Non Linear Modelling of Financial Data Using Topologically Evolved Neural Network Committees

Thesis Submitted for the Degree of
Doctor of Philosophy
of the University of London

Ioannis Fotinopoulos

Centre for Quantitative Finance
Imperial College London
University of London

January 2008
Στη Μπέττυ ...
Abstract

Most neural network modelling methods are difficult to use as maximising or minimising an objective function in a non-linear context involves complex optimisation algorithms. Problems related to the efficiency of these algorithms are often mixed with the difficulty of the a priori estimation of a network's fixed-topology for a specific problem making it even harder to appreciate the real power of neural networks. In this thesis, we propose methods that overcome these issues by optimising a network's topology and weights, simultaneously.

When the data is high dimensional, modelling its often sophisticated behaviour is a very complex task that requires the optimisation of thousands of parameters. To enable optimisation techniques to overcome their limitations or failure, practitioners use methods to reduce the dimensionality of the data space. However, some of these methods are forced to make unrealistic assumptions when applied to non-linear data while others are very complex and require a priori knowledge of the intrinsic dimension of the system which is usually unknown and very difficult to estimate.

The proposed methods are non-linear and reduce the dimensionality of the input space without any information on the system's intrinsic dimension. This is achieved by first searching in a low dimensional space of simple networks, and gradually making them more complex as the search progresses. The high dimensional space of the final solution is only encountered at the very end of the search. This increases the system's efficiency by guaranteeing that the network becomes no more complex than necessary.

The modelling performance of the system is further improved by searching not only for one network as the ideal solution to a specific problem, but a combination of networks. These committees of networks are formed by combining a diverse selection of networks from a population of networks derived by the proposed method. This approach automatically exploits the strengths and weaknesses of each member of the committee while avoiding having all members giving the same bad judgements at the same time.

In this thesis, the proposed methods are used in the context of non-linear modelling of high-dimensional hedge fund returns. Experimental results are encouraging as both robustness and complexity are concerned.
Acknowledgements

First and foremost I would like to express my gratitude and respect for my supervisor Prof. Nicos Christofides for providing me with the academic platform to understand the main dynamics of the so complicated financial world. Over the last years Prof. Christofides has inspired me not only as an academic supervisor but also as a human being. I have learnt lots from him and I am forever indebted for his continuous intellectual and psychological support. Nicos has made me realise that one can always find a solution to almost everything as long as he has the willingness to learn and work hard. This has now become my life’s motto...

One thing that is certain is that I would have never written this thesis without the support from my sponsors, Imperial Chemical Industries plc (ICI) and its people. In particular, I owe a big thanks to Ian Ladd, David Blackwood, Fionnuala Gormley, Robert Mc Anally, Steve Ray and the rest of the treasury team for offering me an intellectually stimulating environment to work and study in. They were there to give me vital advice when most needed and make me develop my professional profile in the most effective manner. They were colleagues, advisors and most of all friends that I will always remember with great respect.

I would also like to thank my colleagues at Imperial College and in particular Gerry Salkin, Prof. Nigel Meade and Alex Christofides for their academic and psychological help.

A special thanks goes to my good friend Kostas, my parents and sister who were there during my most difficult times and gave me comfort and support.

Last but not least, there are no words to express my gratitude to my wife, Betty, for her endless love and understanding. Her tremendous patience and continuous encouragement have played a major role in the successful completion of this thesis. She has been there for me since the beginning of this Ph.D. and has followed me through all the ups and downs always with a smile. The least I can do is dedicate this thesis to her with all my love...
Statement of Originality

The material in chapter 4, chapter 5, chapter 6 and chapter 7 of this thesis are, as far as the author is aware, original contributions to the areas of dimensionality reduction of complex systems, non-linear modelling of financial time series and replication of hedge fund returns. The most significant contributions are the following:

In chapter 4 we present a novel hybrid methodology that exploits the advantages of linear and non-linear techniques in order to effectively reduce the dimensionality of complex systems. In this new approach we exploit the linear dependencies of the data by using independent component analysis to decompose the data into “meaningful” linear driving factors and then utilise the non-linear nature of neural networks to discover the non-linearities in the residuals of the linearly reconstructed system. In chapter 5, the methodology’s implementation procedure is presented and experimental results prove its effectiveness and robustness.

In chapters 4 and 6 we propose two approaches to extend a neural network’s ability to forecast by introducing neural network committees. The proposed committees are composed of several networks and a method for combining their outputs. Each individual network receives the same input and outputs its own decision. Based on outputs from several different networks, the combination algorithm decides the final output. Ideally, the performance of the entire committee will be superior to the performance of any individual network it contains.

In chapter 6 we present a novel mathematical formulation that is applied to the very popular problem of effective replication and forecasting of hedge fund returns. The replication part is implemented by mixed integer quadratic programming and an optimisation heuristic and the forecasting part by committees of neural networks.

In chapter 7 we test the above model with real hedge fund data and discuss practical issues. Experimental results are shown which are very encouraging as far as both robustness and computational complexity are concerned.
# List of Contents

ABSTRACT  ......................................................................................................................... iii

ACKNOWLEDGEMENTS ........................................................................................................ iv

STATEMENT OF ORIGINALITY ........................................................................................... v

1. Introduction ......................................................................................................................... 1

   1.1 Motivation ..................................................................................................................... 1

   1.2 Analytical methods ....................................................................................................... 2

       1.2.1 Technical analysis ............................................................................................... 3

       1.2.2 Fundamental analysis ......................................................................................... 4

       1.2.3 Time series forecasting ..................................................................................... 4

   1.3 The Efficient Market Hypothesis .................................................................................. 5

   1.4 Model Comparisons ..................................................................................................... 6

   1.5 Our choice: Artificial neural networks ....................................................................... 7

   1.6 Structure of thesis ....................................................................................................... 7

2. Asset price dynamics and modelling ............................................................................. 11

   2.1 The nature of financial asset prices ........................................................................... 11

   2.2 Properties of financial asset prices ........................................................................... 12

   2.3 Modelling asset returns with simple models ............................................................ 15

       2.3.1 Geometric Brownian motion ............................................................................. 15

       2.3.2 Mean reversion .................................................................................................. 16

   2.4 Modelling asset returns with more complex models ................................................ 18

       2.4.1 Stochastic volatility ............................................................................................. 19

   2.5 Model calibration ....................................................................................................... 21

       2.5.1 Estimators ........................................................................................................... 21
4.3.1 Drawdown: Estimation of the system's intrinsic dimension 53
4.3.2 The "empty space phenomenon" 57

4.4 The proposed hybrid method ......................................................... 57

4.5 Mathematical interpretation of ICA ............................................... 60

4.6 ICA algorithms .............................................................................. 61

4.7 Original data issues and ICA pre-processing ................................... 63
  4.7.1 Original data considerations 63
  4.7.2 Partition of data into training, validation and testing sets 64
  4.7.3 General pre-processing 64

4.8 Estimating the number of independent components ..................... 66

4.9 Selecting important independent components .............................. 68

4.10 Forecasting the ICs .................................................................... 68
  4.10.1 Smooth independent components 69
  4.10.2 Neural network for forecasting independent components 72

4.11 Forecasting evaluation criteria ..................................................... 74
  4.11.1 Root mean square error 75
  4.11.2 Probability of sign error 75

4.12 Reconstruction and residuals ....................................................... 75

4.13 Reconstruction evaluation criterion for selected ICs .................... 76

4.14 Non-linear contributions via the residuals .................................... 76

4.15 Modelling the residuals ............................................................... 77

4.16 Non-linear forecasting of asset returns ....................................... 78

4.17 Generalisation via cross validation and network committees .................. 78
  4.17.1 Cross validation 78
  4.17.2 Committee of networks 79

4.18 Conclusions ................................................................................. 79

5. Implementation of the hybrid model ............................................... 81

  5.1 Original data .............................................................................. 81

  5.2 Partitioning the data set ............................................................... 81

  5.3 First dimensionality reduction using PCA .................................... 83

  5.4 Second dimensionality reduction using ICA ............................... 83

  5.5 Ordering of the independent components .................................... 84

  5.6 Smoothing the components ......................................................... 84

  5.7 Step-by-step procedure ............................................................... 85
5.8 Forecasting the components ................................................................. 86
5.9 Modelling the residuals ........................................................................ 89
5.10 Linear & non-linear reconstruction ....................................................... 90
  5.10.1 Linear reconstruction using existing components 92
  5.10.2 Linear reconstruction using forecasted components 93
  5.10.3 Non-linear reconstruction using residuals and forecasted components 94
  5.10.4 Non-linear reconstruction using forecasted residuals and components 96
  5.10.5 Forecasting a high volatile asset 97
5.11 Verification of the number of independent components ..................... 99
5.12 The conventional neural network approach ......................................... 99
5.13 Conclusions ...................................................................................... 100

6. Replication of hedge fund returns .................................................. 101
  6.1 Introduction ........................................................................................ 101
  6.2 Overview of replication methodologies ............................................. 102
  6.3 Problem formulation for replication ............................................... 106
    6.3.1 Problem definition 106
    6.3.2 Problem formulation 106
    6.3.3 Some observations 107
  6.4 Portfolio turnover restrictions .......................................................... 107
    6.4.1 Hard constraints for portfolio turnover 107
    6.4.2 Penalty function for portfolio turnover 109
  6.5 Additional constraints ...................................................................... 109
  6.6 Simultaneous consideration of funds ............................................... 110
  6.7 The chosen replication model ........................................................... 110
  6.8 The Bionomic algorithm ................................................................... 111
    6.8.1 Description of the Bionomic algorithm 111
  6.9 Out-of-sample forecasts .................................................................... 116
  6.10 Neural network framework and data issues ...................................... 117
    6.10.1 Neural network input data 117
    6.10.2 The neural network targets 118
    6.10.1 The training period 118
  6.11 Network structure ........................................................................... 118
  6.12 Neural network weights determination ............................................ 119
  6.13 Training a committee of networks .................................................. 119
6.13.1 Varying the start weights and the neural network structure 119
6.13.2 Varying the partition of the training period 120
6.13.3 Producing a forecast 120
6.13.4 Evaluating the effectiveness of a neural network committee forecast 120

6.14 Different neural network committees ........................................................ 121
6.15 Conclusions ........................................................................................... 121

7. Implementation of the replication model ...................................................... 123

7.1 Original data ............................................................................................. 123
7.2 Partitioning the data set .......................................................................... 124
7.3 Step-by-step procedure ........................................................................... 124
7.4 Setting parameters and benchmarks ........................................................ 127
7.5 Scenario 1: Back test for period 1999 to 2007 ....................................... 127
  7.5.1 Replication of the EQ index 127
  7.5.2 Replication of the ED index 128
  7.5.3 Replication of the Macro index 129
  7.5.4 Replication of the RV index 129
7.6 Scenario 2: Back test for period 2003 to 2007 ....................................... 130
7.7 Aggregate level performance: the Comp index ...................................... 131
7.8 Conclusions ............................................................................................ 132

8. Conclusions and future work ........................................................................ 133

8.1 Conclusions ............................................................................................ 133
8.2 Future work ............................................................................................ 136

References 137
1.1 Motivation

Human beings have always wanted to predict the future. Some thousand years ago, in ancient Greece, all major political actions depended on the prophesies of the Oracle at Delphi who at that time had to stand near a spring that emitted vapours and caused it to predict the future based on the lapping water and leaves rustling in the trees. Nowadays human beings prefer to base their actions on more objective and independent predictions of future signals given by computers and scientifically sound mathematical models.

In finance there are several kinds of motivation for trying to predict the evolution of financial signals like, for example, stock market prices. The most important of these from the practitioner’s perspective is financial gain. Any system that can consistently pick winners and losers in the dynamic market place would make the owner of the system very wealthy. Thus, many individuals including, investment professionals, and average investors are continuously looking for this superior system which will yield them high returns.

A more academic motivation for predicting the evolution of financial signals is nothing else than proving that price changes have memory. Academics and practitioners in finance have studied the behaviour of stock market returns for a very long time. In 1914 Bachelier was the first to propose the theory of random walk to
characterise the changes of security prices through time. In 1965, Fama also studied the highly stochastic nature of the stock market behaviour after analysing the thirty stocks of the Dow-Jones Industrial Average during the period 1957-1962. He empirically confirmed the random walk hypothesis which states that a series of price changes has no memory. The main theoretical explanation behind this observation is the efficient market hypothesis. In brief this hypothesis states that if a statistically significant serial dependence exists within time series of financial asset prices, the community of financial analysts will immediately exploit it.

Academic work in the seventies and eighties empirically supported the above hypothesis. Practitioners though have always received those results with a lot of scepticism. That is why they developed alternative practical methodologies of explaining the market behaviour like charts or other technical analysis means. All these technical, fundamental, and statistical tools have been used with varying success. However, no one technique or combination of techniques has been successful enough to consistently "beat the market".

Professionals have claimed that part of this failure is due to the fact that classical forecasting methodologies are mainly linear and therefore, unable to capture the complex patterns that price changes exhibit. As a result things seem to have changed during the last years. The research area of forecasting financial time series has become very active mainly due to financial academics who signal a growing interest in novel, more sophisticated non-linear methodologies. The work that we propose in this thesis can be viewed as a contribution to this field of research. We present some novel non-linear artificial neural network based methodologies in order to test the efficient market hypothesis.

1.2 Analytical methods

Before the age of computers, people traded stocks and commodities primarily on intuition. As the level of investing and trading grew, people searched for tools and methods that would increase their gains while minimizing their risk. Statistics, technical analysis, fundamental analysis, and linear regression are all used to attempt to predict and benefit from the market's direction. None of these techniques has proven to be the consistently correct prediction tool that is desired, and many analysts argue about the usefulness of many of the approaches. However, these
methods are briefly presented as they are commonly used in practice and represent a base-level standard for which artificial neural networks should outperform. Also, many of these techniques are used to pre-process raw data inputs, and their results are fed into artificial neural networks as input.

1.2.1. Technical analysis

The idea behind technical analysis is that share prices move in trends dictated by the constantly changing attitudes of investors in response to different forces. Using price, volume, and open interest statistics, the technical analyst uses charts to predict future stock movements. Technical analysis rests on the assumption that history repeats itself and that future market direction can be determined by examining past prices. Thus, technical analysis is controversial and contradicts the Efficient Market Hypothesis. However, it is used by approximately 90% of the major stock traders. Despite its widespread use, technical analysis is criticized because it is highly subjective. Different individuals can interpret charts in different manners.

Price charts are used to detect trends. Trends are assumed to be based on supply and demand issues which often have cyclical or noticeable patterns. There are a variety of technical indicators derived from chart analysis which can be formalized into trading rules or used as inputs to artificial neural networks. Some technical indicator categories include filter indicators, momentum indicators, trend line analysis, cycle theory, volume indicators, wave analysis, and pattern analysis. Indicators may provide short or long term information, help identify trends or cycles in the market, or indicate the strength of the stock price using support and resistance levels.

An example of a technical indicator is the moving average. The moving average averages stock prices over a given length of time allowing trends to be more visible. Several trading rules have been developed which pertain to the moving average. For example, "when a closing price moves above a moving average a buy signal is generated". Unfortunately, these indicators often give false signals and lag the market.

That is, since a moving average is a past estimate, a technical trader often misses a lot of the potential in the stock movement before the appropriate trading signal is generated. Thus, although technical analysis may yield insights into the
market, its highly subjective nature and inherent time delay does not make it ideal for the fast, dynamic trading markets of today.

1.2.2 Fundamental analysis

Fundamental analysis involves the in-depth analysis of a company’s performance and profitability to determine its share price. By studying the overall economic conditions, the company’s competition, and other factors, it may be possible to determine expected returns and the intrinsic value of shares. This type of analysis assumes that a share’s current (and future) price depends on its intrinsic value and anticipated return on investment. As new information is released pertaining to the general market conditions and the company’s status, the expected return on the company’s shares will change, which affects the stock price.

The advantages of fundamental analysis are its systematic approach and its ability to predict changes before they show up on the charts. Companies are compared with one another, and their growth prospects are related to the current economic environment. This allows the investor to become more familiar with the company.

Unfortunately, it becomes harder to formalize all this knowledge for purposes of automation, and interpretation of this knowledge may be subjective. Also, it is hard to time the market using fundamental analysis. Although the outstanding information may warrant stock movement, the actual movement may be delayed due to unknown factors or until the rest of the market interprets the information in the same way.

However, fundamental analysis is a superior method for long-term stability and growth. Basically, fundamental analysis assumes investors are 90% logical, examining their investments in detail, whereas technical analysis assumes investors are 90% psychological, reacting to changes in the market environment in predictable ways.

1.2.3 Time series forecasting

Time series forecasting analyzes past data and projects estimates of future data values. Basically, this method attempts to model a nonlinear function by a recurrence relation derived from past values. The recurrence relation can then be used to predict
new values in the time series, which hopefully will be good approximations of the actual values.

As shown by Box and Jenkins, models for time series data can have many forms and represent different stochastic processes. Broad classes of practical importance are the autoregressive (AR) models, the integrated (I) models, and the moving average (MA) models. These are classes of parametric models that have a linear dependence on previous data points and can be mixed together to form the autoregressive moving average (ARMA) models, the autoregressive integrated moving average (ARIMA) models and some of its deviations: the vector ARIMA (VARIMA), the seasonal ARIMA (SARIMA) and the fractional ARIMA (FARIMA) models. Other parametric linear models that are often used in finance to model stock price movements are: diffusion (Brownian & Geometric Brownian Motion), jump diffusion (diffusion with possible jumps), mean reversion (diffusion reverting to a long-term mean) and mean reversion with jumps.

In the literature one can find a very good overview of "classical" non-linear models for time series modelling and prediction. Amongst those, the autoregressive conditional Heteroskedasticity (ARCH) family models including the generalized ARCH (GARCH), the threshold GARCH (TGARCH), the exponential GARCH (EGARCH), integrated GARCH (IGARCH) and many others are parametric models that represent the changes of variance along time (Heteroskedasticity).

New and more complex non-linear models that explain financial asset behaviour more accurately have been introduced to the academic literature over the last years. Good representatives of such models include the regime switching model, the projection pursuit, artificial neural networks, data-mining trees and wavelet transform based models. These are all non-parametric models. The difference between parametric and non-parametric models is that the first need to be calibrated to the data by methods like ordinary least squares, maximum likelihood estimation or the Akaike information criterion.

1.3 The Efficient Market Hypothesis

The Efficient Market Hypothesis (EMH) states that at any time, the price of a share fully captures all known information about the share. Since all known information is used optimally by market participants, price variations are random, as
new information occurs randomly. Thus, share prices perform a "random walk", and it is not possible for an investor to beat the market.

Despite its rather strong statement that appears to be untrue in practice, there has been inconclusive evidence in rejecting the EMH.

The EMH is important because it contradicts all other forms of analysis. If it is impossible to beat the market, then technical, fundamental, or time series analysis should lead to no better performance than random guessing. The fact that many market participants can consistently beat the market is an indication that the EMH may not be true in practice. The EMH may be true in the ideal world with equal information distribution, but today's markets contain several privileged players who can outperform the market by using privileged information or other means.

1.4 Model Comparisons

In the wide variety of different modelling techniques presented so far, every technique has its own set of supporters and detractors and vastly differing benefits and shortcomings. The common goal in all the methods is predicting future market movements from past information. The assumptions made by each method dictate its performance and its application to the markets.

We believe that the EMH has some merit theoretically, but in real-world applications, it is obvious that there is an uneven playing field. Some market participants have more information or tools which allow them to beat the market or even manipulate it. Thus, stock market prices are not simply a random walk, but are derived from a dynamic system with complexities too big to be fully accounted for.

If an investor does not believe in the EMH, the other models offer a variety of possibilities. Technical analysis assumes history repeats itself and patterns can be noticed in investor behaviour by examining charts. Fundamental analysis helps the long-term investor measure intrinsic value of shares and their future direction by assuming investors make rational investment decisions. Statistical and other non-linear complex techniques attempt to learn from past behaviour and predict future values.
1.5 Our choice: Artificial Neural networks

We initially defined the objective of this research work as adding value to the area of forecasting financial time series. It is well known that the behaviour of financial markets is determined by a huge amount of parameters which can be related to each other in very complex ways.

From Figure 1 we tried to get a hint of which class of methodologies would be the most appropriate to investigate in order to achieve our objective. Figure 1 is a schematic representation of various forecasting techniques used by most professionals. Based on the comments made in the previous paragraph we believe that artificial neural networks are indeed a good candidate model to be used as a platform to try and predict the behaviour of financial asset movements.

In a highly stochastic and only partially understood environment, such as the financial markets, artificial neural networks have the ability to extract rules without having them explicitly formalized. They not only work well within their domain of knowledge but deliver successful results with missing or incomplete information, too.

For those and other reasons, that will be presented in the following chapters of this thesis, we decided to look more into the unique properties of artificial neural networks and expand their capabilities using novel ideas towards the achievement of better, more robust and more accurate forecasting of financial asset returns.

1.6 Structure of Thesis

This thesis is made of 8 chapters. As the title suggest the focus is on nonlinear modelling techniques and in particular on topologically evolved artificial neural network committees as applied in the forecasting of financial asset prices and the replication of hedge fund returns. A detailed description of each of the chapter’s contents follows below:

In chapter 2, we investigate some issues that are related to the complex dynamics of asset price movements. We present the main (general & neural based) modelling methodologies used by most academics and practitioners and discuss complexity issues associated with univariate and multivariate techniques. We use experimental results to prove the importance of higher moments in modelling.
Figure 1: Schematic representation of the various forecasting techniques used under different circumstances. The neural network methodology seems to be ideal for forecasting time series of financial asset prices (highlighted in yellow).
financial time series and present an overview of our proposed solutions for effective modelling of high dimensional complex data systems.

As the main methodology in all our proposed solutions is that of artificial neural networks, we used chapter 3 and to give an overview of issues related to artificial neural networks and help the reader understand the main implementation difficulties. In this chapter we look at how artificial neural networks have been used in general and in financial applications and discuss their strengths and weaknesses. We also overview the main design principles of artificial neural network systems used in the forecasting of financial time series.

In Chapter 4 of the thesis we concentrate on the particular issue of dimensionality reduction and non-linear forecasting. We first give a brief overview of the various dimensionality reduction techniques that exist in the literature and are used by most researchers and practitioners. We then present a hybrid methodology that exploits the advantages of linear and non-linear techniques in order to effectively reduce the dimensionality of complex systems and forecast them into the future. In this novel approach we exploit the linear dependencies of the data by using independent component analysis to decompose the data into "meaningful" linear driving factors and then utilise the non-linear nature of artificial neural networks to discover the non linearities in the residuals of the linearly reconstructed system.

In Chapter 5 we explain the step-by-step implementation procedure of our proposed methodology and apply it to a 42-dimensional system. Experimental results prove the efficiency of the model in finding independent components, processing them, modelling the residuals and accurately forecasting the corresponding asset prices.

In chapter 6 we present a novel mathematical formulation that is applied to the problem of effective replication and forecasting of hedge fund returns. This method involves mixed integer quadratic programming and heuristics for the replication part, and independent component analysis and artificial neural networks for the forecasting of the portfolio weights. It combines many of the methodologies covered in the previous sections. Experimental results are finally presented on 10 equity and fixed income based indices and their composite.

In chapters 4 and 6 we propose a technique in order to extend the capabilities of standard neural network operation. Our alternative is to combine multiple individuals into a committee. Our neural network committees are composed of
several networks and a method for combining their outputs. Each individual network receives the same input and outputs its own decision. Based on outputs from several different networks, the combination algorithm decides the final output. Ideally, the performance of the entire committee will be superior to the performance of any individual network it contains. Two main choices must be made when designing committees: a method of training the networks in a way that encourages diversity of behaviour, and a mechanism to decide the final output based on outputs from individual networks.

In chapter 6 we present a novel mathematical formulation that is applied to the very popular problem of effective replication and forecasting of hedge fund returns. We propose the use of dynamic weightings of instruments and strategies based on non-linear optimisation and learning techniques to determine the replicating portfolio at each time by minimising the sample mean square error of the replicating portfolio's returns versus the observed fund class return time series. We introduce explicit handling of turnover and risk control measures and rigorous out-of sample back testing. The overall objective is the best out-of-sample effectiveness by balancing the absolute return, tracking error, total risk and turnover.

In chapter 7 we test the above model with real hedge fund data and discuss practical issues. Experimental results are shown which are very encouraging as far as both robustness and computational complexity are concerned.

Finally, in chapter 8 we draw our conclusions and make suggestions for future work.
Chapter 2

Asset Price Dynamics & Modelling

2.1. The nature of financial asset prices

Let $x_t$ be the price of an asset at time $t$. The simple return $\bar{r}_t$ on this asset in the time interval $[t-1, t]$ is defined as

$$\bar{r}_t = \frac{x_t}{x_{t-1}} - 1$$  \hspace{1cm} (2.1)

The continuously compounded return $r_t$ on this asset in the time interval $[t-1, t]$ is defined as

$$r_t = \log\frac{x_t}{x_{t-1}}$$  \hspace{1cm} (2.2)

Continuously compounded returns have the advantage that they are additive i.e. the return over $k$ periods is the sum of the $k$ single-period returns. A usual assumption regarding $r_t$ is that, over time, the returns are independent and identically distributed normal. This forms the basis of many of the models described in the following sections.
2.2 Properties of financial asset prices

Unfortunately, the neat theoretical consequences of the above assumptions are in some respects in conflict with historical stock returns. Skewness and kurtosis of a random variable ε are its normalised 3rd and 4th moments respectively. Skewness is a measure of asymmetry of the distribution and kurtosis a measure of how fat the tails of the distribution are. The kurtosis of \( r_t \) is defined as

\[
K_r = \mathbb{E} \left[ \frac{(r_t - \mu)^4}{\sigma^4} \right]
\]

where \( \mu \) is the mean and \( \sigma^2 \) is the variance of the distribution of \( r_t \).

For an observed time series \( r_1, r_2, ..., r_n \), the kurtosis can be estimated by the sample analogue of (2.3),

\[
\hat{K}_r = \frac{1}{n} \sum_{t=1}^{n} \frac{(r_t - \hat{\mu})^4}{\hat{\sigma}^4}
\]

where

\[
\hat{\mu} = \frac{1}{n} \sum_{t=1}^{n} r_t \quad \text{and} \quad \hat{\sigma}^2 = \frac{1}{n} \sum_{t=1}^{n} (r_t - \hat{\mu})^2
\]

are the sample mean and variance respectively. If \( \varepsilon \) is normally distributed it has 0 skewness and a kurtosis figure of 3. Kurtosis values greater than 3 imply fatter tails and values less than 3 imply thinner tails than the normal distribution.

To investigate the levels of skewness and kurtosis in real financial time series we ran an experiment on indices of the stock markets in Amsterdam (EOE), Frankfurt (DAX), Hong-Kong (Hang Seng), London (FTSE 100), New York (S&P 500), Paris (CAC 40), Singapore (Singapore All Shares) and Tokio (Nikkei). The
sample period was from 1986 to 1997 and the sampling frequencies used were daily and weekly. For the daily case this corresponded to 3127 samples whereas for the weekly one to 625 samples. Summary statistics for the above indices like the mean, median, minimum, maximum, variance, skewness and kurtosis can be seen in table 2.1 below.

The statistical data in table 2.1 show that, assuming daily returns, the skewness average across all the selected indices is -1.69 and the kurtosis 41.837. In the weekly returns case the skewness averages -1.381 and the kurtosis 12.832.

The tendency of the weekly return figures to be closer to normal than the daily return figures is as expected. Since the weekly return is the sum of the (continuously compounded) daily returns, assuming independent and identically distributed daily returns (from a distribution with finite moments) the Central Limit Theorem applies and weekly returns are closer to being normal. The predominant feature of the above indices returns is the very large (much greater than 3) values for kurtosis indicating that "abnormal" returns (positive or negative) occur a lot more frequently than would be expected from the normal distribution.

