Investigation of Artificial Neural Networks for Forecasting and Classification

Submitted for qualification as Doctor of Philosophy

by Paul James Worthy

City University
Northampton Square
London, EC1V OHB.

Department of Systems Science

March 1998

TABLE OF CONTENTS

CHAPTER 1 - INTRODUCTION 1

1.0 Background 1

1.1 Artificial Neural Networks 1

1.2 Research Hypothesis 4

1.3 Research Aim 5

1.4 Research Objectives 5

1.5 Application Areas 5

1.4 Thesis Structure 6

CHAPTER 2 - LITERATURE REVIEW 8

2.0 Introduction 8

2.1 The history of the theory of mind 12

2.2 Neural Networks, Parallelism, Connectionism...Some definitions 15

2.3 Critical Review 16
2.3.1 - Chronology of computational ANN research 17
2.3.2 ANN Reviews 19
2.3.3 - Prominent models 21
2.3.3.1 Feedforward Networks 24
Definition 25
Operating principles 25
Computational capability of feedforward networks 26
Capabilities of feedforward learning algorithms 30
2.3.3.2 The Hopfield Model 35
2.3.3.3 Recurrent Networks 38
Boltzmann machines 38
Recurrent Backpropagation 40
2.3.3.4 Learning Vector Quantization (LVQ) 41
2.3.3.5 Unsupervised Networks 43

2.4 Summary 45

CHAPTER 3 - ANALYTICAL AND EXPERIMENTAL METHOD 47

3.0 Introduction 47

3.1 Proposed Experimental Method 48
3.1.1 Model Selection Knowledge base - form and function 50
3.1.2 Data set characteristics - purpose and form 51

3.2 Example of method using synthetic data sets 52

3.3 Expected benefits and drawbacks of a structured method 54

3.4 Summary 56

CHAPTER 4 - FORECASTING 57

4.0 Introduction 57

4.1 The forecasting process 58
4.1.1 Data set characteristics 59
4.1.2 Forecasting performance metrics 61
4.1.3 Forecasting algorithm characteristics 62

4.2 Forecasting methods - A critical review 62
4.2.1 Established forecasting techniques 63
4.2.1.1 Linear time series models 63
4.2.1.2 Non-linear time series models 64
4.2.2 ANNs for forecasting 65

4.3 Benchmarking studies 66

4.4 The Forecasting problem 66
4.4.1 Foreign Exchange Markets 67
4.4.1.1 Exporters, importers and foreign investors 67
4.4.1.2 Speculators 67
4.4.1.3 The market mechanisms and driving forces 68
4.4.1.4 Potential for forecasting 71

4.5 ANN forecasting experiments 74
4.5.1 Daily time series: characteristics 74
4.5.2 Daily time series : experiments and results 77
4.5.3 Five-minute time series: characteristics 78
4.5.4 Five-minute time series: experiments and results 80

4.6 Discussion of results 81

CHAPTER 5 - CLASSIFICATION 84

5.0 Introduction 84
5.0.1 Hypothesis 85

5.1 The classification process 85
5.1.1 Data set characteristic measures 85
5.1.2 Classifier performance metrics 86
5.1.2.1 Data Sampling 86
5.1.2.2 Accuracy / Error measures 88
5.1.3 Classifier model characteristics 89

5.2 Methods of classification 89
5.2.1 Statistical Methods 90
Linear, Quadratic and Logistic Discriminant Functions 90
Bayes Rule (Non-Parametric) 91
Multivariate Density Estimation 93
k-Nearest Neighbour 94
5.2.2 Rule Structured Classifiers (Machine Learning) 95
Decision Trees 96
CART (Classification And Regression Tree) 96
5.2.3 Artificial Neural Networks 96

Feedforward Network 96
Learning Vector Quantization (LVQ) 97

5.3 Benchmarking studies **97**

5.4 The classification problem **99**
5.4.1 Septicaemia 99
5.4.2 Data Set Features 100

5.5 Experiments **101**
5.5.1 Septicaemia data set: characteristics 101
5.5.2 Algorithms selected for experiment 102
5.5.3 Feedforward Networks: experiment results 103
5.5.4 Learning Vector Quantization (LVQ): experiment results 103

5.6 Summary **106**

CHAPTER 6 - DISCUSSION 108

6.0 Precepts **108**

6.1 Research Objectives **109**

6.2 Limits to ANN learning capabilities **111**

6.3 Problem representation and reasoning **113**

6.4 The utility of a meta-method **115**

CHAPTER 7 - CONCLUSIONS 118

7.0 The objectives **118**

7.1 Future work **120**

APPENDIX A - CHARACTERISTIC MEASURES OF DATA SETS FOR FORECASTING I

A.1 Standard deviation (SD) - σ **i**

A.2 Mean - $\bar{x}$ **i**

A.3 Stationarity **i**

A.4 Autocorrelation Function (ACF) **ii**

A.5 Histogram **ii**

A.6 Phase Diagram **ii**

A.7 Embedding Diagram **iii**

A.8 Synthetic data sets **iii**

APPENDIX B - ANALYSIS OF FORECASTING DATA IV

B.1 - The five minute closing price data iv

B.2 - The daily closing price data viii

B.3 Prediction Results (five minute data) xi

APPENDIX C - CHARACTERISTIC MEASURES OF DATA SETS FOR CLASSIFIERS XII

C.1 Sample Size xii

C.2 Features xii

C.3 Binary Features xii

C.4 Ordinal Features xii

C.5 Classes xii

C.6 Entropy of a feature xii

C.7 Joint Entropy of features and classes xiii

APPENDIX D - ANALYSIS OF CLASSIFICATION DATA XIV

D1. Analysis of feature Entropy H(f) xiv

D2 Analysis of class-feature entropy H(c,f) xiv

D.3 Entropy values for test classification data set xiv

APPENDIX E - SOFTWARE DEVELOPED TO SUPPORT RESEARCH XVII

E.1 LVQ simulation xvii

E.2 Classification analysis xx

E.3 Time series analysis xxi

E.4 Chaos identification and simulation xxiv

APPENDIX F – LVQ EXPERIMENTS XXVI

F.1 Software xxvi

F.2 Data files xxvi

F.3 Results xxvii

F.4 Experimental Data xxviii

List Of Figures

Chapter 1 1.1 Schematic of a neuron with computational abstraction

Chapter 2 2.1 Disciplines contributing to Computational ANN models

2.2 Breadth of models and academic disciplines for neural modelling

2.3 Simple & Two layer perceptron

2.4 Processing Element and example transfer functions

2.5 Weight space separation for perceptron logic functions

2.6 Weight space in Simple and Two Layer Perceptron

2.7 A small Hopfield network with four processing elements

2.8 Schematic of Hopfield Network State Space (Attractors labelled 1-5)

2.9 An arbitrarily connected Boltzmann model

2.10 Learning Vector Quantization classifier

2.11 A network entity relationship diagram.

Chapter 3 3.1 Schematic of research activity and theory

3.2 Generic method for classification and forecasting applications

3.3 Simple characteristics-algorithms mapping

3.4 Complex characteristics-algorithms mapping

3.5 Time series data; d1 (line) and d2 (sinusoid)

3.6 Mapping schematic of time series example

Chapter 4 4.1 The forecasting process

4.2 Hypothetical time series and prediction

4.3 A simple currency exchange market model

4.4 Five currency exchange rates against the US dollar

4.5 Phase diagram of Italian Lira (x vs dx/dt)

4.6 Phase plot of Italian Lira (dx/dt vs. d^2x/dt^2)

4.7 Histogram of daily Lira closing prices

4.8 ACF functions for synthetic time series

4.9 ACF functions for Daily Lira and first differential

List Of Figures (continued)

4.10 Three sample series with mean and a scaled SD (x5)

4.11 Summary of experimental method : forecasting

Chapter 5 5.1 The classification process

5.2 Synthetic 2-D data set

5.3 Class likelihood functions for classes A and B

5.4 Class A and B *posteriori* probability density functions

5.5 1-Nearest Neighbour (1-NN) & 3-Nearest Neighbour (3-NN)

5.6 Variation in class classification accuracy for LVQ solutions.

Chapter 6 6.1 Forecasting problem representation

List Of Tables

Chapter 2

2.1 A sample of ANN survey texts (by discipline)

2.2 Table of ANN network models

2.3 Exponential computational requirement to explore weight space for solutions

Chapter 4

4.1 Data set characteristic measures and visual tools

Chapter 5

5.1 Classification data set characteristic measures

5.2 Data sampling techniques

5.3 A true and false positive matrix

5.4 Characteristic measures for classification

5.5 Class and average accuracies for Best LVQ solution and 1-NN

5.6 True and False positives for 1-NN solution

Acknowledgements

The process of researching and writing a thesis proved more demanding in academic and personal terms than I imagined. I would like to take this opportunity to thank my academic mentors for their patient guidance and friends for their tolerance during the difficult gestation and eventual birth of this thesis.

Ron Summers and Ewart Carson provided guidance and encouragement which was especially welcomed in the latter stages. Andy Morrison helped in getting my many computing related problems solved. Steve, Rouhi, Syed and Graham made the academic side of life much more vibrant and I thank them for the frank contributions to my research efforts. On the (rare) occasions that their opinions were useless they were none the less endowed with wit. I would especially like to thank Richard Dybowski of St. Thomas' Hospital for his unstinting help and scholarly enthusiasm in developing models and analysing the Septicaemia data.

Friends where particularly important in enabling this thesis to emerge. Kevin and Kalok provided a roof over my head (for too long) and deserve special thanks. Tomoko put up with much procrastination and I thank her for the patience. To my Parents, who like all parents seem to believe in their offspring ill-deserved or not, I dedicate this thesis.

Disclaimer

CHAPTER 1 - INTRODUCTION

1.0 Background

This thesis describes research conducted at City University into the *application* of Artificial Neural Networks (ANNs). ANNs have been evaluated as candidate solutions to two common tasks: classification and forecasting. More specifically the ANN models considered were those that could be implemented as computer algorithms suitable for the application domains considered.

ANNs have emerged from a multi-disciplinary field of researchers attempting to understand and model biologically inspired neural systems on both the small and large scale. At the small end of the scale individual processing elements are studied in depth whilst in the large scale, networks containing many interconnected elements are simulated and behaviour analysed. The capabilities of the more mature ANN models have been explored in depth, with several being applied to domains, competing with established techniques such as machine learning, statistical methods and mathematical modelling. The relatively new field of ANN research is characterised by recent expansion in academic activity, rapid and widespread application of models and much debate over the benefits and performance of such models (not without controversy).

The motivation behind this study was to evaluate objectively the potential of ANN models in what can be termed 'real world' problems, as opposed to artificial tasks based on synthetic data. Real, rather than artificial data were used in the applications presented, since one of the perceived benefits of ANN models is the ability to cope with the noisy, complex and often high dimensional data sets found in many 'real world' problem domains.

