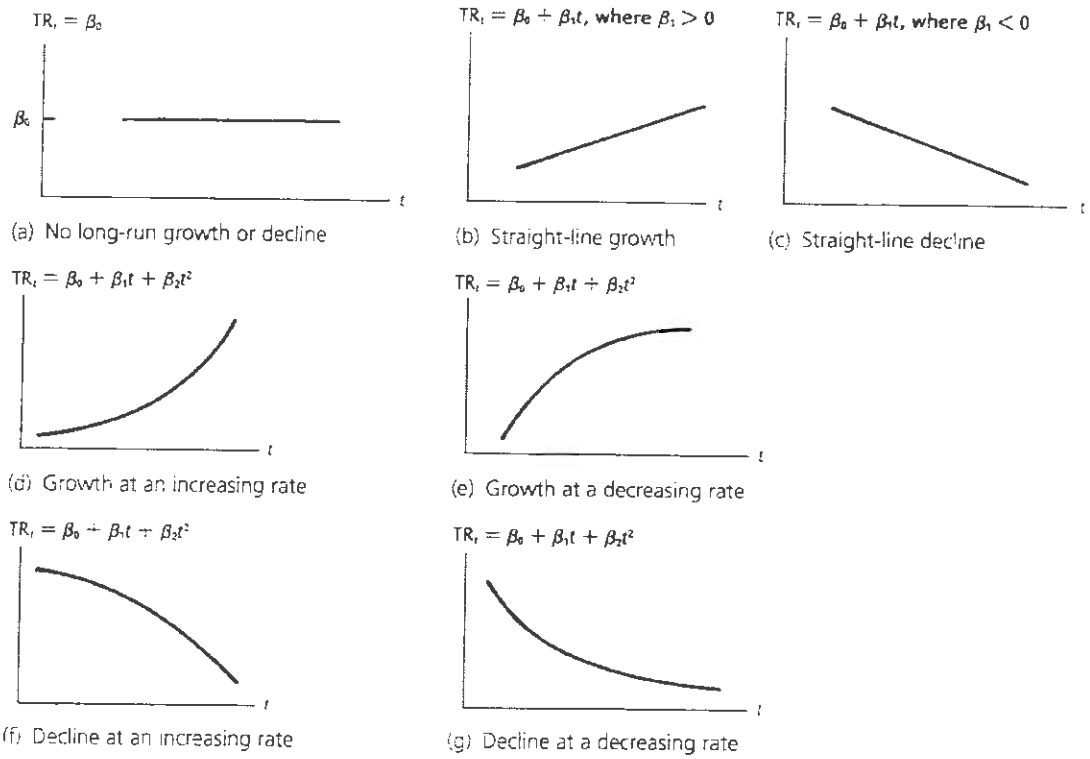


Figure 2.1

More complicated trend models can be modeled by using a p^{th}-order polynomial function where $TR_t = \beta_0 + \beta_1 t + \beta_2 t^2 + ... + \beta_p t^p + \varepsilon_t$.

Assuming a normal distribution of the error term, ε_t, least squares point estimates of the parameters in the above trend models may be obtained using regression techniques. Complete worked examples can be found in Bowerman *et al* (2005).

The validity of regression methods requires that the independence assumption (i.e. error terms occur in a random pattern over time) be satisfied. This assumption is violated when time-ordered error terms are auto correlated. The term autocorrelation can be defined as

"correlation between members of series of observations ordered in time [as in time series data] or space [as in cross-sectional data] (Gujarati, 2003). Bowerman *et al* (2005) explains the concept of positive and negative autocorrelation as follows.

Error terms occurring over time have *positive autocorrelation* if a positive error term in time period t tends to produce, or be followed by, another positive error term in time period t+k (a later period) or if a negative error term in time period t tends to produce, or be followed by, another negative error term in time period t+k. In other words, positive autocorrelation exists when positive error terms tend to be followed over time by positive error terms and negative error terms tend to be followed over time by negative error terms.

Error terms occurring over time have a *negative autocorrelation* if a positive error term in time period t tends to produce, or be followed by, a negative error term in time period t+k and if a negative error term in time period t tends to produce, or be followed by, a positive error term in time period t+k. In other words, negative autocorrelation exists when positive error terms tend to be followed over time by negative error terms and negative error terms tend to be followed over time by positive error terms.

One way of verifying if errors show any kind of pattern is to plot them and use visual inspection. A formal test, called the Durbin-Watson test, can however be performed to test for positive or negative autocorrelation. The Durbin-Watson statistic used in this test is sensitive to the different patterns mentioned above and is defined as follows (Makridakis *et al*, 1983).

$$d = \frac{\sum_{t=2}^{n} (e_t - e_{t-1})^2}{\sum_{t=1}^{n} e_t^2} \qquad (2.7)$$

where e_t are the time-ordered errors.

The test can now be implemented as follows (Bowerman *et al*, 2005):
Consider testing the null hypothesis

H_0 : The error terms are not autocorrelated

versus H_1 : The error terms are positively autocorrelated

There exists points, denoted by $d_{L,\alpha}$ and $d_{U,\alpha}$, such that if α is the probability of a Type I error (probability of rejecting H_0 when in fact it is true) then

1. If $d < d_{L,\alpha}$, H_0 is rejected
2. If $d > d_{U,\alpha}$, H_0 is not rejected
3. If $d_{L,\alpha} \leq d \leq d_{U,\alpha}$ the test is inconclusive.

The theory behind the statistic and the rules for rejection is complicated and beyond the scope of this study. Details on this can be found in Makridakis *et al* (1983).

Should the alternative hypothesis be changed to test for negative autocorrelation

i.e. H_1 : The error terms are negatively autocorrelated

the rejection rules become

1. If $(4-d) < d_{L,\alpha}$, H_0 is rejected
2. If $(4-d) > d_{U,\alpha}$, H_0 is not rejected
3. If $d_{L,\alpha} \leq (4-d) \leq d_{U,\alpha}$ the test is inconclusive.

Finally, the test can also be used to test for positive or negative autocorrelation in which case the alternative hypothesis becomes

H_1 : The error terms are positively or negatively autocorrelated

and the rules

1. If $d < d_{L,\alpha/2}$ or if $(4-d) < d_{L,\alpha/2}$, H_0 is rejected
2. If $d > d_{U,\alpha/2}$ and if $(4-d) > d_{U,\alpha/2}$, H_0 is not rejected
3. If $d_{L,\alpha/2} \leq d \leq d_{U,\alpha/2}$ or if $d_{L,\alpha/2} \leq (4-d) \leq d_{U,\alpha/2}$ the test is inconclusive.

The Durbin-Watson test proves to be useful in testing for autocorrelation and is usually provided as standard output by most computer regression packages. It should be noted however, that time series data can exhibit more complicated auto correlated error

structures. In such cases autocorrelation can be detected by using a sample autocorrelation function. This function will be explained and discussed in chapter 4.

2.2.4 Seasonal analysis

In this section a brief introduction to seasonal analysis as a technique to isolate the influence of seasonal forces on a time series will be given.

One way to measure seasonal influences is to make use of a ratio-to-moving-average method (Wegner, 1993). In this case the seasonal influence is expressed as an index number and measures the percentage deviation of the actual values of the series from a base value which excludes the short term seasonal influences. The method is summarized by Wegner (1993) as follows:

- The first step is to identify the trend or cyclical movement and is done by the moving average approach discussed earlier.
- Next, a seasonal ratio is calculated. This is done by dividing each actual time series value, y, by its corresponding moving average value i.e.

$$\begin{aligned} \text{Seasonal ratio} &= \text{actual } y / \text{ Moving average series} \times 100\% \\ &= (T \times C \times S \times I) / (T \times C) \times 100\% \\ &= S \times I \times 100\% \end{aligned} \tag{2.8}$$

  The seasonal ratio is an index which expresses the percentage deviation of each actual y (which includes seasonal influences) from its moving average value (which contains trends and cyclical influences only) and is a measure of seasonal influence.
- In the third step, the seasonal ratios are averaged across corresponding periods within each time frame (e.g. a year). Averaging has the effect of smoothing out the irregular component inherent in the seasonal ratios. Often the median is used as the average of seasonal ratios – this is to prevent the influence of outliers when using an arithmetic mean.
- Lastly, adjusted seasonal indices are computed. As each seasonal index has a base of 100, the sum of the n median seasonal indices must equal 100n and the adjustment factor is then determined as

  Adjustment factor = $100n / \Sigma(\text{median seasonal indices})$

Each median seasonal index is then finally multiplied by the adjustment factor to ensure a base of 100 for each index. The resulting adjusted seasonal indices are then regarded as a measure of the seasonal influences on the actual values of the time series for each given time period.

By subtracting the base index of 100 from each seasonal index, the extent of the influence of seasonal forces can be gauged. For example, a seasonal index of, say, 79 means that values of the time series are depressed by the presence of seasonal forces to the extent of approximately 21%. Alternatively, values of the time series would be approximately 21% higher had seasonal influences not been present. The same logic is followed to interpret a seasonal index above 100.

2.2.5 Constructing a forecast

Actual time series values are assumed to be a function of trend (T), cyclical I, seasonal (S) and irregular (I) components (see also section 2.2.2). This means that if these components are known, they can be used to re-construct values of the actual time series. In the preceding sections the trend and seasonal components were discussed and it was shown how they can be quantified. To estimate future values of an actual time series, y, taking these two influences into account, the following can be done

- Use a trend line to estimate the trend value of the time series, e.g. $y_t = TR_t + \varepsilon_t$ for time period t and with $TR_t = \beta_0 + \beta_1 t$ (say).
- Incorporate the seasonal influence by multiplying the trend value (TR_t) by the seasonal index for the appropriate time period. This is known as *seasonalising* the trend (Wegner, 1993).

2.3 Measures of forecast accuracy

The previous section concluded with comments on how to construct a forecasting model in an ordinary time series model. The remainder of this chapter, and all subsequent chapters, are focused on forecasting models and forecasting issues and it is therefore appropriate to introduce at this stage some thoughts on forecast evaluation and accuracy measures.

Forecasts will not be completely accurate and will almost always deviate from actual values. A forecast error is the difference between the forecast and actual value (Taylor, 2002). To see how well one model works, or to compare that model with other models, the forecasted values are compared with the actual or observed values. One of the most popular and easiest to use measures is called the mean absolute deviation (MAD). The MAD is computed as

$$MAD = \frac{\sum |actual - forecast|}{(number\ forecasts)} \tag{2.9}$$

The lower the value of the computed MAD relative to the magnitude of the data, the more accurate the forecast.

Computing the MAD value enables a decision maker to compare the accuracy of several different forecasting techniques. It also makes the monitoring of forecasts possible which is necessary to ensure that a chosen forecast model keeps on performing well. A well known instrument to measure how well predictions fit actual data is called a tracking signal (Render *et al*, 2006). A tracking signal is computed as

Tracking signal = (Running sum of the forecast errors) / MAD

= Σ(actual value in time t – forecast value in time t) / MAD (2.10)

Render *et al* (2006) stated that a good tracking signal (one with a low running sum of forecast errors) has about as much positive error as it has negative error – small deviations are acceptable, but the positive and negative deviations should balance so that the tracking signal centers closely around zero. Tracking signals are often computed with predetermined upper and lower control limits to determine possible problems with the forecasting method (Render *et al*, 2006).

Other well known measures of forecasting include:

- The mean absolute percentage error (MAPE) which is the average of the absolute values of the errors expressed as percentages of the actual values and is defined as

$$MAPE = \frac{\sum \frac{|actual - forecast|}{actual} * 100\%}{(number\ forecasts)} \quad (2.11)$$

- The mean squared error (MSE) which is the average of the squared errors.
- The average error, also called bias, which is computed by averaging the cumulative error over the number of time periods (Taylor, 2002). It tells a decision maker whether forecasts tend to be too high or too low and by how much.

There exist a large number of accuracy measures that have been used to evaluate the performance of forecasting methods and this section is concluded by presenting a list, in table 1, of the most commonly used methods (De Gooijer & Hyndman P458).

Table 1 – Commonly used forecast accuracy measures (De Gooijer and Hyndman, 2006)

MSE	Mean squared error	= mean(e_t^2)
RMSE	Root mean squared error	= $\sqrt{MSE}$
MAE	Absolute error	= mean($\|e_t\|$)
MdAE	Median absolute error	= median($\|e_t\|$)
MAPE	Mean absolute percentage error	= mean($\|p_t\|$)
MdAPE	Median absolute percentage error	= median($\|p_t\|$)
sMAPE	Symmetric mean absolute percentage error	= mean($2\|Y_t - F_t\| / (Y_t + F_t)$)
sMdAPE	Symmetric median absolute percentage error	= median($2\|Y_t - F_t\| / (Y_t + F_t)$)
MRAE	Mean relative absolute error	= mean($\|r_t\|$)
MdRAE	Median relative absolute error	= median($\|r_t\|$)
GMRAE	Geometric mean relative absolute error	= gmean($\|r_t\|$)
RelMAE	Relative mean absolute error	= MAE/MAE_b
RelRMSE	Relative root mean squared error	= $RMSE/RMSE_b$
LMR	Log mean squared error ratio	= log(RelMSE)
PB	Percentage better	= 100 mean($I\{\|r_t\| < 1\}$)
PB(MAE)	Percentage better (MAE)	= 100 mean(I{MAE <

		MAE_b})
PB(MSE)	Percentage better (MSE)	= 100 mean(I{MSE < MSE_b})
	r indicates relative error; *e* indicates error term; *b* refers to measures obtained from the base method; I{*u*} = 1 if *u* is true and 0 otherwise.	

2.4 Exponential smoothing

2.4.1 Introduction and Definition

Exponential smoothing, also called exponentially weighted moving average is a method where recent data is weighted more heavily than past data (Moore and Weatherford, 2001). The method is often used for forecasting a time series when there is no trend or seasonal pattern but the level of the time series is slowly changing over time (Bowerman *et al*, 2005). The procedure allows the forecaster to update the estimate of the level of the time series so that changes in the level can be detected and incorporated into the forecasting system.

Moore and Weatherford (2001) define the basic exponential smoothing model as follows: For any time period t ≥ 1 the forecast for period t+1, denoted by $\hat{y}_{t+1}$ is a weighted sum (with weights summing to 1) of the actual observed values in period t (i.e. y_t) and the forecast for period t (which was $\hat{y}_t$). This gives

$$\hat{y}_{t+1} = \alpha y_t + (1-\alpha)\hat{y}_t \qquad (2.12)$$

where α is a user-specified smoothing constant such that $0 \leq \alpha \leq 1$. The value assigned to α determines how much weight is placed on the most recent observation in calculating the forecast for the next period.

To perform an exponential smoothing forecast it would be necessary to estimate an initial value for $\hat{y}_1$. This can be done by simply letting $\hat{y}_1 = y_1$, assuming a perfect forecast for time period 1 (Moore and Weatherford, 2001) or by letting $\hat{y}_1 = \bar{y}$ (Bowerman *et al*, 2005).

The basic exponential smoothing model is of great importance and it is worthwhile to present a more detailed explanation on how it works. Some of its properties as given by Moore and Weatherford (2001) will provide such an explanation.

If $t \geq 2$, it is possible to substitute t-1 for t in (4.1) to obtain

$$\hat{y}_t = \alpha y_{t-1} + (1-\alpha)\hat{y}_{t-1} \tag{2.13}$$

Substituting this relationship for $\hat{y}_t$ back into the original expression (4.1) for $\hat{y}_{t+1}$ yields for $t \geq 2$

$$\hat{y}_{t+1} = \alpha y_t + \alpha(1-\alpha)y_{t-1} + (1-\alpha)^2 \hat{y}_{t-1} \tag{2.14}$$

By successively performing similar substitutions, one is led to the following general expression for $\hat{y}_{t+1}$

$$\hat{y}_{t+1} = \alpha y_t + \alpha(1-\alpha)y_{t-1} + \alpha(1-\alpha)^2 y_{t-2} + ... + \alpha(1-\alpha)^{t-1} y_1 + (1-\alpha)^t \hat{y}_1 \tag{2.15}$$

For example, if t = 3

$$\hat{y}_4 = \alpha y_3 + \alpha(1-\alpha)y_2 + \alpha(1-\alpha)^2 y_1 + (1-\alpha)^3 \hat{y}_1$$

since $0 < \alpha < 1$, it follows that $0 < 1 - \alpha < 1$

thus $\alpha > \alpha(1-\alpha) > \alpha(1-\alpha)^2$

in other words, in the example, where t = 3, the most recent observation, y_3, receives more weight than y_1. This illustrates the general property of an exponential smoothing model – that the coefficients of the y's decrease as the data become older. The sum of all coefficients is one. In the example, $\alpha + \alpha(1-\alpha) + \alpha(1-\alpha)^2 + (1-\alpha)^3 = 1$ when simplified.

It is now easy to observe that as t increases, the influence of $\hat{y}_1$ (which was initially estimated) on $\hat{y}_{t+1}$ decreases and in time becomes negligible. The coefficient of $\hat{y}_t$ in (4.2) is $(1-\alpha)^t$. Thus, the weight assigned to $\hat{y}_1$ decreases exponentially with t.

It should be clear now that the value of α, which is a parameter input by the decision maker, will affect the performance of the model – the larger the value for α, the more strongly the model will react to the last observation. The next section looks at the choice of values for α.

2.4.2 Choosing values for α

Selecting an appropriate value for the smoothing constant, α, can have a significant impact on the accuracy of forecasts. A possible approach is to simply try different values for α and the best value, based on some accuracy measure such as the MAD or MSE, is then used (Makridakis *et al*, 1983). Another way of selecting an optimal value for α is to make use of a linear program as described in section 2.3.1. In this case a linear program that minimizes the MAD is used to choose an optimal value for α. Bowerman *et al* (2005) noted that most computer software packages automatically choose values for α but that different approaches are used and that users should carefully investigate how it is implemented.

To further illustrate the effect of choosing values for α (i.e. putting more or less weight on recent observations), three specific cases are considered (Moore and Weatherford, 2001).

Response to a sudden change

Suppose that at a certain point in time a system experiences a rapid and radical change. Consider an extreme case where

$y_t = 0$ for t = 1, 2, ..., 99

$y_t = 1$ for t = 100, 101, ...

In this case, if $\hat{y}_1 = 0$, then $\hat{y}_{100} = 0$ for any value of α as the weighted sum of a series of zeroes was taken. Thus at time 99 the best estimate of y_{100} is zero, whereas the actual value will be one. The question now is how quickly will the forecasting system respond as time passes and the information that the system has changed becomes available? It is clear that a higher value of α – i.e. more weight on recent observations – will respond quicker. Moore and Weatherford (2001) have shown graphically that a higher value of α, in this case, is more desirable. Therefore, when a system is characterized by a low level of random behavior, but is subject to occasional shocks (rapid and radical change) a relative large α-value would be preferred.

Response to a steady change

Suppose that a system experiences a steady (growing) change in the value of y – this is sometimes called a linear ramp. In this situation, since all previous y's (y_1 .... y_{t-1}) are smaller than y_t, and since the weights sum to one, it can be shown that, for any α between 0 and 1, $\hat{y}_{t+1} < y_t$. Also, since y_{t+1} is greater than y_t, the following is true $\hat{y}_{t+1} < y_t < y_{t+1}$. Thus, the forecast will always be too small and since smaller values of α put more weight on older data, the smaller the value of α, the worst the forecast becomes. Moore and Weatherford (2001) warned that even with α close to one, the forecast will not be good as there may be a steep growth in y-values. In this case the model should be adjusted to include the trend. These types of models are dealt with in section 2.4.4.

Response to a seasonal change

Consider a system that experiences a regular seasonal pattern as illustrated in figure 2.2 below.

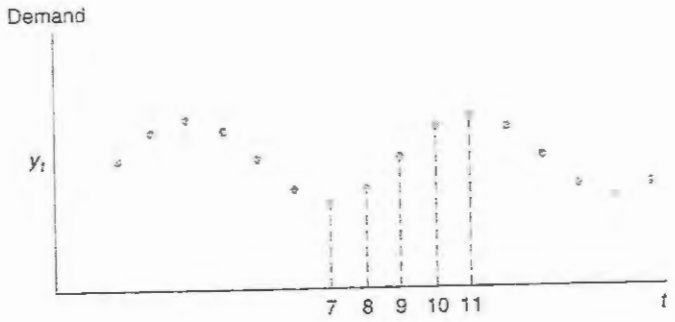


Figure 2.2

Suppose that values for periods 8 through 11 needs to be forecasted based only on data through period 7.

Then $\hat{y}_8 = \alpha y_7 + (1-\alpha)\hat{y}_7$

Now to obtain $\hat{y}_9$, since data is only available through point 7, it is assumed that $y_8 = \hat{y}_8$

Then $\hat{y}_9 = \alpha y_8 + (1-\alpha)\hat{y}_8$ which gives $\alpha \hat{y}_8 + (1-\alpha)\hat{y}_8$ and this simplifies to $\hat{y}_8$.

Similarly, it can be shown that $\hat{y}_{11} = \hat{y}_{10} = \hat{y}_9 = \hat{y}_8$. This means that $\hat{y}_8$ is the best estimate of all future values. To see how good these predictions are, we know that

$$\hat{y}_{t+1} = \alpha y_t + \alpha(1-\alpha)y_{t-1} + \alpha(1-\alpha)^2 y_{t-2} + ...$$

If a small value of α is chosen, the coefficients for the most recent terms change relatively slowly. Thus, $\hat{y}_{t+1}$ will resemble a simple moving average of a number of terms and the future predictions, e.g. $\hat{y}_{11}$, will all be somewhere near the average of the past observations – i.e. the seasonal pattern is ignored. If a large value of alpha is chosen, $\hat{y}_{11}$, which equals $\hat{y}_8$, will be close in value to y_7 which is not a good forecast. The model fares poorly regardless of the choice of α and based on this, Moore and Weatherford (2001) concluded that the exponential smoothing model is intended for situations in which the behavior of the variable of interest is essentially stable, in the sense that deviations over time have nothing to do with *time* per se but are caused by *random effects* that do not follow a regular pattern. If there is a definite trend or seasonal effect in the variable being predicted, it would be better to develop forecasting models that incorporate these figures e.g. those methods discussed earlier in sections 2.2.3 and 2.2.4.

2.4.3 Tracking signals

Different smoothing constant values may produce improved forecasts over time or under specific circumstances as shown in section 2.4.2. One way of deciding whether something is wrong with a forecasting system is to make use of tracking signals. Tracking signals were mentioned and defined in section 2.3 – measures of forecast accuracy. In this section, brief mention will be made of another tracking signal that has had extensive use in practice and that is called the smoothed error tracking signal (Bowerman *et al*, 2005).

Suppose that a history of T single-period-ahead forecast errors, $e_1(\alpha), e_2(\alpha), ...e_T(\alpha)$ exist with α the particular value of α employed to obtain the single-period-ahead forecast errors. The smoothed error tracking signal is defined as the ratio of the smoothed one-period-ahead forecasting error to the smoothed mean absolute deviation. If the smoothed error, E, of the one-period-ahead forecast error is defined as

$$E(\alpha, T) = e_t(\alpha) + \alpha E(\alpha, T-1) \qquad (2.16)$$

then the smoothed error tracking signal is defined as

$$S(\alpha,T)=\left|\frac{E(\alpha,T)}{MAD(\alpha,T)}\right| \tag{2.17}$$

Bowerman *et al* (2005) stated that it should be noted that tracking signals no longer play such an extensive role in forecasting. This is due to modern computer power and capacity which means that smoothing constants can be re-estimated frequently during the forecasting process.

2.4.4 Other exponential smoothing models

The simple exponential smoothing model discussed so far may not perform very well on models that have, for example, obvious up or down trends in the data with no seasonal pattern. For completeness sake, a few of the existing other models are briefly defined here. All definitions are quoted from Bowerman *et al* (2005).

Holt's trend corrected exponential smoothing model is a method that can be used to forecast a time series that has a linear trend and growth rate that is changing over time. It can be defined as follows.

Suppose that the time series $y_1, y_2, \ldots, y_n$ exhibits a linear trend for which the level and growth rate may be changing with no seasonal pattern. Then the estimate ℓ_T for the level of the time series and the estimate b_T for the growth rate of the time series in time period T are given by the smoothing equations

$$\begin{aligned} \ell_T &= \alpha(y_T)+(1-\alpha)(\ell_{T-1}+b_{T-1}) \\ b_T &= \gamma(\ell_T-\ell_{T-1})+(1-\gamma)b_{T-1} \end{aligned} \tag{2.18}$$

where α and γ are smoothing constants between zero and one, and ℓ_{T-1} and b_{T-1} are estimates at time T-1 for the level and growth rate respectively.

The additive Holt-Winters method is appropriate for time series with constant additive seasonal variation and is defined as follows.

Suppose that the time series $y_1, y_2, \ldots, y_n$ exhibits linear trend and has a seasonal pattern with constant additive seasonal variation and that the level, growth rate and seasonal pattern may be changing. Then the estimate ℓ_T for the level, the estimate b_T for the

growth rate, and the estimate sn_T for the seasonal factor of the time series in time period T are given by the smoothing equations

$$\begin{aligned} \ell_T &= \alpha(y_T - sn_{T-L}) + (1-\alpha)(\ell_{T-1} + b_{T-1}) \\ b_T &= \gamma(\ell_T - \ell_{T-1}) + (1-\gamma)b_{T-1} \\ sn_T &= \delta(y_T - \ell_T) + (1-\delta)sn_{T-L} \end{aligned} \tag{2.19}$$

where α, γ and δ are smoothing constants between 0 and 1, ℓ_{T-1} and b_{T-1} are estimates in time period T-1 for the level and growth rate, and sn_{T-L} is the estimate in time period T-L for the seasonal factor.

Time series with an increased (multiplicative) seasonal variation as opposed to the constant (additive) seasonal variation can be dealt with the *multiplicative Holt-Winters method.* It is defined the same way as the additive model but with the following smoothing equations

$$\begin{aligned} \ell_T &= \alpha(y_T / sn_{T-L}) + (1-\alpha)(\ell_{T-1} + b_{T-1}) \\ b_T &= \gamma(\ell_T - \ell_{T-1}) + (1-\gamma)b_{T-1} \\ sn_T &= \delta(y_T / \ell_T) + (1-\delta)sn_{T-L} \end{aligned} \tag{2.20}$$

One last method that will be mentioned here is called the *damped trend exponential smoothing* model. This method is appropriate for forecasting a time series which has a growth rate that will not be sustained into the future and whose effects should be dampened. This means reducing the growth rate in size so that the rate of increase or decrease for the forecasts is slowing down. The method is defined as follows.

Suppose that the time series $y_1, y_2, \ldots, y_n$ exhibits a linear trend for which the level and growth rate are changing somewhat with no seasonal pattern. Furthermore, suppose that it is questioned whether the growth rate at the end of the time series will continue in future. Then the estimate ℓ_T for the level and the estimate b_T for the growth rate are given by the smoothing equations

$$\begin{aligned} \ell_T &= \alpha y_T + (1-\alpha)(\ell_{T-1} + \phi b_{T-1}) \\ b_T &= \gamma(\ell_T - \ell_{T-1}) + (1-\gamma)\phi b_{T-1} \end{aligned} \tag{2.21}$$

where α and γ are smoothing constants between 0 and 1, and ϕ is a damping factor between 0 and 1.

The damped trend can be used with either the additive or multiplicative Holt-Winters method when dealing with seasonal data. Details on this can be found in Bowerman *et al* (2005)

2.5 Conclusion

In this chapter an introductory overview of time series forecasting techniques was presented. Aspects covered included components of time series, trend and seasonal analysis and measures of forecasts accuracy. Exponential smoothing as one of the more popular techniques in time series forecasting, was also briefly reviewed. Another important and widely used time series model, the so-called autoregressive integrated moving average (ARIMA) model, will be discussed in chapter 4.

The next chapter will give an overview and background on artificial neural network models – the other technique that forms the backbone of this research study.

CHAPTER 3
NEURAL NETWORK MODELS OVERVIEW

3.1 Introduction

In the previous chapter an overview of time series forecasting was given. As it is the primary objective of this research study to investigate the use of neural networks in combination with time series forecasting, this chapter will serve as an introduction to neural networks. General concepts of neural networks, and how they work, will be presented in order to provide sufficient background to the empirical experiments described in chapters to follow.

There are a large number of resources available that describe neural networks and in stead of referring to many different resources which all give the same basic information; it was decided to base the discussion in this chapter on the text book by Fausett (1994). Some of the sections are quoted from this source with out always referencing it continually.

3.2 What is a neural network?

A neural network is an information system modeled after the human brain's network of electronically interconnected basic processing elements called neurons (Awad, 1996). They are used for modeling a broad range of non-linear problems and are of interest to researchers and practitioners in many areas for different reasons. The study of neural networks is an interdisciplinary field, both in its development and its application. There are a huge number of neural network applications and some examples include fraud detection, target marketing systems, signature verification, loan approval, mortgage appraisals etc (Awad, 1996).

In the following sub-sections a brief description of what is meant by a neural network is given.

3.2.1 Artificial neural networks

According to Fausett (1994) artificial neural networks have been developed as generalizations of mathematical models of human cognition or neural biology. It is based on the assumptions that

- Information processing occurs at many simple elements called neurons
- Signals are passed between neurons over connection links
- Each connection link has an associated weight, which, in a typical neural net, multiplies the signal transmitted
- Each neuron applies an activation function (usually non linear) to its net input (sum of weighted input signals) to determine its output signal

Neural networks are characterized by the pattern of connections between the neurons (the architecture), the method of determining the weights on the connections (training or learning) and an activation function. These concepts are illustrated in subsequent sections and the defining characteristics are just briefly considered here.

A neural network consists of a large number of simple processing elements called neurons. Each neuron is connected to other neurons by means of directed communication links, each with an associated weight that represents information being used to solve a problem. Each neuron also has an internal state called the activation or activity level, which is a function of the inputs that was received. To illustrate, consider the following example taken from Fausett (1994).

Consider a neuron Y that receives inputs from neurons X_1, X_2 and X_3 - see figure 3.1. The activations (output signals) of these neurons are x_1, x_2 and x_3 respectively. The weights on the connections from X_1, X_2 and X_3 to neuron Y are w_1, w_2 and w_3 respectively. The net input y_in, to neuron Y is the sum of the weighted signals from neurons X_1, X_2 and X_3 i.e.,

$$y\_in = w_1x_1 + w_2x_2 + w_3x_3 . \qquad (3.1)$$

The activation y of neuron Y is given by some function of its net input, $y = f(y\_in)$, e.g., the logistic sigmoid function (an S-shaped curve)

$$f(x) = \frac{1}{1+\exp(-x)}, \tag{3.2}$$

or any of a number of other activation functions that will be mentioned again in section 3.5.