<table>
<thead>
<tr>
<th>Index</th>
<th>Sampling: Daily</th>
<th>Statistical Properties</th>
<th>Sampling: Weekly</th>
<th>Statistical Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Med</td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>EOE</td>
<td>0.038</td>
<td>0.029</td>
<td>-12.788</td>
<td>11.179</td>
</tr>
<tr>
<td>DAX</td>
<td>0.035</td>
<td>0.026</td>
<td>-13.710</td>
<td>7.288</td>
</tr>
<tr>
<td>Hang Seng</td>
<td>0.057</td>
<td>0.022</td>
<td>-40.542</td>
<td>17.247</td>
</tr>
<tr>
<td>FTSE 100</td>
<td>0.041</td>
<td>0.027</td>
<td>-13.029</td>
<td>7.597</td>
</tr>
<tr>
<td>S&amp;P 500</td>
<td>0.049</td>
<td>0.001</td>
<td>-22.833</td>
<td>8.709</td>
</tr>
<tr>
<td>CAC 40</td>
<td>0.026</td>
<td>0.004</td>
<td>-10.138</td>
<td>8.225</td>
</tr>
<tr>
<td>Singapore</td>
<td>0.019</td>
<td>0.001</td>
<td>-9.403</td>
<td>14.313</td>
</tr>
<tr>
<td>Nikkei</td>
<td>0.005</td>
<td>0.002</td>
<td>-16.135</td>
<td>12.430</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Med</td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>EOE</td>
<td>0.190</td>
<td>0.339</td>
<td>-19.962</td>
<td>7.953</td>
</tr>
<tr>
<td>DAX</td>
<td>0.169</td>
<td>0.354</td>
<td>-18.881</td>
<td>8.250</td>
</tr>
<tr>
<td>Hang Seng</td>
<td>0.283</td>
<td>0.556</td>
<td>-34.969</td>
<td>11.046</td>
</tr>
<tr>
<td>FTSE 100</td>
<td>0.207</td>
<td>0.305</td>
<td>-17.817</td>
<td>9.822</td>
</tr>
<tr>
<td>S&amp;P 500</td>
<td>0.246</td>
<td>0.400</td>
<td>-16.663</td>
<td>6.505</td>
</tr>
<tr>
<td>CAC 40</td>
<td>0.128</td>
<td>0.272</td>
<td>-20.941</td>
<td>11.594</td>
</tr>
<tr>
<td>Singapore</td>
<td>0.091</td>
<td>0.110</td>
<td>-27.335</td>
<td>10.510</td>
</tr>
<tr>
<td>Nikkei</td>
<td>0.025</td>
<td>0.261</td>
<td>-10.892</td>
<td>12.139</td>
</tr>
</tbody>
</table>

Table 2.1: Statistical properties of eight financial indices
The kurtosis numbers are far above 3 which indicate fat tails.
The skewness of \( r \) is defined as

\[
S_r = \mathbb{E} \left[ \frac{(r_i - \mu)^3}{\sigma^3} \right]
\]  

and is a measure of the asymmetry of the distribution of \( r \). The skewness for an observed time series \( r_1, r_2, ..., r_n \) can be estimated by the sample analogue of (2.5) as

\[
\hat{S}_r = \frac{1}{n} \sum_{i=1}^{n} \frac{(r_i - \hat{\mu})^3}{\hat{\sigma}^3}
\]  

All symmetric distributions, including the normal distribution, have skewness equal to zero. It is clear that in table 2.1 all indices returns, irrespective of whether they are sampled daily or weekly, exhibit negative skewness which implies that the left tail of the distribution is fatter than the right tail, or that large negative returns tend to occur more often than large positive ones.

Looking at the time series of daily returns for the DAX and the FTSE in figure 2.1 we notice another typical statistical feature of financial asset returns. It is obvious that the relatively volatile periods, characterized by large price changes (and hence large returns) alternate with more tranquil periods in which prices remain more or
less stable and returns are, consequently, small. In other words, large returns seem to occur in clusters.

The above features of financial time series seem to require non linear models simply because linear models would not be able to generate data that have these features. However as one often starts the analysis of empirical time series with linear models even if one is ultimately interested in non linear features we feel that a basic knowledge and understanding of the most important concepts in linear time series analysis are indispensable. The following section highlights only the main aspects of linear models as applied to financial time series.

2.3 Modelling asset returns with simple models

Let \( x_t \) be the price of an asset at time \( t \) and consider a small time interval \([t, t + dt]\) during which the asset price changes by \( dx_t \).

2.3.1 Geometric Brownian Motion

Prices of equities, currencies, commodities and stock indices, in general, exhibit growth over time and the proposed models reflect that behaviour. Geometric Brownian Motion (GBM) is perhaps the simplest useful model describing the dynamics of \( x_t \) and is given by the stochastic differential equation (SDE)

\[
\frac{dx_t}{x_t} = \mu dt + \sigma dW_t
\]

(2.7)

where \( \mu \) (the drift rate) and \( \sigma \) (the volatility per unit time) are both constants.

The variable \( W_t \) is a standard Wiener process with \( dW_t \) the change in the value of this process during the above time interval. GBM is the model chosen by Black and Scholes [2.1] to derive their classic option-pricing formula. GBM is one of very few types of SDEs that are solvable. The solution is:

\[
x_t = x_0 \exp \left( \mu - \frac{\sigma^2}{2} \right) t + \sigma W_t
\]

(2.8)
Thus for a given time $t$, $x_t$ can be simulated directly by simulating $W_t$. The distribution of $x_t$ implied by the above equation is lognormal [2.2]. Note that if we define $r_t$ as the continuously compounded asset returns (implied by the prices $x_t$) then applying Ito’s Lemma produces

$$r_t = \left(\mu - \frac{\sigma^2}{2}\right)dt + \sigma dW_t$$

(2.9)

which implies that the distribution of asset returns under the GBM is normal.

If $\mu$ and $\sigma$ in equation (2.7) are not constants but functions of time only (for example $\mu(\tau)$ and $\sigma(\tau)$), the equation for GBM is still solvable and its solution is given by equation (2.8) substituting $\mu$ with

$$\widetilde{\mu} = \frac{1}{t} \int_0^t \mu(\tau) d\tau$$

and $\sigma$ by

$$\widetilde{\sigma}^2 = \frac{1}{t} \int_0^t \sigma^2(\tau) d\tau$$

2.3.2 Mean Reversion

In the cases where $x_t$ does not exhibit growth but instead reverts to a mean, then the model described above is inappropriate. These mean reversion (MA) processes include interest rates, the rate of inflation and other variables even macro ones like unemployment that can also be modelled as stochastic processes.

A simple model exhibiting mean reversion is the Ornstein-Uhlenbeck (OU) process given by the SDE

$$dx_t = \alpha (\bar{x} - x_t) dt + \sigma dW_t$$

(2.10)
where \( \bar{x} \) (the level to which the mean reverts) and \( \alpha \) (the rate at which \( x_t \) reverts to the mean) are both constants. The OU model was suggested for short interest rate modelling by Vasicek [2.3]. Equation (2.10) is another of the few examples of solvable SDEs. If \( \bar{x} \) is not a constant but a function of time, say \( x(t) \), the above equation remains solvable. Its solutions is given by

\[
x_t = x_0 \exp(-\alpha t) + \alpha \int_0^t \bar{x} \exp(-\alpha (t - \tau)) d\tau + \sigma \int_0^t \exp(-\alpha (t - \tau)) dW_t
\]

(2.11)

The value of the last term above involving the stochastic integral is normally distributed with mean zero and variance

\[
m^2(t) = \sigma^2 \frac{1 - \exp(-2\alpha t)}{2\alpha}
\]

(2.12)

If we now let

\[
n(t) = \alpha \int_0^t \bar{x}(\tau) \exp(-\alpha (t - \tau)) d\tau
\]

(2.13)

the solution can be rewritten as

\[
x_t = x_0 \exp(-\alpha t) + n(t) + m(t)Z_t
\]

(2.14)

where \( Z_t \) is a standard normal variable. Thus again \( x_t \) can be simulated by simulating \( Z_t \).

An undesirable property of the OU model is that it allows negative prices to occur with positive probability. This problem is eliminated by a modification to the OU process which was first suggested for application to interest rates by Cox, Ingersol and Ross [2.4] and known as the CIR model. The corresponding SDE is:

\[
dx_t = \alpha (\bar{x} - x_t) dt + \sigma \sqrt{x_t} dW_t
\]

(2.15)
and has a known solution in the sense that, for a given $x_0$, the probability density function of $x_t$ is known. Scott [2.5] gives the solution as

$$x_t = x_0 + 2cD_t$$

(2.16)

where

$$c = \frac{2\alpha}{\sigma^2 [1 - \exp(-\alpha t)]}$$

(2.17)

and $D_t$ has a non central $\chi^2$ distribution with degrees of freedom

$$\nu = \frac{4\alpha \bar{x}}{\sigma^2}$$

(2.18)

and non centrality parameter

$$\lambda = 2c\bar{x}_0 \exp(-\alpha t)$$

(2.19)

In his paper Scott [2.5] also discusses sampling from the above distribution. The CIR model is not solvable if $\bar{x}$ in equation (2.15) is not a constant but a function of time.

2.4 Modelling asset returns with more complex models

Without question, the biggest shortcoming of the above models is their representation of the volatility of prices (or, equivalently, of returns). Almost any modification to these models leads to unsolvable SDEs in virtually all cases. In these circumstances, some of the very limited available techniques that attempt to solve the problem are numerical or algorithmic methods, including tree and lattice representations [2.6], [2.7], [2.8], [2.9], simulation [2.10], [2.11], [2.12], general graph-theoretic models [2.13], [2.14], [2.15] and neural networks [2.16], [2.17], [2.18].
2.4.1 Stochastic volatility

Making the volatility parameter \( \sigma \) in the three models given by equations (2.8), (2.10) or (2.15) stochastic, results in a major generalization of all the models although direct solutions are now out of the question.

Suppose in the above three models we substitute stochastic volatility \( \sigma_t \) instead of the constant \( \sigma \), and call the Wiener process \( W_t^{(1)} \) instead of \( W_t \). The three equations now become

\[
\frac{dx_t}{x_t} = \mu dt + \sigma_t dW_t^{(1)} \tag{2.20}
\]

\[
dx_t = \alpha (\bar{x} - x_t) dt + \sigma_t dW_t^{(1)} \tag{2.21}
\]

\[
dx_t = \alpha (\bar{x} - x_t) dt + \sigma_t \sqrt{x_t} dW_t^{(1)} \tag{2.22}
\]

A second SDE describing the dynamics of \( \sigma_t \) must now be added and assuming that volatility follows mean reversion could, for example, take the form

\[
d(\sigma_t^2) = \eta (\bar{\sigma}^2 - \sigma_t^2) dt + \theta \sigma_t^m dW_t^{(2)} \tag{2.23}
\]

where the constants \( \bar{\sigma} \) and \( \eta \) are the reversion level and rate of reversion of the volatility, and the positive constant \( \theta \) is the volatility of the volatility parameter.

Power \( m \) is usually assumed to be 1 or 2 ensuring that \( \sigma_t^2 \) remains positive. \( W_t^{(2)} \) is a second Wiener process which may be correlated to the first Wiener process which drives the price \( x_t \) with a correlation factor

\[
\rho = \frac{E[W_t^{(1)} W_t^{(2)}]}{\sqrt{t}} \tag{2.24}
\]

Although for none of the above models is the distribution of \( x_t \) known analytically, it is known that it exhibits fat tails (as described in section 2.2 above) which become more pronounced as the parameter \( \theta \) in equation (2.23) increases.
GARCH [2.19] provides another way of modelling stochastic volatility without the need to introduce additional random factors. Asset prices and volatilities are sampled at discrete time instances with a time step $\delta t$ between them. If we assume that the expression for asset prices takes the GBM form of solution given by equation (2.8), the simplest GARCH(1,1) model for volatility $\sigma_t$ then produces

$$x_t = x_{t-1} \exp \left[ \left( \mu - \frac{\sigma_{t-1}^2}{2} \right) + \varepsilon_{t-1} \right]$$

(2.25)

$$\sigma_t^2 = \omega + \beta \sigma_{t-1}^2 + \alpha \varepsilon_{t-1}^2$$

(2.26)

where $\varepsilon_t = \sigma_t \cdot \tilde{\varepsilon}_t$ with $\tilde{\varepsilon}_t$ a random variable (not necessarily normal) with zero mean and variance one. The other three parameters $\omega$, $\alpha$ and $\beta$ are constants. Parameter $\alpha$ determines how quickly volatility reacts to market movements, while parameter $\beta$ determines the persistency of that volatility.

GARCH is designed in such way that the same random process $\varepsilon_t$ drives both the price and volatility (except that for volatility it is the square of the process that is important). Again, the distribution of asset prices exhibits fat tails that become more pronounced as the parameter $\alpha$ in equation (2.26) increases.

An interesting GARCH type model which does not assume that the terms $\varepsilon_t$ are normal is the so-called J-GARCH model [2.20]. In this model, $\varepsilon_t$ is assumed to be a mixture of two normal distributions with the same mean 0, but two different variances.

Although J-GARCH accounts in a better way for the fat tails of the asset price distribution, it is still a symmetric model. It is shown earlier that a large negative movement in an asset price induces larger changes in the subsequent conditional variance than a large positive move of the same size. GARCH methods have been extended to deal with such asymmetries.

Introducing stochastic volatility into a model complicates matters enormously but also represents a large increase in realism for that model. Indeed, over the last few years, much of the research has concentrated on exploiting market imperfections caused by the difficulty of Black & Scholes formula to effectively capture the
dynamics of asset price movements. The reason for this difficulty is mainly the normality assumptions of the Black & Scholes model which make it underestimate the well proven high probability of occurrence of tail events. As is shown in section 2.2, asset prices and returns do exhibit fat tails, and there is a growing belief that these fat tails are produced by stochastic volatility.

2.5 Model calibration

All the above models involved the use of drift and volatility parameters in their specification. The models need to be calibrated against real historical data before they are useable. This implies choosing the "best" set of parameters so that the model is as good a representation of the time series data as possible. In this section we briefly describe the most usual parameter estimation techniques in the literature.

2.5.1 Estimators

An estimator of a parameter $p$ is a random number $p'$. The estimator is unbiased if $E(p') = p$. If for two different unbiased estimators $p'$ and $p^*$, $\text{Var}(p') < \text{Var}(p^*)$ then $p'$ is called more efficient than $p^*$. Sometimes there are estimators $\tilde{p}$ which are biased but for which $\text{Var}(\tilde{p}) < \text{Var}(p')$ for any unbiased estimator $p'$, in which case we have a subjective decision to make as to which is the best estimator.

2.5.2 Ordinary least squares

Ordinary least squares is one of the widely used methods of parameter estimation based on minimizing the difference between the true and estimated model and as such is at the centre of linear regression analysis. Ordinary least squares is used for parameter estimation of both Geometric Brownian Motion and Ornstein-Uhlenbeck models where the error is represented by Brownian Motion increments.

2.5.3 Maximum likelihood estimation

Maximum likelihood estimation is a popular method of parameter estimation which can be applied when the probability density function (pdf) of the samples is
known in an analytic form. It is based on maximizing the probability that the time series has been produced by the proposed model. It can, for example, be applied to Geometric Brownian Motion giving identical results to ordinary least squares for the estimate of $\mu$ but an almost identical estimate of $\sigma$.

2.5.4 Akaike Information Criterion (AIC)

One of the commonly used methods for choosing the best model for fitting a time series is the Akaike Information Criterion [2.21]. This is a criterion for comparison of models with different numbers of parameters estimated by using maximum likelihood estimation. It gives an indication of goodness of fit of the estimated model. Separation between two models is measured by the non normalised Kullback-Leiber information (also known as the cross-entropy or discrepancy). The Akaike information criterion has an excellent role to play in choosing between the competing models for asset price dynamics described in the previous section.

2.6 Modelling portfolios of assets: Correlations & Dimensionality

In the above sections we discussed the modelling of a single asset price (and, in some cases, also its volatility). In many situations it is necessary to model simultaneously a number $n$ of assets. The generalization of the earlier models to deal with multiple assets is much less straightforward. Calibration now becomes complex, inaccurate and the number of parameters to be estimated becomes very large.

Even simulation of the jointly distributed asset prices is not simple with large numbers of assets, and a very large number of simulations may be needed for any degree of accuracy to be achieved.

With a large number of dimensions, it is necessary to employ variance-reduction techniques (for example Sobol low-discrepancy sequences [2.22]) to improve the simulation. In these cases, not only must we ensure that each asset is individually well modelled, we must also ensure that relative asset prices are correct with respect to each other, in the sense that they satisfy the no arbitrage conditions. In this respect, note that although the model for the simultaneous asset price dynamics may be arbitrage free, generating a (sub)set of scenarios for simulation purposes using this model, may easily contain arbitrage.
Consider any two assets and assume that their prices (or the corresponding returns) are linear functions of a number of random factors which are jointly normally distributed. When a linear combination of normal variables is normal, the change in the value of any portfolio composed of such assets is also normally distributed. Under such assumptions, co-movements in the asset prices (or returns) are usually summarised by a constant covariance matrix.

It has been mentioned earlier that the volatility (standard deviation) of an asset's price is by no means constant and we discussed its stochastic modelling. The same is also true for correlations. Indeed, constant correlations computed directly from financial time series are notoriously unstable for a number of reasons.

2.6.1 The asynchronous nature of data

The asynchronous nature of data obtained from different markets can make two time series appear uncorrelated, even when they are highly correlated. Note that various corrections can be applied, but the resulting covariance matrix is then very often not even positive definite. A method for converting a non-positive definite matrix to a 'near-by' positive definite one is given in [2.23].

2.6.2 Fat tails

Because, as mentioned earlier, the distribution of asset prices or returns is not multivariate normal but has highly pronounced (fat) tails, a covariance matrix is not the right measure for co-movements [2.24].

2.6.3 Right sample

There is a fundamental contradiction concerning the 'right' number of historical data to use in the correlation computations. Few data implies large standard errors whereas a lot of data implies that recent changes in the price interrelationships (due to fundamental economic or political changes) will be lost in the large number of past observations.

2.6.4 Event driven correlations

Correlations not only vary with time, but may change most at times when they matter most. For example [2.25] provides evidence that correlations increased
uniformly and significantly during the October 1987 crash and other large stock market movements.

2.7 Modelling asset returns with neural networks

Neural networks are adaptive computational methods that learn a task from examples without trying to build a physically sound model of the task. They are powerful tools that can be used in a wide variety of applications, when non-linear modelling of data is needed. Neural networks are appropriate extensions to linear data analysis tools which have proven to be both theoretically sound, and of practical interest in real applications.

2.7.1 Multi layer perceptron

Multi layer perceptron neural networks can be used for time series processing.

A neural network theoretically does not have to assume stationarity. Also the fact that it is a non linear model makes it capable of dealing with much more complex underlying characteristics of the series.

Non-linear models are by definition more powerful, since they give more possibilities in the choice of the input-output relation (including the linear case). Working with non-linear models is however more difficult: the increased possibilities may be seen as supplementary degrees of freedom, leading to a better
fitting of the model to the known values, but to a worst generalization ability of the model on unknown data. This learning-generalization dilemma, similar to the bias-variance dilemma in statistics, is the main limitation of neural networks.

Some neural networks possess the universal approximation property and under mild conditions on the data, they can fit any data set with an arbitrary high precision, (provided that there is sufficient number of parameters in the model). However, when there are too many parameters (compared to the number of data available), the "overfitting" phenomenon appears. The known data (used for learning) are well fitted, but the function has no sense between points used for learning or in extrapolation beyond the time series end.

This overfitting problem increases with the model complexity, and is thus more difficult to handle when many input variables (past values and exogenous information) are used.

Examples of multi layer perceptrons used for time series prediction purposes can be found in [2.26], [2.27], [2.28] and [2.29].

2.7.2 Other neural network structures used in time series forecasting

*Jordan networks*

The Jordan networks can be seen as an extension of the ARMA models. It is a multi-layer perceptron with one hidden layer and a feedback loop from the output layer to an additional input layer called context layer. Each node in the context layer is connected to itself via self-recurrent loops with a weight smaller than 1.

Hence, a Jordan network takes into account not only past time series elements, but also its own forecasts. This property has often given rise to the argument that recurrent networks can exploit information beyond a limited time window. However, in practice this cannot really be exploited. If the weight of a connection to a context node is close to 1, the node (if it uses a sigmoid transfer function) quickly saturates to maximum activation, where additional inputs have little effect. If the weight is very small in comparison to 1, the influence of past estimates quickly goes to 0.

*Other network architectures*
While the most important neural network approaches to time series processing have been described in previous sections, there exists a variety of many other design approaches, several of which are listed below.

Many time series applications are tackled with fully recurrent networks, or networks with recurrent architectures different from the ones already discussed [2.34]. Special learning algorithms for arbitrary recurrent networks have been devised, such as backpropagation in time [2.35] and real-time recurrent learning [2.36].

Many authors use a combination of neural networks with so-called hidden Markov models (HMM) for time series and signal processing. HMMs are related to finite state automata and describe probabilities for changing from one state to the other. Detailed treatment of this topic can be found in [2.37] and [2.38].

Unsupervised neural network learning algorithms, such as the self-organizing map, can also be applied in time series processing, both in forecasting [2.39] and classification [2.40].

2.8 Multivariate systems & high dimensional data

In sections 2.7.1 and 2.7.2 we presented various ways in which neural networks can be used for modelling univariate processes and explained why they are perceived as non linear alternatives to classical parametric models.

Using the same rationale as in section 2.6, we can say that modelling the behaviour of multivariate systems is, indeed, very complex. Many believe that neural networks are ideal for such complex systems and they can outperform other methods in discovering the correlation dynamics of multiple assets as long as they are carefully set up. Unfortunately, though, even with a good set up neural networks do fail often and this is mainly due to what was explained in the previous section as “over fitting”. Underlying this is that neural networks suffer from limitations similar to those of conventional tools, concerning their ability to work with high-dimensional data.

In the literature, one can find that neural networks can “beat the curse of dimensionality”. In fact, it is true that neural networks perform better than many other conventional tools, when the dimension of the data increases.
However, it is impossible to demonstrate today that no problem arises in high-dimensional spaces, even with neural networks. It is a fact that the intrinsic dimensionality of data is usually lower than the dimension of the data vectors therefore using dimensionality reduction techniques may help to bypass the problems of high dimensions.
3.1 Preliminaries

In the last decade, scientists have shown an increasing interest in neural
networks and learning mechanisms in general mainly due to the considerable
improvement in computer power and speed which has rendered the solution of many
previously intractable classes of problems, possible. This type of methodology has
since been applied to many fields like system identification and control, game
playing and decision making, pattern recognition, sequence recognition, medical
diagnosis, data mining etc.

Financial services organizations have been the second largest sponsors of
research in neural network applications [3.4] with typical applications in risk rating
of mortgages and fixed income investments, index construction, simulation of
market behaviour, portfolio selection / diversification, identification of economic
explanatory variables, and financial forecasting [3.3]. Other advantages include
greater fault tolerance, robustness and adaptability due to their large number of
interconnected processing elements that can be trained to learn new patterns [3.1],
[3.4].

Neural networks are adaptive computational methods that learn a task from
examples without trying to build a logically sound model of the task. They are
powerful tools that can be used in a wide variety of applications, when non-linear
modelling of data is needed. As most physical phenomena are non-linear, neural networks are appropriate extensions to linear data analysis tools. They have proven to be both theoretically sound, and of practical interest in real applications.

While neural networks are powerful methods it must be highlighted that most of them are difficult to use and usually need some expertise. Problems related to the efficiency of these algorithms are often mixed with the difficulty of estimating the topology of a network for a particular problem which makes it even more difficult to appreciate the real power of neural networks.

This difficulty in using neural networks is easily identifiable in practice as it appears that although many organisations have expressed interest in applying neural network technology, few have actually implemented them successfully. Those that have been successful have spent considerable resources experimenting and fine tuning neural networks for their particular applications as evidenced by the many software packages originally developed as in-house proprietary programs.

3.2 General Issues

Financial and economic forecasters have witnessed the recent development of a number of new forecasting models. Traditionally, popular forecasting techniques include regression analysis, time-series analysis, moving averages and smoothing methods, and numerous judgmental methods. However, all of these have the same drawback insofar as they require assumptions about the form of population distribution. Regression models, for example, assume that the underlying population is normally distributed. Neural networks are members of a family of statistical techniques, as are flexible nonlinear regression models, discriminant models, data reduction models, and nonlinear dynamic systems [3.5], [3.6]. They are trainable analytic tools that attempt to mimic information processing patterns in the brain [3.7]. Because they do not necessarily require assumptions about population distribution, economists, mathematicians and statisticians are increasingly using neural networks for data analysis. Not only do they not require assumptions about the underlying population but are also powerful forecasting tools that draw on the most recent developments in intelligence research.

Hardin in [3.8] observes that statistical methods, such as neural networks, were developed partly as the product of the ordinal revolution in economics and choice
theory. He points out that because our choices have social and interactive contexts, it would be extremely difficult to construct a theoretical model that is capable of tracing out all potential and actual responses and interactions. Such models are bound to exhibit fundamental indeterminacy. These interdeterminacies are the inevitable product of strategic interactions among rational individuals who understand that their actions, or inactions, are going to be followed by reactions - those of the other participants in the strategic game and those of the environment.

In such circumstances, one may find that responses are not similar, let alone unique. This is especially true when we add the time dimension to the discussion. A player may react in ways very different to what was presupposed in response to an unexpected reaction by one's opponent. Such models are inherently dependent upon, and sensitive to, initial conditions, which may not permit accurate predictions even for very near-future states [3.9], [3.10], [3.11].

Supposing such a complex model could be constructed, once the model is subject to empirical testing, the problem of aggregation arises. Although neural networks perform well in the presence of missing data, large data sets are required to train them [3.7]. These data sets can only come from past observations and are used to deduce the main patterns of future events by tracing similar interactions found in historical data.

Neural networks are capable of processing data and performing calculations that humans can not perform mentally. Their answers, therefore, must be accepted on faith; yet neural networks themselves are at the mercy of the sample data and the quality of their features [3.5], [3.12], [3.13], [3.14]. Neural networks process data and make forecasts much in the same way as statistical algorithms perform estimations.

In applications specific to the financial and economic fields, the main focus for neural network technology so far has been with data involving variables in non-linear relations. Researchers have examined the application of neural networks to financial markets, where the non-linear properties of financial data create too many difficulties for traditional methods of analysis [3.15], [3.16], [3.17], [3.18], [3.19], [3.20].
3.3 Neural networks in finance

Neural networks are used in a widening range of applications, including airline security control, investment management and risk control [3.21], industrial management and production [3.22], [3.23], [3.24], as well as in forecasting stock price indexes and derivative securities [3.25], [3.26], [3.27], [3.28], and predicting exchange rates [3.29].

Neural networks have found supporters among various portfolio managers, investment banks, and trading firms. Most of the major investment banks, have implementations of neural networks. Fidelity Investments has set up a mutual fund whose portfolio allocation is based solely on recommendations produced by an neural network. The fact that major companies in the financial industry are investing resources in neural networks indicates that neural networks may serve as an important method of financial forecasting in the future.

Yoon and Swales in [3.30] compare neural networks to discriminant analysis. The technique of discriminant analysis is generally used to build a procedure that not only considers the number of correct and incorrect classifications of the data but also takes into account the cost of each type of classification. Yoon and Swales show that the prediction of stock price performance based on a neural network model is superior to prediction based on a discriminant analysis.

Surkan and Singleton in [3.31] find that neural networks models perform better than discriminant analysis also in predicting future assignments of ratings to bonds. This may lead to an inaccurate assumption of determinacy, where - given a set of initial conditions - it may be presupposed that future prices of stocks and bonds may be predicted. However, the actual values (or neural network outcomes) fluctuate unpredictably, indicating a noticeable behaviour of indeterminacy. Trippi and DeSieno in [3.4] apply an neural network system to the modelling of trades in Standard and Poor's 500 index futures. They find that the dynamics of the neural network system helps to outperform a passive approach to investment (a buy-and-hold strategy) in the index; thus, they favour the implementation of neural networks to the financial decision making process. Donaldson, Kamstra, and Kim in [3.32] contribute to the study of the capital asset pricing model as well as neural networks by examining the fat tails and heteroskedasticity in stock return data. Using data from the stock indexes of London, New York, Tokyo, and Toronto, they find that
neural network models outperformed many traditional models, including the ARCH model, in removing leptokurtosis and symmetric and asymmetric heteroskedasticity from the stock index data. This superior capability allows the neural network model to be utilized (perhaps a little too confidently) as a deterministic tool.