1.1 Artificial Neural Networks

Artificial Neural Network research has grown exponentially since the mid 1980s and now stands at a level of research activity measured by at least 10 Journals and over 200 abstracts per month (INSPEC, 1994). The origins of neural network research, however, date back much further with anthologies frequently citing seminal texts such as McCulloch and Pitts (1943) and Hebb (1949). A respected collection of edited papers (Anderson & Rosenfeld , 1988) includes a landmark text dating back as far as the nineteenth century (James, 1890).

The majority of ANN models are based around the functional building blocks found in the nervous system (predominantly the brain) of animals; neurons (synonymously neurones). A schematic of a generic neuron is given in Figure 1.1. Neurons vary in the number of dendrites (incoming connections), range of their axons (outputs) and number of axon to dendrite connections (synapses). The detail of neural systems reveals far more complexity than that described here. The basic component described, however, has been abstracted into the computational domain summarised here by the title ‘Artificial Neural Networks’.

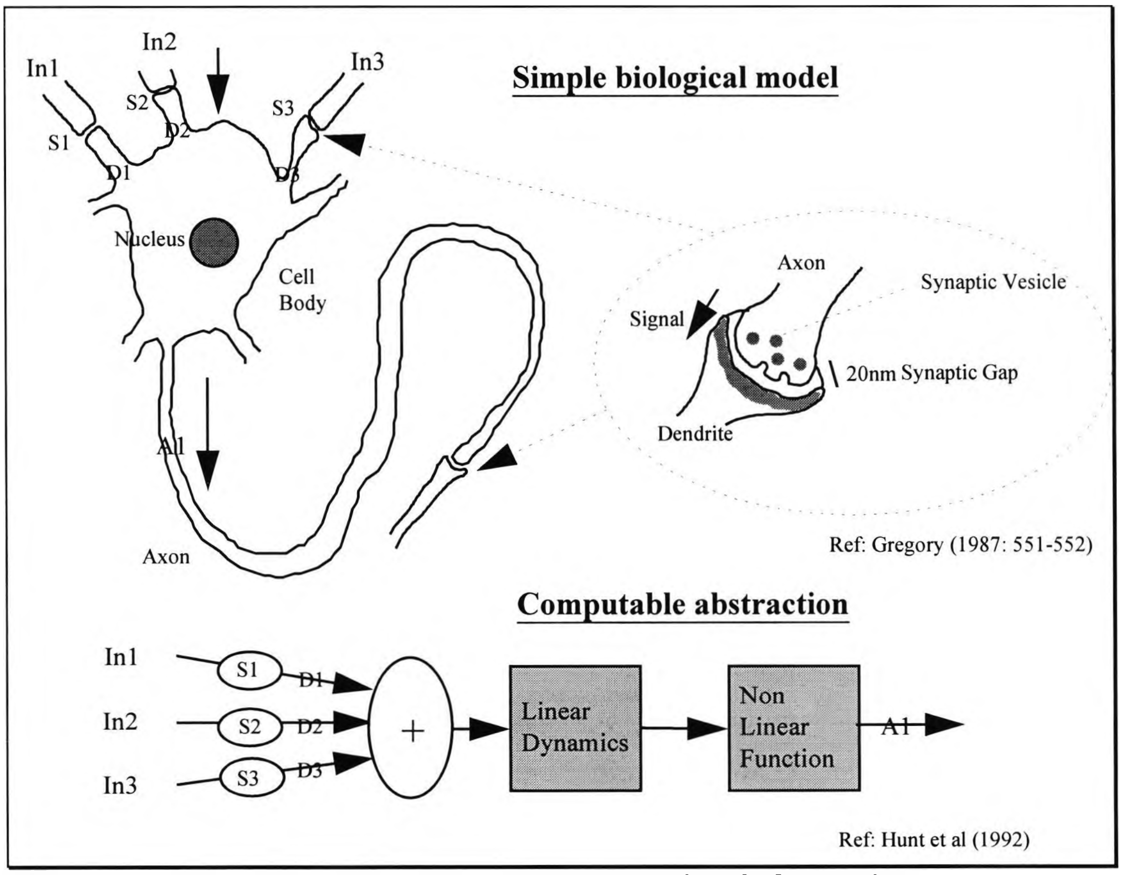


Figure 1.1 - Schematic of a neuron with computational abstraction

The recent expansion in ANN research activity is noteworthy. There are some primary causes for the resurgence:

• The breakthrough of backpropagation for training multi-layered networks
Wide scientific interest in the field waned after the computational limitations of an early ANN model, the 'perceptron' (Rosenblatt, 1958), were exposed by Minsky and Papert (1969). Further development of models received little interest from the rest of the Artificial Intelligence (AI) community. ANNs attracted huge interest again with the popularisation of the 'backpropagation' learning algorithm (Rumelhart et al., 1986). The algorithm enabled learning to take place in complex multi-layer models, which, having increased computational capability, overcame the limitations highlighted by Minsky and Papert.

• The widespread availability of low cost computing power
Most Artificial Neural Network models are computationally expensive requiring powerful computers to facilitate experimentation and analysis. With the advent of the Personal Computer, the 1980s saw the start of a huge and rapid rise in cheap computational power giving more researchers the opportunity to simulate and experiment with ANN models. Early research, limited by the dearth of such computing power, was restricted to predominantly theoretical work. A good example of this is Hebb's (1949) explicit statement of the physiological learning rule. It was not simulated on a computer until some 7 years later by Rochester et al. (1956).

• The ability of ANNs to learn input / output relationships
There is broad appeal for the use of automatic systems that are self learning and capable of generalising the relationships between data inputs and outputs, often in domains where a deterministic model of the system of interest does not exist. One of the difficulties of the rule based Artificial Intelligence (AI) approaches is the problem of generating rules from 'real world' raw data.

• Non-linear modelling capability of ANNs

Many real world modelling problems exhibit non-linear characteristics. The non-linear modelling capability found in prominent ANN models can be successfully applied to such systems.

The recent resurgence of research activity has seen more model propositions and analyses from researchers having little or no biological background. The application and modification of models for specific problem domains is a norm. Despite a significant input from those with no biological interest, neural modelling is still a driving force for many researchers. Rumelhart and the Parallel Distributed Processing (PDP) group (Rumelhart, 1986) presented a collection of cognitive process models that have served as a reference text for researchers in the field since its publication. Edelman (1992) has developed an evolutionary biological theory linking neurophysiology and the mind. Opponents and proponents of his views exist and this area of 'true' neural network research (from the biological perspective) is where academic debates rage, clouded by the long running philosophical arguments surrounding the 'mind-body' problem. Penrose (1989) has added fuel to the fire arguing that computer models of the human mind cannot come to fruition with current limitations in scientific theory.

Current ANN models are of a very small scale when compared to the human brain which contains of the order 10^{11} neurons connected by some 10^{14} interconnections (Anderson and Rosenfeld, 1988). One of the aims of modelling biological systems is to attempt to emulate desirable aspects of their behaviour such as learning. A paradox is that as the artificial models become larger, they rapidly become as difficult to analyse as their biological counterparts due to the high degree of interconnectedness.

1.2 Research Hypothesis

ANNs are self learning, potentially non-linear, data based models. They offer potential where symbolic Artificial Intelligence (AI) techniques, statistical methods or mathematical modelling have had little success. With the large (and growing) number of ANN models and algorithms to choose from, this study set out to investigate which models could be applied to real world problems and what performance they would yield. The hypothesis was :

- Artificial Neural Network models can be applied to a problem domain and offer a performance (by defined metrics) comparable to or better than established techniques commonly applied in that domain.

1.3 Research Aim

The principle aim of this study was to investigate the hypothesis by carrying out performance tests on ANNs in two very different problem domains. The performance differences between ANNs and existing techniques needed to be quantifiable thereby providing verifiable evidence either supporting or contradicting the hypothesis.

The performance of the ANNs combined with heuristic knowledge gained from applying models would lead to the conclusions as to the appropriateness of applying ANNs to the chosen problem domains. Given the range of 'competing' techniques it was valid to question the use ANNs.

1.4 Research Objectives

The research centres on two modelling problems that have already been tackled, with varying degrees of success, by a range of established techniques. The objective of the research is to devise a framework for selecting, applying and evaluating the performance of ANN models. The applicability of the framework will be demonstrated by application to two problem domains; forecasting and classification.

1.5 Application Areas

The modelling tasks used in this study are based on data sets from two domains, the first of which requires a classification solution, the second a forecasting solution.

The time series forecasting task uses foreign exchange data to evaluate the ability of a self learning Artificial Neural Network algorithm to predict future prices. There are immense volumes of data available in this domain, a factor making it appealing to the application of ANN models. The inherent difficulty of forecasting in such a domain places particular emphasis on a well defined metric of 'successful' forecasting.

The classification task is a complex problem based on a database of patient records from a London teaching hospital. The database contains details of over 5000 septicaemic episodes recorded over several years by the Department of Microbiology at St Thomas' hospital, London (Gransden et al., 1990). The task is to classify, using an appropriate algorithm, the most likely infecting micro-organism of a previously unseen patient using only past patient observations. The historical database is the only source of information with which to construct the classifier. The classifier will therefore be objective and data based as opposed to a subjective, knowledge based type.

1.4 Thesis Structure

Chapter 2 gives a broad critical review of the literature documenting Neural Network research. The review gives some history and chronology to seminal papers but focuses on the most recent computational models. A conceptual model of the research domain is presented, facilitating an explanation of the variety of terminology in texts and differing research themes.

In Chapter 3 a novel experimental and analytical method is proposed for the application of ANNs to the two domains. The method defines the application process from data set description and analysis, to proposing potentially successful model solutions, experimentation and evaluation of the model solutions, and final application of the successful solution to the domain.

The forecasting task is described in Chapter 4. A critical review of forecasting methods is followed by a discussion of time series analysis techniques. These techniques can be used to identify criteria for selecting potentially successful forecasting methods. Studies where quantitative evaluation of forecasting algorithms has been carried out are discussed before the application domain (set within the currency exchange markets) is described in detail. Finally, the forecasting experiments and their results are described and analysed.

The classification task based on the selected medical application domain is covered in Chapter 5. The initial discussion details the classification process and reviews the methods by which it can be achieved. A review of methods and benchmarking studies forms a precursor to an overview of the diagnosis and treatment of septicaemia. The data set, experimental results and analysis are presented at the close of the chapter.

A discussion exploring the utility of the experimental framework is presented in Chapter 6. The forecasting and classification results are also discussed, pulling together the strands of insight gained from the experimental process. Key issues brought to light by the application of models to the domain data are explored.

Chapter 7 summarises the experimental work and draws final conclusions on the value of ANN models in the context of classification and forecasting. Areas of further research are highlighted.

CHAPTER 2 - LITERATURE REVIEW

2.0 Introduction

Starting from the late 1980s a significant research momentum built up in the field of 'neural networks'. Computational models, inspired by neural modelling and cognitive research, caught the imagination of researchers from many domains (assisted in some instances by considerable media hyperbole). Artificial Neural Networks (ANNs), it was claimed, did not have to be programmed to perform an operation, they instead learned by examples. From such a series of examples the networks could generalise (developing relationships between inputs and outputs) hence coping with unseen examples (new situations). Machine learning algorithms had been making some headway in this direction but suffered from the limitations of their symbolic operators (AND, OR, NOT etc.). Further, machine learning algorithms are often inadequate when dealing with non-linear problems to which they are sometimes applied. It is in the non-linear arena that neural networks and the parallel field of fuzzy logic showed promise.