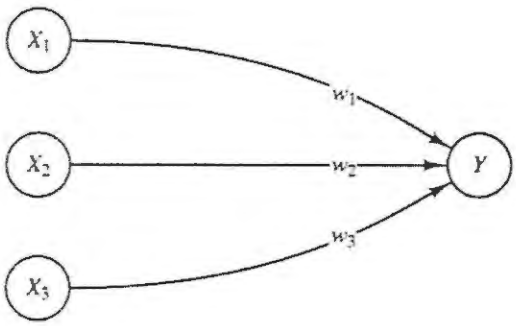


Figure 3.1 – A simple (artificial) neuron

Suppose further that neuron Y is connected to neurons Z_1 and Z_2, with weights v_1 and v_2 respectively as shown in figure 3.2. Neuron Y sends its signal y to each of these units. However, in general the values received by neurons Z_1 and Z_2 will be different, because each signal is scaled by the appropriate weight v_1 and v_2. In a typical net, the activations z_1 and z_2 of neurons Z_1 and Z_2 would depend on inputs from several or even many neurons and not just one as shown in this simple example.

Although the neural network in figure 3.2 is very simple, the presence of a hidden unit, together with a nonlinear activation function gives it the ability to solve many more problems that can be solved by a neural network with only input and output units. On the other hand, it is more difficult to train (i.e. find optimal values for the weights) a net with hidden units. The arrangement of the units (architecture) and the method of training are discussed further in subsequent sections.

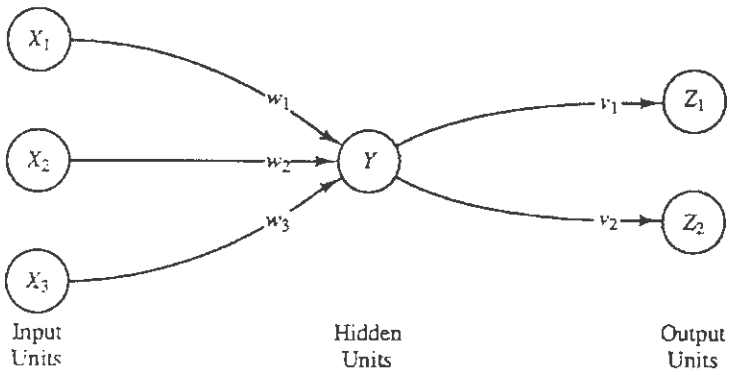


Figure 3.2 – A simple neural network

3.2.2 Biological neural networks

There is a close analogy between the structure of a biological neuron (i.e. a brain or a nerve cell) and the processing element (artificial neuron) in a neural network. In this section a short summarized discussion of some features of biological neurons that may help to clarify the most important characteristics of artificial neural networks are presented (Fausett, 1994).

A biological neuron has three types of components that are of particular interest in understanding an artificial neuron: its *dendrites*, *soma* and *axon*. The many dendrites receive signals from other neurons. The signals are electric impulses that are transmitted across a synaptic gap by means of a chemical process. The action of the chemical transmitted modifies the incoming signal (by scaling the frequency of the signals that are received) in a manner similar to the action of the weights in an artificial neural network.

The soma, or cell body, sums the incoming signals and when sufficient input is received, the cell fires, i.e. it transmits a signal over its axon to other cells. A generic biological neuron is illustrated in figure 3.3 together with axons from two other neurons (from which the illustrated neuron could receive signals) and dendrites for two other neurons (to which the original neuron would send signals).

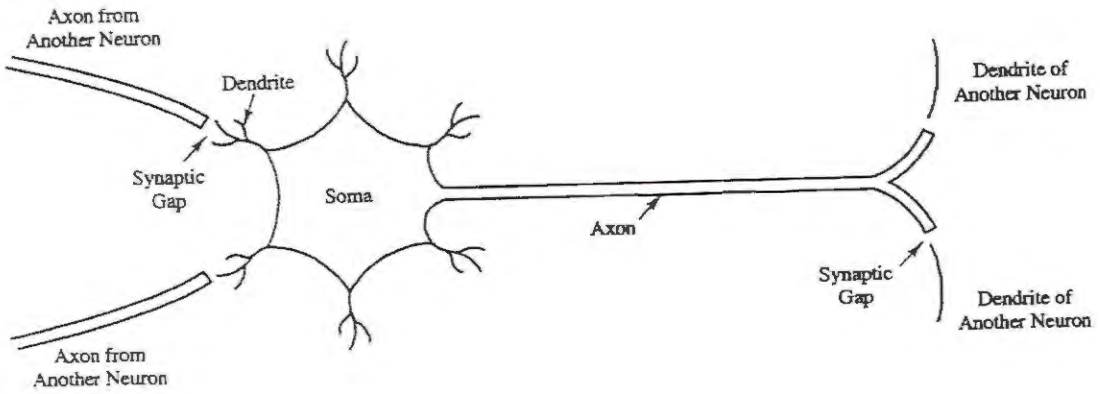


Figure 3.3 – Biological neuron

Several key features of the processing elements of artificial neural networks are suggested by the properties of biological neurons:

- The processing element receives many signals;
- Signals may be modified by a weight at the receiving synapse;
- The processing element sums the weighted inputs;
- Under appropriate circumstances (sufficient input), the neuron transmits a single output; and
- The output from a particular neuron may go to many other neurons (the axon branches).

Other features of artificial neural networks that are suggested by biological neurons are

- Information processing is local;
- Memory is distributed. Long-term memory resides in the neurons' synapses or weights and short-term memory corresponds to the signals sent by the neurons;
- A synapse's strength may be modified by experience; and
- Neurotransmitters for synapses may be excitatory or inhibitory.

There is one other important characteristic that artificial neural networks share with biological neural systems – fault tolerance. Biological neural systems are fault tolerant in

two respects. First, humans are able to recognize many input signals that are somewhat different from any signal we have seen before. An example of this is our ability to recognize a person in a picture we have not seen before or to recognize a person after a long period of time. Second, humans are able to tolerate damage to the neural system itself. Humans are born with as many as 100 billion neurons – most of these are in the brain and most are not replaced when they die. In spite of the continuous loss of neurons, humans continue to learn. Even in cases of traumatic neural loss, other neurons can sometimes be trained to take over the functions of the damaged cells. In a similar manner, artificial neural networks can be designed to be insensitive to small damage to the network, and the network can be retrained in cases of significant damage e.g. loss of data and some connections.

In the next section the connections between neurons (the architecture) in a neural network is briefly explored.

3.3 Architecture

The architecture of a neural network determines its topology and how it operates. It is convenient to visualize neurons as arranged in layers where neurons in the same layer typically behave in the same manner. The arrangement of neurons into layers and the connection patterns within and between layers is called the net architecture (Fausett, 1994).

According to Fausett (1994) neural networks are classified as *single layer* or *multilayer* networks. These can be further distinguished into *feed forward* networks – networks in which the signals flow from the input units to the output units in a forward direction – and *recurrent* networks in which there are closed-loop signal paths from a unit back to itself. These concepts are referred to again in the following paragraphs.

Single Layer Neural Networks

In a single layer network, there is only one layer of connection weights. Figure 3.4 is a representation of a single layer network where it can be seen that the input units are fully

connected to output units without any input or output units connected to each other. This is also an example of a feed forward neural network as there are input units receiving signals and output units from which the response of the neural network can be read.

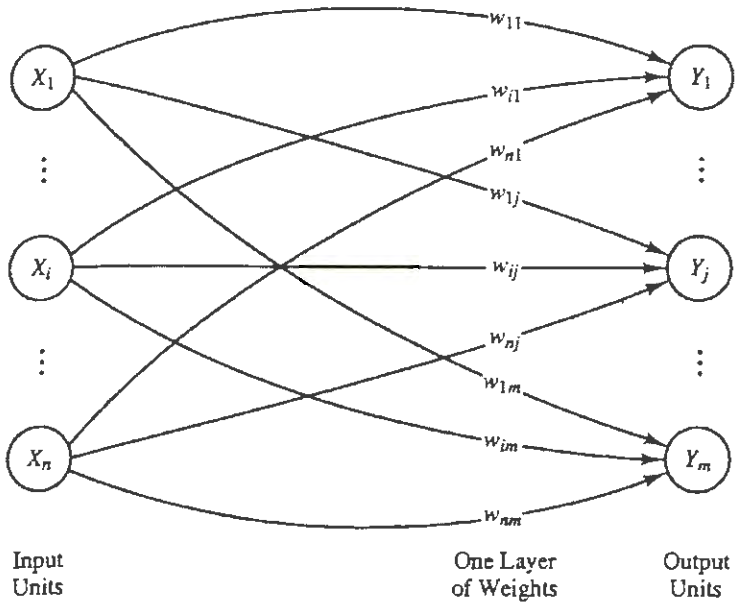


Figure 3.4 – A single layer neural network

Multilayer Neural Networks

A multilayer neural network is a network with one or more layers of nodes – called hidden units – between the input units and the output units. Usually there is a layer of weights between two adjacent levels of units (input, hidden or output). These types of neural networks are also examples of feed forward networks and can solve more complicated problems than the single layer networks. Figure 3.5 gives an illustration of multilayer neural network architecture.

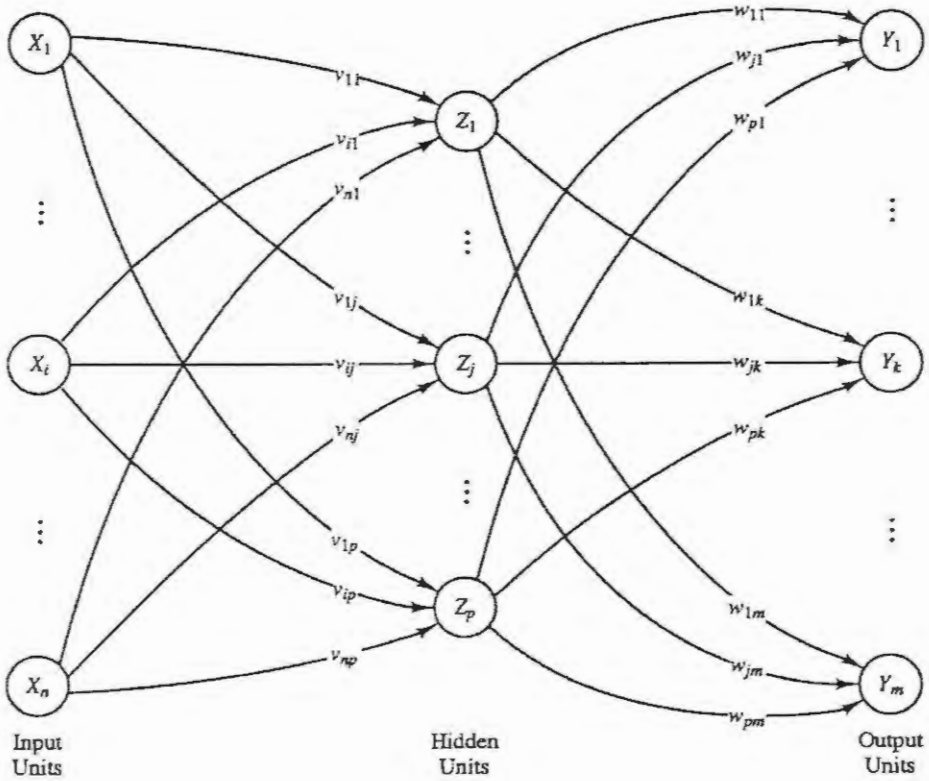


Figure 3.5 – A multilayer neural network

Competitive Layer Neural Networks

A competitive layer forms part of a large number of neural networks. They can have signals traveling in both directions by introducing loops in the network and is known as recurrent networks. An example of the architecture for a competitive layer is given in figure 3.6.

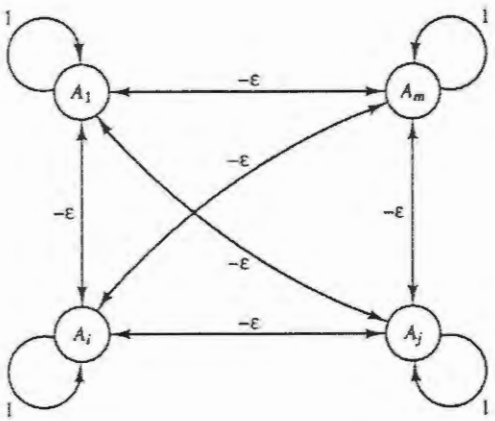


Figure 3.6 – Competitive layer

3.4 Training and Learning

The learning function takes place within a neural network's ability to change the weights and allow a neuron to modify its activity in response to its input (Awad, 1996). There are two main types of training called supervised and unsupervised training. These two types of training are summarized below according to Fausett (1994).

Supervised training

Supervised training is accomplished by presenting a sequence of training vectors, or patterns, each with an associated target output vector. The weights are then adjusted according to a learning algorithm. Examples of problems which can be solved through supervised training include pattern classification i.e. to classify an input vector as either belonging to or not belonging to a given category, and pattern association where the desired output is a pattern and not just a simple yes (belong to) or no (does not belong to). Multilayer neural networks can also be trained to perform a non linear mapping from an n-dimensional space of input vectors (n-tuples) to an m-dimensional output space i.e. the output vectors are m-tuples.

Unsupervised training

Unsupervised training or self-organizing neural networks group similar input vectors together without the use of training data to specify what a typical member of each group looks like or to which group each vector belongs. A sequence of input vectors is provided, but no target vectors are specified. The neural network modifies the weights so that the most similar input vectors are assigned to the same output (or cluster) or unit. The neural network will then produce a representative vector for each cluster formed. The so called Kohonen self-organizing maps is an example of unsupervised training.

There are also other types of training that will not be discussed here, e.g. fixed weight neural networks whose weights are fixed without an iterative training process – see Fausett (1994) for details.

Training algorithm

No overview of neural networks is complete without at least a summarized outline of one of the most popular training methods called backpropagation. Backpropagation, also called the generalized delta rule, is a simple gradient descent method to minimize the total squared error of the output computed by the neural network (Fausett, 1994). It consists of two passes (Awad, 1996)

- In the *forward pass*, the input pattern is applied to the network and allows the resulting activity to spread through the network to the output layer. This output is usually wrong initially.
- In the backward pass, the difference between the actual and the desired output generates an error signal that is propagated back through the network to teach it to do better – to produce a result closer to the desired output.

For completeness sake the backpropagation algorithm, quoted from Fausett (1994), is given below.

Variables used in the algorithm are:

$\mathbf{x}$	Input training vector: $\mathbf{x} = (x_1, \ldots, x_i, \ldots, x_n)$
$\mathbf{t}$	Output target vector: $\mathbf{t} = (t_1, \ldots, t_k, \ldots, t_m)$
δ_k	Portion of error correction weight adjustment for w_{jk} that is due to an error at output unit Y_k; also, the information about the error at unit Y_k that is propagated back to the hidden units that feed into unit Y_k.
δ_j	Portion of error correction weight adjustment for υ_{ij} that is due to the backpropagation of error information from the output layer to the hidden unit Z_j.
α	Learning rate
X_i	Input unit *i*: For an input unit, the input signal and output signal are the same, namely, x_i.
Y_{0j}	Bias on hidden unit *j*

Z_j	Hidden unit *j*: The net input to Z_j is denoted z_in_j: $z\_in_j = v_{0j} + \sum_i x_i v_{ij}$ The output signal (activation) of Z_j is denoted z_j: $z_j = f(z\_in_j)$
w_{0k}	Bias on output unit *k*
Y_k	Output unit *k*: The net input to Y_k is denoted y_in_k: $y\_in_k = w_{0k} + \sum_j z_j w_{jk}$ The output signal (activation) of Y_k is denoted y_k: $y_k = f(y\_in_k)$

The training algorithm is then given by Fausett as follows.

Step 0 Initialize weights (set to small random values)

Step 1 While stopping condition is false, do steps 2 to 9

Step 2 For each training pair, do steps 3 to 8

Feedforward:

Step 3 Each input unit (X_i, i = 1, ... ,n) receives input signal x_i and broadcasts this signal to all units in the layer above (the hidden units)

Step 4 Each hidden unit (Z_j, j = 1, ... ,p) sums its weighted input signals,

$z\_in_j = v_{0j} + \sum_i x_i v_{ij}$,

applies its activation function to compute its output signal $z_j = f(z\_in_j)$ and sends this signal to all units in the layer above (output units)

Step 5 Each output unit (Y_k, k = 1, ... ,m) sums its weighted input signals,

$y\_in_k = w_{0k} + \sum_j z_j w_{jk}$

and applies its activation function to compute its output signal $y_k = f(y\_in_k)$.

Backpropagation of error:

Step 6 Each output unit (Y_k, k = 1, …. ,m) receives a target pattern corresponding to the input training pattern, computes its error information term

$\delta_k = (t_k - y_k) f'(y\_in_k)$,

calculates its weight correction term (used to update w_{jk} later),

$\Delta w_{jk} = \alpha \delta_k z_j$, calculates its bias correction term (used to update w_{0k} later),

$\Delta w_{ok} = \alpha \delta_k$, and sends δ_k to units in the layer below.

Step 7 Each hidden unit (Z_j, j = 1, …. ,p) sums its delta inputs (from units in the layer above),

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk},$$

multiplies by the derivative of its activation function to calculate its error information term

$\delta_j = \delta\_in_j f'(z\_in_j)$,

calculates its weight correction term (used to update υ_{ij} later),

$\Delta \upsilon_{ij} = \alpha \delta_j x_i$, and calculates its bias correction term (used to update υ_{0j} later), $\Delta \upsilon_{0j} = \alpha \delta_j$

Update weights and biases:

Step 8 Each output unit (Y_k, k = 1, …. ,m) updates its bias and weights (j = 0, …. ,p):

w_{jk} (new) = w_{jk} (old) + Δw_{jk}

Each hidden unit (Z_j, j = 1, …. ,p) updates its bias and weights (i = 0, …. ,n):

υ_{ij} (new) = υ_{ij} (old) + $\Delta \upsilon_{ij}$

Step 9

Test stopping condition

Any activation function can be used in the algorithm. The next section considered some of the more important activation functions that are also suitable for use.

Simple illustrations of how the training program was implemented to solve the XOR problem as well as to compress data can be found in Fausett (1994).

3.5 Some common activation functions

As indicated in section 3.2.1, the basic operation of an artificial neuron involves summing its weighted input and then applying an output or activation function. There are different forms of activation functions (linear and non-linear) and in this section a brief introduction of only the logistic sigmoid function is given. The discussion is based on the function's description as found in Fausett (1994). For a more extensive discussion of the choice of activation functions and different forms of sigmoid functions, Fausett (1994) can be consulted.

Sigmoid functions are especially useful in neural networks that are trained by the backpropagation algorithm where the relationship between the value of the function at a point, and the value of the derivative at that point, is used during training. The most common types are the logistic function and the hyperbolic tangent function.

The logistic function with a range from 0 to 1, is often used as the activation function for neural networks in which the desired output values either are binary or are in the interval between 0 and 1. This is called the binary or logistic sigmoid and is defined as

$$f(x) = \frac{1}{1 + \exp(-\sigma x)} \tag{3.3}$$

with

$$f'(x) = \sigma f(x)[1 - f(x)]. \tag{3.4}$$

The σ is a steepness parameter and in figure 3.7 the function is illustrated for $\sigma = 3$ and $\sigma = 5$.

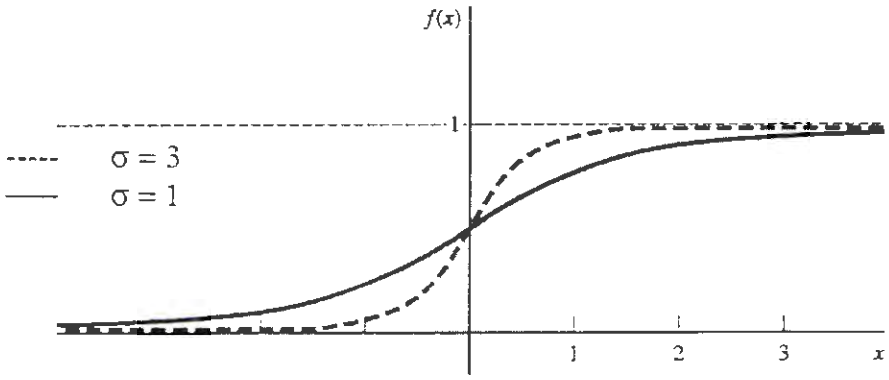


Figure 3.7 – Binary sigmoid. Steepness parameters of 3 and 1

The logistic sigmoid function can be scaled to have any range of values (Fausett, 1994) and the most common range is from -1 to 1. This is then called a bipolar sigmoid and is illustrated in figure 3.8 for $\sigma = 1$. The bipolar sigmoid function is defined as

$$\begin{aligned} g(x) &= 2f(x) - 1 \\ &= \frac{2}{1 + \exp(-\sigma x)} - 1 \\ &= \frac{1 - \exp(-\sigma x)}{1 + \exp(-\sigma x)} \end{aligned} \tag{3.5}$$

$$\text{and } g'(x) = \frac{\sigma}{2}[1 + g(x)][1 - g(x)] \tag{3.6}$$

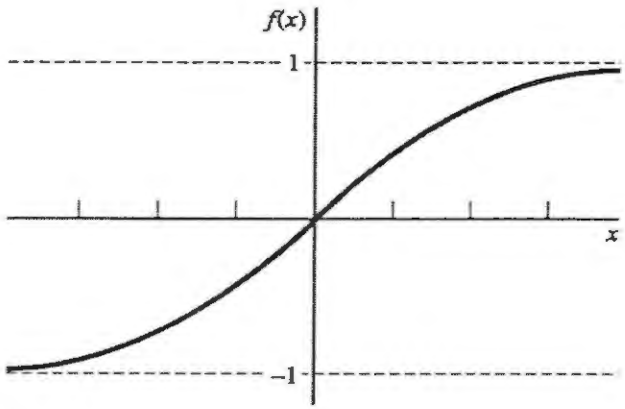


Figure 3.8 – Bipolar sigmoid

When the desired range of output values is between -1 and 1, the hyperbolic tangent function, which is closely related to the bipolar sigmoid, can also be used. The correspondence between the two can be illustrated as follows (for $\sigma = 1$).

The bipolar sigmoid is given by

$$g(x) = \frac{1 - \exp(-x)}{1 + \exp(-x)} \tag{3.7}$$

and the hyperbolic tangent by

$$h(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$

$$= \frac{1 - \exp(-2x)}{1 + \exp(-2x)} \tag{3.8}$$

and $h'(x) = [1+h(x)]\,[1-h(x)]$. (3.9)

This section is concluded by a short list of the most common activation functions used (Payne, 2001).

- Hard limit function

  $f(x) = 0, x < 0$ and

  $f(x) = 1, x \geq 0$.

- Symmetrical hard limit

  $f(x) = -1, x < 0$ and

  $f(x) = 1, x \geq 0$.

- Linear

  $f(x) = x$.

- Saturating linear

  $f(x) = 0, x < 0,$

  $f(x) = x, 0 \leq x \leq 1$ and

  $f(x) = 1, x > 1$.

- Symmetric saturating linear

  $f(x) = -1, x < -1,$

  $f(x) = x, -1 \leq x \leq 1$ and

  $f(x) = 1, x > 1$.

- Log-sigmoid

  $$f(x) = \frac{1}{1+\exp(-x)}$$

- Hyperbolic tangent sigmoid

  $$f(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$$

- Positive linear

  $f(x) = 0, x < 0$ and

  $f(x) = x, x \geq 0$.

- Competitive

  $f(x) = 1$, neuron with max x, and

  $f(x) = 0$, all other neurons.

3.6 Conclusion

Neural network models play a significant role in this study and chapter 3 presented a brief overview of such models in general. The models were explained by concepts such as artificial and biological neural networks. The different architectures of neural networks were also presented as well as an explanation on how training and learning takes place within a neural network. The chapter was concluded with a list of common activation function used in neural networks.

CHAPTER 4

NON SEASONAL BOX-JENKINS APPROACH TO MODELING ARIMA PROCESSES

4.1 Introduction

The objective of chapter 4 is to present a brief overview of a forecast technique that is often referred to as a Box-Jenkins analysis or ARIMA analysis. This is necessary as ARIMA models are often used in combination with other models, such as neural network models, to construct hybrid forecasts. The acronym ARIMA stands for Auto-Regressive Integrated Moving Average and will be more formally defined in the following section.

The Box-Jenkins approach to modeling ARIMA processes was described in 1970 by the two statisticians Box and Jenkins (Box and Jenkins, 1970). It is an iterative process encompassing four stages of development namely model identification, estimation, diagnostic checking and forecasting.

- Model identification uses various graphs of historical data based on transformed and differenced data to try and identify an appropriate Box-Jenkins model.
- Estimation means that historical data are used to find values of the model coefficients which will provide the best fit to the data.
- Diagnostic checking involves testing the assumptions of the model to check the adequacy of the model.
- Forecasting takes place when the final model is obtained and used to forecast future time series values.

To build a proper ARIMA or Box-Jenkins model is often not a straightforward process and requires good judgment and a lot of experience (Pankratz, 1983). Many text books give comprehensive explanations and discussions on the Box-Jenkins process and for a full exposition of details, text books such as Makridakis *et al* (1983), Pankratz (1983) and Bowerman *et al* (2005) can be consulted.

The rest of the chapter discusses the four stages in the Box-Jenkins approach.

4.2 Model identification

According to Zhang (2003), when using an ARIMA model, the future value of a variable is assumed to be a linear function of several past observations and random errors and it has the following form

$$y_t = \theta_0 + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \ldots - \theta_q \varepsilon_{t-q} \qquad (4.1)$$

where y_t and ε_t are the actual value and random error at time period t respectively and ϕ_i (i = 1, …, p) and θ_j (j = 0,1, …, q) are model parameters. p and q are integers and often referred to as orders of the model. Random errors, ε_t, are assumed to be independently and identically distributed with a mean of zero and a constant variance of σ^2. (Zhang, 2003).

The model given above entails several important special cases of the ARIMA family of models. If q = 0, the model becomes an autoregressive (AR) model of order p. When p = 0, the model reduces to a moving average (MA) model of order q. One of the central tasks of the ARIMA model building process then is to determine the appropriate model order (p, q).

To tentatively identify a Box-Jenkins model, it is necessary to determine whether the time series that will be forecasted is stationary. A stationary time series has the property that its statistical characteristics such as the mean and the autocorrelation structure are constant over time (Zhang, 2003). Bowerman *et al* (2005) stated that a plot of time series values is useful in helping to determine stationarity. If n values of a time series $y_1, y_2, \ldots, y_n$ were observed, a plot of these values against time can be used to determine whether the time series is stationary. If the n values seem to fluctuate with constant variation around a constant mean, μ, then it is reasonable to believe that the time series is stationary. If the n values do not fluctuate around a constant mean or do not fluctuate

with constant variation, then it is reasonable to believe that the time series is non-stationary. The time series can then be transformed into a stationary time series by taking the first differences of the non-stationary values.

The first differences of the time series values $y_1, y_2, ..., y_n$ are then given as

$$z_t = y_t - y_{t-1} \text{ where t = 2, ..., n.} \tag{4.2}$$

In some cases, where the first differences of the original time series values

$$z_2 = y_2 - y_1$$

$$z_3 = y_3 - y_2$$

.

.

.

$$z_n = y_n - y_{n-1}$$

are non-stationary, a stationary time series can be produced by taking the second differences of the original time series values. These second differences of the time series values $y_1, y_2, ..., y_n$ are then given as

$$z_t = (y_t - y_{t-1}) - (y_{t-1} - y_{t-2})$$

$$= y_t - 2y_{t-1} + y_{t-2} \text{ where t 3, 4, ..., n.} \tag{4.3}$$

The values $z_b, z_{b+1}, ..., z_n$ obtained from the first, or the second, differences are often called the *working series*.

Once the working series $z_b, z_{b+1}, ..., z_n$, which may be the original time series values, are identified, the next step of model identification can take place. This step involves the investigation of the behavior of the sample autocorrelation function (SAC) and the sample partial autocorrelation function (SPAC) for the values of the stationary time series $z_b, z_{b+1}, ..., z_n$.

The sample autocorrelation function is described as follows by Bowerman *et al* (2005). Consider the working series of time series values $z_b, z_{b+1}, ..., z_n$. The sample autocorrelation at lag k, denoted by r_k, is

$$r_k = \frac{\sum_{t=b}^{n-k}(z_t - \overline{z})(z_{t+k} - \overline{z})}{\sum_{t=b}^{n}(z_t - \overline{z})^2} \tag{4.4}$$

where

$$\overline{z} = \frac{\sum_{t=b}^{n} z_t}{(n-b+1)} \tag{4.5}$$

This quantity measures the linear relationship between time series observations separated by a lag of *k* time units. The value of r_k will always be between -1 and +1 (Bowerman *et al*, 2005) where a value close to 1 indicates that observations separated by a lag of *k* time units have a strong tendency to move together in a linear fashion with a positive slope. Values close to -1 indicates that observations separated by a lag of *k* time units have a strong tendency to move together in a linear fashion with a negative slope.

The sample autocorrelation function (SAC) is then presented as a listing or graph of the sample autocorrelations at lags k = 1, 2, ...

In order to use the Box-Jenkins methodology, it is necessary to examine the *behavior* of the SAC. See for example Bowerman *et al* (2005) and Pankratz (1983) for complete discussions and illustrations on the possible variety of behaviors that a SAC can display. In short, the SAC for a non-seasonal time series can *cut off* – this means that a spike at lag *k* exists in the SAC if r_k, the sample autocorrelation at lag *k*, is statistically large. One can also say that the SAC cuts off after lag *k* if there are no spikes at lags greater than *k* in the SAC. Secondly, the SAC *dies down* if the function does not cut off but rather decreases in a steady fashion – the SAC may die down fairly quickly or extremely slowly. Bowerman *et al* (2005) can be consulted for examples and detailed illustrations.

The behavior of the SAC can also be related to stationarity and Bowerman *et al* (2005) stated that in general it can be shown that for non-seasonal data the following is true.

- If the SAC of the time series values $z_b, z_{b+1}, ..., z_n$ either *cuts off fairly quickly* or *dies down fairly quickly*, then the time series values should be considered *stationary*.
- If the SAC of the time series values $z_b, z_{b+1}, ..., z_n$ *dies down extremely slowly*, then the time series values should be considered *non-stationary*, and data transformation (differencing) is necessary.