A balanced assessment of the potential of neural networks is offered by Hill, Marquez, O'Connor, and Remus in [3.33]. They find neural networks to be comparable to traditional statistical methods. In regard to the forecasting potential of neural networks, they find that such networks perform as well as classical statistical models for forecasting yearly time-series, but that for monthly and quarterly time-series, neural networks may actually outperform statistical models. Neural networks do not necessarily outperform regression in modelling human decision making, except when non-linear elements are involved. This is not a case of explicit determinacy as it is more likely an indication of a weakly deterministic system. Kuo and Reitsch [3.34] test the accuracy of forecasts produced by both multiple regression and neural network models. They test their models on ten different out-of-sample data sets and analyze the forecasting errors of each of the models. The results indicate that neural networks outperform conventional methods in all cases.

### 3.4 Advantages & disadvantages

Econometric models as well as neural networks can be used as forecasting tools, but each tool requires a different method using different procedures, each of which offers specific advantages and disadvantages. Neural networks have many advantages over conventional methods of analysis. First, they have the ability to analyze complex patterns quickly and with a high degree of accuracy.

Second, neural networks make no assumptions about the nature of the distribution of the data. They are not, therefore, biased in their analysis. Instead of making assumptions about the underlying population, neural networks with at least one middle layer use the data to develop an internal representation of the relationship between the variables. Consequently, better results can be expected with neural networks when the relationship between the variables does not fit an assumed model.

Third, since time-series data are dynamic in nature, it is necessary to have non-linear tools in order to discover relationships among the time-series [3.35]. Neural networks are best at discovering these types of relationships.
Fourth, neural networks perform well with missing or incomplete data. Whereas traditional regression analysis is not adaptive, indiscriminately processing older data together with new data, neural networks readjust their weights as new input data becomes available [3.7], [3.34], [3.36].

One treatment of indeterminacy involves probability and statistics; the other is the fuzziness of input and output variables and the fact that economic and finance data arriving in different time intervals are always subject to major revisions.

Traditional econometrics techniques are not able to handle such models due to these indeterminacies. neural networks suggest an alternative not by solving or reducing the indeterminacy but by being able nevertheless to forecast with some degree of accuracy. However, there are some drawbacks connected with the use of neural networks. For one, neural networks are not all-purpose problem solvers. Thus far, there is no structured methodology available for choosing, developing, training, and verifying a neural network. There is no standardized paradigm for development. The output quality of neural networks may be unpredictable regardless of the design and implementation schedule. Some researchers maintain that no estimation or prediction errors are calculable when using neural networks [3.37] due to constant “learning” by the process. Also, neural networks are “black boxes,” for it is impossible to figure out how relations in their hidden layers are estimated [3.7], [3.26].

The difficulty in prediction from the lack of comprehension of the system’s internal relations is a perfect example of observational indeterminability. The main factor in this indeterminacy is that neural networks produce their own patterns from a set of inputs that will be needed to operate the very network of production in the future.

Another drawback is that neural networks have long training times. Excessive iterations are required to train them [3.7]. Reducing training time is crucial because building a neural network forecasting system is a process of trial and error; hence, the more experiments a researcher can run in a finite period of time, the more confident he can be of the result. The network also tends to base its predictions of future events on “memories” of similar situations from the past [3.38].

In the case of financial markets, neural nets quantify the influence of major financial variables and the impact that these relationships have on the future price movement of the target market [3.39].
The networks learn from carefully crafted training data containing such variables as interest rates, currency prices, commodity prices, the slope of the yield curve, the movement in major commodity prices, the movement in major financial averages, internal market data, technical indicators, and so forth.

Neural networks are data-dependent, so the algorithms are only as good as the data shown to them [3.40]. In this light, neural networks may be thought as weakly deterministic systems that converge to a predictable eigen-behaviour.

After the learning process, when given specific input data, it may be possible to predict a general solution that would be produced by an neural network. Since the future is often drastically different from the past in financial markets, the user needs to test the predictions on out-of-sample ranges in order to ensure the reliability of the network. The results are generally indeterministic. Kanas in [3.41] used out-of-sample data of monthly returns forecasts for Dow Jones and the Financial Times stock indices, using both a linear and neural network model. Neither model performed well in predicting directional changes in the two indices, but they did help to support the conclusion that the underlying relationship between stock prices is not linear.

Neural networks tend to under- or over-fit data [3.7]. It is always possible to build a neural net or a mathematical function that exactly fits all the historical data such as a time series, but the predictive capability of such a system is relatively nonexistent. This over-fitting is because the noise and anomalies in the data do not allow the net to predict with any accuracy. This is also the nature of indeterminacy in social systems as we have discussed previously.

A neural network’s rigour thus suffers from poor generalization capability. Enough repetitions will help to produce results with extremely high $R^2$ values, but they will have no relevance to reality. If a user relies on the results of an neural network in the belief that a neural network has high predictive power, a series of decisions will follow that may prove disastrous. The user should always be aware that a neural network is not a strongly deterministic system that contains explicit cause-and-effect relationships or are based on first principles. The severity of the consequences should induce a prudent user to test one’s net on a number of out-of-sample data sets in order to reaffirm the predictive power of the system [3.42].

Problems can be reduced by using already smoothed variables, which give the system less incentive to seek to fit its own curve to the data. It is also extremely
important to use only economically significant variables for inputs. Every user will have to define what one considers being the economically significant variables. Studies show that using a few well-chosen variables will give significantly better results than trying to use every economic indicator as a viable predictor [3.43]. Developing a successful market timing neural network requires specialized expertise in the market model.

Studies show that networks produce the best results when used in conjunction with an expert. The expert prompts the neural network at key decision nodes, and he or she allows the user to enter his opinion as to the weight, or importance, of a specific variable. By removing highly correlated signals that have a fairly direct relationship between two inputs, the user can assess more correctly the predictive power of the inputs still present and thereby construct a better model. With such checked interaction, expert-guided networks command greater predictive power. These interactive choices present a form of indeterminacy, which results from strategic interaction, differing each time for each expert.

3.5 Efficient design methodology

The step-by-step methodology described in the following sections is used by most researchers and practitioners in order to effectively design neural networks that successfully predict financial time series in complex tasks.

3.5.1 Variable selection

Success in designing a neural network depends on a clear understanding of the problem [3.44]. Knowing which input variables are important in the market being forecasted is critical. This is easier said than done because the very reason for relying on a neural network is for its powerful ability to detect complex nonlinear relationships among a number of different variables. However, economic theory can help in choosing variables which are likely important predictors. At this point in the design process, the concern is about the raw data from which a variety of indicators will be developed. These indicators will form the actual inputs to the neural network.

The financial researcher interested in forecasting market prices must decide whether to use both technical and fundamental economic inputs from one or more markets. Technical inputs are defined as historical values of the dependent
variable(s) or indicators calculated from the historical values. Fundamental inputs are economic variables which are believed to influence the dependent variable. The simplest neural network model uses historical values of the dependent variable(s). Such models have outperformed traditional ARIMA-based models in price forecasting, although not in all studies [3.45], [3.46]. A more popular approach is to calculate various technical indicators which are based only on past prices of the market being forecasted [3.47]. As an additional improvement, market data can be used since the close link between all kinds of markets, both domestically and internationally, suggests that using technical inputs from a number of interrelated markets should improve forecasting performance. For example, intermarket data such as the Euro/Yen and Pound cross rates and interest rate differentials could be used as neural network inputs when forecasting the Euro. Fundamental information such as the current account balance, money supply or wholesale price index may also be helpful.

The frequency of the data depends on the objectives of the researcher. A typical off-floor trader in the stock or commodity futures markets would likely use daily data if designing a neural network as a component of an overall trading system.

An investor with a longer term horizon may use weekly or monthly data as inputs to the neural network to formulate the best asset mix rather than using a passive buy and hold strategy. An economist forecasting the gross domestic product (GDP), unemployment or other broad economic indicators would likely use monthly or quarterly data.

3.5.2 Data collection

The researcher must consider cost and availability when collecting data for the variables chosen in the previous step. Technical data is readily available from many vendors at a reasonable cost whereas fundamental information is more difficult to obtain. Time spent collecting data cannot be used for pre-processing, training and evaluating network performance.

Missing observations which often exist, can be handled in a number of ways. All missing observations can be dropped or a second option is to assume that the missing observations remain the same by interpolating or averaging from nearby
values. Dedicating an input neuron to the missing observations by coding it as a one if missing and zero otherwise is also often done.

When using fundamental data as an input in a neural network four issues must be kept in mind. First, the method of calculating the fundamental indicator should be consistent over the time series. Second, the data should not have been revised after its initial publication as is commonly done in databases since the revised numbers are not available in actual forecasting.

Third, the data must be appropriately lagged as an input in the neural network since fundamental information is not available as quickly as market quotations.

Fourth, the researcher should be confident, that the source will continue to publish the particular fundamental information or other identical sources are available.

### 3.5.3 Data pre-processing

As in most other neural network applications, data pre-processing is crucial for achieving a good prediction performance when applying neural networks for financial time series prediction. The input and output variables for which the data was collected are rarely fed into the network in raw form. At the very least, the raw data must be scaled between the upper and lower bounds of the transfer functions (usually between zero and one or minus one and one).

Two of the most common data transformations in both traditional and neural network forecasting are first differencing and then taking the log transformation of a variable. Differencing, or using changes in a variable, can be used to remove a linear trend from the data. Logarithmic transformation is useful for data which can take on both small and large values. Logarithmic transformations also convert multiplicative or ratio relationships to additive which is believed to simplify and improve the network training [3.3].

Another popular data transformation is to use ratios of input variables. Ratios highlight important relationships while at the same time conserving degrees of freedom because fewer input neurons are required to code the independent variables.

Besides first differences, logs and ratios, technical analysis can provide a neural network with a wealth of indicators including a variety of moving averages, oscillators, directional movement and volatility filters. It is a good idea to use a mix
of different indicators to reduce variable redundancy and provide the network with the ability to adapt to changing market conditions through periodic retraining.

Smoothing both input and output data by using either simple or exponential moving averages is often employed. Empirical work on testing the efficient market hypothesis has found that prices exhibit time dependency or positive autocorrelation while price changes around a trend are somewhat random [3.48]. Therefore, attempting to predict price changes around the trend by using either unfiltered prices or price changes as inputs may prove to be difficult. Using moving averages to smooth the independent variables and forecasting trends may be a more promising approach.

Sampling or filtering of data refers to removing observations from the training and testing sets to create a more uniform distribution. The type of filtering employed should be consistent with the objectives of the researcher. For example, a histogram of price changes for a commodity would reveal many small changes from which an off-floor speculator cannot profit after deducting realistic execution costs. However, this dense region of the distribution will greatly impact the training of the neural network since small price changes account for the majority of the training facts. The network minimizes the sum of squared errors (or other error function) over all the training facts. By removing these small price changes, overall trading performance can be improved since the network specializes in the larger, potentially profitable price changes. It is possible for trading systems to be unprofitable even if the neural network predicted 85% of the turning points, as the turning points may be only small unimportant price changes [3.49]. On the other hand, a floor trader holding positions overnight is likely to be interested in these small price changes. The researcher must be clear on what exactly the neural network is supposed to learn. Another advantage of filtering is a decrease in the number of training facts which allows testing of more input variables, random starting weights or hidden neurons rather than training large data sets.

In practice, data pre-processing involves much trial and error. One method to select appropriate input variables is to test various combinations. For example, a “top 20” list of variables consisting of a variety of technical indicators could be pre-tested ten at a time with each combination differing by two or three variables. Although computationally intensive, this procedure recognizes the likelihood that some variables may be excellent predictors only when in combination with other variables.
Also, the top 20 list can be modified over time as the researcher gains experience on the type of pre processing that works for his/her application. This approach is especially useful if the training set is small relative to the number of parameters (weights) which is likely to be the case if all 20 input variables are presented to the neural network at once.

3.5.4 Data partitioning

Common practice is to divide the time series into three distinct sets called the training, testing and validation (out-of-sample) sets. The training set is the largest set and is used by the neural network to learn the patterns present in the data. The testing set, ranging in size from 10% to 30% of the training set, is used to evaluate the generalization ability of a supposedly trained network.

The researcher would select the network(s) which perform best on the testing set. A final check on the validation set chosen must strike a balance between obtaining a sufficient sample size to evaluate a trained network and having enough remaining observations for both training and testing. The validation set should consist of the most recent contiguous observations. Care must be taken not to use the validation set as a testing set by repeatedly performing a series of train-test-validation steps and adjusting the input variables based on the network's performance on the validation set. The testing set can be either randomly selected from the training set or consist of a set of observations immediately following the training set. The advantage of randomly selecting testing facts is that the danger of using a testing set characterized by one type of market is largely avoided. For example, a small testing set may only consist of prices in a strong uptrend. The testing set will favour networks which specialize on strong up trends at the expense of networks which generalize by performing well on both uprends and downtrends. The advantage of using the observations following the training set as testing facts is that these are the most recent observations (excluding the validation set) which may be more important than older data.

The randomly selected testing facts should not be replaced in the training set because this would bias the ability to evaluate generalization especially if the testing set is large relative to the training set (e.g. 30%). A deterministic method, such as selecting every n\textsuperscript{th} observation as a testing fact, is also not recommended since it can
result in cycles in the sampled data due solely to the sampling technique employed [3.3].

A more rigorous approach in evaluating neural networks is to use a walk-forward testing routine also known as either sliding or moving window testing. Popular in evaluating commodity trading systems, walk-forward testing involves dividing the data into a series of overlapping training-testing-validation sets. Each set is moved forward through the time series. Walkforward testing attempts to simulate real-life trading and tests the robustness of the model through its frequent re-training on a large out-of-sample data set.

In walk-forward testing, the size of the validation set drives the retraining frequency of the neural network. Frequent retraining is more time consuming, but allows the network to adapt more quickly to changing market conditions. The consistency or variation of the results in the out-of-sample sets is an important performance measure.

3.5.5 Network architecture

There are an infinite number of ways to construct a neural network. Neurodynamics and architecture are two terms used to describe the way in which a neural network is organized. The combination of neurodynamics and architecture define the neural network's paradigm. Neurodynamics describe the properties of an individual neuron such as its transfer function and how the inputs are combined [3.44]. A neural network's architecture defines its structures including the number of neurons in each layer and the number and type of interconnections.

The number of input neurons is one of the easiest parameters to select once the independent variables have been pre-processed because each independent variable is represented by its own input neuron. The tasks of selection of the number of hidden layers, the number of neurons in the hidden layers, the number of output neurons as well as the transfer functions are much more difficult.

*Number of hidden layers*

The hidden layer(s) provide the network with its ability to generalize. In practice, neural networks with one and occasionally two hidden layers are widely
used and have performed very well. Increasing the number of hidden layers also increases computation time.

Overfitting occurs when a forecasting model has too few degrees of freedom. In other words, it has relatively few observations in relation to its parameters and therefore it is able to memorize individual points rather than learn the general patterns. In the case of neural networks, the number of weights, which are linked to the number of hidden layers and neurons, and the size of the training set (number of observations) determine the likelihood of overfitting [3.3], [3.50]. The greater the number of weights relative to the size of the training set, the greater the ability of the network to memorize idiosyncrasies of individual observations. As a result, generalization for the validation set is lost and the model is of little use in actual forecasting.

Therefore, it is recommended that all neural networks should start with preferably one or at most two hidden layers. If a four-layer neural network (i.e. two hidden layers) proves unsatisfactory after having tested multiple hidden neurons using a reasonable number of randomly selected starting weights, then the researcher should modify the input variables a number of times before adding a third hidden layer. Virtually all empirical work suggests that networks with more than four layers will not improve the results.

**Number of hidden neurons**

Despite its importance, there is no "magic" formula for selecting the optimum number of hidden neurons. Therefore researchers fall back on experimentation. However, some rules of thumb have been advanced. A rough approximation can be obtained by the geometric pyramid rule proposed by Masters [3.3]. For a three layer network with \( n \) input neurons and \( m \) output neurons the hidden layer would have \( \sqrt{n \cdot m} \) neurons. The actual number of hidden neurons can still range from one-half to two times the geometric pyramid rule value depending on the complexity of the problem. Baily and Thompson in [3.51] suggest that the number of hidden neurons in a three-layer neural network should be 75 % of the number of input neurons. Katz in [3.52] indicates that an optimal number of hidden neurons will generally be found between one-half to three times the number of input neurons. Klimasauskas in [3.53]
suggests that there should be at least five times as many training facts as weights, which sets an upper limit on the number of input and hidden neurons.

It is important to note that the rules which calculate the number of hidden neurons as a multiple of the number of input neurons implicitly assume that the training set is at least twice as large as the number of weights and preferably four or more times as large. If this is not the case, then these rules of thumb can quickly lead to overfitted models since the number of hidden neurons is directly dependent on the number of input neurons (which in turn determine the number of weights). The solution is to either increase the size of the training set or, if this is not possible, to set an upper limit on the number of input neurons so that the number of weights is at least half of the number of training facts. Selection of input variables becomes even more critical in such small networks since the luxury of presenting the network with a large number of inputs and having it ignore the irrelevant ones has largely disappeared.

Selecting the best number of hidden neurons involves experimentation. Three methods often used are the fixed, constructive and destructive. In the fixed approach, a group of neural networks with different numbers of hidden neurons are trained and each is evaluated on the testing set using a reasonable number of randomly selected starting weights. The increment in the number of hidden neurons may be one, two or more depending on the computational resources available. Plotting the evaluation criterion (e.g. sum of squared errors) on the testing set as a function of the number of hidden neurons for each neural network generally produces a bowl shaped error graph. The network with the least error found at the bottom of the bowl is selected because it is able to generalize best. This approach is time consuming, but generally works very well.

The constructive and destructive approaches involve changing the number of hidden neurons during training rather than creating separate networks each with a different number of hidden neurons, as in the fixed approach. Many commercial neural network software packages do not support the addition or removal of hidden neurons during training. The constructive approach involves adding hidden neurons until network performance starts deteriorating. The destructive approach is similar except that hidden neurons are removed during training.

Regardless of the method used to select the range of hidden neurons to be tested, the rule is to always select the network that performs best on the testing set
with the least number of hidden neurons. When testing a range of hidden neurons it is important to keep all other parameters constant. Changing any parameter in effect creates a new neural network with a potentially different error surface which would needlessly complicate the selection of the optimum number of hidden neurons.

**Number of output neurons**

Deciding on the number of output neurons is somewhat more straightforward since there are compelling reasons to always use only one output neuron. Neural networks with multiple outputs, especially if these outputs are widely spaced, will produce inferior results as compared to a network with a single output [3.3]. A neural network trains by choosing weights such that the average error over all output neurons is minimized. For example, a neural network attempting to forecast futures prices of one month ahead and six months ahead will concentrate most of its effort on reducing the forecast with the largest error which is likely the six month forecast. As a result, a relatively large improvement in the one month forecast will not be made if it increases the absolute error of the six month forecasts by an amount greater than the absolute improvement of the one month forecast. The solution is to have the neural networks specialize by using separate networks for each forecast. Specialization also makes the trial and error design procedure somewhat simpler since each neural network is smaller and fewer parameters need to be changed to fine tune the final model.

**Transfer functions**

The majority of current neural network models use the sigmoid transfer function, but others such as the hyperbolic tangent and linear transfer functions have also been proposed. Linear transfer functions are not useful for nonlinear mapping and classification. Levich and Thomas [3.54] and Kao and Ma [3.55] found that financial markets are nonlinear and have memory suggesting that nonlinear transfer functions are more appropriate. Transfer functions such as the sigmoid are commonly used for time series data because they are nonlinear and continuously differentiable which are desirable properties for network learning. Klimasauskas in [3.53] states that, if the network is to learn average behaviour, a sigmoid transfer function should be used.
while if learning involves deviations from the average, the hyperbolic tangent function works best. In a standard back propagation network, the input layer neurons typically use linear transfer functions while all other neurons use a sigmoid function. The raw data is usually scaled between 0 and 1 or -1 and +1, so it is consistent with the type of transfer function which is being used. Linear and mean/standard deviation scaling are two of the most common methods used in neural networks. In linear scaling all observations are linearly scaled between the minimum and maximum values.

Simple linear scaling is susceptible to outliers because it does not change the uniformity of the distribution, but merely scales it into the appropriate range for the transfer function. For S&P 500 data, linear scaling results in 98.6% of the training facts being contained within 10% of the neuron’s activation range. Proper training is unlikely to take place with such a distribution. In mean and standard deviation scaling all values plus or minus \( x \) number of standard deviations from the mean are mapped to one and zero respectively. All other values are linearly mapped between zero and one. This type of scaling creates a more uniform distribution and is more appropriate for data which has not been sampled in any way. Most neural network software programs will automatically scale all the variables into the appropriate range. However, it is always a good idea to look at histograms of the scaled input and output variables.

3.5.6 Evaluation of the system

The most common error function minimized in neural networks is the sum of squared errors. Other error functions offered by software vendors include least absolute deviations, least fourth powers, asymmetric least squares and percentage differences. These error functions may not be the final evaluation criteria since other common forecasting evaluation methods such as the mean absolute percentage error (MAPE) are typically not minimized in neural networks.

In the case of commodity trading systems, the neural network forecasts would be converted into buy/sell signals according to a predetermined criterion. For example, all forecasts greater than 0.8 or 0.9 can be considered buy signals and all forecasts less than 0.2 or 0.1 as sell signals [3.56]. The buy/sell signals are then fed into a program to calculate some type of risk adjusted return and the networks with
the best risk adjusted return (not the lowest testing set error) would be selected. Low forecast errors and trading profits are not necessarily synonymous since a single large trade forecasted incorrectly by the neural network could have accounted for most of the trading system’s profits.

3.5.7 Training the neural network

Training a neural network to learn patterns in the data involves iteratively presenting it with examples of the correct known answers. The objective of training is to find the set of weights between the neurons that determine the global minimum of the error function. Unless the model is overfitted, this set of weights should provide good generalization. The backpropagation network uses a gradient descent training algorithm which adjusts the weights to move down the steepest slope of the error surface. Finding the global minimum is not guaranteed since the error surface can include many local minima in which the algorithm can become “stuck”. A momentum term and five to ten random sets of starting weights can improve the chances of reaching a global minimum. The following two sections will discuss when to stop training a neural network and the selection of learning rate and momentum values.

Number of training iterations

There are two schools of thought regarding the point at which training should be stopped. The first stresses the danger of getting trapped in a local minimum and the difficulty of reaching a global minimum. The researcher should only stop training when there is no improvement in the error function based on a reasonable number of randomly selected starting weights [3.3]. The point at which the network does not improved is called convergence. The second view advocates a series of train-test interruptions [3.49], [3.57]. Training is stopped after a predetermined number of iterations and the network’s ability to generalize on the testing set is evaluated and training is resumed. Generalization is the idea that a model based on a sample of the data is suitable for forecasting the general population. The network for which the testing set error bottoms out is chosen since it is assumed to generalize best.
The criticism of the train-test procedure is that additional train-test interruptions could cause the error on the testing set to fall further before rising again or it could even fall asymptotically. In other words, the researcher has no way of knowing if additional training could improve the generalization ability of the network especially since starting weights are randomized.

Both schools of thought agree that generalization on the validation set is the ultimate goal and both use testing sets to evaluate a large number of networks.

The point at which the two approaches differ is the notion of overtraining versus overfitting. The convergence approach states that there is no such thing as overtraining, only overfitting. Overfitting is simply a symptom of a network that has too many weights. The solution is to reduce the number of hidden neurons (or hidden layers if there is more than one) and/or increase the size of the training set. The train-test approach attempts to guard against overfitting by stopping training based on the ability of the network to generalize.

The advantage of the convergence approach is that one can be more confident that the global minimum was reached. Replication is likely more difficult for the train-test approach given that starting weights are usually randomized and the mean correlation can fluctuate wildly as training proceeds. Another advantage is that the researcher has two less parameters to worry about; namely the point at which to stop training and the method to evaluate which of the train-test networks is optimal. An advantage of the train-test approach may be that networks with few degrees of freedom (weights) can be implemented with better generalization than convergence training which would result in overfitting. However, empirical work has not specifically addressed this issue. The train-test approach also requires less training time.

The objective of convergence training is to reach a global minimum. This requires training for a sufficient number of iterations using a reasonable number of randomly selected starting weights. Even then there is no guarantee with a backpropagation network that a global minimum is reached, since it may become trapped in a local minimum. In practice, computational resources are limited and tradeoffs arise. The researcher must juggle the number of input variable combinations to be trained, the interval of hidden neurons over which each network is to be tested, the number of randomly selected starting weights, and the maximum number of runs.
One method to determine a reasonable value for the maximum number of runs is to plot the mean correlation, sum of squared errors, or other appropriate error measure for each iteration or at predetermined intervals up to the point where improvement is negligible. Each iteration can be easily plotted if the neural network software creates a statistics file or, if this is not the case, the mean correlation can be recorded at intervals of 100 or 200 from the computer monitor. After plotting the mean correlation for a number of randomly selected starting weights, the researcher can choose the maximum number of runs based on the point where the mean correlation stops increasing quickly and flattens.

Many studies that mention the number of training iterations report convergence from 85 to 5000 iterations [3.47], [3.58]. However, the range is very wide as 50,000 and 190,000 iterations [3.53], [3.59] and training times of 60 hours have also been reported [3.56]. Training is affected by many parameters such as the choice of learning rate and momentum values, proprietary improvements to the backpropagation algorithm, among others, which differ between studies and so it is difficult to determine a general value for the maximum number of runs. Also, the numerical precision of the neural network software can affect training because the slope of the error derivative can become very small causing some neural network programs to move in the wrong direction due to round off errors which can quickly build up in the highly iterative training algorithm.

It is recommended that researchers determine the number of iterations required to achieve negligible improvement for their particular problem and test as many randomly selected starting weights as computational constraints allow.

**Learning rate and momentum**

During training, a learning rate that is too high is revealed when the error function is changing wildly without showing a continued improvement. A small learning rate and it will be unable to escape from local minima which is evident during training when there is very little or no improvement in the error function. A very small learning rate also requires more training time. In either case, the researcher must adjust the learning rate during training or “brainwash” the network by randomizing all weights and changing the learning rate for the new run through the training set.
One method to increase the learning rate and thereby speed up training time without leading to oscillation is to include a momentum term in the backpropagation learning rule. The momentum term determines how past weight changes affect current weight changes.

The momentum term suppresses side to side oscillations by filtering out high frequency variations. Each new search direction is a weighted sum of the current and the previous gradients. Such a two-period moving average of gradients filters out rapid fluctuations in the learning rate. Momentum values that are too great will prevent the algorithm from following the twists and turns in weight space. McClelland et al. [3.60] indicate that the momentum term is especially useful in error spaces containing long canyons that are characterized by steep, high walls and a gently sloping floor. Without a momentum term, a very small learning rate would be required to move down the floor of the canyon which would require excessive training time. By dampening the oscillations between the canyon walls, the momentum term can allow a higher learning rate to be used.

Most neural network software programs provide default values for learning rate and momentum that typically work well. Common practice is to start training with a higher learning rate and decrease as training proceeds. Many neural network programs will automatically decrease the learning rate and increase momentum as convergence is reached.

3.5.8 Implementation

The implementation step is listed as the last one, but in fact requires careful consideration prior to collecting data. Data availability, evaluation criteria and training times are all shaped by the environment in which the neural network will be deployed. Most neural network software vendors provide the means by which trained networks can be implemented either in the neural network program itself or as an executable file.

If not, a trained network can be easily created in a spreadsheet by knowing its architecture, transfer functions and weights. Care should be taken that all data transformations, scaling and other parameters remain the same from testing to actual use.
An advantage of neural networks is their ability to adapt to changing market conditions through periodic retraining. Once deployed, a neural network’s performance will degrade over time unless retraining takes place. However, even with periodic retraining, there is no guarantee that network performance can be maintained as the independent variables selected may have become less important.

It is recommended that the frequency of retraining for the deployed network should be the same as used during testing on the final model. However, when testing a large number of networks to obtain the final model, less frequent retraining is acceptable in order to keep training times reasonable. A good model should be robust with respect to retraining frequency and will usually improve as retraining takes place more often.
Chapter 4

A Hybrid Model

4.1 Preliminaries

In previous chapters we showed that most conventional tools used in modelling financial time series suffer from the same limitations concerning their ability to work with high-dimensional data. In the literature, one can find that sophisticated models can "beat the curse of dimensionality". In fact, it is true that many tools like, for example, sophisticated versions of neural networks perform better than many other conventional tools, when the dimension of the data increases. However it is a fact that the intrinsic dimensionality of data is usually lower than the dimension of the data vectors therefore using dimensionality reduction techniques may help to bypass the problems of high dimensions.