Currently the field of ANNs attracts contributions from many academic disciplines - primarily those of neuroscience, physics, psychology, statistics, computer science and mathematics. The broad, and historically long running philosophical debate of 'Mind-Body' (Figure 2.1) surrounds the more specialised contributions. Since the late 1980s a vast number of researchers have applied computational Artificial Neural Networks to many problems and studied theoretical issues; consequently a vast number of publications documenting the varying degrees of success have been generated. Some researchers are highly critical of the general quality of this material, sometimes forcefully put; *"much is dross"* (Ripley 1993; p.7).

The analysis within this chapter critically reviews the capabilities of ANNs using theoretical and experimental papers from the literature. The experimental papers often illustrate the practical problems encountered in trying to realise the theoretical capabilities of ANN models.

The analysis presented here covers knowledge contributed from the varied disciplines with an aim to providing a more structured method for selecting appropriate models for applications. A key problem in drawing together knowledge is finding a common language that can express concepts in each discipline. In a field where much inspiration for work comes from biological systems a huge chasm exists between the behaviour of systems on the micro (neuroscience) and the macro (cognitive) levels. Indeed it is still questioned by some whether the two are directly related or indeed if it is possible to relate them in a deterministic way (section 2.1). Mathematics, physics and biology may be useful to model the 'low level' physiology of the brain but the 'higher level' cognitive properties of the brain are difficult to understand with these deterministic models. The physiological models are very difficult to scale to the size where higher level cognitive properties may manifest themselves.

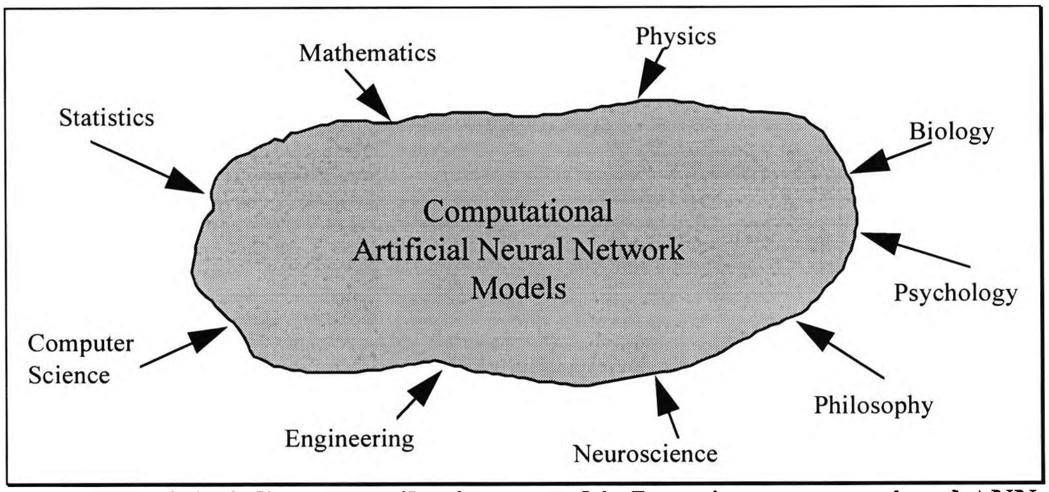


Figure 2.1 - Disciplines contributing to and influencing computational ANN models

The foundation of this thesis was research into ANNs and their potential for application to two real world problems. The numerous ANN experiments found in the literature are often problems where existing modelling techniques could also have been applied. Many of the ANN models utilised in such problems have research origins based in natural systems such as the human brain. It is this crossover in application that causes some conflict in addressing the capabilities of such models. Since many ANN applications compete with methods in the statistical / mathematical realm, analysis using techniques from these disciplines is inevitable. Using models developed in one discipline and exposing them to analytical techniques from another produces a more robust theoretical base. This analytical approach has been used before to great effect in the case of an early ANN model. Rosenblatt, originally a psychologist, introduced a model named the 'perceptron' (Rosenblatt, 1958). The perceptron was greeted with enthusiasm but it did not live up to early expectations. A rigorous mathematical analysis by Minsky and Papert (1968) revealed limitations in the computational capabilities of perceptrons. Their authoritative arguments were valid and the resulting powerful negative influence was sufficient to cause funding and general interest in ANN research to evaporate.

Now firmly re-established with a multi-disciplinary research base, the field of ANNs is producing a wide number of models. The potential for ANN models to develop into sophisticated computational machines capable of exhibiting the characteristics of the human brain is still a subject of highly opinionated debate. Penrose (1989) has addressed the subject; he argues against the possibility of computers modelling human intelligence, qualifying his argument with a discourse including Turing machines, relativity theory and quantum physics. Edelman (1992) has developed a theory of the mind based on biological evolutionary principles - a so called 'Neural Darwinism'. If the theory proves to be correct it would be the first comprehensive theory explaining and linking brain behaviour on the micro scale (neurological systems) and macro scale (higher cognitive processes and consciousness). These arguments represent what can be regarded as the higher aims of ANN research; to synthesise and understand intelligent systems using many processing elements, or groups of elements. The majority of research is, however, still concerned with the application and behaviour of small scale networks (frequently less than 100 nodes in a feedforward network). Figure 2.2 shows a range of neural concepts at different resolutions ranging from atomic scale up to the level of the human organ (brain). The scales to which ANN models correspond to their biological counterparts are also shown. The primary disciplines involved with the investigation of neural systems are represented with an approximation of the scales at which they have a bearing. It shows that computational

ANN models are on the scale of individual to small groups of cells and currently have a small scale correspondence with memory and cognition systems.

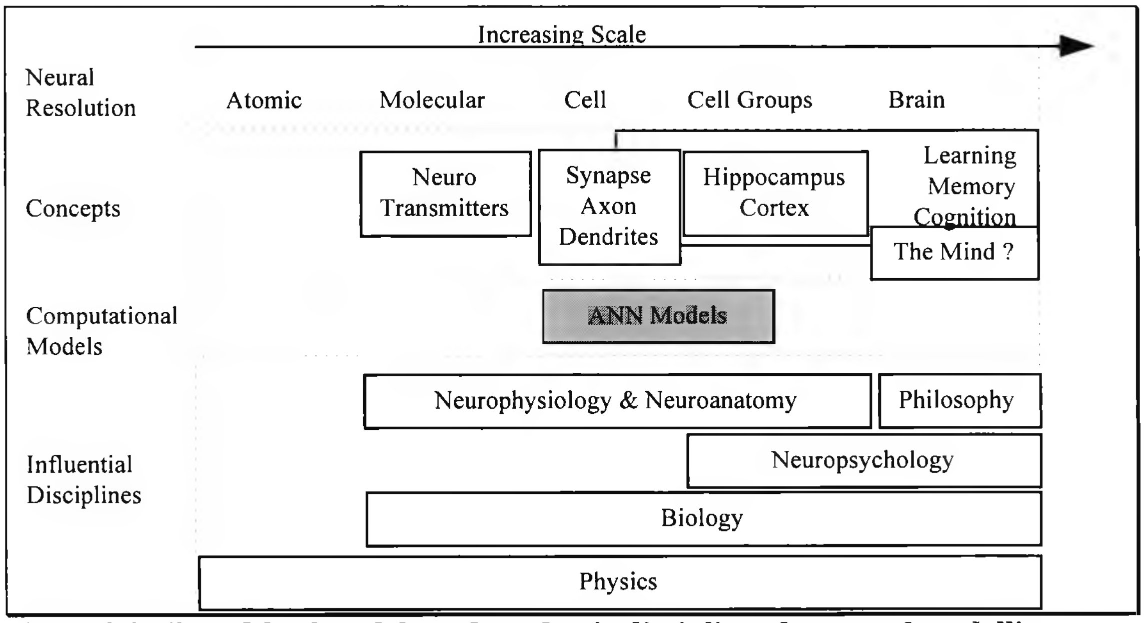


Figure 2.2 - Breadth of models and academic disciplines for neural modelling

The hope of linking small ANN models to cognitive processes is beyond the scope and not the aim of the research presented in this thesis. Ultimately, it is hoped that an ambitious theory such as Edelman's 'Theory of Neuronal Group Selection' (TNGS) will give insight (at least) into the dynamic neurological behaviour of the human brain and link this to the behaviour of the human mind. This possibility is tempered by many counter arguments like that of Penrose concluding that we cannot recreate human consciousness in Turing machines.

Edelman also realises that to reduce 'behaviour to a theory of molecular interactions is simply silly' (Edelman, 1992). This perspective is lucidly illustrated by Gleick when discussing the complexity of models ;

"Only the most naïve scientist believes that the perfect model is the one that perfectly represents reality. Such a model would have the same drawbacks as a map as large and detailed as the city it represents, a map depicting every park, every street, every building, every tree, every pothole, every inhabitant, and every map".

(Gleick, 1988; p.278)

This review is primarily concerned with small (relative to human neural networks) computational ANN models. Large scale networks and the broad issue of the human mind are discussed briefly in section 2.1. The rest of the chapter discusses the ANN models available and notes their genesis and subsequent evolution (pointing out research disciplines motivating changes in models e.g. biology, statistics etc.) and including comparisons to competitive techniques (such as machine learning). Where parallels exist attempts are made to draw upon them. It will therefore be an attempt to bind some of the literature on network models and related techniques where currently few links seem to exist.

2.1 The history of the theory of mind

A primary motivation for creating biologically plausible computational neural models is that we would hope to recreate some of the learning, cognitive and intelligent characteristics displayed by humans. This does however work on the premise that the brain (and surrounding nervous system) is the organ responsible for such functions and is the habitat of the human mind. It is an assumption widely taken for granted but has been, and still is, an area of heated philosophical debate.

Gregory (1987; p.545) discusses the history of neuropsychology and illustrates the increasingly informed viewpoints of great thinkers through the millennia starting with the ancient Greeks. Hippocrates (fifth century BC) identified the brain as the organ of intellect whereas Empedocles located mental processes in the heart. Galen also favoured the brain as the source of intellect, arguably due his experience of dealing with gladiators (in his capacity as a physician). Aristotle (400 BC) discussed mental processes with great perception whilst mistakenly placing the mental function in the heart (the brain serving merely as a cooling device!). It was with René Descartes[1] and the beginnings of modern philosophy that the dualist position was proffered. Descartes viewed the mind as a special substance separate from matter. What was termed the 'Mind-Body' problem has been a vibrant jousting arena for philosophers ever since and now has three discernible groups of theorists (though other classifications are possible);

mentalist, materialist and dualist. The mentalist theories view the mind as more definite than material objects whereas the materialists (or physicalists) view the mind as a phenomenon of physical processes, typically expressed as behaviourism (philosophical sense) or as the identity view. The dualist position holds that the mind and body are separate entities with numerous theories under this banner such as 'bundle', 'parallelist', 'interactionist', 'epiphenomalism' (Hospers, 1988). A lively picture of the current state of philosophical debate is painted by Cornwell (1994) who describes the scene of 'a dozen or so bitterly opposed factions, as well as some marauding lone rangers'. His text delves into the perceived motivations of many of the authors and puts forward the view that their work often carries a subtext. His observation of the personality cult in this area of research is an important one and can help explain (if not predict) the directions of research.