In addition to the SAC, the sample partial autocorrelation function (SPAC) is also used to assist in model identification. According to Pankratz (1983) the SPAC is broadly similar to the SAC – it is also a graphical representation of the statistical relationship between sets of ordered pairs (Z_b, Z_{b+k}) drawn from a single time series. It is used as a guide, along with the estimated SAC, in choosing one or more ARIMA models that might fit the available data points. The SPAC is defined by Bowerman *et al* (2005) as follows.

The sample partial autocorrelation at lag *k* is

$$r_{kk} = r_1 \quad \text{if k = 1 and}$$

$$r_{kk} = \frac{r_k - \sum_{j=1}^{k-1} r_{k-1,j} r_{k-j}}{1 - \sum_{j=1}^{k-1} r_{k-1,j} r_j} \quad \text{if k = 2, 3, ...} \tag{4.6}$$

where

$$r_{kj} = r_{k-1,j} - r_{kk} r_{k-1,k-j} \quad \text{for j = 1, 2, ..., k-1.} \tag{4.7}$$

As in the case of the SAC, it is necessary to examine the behavior of the SPAC in order to employ the Box-Jenkins methodology. Bowerman *et al* (2005) explains it as follows.

- The SPAC for a non-seasonal time series can *cut off*, which means that if a spike at lag *k* exists, the r_{kk} (sample partial autocorrelation at lag *k*) is statistically large. To judge whether a spike at lag *k* exists in the SPAC, the t-statistic related to r_{kk} is evaluated. If the absolute value of the t-statistic is greater than 2, then a spike exists. The t-statistic is given by

$$t_{r_{kk}} = \frac{r_{kk}}{s_{r_{kk}}} \tag{4.8}$$

with $s_{r_{kk}}$ the standard error of r_{kk} and defined as

$$s_{r_{kk}} = \frac{1}{\sqrt{n-b+1}} \tag{4.9}$$

- The SPAC *dies down* if the function does not cut off but rather decreases in steady fashion.

As stated in the beginning of this section, one of the central tasks in model identification is to determine values for p and q which are referred to as the orders of the model. The SAC and SPAC, as discussed here, assist in determining these values and in the table below is a summary (Bowerman *et al*, 2005) of the general non-seasonal models and their orders as indicated by the SAC and SPAC.

Model	**SAC**	**SPAC**
Moving average of order q $z_t = \delta + a_1 - \theta_1 a_{t-1} - \theta_2 a_{t-2} - ... - \theta_q a_{t-q}$	Cuts of after lag q	Dies down
Autoregressive of order p $z_t = \delta + \phi_1 z_{t-1} + \phi_2 z_{t-2} + ... + \phi_p z_{t-p} + a_t$	Dies down	Cuts off after lag p
Mixed autoregressive moving average of order (p, q) $z_t = \delta + \phi_1 z_{t-1} + \phi_2 z_{t-2} + ... + \phi_p z_{t-p} + a_1 - \theta_1 a_{t-1} - \theta_2 a_{t-2} - ... - \theta_q a_{t-q}$	Dies down	Dies down

Most statistical software packages can perform SAC and SPAC analyses and for completeness sake this section is concluded by an example of SAC and SPAC output for first differencing ($z_t = y_t - y_{t-1}$) generated by the SAS code below. The data used in the example was taken from Bowerman *et al* (2005) and represents weekly sales figures of paper towels. The complete data set is presented in Appendix A.

```
proc arima data = work.datafile;
        identify var = y(1) nlag 14;
run;
```

SAS output of the SAC for the first differences of paper towel sales (see appendix A for data set)

The SAS System

The ARIMA Procedure

Name of Variable = y

Period(s) of Differencing	1
Mean of Working Series	0.005423
Standard Deviation	1.099416
Number of Observations	119
Observation(s) eliminated by differencing	1

Autocorrelations

Lag	Covariance	Correlation	-1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1	Std Error
0	1.208715	1.00000	|********************|	0
1	0.370658	0.30665	|****** |	0.091670
2	-0.078249	-.06474	| * |	0.099919
3	-0.086619	-.07166	| * |	0.100271
4	0.126391	0.10457	|** |	0.100700
5	0.101691	0.08413	|** |	0.101609
6	0.027608	0.02284	| |	0.102192
7	-0.160292	-.13261	|*** |	0.102235
8	-0.143891	-.11904	| ** |	0.103671
9	-0.210121	-.17384	|*** |	0.104813
10	-0.142910	-.11823	| ** |	0.107209
11	-0.062396	-.05162	| * |	0.108299
12	0.025252	0.02089	| |	0.108505
13	0.049984	0.04135	|* |	0.108539
14	0.023417	0.01937	| |	0.108672

"." marks two standard errors

Inverse Autocorrelations

Lag	Correlation	-1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1
1	-0.32950	|******* |
2	0.15054	| |*** |
3	0.02394	| | |
4	-0.13848	| ***| |
5	0.02768	| |* |
6	-0.11600	| | **| |
7	0.15425	| |*** |
8	-0.04661	| | *| |
9	0.13497	| |*** |
10	0.03197	| |* |
11	-0.00016	| | |
12	0.01014	| | |
13	-0.05308	| | *| |
14	-0.00804	| | |

SAS output of the SPAC for the first differences of paper towel sales (see appendix A for data set)

Partial Autocorrelations

Lag	Correlation	-1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1
1	0.30665	| . |****** |
2	-0.17525	| **** | . |
3	0.00616	| . | . |
4	0.13335	| . |***. |
5	-0.00953	| . | . |
6	0.01992	| . | . |
7	-0.13965	| .*** | . |
8	-0.04076	| . *| . |
9	-0.17756	| **** | . |
10	-0.05270	| . *| . |
11	-0.01092	| . | . |
12	0.03207	| . |* . |
13	0.07218	| . |* . |
14	0.00992	| . | . |

Autocorrelation Check for White Noise

To Lag	Chi-Square	DF	Pr > ChiSq	Autocorrelations					
6	14.96	6	0.0206	0.307	-0.065	-0.072	0.105	0.084	0.023
12	25.27	12	0.0136	-0.133	-0.119	-0.174	-0.118	-0.052	0.021

Looking at the SAC one can see that the SAC has a spike at lag 1 (the last asterisk is beyond the corresponding two-standard deviation dot) Since there are no spikes in the SAC after lag 1, it may be concluded that the SAC cuts off after lag 1. It can therefore be assumed that the first differences (produced by the transformation $z_t = y_t - y_{t-1}$) are stationary. The SPAC also has a spike at lag 1. There are no other spikes and it can be concluded that it cuts off after lag 1 – however, since the partial autocorrelations at lags 2 and 4 are fairly large, the cut off is not very abrupt and one may also conclude that the SPAC dies down. Based on this analysis, the moving average model of order q = 1 represented by $z_t = a_t - \theta_1 a_{t-1}$ where $z_t = y_t - y_{t-1}$ is the identified model.

4.3 Estimation

The estimation stage implies that precise estimates of the coefficients, of the model chosen in the identification step, be obtained. For example, if the first-order autoregressive model

$$z_t = \delta + \phi_1 z_{t-1} + a_t \qquad (4.10)$$

was chosen, the model will be fitted to the available data series to get estimates of δ and ϕ_1.

There are fundamentally two ways of getting estimates for the parameters (Makridakis *et al*, 1983).

- *Trial and error.* Examine many different values and chose that value (or set of values) that minimizes the sum of squared residuals.
- *Iterative improvement.* Choose a preliminary estimate and let statistical software refine the estimate iteratively.

Estimation is usually carried out using the second option of iterative improvement through the use of a computer program. The main approaches used in statistical software programs for this iterative process are quite complex and is based on non linear least squares (NLS) and maximum likelihood estimations. It is beyond the scope of this chapter to deal in technical detail with the two approaches and this section will therefore be concluded with an example, performed in SAS, to illustrate the parameter estimation process. Complete technical discussions on the different methods may be found in Makridakis *et al* (1983), Pankratz (1983) and Bowerman *et al* (2005).

The following example is based on the same data set used in section 4.2 (Bowerman *et al*, 2005) and shows the output of the parameter estimation process for the model identified in the previous stage. The output is based on the following SAS code.

```
Proc arima data = work.datafile;
        identify var = y(1) nlag 14;
        estimate q=(1);
run;
```

SAS output of least squares estimation of θ_1 in the model $z_t = a_t - \theta_1 a_{t-1}$ where $z_t = y_t - y_{t-1}$

The SAS System

The ARIMA Procedure

Preliminary Estimation

Initial Moving Average Estimates

	Estimate
R	-0.30665

White Noise Variance Est 1.104821

Conditional Least Squares Estimation

Iteration	SSE	MA1,1	Lambda	R Crit
0	127.83	-0.30665	0.00001	1
1	127.49	-0.36024	1E-6	0.055095
2	127.48	-0.35240	1E-7	0.008393
3	127.48	-0.35362	1E-8	0.001299
4	127.48	-0.35343	1E-9	0.000202

ARIMA Estimation Optimization Summary

Estimation Method	Conditional Least Squares
Parameters Estimated	1
Termination Criteria	Maximum Relative Change in Estimates
Iteration Stopping Value	0.001
Criteria Value	0.000537
Alternate Criteria	Relative Change in Objective Function
Alternate Criteria Value	6.8E-8
Maximum Absolute Value of Gradient	0.027435
R-Square Change from Last Iteration	0.000202
Objective Function	Sum of Squared Residuals
Objective Function Value	127.4788
Marquardt's Lambda Coefficient	1E-9
Numerical Derivative Perturbation Delta	0.001
Iterations	4

Conditional Least Squares Estimation

Parameter	Estimate	Standard Error	t Value	Approx Pr > |t|	Lag
MA1,1	-0.35343	0.08650	-4.09	<.0001	1

Variance Estimate	1.080329
Std Error Estimate	1.039389
AIC	347.8977
SBC	350.6769
Number of Residuals	119

* AIC and SBC do not include log determinant.

The first part of the output shows the iterative search while the final point estimates is given in the last part of the output. It shows that SAS obtained a final point estimate of $\hat{\theta}_1$ = -0.35343 after four iterations. The t-value given in the SAS output can be used to test the hypothesis H_0: $\theta = 0$ which will give an indication of whether the parameter θ should be included in the model or not (Bowerman *et al*, 2005).

4.4 Diagnostic checking

Once a tentative model has been identified and parameters have been estimated for the model, it is necessary to perform a diagnostic checking to verify that the model is adequate. According to Makridakis *et al* (1983) there are two basic ways of doing this.

- Study the residuals – to see if any pattern remains unaccounted for.
- Study the sampling statistics of the optimum solution – to see if the model could be simplified.

A statistically adequate model is one whose random shocks (residuals) are statistically independent, meaning not autocorrelated (Pankratz, 1983). If the residuals are correlated it means that the estimated model should be reconsidered and possibly reformulated. To check the adequacy of a Box-Jenkins model therefore implies that the residuals be analyzed.

One way of performing an adequacy check, using the residuals, is to examine a statistic that determines whether the first k sample autocorrelations of the residuals, considered together, indicate adequacy (Bowerman *et al*, 2005). Two such statistics have been suggested and is summarized below.

The Box-Pierce statistic given by

$$Q = n \sum_{\ell=1}^{k} r_\ell^2(\hat{a}), \tag{4.11}$$

and the Ljung-Box statistic given by

$$Q^* = n(n+2)\sum_{\ell=1}^{k}(n-1)^{-1} r_\ell^2(\hat{a}) \tag{4.12}$$

where $n' = n - d$ with n the number of observations in the original time series and d the degree of non-seasonal differencing used to transform the original time series values into stationary time series values. $r_i^2(\hat{a})$ is the square of $r_i(\hat{a})$, the sample autocorrelation of the residuals at lag k.

Both statistics can be used to test the adequacy of a model but Bowerman *et al* (2005) suggested that Q^* is the better of the two. The Q^* statistic should be small. The larger Q^* is, the larger are the autocorrelations of the residuals and the more related the residuals are. A large value of Q^* will therefore indicates that the model is inadequate. The adequacy of the model under consideration can now be rejected by setting the probability of a type I error equal to α if and only if either of the following equivalent conditions is true.

1. Q^* is greater than $\chi^2_{(\alpha)}(K-nc)$, the point on the scale of the χ^2 distribution having *k-nc* degrees of freedom such that there is an area of α under the curve of this distribution above this point. *Nc* is the number of parameters that must be estimated.
2. The p-value is less than α, where the p-value is the area under the curve of the χ^2 distribution having *k-nc* degrees of freedom.

Most statistical software packages can perform the diagnostic checking process and will also provide the required Q^* and p-values. An example, illustrating how SAS is used to perform diagnostic checking and showing how the Ljung-Box statistic (Q^*) and its associated p-values can be found for the model identified during the first stage, is given below.

The example is based on the same data set used in the previous sections (Bowerman *et al*, 2005). The output is based on the following SAS code.

```
Proc arima data = work.datafile;
        identify var = y(1) nlag 14;
        estimate q=(1) noconstant printall plot;
run;
```

SAS output for diagnostic checking for the model $z_t = a_t - \theta_1 a_{t-1}$ where $z_t = y_t - y_{t-1}$

The SAS System

The ARIMA Procedure

Autocorrelation Check of Residuals

To Lag	Chi-Square	DF	Pr > ChiSq	Autocorrelations					
6	4.10	5	0.5345	0.006	-0.037	-0.102	0.129	0.028	0.061
12	10.40	11	0.4944	-0.143	-0.031	-0.146	-0.058	-0.041	0.025
18	17.57	17	0.4165	0.020	0.038	-0.074	-0.003	0.068	0.196
24	18.67	23	0.7199	0.004	-0.028	-0.061	0.000	-0.054	0.012

Autocorrelation Plot of Residuals

Lag	Covariance	Correlation	-1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1	Std Error
0	1.080329	1.00000	|********************|	0
1	0.0059655	0.00552	| . | . |	0.091670
2	-0.039778	-.03682	| . *| . |	0.091673
3	-0.109978	-.10180	| . **| . |	0.091797
4	0.138997	0.12866	| . |***. |	0.092741
5	0.030006	0.02778	| . |* . |	0.094229
6	0.066126	0.06121	| . |* . |	0.094297
7	-0.154399	-.14292	| .***| . |	0.094631
8	-0.033148	-.03068	| . *| . |	0.096428
9	-0.157783	-.14605	| .***| . |	0.096510
10	-0.063136	-.05844	| . *| . |	0.098349
11	-0.044091	-.04081	| . *| . |	0.098641
12	0.027341	0.02531	| . |* . |	0.098783
13	0.021645	0.02004	| . | . |	0.098837
14	0.041283	0.03814	| . |* . |	0.098871

"." marks two standard errors

Partial Autocorrelations

Lag	Correlation	-1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1
1	0.00552	| . | . |
2	-0.03685	| . *| . |
3	-0.10153	| . **| . |
4	0.12960	| . |***. |
5	0.01868	| . | . |
6	0.06077	| . |* . |
7	-0.12057	| . **| . |
8	-0.03638	| . *| . |
9	-0.15573	| .***| . |
10	-0.10387	| . **| . |
11	-0.03157	| . *| . |
12	0.00388	| . | . |
13	0.06414	| . |* . |
14	0.05522	| . |* . |

It can be seen from the example that SAS calculates the Ljung-Box statistic Q^* and its associated p-value for K equal to 6, 12, 18 and 24. According to the standard χ^2-table

that gives values of $\chi^2_\gamma(df)$ (See for example Bowerman *et al*, 2005), if α, the probability of a type I error, is equal to 0.05 then the rejection point would be

$$\chi^2_{(\alpha)}(K-nc) = \chi^2_{(0.05)}(6-1) = 11.0705 \tag{4.13}$$

Since $Q^* = 4.10 < 11.0705 = \chi^2_{[0.05]}(5)$, the adequacy of the model cannot be rejected.

The p-value is the area under the curve of the χ^2 distribution having *K-nc* = 5 degrees of freedom to the right of $Q^* = 4.10$. The p-value on the SAS output is 0.5345 and since 0.5345 > 0.05, the adequacy of the model cannot be rejected.

4.5 Forecasting

The last stage and final objective of any Box-Jenkins model is to forecast future values of a time series.

In order to compute point forecasts (single numerical values) for the model

$$y_t = y_{t-1} + a_t - \theta a_{t-1} \tag{4.14}$$

the estimated model

$$\hat{y}_t = y_{t-1} + \hat{a}_t - \hat{\theta}\hat{a}_{t-1} \tag{4.15}$$

is used.

In this case the point prediction $\hat{a}_t$ of a_t is zero. The point prediction $\hat{a}_{t-1}$ of a_{t-1} is the *(t-1)*st residual $(y_{t-1} - \hat{y}_{t-1})$ if it is possible to calculate $\hat{y}_{t-1}$ and is zero if $\hat{y}_{t-1}$ cannot be calculated. To find the point forecasts, insert the estimates of parameters (e.g. $\hat{\theta}$) obtained during the previous stages, the appropriate values for past observations and assign the expected value of zero to a_t.

It can also be shown (Bowerman *et al*, 2005) that a 100(1-α)% prediction interval, calculated at time origin n for the time series value in time period $n+\tau$ is

$$[\hat{y}_{n+\tau}(n) \pm t_{\alpha/2}^{(n-n_p)} SE_{n+\tau}(n)]$$

$SE_{n+\tau}(n)$ is called the standard error of the forecast error and depends on the standard error

$$S = \sqrt{\frac{SSE}{n-n_p}} \tag{4.16}$$

Almost all statistical software packages will provide the time series forecasting values and below is an example of the forecasting values produced by SAS for the model identified in stage one. The example is based on the same data set used in the previous sections (Bowerman *et al*, 2005). The output is based on the following SAS code.

```
Proc arima data = work.datafile;
        identify var = y(1) nlag 14;
        forecast lead = 10;
run;
```

SAS output of the point forecasts and 95% prediction intervals given by the model $z_t = a_t - \theta_1 a_{t-1}$ where $z_t = y_t - y_{t-1}$

The SAS System
The ARIMA Procedure

Forecasts for variable y

Obs	Forecast	Std Error	95% Confidence Limits	
121	15.8889	1.0394	13.8518	17.9261
122	15.8889	1.7491	12.4608	19.3170
123	15.8889	2.2446	11.4896	20.2882
124	15.8889	2.6490	10.6970	21.0808
125	15.8889	2.9993	10.0103	21.7675
126	15.8889	3.3128	9.3959	22.3820
127	15.8889	3.5991	8.8347	22.9431
128	15.8889	3.8643	8.3151	23.4628
129	15.8889	4.1124	7.8288	23.9490
130	15.8889	4.3463	7.3703	24.4076

The output presents the SAS results of the point forecasts, the values of $SE_{n+\tau}(n)$, and the 95% prediction intervals when forecasting at time origin 120 the values for time periods 121 through 130.

4.6 Conclusion

Chapter 4 presented a brief overview of the Box-Jenkins modeling approach and covered the four different stages viz.

- Model identification;
- Estimation of model parameters;
- Diagnostic checking; and
- Forecasting

The chapter further introduces elementary concepts pertaining to the four stages and provides references where detailed examples and mathematical details can be found.

This chapter is finally concluded by a summary of the characteristics of a good Box-Jenkins model as given by Pankratz (1983) and evidenced by the discussions in the chapter.

A good model is

- Parsimonious (uses smallest number of coefficients needed to explain the available data).
- Stationary
- Has estimated coefficients of high quality
- Has uncorrelated residuals
- Fits the available data (the past) well enough to satisfy the analyst
- Forecasts the future satisfactorily

The next chapter, chapter 5, will be devoted to a summarized review of recent literature resources describing the use of time series modeling, the use of neural networks in time series data and a combination of the two approaches.

CHAPTER 5

METHODOLOGY AND RESEARCH DESIGN

5.1 Introduction

Following the introduction and background review of fundamental principles (chapters 2 to 4) of techniques used in this study, chapter 5 will concentrate on two further aspects. First, the relevance of techniques used in the empirical study will be motivated from appropriate literature resources. A brief overview of similar work carried out using linear models, neural networks and a combination of the techniques will be given. Second, the research design and methodology used in this study will be presented. The hybrid methodology used will be explained as well as the practical workings of the empirical experiments. The next chapter will then focus on the data sets used and the results of the empirical experiments.

5.2 Related work in the literature

Time series forecasting is an important area of forecasting that attracts many researchers and practitioners. An appropriate way to start this introductory literature review would be to refer to the informative and well researched work of De Gooijer and Hyndman (2006). They have published a review that covers the past 25 years of research into time series forecasting. The two researchers give a brief overview of the main developments of recent years as well as proposals for future research. A variety of forecasting techniques, and research work associated with them, is given and include techniques such as exponential smoothing, ARIMA models, non-linear models such as neural networks, ARCH/GARCH models etc. An excellent list of references (more than 300 authors) is also provided.

The next couple of paragraphs will be used to give examples of specific applications of time series forecasting as found in the literature and that is relevant to this study.

Ghiassi, Saidane and Zimbra (2005) experimented with a dynamic neural network model where changes were made to the architecture of the neural network in an effort to improve forecasting results of different time series data. Based on their experiments they claimed that the neural

network approach is more accurate and performs significantly better than traditional models such as ARIMA models.

The work of Ediger and Akar (2007) is an example of the use of a linear model (ARIMA) that was employed to forecast the future energy demand of Turkey. The ARIMA technique was applied to time series data of each item that contributes to primary energy e.g. coal, wood, animal and plant remains, oil, natural gas, hydropower etc. They concluded that the application of the ARIMA forecasting technique delivers more reliable results than other approaches that they have previously tried when forecasting energy demand.

In line with the results from the energy demand study, Dooley and Lenihan (2005) also found that the use of the ARIMA forecasting technique provides marginally better results when forecasting metal prices. They have compared the ARIMA technique with a lagged forward price model.

In a study to compare the accuracy of up to six different methods for short-term electricity demand, Taylor, Menezes and McSharry (2006) have found that the best results were achieved with the exponential smoothing method. Based on their experiments, they conclude that simple and robust methods outperform more complex alternatives and that they require little domain knowledge to be implemented.

Cho (2003) presented a study in forecasting applied to tourism management. In an effort to predict the travel demand (i.e. number of arrivals) from different countries to Hong Kong, three forecasting techniques were investigated. The performance of two linear models, exponential smoothing and ARIMA, was compared with an artificial neural network. The study concludes that in those cases where the time series data have no obvious pattern, the neural network seems to be the best alternative for forecasting visitor arrivals.

In a more recent study, Co and Boosarawongse (2007) also compared the forecasting ability of exponential smoothing, ARIMA and neural network models. They applied these models in an

effort to forecast rice exports from Thailand. Using different measures of forecast errors (e.g. MSE and MAPE), they concluded that the neural network models performed relatively well as they were able to track the dynamic non-linear trend and seasonality in the time series.

There are also a number of studies where only neural networks and ARIMA models were compared. Examples of such studies include the prediction of the Hepatitis A virus performed by Ture and Kurt (2006). They argued that a multi-layered perceptron neural network outperformed an ARIMA model. Interesting results were presented by Mishra and Desai (2006) who also compared neural networks and ARIMA models. Their study was conducted in India and they used these models to perform a forecast of droughts. The results obtained showed that the performance of one of the neural network models, as well as the ARIMA model, decreases over a longer lead time because of the accumulation of errors between the observed and predicted values at each time step. Both the ARIMA and neural network models provided good results as long as the lead time to forecast is not too big.

In this research project the idea is to combine neural network and linear models to provide for both linear and non-linear elements that may be contained in the data. This section will therefore be concluded with a few examples from the literature where this approach was followed.

A simple combination of forecasts using the equal weights method (arithmetic mean of individual forecasts) was used by Zou et al (2007). They compared this combined approach with ARIMA and neural network models to forecast wheat prices in the Chinese market. Results reported by them are somewhat conflicting – according to them, the neural network outperforms the ARIMA model while the combined forecasting approach proofs to be an effective way of improving the forecasting performance of error measures.

Ince and Trafalis (2006) proposed a two stage forecasting model where they used an ARIMA model to select input variables and then, in the second stage, apply neural network and support vector regression models to make forecasts. This approach was used by them to forecast the exchange rate for different currencies e.g. the euro/dollar exchange rate. Their experiments have

indicated that the forecasting performance is highly dependent on the selection of inputs to be used in the final forecasting model.

Valenzuela et al (2008) described a hybrid model that is the same as Zhang's model (Zhang, 2003) but with the interesting difference of using a fuzzy expert system, driven by an evolutionary algorithm, to determine the structure of the ARIMA model. The residuals of this model are then passed on to a neural network to model the non-linear component of the time series. Although the testing of this hybrid approach was more focused on identifying the correct ARIMA model using the expert system, the authors concluded that the synergy of the hybrid model produced excellent results. The same research project, presenting a different level of detail information, is also described in Rojas et al (2008).

Following the same approach as Zhang (2003), Chelani and Devotta (2006) also proposed a hybrid model to forecast air quality as determined by the nitrogen dioxide concentration observed in Delhi, India. An ARIMA model was used to model the linear component of the data and a non-linear dynamic model (based on so-called phase spaces and attractors) was then developed to model the residuals from the ARIMA model. They found that the hybrid model outperformed the individual linear and non-linear models.

To conclude this section, it should be noted that there are, however, other researchers and research studies that have shown that a hybrid model (such as the one proposed by Zhang in 2003) does not always deliver better forecasting results. For example, Taskaya-Temizel and Casey (2005) have shown through a number of experiments that the hybrid model may underperform in certain cases. This is in line with the research results found in the empirical work of this study which are detailed in chapter 6.

5.3 Research design and methodology

In this section the hybrid methodology of combining a linear model with the results of a neural network will be discussed. Following this discussion, an overview of the approach and methodology used to perform the empirical experiments will be presented.

5.3.1 Hybrid methodology

As stated in chapter 1, the primary objective of this research study was to investigate the use of a combined linear and neural network model to determine forecasting performance. This will be done based on the work of Zhang (2003).

Zhang (2003) motivated the use of such a hybrid approach as follows.

- It is often difficult to determine whether a time series under study was generated from a linear or non-linear underlying process. This makes it difficult to choose the correct forecasting technique for the specific situation.
- Real world time series are rarely pure linear or non-linear and they often contain both linear and non-linear patterns.
- It is agreed in forecasting literature that no single method is best in every situation. Real world problems are complex in nature and a single model may not be able to capture different patterns.

The basic idea of model combination in forecasting can therefore been seen as using each model's unique features to capture different patterns in the data. A brief overview of the combination of linear models and neural networks, based on the work of Zhang (2003) will now be presented.

It is clear from the literature sources quoted in section 5.2, that both linear models and neural network models have achieved successes in their own linear and non-linear domains. It is, however, not always clear if each model on its own would be suitable in all circumstances. Using a linear model in complex non-linear problems may not be adequate. Also, the use of a neural network to model linear problems may produce mixed results. As the characteristics of data in real world problems are often unknown, it may be a good idea to consider a hybrid methodology where linear and non-linear models are combined to try and capture different aspects of the underlying patterns. Zhang (2003) explained this hybrid methodology as follows.

Consider a time series composed of a linear structure and a non-linear component. That is

$$y_t = L_t + N_t \tag{5.1}$$

where L_t denotes the linear component and N_t the non-linear component.

If we assume that these two components have to be estimated from the data, a linear model can be used to model the linear component and the residuals from the linear model will then contain the non-linear relationship. Let e_t denote the residual at time t from the linear model, then

$$e_t = y_t - \hat{L}_t \tag{5.2}$$

where $\hat{L}_t$ is the forecast value for time t from the estimated relationship.

Zhang (2003) argued that a linear model is not sufficient if there are still non-linear correlation structures left in the residuals. According to him, residual analysis is not able to detect any non-linear patterns in data and that there is no general diagnostic statistic for non-linear autocorrelation relationships. This implies that a model may still not be adequate, even if it has passed diagnostic checking, as non-linear relationships have not been appropriately modeled. If there is any significant non-linear pattern in the residuals it is an indication of the limitation of a linear model. If the residuals are now modeled using a neural network, non-linear relationships may be discovered. With n input nodes, the neural network model for the residuals will be

$$e_t = f(e_{t-1}; e_{t-2}; \ldots\ldots; e_{t-n}) + \varepsilon_t \tag{5.3}$$

where f is a non-linear function determined by the neural network and ε_t is a random number. If the forecast from the neural network is denoted by $\hat{N}_t$, then the combined forecast will be

$$\hat{y}_t = \hat{L}_t + \hat{N}_t \tag{5.4}$$

To summarize, Zhang's proposed hybrid methodology consists of two main steps. First, a linear model is used to analyze the linear part of the problem and second, a neural network model is developed to model the residuals from the linear model. This hybrid approach exploits the unique features and strengths of a linear model as well as neural network models and could be advantageous to model linear and non-linear patterns separately and then combine the forecasts to improve overall modeling and forecasting performance.

The next section, section 5.3.2, will give an overview of the detailed steps and approach that was followed in this study using Zhang's technique of combining forecast models.

5.3.2 Empirical experiment approach

In order to examine the performance of the proposed hybrid model, the individual linear and non-linear models were also applied to the data sets under consideration. To determine these two models the following approach was followed.

Determine an appropriate neural network model

The construction of a neural network architecture presents two distinct challenges to the model builder. In the first instance, an appropriate number of hidden nodes should be chosen and secondly, the number of lagged observations which is also referred to as the dimension of the input vector must be specified. The number of hidden nodes may impact the degree of over fitting of the model (a good fit to the sample data but poor generalization capability) while the input dimension plays a major role in determining the autocorrelation structure of the time series. There are no systematic rules or theory to guide the selection of these two parameters (Zhang, 2003) and it was decided to perform a series of experiments to help select values for these parameters. The experiments to do this were conducted as follows.

A SAS program was developed to perform a "grid-search" where all possible combinations of input dimensions, ranging from 1 to 20 were tested (see Appendix B for the source code of the program). At the same time, the number of hidden nodes was varied from 1 to 20. Twenty experiments (fitting of a neural network) were performed in each case. Doing it this way, a total of 8000 (20 experiments * 20 nodes * 20 input dimensions) neural network models were fitted in each case and the parameters for the one with the smallest average mean squared error was then selected for the model to be used. The following table explains the sequence of tests performed each time a neural network architecture had to be determined.

Number of lagged observations as input dimension	Number of hidden nodes to be included in architecture	Number of experiments to determine best parameters
1 lagged observation	1 hidden node	Experiment 1
		Experiment 2
		Experiment 3
		.
		.
		Experiment 20
	2 hidden nodes	Experiment 1
		.
		.
		Experiment 20
	.	
	.	
	20 hidden nodes	
2 lagged observations		
.		
.		
20 lagged observations		

For each one of the experiments. The SAS program randomly selected 90% of the data to be used for training of the neural network while the remaining 10% formed part of the validation process.