4.2 Dimensionality reduction & linear transformations

In most real-world situations, information is complex, and many features are necessary to describe data. Data analysis and statistical tools make it possible to extract information from high-dimensional data, in order to make this information usable in an intuitive way. The process of extracting information from high dimensional data is equivalent to finding a suitable representation of the data by means of a suitable transformation. It is important for subsequent analysis of the data, that the data is represented in a manner that facilitates the analysis. In most of the cases such representation is sought as a linear transform of the observed variables. Using linear transformations makes the problem computationally and conceptually
simpler and facilitates the interpretation of the results. In financial data analysis where many parallel time series are available, these transformations can reveal some common underlying driving mechanisms that would otherwise remain hidden.

4.2.1 Common linear transformation methods

Several principles and methods have been developed to find suitable linear transformations. The most representative ones are principal component analysis (PCA) and independent component analysis (ICA).

PCA is an optimal linear dimension reduction technique in the mean-square sense. It's in other words a “faithful” transformation of the data as far as reconstruction error is concerned.

ICA, on the other hand, is searching for a “meaningful” representation. Although meaningfulness is a task-dependent property, ICA seems to be able to find meaningful representations in a wide variety of applications [4.1, 4.2, 4.3, 4.4].

4.2.2 “Faithful” transformations: Principal component analysis

PCA is a member of the “second-order class” of linear techniques. That is because its aim is to optimally reduce dimensionality in the mean square sense. It finds representations using only the information contained in the covariance matrix of the data vector and thus assumes variables with normal or Gaussian distribution because such distributions are completely determined by the second order information.

Principal components is a classical way of representing a universe of financial assets by a linear combination of a smaller number of factors. These representations are mutually uncorrelated but not independent. This, in conjunction with the fact that financial asset returns are fat-tailed, makes the modelling of asset price co-movements particularly difficult especially when the asset price changes are large.

4.2.3 “Meaningful” representations: Independent component analysis

ICA is an alternative methodology to principal components. It is a method from the “higher-order class” of linear techniques, and is used to find “interesting” projections of multidimensional data, or otherwise underlying signals that are mutually independent (and thus uncorrelated). This in practice means that the fourth-
order cross-moments (co-kurtosis) between them is zero (see figure 4.1 below). ICA uses information of the distribution of the data vector that is not contained in the covariance matrix. In order for this to be meaningful the distribution of the data vector must be assumed non-Gaussian. Financial asset price distributions are indeed non-Gaussian which makes this methodology ideal for their representation.

![Fourth Order Cumulant of Five Principal Components](a)

![Fourth Order Cumulant of Five Independent Components](b)

**Figure 4.1:** Projection of the normalized fourth order cumulants of five a) principal and b) independent component deduced from a set of financial time series. The peaks along the diagonal are the kurtosis terms of each component and the off-diagonal peaks are the co-kurtosis terms which represent the co-movements of the fat tails of the component distributions. The off-diagonal peaks in (a) are clearly non zero whereas in (b) they are almost zero.

One major drawback of ICA is that its definition does not imply any sort of ordering of the independent components which is in contrast with PCA. Researchers have suggested two ways for ordering independent components. The first one is to
use the norms of the columns of the mixing matrix which give the contributions of
the independent components to the variances of the data vector. This ordering is
reminiscent to the ordering of the PCA as the components are ordered according to
the descending norm of the corresponding columns of the mixing matrix. The second
suggestion for ordering independent components is to sort them from most to least
non-Gaussian component. When applied to financial data modelling, both these
methods don’t give good results and this is mainly due to the lack of knowledge of
the number and the statistical properties of the actual underlying factors that drive
the universe. Also the fact that this is a linear transformation on a truly non-linear
data set makes the attempts of ordering the components via direct association of their
properties with the properties of the input data, more unreliable.

4.2.4 The need for non linear projections

Both PCA and ICA are linear transformations. This means that any linear
dependency between features can be easily detected, and adequately overcome.
However, linear tools are not adapted to detect non-linear dependencies between
features. Moreover, real-world data are governed by processes that are inherently
non linear, except maybe in some simple, low-dimensional cases. The paradox is
that tools like PCA or ICA are used on their own to help in the analysis of real-world
data, but they are really adapted to problems which are far from reality.

4.3 Non Linear Projection Methods

Finding methods for non-linear projections of high-dimensional data to lower
dimensions has been a very hot topic lately. Despite the fact that some methods have
been known from many years now, the recent needs in data analysis techniques
considerably raised the interest towards non-linear projection. When faced with
difficulties resulting from the high dimension of the space, a possibility is to try to
decrease this dimension, of course without loosing relevant information in the data.
Dimension reduction is used as pre-processing, before applying data analysis models
on data with a lower dimension. Nonlinear projection means to find a lower-
dimensional space in which the data are described as well as in the original space.

There are several ways to design nonlinear projection methods. A first one
consists in using PCA, but locally in restricted parts of the space [4.5]. Joining local
linear models leads to a global nonlinear one. But unfortunately local PCA is not designed to represent the raw data set in a single coordinate system. This obviously compromises any attempt to consider the data set as a whole for any subsequent continuous process and therefore is of limited interest.

Kernel PCA [4.6] consists in first transforming the data into a higher-dimensional space, and then applying PCA on the transformed data. Kernel PCA benefits from the strong theoretical background of kernel methods, and is interesting in specific situations. However, the method suffers from a difficult choice of the initial transformation and from the apparent contradiction to increase the dimension of the data before reducing it.

Non-linear methods not suffering from the above PCA-linked disadvantages are the so called “distance preservation methods”. These methods minimize a criterion which completely differs from the one used by PCA. Their principle is to find a lower dimensional representation of data where the pair-wise distances are respected as much as possible with respect to the original data space.

Sammon’s nonlinear mapping [4.7] belongs to this class of methods. Short distances in the original space are favoured, to allow unfolding of large, nonlinear surfaces and volumes. Demartines and Herault’s CCA (Curvilinear Component Analysis) [4.8] greatly improves Sammon’s method by giving more weight to short distances in the projection space instead of the original one. The limitations of these algorithms come from the distortions that can exist between the distances measured in the data space and the distances measured in the manifold space.

The last evolutions of this kind of method avoid partly the problem by measuring the distances in the original space along a manifold, instead of taking the Euclidean distance between pairs of points. This is achieved using more complex metrics than the Euclidean distance [like the Curvilinear Distance Analysis (CDA) [4.9] and Isomap [4.10]] which compute “curvilinear” or “geodesic” distances, respectively. Isomap results from a global approach of the model while CDA appears with a local approach of the problem.

The main advantage of a global approach, apart from speed, is that a solution always exists for any problem in the framework of the considered model. This solution minimizes an explicitly formulated criterion. The price to pay is often an unrealistic or too constraining model. Similarly the criterion does not always translate exactly to what the user expects.
Contrary to Isomap, CDA combines the advantages of the curvilinear measure and the larger weights on short distances, leading to efficient unfolding in a larger class of situations. One of the main disadvantages of CDA, is that even when the problem perfectly fits the model, an inexperienced user might badly parameterize the algorithm. On the other hand, a well-parameterized local approach is more tolerant and generally delivers a good solution, even if the problem does not fit exactly the underlying model. Obviously, the solution obtained is not guaranteed to be a global optimum, since no explicit criterion exists at the global level. In general it could be said that CDA relies on more complicated techniques and needs to be well parametrized.

Finally a last group of non-linear methods used in dimensionality reduction consists of the Self-Organizing Maps (SOM) or Kohonen’s Maps [4.11] which may be viewed as neighbour-preservation nonlinear projection tools. SOMs are used in representation tasks, where the dimension of the projection space is limited to three. This is clearly a limitation, as it cannot be expected that real data will have an intrinsic dimension of three or less. Nevertheless, from a theoretical point of view, there is no technical difficulty to extend the use of SOM to higher dimensional projection spaces. SOMs are used when a combination of vector quantization, clustering and projection is looked for. However, the loss of information in the transformation is not as limited as for the distance preservation methods.

4.3.1 Drawdown: Estimation of the system’s intrinsic dimension

The non-linear methodologies described above are used to exploit redundancies and dependencies between the raw data and reduce its dimension. This is possible because raw data often lie on a manifold whose dimension is smaller than the dimension of the embedding space.

It is evident from the comments in the previous section that an ideal algorithm does not exist for effective non-linear projection. This is mainly due to the quality criterion which is not as obvious as in the linear case and also due to the adaptive / iterative nature of the non-linear methods which makes them very sensitive to all the parameters and imposes continuous fine tuning.

The non-linear methodologies described in the section above aim to find an invertible continuous mapping between the embedding space and the unknown
Chapter 4

A hybrid model manifold space. A vital ingredient for a successful mapping is the knowledge of the number of explicative variables of the data set or the true dimensionality of the manifold. This is also commonly known as the intrinsic dimension of the data [4.12, 4.13]. Most of the non-linear modelling methodologies described above require a-priori knowledge of the intrinsic dimension in order to avoid both under- or over-fitting.

There are several methods that attempt to compute the intrinsic dimension of a system. The most common ones are the box counting method, the Grassberger-Procaccia method [4.13] and the a-posteriori computation of dimensions. These methods although extensively used, have some serious drawbacks which are explained below.

In the box counting method case when the intrinsic dimension is smaller than the dimension of the space (which is usually the case when the goal is to estimate the intrinsic dimension), the number of empty boxes dramatically increases, leading to a heavy computational load. Also the local character of the intrinsic dimension is only preserved when the largest of the boxes considered in the method is small enough to keep the constant density hypothesis valid (this makes the first drawback even worse).

The Grassberger-Procaccia method [4.13] is similar to the box-counting procedure while different in its implementation. In the Grassberger-Procaccia method, all distances between any pair of points are computed. The method then leads to a global intrinsic dimensionality (instead of a local one). However, the advantage is that \( N \) points give \( N(N-1)/2 \) mutual distances, so that the number of points necessary to have an acceptable estimation of the intrinsic dimension is lower than in the box-counting method.

The a-posteriori computation presents two major difficulties. First, it is clear that it is computationally intensive, since the projection method itself must be considered for several dimensions of the projection space, rather than once. Secondly, an appropriate criterion must be defined to evaluate the quality of the projection. Furthermore, even with an adequate criterion, the limit between “good” and “bad” projections may be difficult to set in practical situations.

It needs to be said that the implementation of all three methods remains very difficult, not to say impossible, for high dimensional data sets.
4.3.2 The "empty space phenomenon"

Although the computation of a system's intrinsic dimension is of great interest and practical use in adaptive and learning methods, the definition of the intrinsic dimension itself has been purposely kept vague. While all methods presented above are linked, there is no guarantee that two methods will give the same result on the same datasets.

But even if the definition of the intrinsic dimension and the way to estimate it were clear there would still be a problem which can be summarized as follows: in practice one virtually never has enough samples to be confident in the intrinsic dimension estimation.

The fact that the number of data points available in real applications is never sufficient for learning, at least in theory, is widely known as the "empty space phenomenon". The effects of this phenomenon are much worse in higher dimensions. That is why many data analysis tools based on learning mechanisms sometimes fail with high-dimensional data.

However, data represented in high-dimensional spaces do not necessarily fill the whole space. If they are located near a sub manifold, a convenient way to bypass some of the problems is to identify the dimension of the sub manifold (its intrinsic dimension) and to project the data in a lower dimensional space. If such projection is not possible, one has to develop learning methods dealing with high dimensional data. In the following section we present such an alternative methodology.

4.4 The proposed hybrid model

In the previous sections we showed that there exist linear and non-linear algorithms that reduce the dimensionality of the data space to enable forecasting models to overcome their limitations or failure. Unfortunately, we proved that most of these algorithms are unreliable when it comes to high dimensional non-linear data as they make unrealistic assumptions which can greatly affect the solution.

Having acknowledged the drawdowns of the dimensionality reduction techniques described in the previous sections we came up with a new methodology which combines some of the above techniques in a novel way to overcome some of the problems. Our methodology is based on the assumption that dimensionality
reduction by using only a subset of independent components is possible, since it is almost certain that a relatively small number of factors drive all asset prices.

Under this assumption, we initialise our methodology by an orthogonal decomposition of the asset price time series into principal components and from those we consequently deduce the "most important" independent components. We then model the behaviour of those independent components in an agnostic non-parametric manner using data mining techniques which fully maintain any nonlinear behaviour of the components. The resulting reconstruction 'residuals' are partly the result of nonlinear dependence amongst the prices (e.g. time varying, stochastic or time-scale dependent "correlations" induced by investor behaviour) and partly the result of random errors. Our methodology uses a neural network to extract the nonlinear behaviour of these residuals. The process ends by modelling and forecasting each one of the independent components and then constructing an overall hybrid model of asset price dynamics by combining the linearly reconstructed future returns (using the ICs) and the corresponding forecasted residuals.

In the following sections we will present the proposed hybrid forecasting methodology in more detail. The model's implementation procedure is summarized below (see also a schematic representation in figure 5.2).

Step 1: Select an original set of time series of asset returns and exogenous variables that could influence the series.
Step 2: Perform application-dependent pre-processing (optional)
Step 3: Define the training, validation and testing sets.
Step 4: Pre-process the data by centering and whitening to make the problem of ICA estimation simpler and better conditioned
Step 5: First system's dimensionality reduction by PCA
Step 6: Perform ICA to decompose the dataset into sets of "meaningful" independent (univariate) components
Step 7: Find an optimal number of independent components that will represent the initial multivariate dataset of asset returns
Step 8: Model each independent component separately, for example using neural networks and forecast into the future
Step 9: From (8) linearly reconstruct the expected future asset returns
Step 10: From (7) reconstruct the initial asset returns and find the residuals between the reconstructed and the original dataset

Step 11: Model the residuals and find the mixing matrix by using neural networks.

Step 12: Use the forecasted independent components of Step (8) and forecast the corresponding residuals

Step 13: Add the resulting non-linear residuals of (11) to the linearly reconstructed system of (8) to account for the non-linearities of the original dataset

The suggested technique provides significant improvements in representing spatial dependence compared with correlation-based methods such as PCA. In the following section we look at why and how we have implemented each one of the steps above and show some encouraging experimental results in forecasting financial time series.

Figure 4.2: Schematic representation of the process behind the hybrid model of chapter 4
4.5 Mathematical interpretation of ICA

Let's assume a set of time series of financial asset returns $x_i(t)$ with index $i$ indexing the individual time series $i = 1, 2, \ldots, n$ and $t$ denoting time instances. In the basic ICA the original signals $x_i(t)$ are instantaneous linear mixtures of independent source signals or underlying factors $s_j(t)$ where $j = 1, 2, \ldots, m$ with some unknown mixing coefficients $\alpha_{i,j}$ and an assumed number $m$. In that sense each signal $x_i(t)$ can be presented by:

$$x_i(t) = \sum_{j=1}^{m} \alpha_{i,j} s_j(t)$$  \hspace{1cm} (4.1)

We assume the effect of each time-varying underlying factor $s_j(t)$ on the measured time series to be approximately linear.

Utilising information of either high order moments or time structure of the observed time series $x_i(t)$ ICA is able to find good estimates for the underlying independent signals $s_j(t)$ and the unknown mixture coefficients $\alpha_{i,j}$.

If we convert the above to vector-matrix formulation then the mathematical representation of the financial time series becomes $x(t) = [x_1(t), \ldots, x_n(t)]$ with elements $x_i(t)$, the underlying factors $s(t)$ with elements $s_j(t)$, and matrix $A = (a_{i,j})$. Now we can write equation (4.1) as:

$$x(t) = As(t)$$  \hspace{1cm} (4.2)

Matrix $A$ is called the mixing matrix. The basic idea of ICA is that we do not need to know either the matrix $A$ nor the vector $s(t)$. Instead we can estimate the model and obtain both matrix $A$ and the underlying factors $s(t)$ given sequential observations on $x(t)$ and assuming that the factors are statistically independent. In financial applications this is the case with underlying factors like economic events that affect a number of simultaneous time series but can be assumed to be independent. Another assumption that needs to be made is that the components
Chapter 4

A hybrid model exhibit nongaussian distributions which are unknown. A final assumption which can sometimes be relaxed, as will be explained latter, is that the unknown mixing matrix is square. If this is the case and after estimating matrix $A$, its inverse can be computed, say $W$, and the independent component can be obtained by:

$$s(t) = Wx(t)$$ (4.3)

4.6 ICA Algorithms

In the basic approach to solve the ICA problem [4.14, 4.15, 4.16, 4.17, 4.18], the temporal structure of the time series is in fact omitted and $s(t)$ and $x(t)$ of equation (4.3) are regarded as realisations of random vectors $s$ and $x$. Equation (4.2), therefore, becomes

$$x = As$$ (4.4)

and the solution takes the form:

$$s = Wx$$ (4.5)

The objective is to now find a matrix $W$ that makes the elements of $s$ statistically independent. Such matrix is called the separating matrix. Cardoso in [4.15] gives a review of various information theoretic contrast functions for solving for $W$, like mutual information, negentropy, maximum entropy, infomax as well as the maximum likelihood approach and discusses the numerical problems in minimising / maximising such contrast functions.

The problem of solving for the separating matrix $W$ is somewhat simplified if we consider only one of the source signals at a time. From equation (4.5) it follows

$$s_i = w_i^T x$$

with $w_i$ the $i^{th}$ row of $W$. Cardoso has suggested and analysed neural type one-unit learning rules [4.19] that give as solutions one row $w_i^T$ of the separating matrix. A
condition of local convergence to a correct solution was given. The condition is very robust and shows that a wide range of non-linear functions in these learning rules are possible.

The problem is further simplified by performing a preliminary whitening of the data $x$. The observed vector $x$ is first linearly transformed to another vector whose elements are mutually uncorrelated and all have unit variances. This transformation is always possible and will be discussed in the following sections in more detail. At the same time the dimensionality of the data should be reduced so that the dimension of the transformed data vector equals, say $m$, the number of independent components. After this pre-processing $W$ will be an orthogonal matrix.

As an example of contrast functions, consider the simple case of maximising the kurtosis $E\{s_i^4\} - 3[E\{s_i^2\}]^2$ of the estimated signals $s_i$. Because we assumed that the estimated signals have unit variance this reduces to maximising the fourth moment $E\{s_i^4\}$. Its gradient with respect to $w_i$ is $4E\{(w_i^T x)^3 x\}$. In a gradient type learning rule, the row $w_i^T$ of the separating matrix $W$ would be sought using an instantaneous version of this gradient, in which the expectation is dropped and the gradient is computed separately for each input vector $x$. In addition a normalisation term would be needed that keeps the norm of $w_i$ equal to one ($W$ matrix must be orthogonal).

In this thesis we use a much more efficient algorithm which is implemented in MATLAB as FastICA. This algorithm is based on the following fixed point iteration [4.20]:

1. Take a random initial vector $w(0)$ of norm 1. Let $k = 1$.

2. Let $w(k) = E\{x(w(k - 1)^T x)^3\} - 3w(k - 1)$. The expectation can be estimated using a large sample of $x$ vectors.

3. Divide $w(k)$ by its norm.

4. If $|w(k)^T w(k - 1)|$ is not close enough to 1, let $k = k + 1$ and go back to step 2 otherwise output the vector $w(k)$.

The final vector $w(k)$ given by the algorithm equals the transpose of one of the rows of the orthogonal separating matrix $W$. To estimate $m$ independent
components we run this algorithm \( m \) times. To ensure that each time we estimate a different independent component we use the deflation algorithm that adds a simple orthogonalising projection inside the loop. Recall that the rows of the separating matrix \( W \) are orthonormal because of the whitening. Thus we can estimate the independent components one-by-one by projecting the current solution \( w(k) \) on the space orthogonal to the rows of the separating matrix \( W \) previously found.

An interesting property of the FastICA is that it uses fewer iterations than the conventional ICA technique and still obtains the maximum accuracy allowed by the sample data. This is due to the cubic convergence of the algorithm shown in [4.20].

### 4.7 Original data issues and ICA pre-processing

#### 4.7.1 Original data considerations

In many applications the reason behind the large number of variables can be the lack of information about the usefulness of the variables. In the prediction of financial data fluctuations in the price of a specific stock are usually thought of being largely influenced by fluctuations in the financial markets. One common approach to predict that stock's price is to use an as diversified universe of exogenous input variables as possible including values of many stock indices, assets from several financial sectors, interest rates, foreign exchange rates, macroeconomic data and even commodities prices as inputs to the forecasting method. As there is no standard way to construct such universe of assets we are tempted to use as many inputs as possible in order to effectively capture the underlying dynamics of the system. This increases the number of variables for a fixed number of observations, which is one of the main reasons for using ICA to reduce dimensionality. All necessary data was collected via either Bloomberg or Datastream.

Like most time series approaches ICA requires the observed signals to be stationary. A signal \( x(t) \) is considered to be stationary if the expected value is constant, or if after removing a constant mean \( E[x(t)] = 0 \). In this thesis for an asset \( i \), we transform the non stationary asset prices \( p_i(t) \) to asset returns as follows:

\[
r_i(t) = \frac{p_i(t) - p_i(t-1)}{p_i(t-1)}
\]
The data were carefully selected so as to be regularly spaced (daily data) and contain no missing values in the period under consideration. Finally, equity or bond data were adjusted for the effects of dividends or coupons.

4.7.2 Partition of data into training, validation and testing sets

At some point in time during the proposed procedure we will use neural networks to model and forecast the behaviour of independent components and their residuals into the future. To train and eventually test the neural network we will need to partition the data into three sets: the training, validation and testing sets.

Neural networks are trained on the training set, tested for data overfitting in the validation set, and finally tested for generalisation in the testing set. This last step provides a measure of how effective the neural network is with completely out-of-sample data. The testing set is used only after all of the prediction and the models have been trained and optimised.

Because of the dynamic behaviour of the financial markets it is very difficult to partition the data and avoid either biases in the prediction due to long training windows or too much estimation error due to limited training data. In this work we found that having larger sets of training data offsets the risk of overfitting. From the available samples we follow common practice and use 80% for training, 10% for validation and 10% for testing. To account for the evolution of the financial markets and allow the network to adapt to the changes we use a variable length window for training and validation.

4.7.3 General pre-processing

Centering

Before applying ICA to data it is usually very useful to do some pre-processing. The most basic and necessary pre-processing is to center $x$, i.e. subtract its mean vector $m = E(x)$ so as to make $x$ a zero-mean variable. This implies that $s$, where $s = Wx$, is zero mean as well, as can be seen by taking expectations on both sides of equation (4.2).

After measuring the mixing matrix $A$ with centered data, we can complete the estimation be adding the mean vector of $s$ back to the centered estimates of $s$. The
mean vector of \( s \) is given by \( A^{-1}m \), where \( m \), is the mean that was subtracted in the pre-processing.

**Whitening**

Another useful pre-processing strategy in ICA is to first whiten the observed variables which in our case are the financial time series. This means just after centering the time series, we linearly transform vector \( x \) so that a new white vector \( \tilde{x} \) is obtained, i.e. its components are uncorrelated and their variances equal unity. In other words, the covariance matrix of \( \tilde{x} \) equals the identity matrix:

\[
E\{\tilde{x}\tilde{x}^T\} = I
\]

One popular method for whitening is to use the eigenvalue decomposition of the covariance matrix \( E\{\tilde{x}\tilde{x}^T\} = ED\Gamma^T \), where \( E \) is the orthogonal matrix of eigenvectors of \( E\{\tilde{x}\tilde{x}^T\} \) and \( D \) is the diagonal matrix of its eigenvalues,

\[
D = \text{diag}(d_1, \ldots, d_n)
\]

Note that \( E\{\tilde{x}\tilde{x}^T\} \) can be estimated in a standard way from the sample \( x(1), \ldots, x(T) \). Whitening can now be done by

\[
\tilde{x} = ED^{-1/2}\Gamma^T x
\]

where the matrix \( D^{-1/2} \) is computed by

\[
D^{-1/2} = \text{diag}(d_1^{-1/2}, \ldots, d_n^{-1/2})
\]

and \( E\{\tilde{x}\tilde{x}^T\} = I \).

Whitening transforms the mixing matrix into another matrix \( \tilde{A} \). From equations (4.4) and (4.6) we get:

\[
\tilde{x} = ED^{-1/2}\Gamma^T A s = \tilde{A}s
\]

The new mixing matrix \( \tilde{A} \) is orthogonal and this is proved by:
\[ E\{xx^T\} = \tilde{A} E\{ss^T\} \tilde{A}^T = \tilde{A} \tilde{A}^T = I \]

In larger dimensions converting a matrix \(A\) into an orthogonal matrix \(\tilde{A}\) reduces the complexity of the system to almost estimating half the parameters. This is because instead of having to estimate the \(n^2\) parameters that are the elements of the original matrix \(A\), we only need to estimate the orthogonal matrix \(\tilde{A}\) which contains \(n(n-1)/2\) degrees of freedom.

**Principal component analysis**

PCA can be used together with whitening as it results in reducing the dimensionality of the system. This is achieved by discarding all the eigenvalues \(d_j\) of \(E\{xx^T\}\) that are very small and it often has the additional effect of reducing noise. It is up to the user to decide the number of principal components to keep. The nice thing is that as PCA is a linear transformation any dimension reasonably estimated by PCA (for example by explaining 95 or 99\% of the variance of the initial set) is usually larger than the true dimension of the data set.

### 4.8 Estimating the number of independent components

One of the main problems that researchers face when using ICA is that they can not estimate the exact number of independent components to be extracted from their system. Under the ICA framework, Ikeda [4.21, 4.22] proposed the use of the minimum description length (MDL) principle to select \(m\) underlying components. This principle is based on factor analysis and is superior to PCA in finding uncorrelated components when the data is noisy. In this thesis we decided to use this methodology as a extra tool towards the discovery of the true dimensions of our system.

Factor analysis is one of the well known techniques for analysing multivariate data. In factor analysis real valued \(n\)-dimensional observations \(x\) are modelled as

\[ x = Af + \varepsilon \]  

(4.7)
where \( x, \epsilon \in \mathbb{R}^n \), \( f \in \mathbb{R}^m \), \( A \in \mathbb{R}^{m \times m} \), \( \Sigma \in \mathbb{R}^{m \times m} \): diagonal matrix

The goal of factor analysis is to estimate \( m \), \( A \) (factor loading matrix) and \( \Sigma \) (unique variance matrix) using the second order statistics \( C \) which for a data set \( x_i \), \( i = 1, \ldots, N \) is the covariance matrix

\[
C = \frac{1}{N} \sum_{i=1}^{N} x_i x_i^T
\]

The difference of equation (4.4) and equation (4.7) is that \( f \) is assumed to be drawn from a normal distribution but \( s \) is not. This is something that we acknowledge but are willing to explore as this method could be used as an extra tool to direct us towards the true dimension of the system in question.

Suppose we know \( A \) and \( \Sigma \). Let \( Q \in \mathbb{R}^{m \times m} \) be the pseudo-inverse of \( A \) where \( AQA = A \) holds and we transform the data as \( z = Qx \). Then \( z \) becomes the sphered data because

\[
\frac{1}{N} \sum z z^T = I_m + Q\Sigma\Sigma^T
\]

Pseudo-inverse has \((n - m)m\) degrees of freedom. And we selected the one which minimises \( tr(Q\Sigma\Sigma^T) \). This part shows that we can make the part of observation \( x \) due to the sources uncorrelated, and this is the aim of sphering. Therefore we have to estimate \( A \) and \( m \).