The present day debate includes participants from increasingly varied disciplines such as neurobiology, physics, computer science and philosophy. Of direct implication for the potential of computational ANN models and illustrative of the wider debate are the positions of Edelman and Penrose[2], both eminent scientists in their fields of neurobiology and physics respectively. They stand opposed in many of their views on the nature of the human mind. Edelman draws on his knowledge of biology and neuroanatomy to promote an as yet unproved Theory of Neuronal Group Selection (TNGS). This theory describes the activity of groups of neurons and links their development and interconnections with other groups to consciousness. Penrose wanders through intricate mathematical expositions of Turing machines, mathematical philosophy, classical and quantum physics and the limitations of scientific knowledge. The core argument is that a new theory (quantum gravity) to cope with the limits of quantum theory (dimensions $<10^{-33}$ cm) could explain consciousness. The relevance of his arguments to understanding the brain are questionable on the grounds of appropriate resolution of models (see Figure 2.2). Edelman recognises the scholarship of Penrose's discourse but dismisses his arguments;

[1]French philosopher (1596-1650). Famous dictum '*cogito, ergo sum*' (I think, therefore I am). His text *The Discourse on Method* caused a new 'doubting' philosophy to permeate scientific investigation with only the mind being trusted.

"Penrose's account is a bit like that of a schoolboy who, not knowing the formula of sulfuric acid asked for on an exam, gives instead a beautiful account of his dog Spot".

(Edelman, 1992; p.217)

Numerous other positions are taken in the literature. A varied sample of the many available are those of Crick (eminent biologist), Moravec (computer scientist) and Dennett (philosopher). A 'strong AI' position is taken by Moravec (1988) who presents the brain as being analogous to a digital computer (utterly at odds with Penrose). Crick (1994) takes a reductionist approach and in course has questioned Edelman's theory (Crick, 1989). Dennett (1991), a modern philosopher, presents consciousness as an illusion and draws as many critics as supporters.

The debate will undoubtedly continue, but Edelman's theory at least attempts to provide a framework linking neurophysiology to neuropsychology. More importantly the theorem's validity can be investigated by scientific enquiry. Electronic discussion groups such as that moderated by Wilken (1994) continue to oversee debate about the philosophical issues but inevitably it will be an increase in neurophysiological knowledge and simulation that will yield scientific answers to the debate. From Aristotle, Hippocrates, Empedocles to Descartes and through to modern philosophers such as Dennett the argument over mind and matter has been primarily influenced by increasing scientific knowledge in physiology, biology, neuroscience and psychology. It is hoped that a broad theory, by necessity crossing the boundaries of these disciplines, will allow the testing of scientifically verifiable hypotheses on the links between matter and mind . Edelman has developed such a theory (TNGS) and has invited for it to be tested in such a scientific way (Edelman, 1992; p.97). It would appear a laudable step.

It is interesting to note that the current ANN debate still suffers from lack of access to knowledge from participating protagonists. Anderson and Rosenfeld (1988; p.43) noted that early researchers such as McCulloch and Hebb 'really knew their neuroscience'. With the increasing spread of contributing disciplines it requires some

[2]Edelman received the Nobel Prize for Physiology or Medicine in 1972. Penrose shared with Stephen Hawking the Wolf Prize for physics for contributions to the understanding of the universe.

degree of scholarship to understand the material available and a particularly clear exposition of new ideas to reach the widest possible audience. Edelman for example shows an impressive degree of scholarship in his work but his hypotheses have been criticised for lacking clarity.

2.2 Neural Networks, Parallelism, Connectionism...Some definitions

The thesis title defines the models of interest to be 'Artificial Neural Networks'. Frequently models in the surveyed literature will appear under the inaccurate synonym 'Neural Networks'. The term artificial is used as a prefix to make clear that the models investigated are not biological neural models but computational abstractions that often draw inspiration from biological systems. They are, therefore, very much *artificial.*

Initial surveys of the literature uncovers evidence of an infant area of research lacking in clear definitions. There are still many terms that are used synonymously; 'Artificial Neural Networks', 'Neural Networks', 'Connectionist Paradigms', 'Parallel Distributed Processing'. A primary reason for this lack of definition is the number of established disciplines that are contributing research material, often written from their own perspective and in their own terminology (e.g. psychology, neuroscience, mathematics and physics).

Rumelhart et al. establish a general framework for their 'Parallel Distributed Processing' (PDP) models. They identify eight major assumptions that can be made about the components of a PDP model;

- *"A set of processing units*
- *A state of activation*
- *An output function for each unit*
- *A pattern of connectivity among units*
- *A propagation rule for propagating patterns of activities through the network of connectivities*

• An activation rule for combining the inputs impinging on a unit with the current state of that unit to produce a new level of activation for the unit.

• A learning rule whereby patterns of connectivity are modified by experience.

• An environment within which the system must operate"

(Rumelhart et al., 1986; p.44-45)

Another definition from Hecht-Nielsen can be considered:

"A neural network is a parallel, distributed information processing structure consisting of processing elements (which can possess a local memory and carry out localised information processing operations) interconnected together with unidirectional signal channels called connections. Each processing element has a single output connection which branches ('fans out') into as many collateral connections as desired (each carrying the same signal - the processing element output signal). The processing element output signal can be of any mathematical type desired. All of the processing that goes on within each processing element must be completely local: i.e. must depend only upon the current values of the input signals arriving at the processing element via impinging connections and upon values stored in the processing elements local memory.'

Hecht-Nielsen (1989)

The Hecht-Nielsen definition does not give any insight to the learning characteristics of an ANN, merely how a network should operate. The definition from Rumelhart et al. is more appealing due to its broader scope and less specificity and is therefore the working definition of ANNs adopted for this study. It should be noted that some models widely published in ANN literature (such as LVQ) do not satisfy even this broader definition of an ANN. The definition will be used, therefore, to highlight models that do not fully fit the (biologically plausible) criteria rather than to exclude.

2.3 Critical Review

Structuring a critical review of the field of Artificial Neural Networks with a meaningful chronology is difficult. The field itself has only become a widely recognised within the past decade. For example, the first conference on Neural

Information Processing Systems was held in 1988. Anthologies will, however, commonly trace the field back to McCulloch and Pitts (1943). Some go back further still to William James (1890). It can be argued that the starting point is somewhat meaningless when compared to the developments in the past decade.

This review is divided into three parts. The first part is focused on the history of ANN research, widely referred to as 'Neural Networks'. It is a brief chronological account of the important research milestones and major issues. The second part discusses useful papers that have also analysed ANN models from a similar broad perspective to that presented here. Invariably the texts are written in a language or with a focus that reflects the academic discipline of the authors. There are a few notable anthologies and critical reviews and to repeat the information that can found within them would be unproductive. It is more meaningful to summarise such texts in terms of the content and audience to which they are addressed. Those familiar with the field will recognise the difficulty of wading through the plethora of texts available offering an interpretation of ANN models in the academic language of the authors. Scholarship in the ANN field is especially important (and difficult) due to the wide ranging contributing disciplines and the numerous publications available in which to present work. The third part presents important ANN models in current use and constitutes the bulk of the review

2.3.1 - Chronology of computational ANN research

From a computational perspective , the first explicit model is universally documented to be that of McCulloch and Pitts (1943). Their model neuron is an "all-or-none" binary device. This facilitated an analysis of its computational capability using mathematical logic. The result was a proof that any finite logical expression could be implemented using McCulloch-Pitts neurons. Although the "all-or-none" physiological assumption is incorrect the paper was a positive first step towards biologically plausible computational models.

Donald Hebb provided a description of a learning rule for the modification of synapses;

> "*When an axion of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic*

change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased'. Donald Hebb (1949)

Although not a computational model it provided an explicit statement of how learning could take place in neural systems. Some simulation of this learning system was carried out by Rochester et al.(1956) but it was not until Rosenblatt's 'perceptron' (1958) that a model generated wide scale interest. The model was well defined, open to analysis and simulation, and it could respond to unseen patterns that were similar to previous patterns, thereby displaying generalization. The perceptron caused excitement as the possibilities of an adaptive, learning machine appeared to have great potential. Other models emerged in the period following this publication, notably that of Widrow and Hoff' (1960); the 'ADALINE'. This 'adaptive switching circuit' model was similar to a perceptron but had a simple learning rule that modified the gains (analogous to weights) continuously so as to minimise the output error.

These early models did not live up to the early expectations and it was the now infamous analysis performed by Minsky and Papert (1969) that precipitated the rapid demise in the research and application of such models. The text proved the limitations of the simple perceptron to solving defined logic tasks and the newer symbolic AI approaches became the focus of research.

Some researchers persevered with ANN models and Grossberg (1976) proposed several architectures with non-linear dynamics based on biological and psychological evidence. His mathematical approach was difficult to follow and broader interest in ANNs was not stimulated again until Hopfield (1982, 1984). Hopfield's academic weight led to some credibility being bestowed on ANN research and his network model (a specific type of a more general Grossberg model) rekindled wider academic interest.

The PDP group later published their seminal text containing numerous models (Rumelhart et al., 1986) one of which being the now ubiquitous 'backpropagation' algorithm. The algorithm was a generalization of the Widrow-Hoff learning rule and allowed learning in multi-layered perceptrons to solve the famous X-OR problem highlighted by Minsky and Papert (1969). The field was once again in a strong position with high expectations.

Notably, relating ANN models to theorems in physics and statistics (e.g. the Hopfield ANN model being isomorphic with the Ising model) had led to more rigorous analysis and insight. The methods of statistics and physics can be useful tools for analysing complex systems with many components - ANNs would seem to lend themselves naturally open to this type of analysis. With such analyses the field has achieved some degree of (academic) credibility in the eyes of these established disciplines. With such methods, however, the already limited biological plausibility of models can easily be lost or forgotten. The issues of model purpose, methods for analysis and development direction are important and are reflected in the current diverse strands of research pulling existing models in numerous biased scholastic directions.

2.3.2 ANN Reviews

The interdisciplinary nature of the field has lead to a range of texts each with a particular bias reflecting the author's academic discipline. The most broad ranging anthology is that of Anderson and Rosenfeld (1988). The text is an edited collection of 43 original papers from diverse sources. This popular collection was followed by a second; Anderson et al. (1990) containing 41 articles with the unifying theme being 'research directions', the aim being to map out the important directions that various researchers were taking ANN research (in 1990). The texts provide, therefore, a view back and a view forward of ANN research.

An example of a discipline specific review of the field can be found in Ripley (1991) who gives a very critical comparative assessment of some ANN models and statistical techniques. The paper is written from a statistical perspective and tackles an area of application where ANN models have been used extensively - classification. Geman et al.(1992) present a review of the learning capabilities of neural networks from a statistical view and the paper has important implications for the limitations of ANN models. Hertz et al.(1991) tackle the field from a theoretical angle and their exposition draws on their background as physicists. Not surprisingly, perhaps, the first network that they discuss is the Hopfield model (Hopfield being an eminent physicist).