Determine an appropriate linear model

There are a variety of ways to fit a linear model to a time series data set. One popular way of doing it is to follow the well known Box-Jenkins approach of building an ARIMA model – this approach was discussed and illustrated in chapter 4. In this study however, it was decided to make use of the automatic linear model selection feature offered by the SAS software package. This means that all tests carried out on a data set during a manual process (e.g. in the Box-

Jenkins approach) to select the best model is now performed automatically by the SAS software. It is a powerful feature which uses a wide variety of linear models to select the best one – several different kinds of exponential smoothing, ARIMA and log linear models are used in the selection process. The function also offers the usual features associated with time series forecasting such as goodness-of-fit measures, predictions, evaluations of errors, graphing facilities etc.

The two abovementioned methods to choose appropriate neural network and linear models form the basis of the background work to be performed in the actual methodology which can be summarized as follows.

For each data set under review

1. Determine the most appropriate *neural network model* architecture that can be applied to the time series and perform a forecast using the identified model.
2. Determine the most appropriate *linear model* that can be applied to the time series and perform a forecast using the identified model.
   (Steps 1 and 2 are necessary to compare the performance of the individual linear and non-linear models with the proposed hybrid model)
3. Determine the most appropriate neural network model architecture that can be applied to the errors of the linear forecasts obtained in step 2 and perform a neural network forecast of the errors.
4. Construct a hybrid model by combining the linear forecasts (obtained in step 2) with the neural network forecast of errors (obtained in step 3). Perform the hybrid forecast.
5. Compare the means square error (MSE) and the mean absolute deviation (MAD) of the three forecasts (neural network, linear and hybrid model).
6. Interpretation of results.

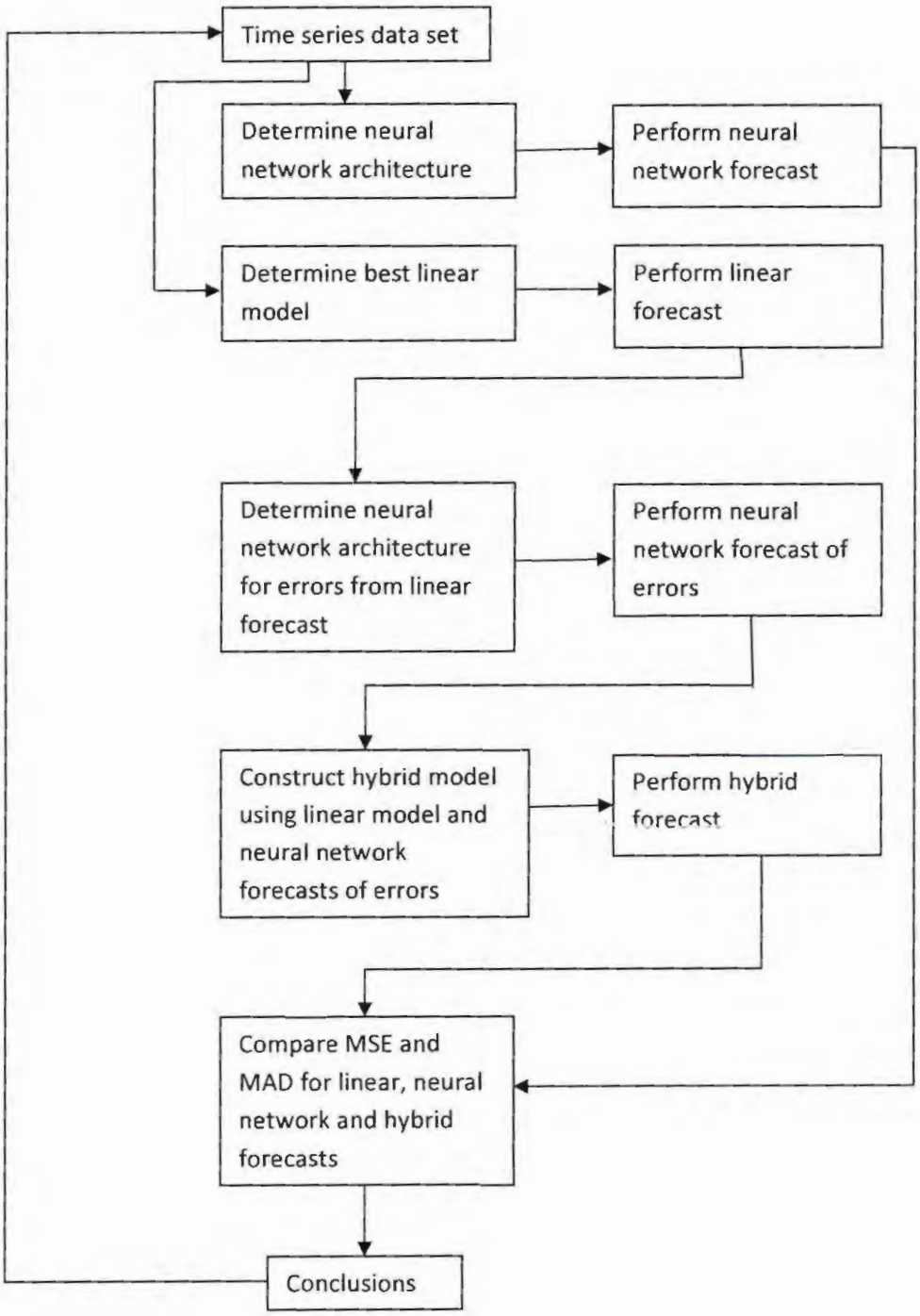
This technique was repeatedly applied for three different forecasting periods for each data set under consideration. The forecasting periods and results are detailed in chapter 6.

Schematically the 6 steps of the empirical experiment approach can be represented as depicted in figure 5.1 at the end of this chapter.

5.4 Conclusion

Chapter 5 consists of two main sections. The relevance of the techniques used in this study was firstly motivated from literature resources and, secondly, the research design and approach followed to conduct the empirical work was explained. In the next chapter the data sets used and results obtained during the practical experiments will be discussed.

Figure 5.1 – Empirical experiment approach



CHAPTER 6

EMPIRICAL RESULTS AND DISCUSSION

6.1 Introduction

The primary objective of this study was to investigate the use of a combined linear and neural network model to determine the forecasting performance of such a hybrid model. The purpose of this chapter is to report the empirical results from five different real data sets that were used to evaluate forecasts obtained by a linear, a neural network and a combination of the two techniques. A brief description of the data sets will be given followed by the modeling and forecast results. The chapter will then be concluded with a discussion of the results obtained.

6.2 Data sets

To demonstrate the results of the empirical experiments, five real world data sets were used. They were taken from different areas and have different statistical characteristics. The data is well known and it is assumed and accepted that these types of data have been widely studied in statistical and forecasting literature. However, no known generally accepted forecasting model for these data exist and using them for further evaluation is appropriate.

The five data sets chosen are

- the gold price;
- demand for electricity;
- the rand/dollar exchange rate;
- the oil price; and
- return on the money market.

The gold price time series data set

The gold price data that was considered contains the daily (working days) price in US dollars from 3 January 2006 to 28 December 2007 giving a total of 499 observations. The importance of studying the gold price is obvious – in South Africa, gold is one of the largest contributors to the country's gross domestic product and the industry also provides a significant number of employing opportunities. Figure 6.1 shows a plot of the data set.

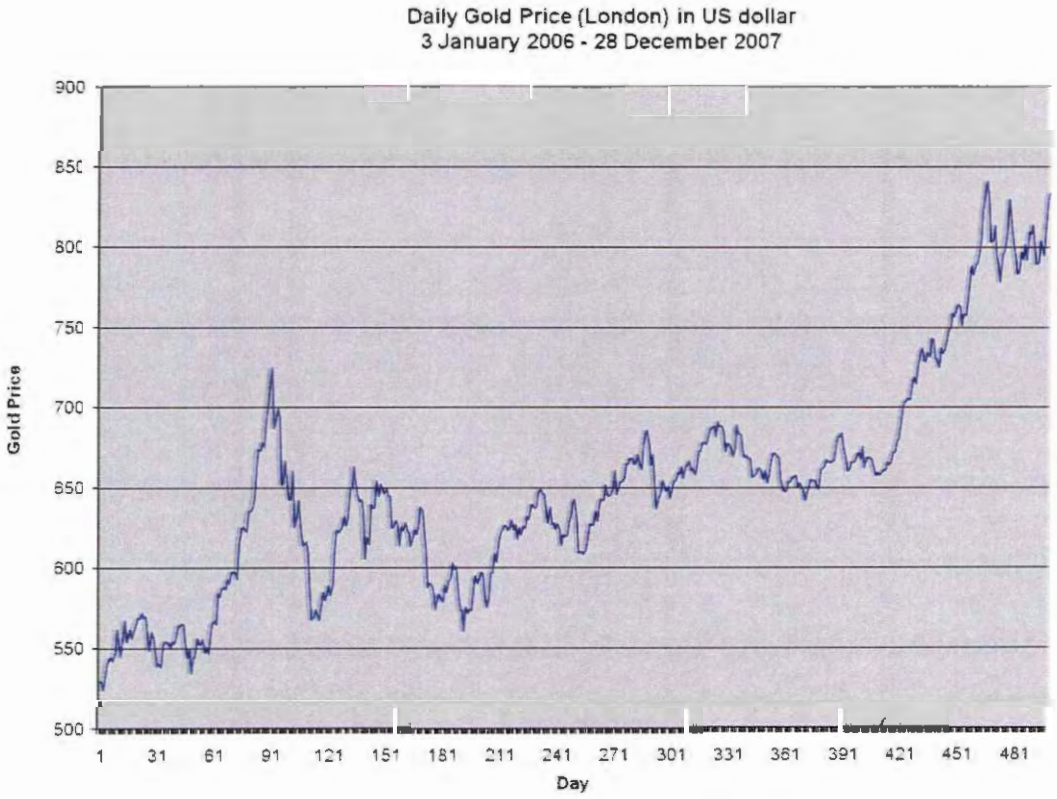


Figure 6.1 – Daily gold price (3 January 2006 – 28 December 2007)

Demand for electricity time series data set

The second data set evaluated was the demand for electricity in South Africa. The delivery of electricity is currently a sensitive issue in South Africa as the national electricity supplier apparently does not have sufficient capacity to provide in the national demand. Electricity forms the backbone of economic activity and a shortage will seriously impact the well being of the country. The time series used contains the demand (measured in mega watts) for electricity in South Africa recorded every 5 minutes during a 24 hour day (288 observations per 24 hour day). A total of 884 observations were used which represents the demand at a 5 minute interval from 24h00 on 1 July 2007 to 01h35 on 4 July 2007. A plot of the data is shown in figure 6.2.

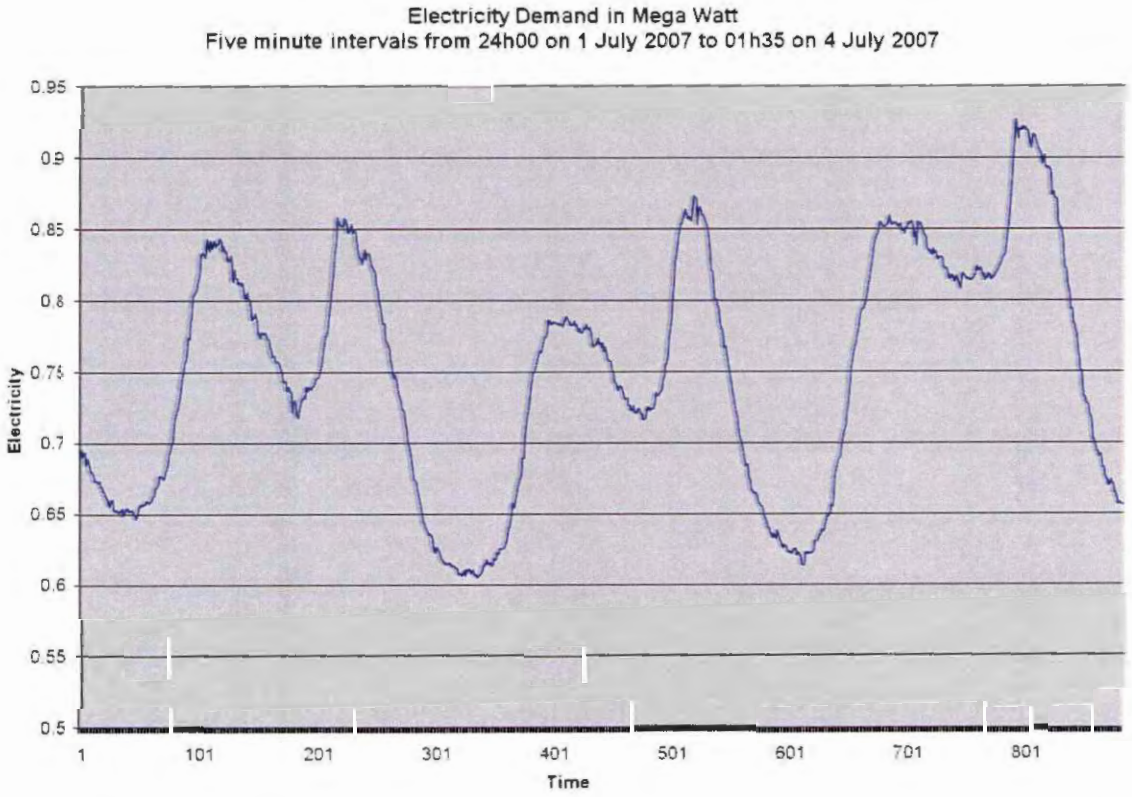


Figure 6.2 – Demand for electricity recorded every 5 minutes (1 July 2007 – 4 July 2007)

Rand/dollar exchange rate time series data set

Predicting the exchange rate is an important yet difficult task in international finance. Zhang (2003) reported that various linear and non-linear models have been developed but few of them proved to be more successful than a simple random walk model. In South Africa, the rand/dollar exchange rate is of great importance as South Africa is a developing country where import and export activities may have a huge impact on the country's balance of payments. The time series selected for this study contains the daily closing exchange rate from 6 February 2003 to 29 January 2008 giving a total of 1244 observations. Figure 6.3 shows a plot of the data.

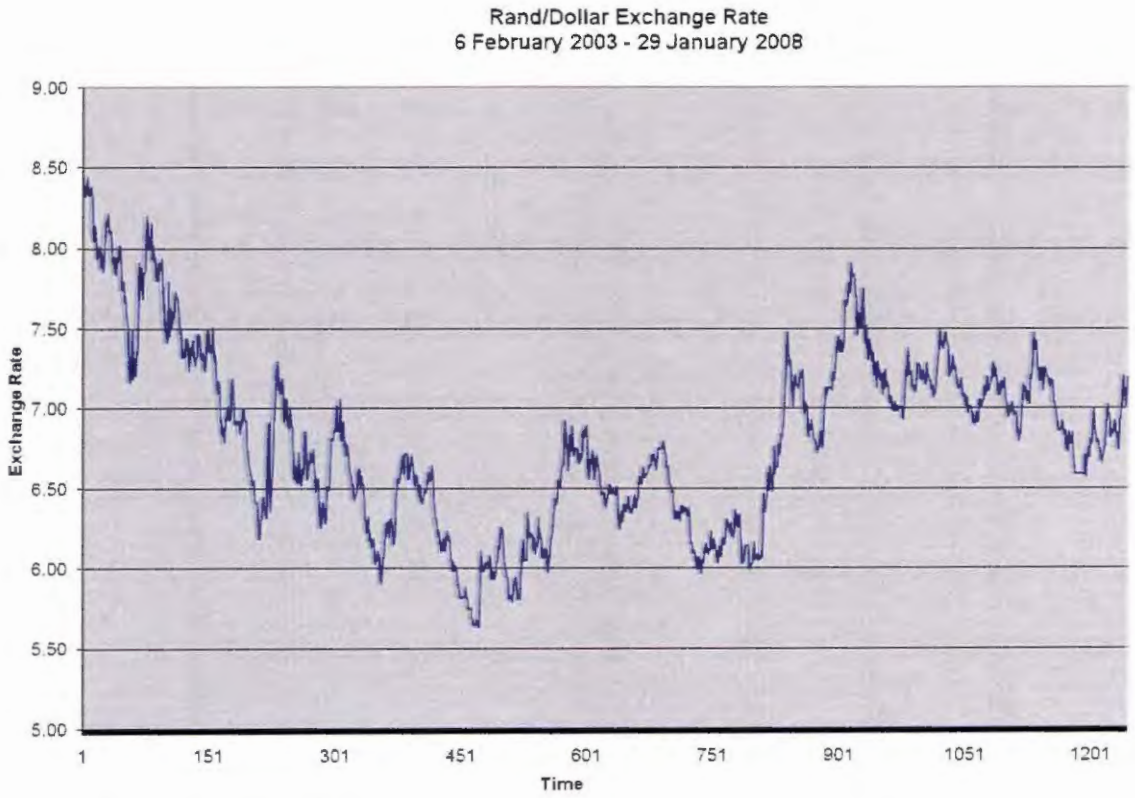


Figure 6.3 – Rand/dollar exchange rate (6 February 2003 – 29 January 2008)

Oil price time series data set

As with the demand for electricity, the demand for energy in the form of crude oil is vital to the economic well being of South Africa. Oil prices rise above 140 US dollars per barrel during 2008 and had a tremendous rippling effect on the economy that impacted each citizen in the form of higher petrol prices, higher food prices etc. The oil prices considered contains the daily closing price, in US dollars, per barrel of Brent crude oil from 6 February 2003 to 5 February 2008 giving a total of 1249 observations. Figure 6.4 shows a plot of the oil price data set.

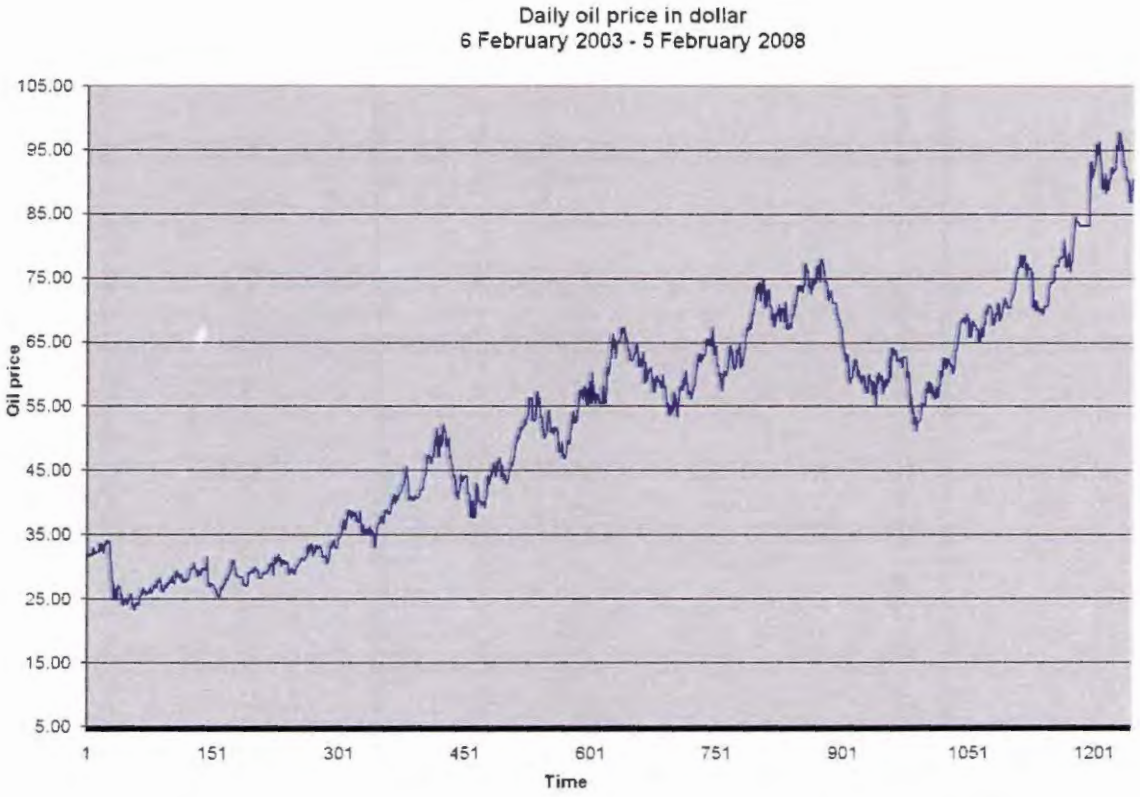


Figure 6.4 – Daily oil price (3 February 2003 – 5 February 2003)

Return on the money market time series data set

The last data set is the 3-monthly return recorded daily in the South African money market. The data set contains 498 observations from 3 January 2006 to 31 December 2007. Interest rates, which are traditionally difficult to predict, play an important role in the economy and evaluating them seems to be appropriate. Figure 6.5 shows a plot of the interest rates used in the study.

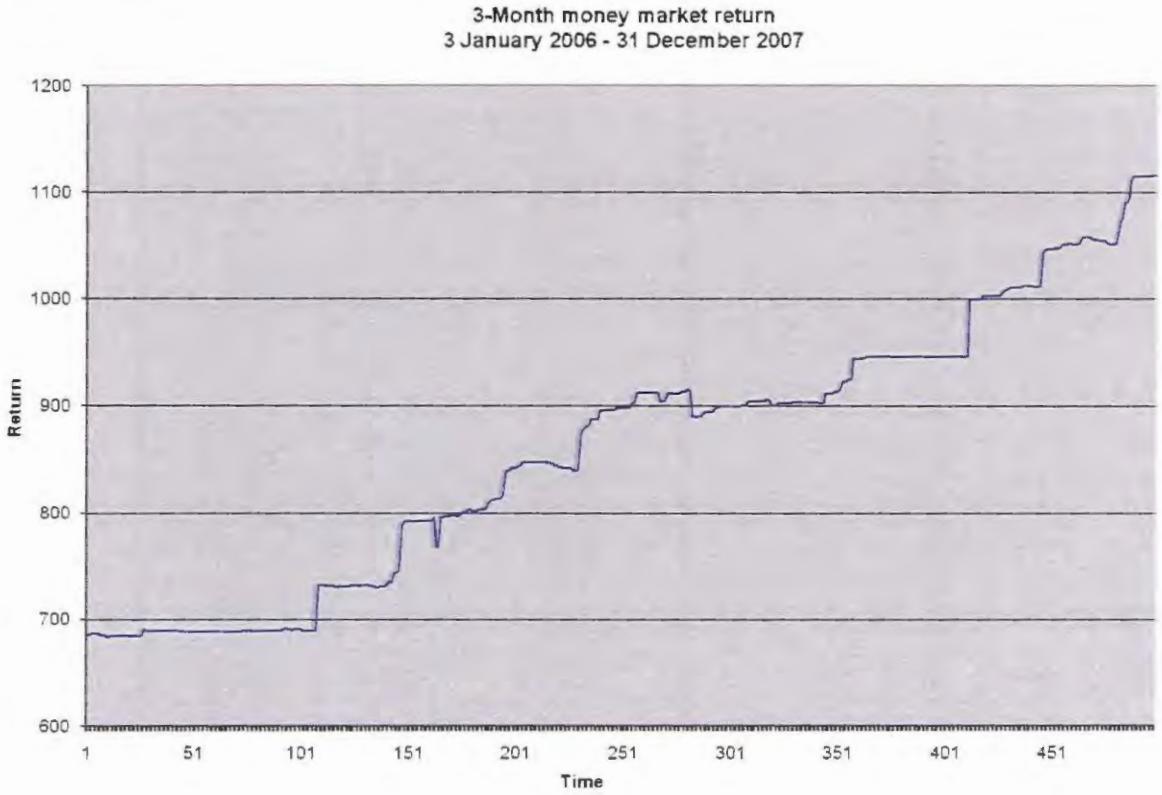


Figure 6.5 – Return on the money market (3 January 2006 – 31 December 2007)

The next section will present the results of the fitted models and their forecasting performance.

6.3 Modeling and forecasting results

As explained in chapter 5, the linear as well as the neural network models were implemented in SAS. When constructing a neural network forecasting model, researchers usually consider a one-step-ahead forecasting approach which is then used to forecast any other period e.g. to forecast 35 periods ahead. These 35 points however, refer to the test set which forms part of the original data set (Zhang, 2003). This means that the original data set was split into a training set and a test

set and the results obtained from the test set is then presented as a forecast. In this study, three different period forecasts were constructed. These were real forecasts ahead and did not form part of the original data set that was used for training and testing purposes. The three forecasting horizons chosen were a one-step-ahead forecasting, a five-step-ahead forecasting and a twenty-step-ahead forecasting. This choice was based on a one day ahead forecasting, a one week ahead (five working days) forecasting and a one month ahead (twenty working days) forecast. The mean square error (MSE) and the mean absolute deviation (MAD) were used as forecasting accuracy measures.

The results of the five different data sets will now be presented. To ensure that a comprehensive explanation is provided and at the same time adhere to the practical limitations of the written study report, most of the applicable graphs, for the first data set (gold price) will be given while for the remaining four data sets only the final results will be graphically presented.

6.3.1 Gold price time series data set

The first step in the proposed methodology was to find the "best" linear model for forecasting the time series data. Using the SAS system, the following model was selected. A linear (Holt) exponential smoothing model with root mean square error = 8.29919; MSE = 68.8765; MAPE = 0.92817; and mean absolute error = 6.01319. Additional parameters for this model is given below

Parameter	Value	Std error	T-value	P-value
LEVEL Smoothing Weight	0.999	0.031842561	31.37310471	0
TREND Smoothing Weight	0.001	0.004794163	0.208586999	0.834856117
Residual Variance (sigma squared)	69.15366516			
Smoothed Level	833.7457655			
Smoothed Trend	0.537711937			

Statistic	Value
Number of Non-missing Observations	499
Number of Observations	499

Number of Missing Actuals	0
Number of Missing Predicted Values	0
Number of Model Parameters	2
Total Sum of Squares (Uncorrected)	212916010.2
Total Sum of Squares (Corrected)	2284179.816
Sum of Square Error	34369.37158
Mean Square Error	68.87649616
Root Mean Square Error	8.299186476
Mean Absolute Percent Error	0.928171971
Mean Absolute Error	6.013191411
R-Square	0.984953299
Adjusted R-Square	0.984923024
Amemiya's Adjusted R-Square	0.984832198
Random Walk R-Square	0.000860419
Akaike Information Criterion	2115.92518
Schwarz Bayesian Information Criterion	2124.350392
Amemiya's Prediction Criterion	69.43083416
Maximum Error	24.21354604
Minimum Error	-42.59079093
Maximum Percent Error	3.877751146
Minimum Percent Error	-6.537343197
Mean Error	0.140974643
Mean Percent Error	0.010031416

The graph in figure 6.6 shows a plot of the original data and the values predicted by the linear (Holt) exponential smoothing model. Due to the size of the data set and the "closeness" of the actual and predicted values, an extraction of the data set is shown in figure 6.7 to present a

clearer view (easier to see) of the results. The extraction contains 108 data points and covers the period 3 August 2006 to 9 January 2007.

Figure 6.6 – Gold price: Linear model

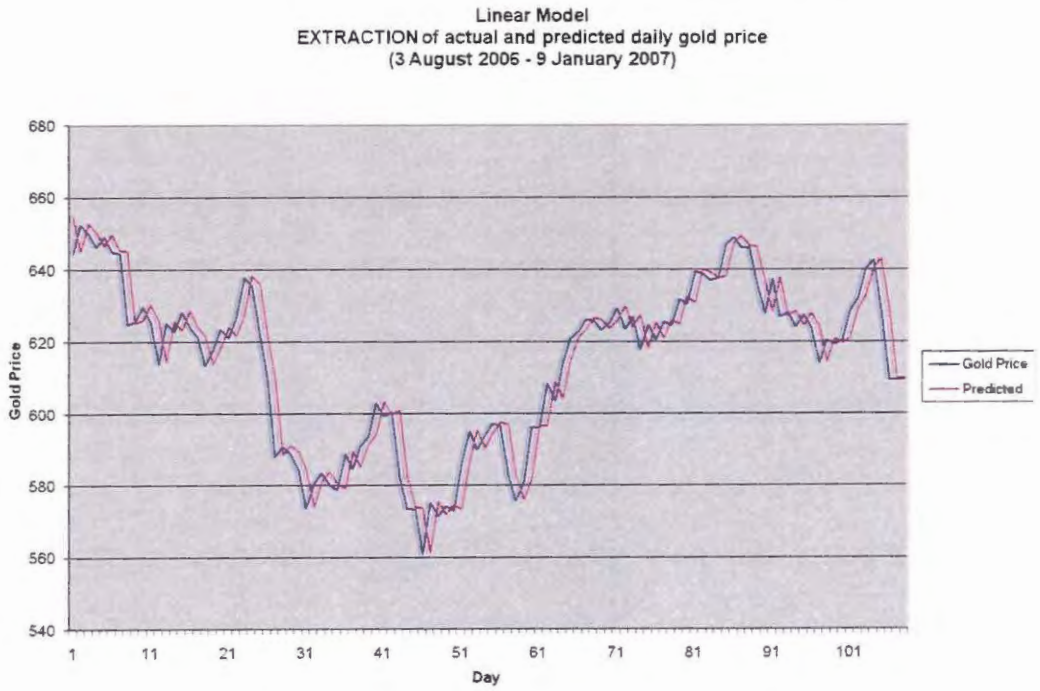


Figure 6.7 – Gold price: Linear model extraction

A grid search (explained in chapter 5) was then performed to obtain the best neural network model to produce forecasts of the original data set. The best neural network model found was one consisting of a 4-period lagged observation and one hidden node. The average squared error for this model was given as 66.1271. In figure 6.8 a plot of the original data and the values predicted by the neural network model is shown. Figure 6.9 shows the extraction for a clearer view.