For the estimation of \( m \), \( A \) (factor loading matrix) and \( \Sigma \) (unique variance matrix), we can use some techniques of factor analysis. There are various estimation methods for \( A \) and \( \Sigma \) when \( m \) is given. We are going to use the Maximum Likelihood Estimation (MLE). MLE is defined as

\[
(\hat{A}, \hat{\Sigma})_{MLE} = \arg \max_{A, \Sigma} \left( -\frac{1}{2} tr(C(C + AA^T)^{-1}) + \log(\det(C + AA^T)) + n \log 2\pi \right)
\]

For solving the above equation we can use the gradient descent algorithm. In order to select the number of factors, \( m \), there are many approached and we are
going to use the Minimum Description Length (MDL) principle. MDL is defined as follows,

\[
MDL = -L(\hat{A}, \hat{\Sigma}) + \frac{\log N}{N} \times \text{the number of free parameters}
\]  

There are \(n(m+1)\) parameters in \(A\) and \(\Sigma\). But \(A\) has an ambiguity of rotation and \(m(m-1)/2\) is the freedom of the ambiguity. Subtracting \(m(m-1)/2\) from \(n(m+1)\) the number of free parameters is \(n(m+1) - m(m-1)/2\) and equation (4.8) can be written as

\[
MDL = -L(\hat{A}, \hat{\Sigma}) + \frac{\log N}{N} \times \left( \frac{n(m+1) - m(m-1)}{2} \right)
\]  

For the existence of the estimates, a necessary condition for \(A\) has been derived [4.31]. It comes from the fact that \(n(n+1)/2 \geq n(m+1) - m(m-1)/2\) has to be satisfied since \(C\) only has \(n(n+1)/2\) degrees of freedom. By taking \(m < n\) into account, the following bound is obtained.

\[
m \leq \frac{1}{2} \left( 2n + 1 - \sqrt{8n + 1} \right)
\]

A non-linear transformation for dimensionality reduction is usually used as pre-processing to a learning task. The MDL helps by giving an estimate of how many components are needed for linearly capturing most of the information. In difficult situations though, when for example the number of observations is low we suggest trying several dimensions (number of ICs), performing the learning task on each dimension, and finally selecting the dimension that gives the best reconstruction result. This procedure is computationally complex but sometimes helps in getting closer to the true dimension of the system.

4.9 Selecting important independent components

After estimating the number of components that need to be used, we choose the components that constitute that set. In order for this to happen we need to measure
various properties of the components to assist us in picking the right candidates. A number of criteria have been used to measure component properties. One of them is Kurtosis. This is the fourth-order cumulant which has been widely used in the ICA community to measure the nongaussianity of a signal [4.26, 4.27]. A nongaussian signal is unlikely to be the result of a mixture of signals [4.28]. So kurtosis is introduced to select those nongaussian signals. A gaussian random variable has zero kurtosis. Subgaussian and supergaussian variables would have positive and negative kurtosis respectively. Another criterion to measure component properties is the Euclidean norm which measures the energetic significance of the component so that the most energetically significant component appears first [4.23, 4.24, 4.15]. Norm is a criterion which has been used extensively. It focuses on the maximum value of the factors. In this thesis we use the \( L_{\infty} \) norm to measure those components causing the maximum price change in the stock [4.25].

If the suggested number for the set of independent components, found in the previous section, is \( q \), only the first \( q \) components in descending order of \( L_{\infty} \) norm will be considered.

### 4.10 Forecasting the ICs

#### 4.10.1 Smooth independent components

Finding “meaningful” components for a particular data set may rely on some specifically targeted pre-processing steps applied directly to the original data. These can be for example some sort of filtering like band pass filtering.

The interesting fact about this kind of pre-processing is that ICA can be applied to the new transformed time-series with the same mixing matrix. If \( X \) is the matrix that contains \( x(1),...,x(T) \) and similarly \( S \) is the matrix that contains \( s(1),...,s(T) \) then by performing ICA we get \( X = AS \). Filtering of \( X \) is equivalent to multiplying \( X \) from the right by a matrix, say \( M \) which makes the ICA model still valid as shown below:

\[
X' = XM = ASM = AS'
\]

Because our initial data set of asset returns is a continuous data set and we need the output filtered values to be at the same rate as the raw data we use the Savitsky-
Golay smoothing filter which is well adapted for data smoothing. Another important issue with this kind of filter is that it preserves features of the data such as peak height and width which are usually not preserved by adjacent averaging. This filter derives from a particular formulation of the data smoothing problem in the time domain and its goal is to find filter coefficients $c_n$ in the expression

$$\tilde{s}_i = \sum_{n=-n_{\text{LEFT}}}^{n_{\text{RIGHT}}} c_n s_{i+n}$$

(4.11)

where \{s_{i+n}\} represent the values for the ICs in a window of length $n_{\text{LEFT}} + n_{\text{RIGHT}} + 1$ centered on $i$ and $\tilde{s}_i$ are the smoothed ICs preserving higher moments [4.29].

Equivalently the idea is to approximate the underlying function within the moving window not by a constant (whose estimate is the average), but by a polynomial of higher order. This procedure for each point $s_i$, least-squares fit a polynomial to all $n_{\text{LEFT}} + n_{\text{RIGHT}} + 1$ points in the moving window and then sets $\tilde{s}_i$ to be the value of that polynomial at position $i$. When it moves on to the next point $s_{i+1}$ a whole new least-squares fit is performed using a shifted window [4.29].

All these least-squares fits would be very difficult to compute if done as described in the above paragraph. Luckily, since the process of least-squares fitting involves only a linear matrix inversion, the coefficients of a fitted polynomial are themselves linear in the values of the data. Because of this one can do all the fitting in advance and do the fits on the real data by taking linear combinations [4.29].

There are particular sets of filter coefficients $c_n$ for which equation (4.11) automatically accomplishes the process of polynomial least-squares fitting inside a moving window. To derive such coefficients consider how $\tilde{s}_0$, for example, could be obtained. If we wanted to fit a polynomial of degree $M$ in $i$, namely $a_0 + a_1 i + \ldots + a_M i^M$ to the values $s_{-n_{\text{LEFT}}}, \ldots, s_{n_{\text{RIGHT}}}$ then $\tilde{s}_0$ would be the value of that polynomial at $i = 0$, namely $a_0$. For the above we need the following set up:

$$A_y = i^j, \quad i = -n_{\text{LEFT}}, \ldots, n_{\text{RIGHT}}, \quad j = 0, \ldots, M$$
and the normal equations for the vector of $a_j$'s in terms of the vector of $s_i$'s is in matrix notation

$$(A^T A) \cdot a = A^T f \quad \text{or} \quad a = (A^T A)^{-1} (A^T f)$$

There are also the specific forms

$$\{A^T A\}_{ij} = \sum_{k=-n_{\text{LEFT}}}^{n_{\text{RIGHT}}} A_{ki} A_{kj} = \sum_{k=-n_{\text{LEFT}}}^{n_{\text{RIGHT}}} k^{i+j}$$

And

$$\{A^T f\}_{j} = \sum_{k=-n_{\text{LEFT}}}^{n_{\text{LEFT}}} A_{kj} f_{k} = \sum_{k=-n_{\text{LEFT}}}^{n_{\text{LEFT}}} k^{j} f_{k}$$

Since the coefficient $c_n$ is the component $a_0$ when $f$ is replaced by the unit vector $e_n$, $-n_{\text{LEFT}} \leq n < n_{\text{RIGHT}}$ then:

$$c_n = \{(A^T A)^{-1} \cdot (A^T e_n)\}_0 = \sum_{m=0}^{M} \{(A^T A)^{-1}\}_{0m} \cdot n^{m}$$

Based on the above equation there is only need for one row of the inverse matrix which can be done numerically by LU decomposition with only a single back substitution.

We come to an answer to what the optimal smoothing tolerance of our system is, by using the above smoothing method and observing smoothed curves on a graphic display. The procedure is the following: smooth with an initial tolerance, visualize both the smoothed and original time series and smooth again with the tolerance most visually appropriate so that the characteristics of the time series are preserved.

When dealing with noisy data it makes sense to use a more slowly varying curve which does not interpolate the data points but damps out the noise component. By increasing the smoothness of the curve the variance of predicted values is
reduced. However, in the same time this can introduce bias in the predicted values where the true underlying relationship is changing rapidly.

The smoothing could be also done in a more principled manner: a first step in this direction could be the minimization of the final prediction error using cross-validation.

![Smoothed signal](image)

**Figure 4.3:** Smoothed signal (red) after having applied the Savitsky-Golay filter on the original signal (blue)

### 4.10.2 Neural network for forecasting independent components

Now that the issue of the system's dimensionality has been addressed and the system has been successfully decomposed into "meaningful" independent components, or driving factors, we are still faced with the equally difficult problem of forecasting the non linear behaviour of those components.

In order to model the independent components we use a multi-layer perceptron, the most commonly used neural network architecture. The multi-layer perceptron is composed of hierarchical layers of neurons arranged so that information flows from the input layer to the output layer of the network. The device hence provides a non-linear mapping of input vectors to output responses.

\[ F : R^{I_n} \rightarrow R^{O_n} \]

where \( I_n \) is the number of neurons in the input layer and \( O_n \) is the number of neurons in the output layer. The particular mapping performed by the multi-layer is
determined by the adjustable weighted links between nodes. It can be shown that a
three layer multi-layer perceptron with an arbitrarily large number of nodes in the
hidden layer acts as a universal approximator and can realise any continuous
function [4.30].

In neural networks a relatively large number of parameters, together with the
inherent non-linearity will produce a non-convex error function with a large number
of local minima. In this thesis we will use the simple 3-layer architecture shown in
figure 4.4.

Because the components are independent we can model each one of them
separately and therefore we are dedicating a neural network to each one of them.
Because of this, we only utilise one time-series of independent components at a time
as input to the network.

As seen in figure 4.4, to improve the neural networks ability to forecast we
decided to extract some information from the existing time-series that could be
useful in the learning process and include it as extra input to the network. More
specifically we concatenated the past 60 normalised daily returns and also found
estimates of the first four moments of the daily returns for the time series. These
estimates of the mean, variance, skewness and kurtosis of daily returns were
calculated using a range of between the last 10 and 120 lagged observations. As an

![Figure 4.4: Schematic representation of the neural network set up for forecasting component 1](image-url)
example, if we used the last 10, 60 and 120 lags we would have a total of 14 inputs.

There is only one hidden layer of say, \( h \) neurons each of which has a hyperbolic tangent sigmoid function as its transfer function. The output layer of the network, comprising a single neuron, is required to give an output response at time \( t \) of what the component’s value will be at \( t + \text{horizon} \). Because of this the network is trained using a set of desired non-contemporaneous input/output vector pairs. That means that the network will be trained and try to uncover the relationship between the inputs at time \( t \) and the outputs at time \( t + \text{horizon} \).

The training process involves the iterative adaptation of the weighted links between neurons to minimize the mean square error between the desired outputs and the actual ones produced by the neural network. A number of techniques may be used to achieve this. In this work after running several tests we decided to go with the conjugate gradient algorithm.

To avoid overfitting the data, we use the training and validation data sets and we stop training when the error of the validation set is minimised. The performance of the network is then assessed on the out-of-sample data of the testing set.

In order to improve learning we initialised the starting weights with the Nguyen-Widrow method [4.30]. This algorithm chooses the initial values so that the values of a layer’s units are distributed evenly over the range of the input space.

The optimal number of hidden nodes to use for each architecture is determined by an exhaustive search between 2 nodes and 80% of the size of the input layer nodes (rules of thumb). This involves training the networks with the number of hidden nodes in this range and then choosing the five architectures providing the lowest validation error. These five architectures are then placed into a committee network structure. The committee provides predictions that are weighted linear combinations of the predictions of each of its members. The weighting can be equal for all the members or can be optimally estimated for each of the members through a direct ordinary least squares method applied to the training data.

4.11 Forecasting evaluation criteria

In order to present the following criteria let \( y \), be the target (output to be predicted) and \( \hat{y} \), its prediction.
4.11.1 Root Mean Square Error and Mean Absolute Error

The Root Mean Square Error and Mean Absolute Error are defined by

\[ RMSE = \sqrt{\frac{1}{T} \sum_{t=1}^{T} (y_t - \tilde{y}_t)^2} \quad \text{and} \quad MAE = \frac{1}{T} \sum_{t=1}^{T} |y_t - \tilde{y}_t| \]

respectively. These measures must be made as small as possible for a successful prediction.

4.11.2 Probability of Sign Error

To define the PSE let a function \( g_t \) be 1 when \( \text{sign}(y_t) \cdot \text{sign}(\tilde{y}_t) < 0 \) and 0 otherwise. Then

\[ PSE = \left( \frac{1}{T} \sum_{t=1}^{T} g_t \right) \cdot 100 \]

The PSE has to be as small as possible.

4.12 Reconstruction and residuals

Let’s remind ourselves of equation (4.4) which is the vector-matrix formulation for the original financial signals \( x \) under the main ICA approach:

\[ x = As \]

The objective was to find good estimates for the underlying independent signals \( s \) and the unknown mixing matrix \( A \). The mapping from \( x \) to \( s \) given in equation (4.4) as \( s = Wx \) defines an optimal unmixing matrix \( W \). It can be seen that \( W \) reverses or inverts the effects of \( A \) and indeed \( W \) could be estimated from the matrix inverse \( W = A^{-1} \), if \( A \) were known. However, as we are ultimately concerned with finding \( W \) when \( A \) is not known, we can not use \( A^{-1} \) to estimate \( W \). For arbitrary values of the unmixing coefficients the unmixing matrix is suboptimal and is denoted \( \tilde{W} \). Because of this, the signals extracted by \( \tilde{W} \) and in our case the reconstructed prices of the financial assets do not necessarily match
those of the original signals and are denoted as $y = \tilde{W}x$. The difference between the original signals and the reconstructed signals is the residual and is denoted as $e = x - s$.

### 4.13 Reconstruction evaluation criterion for selected ICs

Due to the vast choice of available transformation techniques researchers need to use various criteria to evaluate their preferred methods and select the one that suits their system best.

#### 4.13.1 Reconstruction measure

As shown in section 4.12 Although ICA is a reversible operation, the result of the inverse transformation will not correspond to the initial inputs. The mean distance between the initial inputs and the results of the inversion is called the reconstruction measure and it should be as small as possible. If the dimension of the system decreases below its true dimension, then the reconstruction measure will increase; this will validate or invalidate the rough system's dimension estimation. In ideal situations, most of the researchers would relate this measure to the level of noise surrounding the information. The main advantage of this criterion is that it is totally independent of the prediction method used afterwards. Unfortunately, this advantage has a counterpart: there is no guarantee that a chosen technique according to this measure will be optimal when the best prediction outputs are looked for.

### 4.14 Non-linear contributions via the residuals

We have seen in previous sections that a large part of the asset prices is accounted for by a linear combination of a few ICs. Although the say, $\hat{n}$-dimensional residual price signal in the approximation by, for example, $\hat{p}$ ICs is linear in the remaining $\hat{n} - \hat{p}$ ICs, much of this signal may in fact be explainable by non-linear combinations of the already used ICs and does not need additional stochastic factors for a reasonable explanation. It may therefore be possible to consider non-linear combinations of the already chosen $\hat{p}$ ICs to explain part of the residual signal.
In the following sections we will present a novel way in trying to model the residuals with neural networks. Once we forecast the independent components into the future we can then apply this model to the forecasted values of the ICs and find the non-linear part of the residual that should be added to the linearly reconstructed asset returns for a non-linear representation.

4.15 Modelling the residuals

Once more the modelling tool that we use is a neural network. The network's architecture and training procedure is almost identical to that of the neural network described in section 4.10 for the modelling of the independent components. Their only differences are: the contemporaneous treatment of the input / target signals and the nature of the input signals themselves.

This time, as seen in figure 4.5, there are $n$ input vertices, the $n^{th}$ one corresponding to the $n^{th}$ independent component. There is only one hidden layer of say, $\hat{h}$ neurons each of which has a hyperbolic tangent sigmoid function as its transfer function. The output layer of the network, comprising a single neuron, is required to give an output response that is the residual $e_k$ that needs to be added at time $t$ to the linearly reconstructed asset price of asset $k$ at time $t$. Because there are, say $K$ assets, $K > 1$, to be reconstructed, we are constructing $K$ networks that

![Diagram of a neural network for modeling residuals](image)

**Figure 4.5:** Schematic representation of the neural network set up for modeling the residuals of an asset
will output the corresponding residuals needed for the non-linear reconstruction of all the asset prices. The network is trained using a set of desired contemporaneous input/output vector pairs. That means that the network will be trained and try to uncover the relationship between components and residuals measured both at time $t$.

### 4.16 Non-linear forecasting of asset returns

Following the procedure covered in the previous sections, we successfully decompose the original financial asset prices into a set of "meaningful" independent components. From those components we estimate and model the residual between the original and the reconstructed signals.

Once the independent components are forecasted into the future, the expected residuals of the newly linearly reconstructed signals are estimated and added back to the system to produce non-linear forecasting of the asset prices.

### 4.17 Generalisation via cross validation and network committees

In this methodology we face three major issues. The dimensionality reduction of the input space, the modelling/forecasting of independent components and finally the modelling of the residuals. The dimensionality reduction affects the other two and the forecasting of the independent components affects the forecasted residuals. This is a very fragile situation and for that reason we use cross validation and committees of networks in order to control overfitting and achieve generalisation.

#### 4.17.1 Cross validation

Cross validation involves multiple repetitions of the operations described in sections 4.7 to 4.16 under different parameter assumptions. During this procedure performance evaluation measures like the root mean square error, the mean average error and the probability of sign error are evaluated on the validation set and averaged. These averages are then considered in determining the constitution of the network committees and the topologies of the committee members for effective modelling/forecasting of the independent components and the residuals. This process is repeated several times until the reconstruction error verifies the right number of independent components to be used.
4.17.2 Committees of networks

We propose the creation of committees of networks out of the best candidates of many topologically evolved networks. This idea is based on the fact that given a committee of \( i \) neural networks and considering a linear combination

\[
\sum_i \lambda_i f_i \quad \text{with} \quad \sum_i \lambda_i = 1
\]

of the outputs \( f_i \) of each member at a single point, the quadratic error of the ensemble will be lower than the weighted error of each individual network [4.32]. To construct these committees, we change the topology of the initial default network and the initialisation conditions and use the mean square error measure to check each candidate's performance on the same data set. After a sufficient number of networks is examined we select the five best networks to form a committee.

4.18 Conclusions

We showed in previous chapters that most conventional tools used in modelling financial time series suffer from the same limitations concerning their ability to work with high-dimensional data. In the literature, one can find that sophisticated models can "beat the curse of dimensionality". In fact, it is true that many tools like, for example, sophisticated versions of neural networks perform better than many other conventional tools, when the dimension of the data increases. However, problems still remain in high-dimensional spaces, even with improved tools. It is a fact that the intrinsic dimensionality of data is usually lower than the dimension of the data vectors therefore using dimensionality reduction techniques may help to bypass the problems of high dimensions.

In this chapter we presented the theoretical framework of such dimensionality technique, the hybrid model. The proposed model exploits the advantages of linear and non-linear techniques in order to effectively reduce the dimensionality of complex systems and forecast them into the future. In this novel approach we exploit the linear dependencies of the data by using independent component analysis to
decompose the data into "meaningful" linear driving factors and then utilise the non-linear nature of artificial neural networks to discover the non-linearities in the residuals of the linearly reconstructed system. Committees of networks were also introduced as ways to eliminate bad judgements from single networks and boost performance by taking into account results from a group of selected "best" networks.

In the following chapter we will show how this theoretical framework is implemented in practice. We will present experimental results from tests that we have conducted on real data and cover all practical issues in a step-by-step approach.
Chapter 5

Implementation of the Hybrid Model

In this chapter we implement the proposed hybrid model of chapter 4. Experimental results show the main advantages for modelling a signal's ICA residuals and prove the ability of this non-linear methodology to forecast high dimensional systems of well diversified assets.

5.1 Original Data

The data used in this thesis was provided by a major investment bank in the context of the problem of replicating hedge fund returns that we will investigate in the next chapter. As the hybrid method proposed in this chapter will be part of the solution to the replication problem we decided to run our tests on that set of data.

The data consists of daily prices of 42 global indices and investment strategies. This set of data is highly diversified and includes major equity indices, commodities indices, credit indices, FX indices, interest rates linked indices, volatility indices and traded strategies. A list of the assets can be seen in table 4.1.

5.2 Partitioning the data set

The time series of the above data starts on 10 December 2001 and ends on the 30 November 2004 (760 samples). We used a variable length window from the first sample to sample $t$ to forecast the value at $t+1$. The period 1 to $t$ was partitioned in two sets, the training and validation set by using an 80-20 rule (80% of the samples used for training and the remaining for validation).
### Table 5.1: List of global indices and investment strategies

<table>
<thead>
<tr>
<th>#</th>
<th>Type of asset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Strategy</td>
<td>Basket of US stocks selected for their optimal value growth characteristics</td>
</tr>
<tr>
<td>2</td>
<td>Strategy</td>
<td>Basket of EU stocks selected for their optimal value growth characteristics</td>
</tr>
<tr>
<td>3</td>
<td>Strategy</td>
<td>Baskets of global stocks involved in mergers and acquisitions activity</td>
</tr>
<tr>
<td>4</td>
<td>Strategy</td>
<td>Basket of Japanese stocks selected for their optimal value growth characteristics</td>
</tr>
<tr>
<td>5</td>
<td>Strategy</td>
<td>Basket of global stocks including ones that show riskiness of growth</td>
</tr>
<tr>
<td>6</td>
<td>Strategy</td>
<td>Volatility strategy</td>
</tr>
<tr>
<td>7</td>
<td>Strategy</td>
<td>Currency arbitrage strategy – G10 currencies (this has 9 cross rates included)</td>
</tr>
<tr>
<td>8</td>
<td>Strategy</td>
<td>Momentum strategy basket including interest rates, commodities and FX</td>
</tr>
<tr>
<td>9</td>
<td>Strategy</td>
<td>Currency arbitrage strategy</td>
</tr>
<tr>
<td>10</td>
<td>Index</td>
<td>FTSE 100 Index</td>
</tr>
<tr>
<td>11</td>
<td>Index</td>
<td>FTSE 250 equity index</td>
</tr>
<tr>
<td>12</td>
<td>Index</td>
<td>FTSE 100 equity Index – FTSE 250 equity index</td>
</tr>
<tr>
<td>13</td>
<td>Index</td>
<td>Eurostoxx 50 equity index</td>
</tr>
<tr>
<td>14</td>
<td>Index</td>
<td>S&amp;P500 equity index</td>
</tr>
<tr>
<td>15</td>
<td>Index</td>
<td>S&amp;P500 Index – Russell 2000 Index</td>
</tr>
<tr>
<td>16</td>
<td>Index</td>
<td>Nikkel 225 index</td>
</tr>
<tr>
<td>17</td>
<td>Index</td>
<td>Basket of stocks from the EU</td>
</tr>
<tr>
<td>18</td>
<td>Index</td>
<td>Basket of stocks from Japan</td>
</tr>
<tr>
<td>19</td>
<td>Index</td>
<td>Developing countries stocks index</td>
</tr>
<tr>
<td>20</td>
<td>Index</td>
<td>Emerging markets stocks index</td>
</tr>
<tr>
<td>21</td>
<td>Index</td>
<td>US Dollar Index</td>
</tr>
<tr>
<td>22</td>
<td>Index</td>
<td>Lehman’s index</td>
</tr>
<tr>
<td>23</td>
<td>Index</td>
<td>Large cap European equity index</td>
</tr>
<tr>
<td>24</td>
<td>Index</td>
<td>Large cap European equity index – Small cap equity index</td>
</tr>
<tr>
<td>25</td>
<td>Index</td>
<td>Large value stocks index – Small value stock index</td>
</tr>
<tr>
<td>26</td>
<td>Index</td>
<td>Large value stocks index - Large growth stocks index</td>
</tr>
<tr>
<td>27</td>
<td>Index</td>
<td>Mid value stocks index</td>
</tr>
<tr>
<td>28</td>
<td>Index</td>
<td>Small value stocks index – Small growth stocks index</td>
</tr>
<tr>
<td>29</td>
<td>Index</td>
<td>Large growth stocks index – Small growth stocks index</td>
</tr>
<tr>
<td>30</td>
<td>Index</td>
<td>Large growth stocks index – Lehman’s aggregate</td>
</tr>
<tr>
<td>31</td>
<td>Index</td>
<td>Mid growth stocks index</td>
</tr>
<tr>
<td>32</td>
<td>Index</td>
<td>US 10 year swap rates</td>
</tr>
<tr>
<td>33</td>
<td>Index</td>
<td>US 3 year swap rates</td>
</tr>
<tr>
<td>34</td>
<td>Index</td>
<td>US 10 year swap rates – US 3 year swap rates</td>
</tr>
<tr>
<td>35</td>
<td>Index</td>
<td>Global high yield index</td>
</tr>
<tr>
<td>36</td>
<td>Index</td>
<td>High yield US credit Index</td>
</tr>
<tr>
<td>37</td>
<td>Index</td>
<td>High yield emerging markets credit Index</td>
</tr>
<tr>
<td>38</td>
<td>Index</td>
<td>Investment grade credit Index</td>
</tr>
<tr>
<td>39</td>
<td>Index</td>
<td>European 5Y credit Index</td>
</tr>
<tr>
<td>40</td>
<td>Index</td>
<td>European over 5 year credit Index</td>
</tr>
<tr>
<td>41</td>
<td>Index</td>
<td>Goldman Sachs Commodities Index</td>
</tr>
<tr>
<td>42</td>
<td>Index</td>
<td>Crude Oil-Brent Cur, Month FOB US/BBL</td>
</tr>
</tbody>
</table>
5.3 First dimensionality reduction using PCA

We carried out PCA on these 42 variables and we noted that 95% of the original variance is kept with the first 25 principal components (figure 5.1). That means that 17 variables can be removed without considerable loss of information.

![Figure 5.1: Principal Component Analysis performed on the data set](image)

5.4 Second dimensionality reduction using ICA

According to equation (4.9) the number of independent components that should capture most of the information in the newly constructed 25-dimensional system should not be more than 18. We used the Minimum Description Length Principle in equation (4.10) and calculated the MDL for different number of factors ranging from 0 to 18. Figure 5.2 below shows that the smallest minimum description length was 0.4351 and it was observed for a seven factor model.

![Figure 5.2: Minimum Description Length for different numbers of m factors](image)
5.5 Ordering of the independent components

The important task was now to find which 7 components should be chosen as the most representative ones. We used the $L_\infty$ norm to sort the independent components according to which one causes the maximum price change in the stock. In figure 5.3 below we give a graphical representation of the 5 best ones.

![Figure 5.3: The most important ICs according to the $L_\infty$ norm](image)

5.6 Smoothing the components

We smoothed the components in MATLAB using a Savitsky-Golay filter of order 5 on a window of 101 samples. The results are shown in the graphs below.

![Figure 5.4: Smoothed ICs by Savitsky-golay filter. Smoothed ICs in blue, original ICs in red](image)
5.7 Step-by-step procedure

The experimental procedure for modelling and forecasting independent components and residuals is quite complicated as the cross validation on such system involves multiple loops of optimisation at different levels. In this section we present a step-by-step explanation to assist the reader in following the procedure.

Part A: Forecasting future values of independent components

Step 1. Consider the independent components of section 5.6. In this case the number of components was 7 as was suggested by the MDL principle in section 5.4.

Step 2. Model each independent component by using multiple initialisations on the default neural network (3 hidden neurons) and find multiple point estimates for each time instance in the testing set.

Step 3. Repeat step 2 for neural networks of different number of neurons in the hidden layer.

Step 4. For each independent component choose the 5 network topologies that give the smallest forecasting error and group them in a committee.

Step 5. Repeat steps 2 to 4 for different numbers of independent components

Part B: Verifying the number of independent components to be used

Step 7. Use the results of Phase A and the reconstruction measure, to find the number of independent components that best represent the original system.

Part C: Forecasting future values of residuals

Step 8. Find as many independent components as was suggested in step 7

Step 9. Find the residuals left by the reconstruction using those components

Step 10. Model each independent component by using multiple initialisations on the default neural network (3 hidden neurons) and find multiple point estimates for each time instance in the testing set.

Step 11. Repeat step 10 for neural networks of different number of neurons in the hidden layer.

Step 12. For each independent component choose the 5 network topologies that give the smallest forecasting error and group them in a committee.
5.8 Forecasting the components

The neural network that we use to forecast the components is a 3-layer network as shown in section 4.10.2. The neurons in the hidden layer have a hyperbolic tangent sigmoid transfer function whereas the transfer function of the neuron in the output layer is linear. The hidden layer has initially 3 neurons. We use 1000 different initialisations using the Nguyen-Widrow method and train 7 networks for as many independent components. We train and validate the networks on the first 658 samples of the information set (up to 12 July 2004). For the estimates of the first four moments of the daily returns we use the last 10, 30, 60, 90 and 120 lagged observations. If \( t \) corresponds to the last sample of the learning set (658th sample) the objective is to forecast the return of each independent component at time \( t+1 \) (659th sample). In the case where forecast beyond \( t+1 \) is needed this procedure has to be repeated for \( t+2, \ldots, t_H \) where \( t_H \) denotes the end of the testing set. In such case the learning data set has to be extended to cover all samples up and including \( t \).