A cognitive viewpoint can be found in the texts of Rumelhart et al.(1986) and McClelland et al.(1986) who can be credited with publicising the field and initiating the current resurgence in interest with the backpropagation algorithm. The books document the interdisciplinary work of the PDP group with cognitive psychologists mixing with researchers trained in physics, mathematics, neuroscience, molecular biology, and computing. The authors' aim was that the text would *'present PDP models to the community of cognitive psychologists as alternatives to the models that have dominated cognitive psychology for the past decade or so'*.

Numerous reviews of varying depth exist in addition to those already mentioned. An engineering perspective is taken in the review of ANN models given by Hush and Horne (1993). Another review with an engineering orientation is that of Hunt et al.(1992) which analyses the ANN models using notation familiar to control engineers. Only a minority of texts present ANN's from an interdiscplinary viewpoint. A sample of survey texts and their intended audience is given below in table 2.1.

Text	**Author Approach / Audience**	**Notes**
Anderson & Rosenfeld (1988)	Various / General	43 Seminal Papers
Ripley (1993)	Statistics / General	Comparative Review [175 refs.]
Michie et al.(1994)	Statistics & Machine Learning / Statistics & AI	Comprehensive comparative exercise;3 year group project. [> 260 refs.]
Hertz et al.(1991)	Physics & Statistical Mechanics / General	Very comprehensive with abundant mathematical content. [431 refs.]
Rumelhart et al.(1986)	Various / Cognitive Psychology	Multi-disciplinary Vol.1[193 refs], Vol.2[460 refs]
Hunt et al.(1992)	Control Systems / Control	An attempt to unify ANN's with existing control models[187 Refs]

Text	Author Approach / Audience	Notes
Hush and Horne (1993)	Engineering(Signal Processing) / Engineering (General)	Summary of main models, [152 refs.]

Table 2.1 - A sample of ANN Survey texts

2.3.3 - Prominent models

This section details the ANN models developed and used most commonly in applications today. The total number of models in existence is difficult to quantify as there have been numerous new developments and variations of established models published in recent years. The ANN models presented here are widely accepted as significant developments. They are presented in their simplest form with references to later developments and advances. A discussion of each model's computational capability, advantages, disadvantages and typical application (if applicable) is also given.

Table 2.2 is a chronological list of important ANN developments. The list contains key models and algorithms, primarily those that have widely accessible software implementations. Classifications are given, drawing together many found in the literature along with other less common, but useful distinctions

For each development a distinction is made as to its form; a model and/or learning algorithm. For example, much recent ANN research has been the development of algorithms for feedforward networks.

A classification of network type is given which includes the operating mode of the network; static or dynamic. A static network achieves an output from a given set of inputs by a single algorithmic pass, in the sense that there is no feedback of values in the processing. A dynamic network has some element of feedback during processing.

The common distinction of supervised/unsupervised is given for each development. Supervised networks require examples of expected output during the training phase whereas unsupervised networks create an output representation without examples.

The ‘processing’ category indicates how values are processed internally by a network. The values are considered to be either binary or continuous valued. The ability to process continuous values may be important for some applications (such as forecasting).

The uncommon ‘Biological’ distinction is given to indicate any biological basis for a development. This has been added to make apparent how many of the later developments have little biological basis.

A final distinction is made between fixed and dynamic network topologies. Most ANNs require a fixed network topology which is determined before training. There are however, dynamically configured ANNs the topology of which changes during training. Topology, as will be discussed later in this chapter, is an important factor in the computational and learning capability of an ANN.
An important issue often omitted from many discussions of ANN models is the availability of reliable, portable and easy to use software implementations of ANN models. Easy availability of such software facilitates reliable experimentation and widespread use.

A good example of such software are those of Kohonen who has made public domain implementations of the Learning Vector Quantization (LVQ) and Self Organising Map (SOM) algorithms. This has resulted in widespread experimentation and use of the algorithms and the database of published texts relating to these models (also publicly available) contains in excess of 1200 references. A strong impression remains that many proposed variations of algorithms and indeed new algorithms suffer from lack of supporting software. In a field that is based on computer simulation and investigation it seems to be a major oversight of many authors. An important tenant of scientific research in all fields is that theoretical results should be reproducible. In the context of computational models this is greatly facilitated by access to documented and validated software.

Name	Introduced	Model	Algorithm	Network Type (Operation)	Supervised	Processing	Biological	Topology	Notes	References
McCulloch & Pitts	1943	Y		**Processing Element**	Y	Bin	Y	Fixed	Neural Model	McCulloch & Pitts (1943)
Perceptron	1958	Y	Y	**Feedforward,(Static)**	Y	Bin	Y	Fixed	Neural Model,Classifier	Rosenblatt (1958)
Widrow-Hoff ADALINE	1960		Y	**Feedforward,(Static)**	Y	Bin		Fixed	Classifier	Widrow & Hoff (1960)
Backpropagation	1986		Y	**Feedforward,(Static)**	Y	Con	Y	Fixed	General Learning Algorithm	Rumelhart et al (1986)
Quickprop			Y	**Feedforward,(Static)**	Y	Con		Fixed	Quicker Version of BP	
Simulated Annealing	1989		Y	**Feedforward,(Static)**	Y	Con		Fixed	Physics analogy of cooling	Makram-Ebeid et al (1989)
Probabilistic Neural Network	1988	Y	Y	**Feedforward,(Static)**	Y	Con		Fixed		Specht (1988)
Tiling Algorithm	1989	Y	Y	**Feedforward,(Static)**	Y	Con		Dyn.		Mezard & Nadal (1989)
Upstart Algorithm	1990	Y	Y	**Feedforward,(Static)**	Y	Con		Dyn.		Frean (1990)
Cascade Correlation	1990	Y	Y	**Feedforward,(Static)**	Y	Con		Dyn.		Fahlman & Lebiere (1990)
Hopfield	1982	Y		**Hopfield,(Dynamic)**	Y	Bin	Y	Fixed		Hopfield (1982)
Hopfield	1984	Y		**Hopfield,(Dynamic)**	Y	Con	Y	Fixed		Hopfield (1984)
ART 1	1987	Y	Y	**ART,(Dynamic)**	N	Bin	Y	Fixed		Carpenter & Grossberg (1987a)
ART 2	1987	Y	Y	**ART,(Dynamic)**	N	Con	Y	Fixed		Carpenter & Grossberg (1987b)
LVQ 1,2,3 & OLVQ1	1990	Y	Y	**Vector Matching,(Static)**	Y	Con	Y	Fixed	Classification Algorithms	Kohonen (1990)
SOFM	1989	Y	Y	**Feature Mapping**	N	Con	Y	Fixed	Visualisation of high dim. data	Kohonen (1989)
Boltzmann	1983	Y	Y	**Recurrent Network, (Dynamic)**	Y	Bin		Fixed		Hinton & Sejnowski (1983)
Peterson & Anderson	1987		Y	**Recurrent Network, (Dynamic)**	Y	Con		Fixed	10-30 x faster than Boltz.	Peterson & Anderson (1987)
Recurrent Backpropagation	1987		Y	**Recurrent Network, (Dynamic)**	Y	Con		Fixed		Pineda (1987), Almeida (1987)
Counterpropagation	1987	Y	Y	**Feature Mapping,(Hybrid)**	Both	Con		Fixed		Hecht-Nielsen (1987)

Table 2.2 - Table of ANN network models

(i) The operation of a network is distinct from the training phase. A dynamic network in operation will normally achieve a stable output state (after some period of time) according to the dynamics of the network. A static network will generate the output in a predetermined algorithmic 'single pass'.
(ii) The term 'biological' refers to networks that have a strong biological basis.
(iii) 'Model' and 'Algorithm' denote that the object is a model and /or a learning algorithm.
(iv) 'Supervised' networks require presentation of the expected network output with training samples.
(v) Processing denotes the data types supported (Binary or Continuous)
(vi) The network topology can be fixed or changed dynamically by the algorithm.

2.3.3.1 Feedforward Networks

Feedforward network applications dominate the literature. They are still the most widely used 'family' of networks today. The origins of these networks can be traced back to the 'perceptron' introduced by Rosenblatt (1958). The simple perceptron shown in Figure 2.3 was the first network to be precisely defined in such a way that it could be studied mathematically and by computer simulation.

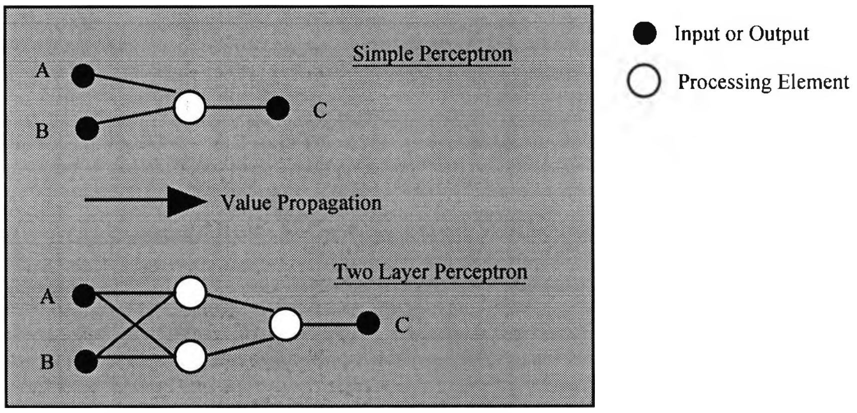


Figure 2.3 - Simple & Two Layer[3] Perceptron.

The models proposed by Rosenblatt use binary threshold processing elements. Subsequent developments yielded changes to allow continuous valued outputs and non-linear transformations of the signal. To facilitate the analysis of feedforward networks it is instructive to consider them as containing a generic processing element of the form shown in Figure 2.4.

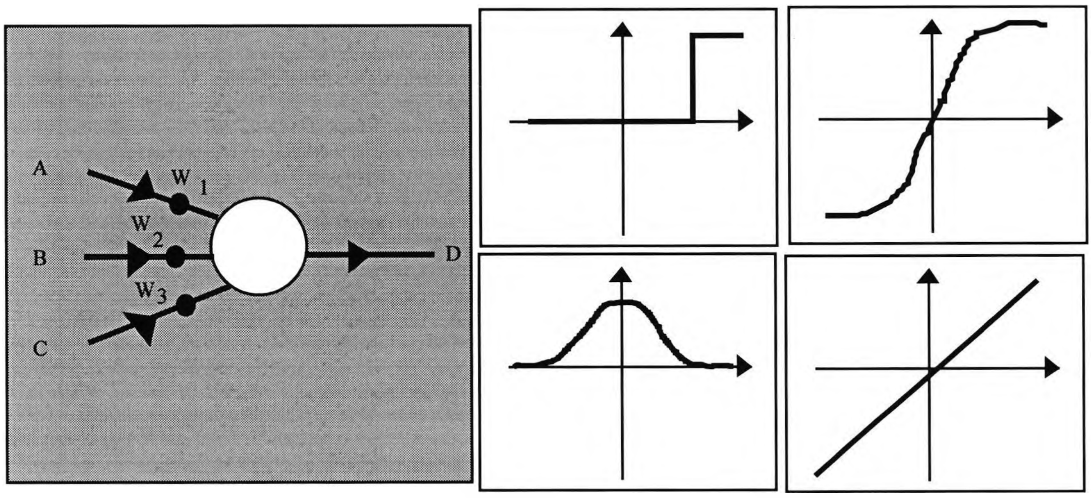


Figure 2.4 - Processing Element and example transfer functions

[3]Only layers containing processing elements are counted as layers. Many texts in the literature include non-processing input and output layers but the convention adopted here is becoming more widespread (Hertz et al, 1991; p.90).