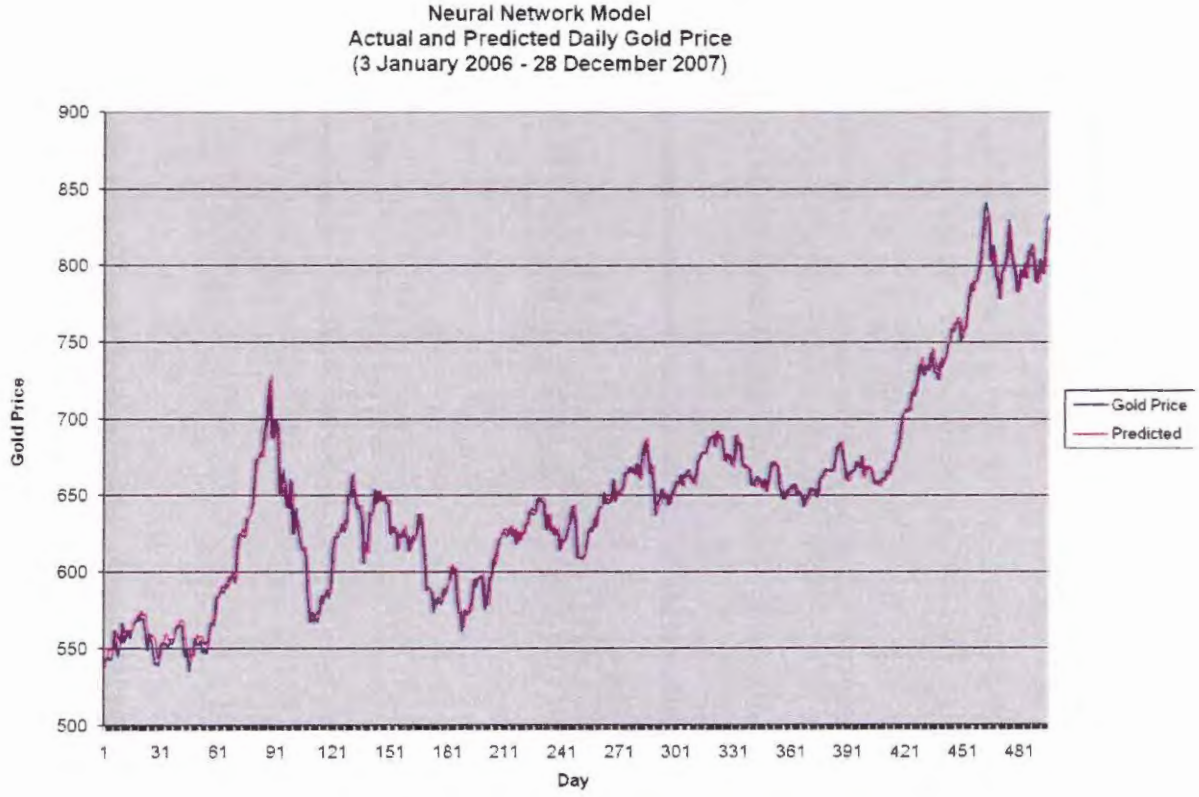


Figure 6.8 – Gold price: Neural network model

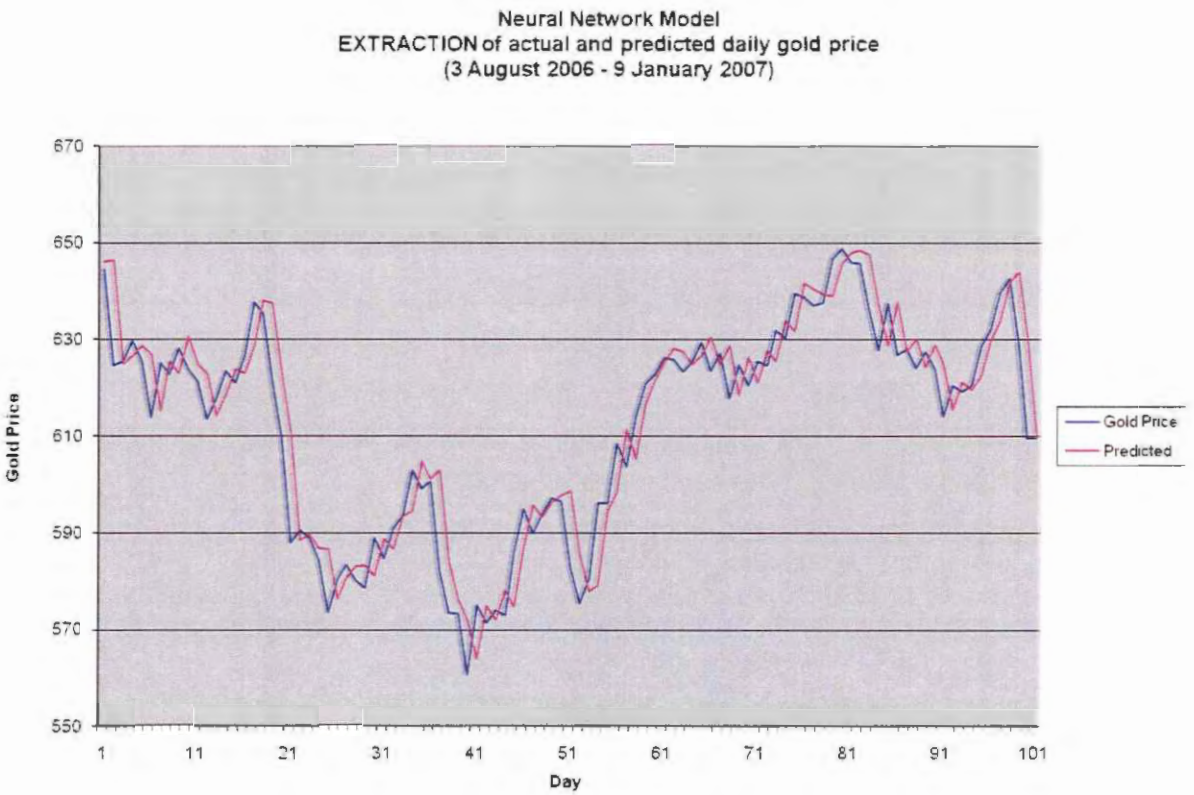


Figure 6.9 – Gold price: Neural network model extraction

6.3.1.1 One-step-ahead forecasting (gold price)

The forecasting errors from the linear forecasting model were then used to construct a neural network with the objective of forecasting the errors. A one-step-ahead approach was used and following another grid search, the best neural network model to forecast the errors was found to be one with a 1-period lagged observation and one hidden node. The average squared error in this case was 65.0822. Figure 6.10 shows a graph of the average error during training and validation of the neural network while table 6.1 presents the relevant neural network model statistics that were generated.

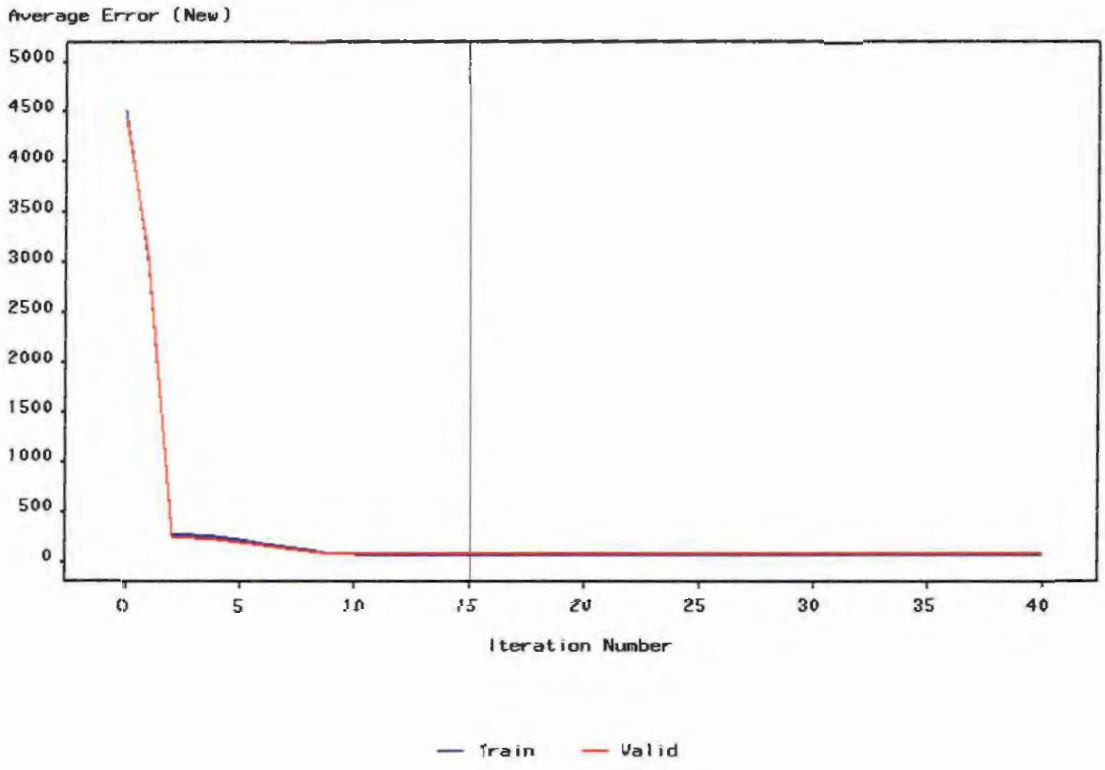


Figure 6.10 – Validation and training

Table 6.1 – Neural network statistics

Fit Statistic	Training (Train)	Validation (Valid)
Average Error	66.901327612	71.533752238
Average Squared Error	66.901327612	71.533752238
Sum of Squared Errors	23214.760681	10801.596588
Root Average Squared Error	8.1793231757	8.4577628388
Root Final Prediction Error	8.4185350752	.

Root Mean Squared Error	8.299790974	8.4577628388
Error Function	23214.760681	10801.596588
Mean Squared Error	68.886530212	71.533752238
Maximum Absolute Error	42.818014624	27.826233243
Final Prediction Error	70.871732812	.
Divisor for ASE	347	151
Model Degrees of Freedom	10	.
Total Degrees of Freedom	347	.
Sum of Frequencies	347	151
Sum Case Weights times Frequencies	347	151
Akaike's Information Criterion	1478.5169276	.
Schwarz's Bayesian Criterion	1517.0101754	.

In the final step, the hybrid model was constructed. The forecasted errors produced by the neural network model were added to the linear forecasts produced by the linear Holt exponential smoothing model. This constitutes the combined forecasting consisting of a linear and a non-linear component. Figure 6.11 presents a graph of the original data and the hybrid forecast values. Figure 6.12 shows the extraction.

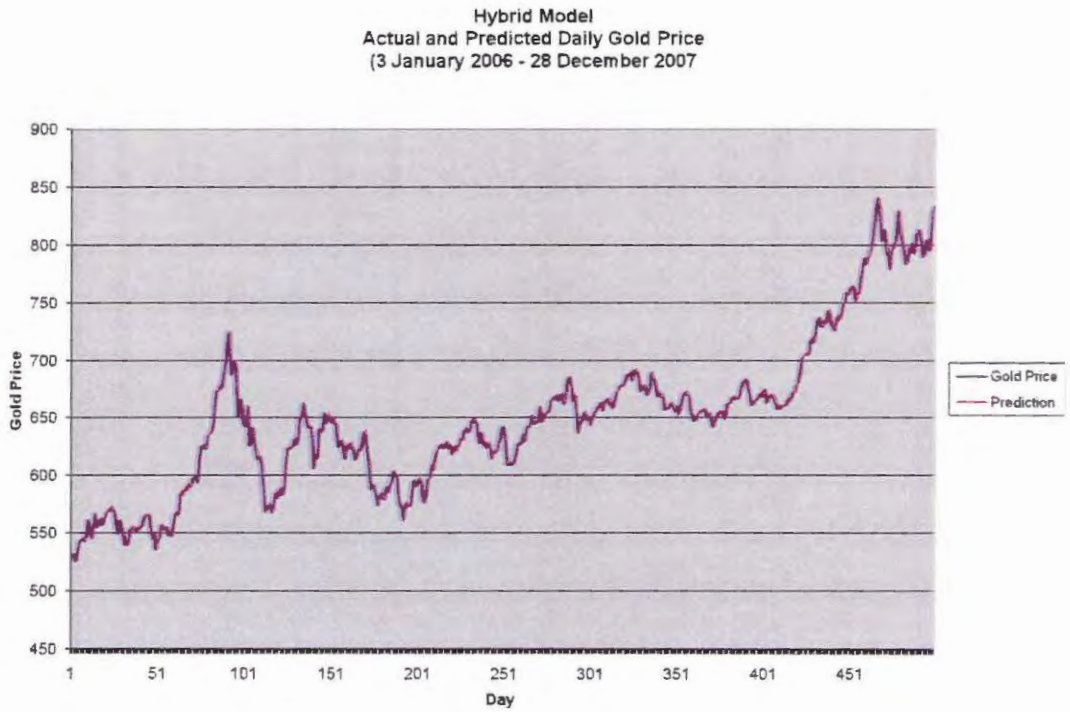


Figure 6.11– Gold price: Hybrid model

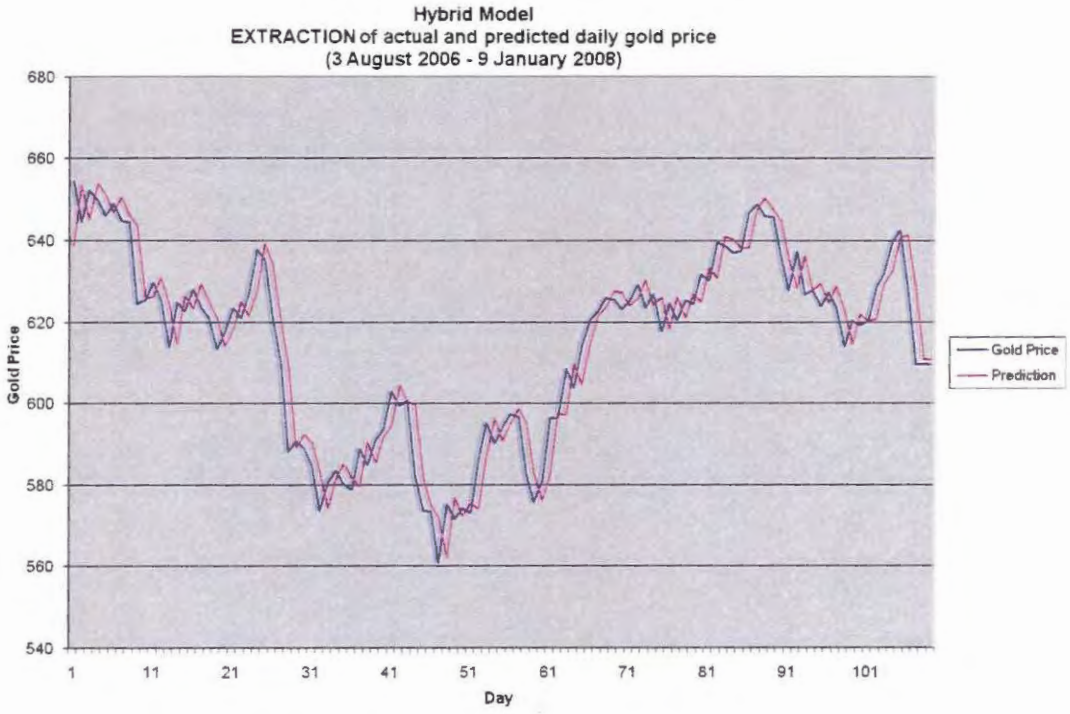


Figure 6.12 – Gold price: Hybrid model extraction

The MSE and MAD were then calculated for each forecast to be able to comment on the forecasting accuracy. The results were as follows.

	MSE	MAD
Linear (Holt exponential smoothing) forecast	67.86232	5.92614
Neural network forecast	67.77836	5.964544
Hybrid forecast	**62.53307544**	**5.9222**

The results show that the overall forecasting error can be significantly reduced by combining two models together to make provision for a linear and non-linear component in the data. In terms of the MSE, the percentage improvements of the hybrid model over the linear and neural network models for a one-step-ahead forecast are 8.5% and 8.3% respectively. A general discussion on all results will be given in section 6.4.

6.3.1.2 Five-step-ahead forecasting (gold price)

The linear model discussed earlier, is off course still applicable and all that is needed is to specify that a five-step-ahead forecast is required. The same is true for the general neural network that was constructed initially. However, to create a hybrid model that is based on a five-step-ahead forecasting, it was necessary to perform a new grid search in order to determine the best neural network model to predict five-step-ahead linear errors. The best model obtained was one using a 4-period lagged observation with one hidden node. The average squared error for this model was 78.0929. The hybrid model for the five-step-ahead forecasting was then once again constructed by adding the linear forecast and the forecasted errors together. Figure 6.13 shows a plot of the different forecasts for 5 periods ahead.

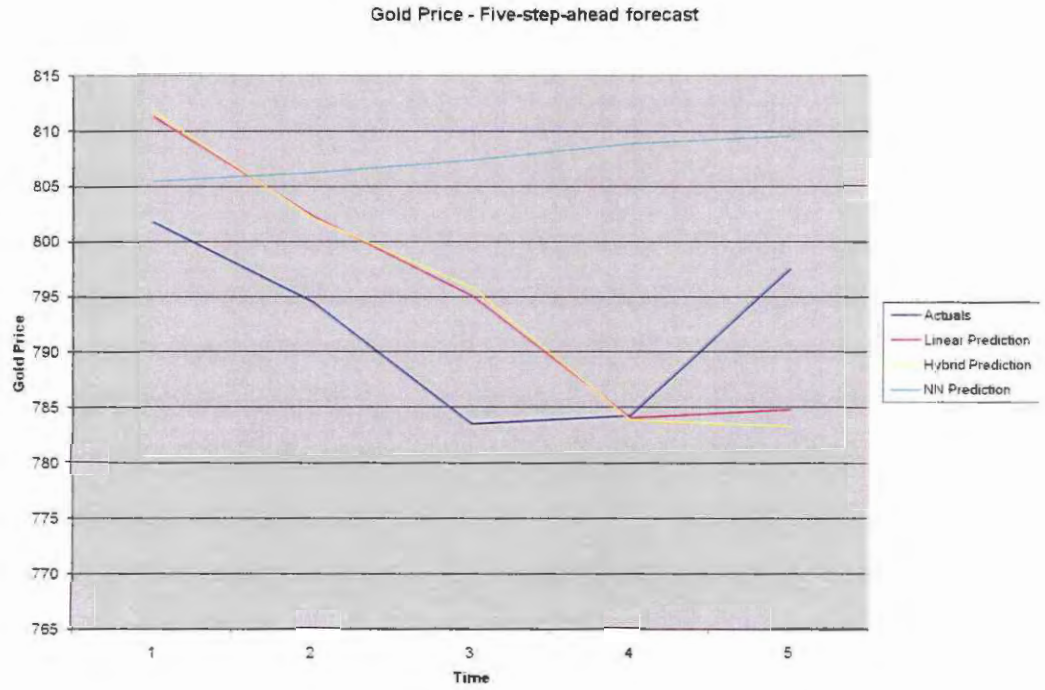


Figure 6.13 – Gold price: 5-step-ahead forecast

The MSE and MAD for the different forecasts were as follows.

	MSE	MAD
Linear (Holt exponential smoothing) forecast	**89.37518**	**8.366259**
Neural network forecast	295.1084	15.21522
Hybrid forecast	102.1858521	8.8994743

For the five-period-ahead forecast, the linear Holt exponential smoothing model performed best with the hybrid forecast relatively close to the linear forecast. The neural network model performed badly in this case. This makes sense because whenever there is a huge difference in values from one data point to the next one, the linear forecast is expected to perform better. This is due to the fact that each linear forecast is influenced by the previous observed value. If the forecast horizon is not big enough, the neural network will not be able to make adjustments to

the new forecasts quickly enough. This can clearly be seen from the graph presented in figure 6.13. The data below, and plotted in figure 6.13, also shows this clearly.

Actual	Linear Prediction	Hybrid Prediction	NN Prediction
801.75	811.2947141	811.7697444	805.4896487
794.5	802.2749432	802.0955222	806.2570478
783.5	795.0154063	795.7949216	807.4074452
784.25	784.0076429	783.8587255	808.8846416
797.5	784.7461272	783.3040909	809.5373213

6.3.1.3 Twenty-step-ahead forecasting (gold price)

A grid search was carried out again to determine the best neural network architecture to forecast errors for the twenty-step-ahead forecasting. The best neural network model found was one using a 6-period lagged observation with one hidden node. The average squared error in this case was 87.3985. Figure 6.14 shows a graph of the different forecasts for 20 periods ahead.

Figure 6.14 – Gold price: 20-step-ahead forecast

The MSE and MAD were calculated as follows.

	MSE	MAD
Linear (Holt exponential smoothing) forecast	101.4591	8.653258
Neural network forecast	201.8027	10.97838
Hybrid forecast	**95.02055394**	**8.443926**

In this case, the hybrid forecast once again outperformed the linear forecast (6.8% improvement in the MSE) with the neural network performing bad again. The same argument for the five-step-ahead neural network forecast can be used to try and explain the poor performance of the neural network model. Below is the data plotted in figure 6.14.

Actual	Linear Prediction	Hybrid Prediction	NN Prediction
801.75	811.2947141	810.7302324	784.4241649
794.5	802.2749432	802.131793	784.4941773
783.5	795.0154063	795.654873	785.6731112
784.25	784.0076429	783.6571611	786.248012
797.5	784.7461272	784.1003376	787.3720737
793	797.9963568	798.4655281	788.1385661
801.5	793.5091157	793.533676	791.1534747
792.5	802.0041113	802.4201978	790.3913254
809.5	793.0121117	794.0459772	791.9488143
808.75	810.0025911	810.4265239	791.9487711
814	809.2690803	809.1865494	793.1386129
800.7	814.5178229	813.2455098	792.9663975
789.5	801.2225677	800.9166471	795.3034878
790.75	790.0087616	790.2086644	795.2589536
804.25	791.2470383	791.8238964	797.6338308

799.75	804.7477665	803.2082274	798.1972506
795.25	800.2607745	799.1892615	799.236993
810.5	795.7557817	796.8824198	799.2067427
829	811.0007562	812.0712484	800.0480885
833.75	829.5154824	827.8003435	798.7906171

Based on the results of the experiment with the gold price data it seems as if it would make good sense to make use of a combination of techniques in order to cater for both a linear and a non-linear component.

The following sections present the results of the other four experiments. The approach used with each of the data sets was the same as what was described in this section for the gold price and results will therefore presented without repeating detailed descriptions again. It should also be noted that a discussion on all the results will follow in section 6.4.

6.3.2 Demand for electricity time series data set

The best linear model, suggested by SAS, to forecast the electricity demand was a simple exponential smoothing model with root mean square error = 0.004462; MSE = 0.0000199; MAPE = 0.43087; and mean absolute error = 0.003262. Other parameters for the model were as follows.

Model Parameter	Estimate	Std error	T-value	Prob>|T|
LEVEL Smoothing Weight	0.999	0.0238	42.0253	0
Residual Variance (sigma squared)	0.0000199			
Smoothed Level	0.6562			

Statistic of Fit	Value
Number of Nonmissing Observations	884
Number of Observations	884
Number of Missing Actuals	0

Number of Missing Predicted Values	0
Number of Model Parameters	1
Total Sum of Squares (Uncorrected)	503.559253
Total Sum of Squares (Corrected)	6.23485
Sum of Square Error	0.017598
Mean Square Error	1.9907E-05
Root Mean Square Error	0.004462
Mean Absolute Percent Error	0.430865
Mean Absolute Error	0.003262
R-Square	0.997177
Adjusted R-Square	0.997177
Amemiya's Adjusted R-Square	0.997171
Random Walk R-Square	0.000265
Akaike Information Criterion	-9566.7829
Schwarz Bayesian Information Criterion	-9561.99844
Amemiya's Prediction Criterion	1.9953E-05
Maximum Error	0.016891
Minimum Error	-0.0183
Maximum Percent Error	2.019356
Minimum Percent Error	-2.143885
Mean Error	-0.00004393
Mean Percent Error	-0.00819

Figure 6.15 presents an extraction of the linear forecasts for a 100 data points (from the original 884 data points) which covers the period 1 July 2006 to 2 July 2006.

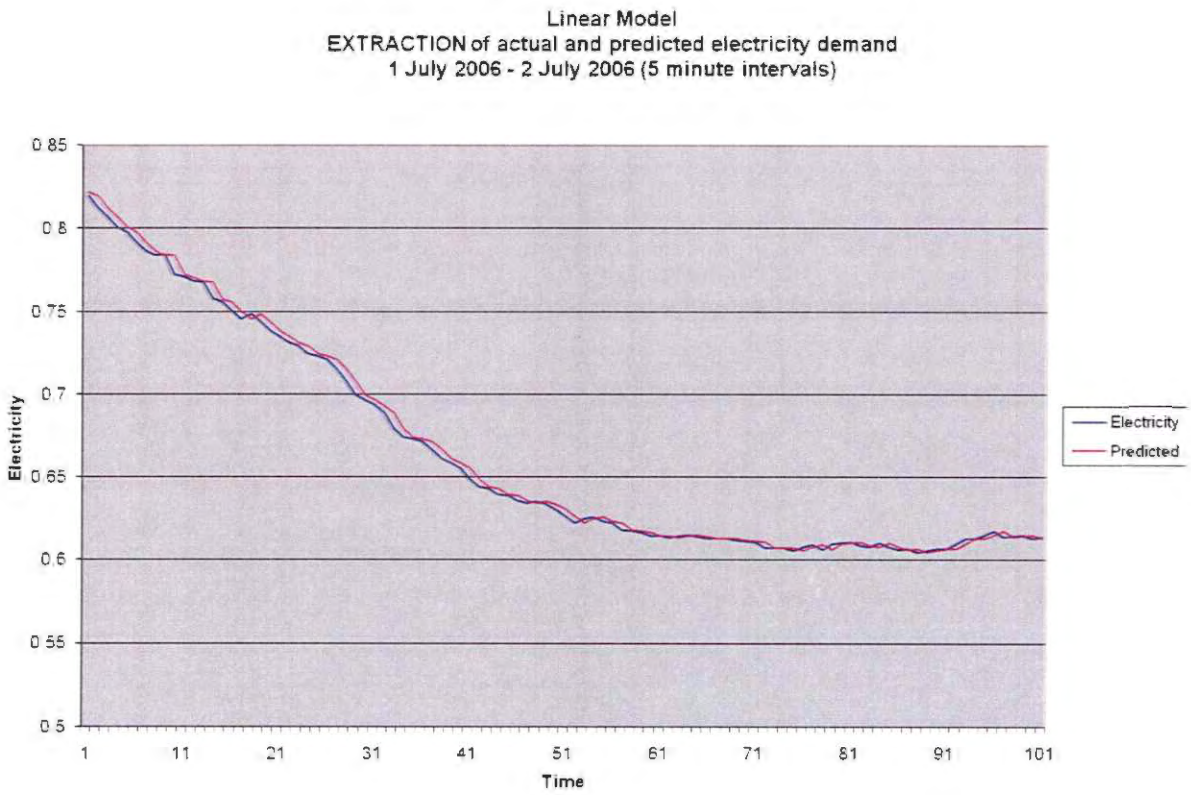


Figure 6.15 – Electricity demand: Linear model extraction

Following a grid search, the best neural network model to perform forecasts of the original data set, was found to be one using a 10-period lagged observation with one hidden node. The average squared error was 0.000012. Figure 6.16 shows an extraction of the neural network forecasts.

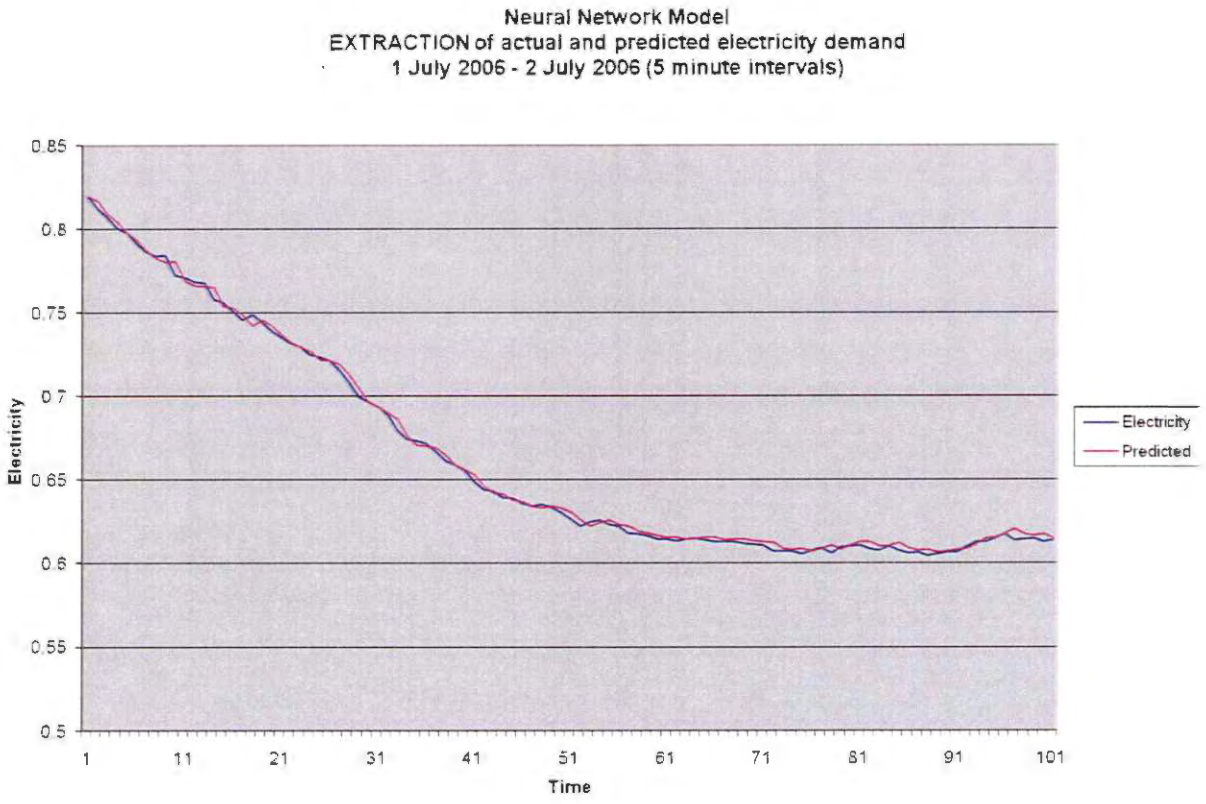


Figure 6.16 – Electricity demand: Neural network model extraction

6.3.2.1 One-step-ahead forecasting (electricity demand)

A grid search indicated that the best neural network model to forecast errors, using the one-step-ahead approach, was one with an 11-period lagged observation and 18 hidden nodes. The average squared error was determined as 0.000012. A hybrid model was then constructed using the linear forecasts together with the neural network forecasts of the errors. Figure 6.17 shows the extraction of the hybrid forecasts.