Results and discussion

The actual returns for 13 July '04 (659th sample) for the selected components 1 to 7 are shown in table 5.2 together with the mean and standard deviation of the results\(^1\) from each neural network. Overall, the means have been forecasted well and the multi-initialisation procedure has given low standard deviation across all networks. Moreover, all networks showed percentages of sign error at around 20%. Because of the eventual reconstruction of asset returns from the independent components it is vital that errors and volatility are kept to a minimum. Big errors at this stage will impact the forecasting of the residuals and these in turn will affect the quality of the reconstruction.

<table>
<thead>
<tr>
<th></th>
<th>Actual value</th>
<th>Mean</th>
<th>Standard Deviation</th>
<th>PSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC1</td>
<td>-0.0036</td>
<td>-0.389</td>
<td>0.0027</td>
<td>5.8%</td>
</tr>
<tr>
<td>IC2</td>
<td>-0.0240</td>
<td>-1.145</td>
<td>0.0041</td>
<td>8.4%</td>
</tr>
<tr>
<td>IC3</td>
<td>-0.0260</td>
<td>0.612</td>
<td>0.0028</td>
<td>5.4%</td>
</tr>
<tr>
<td>IC4</td>
<td>-0.0530</td>
<td>-0.972</td>
<td>0.0046</td>
<td>6.8%</td>
</tr>
<tr>
<td>IC5</td>
<td>-0.3910</td>
<td>0.031</td>
<td>0.0037</td>
<td>5.9%</td>
</tr>
<tr>
<td>IC6</td>
<td>0.0739</td>
<td>0.719</td>
<td>0.0019</td>
<td>3.8%</td>
</tr>
<tr>
<td>IC7</td>
<td>0.0890</td>
<td>1.124</td>
<td>0.0021</td>
<td>2.8%</td>
</tr>
</tbody>
</table>

Table 5.2: Summary statistics of the 7 default neural networks when forecasting the 659th sample

\(^1\) Each neural network has been initialised 1000 times, thus the distribution of results
We performed a back test on the whole testing set from 13 July '04 to 30 November '07 (102 samples) and the average results for this period in terms of RMSE and MAE are shown in table 5.3.

<table>
<thead>
<tr>
<th></th>
<th>Average RMSE</th>
<th>Average MAE</th>
<th>Average PSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC1</td>
<td>0.0439</td>
<td>0.0318</td>
<td>21%</td>
</tr>
<tr>
<td>IC2</td>
<td>0.0394</td>
<td>0.0297</td>
<td>18%</td>
</tr>
<tr>
<td>IC3</td>
<td>0.0409</td>
<td>0.0315</td>
<td>23%</td>
</tr>
<tr>
<td>IC4</td>
<td>0.0346</td>
<td>0.0269</td>
<td>20%</td>
</tr>
<tr>
<td>IC5</td>
<td>0.0423</td>
<td>0.0306</td>
<td>19%</td>
</tr>
<tr>
<td>IC6</td>
<td>0.0410</td>
<td>0.0293</td>
<td>17%</td>
</tr>
<tr>
<td>IC7</td>
<td>0.0415</td>
<td>0.0307</td>
<td>23%</td>
</tr>
</tbody>
</table>

Table 5.3: Descriptive statistics for each of the 7 default neural networks in terms of RMSE, MAE and PSE for a back test conducted on the testing period (13/07/04 – 30/11/07)

The above procedure was conducted using the default neural networks with 3 neurons in the hidden layer. Many researchers and practitioners in the area of neural networks claim that the number of neurons in the hidden layer should be approximately equal to 75% of the dimensions of the input space. In our case the inputs to the network are: 5 sets of lagged observations that produce the four moments of the time series for different windows, the past 60 normalised daily returns and the values of the component themselves. This makes the dimension of the input space equal to 22. Following the empirical rule above on the number of neurons in a network’s hidden layer, we incrementally changed the topology of the 7 default networks from having 2 to 15 neurons in the hidden layer.

Each one of the new networks was retrained following the same procedure and tested on the same testing set as before. All other parameters i.e. transfer functions, partition of data into different sets etc were kept the same. Moreover, the seven seeds to the random number generator of the Nguyen-Widrow network initialisation methodology were kept consistent with the ones that were used previously in order to make the results comparable.

Table 5.4 shows the average RMSE, MAE and average PSE error results for each of the resulting 91 networks. On average, networks with 3 to 7 hidden neurons performed better in terms of RMSE and MAE. Bigger networks suffered from overfitting and larger errors, but in terms of PSE they are were not inferior to the smaller networks.
Table 5.4: Descriptive statistics of the 91 topologically modified networks in terms of average RMSE, MAE and PSE for a back test conducted on the testing period (13/07/44 – 30/11/07)

We ranked the networks from best to worst in terms of the three error criteria in order to decide which ones to combine to form committees. In table 4.4 the five best networks for each component are highlighted. In the bottom table (PSE) there are no clear “winners” and for that reason we use the other 2 tables to determine which
networks to include in a committee. The RMSE and MAE tables show consistent answers. We used those tables and decided to form the committees based on the 5 best networks. In the very few cases where there were small inconsistencies (IC1, IC2 and IC7) we showed preference to the networks with the smallest dimension.

We applied equal weights to the outputs of the committee members and performed a back test on the whole testing set. The descriptive statistics are shown in table 5.5. On average the errors were smaller than the errors of the individual networks. The results prove that well diversified committees are less risky than individual networks.

<table>
<thead>
<tr>
<th>NN Committee</th>
<th>Average RMSE</th>
<th>Average MAE</th>
<th>Average PSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC1</td>
<td>0.0211</td>
<td>0.0155</td>
<td>14.0%</td>
</tr>
<tr>
<td>IC2</td>
<td>0.0221</td>
<td>0.0167</td>
<td>15.0%</td>
</tr>
<tr>
<td>IC3</td>
<td>0.0202</td>
<td>0.0142</td>
<td>16.0%</td>
</tr>
<tr>
<td>IC4</td>
<td>0.0234</td>
<td>0.0164</td>
<td>15.0%</td>
</tr>
<tr>
<td>IC5</td>
<td>0.0222</td>
<td>0.0176</td>
<td>14.0%</td>
</tr>
<tr>
<td>IC6</td>
<td>0.0206</td>
<td>0.0134</td>
<td>12.0%</td>
</tr>
<tr>
<td>IC7</td>
<td>0.019</td>
<td>0.0129</td>
<td>9.0%</td>
</tr>
</tbody>
</table>

Table 5.5: Descriptive statistics for each of the 7 network committees in terms of RMSE, MAE and PSE for a back test conducted on the testing period (13/07/04 – 30/11/07)

We can now claim that we managed to construct committees of networks that model and forecast the behaviour of independent components in a consistent manner.

5.9 Modelling the residuals

To model the residuals we had to construct a very similar network to the one in the previous section. The main differences were the contemporaneous mapping between inputs (components) and outputs (residuals) and the construction of 42 network committees instead of seven. The number of committees is such because there are 42 assets in the input universe each of which was modelled separately.

There are two issues that needed to be addressed at this point. The first one was whether the target should be the residuals from the reconstruction of the original components or those from the reconstruction of the smoothed components. The second issue was whether the inputs to the network should be the original or the smoothed components.
Regarding the first issue, we decided to choose the residuals from the original independent components. The main reason behind this choice was that the mixing matrix was found by ICA before we smoothed the components. Therefore, if we attempted to reconstruct the prices from the corresponding un-mixing matrix and smoothed components we would lose information which we would then have to recover through the non-linear rather than the easier and already defined linear path.

Regarding the second issue, we decided to use smoothed independent components as the input to the network. There are mainly two reasons for this. The first one is that smoothing was achieved by a Savitsky-Golay filter which is a one-to-one filtering procedure introducing no lags etc. Therefore, there would be no mismatches in the contemporaneous mapping of inputs and outputs. Secondly, we forecast smoothed independent components so why not take advantage of this and use them directly as inputs to the system. We then look for the neural network that would effectively map smoothed independent components to residuals that were found from reconstruction using non-smoothed components.

Table 5.6 summarises the experimental results obtained. Once more very similarly to the forecasting of independent components the committees gave better results than the individual neural networks. There was again a tendency for picking networks with lower dimensions which this time is also verified by the fact that we only have 7 inputs rather than 22 that were there when forecasting independent component values.

The reason behind modelling each of the assets separately is because in such a big system there will always be assets requiring simpler networks structures than others. Thus, although it is very time consuming to process them separately the generalisation properties of the resulting networks can be much better.

## 5.10 Linear and non-linear reconstruction

The experimental results in this section are based on either the whole universe of assets (mainly descriptive statistics) or on two representatives (where there is more detail included). These representatives are the first and fifth assets in our list of time series data shown in table 5.1. Asset number 1 is a US index of stocks with optimal value-growth characteristics and asset number 2 a risky index with global stocks. From this point forward we will refer to asset 1 as asset USB and asset 2 as asset GLB. In this section we will first experiment with asset USB and discuss the
Table 5.6: Forecasting the residuals: Descriptive statistics (based on 7 independent components) for each of the 42 network committees in terms of RMSE, MAE and PSE for a back test conducted on the whole of the testing period (13/07/04 – 30/11/07)

<table>
<thead>
<tr>
<th>Asset No</th>
<th>Average RMSE</th>
<th>Average MAE</th>
<th>Average PSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0361</td>
<td>0.0200</td>
<td>20.0%</td>
</tr>
<tr>
<td>2</td>
<td>0.0350</td>
<td>0.0196</td>
<td>23.0%</td>
</tr>
<tr>
<td>3</td>
<td>0.0409</td>
<td>0.0199</td>
<td>24.0%</td>
</tr>
<tr>
<td>4</td>
<td>0.0349</td>
<td>0.0196</td>
<td>22.0%</td>
</tr>
<tr>
<td>5</td>
<td>0.0411</td>
<td>0.0196</td>
<td>25.0%</td>
</tr>
<tr>
<td>6</td>
<td>0.0392</td>
<td>0.0196</td>
<td>23.0%</td>
</tr>
<tr>
<td>7</td>
<td>0.0368</td>
<td>0.0196</td>
<td>22.0%</td>
</tr>
<tr>
<td>8</td>
<td>0.0334</td>
<td>0.0200</td>
<td>20.0%</td>
</tr>
<tr>
<td>9</td>
<td>0.0329</td>
<td>0.0196</td>
<td>24.0%</td>
</tr>
<tr>
<td>10</td>
<td>0.0366</td>
<td>0.0197</td>
<td>24.0%</td>
</tr>
<tr>
<td>11</td>
<td>0.0449</td>
<td>0.0200</td>
<td>22.0%</td>
</tr>
<tr>
<td>12</td>
<td>0.0419</td>
<td>0.0199</td>
<td>20.0%</td>
</tr>
<tr>
<td>13</td>
<td>0.0405</td>
<td>0.0196</td>
<td>21.0%</td>
</tr>
<tr>
<td>14</td>
<td>0.0421</td>
<td>0.0199</td>
<td>21.0%</td>
</tr>
<tr>
<td>15</td>
<td>0.0415</td>
<td>0.0198</td>
<td>22.0%</td>
</tr>
<tr>
<td>16</td>
<td>0.0397</td>
<td>0.0199</td>
<td>24.0%</td>
</tr>
<tr>
<td>17</td>
<td>0.0433</td>
<td>0.0196</td>
<td>24.0%</td>
</tr>
<tr>
<td>18</td>
<td>0.0391</td>
<td>0.0196</td>
<td>24.0%</td>
</tr>
<tr>
<td>19</td>
<td>0.0468</td>
<td>0.0196</td>
<td>20.0%</td>
</tr>
<tr>
<td>20</td>
<td>0.0560</td>
<td>0.0197</td>
<td>20.0%</td>
</tr>
<tr>
<td>21</td>
<td>0.0369</td>
<td>0.0196</td>
<td>22.0%</td>
</tr>
<tr>
<td>22</td>
<td>0.0449</td>
<td>0.0196</td>
<td>23.0%</td>
</tr>
<tr>
<td>23</td>
<td>0.0397</td>
<td>0.0197</td>
<td>25.0%</td>
</tr>
<tr>
<td>24</td>
<td>0.0353</td>
<td>0.0196</td>
<td>20.0%</td>
</tr>
<tr>
<td>25</td>
<td>0.0439</td>
<td>0.0197</td>
<td>24.0%</td>
</tr>
<tr>
<td>26</td>
<td>0.0389</td>
<td>0.0196</td>
<td>24.0%</td>
</tr>
<tr>
<td>27</td>
<td>0.0375</td>
<td>0.0197</td>
<td>20.0%</td>
</tr>
<tr>
<td>28</td>
<td>0.0531</td>
<td>0.0197</td>
<td>26.0%</td>
</tr>
<tr>
<td>29</td>
<td>0.0423</td>
<td>0.0199</td>
<td>22.0%</td>
</tr>
<tr>
<td>30</td>
<td>0.0420</td>
<td>0.0198</td>
<td>25.0%</td>
</tr>
<tr>
<td>31</td>
<td>0.0426</td>
<td>0.0199</td>
<td>20.0%</td>
</tr>
<tr>
<td>32</td>
<td>0.0373</td>
<td>0.0196</td>
<td>27.0%</td>
</tr>
<tr>
<td>33</td>
<td>0.0385</td>
<td>0.0196</td>
<td>20.0%</td>
</tr>
<tr>
<td>34</td>
<td>0.0386</td>
<td>0.0199</td>
<td>22.0%</td>
</tr>
<tr>
<td>35</td>
<td>0.0419</td>
<td>0.0197</td>
<td>22.0%</td>
</tr>
<tr>
<td>36</td>
<td>0.0414</td>
<td>0.0198</td>
<td>22.0%</td>
</tr>
<tr>
<td>37</td>
<td>0.0431</td>
<td>0.0197</td>
<td>24.0%</td>
</tr>
<tr>
<td>38</td>
<td>0.0416</td>
<td>0.0196</td>
<td>25.0%</td>
</tr>
<tr>
<td>39</td>
<td>0.0461</td>
<td>0.0200</td>
<td>26.0%</td>
</tr>
<tr>
<td>40</td>
<td>0.0417</td>
<td>0.0196</td>
<td>24.0%</td>
</tr>
<tr>
<td>41</td>
<td>0.0417</td>
<td>0.0199</td>
<td>19.0%</td>
</tr>
<tr>
<td>42</td>
<td>0.0366</td>
<td>0.0196</td>
<td>22.0%</td>
</tr>
</tbody>
</table>
results during various reconstruction stages. Results on the risky GLB will follow.

5.10.1 Linear reconstruction using existing components on the testing set

We tested the system’s performance under two scenarios. The first one involved training the system with residuals from the linear reconstruction of the 7 original independent components and the second one involved training with residuals from the linear reconstruction of the smoothed independent components. In this experiment we did not do any forecasts. We simply used the pre-computed independent component values for all the testing samples. Because of this, in this experiment there are no forecasting effects. In figures 5.5 and 5.6 we show graphical representations of the original and reconstructed signals for both cases respectively. The results reported have been produced on the 102 testing set samples.

![Figure 5.5: Original & reconstructed signals for USB. The reconstructed signal has been derived by non-smoothed components](image)

![Figure 5.6: Original & reconstructed signals for USB. The reconstructed signal has been derived by smoothed components](image)
In the first case (non-smoothed ICs) the RMSE was 3.77 whereas in the other case (smoothed ICs) it was 8.71. The MAE was 3.22 and 7.42 respectively. The main reason behind this difference is that the mixing matrix found with ICA has been optimised for the non-smoothed components and therefore independent components perform better in its inverse operation. We did similar tests for all the 42 assets in the portfolio and the average description statistics of the results are summarized in table 5.6. The results were very encouraging as despite the bigger errors smooth components prove to be good approximators.

<table>
<thead>
<tr>
<th></th>
<th>Average RMSE</th>
<th>Average MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unsmoothed</td>
<td>6.200</td>
<td>4.800</td>
</tr>
<tr>
<td>Smoothed</td>
<td>7.600</td>
<td>6.300</td>
</tr>
</tbody>
</table>

Table 5.6: Average descriptive statistics for the reconstruction of all 42 time series of the data set in terms of RMSE and MAE. This test uses component values that have been obtained by ICA only (no forecasting). Results are shown for the testing period (13/07/04 – 30/11/07)

5.10.2 Linear reconstruction using forecasted components

The following figures (5.7-5.8) show results on the same data set but this time including forecasted smoothed and non-smoothed components. The results are not as good as before and this is due to the forecasting error. The non-smoothed version has a RMSE of 14.26 whereas for the smoothed case it is 10.92. The MAE is 12.18 and 9.31 respectively. This time the results on the smoothed version are better than the non-smoothed one which proves that using neural networks to forecast smooth components is more robust. The results for the remaining time series showed similar behaviour as it is reflected in the average description statistics table 5.7.

Figure 5.7: Original & reconstructed signals for USB. The reconstructed signal has been derived by forecasted non-smoothed components.
5.10.3 Non-linear reconstruction using forecasted components & residuals

The following figures (5.9-5.11) show results on the same data set, including forecasted smoothed and non-smoothed components and also including the effect of adding residuals. The residuals in this experiment were not forecasted for the testing set. Instead in the case of the non-smoothed components they were derived from the linear reconstruction of the pre-calculated non-smoothed components on the testing set (test 1). In the case of the smoothed components the residuals were derived from reconstruction using the pre-calculated smoothed (test2) and non-smoothed (test3) components on the testing set.

In test 1 the results are slightly better than those obtained in the previous section (no residuals). The RMSE was at 9.73 and the MAE at 8.3. In tests 2 and 3 the RMSE were at 2.17 and and 6.53 and the MAE at 1.71 and 5.37 respectively.

We notice that the best combination is that of forecasted smoothed components and residuals from the reconstruction of non-smoothed components. This is because this combination avoids the difficult forecasting of non-smoothed components and takes
advantage of the all the information hidden in the "true" residuals. The results for the remaining time series showed similar behaviour as it is reflected in the average description statistics table 5.8.

![Figure 5.9: Original & reconstructed signals for USB. The reconstructed signal has been derived by non-smoothed components and residuals from non-smoothed reconstruction](image1)

![Figure 5.10: Original & reconstructed signals for USB. The reconstructed signal has been derived by smoothed components and residuals from non-smoothed reconstruction](image2)

![Figure 5.11: Original & reconstructed signals for USB. The reconstructed signal has been derived by smoothed components and residuals from smoothed reconstruction](image3)
Table 5.8: Average descriptive statistics for the reconstruction of all 42 time series of the data set in terms of RMSE and MAE. This test uses forecasted components obtained by the network committee. Results are shown for the testing period (13/07/04 – 30/11/07)

5.10.4 Non-linear reconstruction using both forecasted components & residuals

Figures 5.12-5.13 show results on the same data set, including forecasted smoothed and non-smoothed components and also including the effect of forecasted residuals. In the previous section experimental results proved that non-linear reconstruction with the residuals from the non-smoothed version of independent components is more robust compared to that from the smoothed version of components. That is why in this last experiment for asset USB we have only used the better version.

Figure 5.12: Original & reconstructed signals for USB. The reconstructed signal has been derived by non-smoothed components and forecasted residuals

Figure 5.13: Original & reconstructed signals for USB. The reconstructed signal has been derived by smoothed components and forecasted residuals
The non-smoothed version results prove that its performance still suffers from the bad forecasting of the non-smoothed independent components. The RMSE was 10.47 and the MAE 8.94 which is equivalent to the smoothed version’s performance without adding residuals. Moreover the smoothed version’s performance at RMSE 3.26 and MAE 2.71 is still better than the non-smoothed version’s corresponding performance of 3.77 for 3.22 from direct reconstruction using non-forecasted values.

When applied to all the 42 assets of the input universe the smoothed version outperformed the non-smoothed version in almost 79% of the cases (33 out of 42).

<table>
<thead>
<tr>
<th></th>
<th>Average RMSE</th>
<th>Average MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unsmoothed</td>
<td>11.0724</td>
<td>8.0133</td>
</tr>
<tr>
<td>Smoothed</td>
<td>5.0300</td>
<td>4.1589</td>
</tr>
</tbody>
</table>

**Table 5.9:** Average descriptive statistics for the reconstruction of all 42 time series of the data set in terms of RMSE and MAE. This test uses forecasted components and residuals obtained by the network committee. Results are shown for the testing period (13/07/04 – 30/11/07)

5.10.5 Forecasting a highly volatile asset

We repeated the above experiments with GLB in order to investigate the model’s reaction to more volatile assets. The results were obtained by following the exact same procedure that was used for USB (figures 5.14 - 5.17).

The model’s performance in forecasting the price of GLB was not as good as it was in the USB’s case. This is mainly due to the fact that this asset was quite volatile and the smoothing of the components must have filtered important information. The encouraging result though is that the model still outperformed the linear IC reconstruction.

**Figure 5.14:** Original & reconstructed signals for GLB. The reconstructed signal has been derived by a) non-smoothed components b) smoothed components
Figure 5.15: Original & reconstructed signals for GLB. The reconstructed signal has been derived by a) forecasted non-smoothed components and b) forecasted smoothed components.

Figure 5.16: Original & reconstructed signals for GLB. The reconstructed signal has been derived by a) non-smoothed components and b) smoothed components and residuals from non-smoothed reconstruction (for both)

Figure 5.17: Original & reconstructed signals for GLB. The reconstructed signal has been derived by a) forecasted non-smoothed components and residuals and b) forecasted smoothed components and residuals
5.11 Verification of the number of independent components

All the experiments that we conducted so far assumed 7 independent components. Cross validation involves multiple repetitions of the whole procedure for different assumptions on the number of independent components used. The optimal number of components to consider is the one that minimises the average error.

In this section we present results for 5-12 independent components. In table 5.10 we show how the description statistics change for different numbers of independent components assuming exactly the same problem.

<table>
<thead>
<tr>
<th>8 of ICs</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average RMSE</td>
<td>0.1962</td>
<td>0.8957</td>
<td>4.6699</td>
<td>3.4779</td>
<td>6.3751</td>
<td>5.7355</td>
<td>6.9787</td>
<td>8.7607</td>
</tr>
<tr>
<td>Average MAE</td>
<td>6.2281</td>
<td>6.6259</td>
<td>3.8996</td>
<td>3.7446</td>
<td>5.4499</td>
<td>4.6391</td>
<td>5.3292</td>
<td>5.5907</td>
</tr>
</tbody>
</table>

Table 5.10: Average descriptive statistics for different number of independent components for the reconstruction of all 42 time series of the data set. Results are shown for the testing period (13/07/04 - 30/11/07)

From the table above we notice that there is no clear answer to which number of independent components is best to use. Our initial choice of 7 components proved to be the second best but it was very close to the component with the smallest error. In general there were many results in the region 7 to 10 that were very close. In such situations a good choice would be to go for the smallest number as it is now clear that forecasting of the components accounts for most of the error.

5.12 The conventional neural network approach

We conducted a final analysis where we compared the results of the above experiments with those obtained using a conventional neural network directly on the initial asset prices. The results of using a neural network to directly forecast the behaviour of assets USB and GLB from their initial prices (and without using independent components and/or their residuals) are shown in figure 5.18.

We notice that the RMSE and the MAE for USB are 29.87 and 25.26 respectively whereas for GLB they are 35.91 and 28.43. These values are much higher than the worst values obtained by our hybrid methodology (USB→ RMSE: 13.43, MAE: 11.28 / GLB→ RMSE: 17.71, MAE: 12.87) which indicate that the
usage of both independent components and the residuals adds significant value to the solution of the problem.

![Graph](image)

**Figure 5.18:** Original & reconstructed signals for a) USB and b) GLB. The reconstructed signal has been derived by a conventional neural network on the original prices

The conventional neural network method was used to forecast the behaviour of each one of the 42 assets of the input universe. All the results were consistently worse than those obtained using our proposed hybrid method. This can also be observed by looking at the average RMSE and MAE results of the two approaches in tables 5.11 and 5.9.

<table>
<thead>
<tr>
<th>Average RMSE</th>
<th>Average MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>25.3671</td>
<td>19.5909</td>
</tr>
</tbody>
</table>

**Table 5.11:** Average descriptive statistics for the reconstruction of all 42 time series of the data set in terms of RMSE and MAE. This test uses a conventional neural network applied directly to the asset prices. Results are shown for the testing period (13/07/04 – 30/11/07)

### 5.13 Conclusions

In this chapter we demonstrated a step-by-step implementation of the hybrid model that was introduced in chapter 4. We run exhaustive experimental tests under cross validation on a well diversified universe of global financial assets. Results from the hybrid model were consistently better that those achieved by direct ICA reconstruction without residuals (conventional way). These good forecasting results were also obtained for extra volatile signals. Back tests on 42 assets revealed that the algorithm was indeed capable of forecasting as it successfully reconstructed (within a degree of error) 33 out of 42 of the original asset time series. At the end of the cross validation a very important verification of the number of optimal components was achieved. The main disadvantage behind this methodology was the long computational time.
Chapter 6

Replication of Hedge Fund Returns

6.1 Introduction

In the last decade, interest in Hedge Funds from both academics and investors has grown dramatically. Because hedge funds are typically organized as private investment vehicles for wealthy individuals and institutional investors, they do not have to disclose their activities publicly. Hence, little is known about the risk in hedge fund strategies. Academics are intrigued by the unconventional performance characteristics in hedge funds, while investors are attracted by the option-like returns and the low correlation with different asset classes. However, high performance fees charged by the managers, lack of liquidity, lack of transparency and fear for style risk have raised the question whether it is possible to generate hedge-fund-like returns with less effort using more liquid assets such as stock indexes, bonds, currency, commodity and interest rate futures.

The idea is to employ statistical methods and factor models to replicate the hedge fund mean returns. Thus, clones will capture on average, the main performance trends of the hedge funds and provide to investors similar return characteristics with a much lower cost. But in which way can hedge funds be replicated? There exist two main approaches. One possible approach aims to replicate hedge fund returns using several observed factors in order to capture the main risk exposures of the hedge fund returns. For example Lo and Hasanhovbic [6.1] adopt a dynamic linear approach of return replication using stock indexes, bonds, and commodity indexes. They used a 24 months rolling window which
counts for the variability of hedge fund risk exposures. However, the factor selection is *ad hoc* and is not based on any theoretical framework.

Kat and Palaro [6.2] criticize factor-based approaches explaining that since we generally use portfolios of traditional assets indices, factor models for hedge funds explain only a small proportion of their returns. These authors propose another approach which relies on statistical tools in order to simply replicate the statistical properties of Hedge Fund returns without aiming to replicate month-to-month returns. For a further discussion on that point refer to [6.2].

Our proposal to this problem is the use of dynamic weightings of instruments and strategies based on non-linear optimisation and learning techniques to determine the replicating portfolio at each time by minimising the sample mean square error of the replicating portfolio’s returns versus the observed fund class return time series. We introduce explicit handling of turnover and risk control measures and rigorous out-of-sample back testing. The overall objective is the best out-of-sample effectiveness by balancing the absolute return, tracking error, total risk and turnover.

### 6.2 Overview of replication models

A substantial part of the research effort in finance is directed toward improving our understanding of how investors value risky cash flows. Several capital asset pricing models have been suggested in the literature that attempt to assess the risk exposures that drive the covariance of asset returns. We can distinguish two main approaches. The first one is based on the arbitrage pricing theory framework developed by Ross [6.3]. Statistical methods, such as factor analysis or principal components analysis are used to estimate latent factors from the covariance structure of the data. The theory does not determine the number, or the nature of the latent factors. The most important drawback of this approach is the lack of economical interpretation of asset’s risk exposures.

A more popular approach is to rely on intuition and theory as guides to select a set of observed variables as proxies of the unobserved theoretical factors. For example in the CAPM analysis the equal-weighted and value weighted market returns are used as proxies of the unobserved theoretical market factor. Although Fama and Macbeth [6.4] validate the model empirically using the US stock market data, more recent studies highlight the lack of empirical support of the CAPM. For
example, in their widely cited study, Fama and French [6.5] present evidence of the inability of the CAPM to explain the stock returns. They pointed out the robustness of the size and the book-tomarket effect. The empirical lack of a one-factor approach motivates the development of multifactor asset pricing models. Chen et al. [6.6] found that the factors in APT are related to macroeconomic variables while Jagannathan and Wang [6.7] use the return on labor income to improve the proxy of market portfolio and the credit spread as a proxy for the systematic risk instability. Perhaps the most well-known of observable risk factors are the three discussed in [6.8]: the market excess return (MKT), small minus big factor (SMB), and high minus low factor (HML).