The key stages in processing are; (i) inputs are scaled by the weights and summed; (ii) the scaled and summed value is then passed via some form of transfer function to the output. The example transfer functions shown are a threshold/step function, non-linear sigmoidal function, non-linear gaussian function and a linear function.

Sigmoid or logistic transfer function : $f(x) = \frac{1}{1+e^{-(x+\theta_j)/\theta_o}}$ (2.1)

The sigmoid function gives a smooth and differentiable response *($-1<=f(x)<= +1$)* for the full input range $\infty<x<\infty$. The property of continuous valued response over the full input range has led to wide spread use of the sigmoid transfer function in applications.
Gaussian activation function (an example of radial basis) :

$$g_j(\xi) = \frac{\exp[(\xi-\mu_j)^2/2\sigma_j^2]}{\sum_k \exp[(\xi-\mu_k)^2/2\sigma_k^2]}. \quad (2.2)$$

In this example taken from Hertz et al.(1991; p.248) ξ is the input vector and each unit j gives a maximum response to input vectors close to μ_j with size proportional to σ_j. By giving a maximum response at a given point in the input range, guassians give the processing element a localised response which can be a useful characteristic in certain applications.

Definition

A feedforward network contains one or more layers of processing elements that are interconnected via forward connections only (no feedback or intra-layer connections). This definition applies to the operating mode of the network rather than training where popular algorithms such as backpropagation feedback error values through the network.

Operating principles

The simple and multi-layer perceptron (MLP) shown in Figure 2.3 are excellent vehicles for illustrating the general operating principles of feedforward models. Feedforward models have two modes of operation:

I) The *feedforward mode,* where input signals are fed forward through the network to the output. The final output signal will be a function of the input signals which have been scaled (weights), summed and transformed (transfer functions) in many combinations. The complexity of the mapping of the output(s) to the inputs increases with the number of processing elements and number of layers.

ii) The *learning mode,* where the weights of the network are adjusted in some fashion so that the output of the network provides the desired range of output values. The weights are usually adjusted by what is termed a *learning algorithm.*

An ANN model with a feedforward *architecture* can be defined in the following terms. The ANN model will have a fixed[4] *topology* with inputs and outputs connected by *processing elements* with *weighted* interconnections. The topology of an ANN is the arrangement of processing elements and the connections between them. Each processing element will have a defined *transfer function.* The only ANN parameters that are free to change are the weights. They are adjusted by a suitable *learning algorithm* that adjusts the weights to minimise a defined *error function.*

The number of terms italicised in the above summary is indicative of the number of parameters that a feedforward ANN model solution can contain. Although, with most learning algorithms, the only free parameters in the learning mode are network weights, the fixed topology requires the experimenter to determine the number of processing elements, layers, type of transfer functions, error function and learning algorithm. This somewhat bewildering array of properties leads conveniently into a discussion of the capabilities of feedforward networks.

Computational capability of feedforward networks

As with other techniques from machine learning to statistics, a knowledge of the limitations and capabilities is essential for appropriate use. It is not surprising to find that the aims of some of the research and development of feedforward networks can be categorised into two investigative directions:

i) Investigation of what feedforward ANNs can *theoretically compute* ?

ii) Given the theoretical computational ability of a given ANN to compute a problem, investigation of the topology, network properties and learning algorithm needed to *reliably learn* a solution for that problem.

Rosenblatt's (1958) 'perceptron' introduced the first well defined computational ANN model. His analysis of its computational and learning capabilities was 'sketchy' (Anderson & Rosenfeld, 1988:91). It took another decade before a rigorous investigation documented the computational limits of these devices. Indeed it was Minsky and Papert's (1969) text 'Perceptrons' drawing attention to the *theoretical* limitations of the simple perceptron and the now infamous 'eXclusive-OR problem' (the simplest case of the more general parity problem) that caused hopes (and funding) to be dashed. It was realised at the time that the multi-layer perceptron was capable of computing complex functions but a proven learning algorithm was not available. This led to great disappointment and minimal research activity for nearly 20 years.

The simple perceptron, having a threshold output and binary inputs, is a very basic model which facilitates analysis. The simple perceptron can separate input space with a hyper plane and can therefore only solve problems that are linearly separable. It can compute many logic functions quite successfully as Figure 2.5 demonstrates. The diagram shows the 16 (2^4) binary logic combinations possible for a 2 input, 1 output perceptron. Two of the logic functions are, however, not linearly separable; the X-OR function and its complement (both denoted by '?' in Figure 2.5). The multi-layer perceptron can, however, compute this function since its division of weight space does not have to be linear. A solution to the X-OR problem is shown for a MLP with weight values set 'by hand' (Figure 2.6).

From the simple threshold function found in the perceptron, a range of non-linear, continuous transfer functions have been investigated. Several learning algorithms for networks of these types have also been proposed.

[4]This is not strictly the case. There are constructive ANN methods such as cascade correlation and the tiling algorithm which dynamically create a network topology in the learning mode. The result of the learning mode will, however, yield a fixed network topology.

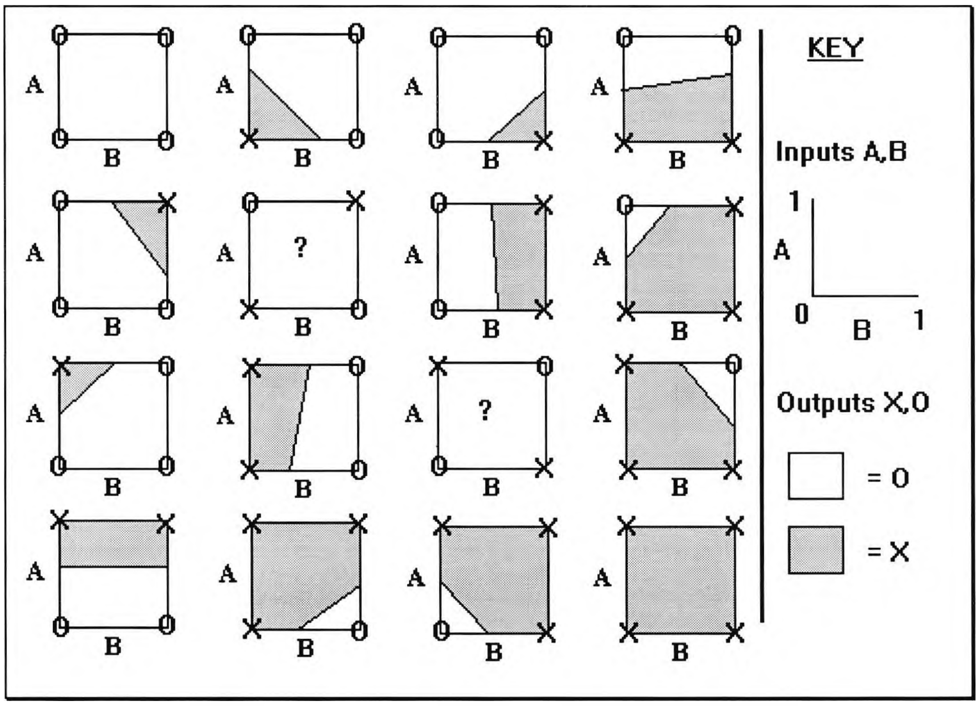
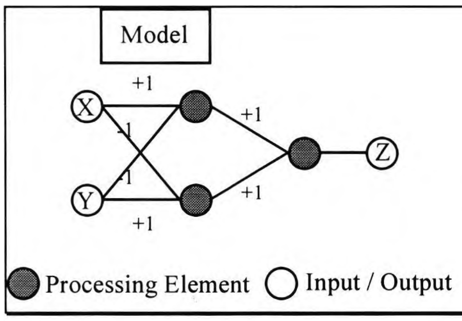


Figure 2.5 - Weight space separation for (arbitrary) solutions to perceptron logic functions



Model Function

Z=H(H(X-Y)+H(Y-X))

where H(v) is a threshold function such that for v>=0: H(v)=1 and v<0: H(v)=0

Truth Table

X Y	Z
0 0	0
0 1	1
1 0	1
1 1	0

Figure 2.6 MLP solution to the X-OR problem

Since the developments of the mid-1980s there have been further extensions of the theoretical knowledge relating to the computational capabilities of networks (some drawn from existing theorems). A popular theorem from Kolmogorov (1957) proved that any continuous function g from $[0,1]^n \rightarrow \Re^m$ can be represented exactly by a three-layer network with inputs *n* connected to each of 2*n*+1 elements in the hidden layer. Unfortunately due to the unusual form of the summation and non-linear functions it would appear to be inapplicable to the current feedforward networks. Indeed Girosi & Poggio's (1989) paper draws attention to this forcefully. An applicable proof is that of Cybenko (1989) who uses the sigmoidal function to obtain results on the approximation of continuous multivariate functions. It was found that any *continuous* function can be approximated with a network containing one layer of processing elements. Cybenko (1988) also gives a proof that a set of functions that can be approximated with at most two layers of processing elements. Hecht-Nielsen (1989) also defined that for any square integrable function there is a two layer network that approximates it to within the mean square error. The activation function is not general, but selected for a given problem.

The proofs discussed do not specify the minimum size of a network (they use approximation error tending to zero as the number of hidden units tends to infinity). This means that *theoretically* we know that all continuous functions can be represented by a two layer network but the size of the network and the ability of the available learning algorithms to find a solution set of weights remains unknown. Indeed it may be possible to use fewer nodes in a solution by using more layers, but this remains indeterminate due to a lack of theory. It is interesting to note that the constructive methods used by Frean(1990), Fahlmann & Lebiere (1990) and Refenes (1992) are a practical approach attempting to overcome the selection of an appropriate network size for a given problem. Their algorithms construct the topology dynamically during learning, starting with a minimal network architecture and appending nodes iteratively to minimise the output error until a limit is achieved. The opposite to this approach is pruning (Sietsma and Dow, 1988) where processing elements are removed if they are judged to be ineffectual. Pruning has a drawback in that the network topology has to be defined before learning and is then pruned according to given criteria. Learning speed is obviously reduced if an extremely large network is specified.