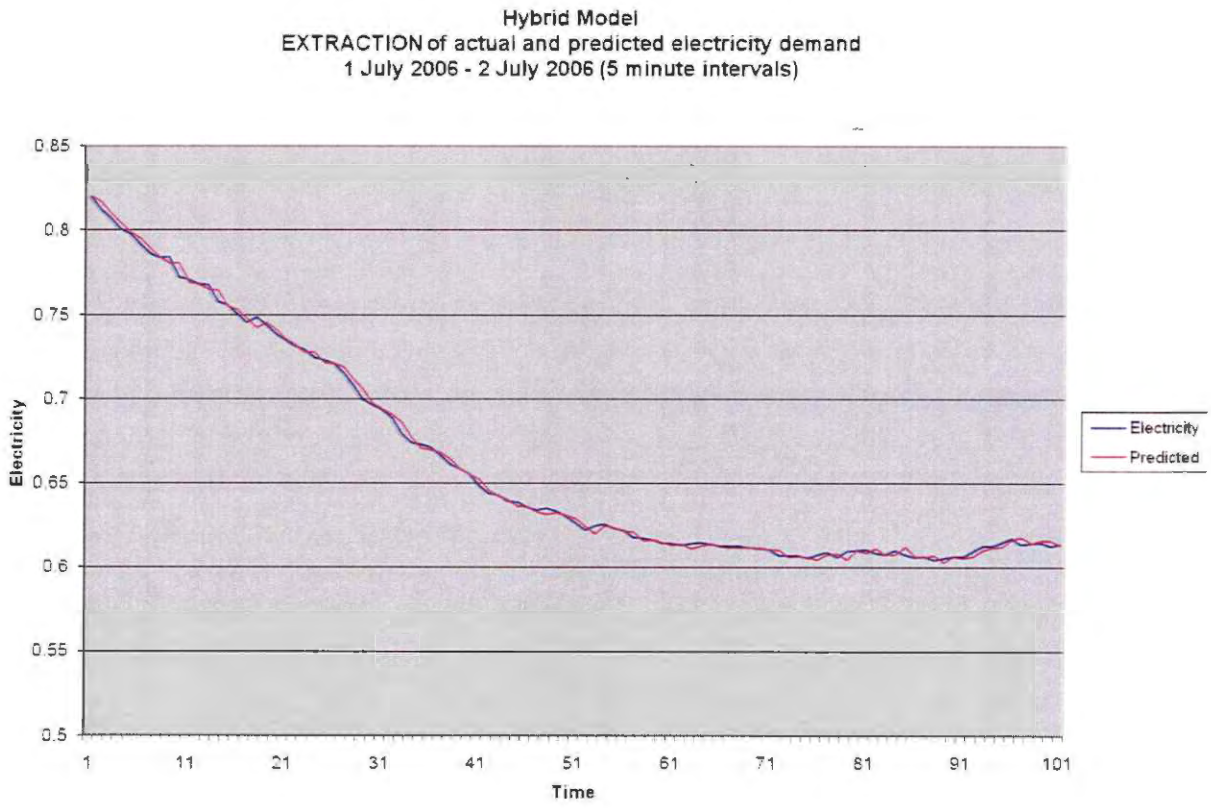


Figure 6.17 – Electricity demand: Hybrid model extraction

The MSE and MAD were calculated as

	MSE	MAD
Linear (exponential smoothing) forecast	0.0000201712	0.00328
Neural network forecast	**0.000012855**	**0.002657**
Hybrid forecast	0.0000143209	0.002766

The neural network forecast performed best with the hybrid model second best. The differences in the MSE and MAD for the different models were very close and it is difficult in this case to make definite conclusions regarding the hybrid model. Perhaps the most significant conclusion is that the hybrid model did not perform significantly worse than the other two models.

6.3.2.2 Five-step-ahead forecasting (electricity demand)

The best neural network model to forecast errors using the five-step-ahead forecasting was a model with an 11-period lagged observation and 20 hidden nodes. The average squared error for this architecture was 0.000014. A hybrid model was then constructed by combining the linear forecasts and the neural network forecasts of errors. Figure 6.18 shows a plot of the different forecasts for 5 periods ahead.

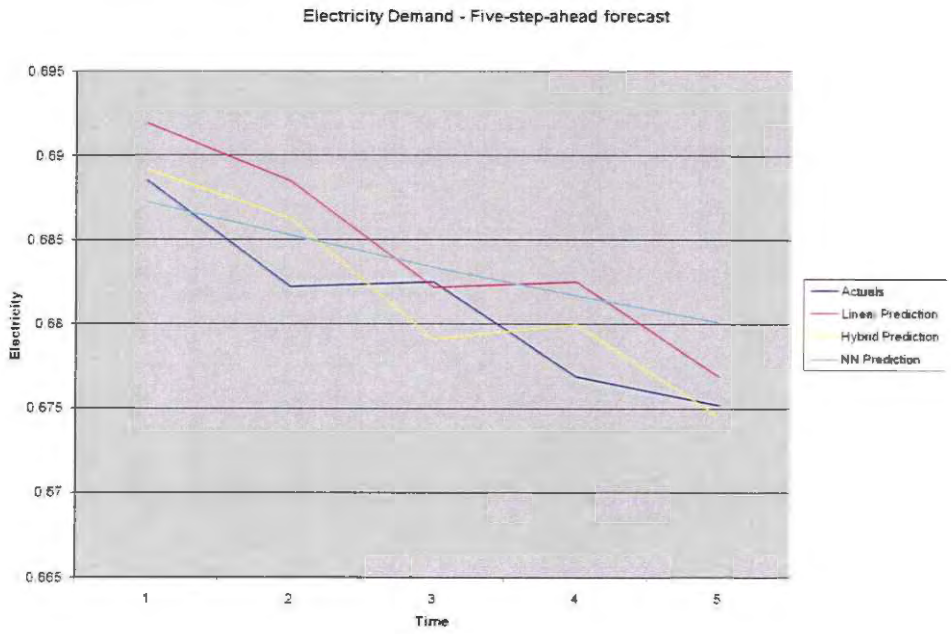


Figure 6.18 – Electricity demand: 5-step-ahead forecast

The MSE and MAD were as follows.

	MSE	MAD
Linear (exponential smoothing) forecast	0.000017118	0.00346
Neural network forecast	0.0000117565	0.002982
Hybrid forecast	**0.00000771548**	**0.002376**

Even though the MSE and MAD values are very close, it is significant to note that the hybrid model performed best.

6.3.2.3 Twenty-step-ahead forecasting (electricity demand)

The best neural network model to forecast errors for a twenty-step-ahead forecasting was one using a 5-period lagged observation with 6 hidden nodes. The average squared error for this model was 0.000019. Figure 6.19 shows the different forecasts for twenty periods ahead.

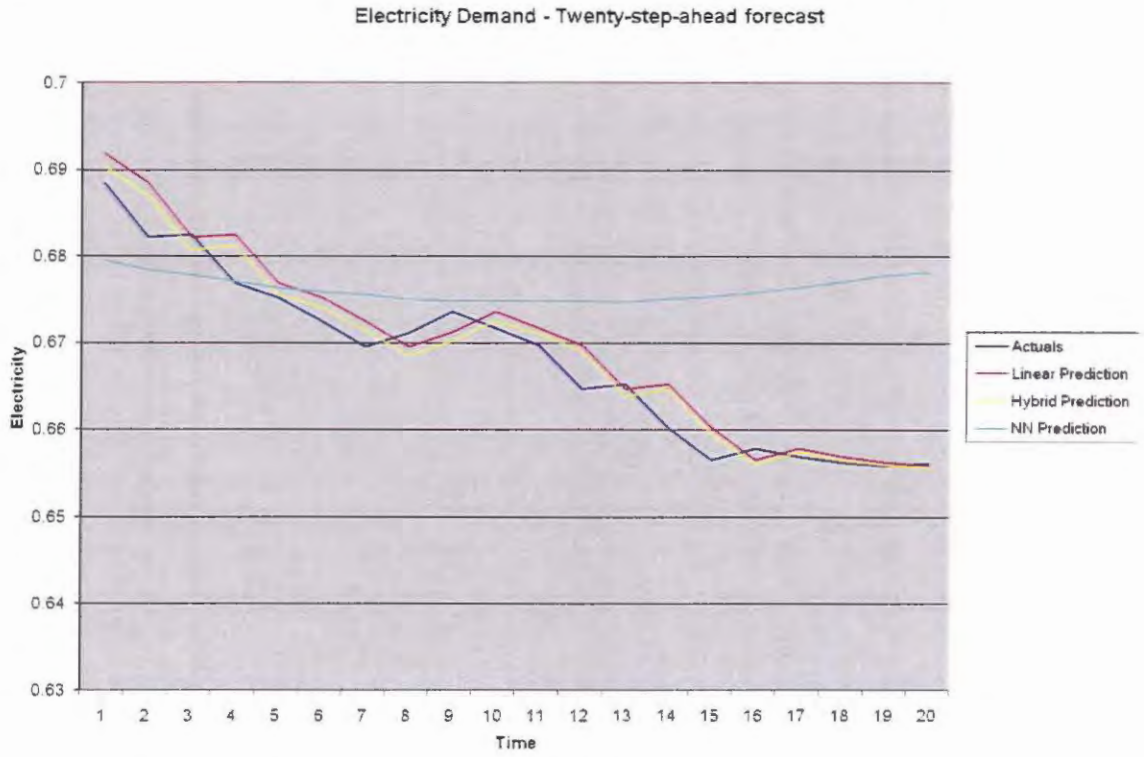


Figure 6.19 – Electricity demand: 20-step-ahead forecast

The MSE and MAD were as follows.

	MSE	MAD
Linear (exponential smoothing) forecast	0.0000092745	0.002445
Neural network forecast	0.000154442	0.009856
Hybrid forecast	**0.00000652141**	**0.002068**

As with the five-step-ahead forecast, the differences are very small with the hybrid model that performed best.

The experiment with the electricity data has shown that it might be worthwhile to consider constructing a hybrid model. The hybrid model performed best in the last two cases and second best in the first case and although it did not outperform the other models by far, it appears as if it is still a better model.

6.3.3 Rand/Dollar exchange rate time series data set

Using the SAS system, the best linear model to forecast the rand/dollar exchange rate was again found to be a simple exponential smoothing model with root mean square error = 0.07679; MSE = 0.005897; MAPE = 0.80757; and mean absolute error = 0.05532. The other parameters for the model were as follows.

Model Parameter	Estimate	Std error	T-value	Prob>|T|
LEVEL Smoothing Weight	0.98372	0.02	49.1695	0
Residual Variance (sigma squared)	0.0059			
Smoothed Level	7.47108			

Statistic of Fit	Value
Number of Nonmissing Observations	1249
Number of Observations	1249
Number of Missing Actuals	0
Number of Missing Predicted Values	0
Number of Model Parameters	1
Total Sum of Squares (Uncorrected)	58255
Total Sum of Squares (Corrected)	391.711752
Sum of Square Error	7.36493
Mean Square Error	0.005897
Root Mean Square Error	0.07679

Mean Absolute Percent Error	0.807566
Mean Absolute Error	0.055321
R-Square	0.981198
Adjusted R-Square	0.981198
Amemiya's Adjusted R-Square	0.981168
Random Walk R-Square	0.000949
Akaike Information Criterion	-6409.577789
Schwarz Bayesian Information Criterion	-6404.44769
Amemiya's Prediction Criterion	0.005906
Maximum Error	0.431485
Minimum Error	-0.312644
Maximum Percent Error	6.395407
Minimum Percent Error	-4.736962
Mean Error	-0.00075
Mean Percent Error	-0.015783

Figure 6.20 presents an extraction of the linear forecasts for a 100 data points (from the original 1244 data points) which covers the period 12 February 2004 to 9 July 2004.

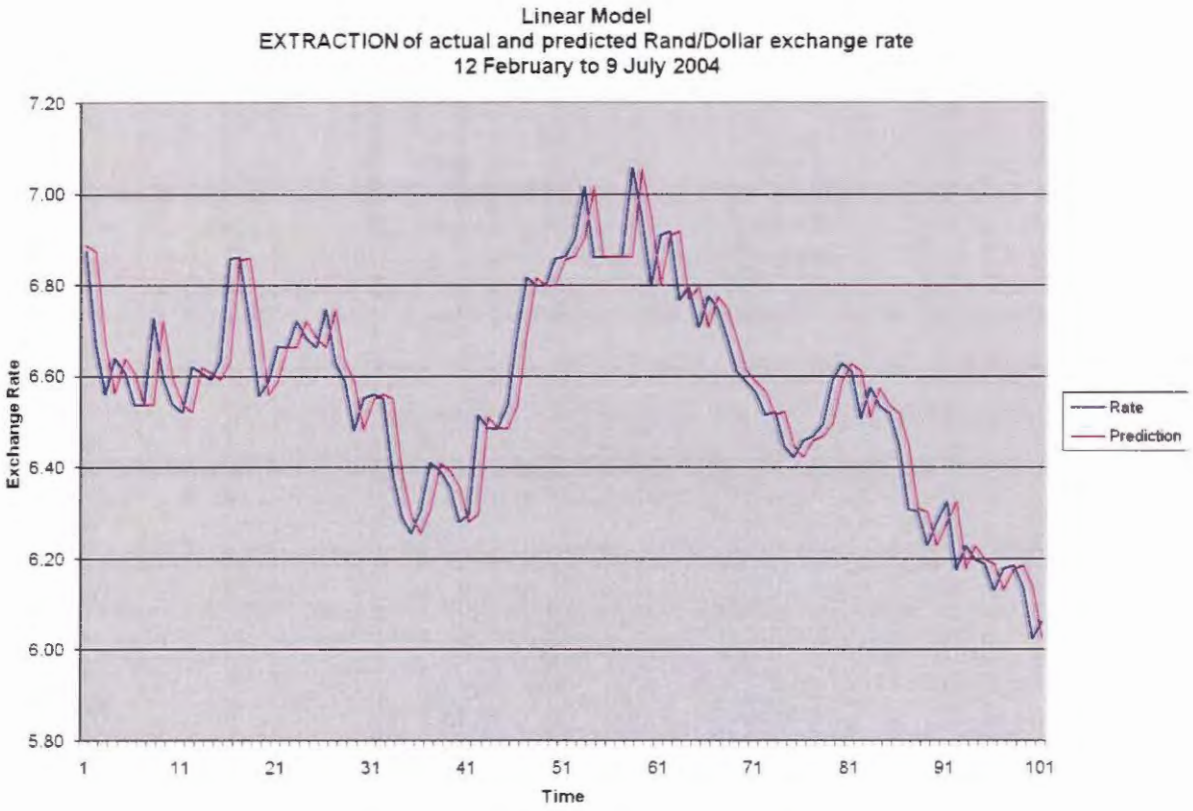


Figure 6.20 – Exchange rate: Linear model extraction

Using the grid search again, the best neural network model to perform forecasts of the original data set was found to be one using a 1-period lagged observation with eight hidden nodes. The average squared error was 0.005744. Figure 6.21 shows the extraction of the neural network forecasts.

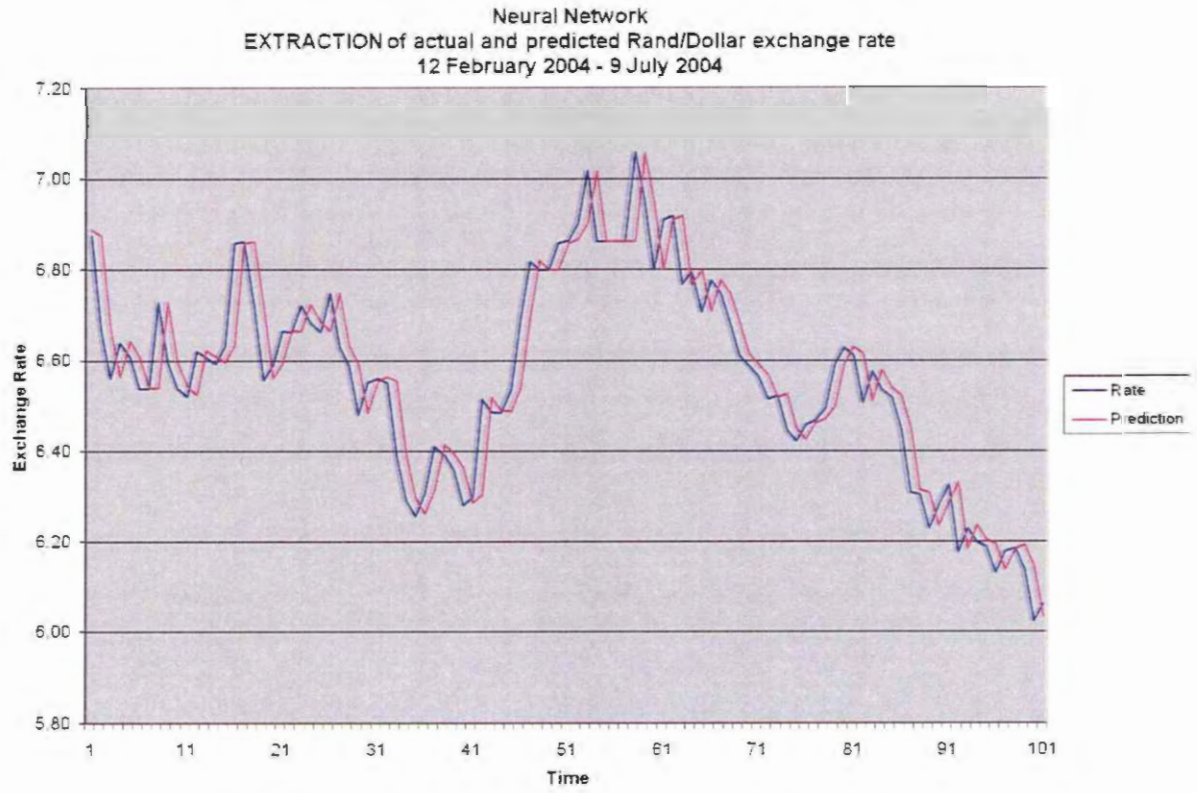


Figure 6.21 – Exchange rate: Neural network model extraction

6.3.3.1 One-step-ahead forecasting (exchange rate)

The best neural network architecture, to forecast the errors, was one using a 9-period lagged observation and 1 hidden node. The average squared error was 0.005875. The linear forecasts and the errors forecasted by the neural network were then used to construct the hybrid forecast of which the extraction is plotted in figure 6.22.

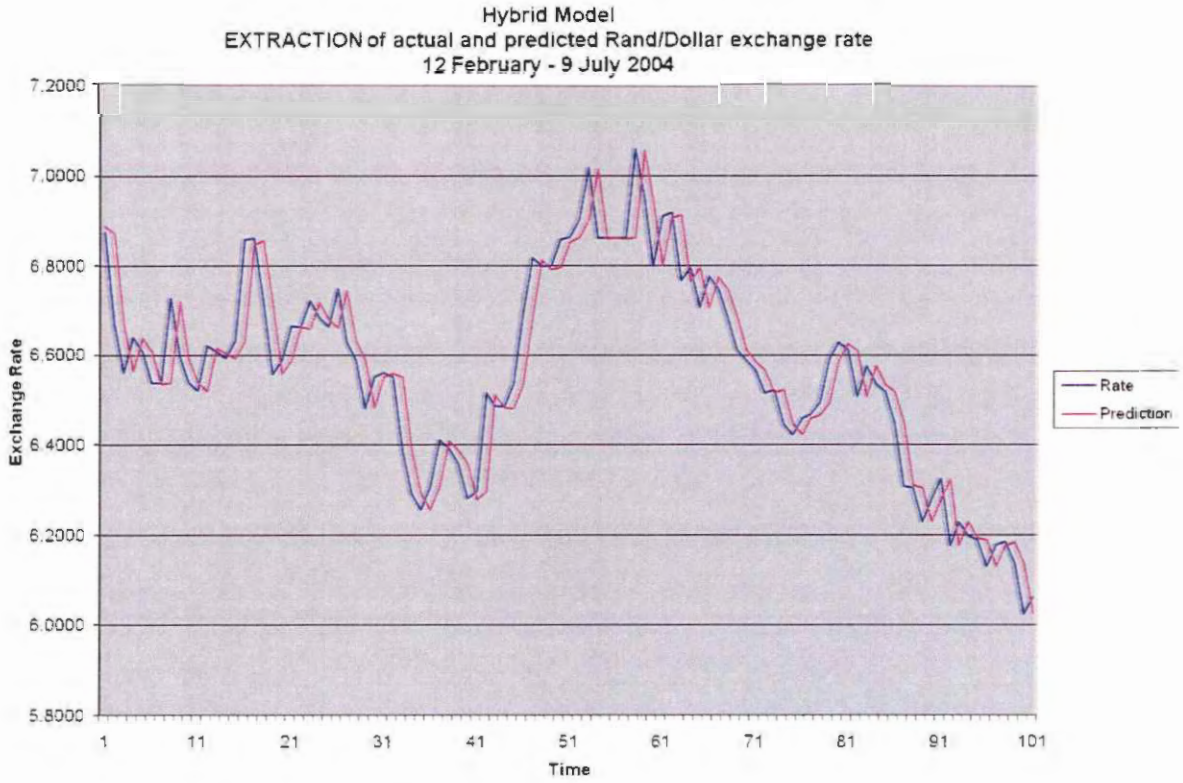


Figure 6.22 – Exchange rate: Hybrid model extraction

The MSE and MAD were

	MSE	MAD
Linear (exponential smoothing) forecast	**0.005817**	0.0551898
Neural network forecast	0.005871	0.055275
Hybrid forecast	0.005875	**0.055165**

For the one-step-ahead forecasting, the linear model performed best in terms of the MSE while the hybrid model performed best when using the MAD as an accuracy measure. As with the electricity data set, the differences were very small and the only significant conclusion is that the hybrid model did not perform worse.

6.3.3.2 Five-step-ahead forecasting (exchange rate)

The best neural network model to forecast errors using the five-step-ahead forecasting was a model with a 2-period lagged observation and 3 hidden nodes. The average squared error in this case was 0.006594. A hybrid model was then once again constructed by combining the linear forecasts and the neural network forecasts of errors. Figure 6.23 shows a graph of the different forecasts for 5 periods ahead.

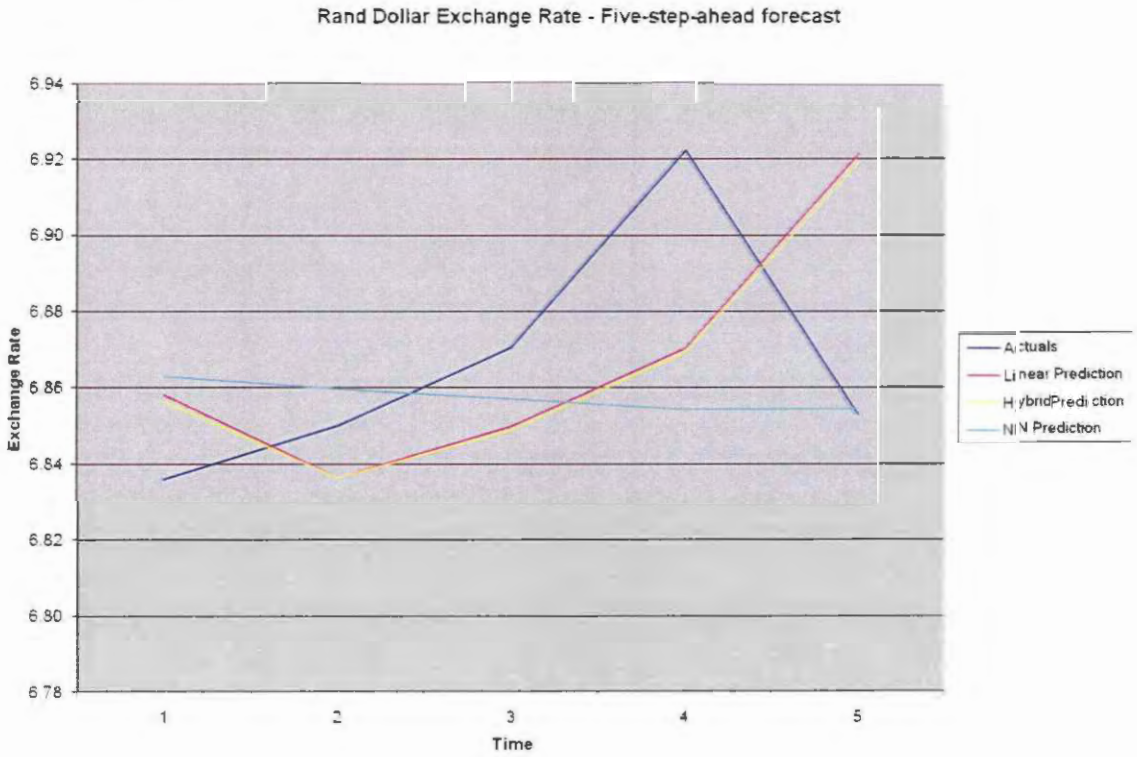


Figure 6.23 – Exchange rate: 5-step-ahead forecast

The MSE and MAD for the five-step-ahead forecasts were as follows.

	MSE	MAD
Linear (exponential smoothing) forecast	0.001701	0.03544
Neural network forecast	**0.001124**	**0.02393**
Hybrid forecast	0.001662	0.035149

In this case the neural network model performed best with the hybrid model second best.

6.3.3.3 Twenty-step-ahead forecasting (exchange rate)

Following a grid search, the best neural network model to forecast errors for a twenty-step-ahead forecasting was found to be one using a 15-period lagged observation with 3 hidden nodes. The average squared error was 0.007022. Figure 6.24 shows the different forecasts for twenty periods ahead.

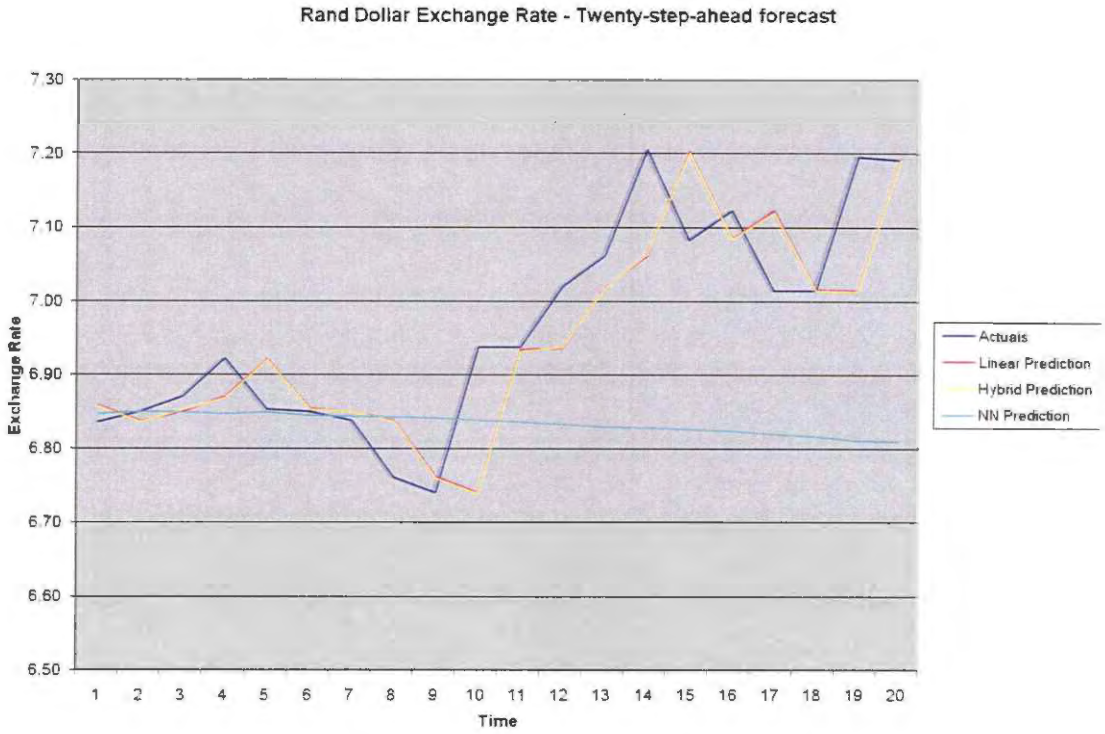


Figure 6.24 – Exchange rate: 20-step-ahead forecast

The MSE and MAD were as follows.

	MSE	MAD
Linear (exponential smoothing) forecast	0.00718	0.06063
Neural network forecast	0.040013	0.150828
Hybrid forecast	**0.007102**	**0.060121**

In line with the gold and electricity data sets, the hybrid model once again performed best for a twenty-step-ahead forecast despite the very small difference in MSE for the linear and hybrid models.

The experiment with the rand/dollar data set did not reveal any superiority of any of the three models. Maybe the most significant finding again is that the hybrid model never performed worst of the three models. This might be an indication (or confirmation) that hybrid models of this nature should not be completely disregarded.

6.3.4 Oil price time series data set

The best linear model to forecast the daily oil price was suggested by the SAS system as a damped trend exponential smoothing model with root mean square error = 1.14859; MSE = 1.31927; MAPE = 1.54706; and mean absolute error = 0.79142. The other parameters for the model were as follows.

Model Parameter	Estimate	Std error	T-value	Prob>|T|
LEVEL Smoothing Weight	0.71836	0.4201	1.7101	0.08749209
TREND Smoothing Weight	0.999	5.631	0.1774	0.85921384
DAMPING Smoothing Weight	0.21882	0.53	0.4129	0.67974762
Residual Variance (sigma squared)	1.32245			
Smoothed Level	90.19155			
Smoothed Trend	-2.33489			

Statistic of Fit	Value
Number of Nonmissing Observations	1249
Number of Observations	1249
Number of Missing Actuals	0
Number of Missing Predicted Values	0
Number of Model Parameters	3

Total Sum of Squares (Uncorrected)	3966511
Total Sum of Squares (Corrected)	422677
Sum of Square Error	1647.766813
Mean Square Error	1.319269
Root Mean Square Error	1.148594
Mean Absolute Percent Error	1.547057
Mean Absolute Error	0.791417
R-Square	0.996102
Adjusted R-Square	0.996095
Amemiya's Adjusted R-Square	0.996083
Random Walk R-Square	0.016794
Akaike Information Criterion	352.070039
Schwarz Bayesian Information Criterion	367.460335
Amemiya's Prediction Criterion	1.325622
Maximum Error	9.89
Minimum Error	-5.262413
Maximum Percent Error	10.853975
Minimum Percent Error	-12.56334
Mean Error	0.050535
Mean Percent Error	0.065131

An extraction of the linear forecast, showing 100 data points (from the original 1249 data points) is given in figure 6.25. The 100 data points cover the period 3 November 2004 to 30 May 2005.