Both, statistical and observed factors are widely used to assess the risk exposures of equity returns. In fact, several academic studies bring empirical evidence on the ability of multifactor models to explain the cross-section of stock returns. Moreover the need to use multifactor models in assessing the risk of stock returns is also demonstrated by the commercial success of firms which provide beta estimates for risk management and valuation purposes, using elaborate time series models.

The growth of hedge fund industry this last two decades has reoriented the asset pricing efforts toward alternative returns offered by hedge funds. In fact, investors looking for alternative returns must be concerned by how do hedge funds managers deliver return characteristics that are different from the returns of the very asset classes they are trading. A significant gap has emerged between the expectations of institutional investors and the hedge fund managers. Hedge fund managers rarely provide position-level transparency.

Furthermore, they routinely impose lock-ups of one to three years and charge very important performance fees. These points raise the natural question of whether it is possible to obtain hedge-fund-like returns using liquid asset classes without investing in hedge funds. Another issue of hedge fund return replication is to construct a plausible benchmark providing a better evaluation of the alpha creation of a particular hedge fund.

An extensive literature has documented that hedge fund returns differ from the returns of the traditional asset classes. Mutual fund returns have high and positive correlation with asset class returns which suggest that they behave as a “buy and hold” strategy. Hedge fund returns seem to have low and sometimes negative
correlation with asset class returns, which suggest that they behave as if deploying a dynamic strategy including short sells and leverage.

For example, Fung and Hsieh [6.9] pointed out the option-like characteristics for the returns of several hedge fund strategies, such as the trend followers and the merger arbitrage. Note that the option-like feature is a possible explanation of the observed fat-tailed hedge fund return distribution.

However another, less documented, plausible cause of the fat-tailed return distribution is that the moments of the returns distribution are time-varying. Agarwal and Naik [6.10] examine the persistence of the hedge fund returns which can be directly linked to serial correlation. The serial correlation of hedge funds has also been studied by Lo and Marakov [6.11] and Chan, Lo and Getamansky [6.12]. Thus, the replication of the hedge fund returns needs to adopt dynamic approaches that count for the dynamic risk exposures of the alternative returns. Different approaches have been developed.

The approach proposed in [6.2], uses option replication theory to dynamically allocate between a risky asset and cash. When a bank sells an option, with a fixed payoff function, it executes at the same time the dynamic trading strategy allowing it to hedge the risk related to this short option position. Kat [6.2] uses the same type of reasoning in characterising the target fund return distribution by a payoff function of a single risky asset.

The investment horizon is taken equal to the synthetic option maturity. Once this payoff function and the maturity are fixed, we only have to use the dynamic replicating strategy corresponding to this payoff to obtain at the maturity the fund return distribution. Finally, Kat argues that the sequence of returns is of no importance for investors as long as replication can give us a return distribution with the desired statistical properties, i.e. mean, variance, correlation with market indexes.

Different approaches consisting of using options to replicate the hedge fund returns exist in the literature. Fung and Hsieh [6.13] showed that the returns from trend following strategies can be replicated by a dynamically managed option-based strategy known as a "lookback option". Mitchell and Pulvino [6.14] modelled the return to merger arbitrage funds by using announced transactions from 1963 to 1998 to construct the return of a specific merger arbitrage strategy. Agarwal and Naik [6.10] combine passive buy-and-hold strategies and option-based strategies to characterize the risks of different hedge fund strategy indexes.
These replicating strategies count for nonlinearities in hedge fund returns using linear replicating methods. However, Amin and Kat [6.15] point out that the replicating strategies involving derivatives seem difficult to be implemented in practice for two reasons. First, it is not clear how many options and which strike prices should be included. Second, since only a small number of ordinary puts and calls can be included, there is a definite limit to the range and type of non-linearities that can be captured.

For this reason, another issue of interest is to use a linear portfolio of liquid assets – such as several traded indexes on stocks, bonds, commodities and interest rates. Fung and Hsieh [6.16] showed that equity-oriented hedge fund indexes have two major exposures: the equity market as a whole, and the spread between small cap and large cap stocks. Thus, the multifactor approaches used to assess the risk exposures of equity returns are being transposed in a dynamic risk exposure framework. But to analyse the hedge fund returns, one has to count for the fact that their risk exposures are likely to change very frequently. For example, as pointed out by Fung and Hsieh [6.9], the trend follower returns are positively correlated with the stock market in bullish market situations and negatively correlated in bear market situations.

Thus, to replicate hedge fund returns using liquid asset classes, it is important to employ dynamic replicating approaches in order to count for dynamic risk exposures of the alternative returns. Lo and Hasanhobvic [6.1] use several observed factors – such as the S&P500 index, the USD return index, the Bond Index, - in order to replicate the returns of more than 1610 individual hedge funds extracted from the TASS Database. They employ a dynamic approach which consists of estimating the risk exposures using a 24 months rolling window. This dynamic approach takes into account the variability of hedge fund risk exposures. Another important issue of their technique is the estimation of beta coefficients using individual hedge funds rather than hedge fund indexes. As pointed out by Chan, Getmansky and Lo [6.11], a disaggregated approach may yield additional insights not apparent from index-based risk models. However, the methodology employed by Lo and Hasanhobvic [6.1] has drawbacks as well. The factor selection is rather ad hoc and does not count for the covariance structure of the fund returns.
6.3 Problem formulation for replication

6.3.1 Problem definition

We are given \( f \) funds. The return of fund \( f = 1, \ldots, f \) between time \( t - 1 \) and time \( t \) is \( R_t(f) \) and is given as a time series for times \( t = 1, \ldots, f \). Each fund \( f \) belongs to class \( C(f) \) depending on the strategy that the fund follows. Typical classes include: statistical arbitrage, macroeconomic, volatility arbitrage, event driven, trend following, etc.

Also given are \( n \) assets. By assets we mean tradeable financial assets or rates and non-tradeable entities such as macroeconomic variables. The return (or change) of asset \( n = 1, \ldots, n \) between time \( t - 1 \) and time \( t \) is \( r_t(n) \) and is also given as a time series for times \( t = 1, \ldots, f \).

For a given fund \( f \), the objective is to replicate its returns by a linear or non-linear combination of the asset returns. The universe of assets to be used for replicating a given fund need not be the complete universe defined above but some sub-universe depending on the class \( C(f) \) of the fund. This, clearly, does not affect the problem complexity in any way.

6.3.2 Problem formulation

A formulation of the above problem for a given fund \( f \) is as follows:

Consider a subset \( T \subset \{1, \ldots, f\} \) of the time sequence \( t = 1, \ldots, f \) and let us call observations at times \( t \in T \), in sample observations. Let \( \omega_{f,t}(n) \) be the weight of asset \( n \) at time \( t \) in the replicating portfolio. Because we are interested in a particular fund \( f \) at this stage, we will drop the suffix \( f \) (for notational simplicity) and write instead \( \omega_t(n) \). For the given fund \( f \) the replicating portfolio is then \( w = \{\omega_t(n) \mid n = 1, \ldots, n; \ t \in T\} \).

The return of this replicating portfolio at time \( t \) is \( R_t(w) = \sum_{n=1}^{n} \omega_t(n)r_t(n) \).

The in-sample mean square error based on the differences between the observed fund returns and the returns of the replicating portfolio, is
\[ e_r(w) = \frac{1}{|T|} \sum_{t \in T} [R_t(w) - R_t(f)]^2 \]

The problem is then to choose \( w \) which produces the minimum in the expression

\[ z = \min_w e_r(w) = \min_w \frac{1}{|T|} \sum_{t \in T} \left[ \sum_{n=1}^{n_{\text{max}}} \omega_t(n) r_t(n) - R_t(f) \right]^2 \]  \hspace{1cm} (6.1)

6.3.3 Some observations

Note that if \( w \) is an unrestricted function of \( t \) (i.e. can be chosen, for a given \( t \), at will) then the problem is trivial and the solution is obvious. In this case, let \( n^* \) be the value that maximises \( \max_n [r_t(n)] \) and \( n_* \) be the value that minimises \( \min_n [r_t(n)] \).

If \( r_t(n_*) \leq r_t(f) \leq r_t(n^*) \) an easily computable portfolio with \( \omega_t(n^*), \omega_t(n_*) \neq 0 \)
and \( \omega_t(n) = 0, \forall n \neq n^*, n_* \) has \( z = 0 \), although alternative solutions may exist. If this condition is not satisfied, then it is clear that the solution is \( \omega_t(n^*) = 1 \) and \( \omega_t(n) = 0, \forall n \neq n^* \) if \( r_t(f) \geq r_t(n^*) \), or \( \omega_t(n) = 0, \forall n \neq n_* \) if \( r_t(f) \leq r_t(n_*) \), although \( z \neq 0 \) in both cases.

In view of the above the linear problem formulation only becomes interesting if additional constraints in the replicating portfolio are included. One such set of constraints is to express the fact that for a given find \( f \), the replicating portfolio changes with time only slowly as explained below.

6.4 Portfolio turnover restrictions

The condition that the replicating portfolio changes slowly with time can be expressed in two different ways. One way is via a hard set of constraints as described below.

6.4.1 Hard constraints for portfolio turnover

Let \( \xi_t(n) = 1 \) if asset \( n \) is in the replicating portfolio at time \( t \) and \( \xi_t(n) = 0 \) otherwise. Let \( \Delta, (0 \leq \Delta \leq 1) \) be a given constant specifying the maximum change
of the portfolio asset weights from one period to the next. Let \( \hat{w}(n), n = 1, \ldots, \hat{n}, \) 
\((0 \leq \hat{w}(n) \leq 1)\) be a given set if constants specifying the maximum proportion of wealth that can be allocated to asset \( n \) at any one time. Let \( M \) be a large number, say \( M = 10^6 \).

We want to impose the following restriction on portfolio turnover.

- An asset can become part of the replicating portfolio and go out of the portfolio at any time i.e. its weight can go from zero to a value and from a value to zero at any time.
- When an asset forms part of the replicating portfolio its weight cannot change abruptly unless it goes to zero. The possible change at each rebalancing time is \( \pm \Delta \).
- When an asset is not in the replicating portfolio it can either remain outside the portfolio or become part of the portfolio with a weight \( \Delta \).

The above three conditions can be expressed as constraints as follows.

\[-(1 - \xi_t(n)) \cdot M + \omega_{t-1}(n) - \Delta \leq \omega_t(n) \leq (1 - \xi_t(n)) \cdot M + \omega_{t-1}(n) + \Delta \quad (6.2)\]

for \( n = 1, \ldots, \hat{n}; \ t, \ t - 1 \in T \)

and

\[-\xi_t(n) \cdot \hat{w}(n) \leq \omega_t(n) \leq \xi_t(n) \cdot \hat{w}(n) \quad (6.3)\]

for \( n = 1, \ldots, \hat{n}; \ t \in T \).

If \( \xi_t(n) = 0 \) then from equation (6.2), \( \omega_t(n) = 0 \) and asset \( n \) is not in the replicating portfolio. If for a specific asset \( n, \xi_t(n) = 1 \), constraint (6.2) becomes

\[\omega_{t-1}(n) - \Delta \leq \omega_t(n) \leq \omega_{t-1}(n) + \Delta\]

which implies that \( \omega_t(n) \) can move up or down only by a certain amount relative to the asset weight of the previous time instance \( \omega_{t-1}(n) \). If \( \xi_t(n) = 1 \) constraint (6.3) becomes \(-\hat{w}(n) \leq \omega_t(n) \leq \hat{w}(n)\), as required.
6.4.2 Penalty function for portfolio turnover

The alternative, and perhaps more realistic way, is to require the portfolio changes to take place slowly on average. A measure of portfolio changes over the entire period is:

\[
\frac{1}{|T|} \sum_{t \in T} \sum_{n=1}^{d} (\omega_{r1}(n) - \omega_{1}(n))^2
\]

which when added as a penalty in the objective function of equation (6.1) using a multiplier \( \lambda \) produces a penalised objective:

\[
z = \min_{w} \left[ \frac{1}{|T|} \sum_{t \in T} \sum_{n=1}^{d} (\omega_{1}(n) r_{1}(n) - R_{1}(f))^2 + \lambda \cdot \frac{1}{|T|} \sum_{t \in T} \sum_{n=1}^{d} (\omega_{r1}(n) - \omega_{1}(n))^2 \right]
\] (6.4)

The multiplier \( \lambda \) is chosen by the user. If \( \lambda \) is set to 0, portfolios can change at will. If \( \lambda \) is set to \( M \), all portfolios will be set to a constant portfolio and would not change from one time to the next.

6.5 Additional constraints

In addition to constraints (6.2) and (6.3) there is one more condition we may wish to impose.

- The performance error between the replicating portfolio and the actual hedge fund strategy should be within certain bounds at all time instances \( t = 1, \ldots, T \).

This is expressed as follows.

\[
R_{1}(f) \cdot (1 - \delta) \leq \sum_{n=1}^{d} \omega_{1}(n) \cdot r_{1}(n) \leq R_{1}(f) \cdot (1 + \delta), \quad \text{for} \quad t \in T
\] (6.5)

Where \( \delta \) is a constant specifying the upper and lower bounds on the replication error with respect to the performance of the hedge fund.

We may also wish to impose a specification on the number of assets in any portfolio. For example, if we want just \( m \) assets, we have,
\[
\sum_{n=1}^{\hat{n}} \xi_i(n) = m, \text{ for } i \in T 
\]  

(6.6)

### 6.6 Simultaneous consideration of funds

The above formulations can be solved individually for each fund \( f \) in isolation or for a-priori-defined "fund indices" created by portfolio of funds with the same classification \( C(f) \). In particular if \( f^* \) is an index created from a given classification \( C^* \) say then the above QP can be defined on this index. We can then insist that funds that are components of this index are individually well-replicated (or that at least a reasonable subset of them are well-replicated) by the same replicating portfolio that replicates \( f^* \). For example, suppose that \( F \) is the set of funds \( f \) with \( C(f) = C^* \) and from which the fund index \( f^* \) was created. We want the portfolio that replicates \( f^* \) well to also replicate well at least a fraction, say \( n \) of the funds in \( F \). This is imposed by:

\[
R_i(f) - \delta - (1 - \Psi_f)M \leq \sum_{n=1}^{\hat{n}} \omega_i(n) \cdot r_i(n) \leq R_i(f) + \delta + (1 - \Psi_f)M
\]

(6.7)

for \( f \in F; \ t \in T \)

were \( \Psi_f = 1 \) if fund \( f \in F \) is well replicated and \( \Psi_f = 0 \) otherwise.

The number of well replicated individual funds is then \( \sum_{f \in F} \Psi_f \) and hence we want \( \sum_{f \in F} \Psi_f \geq n|F| \)

### 6.7 The chosen replication model

The problem, in its generality, can now be defined by equations (6.1), (6.2), (6.3), (6.5). For a specified set of variables \( \xi_i(n) \), and \( \Psi_f \), the above problem formulation is a quadratic programming (QP) problem which is easily solvable (note however that feasibility is not guaranteed for every value of \( \delta \)). Without the 0-1 variables specified, however the above formulation is a non-convex mixed integer QP (MIQP) whose approximate solution can be found by heuristics like the Bionomic algorithm described in section 6.8 below. In our investigation we deal
only with the second of the above formulations and also consider only a subset of constraints, so our model is defined by:

Problem P: Defined by objective function (6.4) and constraints (6.3) and (6.6)

6.8 The Bionomic Algorithm

In order to deal with the complexity of the model, it is necessary to use heuristic methods which are not guaranteed to produce optimal answers, but are designed to give good solutions in reasonable time. The main heuristic families are: simulated annealing, tabu search [6.17, 6.18] and genetic algorithms [6.19].

The Bionomic Algorithm (BA) [6.22] is an intelligent probabilistic search heuristic of the "population-type", similar to genetic algorithms [6.19, 6.20] and scatter search [6.17, 6.18], but with a different motivation for searching the solution space. Like all heuristics, its success or failure depends on its particularization to the specific problem type. Below we explain this particularization for the specific application.

6.8.1 Description of the BA

(1) Generate an initial population of solutions

(1.1) Set \( \xi = 1 \), i.e. \( \xi_t(n) = 1; \ n = 1, \ldots, \hat{n}; \ t = 1, \ldots, \hat{t} \)

Solve the QP for this \( \xi \) and let the solution be \( \omega = [\omega_t(n)] \).

Repeat: Generate a solution \( s \)

(1.2) For each \( t = 1, \ldots, \hat{t} \) generate \( \xi_t \) (i.e. \( \xi_t(n); \ n = 1, \ldots, \hat{n} \)) for that \( t \) by randomly choosing a set \( M \) of \( m = |M| \) assets, setting \( \xi_t(n) = 1 \) for \( n \in M \) and \( \xi_t(n) = 0 \) otherwise. The probability that asset \( n \) is picked for \( M \) is \( \omega_t(n) \).

(1.3) Solve the QP with the \( \xi \) chosen in (1.2) above and with large value of the Lagrangean penalty \( \lambda \) (say \( \lambda' \)) resulting in a value \( \Delta(s) \) for this solution \( s \).
until: enough solutions are generated
Let S be the set of solutions. (Note that all solutions are now feasible for \( \Delta = \max_{s \in S} \{ \Delta(s) \} \).)

(2) **Improve each solution in the initial population**

repeat: for each solution \( s \in S \)
repeat: attempt to improve solution \( s \) to \( s' \)

(2.1) Randomly choose any time \( r \in \{2, \ldots, t - 1\} \)

(2.2) Consider the subproblem consisting of times \( r - 1, r \) and \( r + 1 \) only and call SQP the corresponding QP applied to these 3 times only. The solution to SQP will be superscripted by '.

(2.3) Set \( \xi_{r-1}^{s'} = \xi_{r-1}^s, \omega_{r-1}^{s'} = \omega_{r-1}^s, \xi_{r+1}^{s'} = \xi_{r+1}^s, \omega_{r+1}^{s'} = \omega_{r+1}^s \)

The variables \( \xi_{r-1}^{s'} \) and \( \omega_{r-1}^{s'} \) are to be determined.

(2.4) Solve two SQPs for the variables \( \xi_{r}^{s'} \) and \( \omega_{r}^{s'} \), as follows:
   (i) Set \( \xi_{r}^{s'} = \xi_{r-1}^{s'} \) and solve for \( \omega_{r}^{s'} \) and
   (ii) Set \( \xi_{r}^{s'} = \xi_{r+1}^{s'} \) and solve for \( \omega_{r}^{s'} \)

(2.5) If for either problem (i) or (ii) above, the problem is feasible and the error term for time \( r \) is lower than the corresponding error term for solution \( s \), replace \( s \) with \( s' \).

until: enough attempts have been made
until: all solutions \( s \) have been considered

(3) **Repeat: Select a parent set**
(3.1) Choose a random set of $k$ solutions to form the parent set $P$.

**repeat**: attempts to improve $P$

Choose a random solution $p^+ \in S - P$

**repeat**: Choose an element $p^- \in P$

(3.2) if $z(p^+) < z(p^-)$ and $\Omega(P + p^+ + p^-) \geq \Omega(P)$, replace $p^-$ with $p^+$ in $P$ and exit the loop.

Note: Let $h_s(n)$ be the number of times asset $n$ appeared i.e. has $\xi_s(n) = 1$, in a solution $s \in P$. Then $g(s) = \sum_{n=1}^{\hat{n}} [\hat{\mu} - h_s(n)]^2$ is a measure that increases as the portfolios (at different times) in a solution $s$ are more concentrated in a small number of assets, a desirable feature. On the other hand, we want the totality of portfolios in different solutions of the parent set to be as diverse (i.e. spanning as much of the asset space) as possible.

For $\mu(n) = \sum_{s \in P} h_s(n)$ we maximize $\Omega(P) = [\sum_{s \in P} g(s)] \cdot [\sum_{n=1}^{\hat{n}} \sqrt{\mu(n)}]$.

**until**: all elements of $P$ have been considered.

**until**: enough attempts (successful or not have been made).

(4) **Combine parents for children**

**repeat**: Produce child solutions

(4.1) Choose a random time $\tau \in \{2, ..., \hat{\tau} - 1\}$.

(4.2) Compute the pair $s_1, s_2 \in P$ of solutions for which the distance $\text{dist}[(\xi_{s_1}^\tau, \omega_{s_1}^\tau), (\xi_{s_2}^\tau, \omega_{s_2}^\tau)]$ is minimum.
Consider the subproblem consisting of times $\tau - 1$, $\tau$ and $\tau + 1$ only and call SQP the corresponding QP applied to these 3 times only. The solution to SQP will be superscripted by $'$. 

From $s_1$ and $s_2$ two child solutions can be produced as follows 

(a) $[(x_i^{r_i}, x_o^{r_i}); t = 1, ..., \tau - 1], [()^{\prime}, o^{\prime}]$, $[(x_i^{r_i}, x_o^{r_i}); t = \tau + 1, ..., \hat{t}]$ 

(b) $[(x_i^{r_i}, x_o^{r_i}); t = 1, ..., \tau - 1], [()^{\prime}, o^{\prime}]$, $[(x_i^{r_i}, x_o^{r_i}); t = \tau + 1, ..., \hat{t}]$ 

Solve the SQP for the variables $x_i^{\prime}$ and $o^{\prime}$ as follows 

For case (a): 

Set $x_i^{\prime} = x_i^{r_i}$, $o^{\prime} = o^{r_i}$, $x_i^{r+1} = x_i^{r+1}$, $o^{r+1} = o^{r+1}$ 

The variables $x_i^{\prime}$ and $o^{\prime}$ are to be determined and we make two solution attempts 

(i) Set $x_i^{\prime} = x_i^{r_i}$ and solve for $o^{\prime}$ or 

(ii) Set $x_i^{\prime} = x_i^{r+1}$ and solve for $o^{\prime}$ 

For case (b): 

Set $x_i^{\prime} = x_i^{r_i}$, $o^{\prime} = o^{r_i}$, $x_i^{r+1} = x_i^{r+1}$, $o^{r+1} = o^{r+1}$ 

The variables $x_i^{\prime}$ and $o^{\prime}$ are to be determined and we make two solution attempts 

(i) Set $x_i^{\prime} = x_i^{r_i}$ and solve for $o^{\prime}$ or 

(ii) Set $x_i^{\prime} = x_i^{r+1}$ and solve for $o^{\prime}$ 

We now have 4 possible solutions; two for child (a) and two for child (b) with values for the corresponding SQPs: $z_{a,t}(SQP)$, $z_{a,i}(SQP)$, $z_{b,i}(SQP)$, $z_{b,i}(SQP)$. Let $z_a(SQP) = \min[z_{a,t}(SQP), z_{a,i}(SQP)]$ and $z_b(SQP) = \min[z_{b,t}(SQP), z_{b,i}(SQP)]$.
Chapter 6

Replication of hedge fund returns

(4.9) The values of the two child solutions are then

(a) \( z^a = z^{h_1}_{[t_{r-1}, t]}(QP) + z_a(SQP) + z^{h_1}_{[t_{r-1}, t]}(QP) \) and

(b) \( z^a = z^{h_2}_{[t_{r-1}, t]}(QP) + z_b(SQP) + z^{h_2}_{[t_{r-1}, t]}(QP) \)

Where for (a) \( z^{h_1}_{[t_{r-1}, t]}(QP) \) is the value of solution \( s_1 \) from time \( t = 1, \ldots, t - 1 \)

\( z^{h_2}_{[t_{r-1}, t]}(QP) \) is the value of solution \( s_2 \) from time \( t = t + 1, \ldots, \hat{t} \)

and similarly for (b). Note that these values are simply the sum of the terms in the QP solutions for the corresponding time periods.

(4.10) Store the child solutions in a list \( C \)

until: enough child solutions are produced

(5) Add a child to the population

(5.1) Let \( c \in C \) be the least cost child solution and improve this solution using the same procedure described in step 2 above.

(5.2) Form a set \( D \) of all solutions \( s \in S \) for which \( z(c) < z(s) \) and

\( \Omega(S + c - s) \geq \Omega(S) \).

if \( D \) is empty, form a set \( D \) of all solutions \( s \in S \) for which \( z(c) < z(s) \)

if \( D \) is empty, set \( D = \{s\} \) where solution \( s \in S \) is the “oldest” solution

in \( S \) (excluding the currently best solution)

Note: When a solution enters the population, its age is set to 0. Whenever the solution is chosen for a parent set, its age is increased by 1. The reason for replacing an “old” solution in the population with a child is that does not dominate it is to avoid premature convergence and cycling, namely that the same solutions are continuously picked for parent sets, but their offspring are continuously rejected.
(5.3) Let $s^*$ be the solution in $D$ with the largest cost. Delete $s$ from $S$ and replace it with $c$.

(6) until: Enough parent sets are considered

(6.1) The stopping criterion is reached if:
(i) The population remained unchanged after 10 parent sets were tried sequentially, or
(ii) the number of child solutions that entered the population exceeded 10 times the initial population

(6.2) Report the best solution encountered

6.9 Out-of-sample forecasts

The output from the optimization model (OM) is a sequence of portfolios. One for each time, whose returns replicate approximately the fund returns. At different times, the portfolios are different both structurally (i.e. in terms of the set of assets that the wealth is invested in) and in terms of the fraction of wealth invested in each asset of the portfolio. The constraint is imposed that the structure of the portfolio can change only slowly. At a given time $t$ the inputs are the asset returns at time $t$ (say the end of month $t$) and the target we try to replicate is the fund return $R_t(f)$ (which is the fund return at the end of the month $t$). Note that $R_t(f)$ is not known at the end of the month $t$ but at the end of the month $t+1$. Thus if the time now is $\tau$, our time series for OM goes from time 1 to time $\tau-1$, because the return $R_t(f)$ is unknown for $t > \tau-2$. In order to forecast the value of $R_t(f)$ at the end of month $\tau$ the last portfolio we have from the OM is $(\xi, \omega)_{\tau-2}$ and the last asset returns we have are for the end of month $\tau-1$. Thus, we need to predict the asset returns at the end of month $\tau$ (one month ahead of the last observation) and the portfolio at the beginning of month $\tau$ (two months ahead of the last observation).

The most trivial forecast is to assume the same portfolio at $\tau$ as at $\tau-2$ and the same asset returns at $\tau$ as at $\tau-1$. A better way is to produce an in-sample map
Chapter 6 Replication of hedge fund returns

from asset returns at the end of the month \( t \) (or include the last few months before that) to the portfolio at time \( t+1 \). Using this map we can get the portfolio at time \( \tau \) which is the replicating portfolio. A forecast of the asset returns at the end of the month \( \tau \) (clearly this is a much easier problem than the one above) would then also enable a forecast of the fund return at the end of the month \( \tau \).

This can be done, for example, as an independent a-posteriori addition to the OM. For example using a neural network we can train with the asset returns as inputs and the \( \omega_i(n) \) as outputs, one NN per asset \( n \) in the replicating portfolio.

6.10 Neural network framework and data issues

This section explains the use of a NN-based algorithm for forecasting fund portfolios and returns. It is assumed that the mixed-integer quadratic optimisation (MIQP) code has already been run to derive some of the information needed for this section.

We start by summarising the results from MIQP. At time \( t \), the asset returns are for period \([t-1,t]\) and are represented by \( r_i(n) \). The fund returns during this period are \( R_i(f) \). From these returns (and some reasonable assumptions) MIQP computed the portfolio of assets \((\xi,\omega_i)\) that could have been bought at time \( t-1 \) and held until time \( t \), so as to produce a return approximately equal to \( R_i(f) \) during period \([t-1,t]\).

6.10.1 Neural network input data

The input vector to the neural network needs to be a combination of time series representing the asset returns \( r(n) \) as well as a lot of global financial data that could be influential in forecasting the outcome. Not all items need to be available for trading.

We first need to decide how many and what are going to be the inputs to the neural network. These inputs are supposed to describe the state of the system from where a prediction will be made.

The more data we use as inputs the higher the dimensions of the system to forecast. In chapter 4 we showed that this can lead to bad forecasting results as the
well known linear and non-linear forecasting techniques fail to produce good results when the dimensionality of the system is high. An alternative would be to use the part of the hybrid model (proposed in chapter 4) that decomposes the high dimensional system into a few “meaningful” smooth independent components and use those as inputs to the neural network.