The trade off between accuracy and number of network nodes is strongly related to the issues of generalization and overfitting. The concepts of generalization and overfitting are well known in statistics (Ripley, 1993; p.42) and are discussed in the ANN context by Hertz et al.(1991; p.147-155). Generalization is a desirable feature in most ANN applications where there is noise in the data set; a trained network that can produce an appropriate output for a previously unseen (but similar) input is said to be able to generalise. If a network has a capability to learn all of its input-output associations exactly, it may well show a poor ability to generalise when presented with previously unseen input patterns. This is analogous to overfitting a curve from a noisy data set by using to high a degree polynomial fit.

The theory related to generalization and overfitting is limited. A theoretical framework described by Hertz et al.(1991; based on work by Schwartz et al., 1990) provides some interesting results. The average generalisation ability of a network is derived by projecting all possible ANN solutions to a given input-output mapping function into weight space. The weight space can be partitioned into volumes of weight values that yield solutions (within a defined error value) to different input-output problems. The concept of generalisation is that as the number of training samples increases, the volume of possible solutions will reduce. This allows an analysis of generalisation in terms of the probability of finding the correct volume of solutions in weight space. The framework is however, impractical for all but trivial problems as it requires the calculation of volumes mathematically. There is also an assumption made in the framework that as more training cases are included the volume of possible network weight solutions decreases. This assumption may not be valid with real data which is invariably noisy (which increases the variance). The issue of variance is discussed by Geman et al.(1992) in some depth.

Capabilities of feedforward learning algorithms

Rumelhart et al.(1986) led the resurgence of interest in feedforward ANNs with the backpropagation algorithm. It has been widely acknowledged that other researchers discovered this algorithm independently at various times (Bryson and Ho (1969), Werbos (1974) and Parker (1985)) but credit can be given to Rumelhart et al. for popularising this approach. Backpropagation, through iterative weight modification can adapt the weights of the multi-layer perceptron to yield a solution to the X-OR problem. It is important to note, however, that the backpropagation algorithm is not a general algorithm that can be applied to all learning tasks. Much of the existing research is aimed at discovering limitations to the algorithm and extensions that can overcome such limitations.

As an illustration of the non-trivial problem of learning in feedforward networks it is instructive to consider the crude search technique shown below.

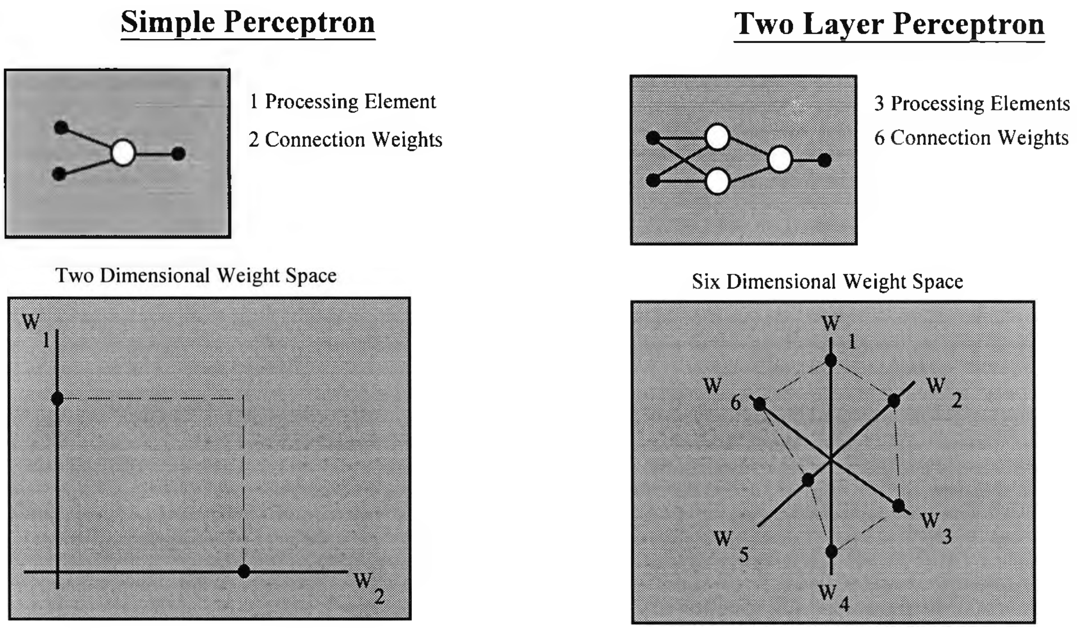


Figure 2.6 - Weight Space in Simple and Two Layer Perceptron.

Figure 2.6 demonstrates the increase in weight space dimensionality when moving from a simple two input perceptron to a two layer two input multi-layer perceptron (MLP). An output error function and test data set are defined for the network and the optimum solution set of weights is deemed to be the set yielding the lowest error for the test set. The search of weight space for a combination of weights yielding the minimum error rapidly becomes computationally expensive as a network increases in size. This computational effort is illustrated below using a crude search technique.

The search explores the possible combination of weights in the bounded weight space - $10 \le W_x \le +10$ (where *x* ranges from 1 to the number of connection weights) using a step size of 0.1. To search and calculate the error for all values of an individual weight would therefore require 200 iterations ((10--10)/0.1). This very crude search uses only a small amount of the total weight space ($-\infty < W_x < +\infty$) and has a fixed step size. Table 2.3 illustrates how computational effort to explore weight space is exponentially related to the number of weights in a network.

Connection Weights	Combinations	Iterations required for a complete, bounded search
2	200^2	40,000
3	200^3	8,000,000
4	200^4	1,600,000,000
5	2005	320,000,000,000
6	200^6	64,000,000,000,000

Table 2.3 - Exponential computational requirement to explore weight space for solutions

The efficiency of algorithms searching the weight space of networks for the combination of weights yielding minimum error is a critical limiting factor in the successful application of feedforward ANNs to problems. A range of algorithms, with numerous extensions and variations, exist. Early ANN models such as the perceptron (Rosenblatt, 1958) and the ADALINE (Widrow & Hoff, 1960) were accompanied by relatively simple learning algorithms compared to modern examples. The simple learning rule used for the perceptron and the variation of, due to Widrow and Hoff (1960), are guaranteed to learn for all cases that the perceptron is capable of computing (Block, 1962). The limitations of the architecture pointed out by Minsky and Papert (1969) forced the researchers to seek a similar general purpose learning rule for the more complex and computationally more versatile architectures containing hidden units.

The threshold transfer function used in the perceptron is a discontinuous function that cannot be differentiated (zero at all places except an infinite value at the threshold point). If we now consider network processing elements with differentiable transfer functions, the possibility of using more sophisticated learning algorithms is opened up. The linear transfer function is differentiable but a multi-layer feedforward network using such functions is exactly equivalent in computational capability to a one layer network (Hertz et al., 1991:108). Rumelhart et al. (1986:318-362) considered the use of semi-linear functions of the sigmoid type that introduce non-linearities yet are continuous and differentiable. The use of non-linear, differentiable transfer functions in feedforward networks has been the basis of the backpropagation and other sophisticated learning algorithms developed to overcome the limitations of the perceptron.

Investigation of the advances made in learning algorithms for the more complex architectures is necessarily preceded by a definition of the network error (or cost) function, *E[w]*. That is, the metric by which we judge how well a network is producing the correct output for given inputs. As the measure is of output error, the objective of any learning algorithm is to determine the set of weights yielding the minimum value of the function for a set of input/output examples (the training set). The error function for all possible network weights is often described in the literature as the error surface, having a dimension of $n+1$, where n is the number of weights in the network.

An obvious, and widely chosen error function, is the square of the difference between the actual output and desired output. This returns a positive error value tending towards zero as the weights yield a more accurate solution. Using the notation of Rumelhart et al. (1986:323) the error function for a network output with input/output pattern p can be defined as:

$$E_p = \frac{1}{2}\sum_j (t_{pj} - o_{pj})^2 \tag{2.3}$$

where there t_{pj} is the target output value of the jth unit and o_{pj} the output. The total error $E=\sum E_p$. The gradient descent method can be employed to minimise the value of *E[w]* by changing the values of the weights in the network. We therefore need to rewrite error formula (2.3) in a functional form using weight values and by implication, including the transfer functions of processing elements. First we define the output of element j as :

$$net_{pj} = \sum_i w_{ji} o_{pi} \tag{2.4}$$

where the output of a processing element is defined by the differentiable and non-decreasing function:

$$o_{pj} = f_j(net_{pj}) \tag{2.5}$$

The error function from (2.3) can be differentiated with respect the weights w_{ji} by using the chain rule as follows:

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial net_{pj}} \frac{\partial net_{pj}}{\partial w_{ji}} \tag{2.6}$$

Following the computation of derivatives and using substitution, a formulation of the error for individual processing elements can be arrived at with (2.7) being the error for an output unit and (2.8) for hidden units. These modify the weights after being scaled by a learning rate value, η. The general form being (2.9).

$$\delta_{pj} = (t_{pj} - o_{pj}) f_j'(net_{pj}) \tag{2.7}$$

$$\delta_{pj} = f_j'(net_{pj}) \sum_k \delta_{pk} w_{kj} \tag{2.8}$$

$$\Delta_p w_{ji} = \eta \delta_{pj} o_{pi} \tag{2.9}$$

Equation (2.8) is of interest since it describes the error fed to a processing element that is not directly connected to the output of the network (e.g. hidden). The error is a summation of the error values fed to the nodes in the layer closer to the output scaled by the connection weights to that layer. A more mathematically rigorous derivation of the generalised delta rule has been made by Hecht-Nielsen (1989).

A momentum term is normally used in addition to the basic equation (2.9) to reduce possible oscillations during training when using high learning rates. This is shown formally in equation (2.10) where the momentum term, α, scales the previous weight change.

$$\Delta_p w_{ji}(n+1) = \eta \delta_{pj} o_{pi} + \alpha \Delta w_{ji}(n) \tag{2.10}$$

The setting of the learning and momentum parameters (η,α) is open to experimentation but a heuristic given by Rumelhart et al. (1986:328) is $\eta \leq 0.25$. The generalised delta rule is a gradient descent technique and can be prone to local minima as was noted by Rumelhart et al. (1986:331). Local minima are a critical issue in the backpropagation algorithm as they can cause network learning to stop with a less than optimal error value.

Other algorithms have been proposed to overcome some of the difficulties experienced with the gradient descent approach (such as long descent time and local minima). A noise term can be added to the delta rule which is analogous to the annealing of solids in physics terms. The motivation for the approach is that by adding random noise, a descent that has become stuck in a local minima may be 'nudged out' by the noise. There are obvious problems of how large the noise should be and also if it should be a constant value. An entirely different approach is that of using Genetic Algorithm (GA) techniques to search the weight space for the minimum error value. The GA techniques are a method for searching the whole of weight space but are computationally expensive. The key advantage of Genetic Algorithms is that they should not suffer from local minima problems.

2.3.3.2 The Hopfield Model

In the respected collection of seminal papers edited by Anderson and Rosenfeld (1988) they comment on the impact of the Hopfield (1982) paper;

'As far as public visibility goes, the modern era in neural networks dates from the publication of this paper....John Hopfield is a distinguished physicist. When he talks people listen. Neural Networks became instantly legitimate, whereas before, most developments in networks had been the province of somewhat suspect psychologists and neurobiologists, or by those removed from the hot centres of scientific activity.'