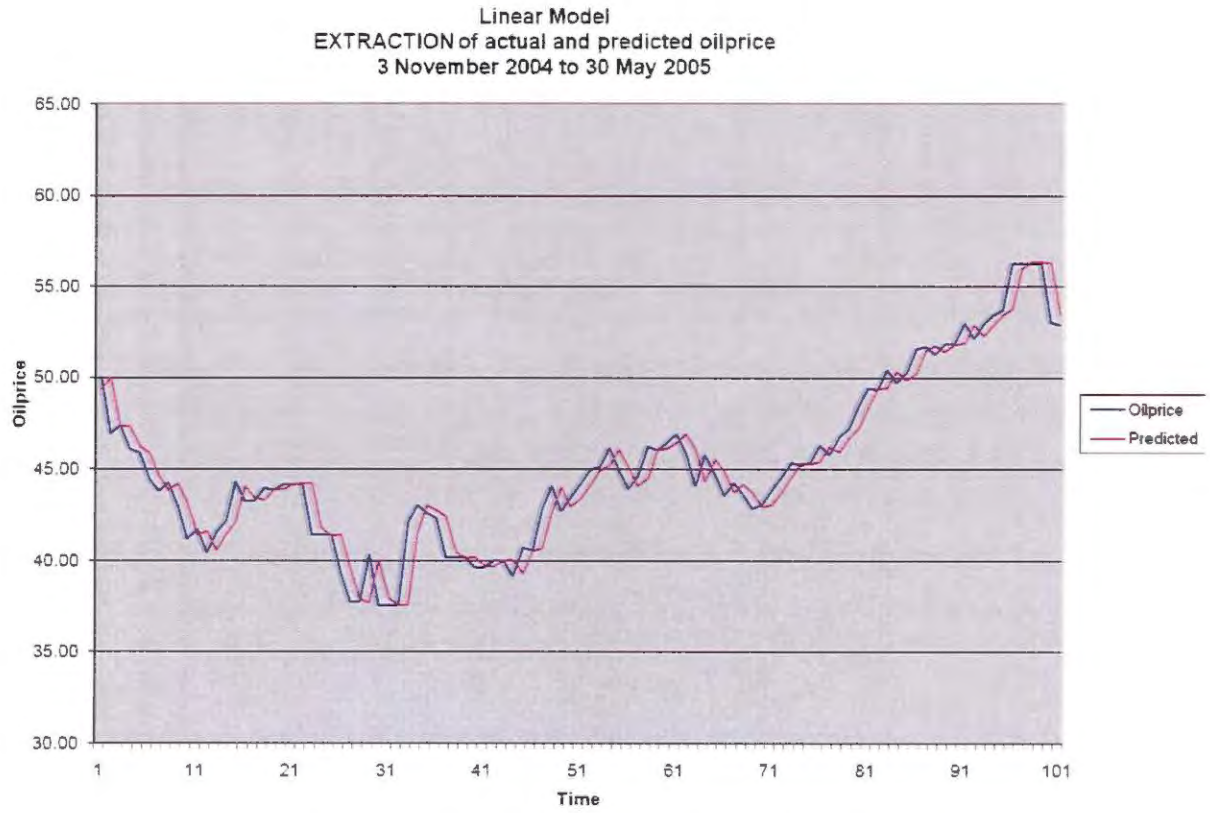


Figure 6.25 – Oil price: Linear model extraction

A grid search was then performed to obtain the best neural network model to forecast the original data set. The best model found was one using an 8-period lagged observation with one hidden node. The average squared error was given as 1.265244. Figure 6.26 shows the extraction of the neural network forecasts.

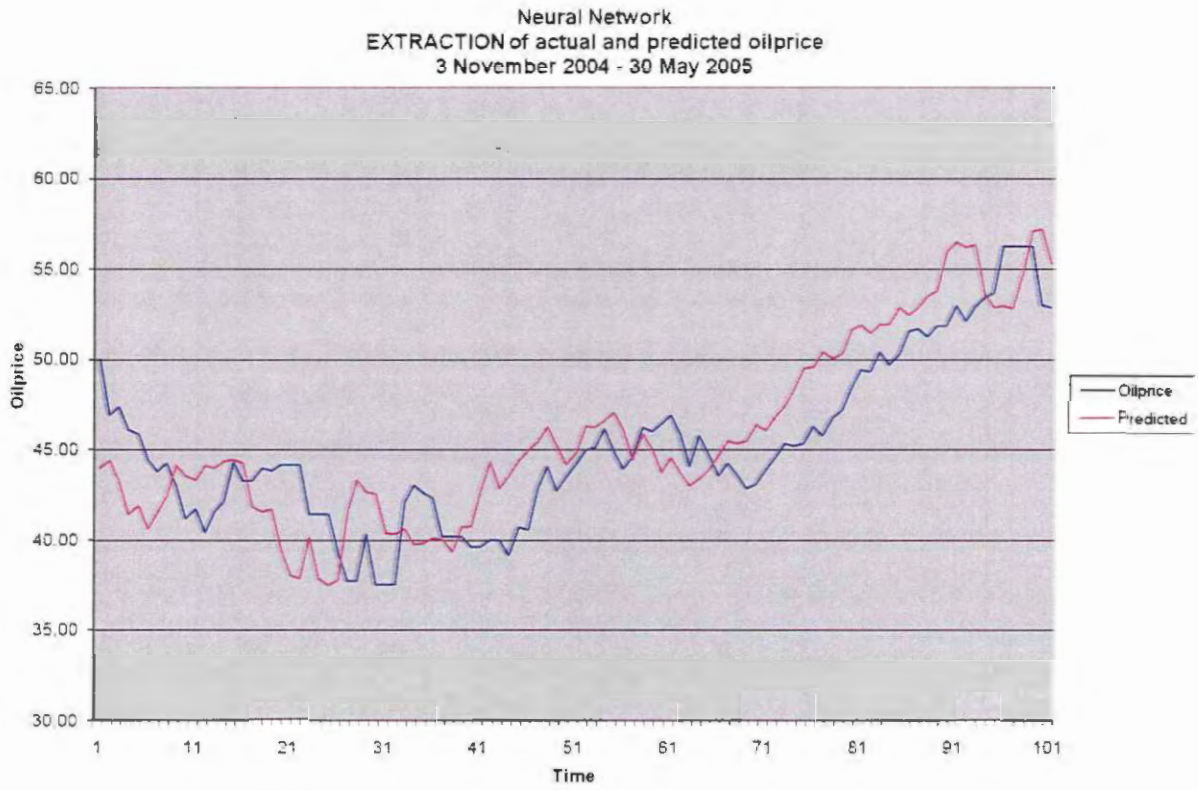


Figure 6.26 – Oil price: Neural network model extraction

6.3.4.1 One-step-ahead forecasting (oil price)

A 1-period lagged observation with 3 hidden nodes was the best neural network architecture to forecast the errors. The average squared error for this neural network model was 1.253564. The forecasted errors and the linear forecast were then used to construct the hybrid forecast. An extraction is plotted in figure 6.27.

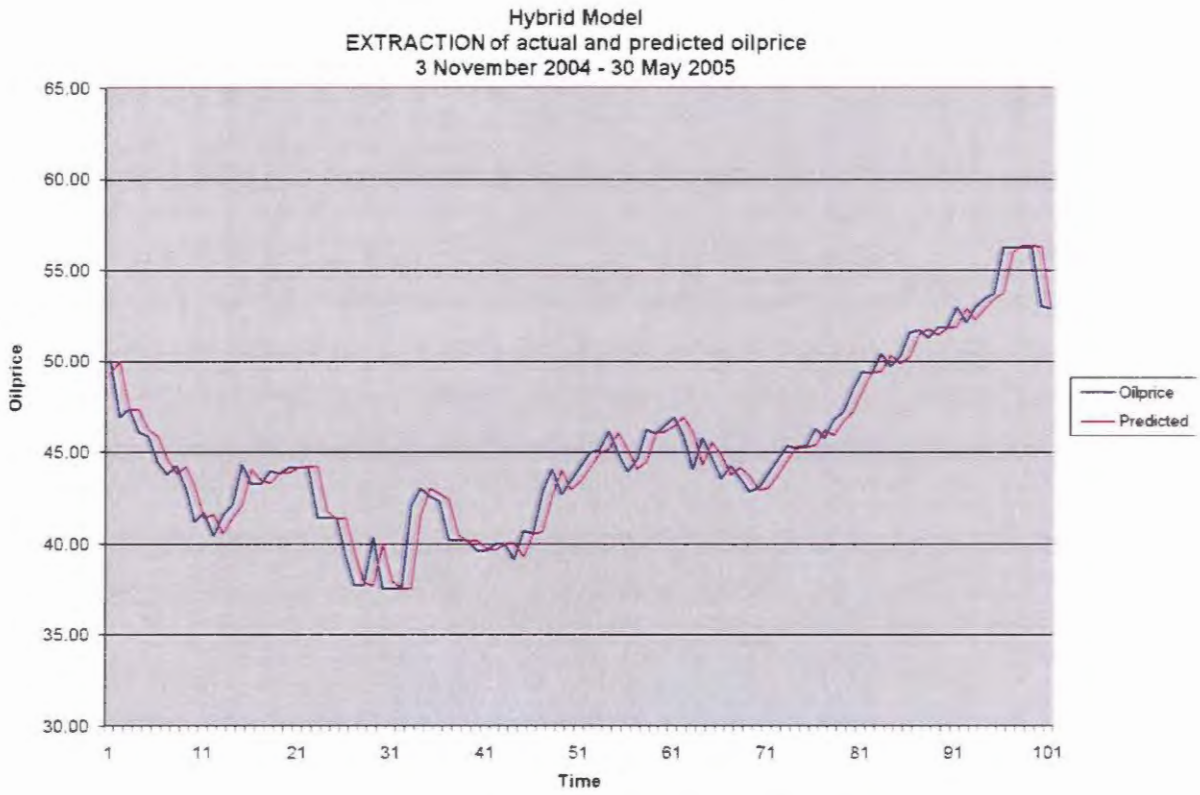


Figure 6.27 – Oil price: Hybrid model extraction

The MSE and MAD were

	MSE	MAD
Linear (damped exponential smoothing) forecast	1.30581	0.791416173
Neural network forecast	1.317175346	0.805147948
Hybrid forecast	**1.304536**	**0.790**

The hybrid forecast performed best but, consistent with some of the previous data sets, the differences between the MSE and MAD for the three forecasts were very small. The fact that the hybrid forecast is still the best, even though just marginally, may be an indication that a combination of forecasts should be considered.

6.3.4.2 Five-step-ahead forecasting (oil price)

For the five-step-ahead forecasting, the best neural network model to forecast the errors were one using a 2-period lagged observation and 1 hidden node. The average squared error for this model was 1.413214. A hybrid model was then once again constructed by combining the linear forecasts and the neural network forecasts of errors. Figure 6.28 shows a graph of the different forecasts for 5 periods ahead.

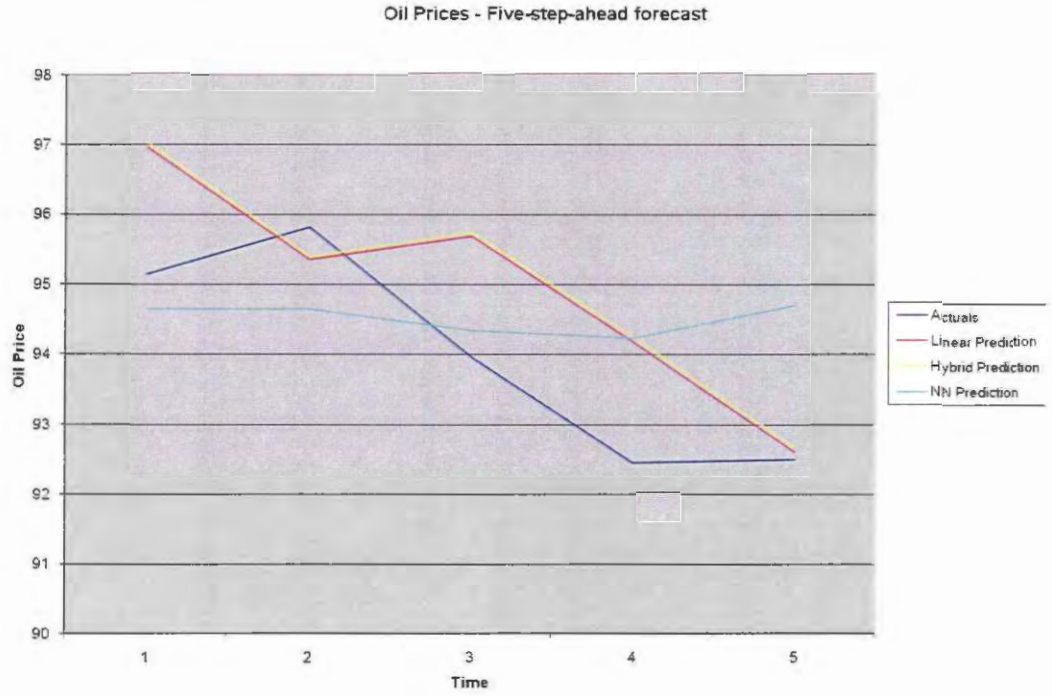


Figure 6.28 – Oil price: 5-step-ahead forecast

The MSE and MAD were as follows.

	MSE	MAD
Linear (damped exponential smoothing) forecast	**1.8977258**	**1.17014**
Neural network forecast	1.950057	1.20475
Hybrid forecast	2.035076466	1.2137672

The linear forecast performed best. Of all the experiments carried out in the entire study, this was the only case where the hybrid forecast performed worst of the three forecasts.

6.3.4.3 Twenty-step-ahead forecasting (oil price)

A grid search indicated that the best neural network model to forecast errors for a twenty-step-ahead forecasting was one using a 1-period lagged observation with 6 hidden nodes. The average squared error was 1.518942. Figure 6.29 shows a plot of the different forecasts for twenty periods ahead.

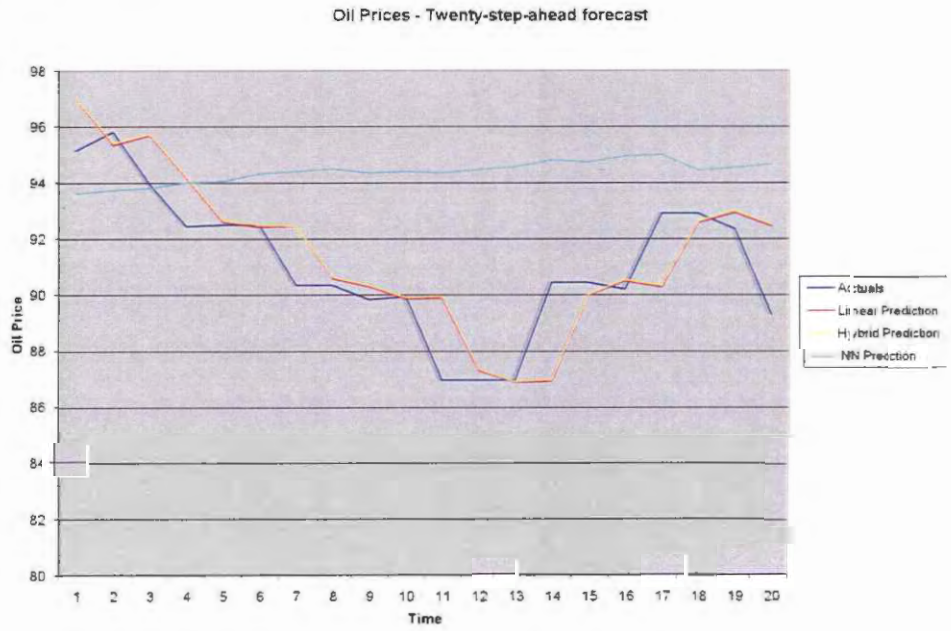


Figure 6.29 – Oil price: 20-step-ahead forecast

The MSE and MAD were as follows.

	MSE	MAD
Linear (damped exponential smoothing) forecast	2.656339924	1.15695
Neural network forecast	17.93737	3.652232
Hybrid forecast	**2.630927338**	**1.1155.20056**

The hybrid forecast was once again the best for a twenty-step-ahead forecast. The linear forecast was also good but the neural network performed poor in this case.

The experiment with the oil price data set showed that a hybrid forecast performed marginally better than the linear forecast when predicting 1 and 20 periods ahead. For the 5-period-ahead forecast the hybrid model performed worst. The results for the oil price data show that it may be worthwhile to combine forecasts but that it should also be treated with care as improved forecasts are not always achieved.

6.3.5 Money market returns time series data set

The SAS system suggested a log linear (Holt) exponential smoothing model as the best linear model to forecast the money market return. The statistics for this model were given as root mean square error = 4.99029; MSE = 24.90304; MAPE = 0.20451; and mean absolute error = 1.77123. The other parameters for the model were as follows.

Model Parameter	Estimate	Std error	T-value	Prob>|T|
LEVEL Smoothing Weight	0.999	0.0318	31.4322	0
TREND Smoothing Weight	0.001	0.0037	0.2698	0.78746322
Residual Variance (sigma squared)	0.0000346			
Smoothed Level	7.01751			
Smoothed Trend	0.0009912			

Statistic of Fit	Value
Number of Nonmissing Observations	498
Number of Observations	498
Number of Missing Actuals	0
Number of Missing Predicted Values	0
Number of Model Parameters	2
Total Sum of Squares (Uncorrected)	375610607

Total Sum of Squares (Corrected)	7776406
Sum of Square Error	12402
Mean Square Error	24.903042
Root Mean Square Error	4.990295
Mean Absolute Percent Error	0.204507
Mean Absolute Error	1.771225
R-Square	0.998405
Adjusted R-Square	0.998402
Amemiya's Adjusted R-Square	0.998392
Random Walk R-Square	0.002189
Akaike Information Criterion	1605.064999
Schwarz Bayesian Information Criterion	1613.486199
Amemiya's Prediction Criterion	25.103873
Maximum Error	53.120558
Minimum Error	-28.79078
Maximum Percent Error	5.652303
Minimum Percent Error	-3.7488
Mean Error	0.025646
Mean Percent Error	-0.001326

An extraction of the linear forecast, showing 101 data points (from the original 498 data points) is given in figure 6.30. The 101 data points cover the period 29 December 2006 to 29 May 2007.

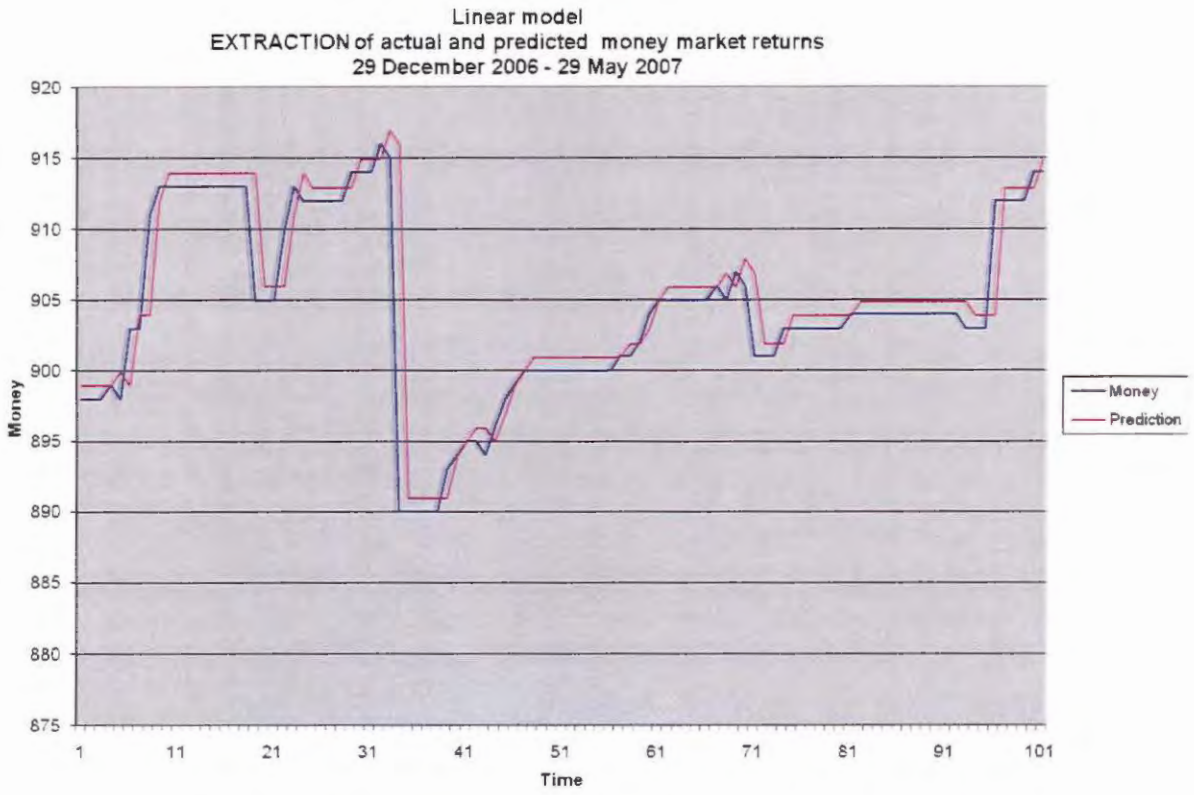


Figure 6.30 – Money market return: Linear model extraction

A grid search was once again performed to obtain the best neural network model to forecast the original data set. The best model found was one using a 1-period lagged observation with 6 hidden nodes. The average squared error was given as 20.78686. In figure 6.31 the extraction of the neural network forecasts is presented.

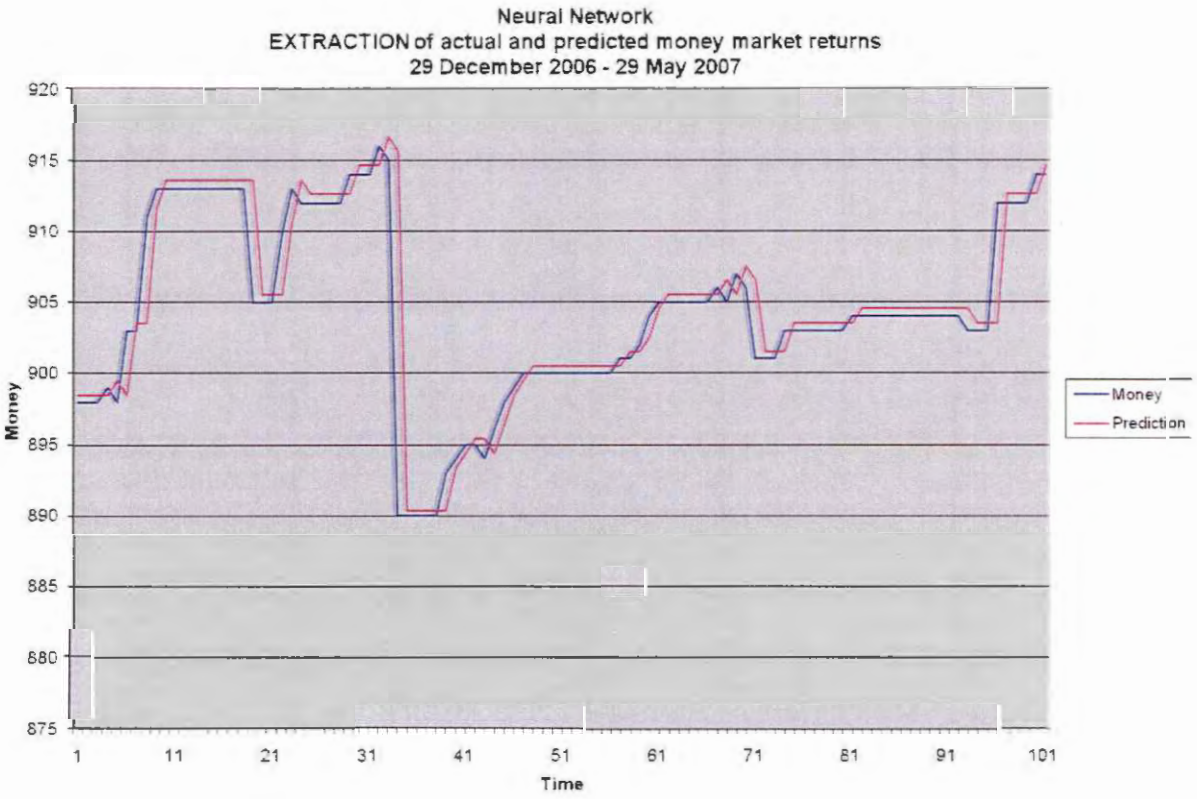


Figure 6.31 – Money market return: Neural network model extraction

6.3.5.1 One-step-ahead forecasting (money market)

The best neural network model to forecast the errors was one using a 9-period lagged observation with one hidden node. The average squared error was given as 22.12456. The linear forecasts and the errors forecasted by the neural network were then used to construct the hybrid model. The extraction is plotted in figure 6.32.

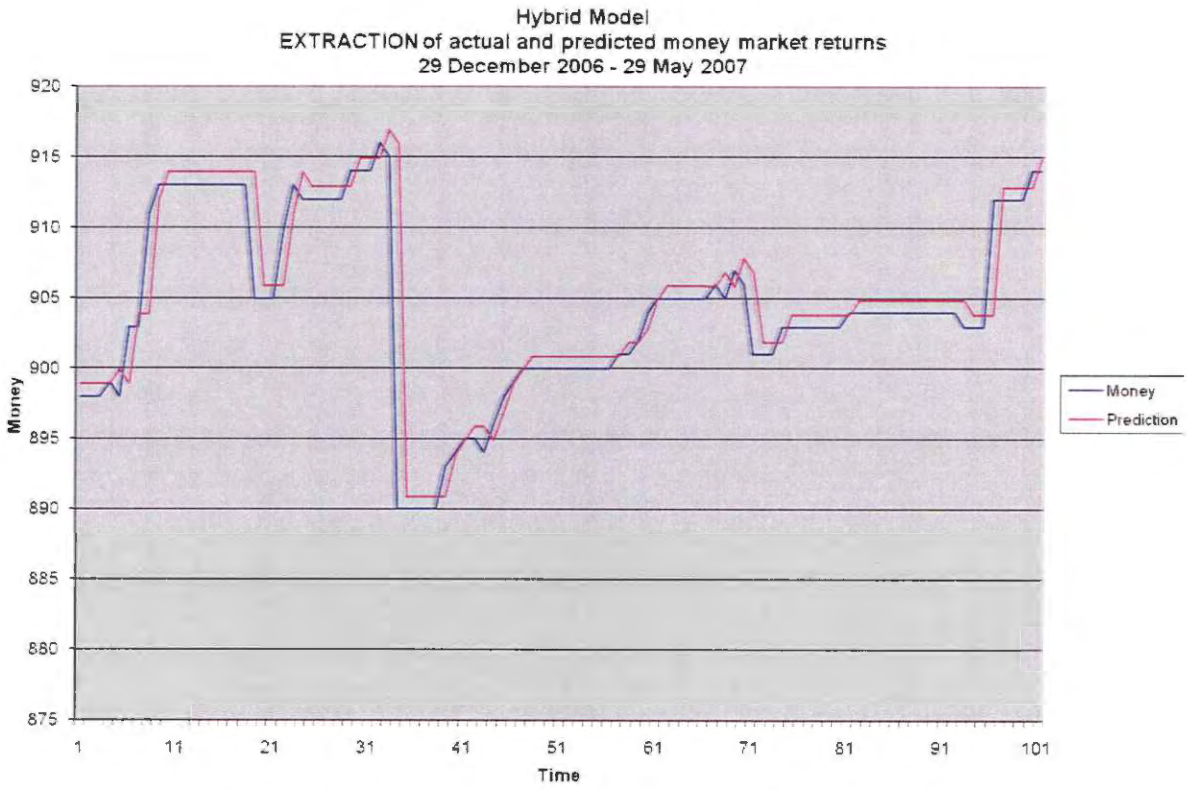


Figure 6.32 – Money market return: Hybrid model extraction

The MSE and MAD were as follows

	MSE	MAD
Linear (log linear Holt exponential smoothing) forecast	24.63179	**1.701708**
Neural network forecast	24.84483	1.716228
Hybrid forecast	**24.58424**	1.713458

The linear model performed best in terms of the MAD while the hybrid model performed best when using the MSE as an accuracy measure. The differences were once again very small and the only significant conclusion from this experiment is that the hybrid model did not perform worse than the other two forecasts.

6.3.5.2 Five-step-ahead forecasting (money market)

The best neural network model to forecast the errors, using a five-step-ahead approach, was a model with an 8-period lagged observation and 1 hidden node. The average squared error for this model was 42.91494. A hybrid model was then once again constructed by adding the linear forecasts and the neural network forecasts of errors together. Figure 6.33 shows a graph of the different forecasts for 5 periods ahead.

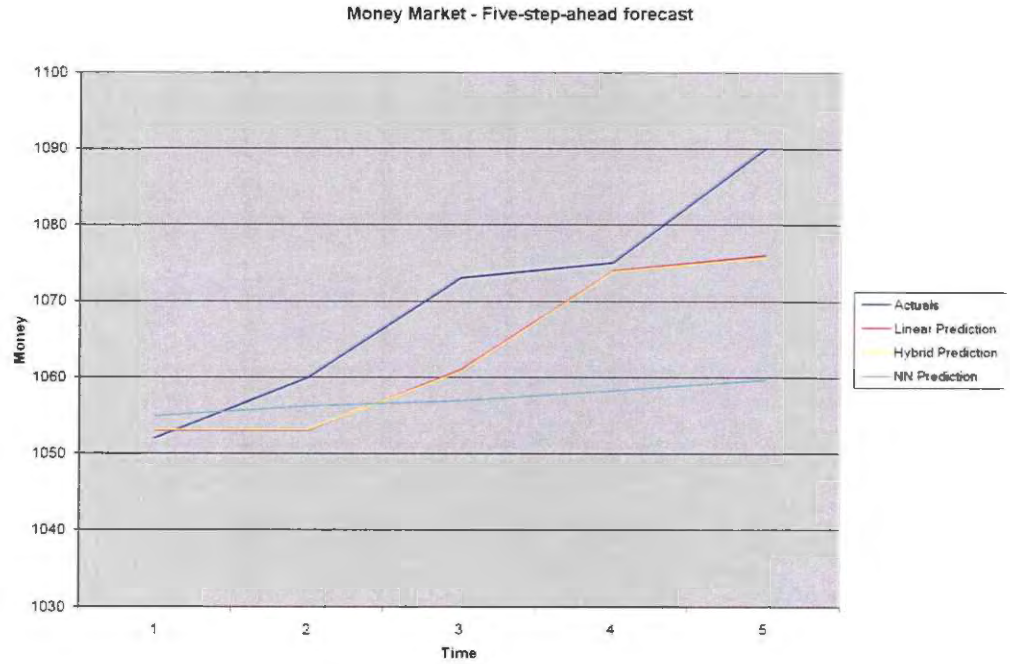


Figure 6.33 – Money market return: 5-step-ahead forecast

The MSE and MAD were as follows.