6.10.2 The neural network target

The neural network will be trained so that given the set of the above inputs at a given instance \( t \) the neural network predicts the portfolio \((\xi_{t+1}, \omega_{t+1})\) that is to be bought at time \( t \) and held until time \( t+1 \). Note that MIQP used \( r_{t+1} \) and \( R_{t+1} \) to produce \((\xi_{t+1}, \omega_{t+1})\), whereas now we will bypass the prediction of \( r_{t+1} \) and \( R_{t+1} \) and forecast \((\xi_{t+1}, \omega_{t+1})\) directly from \( r_t \). Thus for each time \( t \) for the neural network inputs we have a corresponding target \((\xi_{t+1}, \omega_{t+1})\). Let's assume that these targets form \( S_t = (\xi_{t+1}, \omega_{t+1}) \).

6.10.3 The training period

From the total time series observations we have to use some consecutive observations (say from date \( t_a \) to date \( t_a + p \)) to train the neural network. Let us refer to this time period as training period \( a \). The \( p \) observations of training period \( a \) are partitioned randomly into 3 contiguous sets the training set, validation set and testing set.

6.11 Network's structure

The network used is a multi-layer perceptron. It structure is very similar to that of the network in chapter 4 section 4.15 and is shown in figure 6.1.

The number of input vertices in the input layer equal the number of independent components that represent the system. In layer 2 there are \( h \) hidden neurons each of which has a hyperbolic tangent sigmoid function as its transfer function. The output layer of the network outputs the predicted portfolio at time \( t + 1 \). It has as many vertices as there are assets in the portfolio. Notice that in figure
6.1, the transfer functions of the hidden neuron are shown as hyperbolic tangent sigmoid functions. This was chosen for demonstration purposes only.

![Diagram of neural network](image)

**Figure 6.1: Structure of neural network for replication of hedge fund returns**

### 6.12 Neural network weights determination

Starting from a random set of arc weights say $g$, the conjugate gradient algorithm is applied to improve the weights so that the mean-square-error (MSE) between the neural network output (of predicted portfolios) and the targets, for those observations in the training set, is minimized. The validation set is used to stop the training iterations before the 'optimum' is reached, to avoid overfitting. The evaluation of the resulting (trained) neural network is the MSE performance out-of-sample, on the test set. Note that because the objective function for the neural network weight optimization is non-convex, we a-posteriori apply a randomization step after each gradient step taken by the conjugate gradient algorithm. This randomization involves not only changing the arc weights, but also setting some arc weights that are near-zero to zero (i.e. removing the arc). If a situation arises whereby all in-arcs to a given vertex are eliminated, the vertex (neuron) itself and all
its out-arcs are also eliminated, etc. Thus, the topology of a network can change during the training process.

6.13 Training a committee of networks

6.13.1 Varying the start weights and the neural network structure

Because, as mentioned above, the problem of finding the best weights for a neural network is a non-convex problem with multiple minima and because the conjugate algorithm and interleaved randomization iterations may be stopped prematurely, different trained neural networks result from different starting weights $g$ for a given initial NN structure.

Thus, if we repeat the procedure for $g_1, g_2, ...$ (say 10 times), we will have 10 trained neural networks. If, in addition, we repeat the above procedure for a total of 15 different initial structures, we have 150 trained networks. A simple criterion for choosing which one of those 150 networks is likely to perform better on unseen data, is called the progression coefficient [6.21]. Let us call the best selected NN from the 150 neural networks above, $NN(a, 1)$.

6.13.2 Varying the partition of the training period

Training period $a$ above can be partitioned into training set, validation set and test set, randomly, in a different way, in which case the above procedure will lead to another trained neural network $NN(a, 2)$. Similarly, other partitions of training period $a$ lead to $NN(a, 3), ...$ etc. Note that if there is a very small number of available data, then it won’t be possible to generate many neural networks in this way. Let’s suppose that we generate 4 networks this way. Note that each of them was obtained as the “best” of 150 neural networks, so in this case a total of 600 neural networks had to be trained to obtain a committee of 4.

6.13.3 Producing a forecast

A given neural network from the 4 in the committee, when presented with the inputs of today will produce a forecast of the portfolio to be held for the period ahead. The 4 neural networks will, in general, produce different forecasts and we take the average of these as the final forecast which will be much more robust.
6.13.4 Evaluating the effectiveness of a neural network committee forecast

The most recent observations of the data time series are kept purely for evaluating the effectiveness of a neural network committee forecast. We call this set the evaluation set. A multidimensional effectiveness vector is computed comprising of:

- **PSE**: The percentage when the sign of the portfolio return was predicted correctly, i.e. was of the same sign as the fund return. (Should be as high as possible)
- The ‘worst’ prediction error in the evaluation set, considering only under-predictions. (Should be as close to 0 as possible.)
- The correlation coefficient between the forecasted and actual returns. (The higher the better.)
- The ratio of the Sharp ratios of actual and forecasted returns. (Should be as close to 1 as possible)

6.14 Different neural network committees

Note that the entire neural network population and forecast above, depends on the initial choice of the training period $\alpha$. If another training period say from date $t_\beta$ to date $t_\beta + p$ is chosen to train the neural networks then a different population $NN(\beta, 1), NN(\beta, 2), \ldots$ etc results. Similarly, for yet another training period $\gamma$, say from date $t_\gamma$ to date $t_\gamma + p$, a different population $NN(\gamma, 1), NN(\gamma, 2), \ldots$ etc results. When these NN populations are evaluated on the evaluation set they produce an ordering from the best to the worst predictor, as far as recent history is concerned.

When used to forecast one period ahead, the 3 populations above may agree or disagree. If the 3 NN population forecasts have good “effectiveness vectors” and their forecasts agree, then the confidence in the return forecast of the fund as realized by the return of the chosen portfolio is high.

If either the ‘effectiveness vectors’ are bad or there are major disagreements in the 3 predictions (e.g. the most effective network population forecasts one thing and the other 2 another), then the confidence in the portfolio is low.
6.15 Conclusions

In this chapter we presented a novel mathematical formulation applied to the problem of effective replication and forecasting of hedge fund returns. The objective was to minimise tracking risk over time of the replicating portfolio relative to non-investible target hedge fund index, subject to turnover and risk criteria.

The proposed methodology is split in two parts. The replication (modelling) part, that involves dynamic weighting of instruments and strategies and the forecasting part. In the first part the methodology uses mixed integer quadratic programming and heuristics and in the second part committees of neural networks designed with different initialisations and trained on different training sets.

This methodology utilises top-down robust optimisation tools to capture time varying factor exposures, it makes no explicit assumptions on the underlying hedge fund process and has disciplined risk control tracking and turnover constraints.

In the following chapter we will explain the implementation procedure and discuss any practical considerations in more detail. Finally we will present experimental results on 4 hedge fund strategies (targets) that form part of a composite index.
In this chapter we implement the method that was proposed in chapter 6 for effective replication of hedge fund returns. The model and all the practicalities surrounding it are discussed and experimental results derived by running multiple scenarios on various data sets.

### 7.1 Original Data

The input data to this application is time series of global asset prices and hedge fund indices.

The global assets are 42 global indices and investment strategies. This set of data is highly diversified and includes major equity indices, commodities indices, credit indices, FX indices, interest rates linked indices, volatility indices and traded strategies. A list of the assets can be seen in table 5.1 of chapter 5. This time, unlike in chapter 5 we will use monthly data as the hedge funds whose underlying strategy we are trying to "guess" only report their results once a month.

In terms of target hedge fund indices we chose 4 indices that represent different styles of investment. The first one is an equity index (EQ), the other an event driven index (ED), the third is a macro index (Macro) and the final one a relative value index (RV). These indexes form a composite (Comp) with respective weights of 50%, 10%, 20% and 20%.

Before doing any analysis on the data we transform the non stationary asset prices to asset returns.
7.2 Partitioning the data set

The time series of the above data starts in January 1999 and ends in December 2007 (108 samples). As the data is not ideally long for training we create two scenarios for testing the performance of the model. The first one is to use each time series as it is and try and do well with only a few samples for training and validation.

The alternative is to increase the number of samples by repeating history. In other words, if the first sample of the time series is \( t_1 \) and the last one \( t_n \), create a time series \( \{ t_1, \ldots, t_{2n} \} \) whose first half \( t_1, \ldots, t_n \) is identical to the second one \( t_{n+1}, \ldots, t_{2n} \). Then exclude the sample that corresponds to the out-of-sample time instance for which a portfolio needs to be forecasted to avoid biases. In order to be more restrictive we also tried excluding to the two previous and two next samples.

In both the above scenarios, we assume a variable length window from the first sample to sample \( t \) to forecast the value at \( t+1 \). The period \( 1 \) to \( t \) was partitioned in two sets, the training and validation set by using an 80-20 rule (80% of the samples used for training and the remaining for validation).

7.3 Step-by-step procedure

As shown in diagram 7.1 below there are two main implementation phases to replicating and forecasting hedge fund returns. The first one includes modules that are involved in the in-sample modelling (yellow) and the second those that are involved in the out-of-sample forecasting (blue).

![Figure 7.1: Schematic representation of the replication and forecasting procedure.](image-url)
The step-by-step procedure is presented below.

**Phase A: In-sample modelling**

Step 1. Gather data for the replicating portfolio candidate instruments and the hedge fund indices to be replicated.

Step 2. If there is any strong view about any of the assets in the universe determine its presence in the portfolio by setting its $\xi$ value (defined in 6.4) either to 1 (if this asset is definitely a candidate), or a 0 (if this asset needs to be excluded from the candidates), or let the algorithm decide whether to include it or not by showing no preference.

Step 3. Adjust the target returns to account for hedge fund fees. This involves increasing the target returns for every target index at each time period $t$ by incorporating the corresponding management and or performance fees to the NAV of the sub index at period $t-1$ [Box 2].

Step 4. Adjust the inputs and target so that they are mapping inputs at time $t$ to targets at $t+1$.

Step 5. Set some parameters for the optimisation constraints. Set a number for the maximum number of possible assets in your portfolio. Define the upper and lower boundaries of weights for each one of the portfolios. Solve the mixed integer quadratic programming to generate initial populations of solutions for various penalty parameters $\lambda$. [Box 3-4]

Step 6. Use the Bionomic algorithm to loop through the following 5 items

a) take the initial populations generated by the quadratic programming
b) improve each solution in the initial population
c) select a parent set
d) combine parents for children
e) add a child to the population

until either the population remains unchanged after 10 parent sets were tried sequentially or the number of child solutions that entered the population exceeds 10 times the initial population [Box 3-5].

**Phase B: Out-of-sample forecasting**
Chapter 7

Implementation of the replication model

Step 7. Select your inputs and targets for forecasting. These include exogenous data like global financial data, macro variables, instrument returns at time $t$, and the portfolios to be bought at time $t$ and be held until time $t+1$.

Step 8. Use principal component analysis and independent component analysis to decompose the high dimensional input system to few "important" driving factors.

Step 9. Partition time for training and evaluation. Select some consecutive time periods for the training set. The starting period can be anything. Use the validation set to stop the training before you reach the optimum in order to avoid overfitting. Evaluate the resulting neural network by looking at the MSE performance out-of-sample on the testing set. Keep the last samples for the evaluation set in order to evaluate the effectiveness of a neural network population forecast.

Step 10. Initialize weights. Start with a random set of arc weights. Apply the conjugate algorithm to improve them.

Step 11. Create committees of neural networks in two ways:
   a) use different initial weights for a given initial neural network structure
   b) use different initial neural network structures

   Use the progression coefficient to which neural network from the population is the best.

Step 12. Repeat step 9 using different partitions of the training period

   Each time pick the best neural network and form a family of networks from those. For the final result take their average.

Step 13. The neural network when presented with the inputs of today will produce a forecast of the portfolio to be held for the period ahead.

Phase C: Composite composition using predefined weight matrix

This procedure is specific to the examples in this chapter. As mentioned in section 7.1 the four hedge fund indices that we use as targets for replication, form a composite index with weights as specified in section 7.1. So after we have constructed trading portfolios for each one of the indices we reconstruct the composite value.
7.4 Setting parameters and benchmarks

Analysis has been performed on all 4 hedge fund indices described in 7.1. We assumed a maximum number of candidate assets of 10 and parameter $\lambda = 0.2$ (which has in general produced good results in other previous tests). All strategies are long-short. Results are compared against the targets themselves and equivalent Hedge Fund Research Indices (HFRI). These indices are produced by Hedge Fund Research Inc, a data provider with more than 7000 funds in its database. Their indices are good benchmarks for cases were investable exposure is needed.

7.5 Scenario 1: Back test for period 1999 to 2007

As explained in section 7.2, in this scenario we have twice the data available, therefore 215 samples. We train on the first 107 and then we use the remaining ones for the back test i.e. 1999 to 2007.

7.5.1 Replication of the EQ index

In figure 7.2 we present the replication of the EQ index. This figure is supported by table 7.1. It is obvious that our models performance is quite close to the Hedge Fund’s one and much better than the HFR Index.

![Figure 7.2](image)

**Figure 7.2:** Cumulative return of the EQ index from 1999 to 2007 and of the replicating portfolios of the HFRI and the proposed model

More specifically the average annual return performance from 1999 to 2007 our model ranked first which is good as what we want is to deliver return. In terms of
volatility we are still below that of the target but above the HFRI one. So overall we outperform the target with less volatility. The maximum draw down is the smallest of the three and the information at 1.42 is the highest.

<table>
<thead>
<tr>
<th></th>
<th>Our model</th>
<th>Target</th>
<th>HFRI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Price return performance</td>
<td>9.29%</td>
<td>7.25%</td>
<td>5.91%</td>
</tr>
<tr>
<td>Risk (annualized volatility)</td>
<td>6.52%</td>
<td>7.92%</td>
<td>6.03%</td>
</tr>
<tr>
<td>Information ratio</td>
<td>1.42</td>
<td>0.92</td>
<td>0.99</td>
</tr>
<tr>
<td>Maximum drawdown</td>
<td>5.14%</td>
<td>10.30%</td>
<td>8.63%</td>
</tr>
<tr>
<td>Tracking risk</td>
<td></td>
<td></td>
<td>4.74%</td>
</tr>
</tbody>
</table>

Table 7.1: Average annual statistics of the performance of the EQ Index, the HFRI and our model’s strategy for years 1999 to 2007

7.5.2 Replication of the ED index

Similarly for the ED index the performance of our model is superior. The annual average return is 11.35% whereas that of HFRI is at 7.67. Even for this strategy we are outperforming the target in both returns and risk.

![Cumulative return of the ED index from 1999 to 2007 and of the replicating portfolios of the HFRI and the proposed model](image)

Figure 7.3: Cumulative return of the ED index from 1999 to 2007 and of the replicating portfolios of the HFRI and the proposed model

<table>
<thead>
<tr>
<th></th>
<th>Our model</th>
<th>Target</th>
<th>HFRI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Price return performance</td>
<td>11.35%</td>
<td>9.22%</td>
<td>7.67%</td>
</tr>
<tr>
<td>Risk (annualized volatility)</td>
<td>5.77%</td>
<td>5.91%</td>
<td>5.05%</td>
</tr>
<tr>
<td>Information ratio</td>
<td>1.97</td>
<td>1.59</td>
<td>1.52</td>
</tr>
<tr>
<td>Maximum drawdown</td>
<td>4.11%</td>
<td>9.34%</td>
<td>7.96%</td>
</tr>
<tr>
<td>Tracking risk</td>
<td></td>
<td>3.99%</td>
<td></td>
</tr>
</tbody>
</table>

Table 7.2: Average annual statistics of the performance of the ED Index, the HFRI and our model’s strategy for years 1999 to 2007
7.5.3 Replication of the Macro index

Figure 7.4 and table 7.3 show that our model’s replication of this index is particularly good. The model produced the best results in all the statistics. With the highest return, the smallest volatility, highest information ratio and minimum drawdown it clearly outperformed the other two.

**Figure 7.4:** Cumulative return of the Macro index from 1999 to 2007 and of the replicating portfolios of the HFRI and the proposed model

<table>
<thead>
<tr>
<th></th>
<th>Our model</th>
<th>Target</th>
<th>HFRI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Price return performance</strong></td>
<td>8.64%</td>
<td>8.42%</td>
<td>7.34%</td>
</tr>
<tr>
<td><strong>Risk (annualized volatility)</strong></td>
<td>5.12%</td>
<td>5.65%</td>
<td>8.33%</td>
</tr>
<tr>
<td><strong>Information ratio</strong></td>
<td>1.89</td>
<td>1.49</td>
<td>0.82</td>
</tr>
<tr>
<td><strong>Maximum drawdown</strong></td>
<td>4.82%</td>
<td>7.31%</td>
<td>10.28%</td>
</tr>
<tr>
<td><strong>Tracking risk</strong></td>
<td></td>
<td>5.75%</td>
<td></td>
</tr>
</tbody>
</table>

**Table 7.3:** Average annual statistics of the performance of the Macro Index, the HFRI and our model’s strategy for years 1999 to 2007

7.5.4 Replication of the RV index

In Table 7.4 we can see that the statistics were still ranking our strategy number 1 in terms of volatility, information ratio and maximum drawdown. Despite this the strategy failed to deliver high returns (below target). Moreover the replication as seen in figure 7.5 was very bad.
Figure 7.5: Cumulative return of the RV index from 1999 to 2007 and of the replicating portfolios of the HFRI and the proposed model

<table>
<thead>
<tr>
<th></th>
<th>Our model</th>
<th>Target</th>
<th>HFRI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Price return performance</td>
<td>7.25%</td>
<td>7.91%</td>
<td>3.31%</td>
</tr>
<tr>
<td>Risk (annualized volatility)</td>
<td>2.16%</td>
<td>2.45%</td>
<td>3.95%</td>
</tr>
<tr>
<td>Information ratio</td>
<td>3.35</td>
<td>3.23</td>
<td>0.84</td>
</tr>
<tr>
<td>Maximum drawdown</td>
<td>1.55%</td>
<td>2.30%</td>
<td>3.72%</td>
</tr>
<tr>
<td>Tracking risk</td>
<td></td>
<td></td>
<td>2.38%</td>
</tr>
</tbody>
</table>

Table 7.4: Average annual statistics of the performance of the RV Index, the HFRI and our model’s strategy for years 1999 to 2007

7.6 Scenario 2: Back test for period 2003 to 2007

In this scenario we don’t repeat historical data and use only the actual data that we have been given originally. Therefore we have 108 samples, 57 of which we use for training and the remaining for the back test. After performing exactly the same analysis as above we get the following summary results.

<table>
<thead>
<tr>
<th></th>
<th>Our model</th>
<th>Target</th>
<th>HFRI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Price return performance</td>
<td>11.20%</td>
<td>8.80%</td>
<td>5.90%</td>
</tr>
<tr>
<td>Risk (annualized volatility)</td>
<td>5.20%</td>
<td>6.40%</td>
<td>6.00%</td>
</tr>
<tr>
<td>Information ratio</td>
<td>2.20</td>
<td>1.30</td>
<td>0.98</td>
</tr>
<tr>
<td>Maximum drawdown</td>
<td>2.10%</td>
<td>8.00%</td>
<td>8.70%</td>
</tr>
<tr>
<td>Tracking risk</td>
<td></td>
<td>4.70%</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Our model</th>
<th>Target</th>
<th>HFRI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Price return performance</td>
<td>11.00%</td>
<td>9.60%</td>
<td>7.70%</td>
</tr>
<tr>
<td>Risk (annualized volatility)</td>
<td>5.80%</td>
<td>5.00%</td>
<td>5.10%</td>
</tr>
<tr>
<td>Information ratio</td>
<td>2.00</td>
<td>1.90</td>
<td>1.50</td>
</tr>
<tr>
<td>Maximum drawdown</td>
<td>2.90%</td>
<td>5.80%</td>
<td>9.00%</td>
</tr>
<tr>
<td>Tracking risk</td>
<td></td>
<td>3.90%</td>
<td></td>
</tr>
</tbody>
</table>

Table 7.5: Average annual statistics of the performance of the a) EQ and b) ED Indices for period 2003 to 2007
Chapter 7 Implementation of the replication model

Maximum drawdown

Table 7.6: Average annual statistics of the performance of the a) Macro and b) RV Indices for period 2003 to 2007

<table>
<thead>
<tr>
<th></th>
<th>Our model</th>
<th>Target</th>
<th>HFRI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Price return performance</td>
<td>9.00%</td>
<td>9.20%</td>
<td>7.30%</td>
</tr>
<tr>
<td>Risk (annualized volatility)</td>
<td>4.70%</td>
<td>5.00%</td>
<td>9.00%</td>
</tr>
<tr>
<td>Information ratio</td>
<td>1.90</td>
<td>1.60</td>
<td>0.80</td>
</tr>
<tr>
<td>Maximum drawdown</td>
<td>1.50%</td>
<td>4.20%</td>
<td>10.30%</td>
</tr>
<tr>
<td>Tracking risk</td>
<td>4.00%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) Price return performance | 7.10%   | 7.20%  | 3.30%|
Risk (annualized volatility) | 2.30%   | 2.80%  | 4.00%|
Information ratio | 3.10  | 2.60   | 0.80 |
Maximum drawdown | 1.50%  | 2.30%  | 9.70%|
Tracking risk | 2.50%  |        |      |

The model has given worse results as expected. The fewer the samples that are used for training the neural network, the worse the performance. Still though in general it outperformed the other indices with the only exception that of RV Index were it failed to deliver high returns

7.7 Aggregate level performance: the Comp index

Figure 7.6 shows the aggregate results for Comp index which is composed by

Figure 7.6: Price of the Comp index from 1999 to 2007 and of the replicating portfolios of the HFRI and the proposed model
the indices as explained in 7.1. Table 7.8 shows the statistics of the performances of the 3 strategies. At the aggregate level our model still outperforms the rest in terms of returns. In volatility the HFRI has the smallest one but with only 3.6\% return compared to 10.6\% for our model. With 82\% and 76\% correlation with the target our model’s strategy proves to be a good tool for replication.

### 7.8 Conclusions

In this chapter we discussed the implementation of the methodology described in chapter 6 for the replication of hedge fund returns. We presented a step-by-step implementation procedure and discussed practical assumptions that need to be taken into account.

We then tested our methodology on real data from 4 hedge fund indices of different investment style. From the results it is evident that the methodology works as it delivered good results for 3 out of the 4 strategies. The encouraging result is that not only it tracks well but in most of the cases outperforms the target by achieving higher returns with smaller volatility.
8.1 Conclusions

The aim of this research work was to add value to one of the most important areas in finance, the modelling and forecasting of asset price dynamics. Most theoretical and practical analysis in finance requires a thorough understanding of the underlying processes that drive the markets.

Linear data analysis tools have proven to be both theoretically sound, and of practical interest in simpler applications but fail to generalise to deal with non-linear financial and economic phenomena. Their calibration becomes complex, inaccurate and the number of parameters to be estimated very large. Similarly with non-linear parametric tools, although they are suitable for financial non-linear modelling, they too, become too complex in trying to build a physically sound model for the underlying process.

The work presented in this thesis was based on a non-linear, non-parametric tool, the neural network. Neural networks are adaptive computational methods that have the ability to extract rules without having them explicitly formalized. They not only work well within their domain of knowledge but deliver successful results with missing or incomplete information, also. They handle dynamic data better and can generalize and make "educated guesses".

Unfortunately, even with neural networks, the modelling of high-dimensional systems, can be problematic. This was the reason that led us to dedicate a substantial part of our research in dimensionality reduction techniques (as presented in chapter
4). We wanted to bypass the problems of high dimensions in order to still make use of the very powerful properties and unique forecasting abilities of a neural network.

The fact that neural networks maximise or minimise an objective function in a non-linear context implies the use of sophisticated optimisation algorithms. Problems related to the efficiency of these algorithms are often mixed with the difficulty of estimating the topology of a network for a particular problem.

By assessing the situation it was obvious that our objective to use neural networks to forecast high dimensional systems could not materialise without first dealing with both issues of a) reducing the system’s dimensionality and b) optimising the neural network’s topology for our particular task.

Our proposed solution, a hybrid model, is an original piece of work that combines linear and non-linear techniques and was presented in this thesis in the context of forecasting high dimensional financial systems. It is called a “hybrid” because its operations include a mixture of linear decompositions/reconstructions of asset prices, non-linear forecasting of the decomposed signals, non-linear modelling of the reconstruction’s residuals and evolutionary design of neural network committees.

More specifically, our proposed methodology starts with an orthogonal decomposition of the asset price time series into principal components and from those it deduces the “most important” independent components. It then models the behaviour of those independent components in an agnostic non-parametric manner using neural networks which fully maintain any nonlinear behaviour. It continues by using another neural network to extract the non-linear behaviour of the resulting reconstruction ‘residuals’ which are partly the result of nonlinear dependence among the prices and partly the result of random errors. The process ends with the modelling and forecasting of each one of the independent components and the construction of an overall hybrid model of asset price dynamics by combination of the linearly reconstructed future returns (using the ICs) and the corresponding forecasted residuals.

The two main highlights of the proposed model are that it is non-linear and that it reduces the dimensionality of the input space without any information on the system’s intrinsic dimension. This is achieved by first searching in a low
dimensional space of simple networks, and gradually making them more complex as the search progresses by elaborating on existing solutions. The high dimensional space of the final solution is only encountered at the very end of the search. This increases the system’s efficiency by guaranteeing that the network becomes no more complex than necessary.

The modelling performance of the system is further improved by searching not only for one network as the ideal solution to a specific problem, but a combination of networks. These committees of networks are formed by combining a diverse selection of networks from a population of networks derived by the proposed method. This approach automatically exploits the strengths and weaknesses of each member of the committee while avoiding having all members giving the same bad judgements at the same time.

Exhaustive experimental tests were conducted under cross validation on a well diversified universe of global financial assets. Results from the hybrid model were consistently better that those achieved by direct ICA reconstruction without residuals (conventional way). These good forecasting results were also obtained for extra volatile signals. Back tests on 42 assets revealed that the algorithm was indeed capable of forecasting as it successfully reconstructed (within a degree of error) 33 out of 42 of the original asset time series. At the end of the cross validation a very important verification of the number of optimal components was achieved. The main disadvantage behind the implementation of the methodology was the long computational time.

In this thesis we also presented a novel mathematical formulation applied to the problem of effective replication and forecasting of hedge fund returns. This methodology consists of effectively “guessing” the portfolio composition of successful hedge fund strategies. Our solution could be applied by a bank to provide customers who can not invest in a hedge fund, with aggregate hedge-fund-like returns at a fraction of the cost. The objective is to minimise tracking risk over time of the replicating portfolio relative to non-investible target hedge fund index, subject to turnover and risk criteria.

The proposed methodology is split in two parts. The replication (modelling) part, that involves dynamic weighting of instruments and strategies and the forecasting part. In the first part the methodology uses mixed integer quadratic
programming and heuristics and in the second part our proposed hybrid model of chapter 4.

This methodology utilises top-down robust optimisation tools to capture time varying factor exposures, it makes no explicit assumptions on the underlying hedge fund process and has disciplined risk control tracking and turnover constraints.

Experiments using the methodology were run on an asset universe of 42 well diversified global assets and strategies (inputs) and 13 hedge fund strategies (targets) that form part of a composite index. The results were encouraging for most of the cases. The few exceptions were very volatile strategies like emerging market strategies, for which the methodology failed to converge. Overall at the composite level the methodology showed substantial tracking performance.

8.2 Future work

In the context of neural network design we propose the use of genetic algorithms to address problems of simultaneous evolution of topology and weights. Genetic algorithms make matching of genes without loss of information possible due to their robust genome encoding. Therefore they can be used to encourage new neural network solutions to do well by placing them into appropriate species and minimising the number of parameters being searched by starting with a population of minimal structure and then add structure as necessary. In a similar note to our proposed combination of networks into committees, genetic algorithms could mate multiple individuals from their final population into a committee, too.

When we implemented our proposed hybrid model (chapter 4) we noticed that the behaviour of some of the assets that belonged to a specific sector/class were affected more by some of the independent components than others. In some cases they were very highly correlated with some of the independent components. We were expecting assets of the same class to be affected by the same independent components which was mostly the case. But there were occasions when assets from different classes were highly correlated with some of the component which makes us suggest the usage of ICA as a sectorization tool. This would allow the sectorization of the input space into baskets of assets affected by the same underlying signals.
Chapter 2


[2.05] Scott L.O. (1996), "Simulating a multifactor term structure model over relatively long discrete time periods", in IAFE: First Annual Computational Finance Conference, Graduate School of Business, Stanford University, USA


References


Chapter 3


References


References


References

Artificial Intelligence to Improve Real World Performance, pp 64–65. Probus, Chicago


Chapter 4


References


Chapter 6