	MSE	MAD
Linear (log linear Holt exponential smoothing) forecast	**78.2**	**7.0**
Neural network forecast	295.6437	13.94431
Hybrid forecast	80.45477	7.104434

The results are consistent with the results for the gold price time series forecast using a five-step-ahead approach. There were large differences in the values of the five data points to be forecasted causing the neural network to perform extremely poor. The linear model performed best because each forecast is influenced by the previous observation.

6.3.5.3 Twenty-step-ahead forecasting (money market)

Following the usual grid search process, the best neural network model to forecast errors for a twenty-step-ahead forecasting was one using a 9-period lagged observation with 4 hidden nodes with an average squared error of 56.79348. Figure 6.34 prewsents a plot of the different forecasts for twenty periods ahead.

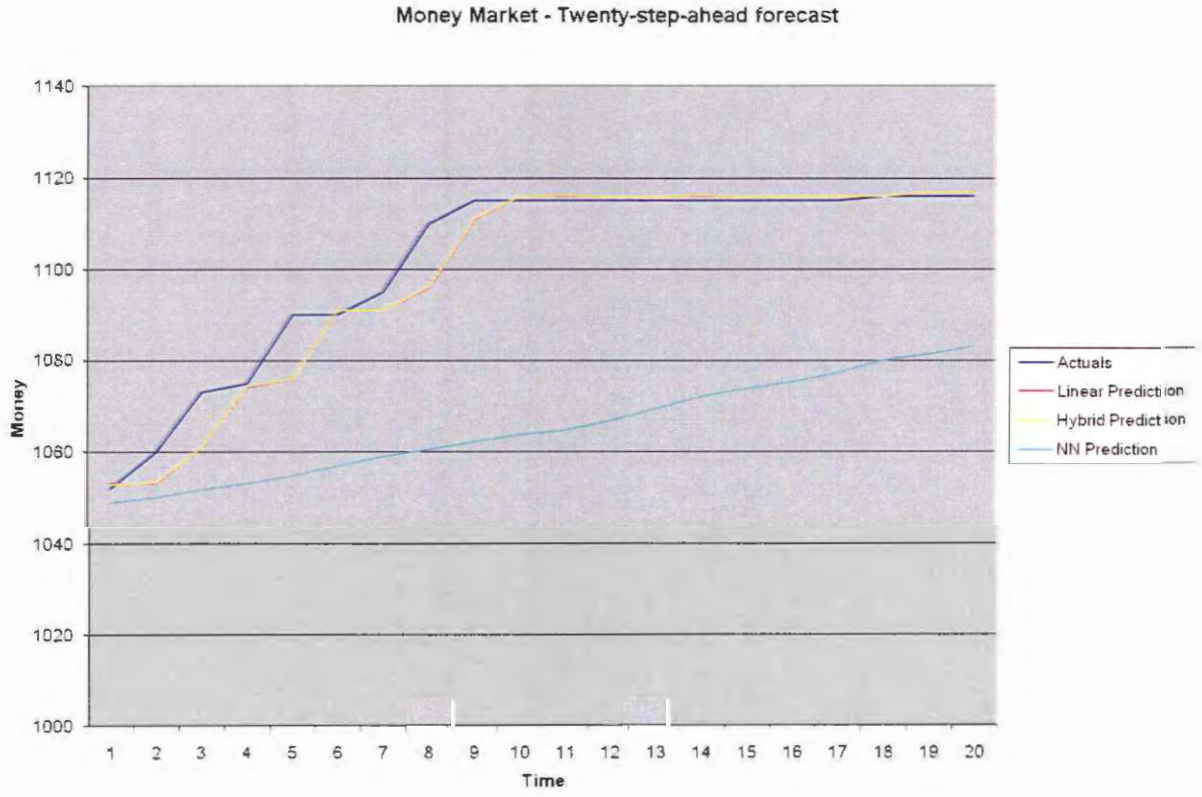


Figure 6.34 – Money market return: 20-step-ahead forecast

The MSE and MAD were as follows.

	MSE	MAD
Linear (log linear Holt exponential smoothing) forecast	31.5	3.4
Neural network forecast	1488.937	36.28583
Hybrid forecast	**30.72577**	**3.370916**

The same argument as for the five-period-ahead forecasting is applicable to explain the poor performance of the neural network model. In this case, however, the hybrid model performed best due to the longer forecast window available to forecast the errors more accurately.

As with the other experiments, this specific data set showed once again that it might be worthwhile to consider a combination of forecasts. This seems to be especially true when working with a longer forecasting period.

6.4 Summary discussion of results

Performing the empirical experiments with the five real world data sets involved a lengthy and intensive process of identifying suitable models to forecast future values. For example, to perform a single grid search process (described in chapter 5) to obtain the most suitable neural network architecture, took on average 10.5 hours to complete on a personal desktop computer. A total of more or less 310 hours were spent to perform all the required grid searches on a PC. Following all these experimental work, interesting results that were presented above in section 6.3, were obtained. These results were already discussed in section 6.3 and the purpose of this section is to summarize the results in one presentation.

- For the one-step-ahead forecasting approach, the hybrid model performed best in at least one of the accuracy measures. In one instance, significant improvements of up to 8.5% were noticed (gold price data). The exception was the demand for electricity data where the neural network model performed best with the hybrid model second best.

- In the case of a five-step-ahead forecasting it was noticed that (with the exception of the electricity data) the hybrid model did not perform particularly good and never gave the best forecast. It was also interesting to note that the linear model in most cases outperformed the other two models. It was further also observed that whenever there are big changes in the values, the linear models performed best because each forecasted value is influenced by the previous observed value. The time horizon (5 periods) is not long enough for a neural network or hybrid model to adjust forecasted values quickly enough which then resulted in higher forecast errors – see for example the results of the money market data presented in section 6.3.
- For the twenty-step-ahead forecasting, the hybrid model consistently outperformed the linear and neural network forecasts for all data sets.

The conclusions based on the results can be summarized as follows.

- It would be incorrect to claim that a combination (hybrid) of forecasts will always consistently outperform a single model. Results here have shown, especially in the five-step-ahead forecasts, that the linear model outperformed the hybrid model. This finding is in line with other studies that also obtained mixed results (Taskaya-Temizel and Casey, 2005).
- In only one of the cases (the five-step-ahead forecasting of the oil price), the hybrid model performed the worst. In all other cases the hybrid forecasts were either the best or the second best forecast. This leads to the conclusion that it is definitely worthwhile to investigate and apply the type of combination described in this study.
- Depending on the lead time to forecast and the observed data it might be worthwhile to use a linear model when the lead time is short and when there are significant differences in the observed data points.
- On the other hand, if the lead time becomes longer it becomes more important to consider the use of a hybrid model to improve forecasts.

In general, the empirical experiments in this study have shown that the use of the type of hybrid model described should definitely be considered. Results should however be treated with care but at the same time it is also clear that significant improvements are possible.

6.5 Conclusion

Chapter 6 presented the empirical results and a discussion of the results. First, the five real world data sets used in the study were presented and then the modeling and forecast results were given. Graphs, accuracy measures and discussions were presented for each one of the data sets. The chapter was concluded with some general remarks based on the findings. The general conclusion was that the type of hybrid model, as described in this study, does present advantages and will improve forecasts under certain circumstances.

The next chapter presents the conclusion for the study and will summarize the goals set forth for the study and how they were achieved.

CHAPTER 7

SUMMARY AND CONCLUSION

7.1 Introduction

Chapter 7 presents the final comments and concluding remarks for the study. The objectives of the study and how they were achieved will be summarized. New problems and opportunities for further study that presented itself during the research project will also be outlined.

7.2 Objectives of the study

Chapter 1 stated that the primary objective of this study was to investigate the use of a combined linear and neural network model in order to determine the forecasting performance of such a hybrid model. To accomplish this, a list of five secondary research objectives were defined that needed to be achieved. These five objectives were

- To gain a clear understanding of and to present an introductory overview of time series analysis and forecasting models;
- To gain a clear understanding of and to present an introductory overview of neural network models;
- To gain a clear understanding of and to present a brief introduction to the well known Box-Jenkins approach;
- To investigate, describe and formulate a combined linear and neural network model; and
- To investigate the performance and forecasting accuracy of the combined model.

A summary of how these objectives were achieved will now be given.

To gain a clear understanding of and present an introductory overview of time series analysis and forecasting models.

Time series forecasting is an important area of forecasting and can be modeled in a variety of ways. The models used in this study were empirically tested on time series data and it is therefore imperative to have a clear understanding of time series analysis and forecasting models.

The objective was achieved by describing time series forecasting models in terms of components and decomposition of a time series, trend analysis e.g. the moving average method and time series regression, seasonal analysis and how to construct a forecast (Chapter 2, sections 2.2.1 – 2.2.5). Measures of forecast accuracy were presented (Chapter 2, section 2.3) as well as an extensive discussion of exponential smoothing (Chapter 2, section 2.4).

To gain a clear understanding of and to present an introductory overview of neural network models.

The primary objective is to investigate the use of neural network models in combination with time series forecasting. Neural networks therefore play a major role in the study and it is essential that the general concepts and how they work are well understood.

This objective was reached by giving a fairly comprehensive discussion on "what" a neural network is (Chapter 3, section 3.2). The architecture, which refers to the arrangement of layers and the connection patterns, were also explained (Chapter 3, section 3.3) while the training and learning concepts were made clear through definitions and the presentation of a training algorithm (Chapter 3, section 3.4). Finally the most common activation functions used in neural networks were highlighted (Chapter 3, section 3.5).

To gain a clear understanding of and to present a brief introduction to the Box-Jenkins approach.

The Box-Jenkins analysis, or ARIMA analysis, is one of the well known approaches for forecasting time series. ARIMA models are often used in combination with other models, such as neural network models, to construct hybrid forecasts. As this study also concentrates on combining linear and neural network models, it is necessary to gain appropriate background knowledge on the Box-Jenkins approach.

The objective was achieved by discussing the four stages in the Box-Jenkins approach namely model identification, estimation, diagnostic checking and forecasting (Chapter 4, sections 4.2 – 4.5). Appropriate examples of SAS output during the different stages were also presented.

To investigate, describe and formulate a combined linear and neural network model.
It is important to have a clear understanding of the methodology that will be followed in the study as well as to ensure that the practical work is relevant.

The objective was achieved by firstly motivating the relevance of the suggested technique from appropriate literature resources (Chapter 5, section 5.2) and secondly, by giving a comprehensive overview of the research design and methodology used. This overview included a discussion on the hybrid methodology as well as the approach followed with the empirical experiments (Chapter 5, sections 5.3.1 – 5.3.2).

To investigate the performance and forecasting accuracy of the combined model.
Models that forecast future values of a time series need to be evaluated to ensure that forecasting performance is acceptable. In this study a hybrid model was constructed and to be able to make meaningful recommendations and conclusions, the model's performance needed to be evaluated.

This objective was achieved by using five real world data sets and for each data set applies a linear forecast, a neural network forecast and a combined forecast to the data set. These forecasts were done for a one, five and twenty period ahead and each time the MSE and MAD were used as a forecast accuracy measure. Comprehensive results including graphs, forecasts, accuracy measures etc. were presented. Finally it was possible to make interesting and significant findings and recommendations based on these results. All information concerning this final objective was presented in chapter 6.

7.3 Problems experienced

Consistent with the comment made by De Gooijer and Hyndman (2006) that there is a lack of empirical research on robust forecasting algorithms for multivariate models, it was found in this study that there is not sufficient empirical evidence to which the study's results can be compared with. To compare one's results with other results (using similar type of data sets) is one way of validating work that was performed. It is hoped that the results of the particular data sets used in

this study can be used by other researchers to compare and validate results from different experiments.

Another difficulty (to a lesser extent) experienced was the lack or unavailability of existing software that can be used when constructing hybrid models. For example, programs and/or procedures to perform grid searches (to determine the "best" neural network model) and to do the actual combination of forecasts were not available and had to be developed in order to perform the study in a meaningful way.

7.4 Possibilities for further studies

In this study, raw data was used exactly as it was recorded. It might be worthwhile to conduct the same type of experimental work using time series data sets that were explored in more depth for inefficiencies – e.g. smoothing the data or maybe removing any seasonal influences before the models are constructed and combined.

The differentiation and use of different neural network models should be investigated. For example, is a recursive multi-step approach (based on a single step ahead forecast – one output node – where previous predictions are used as inputs for subsequent forecasts) better than a direct multi-step approach (based on several output nodes where each output node represent a time step to be forecasted)? In addition to this, it may also be possible to improve the neural network models by experimenting with different training, validation and test sizes. More intensive grid searches (e.g. increasing the number of experiments) may also have an impact on the neural network model's ability to predict future values.

Finally, an investigation into and development of easy to use software that can assist researchers with building hybrid forecast models, would assist many researchers and may lead to an increase in research projects in this interesting and important area of model building and forecasting.

7.5 Conclusion

Chapter 7 is the final chapter of this study. The chapter presented a summary of the initial objectives and how they were achieved. In conclusion, problems and possible future research opportunities were outlined.

Bibliography

Awad, E.M. 1996. Building expert systems: principles, procedures and applications. West publishing company. Minneapolis. 638p.

Bodyanskiya, Y & Popov, S. 2006. Neural network approach to forecasting of quasiperiod financial time series. *European Journal of Operational Research.* 175(3):1357-1366.

Bowerman, B.L., O'ConnelL, R.T. & Koehler, A.B. 2005. Forecasting, time series and regression. An applied approach. 4th Edition. Thomson Brookes/Cole. 517p.

Box, G.E.P. & Jenkins, G.M. 1970. Time series analysis: Forecasting and control. San Francisco: Holden Day (revised ed. 1976).

Chelani, A.B. & Devotta, S. 2006. Air quality forecasting using a hybrid autoregressive and nonlinear model, *Atmospheric Environment.* 40:1774-1780.

Cho, V. 2003. A comparison of three different approaches to tourist arrival forecasting, *Tourism Management.* 24:323-330.

Co, H.C. & Boosarawongse, R. 2007. Forecasting Thailand's rice export: Statistical techniques vs. artificial neural networks, *Computers & Industrial Engineering.* 53:610-627.

De Gooijer, J.G. & Hyndman, R.J. 2006. 25 years of time series forecasting, *International journal of Forecasting.* 22:443-473.

Dooley, G. & Lenihan, H. 2005. An assessment of time series methods in metal price forecasting, *Resources Policy.* 30:208-217.

Ediger, V.S. & Akar, S. 2007. ARIMA forecasting of primary energy demand by fuel in Turkey, *Energy Policy.* 35:1701-1708.

Fausett, L. 1994. Fundamentals of neural networks. Prentice Hall. Upper Saddle River, New Jersey. 461p.

Gareta, R., Romeo, L.M. & Gil, A. 2006. Forecasting of electricity prices with neural networks. *Energy Conversion and Management.* 47(13):1770-1778.

Ghiassi, M., Saidane, H. & Zimbra, D.K. 2005. A dynamic artificial neural network model for forecasting time series events, *International journal of Forecasting.* 21:341-362.

Gujarati, D.N. 2003. Basic econometrics, 4th edition. McGraw-Hill, New York. 1002p.

Ho, S.L., Xie, M. & Goh, T.N. 2002. A comparative study of neural network and Box-Jenkins ARIMA modeling in time series prediction, *Computers & Industrial Engineering.* 42:371-375.

Ince, H. & Trafalis, T.B. 2006. A hybrid model for exchange rate prediction, *Decision Support Systems.* 42:1054-1062.

Makridakis, S., Wheelwright, S.C. & McGee, V.E. 1983. Forecasting: Methods and Applications. John Wiley and Sons, New York. 926p.

Mishra, A.K. & Desai, V.R. 2006. Drought forecasting using feed-forward recursive neural network, *Ecological Modeling.* 198:127-138.

Moore, J.H. & Weatherford, L.R. 2001. Decision modeling with Microsodt Excel, 6th edition. Prentice Hall. Upper Saddle River, New Jersey. 693p.

Pai, P.F. & Lin, C.S. 2005. A hybrid ARIMA and support vector machines model in stock pricing forecasting, *Omega.* 33:497-505.

Pankratz, A. 1983. Forecasting with univariate Box-Jenkins models. Concepts and cases. John Wiley and Sons, New York. 562p.

Payne, G.O. 2001. Forecasting South African financial indicators using neural networks. MSc thesis, Potchefstroom University for Christian Higher Education. 181p.

Prybutok, V.R., Yi, J. & Mitchell, D. 2000. Comparison of neural network models with ARIMA and regression models for prediction of Houston's daily maximum ozone concentrations, *European Journal of Operational Research.* 122:31-40.

Render, B., Stair, R.M. & Hanna, M.E. 2006. Quantitative analysis for management, 9th edition. Prentice Hall. Upper Saddle River, New Jersey. 731p.

Rojas, I., Valenzuela, O., Rojas, F., Guillen, A., Herrera, L.J., Pomares, H., Marquez, L. & Pasadas, M. 2008. Soft-computing techniques and ARMA model for time series prediction, *Neurocomputing.* 71:519-537.

Taskaya-Temizel, T. & Casey, M.C. 2005. A comparative study of autoregressive neural network hybrids, *Neural Networks.* 18:781-789.

Taylor, B.W. 2002. Introduction to Management Science, 7th edition. Prentice Hall. Upper Saddle River, New Jersey. 765p.

Taylor, J.W., De Menezes, L.M. & McSharry, P.E. 2006. A comparison of univariate methods for forecasting electricity demand up to a day ahead, *International journal of Forecasting.* 22:1-16.

Tseng, F.M., Yu, H.C. & Tzeng, G.H. 2002. Combining a neural network model with seasonal time series ARIMA model, *Technological Forecasting & Social Change.* 69:71-87.

Ture, M. & Kurt, I. 2006. Comparison of four different time series methods to forecast hepatitis A virus infection, *Expert systems with applications.* 31:41-46.

Valenzuela, O., Rojas, I., Rojas, F., Pomares, H., Herrera, L.J., Guillen, A., Marquez, L. & Pasadas, M. 2008. Hybridization of intelligent techniques and ARIMA models for time series prediction, *Fuzzy sets and Systems.* 159:821-845.

Wegner, T. 1993. Applied business statistics. Methods and applications. Juta & Co, Ltd. 488p.

Zhang, G.P. 2003. Time series forecasting using a hybrid ARIMA and neural network model, *Neurocomputing.* 50:159-175.

Zou, H.F., Xia, G.P., Yang, F.T. & Wang, R. 2007. An investigation and comparison of artificial neural network and time series models for Chinese food grain price forecasting, *Neurocomputing.* 70:2913-2923.

APPENDIX A

Data set used for explanatory purposes in chapter 4.

Weekly sales figures of paper towels (Bowerman et al, 2005)

t	y_t	t	y_t	t	y_t	t	y_t
1	15	31	10.7752	61	-1.3173	91	10.5502
2	14.4064	32	10.1129	62	-0.6021	92	11.4741
3	14.9383	33	9.933	63	0.14	93	11.5568
4	16.0374	34	11.7435	64	1.403	94	11.7986
5	15.632	35	12.259	65	1.928	95	11.8867
6	14.3975	36	12.5009	66	3.5626	96	11.2951
7	13.8959	37	11.5378	67	1.9615	97	12.7847
8	14.0765	38	9.6649	68	4.8463	98	13.9435
9	16.375	39	10.1043	69	6.5454	99	13.6859
10	16.5342	40	10.3452	70	8.0141	100	14.1136
11	16.3839	41	9.2835	71	7.9746	101	13.8949
12	17.1006	42	7.7219	72	8.4959	102	14.2853
13	17.7876	43	6.83	73	8.4539	103	16.3867
14	17.7354	44	8.2046	74	8.7114	104	17.0884
15	17.001	45	8.5289	75	7.378	105	15.8861
16	17.7485	46	8.8733	76	8.1905	106	14.8227
17	18.1888	47	8.7948	77	9.972	107	15.9479
18	18.5997	48	8.1577	78	9.693	108	15.0982
19	17.5859	49	7.9128	79	9.4506	109	13.877
20	15.7389	50	8.7978	80	11.2088	110	14.2746
21	13.6971	51	9.0775	81	11.4986	111	15.1682
22	15.0059	52	9.3234	82	13.2778	112	15.3818
23	16.2574	53	10.4739	83	13.591	113	14.1863
24	14.3506	54	10.6943	84	13.4297	114	13.9996
25	11.9515	55	9.8367	85	13.3125	115	15.2463
26	12.0328	56	8.1803	86	12.7445	116	17.0179
27	11.2142	57	7.2509	87	11.7979	117	17.2929
28	11.7023	58	5.0814	88	11.7319	118	16.6366
29	12.5905	59	1.8313	89	11.6523	119	15.341
30	12.1991	60	-0.9127	90	11.3718	120	15.6453

APPENDIX B

Source code of SAS program to perform a grid search in order to determine the best neural network model – see chapter 5 section 3.2.

```
options nosource nonotes nodate;
* options mlogic mprint symbolgen;
options mstored sasmstore=gann;
options pagesize=32767 mvarsize=max;
ods listing close;

%macro settime;

  %let ttime = %sysfunc(datetime());

%mend settime;

%macro gettime;

  %let tdelay = %sysevalf(%sysfunc(datetime()) - &ttime);

%mend gettime;

%macro showtime;

  %local a b c d e;

  %let a = &tdelay;
  %let b = %sysevalf(&a / 86400, integer);
  %let a = %sysevalf(&a - 86400 * &b);
  %let c = %sysevalf(&a / 3600, integer);
  %let a = %sysevalf(&a - 3600 * &c);
  %let d = %sysevalf(&a / 60, integer);
  %let e = %sysevalf(%sysfunc(round(&a - 60 * &d, 0.01)));

  %put ;
  %put elapsed time = &b:&c:&d:&e;
  %put ;

%mend showtime;

%macro beep;

  data _null_;
    call sound(400,80);
  run;
```

```
%mend beep;

%macro partition2(source,p1,p2,set1,set2);

  %let seed = 0;

  %obsnvars(&source,nvars,nobs);

  %let cutoff1 = %sysevalf(&nobs * &p1 / 100, ceil);
  %let cutoff2 = %eval(&nobs - &cutoff1);

  data &set1 &set2;
    drop _c00: ;
    set &source;
    if (ranuni(&seed) * 1000 < %sysevalf (&p1 * 10, ceil)
      and _c000001 < &cutoff1) then do;
        _c000001 + 1;
        output &set1;
    end;
    else
    if _c000002 < &cutoff2 then do;
      _c000002 + 1;
      output &set2;
    end;
    else do;
      _c000001 + 1;
      output &set1;
    end;
  run;

%mend partition2;

%macro partition3(source,p1,p2,p3,set1,set2,set3);

  %let total1 = %sysevalf(&p2 + &p3);
  %partition2(&source, &p1, &total1, &set1, gann.temp9999);
  %let total2 = %sysevalf(&p2 / (&p2 + &p3) * 100, integer);
  %let total3 = %sysevalf(100 - &total2);
  %partition2(gann.temp9999, &total2, &total3, &set2, &set3);

  proc datasets nolist library=gann;
    delete temp9999;
  run;

%mend partition3;
```

```
%macro importdata;

    proc import out = gann.&sasdataset
      datafile = &excelfile
      dbms = excel replace;
      sheet = &excelsheet;
      getnames = yes;
      mixed = no;
      scantext = yes;
      usedate = yes;
      scantime = yes;
    run;

%mend importdata;

%macro obsnvars(ds,nvarsp,nobsp);

  %global dset nvars nobs;
  %let dset = &ds;
  %let dsid = %sysfunc(open(&dset));
  %if &dsid %then
    %do;
      %let nobs =%sysfunc(attrn(&dsid,NOBS));
      %let nvars=%sysfunc(attrn(&dsid,NVARS));
      %let rc = %sysfunc(close(&dsid));
    %end;
  %else
    %put Open for data set &dset failed - %sysfunc(sysmsg());

%mend obsnvars;

%macro convertdata(windowsize);

    %local i j k m n;

    %obsnvars(gann.&sasdataset,nvars,nobs);

    proc datasets library=gann nolist;
        delete InputData temp;
    run;

    %do i = 1 %to %eval(&nobs - &windowsize);

    proc sql noprint;
        select &targetvar into :k separated by ' ' from
```

```
        gann.&sasdataset (firstobs=&i obs=%eval(&i+&windowsize));
    quit;

    %let n = ;

    %do j = 1 %to &windowsize;

        %let m = %scan(&k, &j, ' ');

        %let n = &n format input&j best12. %str(;);
        %let n = &n input&j = &m %str(;);

    %end;

    %let m = %scan(&k, %eval(&windowsize + 1), ' ');

    %let n = &n format target best12. %str(;);
    %let n = &n target = &m %str(;);
    %let n = &n &timevar = %eval(&i + &windowsize) %str(;);

        data gann.temp;
            &n
        run;

        proc append base=gann.InputData data=gann.temp;
        run;

    %end;

%mend convertdata;

%macro fitmlp;

    %local i;

    %let vars = ;

    %do i = 1 %to &windowperiod;
        %let vars = &vars input&i;
    %end;

    proc dmdb batch data=gann.&trainset out=gann.dus dmdbcat=gann.cus;
        var &vars target ;
    run;

    proc neural data=gann.&trainset dmdbcat=gann.cus random=12345;
```

```
        input &vars / level=int id=in;
        target target / level=int id=out act=linear error=normal bias;
        archi mlp hidden = &hidden;
        train outfit=gann.trainfit maxiter=2000;
        score data=gann.&validateset nodmdb out=gann.&scoreset outfit=gann.validatefit
role=validation;
    run;

%mend fitmlp;

%macro gridsearch;

    %local i j k;

    %settime;

    %importdata;

    data gann.randdollarforecast;
        set gann.randdollar (firstobs=1230 obs=1249);
    run;

    data gann.randdollarbase;
        set gann.randdollar (firstobs=1 obs=1229);
    run;

    %let sasdataset = randdollarbase;

    proc datasets nolist library=gann;
        delete results;
    run;

    %do windowperiod = 19 %to 19; /* Stel vensterperiode */

        %convertdata(&windowperiod);

        %do hidden = 3 %to 3; /* Stel hidden node */

            proc datasets nolist library=gann;
                delete experiment;
            run;

            %do exper = 1 %to 20; /* Stel aantal eksperimente */

                %put [ Windowperiod = &windowperiod | Hidden = &hidden | Experiment =
&exper ];
```

```
        %partition2(gann.inputdata, 90, 10, gann.&trainset, gann.&validateset);

        %fitmlp;

        data gann.temp;
            set gann.validatefit (firstobs=2 obs=2 drop=_iter_ _name_);
        run;

        proc append base=gann.experiment data=gann.temp;
        run;

    %end;

ods output Variables=gann.variables;

proc contents data=gann.experiment;
run;

%obsnvars(gann.variables, nvars, nobs);

%do i = 1 %to &nobs;

  data _null_;
    set gann.variables;
    if num = &i then do;
      call symput('j', variable);
    end;
  run;

    proc sql noprint;
        select mean(&j) into :k from
            gann.experiment;
    quit;

  data gann.experiment;
    set gann.experiment;
    if _n_ = 1 then &j = &k;
  run;

%end;

data gann.experiment;
  set gann.experiment (firstobs=1 obs=1);
run;
```

```
        data gann.temp;
            set gann.experiment;
            Hidden = &hidden;
            Window = &windowperiod;
        run;

        proc append base=gann.results data=gann.temp;
        run;

        %end;

    %end;

    %gettime;

    %showtime;

%mend gridsearch;

%macro graph;

    %local i;

    ods listing;

    /*

    %let windowperiod = 8;

    %let hidden = 25;

    %importdata;

    data gann.randdollar;
        set gann.randdollar (firstobs=2 obs=499);
    run;

    %convertdata(&windowperiod);

    %partition2(gann.inputdata, 95, 5, gann.&trainset, gann.&validateset);

    %let targetvar = error;

    %fitmlp;

    data gann.final;
```

```
    merge gann.randdollar gann.&scoreset;
    by time;
run;

data gann.final2;
    set gann.final;
    if target ne . then output;
run;

%let windowperiod = 4;

%let hidden = 3;

%let targetvar = prys;

%convertdata(&windowperiod);

%let windowperiod = 8;

%let temp = ;

%do i = 1 %to &windowperiod;
    %let temp = &temp input&i;
%end;

data gann.trainidx;
    set gann.&trainset;
    drop &temp target;
    abc = _n_;
run;

data gann.trainnew;
    merge gann.trainidx gann.inputdata;
    by &timevar;
run;

data gann.trainsetnew;
    set gann.trainnew;
    if abc ne . then output;
    drop abc;
run;

data gann.validatesetnew;
    set gann.trainnew;
    if abc eq . then output;
    drop abc;
```

```
run;

*/

data gann.validatesetnew;
    set gann.validatesetnew (firstobs=5 obs=102);
run;

%let trainset = trainsetnew;
%let validateset = validatesetnew;
%let hidden = 3;
%let windowperiod = 4;
%let scoreset = score2;

%fitmlp;

data gann.score1b;
    merge gann.score gann.randdollar;
    by time;
run;

data gann.score1b;
    set gann.score1b;
    if target ne . then output;
run;

data gann.score1b;
    set gann.score1b;
    keep time forecast error p_target;
run;

data gann.score2b;
    merge gann.score2 gann.randdollar;
    by time;
run;

data gann.score2b;
    set gann.score2b;
    if target ne . then output;
run;

data gann.score2b;
    set gann.score2b;
    keep time r_target p_target target;
run;
```

```
    goptions
    reset=all
    device=gif
    gsfmode=replace
    goutmode=append
    ftext=swissb;

  symbol1 c=black v=none i=spline;
  symbol2 c=red v=none i=spline;

  proc gplot data=gann.final2;
    plot
        target * &timevar = 1
        p_target * &timevar = 2
        / overlay frame;
  run;

    ods listing close;

%mend graph;

%macro main;

    libname gann 'c:\SasRandDollar';

    %global excelfile excelsheet targetvar timevar sasdataset;
    %global hidden exper windowperiod;
    %global ttime tdelay;
    %global trainset validateset scoreset;

    %let excelfile = "c:\SasRandDollar\RandDollar.xls";
    %let excelsheet = "data$";
    %let targetvar = randprice;
    %let timevar = time;
    %let sasdataset = randdollar;
    /* %importdata; */
    %let trainset = trainset;
    %let validateset = validateset;
    %let scoreset = score;

    %gridsearch;

    /*

    data gann.resultsError;
        set gann.results;
```

```
    run;

    %let targetvar = prys;

    %gridsearch;

    data gann.resultsPrys;
        set gann.results;
    run;

    */

%mend main;

%main;
```