Anderson and Rosenfeld (1988:457)

The model was conceived by the physicist, inspired by neuroanatomy. The purpose of the model was to explore the emergent properties of a system of simple interacting neurons. The Hopfield network is a dynamic network in the sense that during its operating phase it oscillates through a series of states until it becomes stable. The network is single layered and fully interconnected (see Figure 2.7). It can operate in continuous or discrete time, allowing implementation in hardware or software. The update of processing elements is asynchronous and takes place randomly in time with each element having the same update attempt rate.

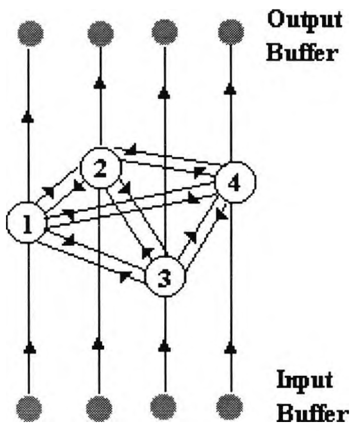


Figure 2.7 - A small Hopfield network with four processing elements

The processing elements are joined by connections with strength T_{ij} indicating a connection from element j to i. I_i is the input to a processing element from external (to the network) inputs. For each processing element i there is a fixed threshold U_i. The processing element updates its output according to the rule:

$$V_i = 1 \text{ if } \sum_{j \neq i} T_{ij} V_j + I_i > U_i \text{ and} \quad (2.11)$$

$$V_i = 0 \text{ if} \sum_{j \neq i} T_{ij} V_j + I_i < U_i \quad (2.12)$$

Hopfield also defined an energy term

$$E = -\frac{1}{2} \sum_{j \neq i} T_{ij} V_j + V_i \quad (2.13)$$

which results in any algorithm altering the output of the network nodes, ΔV_i (such as the stochastic update of nodes), decreasing the overall energy, E.

$$\Delta E = -\Delta V_i \sum_{j \neq i} T_{ij} V_j \quad (2.14)$$

The equations above relate to the threshold network first proposed by Hopfield (1982). He extended this to the continuous valued output network not long after this (Hopfield, 1984) which he considered to be more biologically plausible. The so called 'graded response' elements of this second type of network use a sigmoid transfer function:

$V_i = f\left(\sum_{j \neq i} T_{ij} V_j + I_i\right)$ where $0 \leq V_i \leq 1$ and $f(x)$ is a sigmoid type function. (2.15)

A very instructive state space schematic found in Hertz et al.(1991; p.13) is abridged below for a network with five stored patterns which form attractors (labelled 1-5). In operation, with the input values 'clamped', the network state will move in state space, attracted to one of the five stable states and eventually settle in one. The final stable state can then be taken as the output.

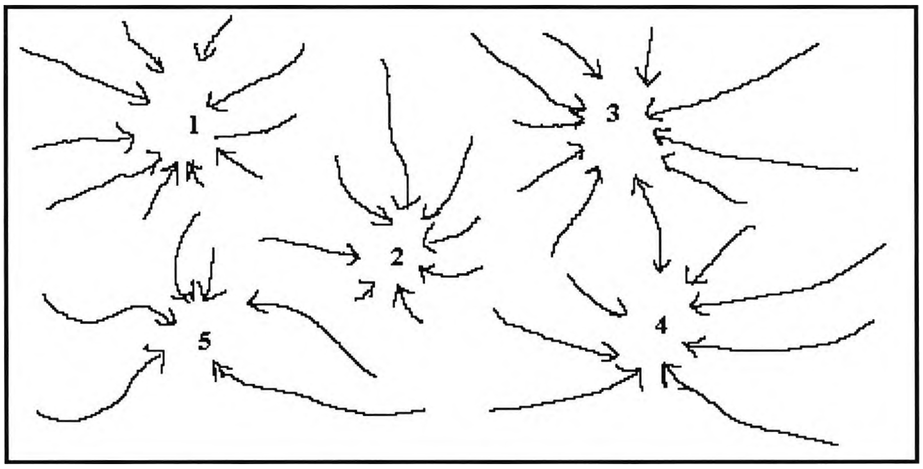


Figure 2.8 - Schematic of Hopfield Network State Space (Attractors labelled 1-5)

An important proof given the dynamic nature of the Hopfield model is that of stability. With the potentially large range of states and inputs in a Hopfield model it is necessary to ensure that a stable state will occur for a range of possible inputs. The proof of stability is derived for the network where the processing element connections are symmetric and the energy term is based on a summation of output values scaled by weights. The energy function monotonically decreases with time as modification of the weights takes place during learning. The system is isomorphic with the Ising model (spin glass) in physics. This similarity enables the weight of theory associated with the Ising model to be applied.

The learning algorithm for the Hopfield network is relatively simple in concept and consists of a Hebb (1949) type rule whereby the weights are adjusted for the patterns to be stored according to

$$\Delta T_{ij} = \left[V_i(t) V_j(t) \right]_{average} \tag{2.16}$$

The maximum number of patterns, p_{max}, that can be stored in a Hopfield network is related to the number of elements in the network N, by $p_{max}=0.138N$. A detailed exposition of this result can be found in Hertz et al.(1991; p.17-20).

The Hopfield network is not as widespread in application as the feedforward network. A probable reason for this is its mathematical complexity and demand for computational power. Hopfield specifically addressed methods of implementing his network in hardware (Hopfield, 1982) and this theme was rapidly taken up by engineers leading to the development of custom Integrated Circuits (ICs). A primary application of such networks is in image recognition and, by using an autoassociative approach, image reconstruction.

In summary, the Hopfield network benefits from the wealth of statistical mechanics theory that has been applied to it. In contrast to many other ANN models there are mature theorems detailing the capacity and stability of Hopfield models.

2.3.3.3 Recurrent Networks

Recurrent networks refers here to a class of networks with connections permitted in both directions between processing elements and even self connections (to a processing element). They are not necessarily symmetric and hence the special (Ising model) proof of stability used for the Hopfield network cannot be applied in all cases. Two important models are discussed in this section; the Boltzmann machine which can be treated as an extension of the Hopfield network and recurrent backpropagation which is an extension to the popular backpropagation algorithm.

Boltzmann machines

The term 'Boltzmann' is applied to a class of networks capable of using the learning rule defined by Ackley et al.(1985). The probability of the system states is defined by a

Boltzmann distribution (from statistical mechanics), hence the name. One of the criteria for such a network is that it must have symmetric connections between processing elements. It is therefore similar to the Hopfield network but hidden units are permitted, as shown schematically in Figure 2.9.

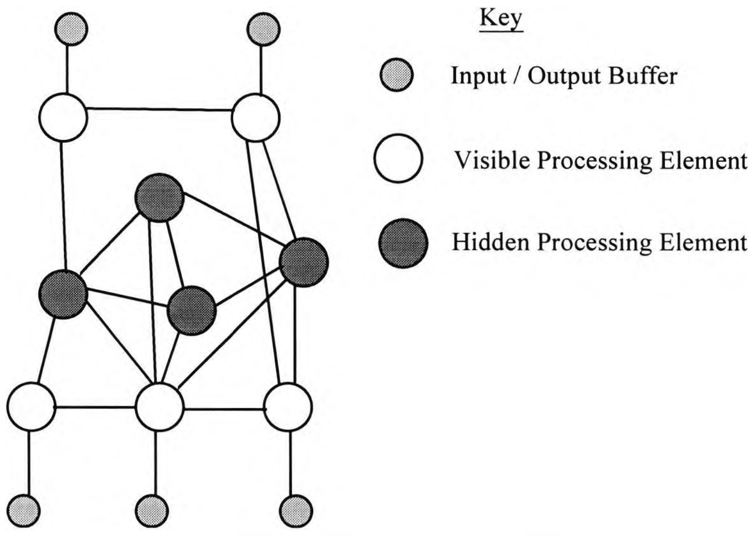


Figure 2.9 - An arbitrarily connected Boltzmann model

The network has an associated energy function which consists of a quadratic expression based on state values, weights and a threshold term:

$$E=-\sum_{i<j} w_{ij} s_i s_j + \sum_i \theta_i s_i \tag{2.17}$$

where w is the connection strength in the direction indicated. s is 1 if the processing element is in the on state and θ is a threshold. It is by the minimization of this global energy function (with input values clamped) that the network finds a stable, lower energy state. The output values of the processing elements in the stable state are the response to the initial input.

The model performs in a very similar way to the Hopfield model but with the inclusion of a probabilistic term to simulate annealing. Annealing is an analogy drawn from physical systems whereby a substance has its temperature raised (the error function is regarded as the temperature in these models) and the subsequent slow cooling should result in a minimum energy configuration (overcoming local minima). In operation the temperature value is reduced gradually until it becomes 0 and the network has the same dynamics as a Hopfield type network. The Boltzmann network stochastic processing elements (PEs) are binary with output values of +1 and -1 with a probability of

$PE_{out} = +1$ being $\dfrac{1}{1+e^{-\frac{h}{T}}}$ and (2.18)

$PE_{out} = -1$ being $1 - \dfrac{1}{1+e^{-\frac{h}{T}}}$ (2.19)

where h is the sum of the inputs to the processing element and T is the temperature. A motivation for using the Boltzmann machine is that the simulated annealing process should yield an optimum solution for a given network with time. The problem is that this is computationally expensive on serial computers. Peterson and Anderson (1987) proposed a process of 'mean field annealing' which yields a speed increase of 10 - 30 times. This is discussed in greater detail by Hertz et al.(1991; p.171-172).

Recurrent Backpropagation

Extending the popular backpropagation learning algorithm to a broader range of network architectures (where feedback and self connections are permitted) led to the development of what is termed recurrent backpropagation. Pineda (1987) explained how backpropagation could be applied to recurrent network architectures provided that learning converged to a stable state.

The learning algorithm is relatively simple (matrix inversion is not required) since it can be shown that an error propagation network of the same topology as the network can be used to update the weights. Pineda (1989) further shows that the calculation is much more efficient than matrix inversion methods for N fully connected units since calculation time is proportional to N^2 rather than N^3 for matrix inversion.

The advantage of recurrent backpropagation is that it allows greater flexibility in creating the architecture of an ANNs (nodes can be arbitrarily connected). Almeida (1987) demonstrated improvements in performance compared to normal feedforward networks in a selection of cases.

2.3.3.4 Learning Vector Quantization (LVQ)

Kohonen introduced Learning Vector Quantization in 1986 (Kohonen, 1986). More accessible accounts and extensions to the algorithm followed (Kohonen, 1988a, 1988b, 1988c, 1990). The algorithm performs a classification of input data and has been applied most notably in the area of speech recognition (Kohonen, 1990). The basic algorithm, LVQ1, has variants and complementary algorithms named LVQ2, LVQ2.1, LVQ3 and OLVQ1 which have evolved to enhance the original algorithm or overcome instability problems in training. The three principle phases in the application of LVQ are:

a) Selection of 'codebook' vectors to represent each class

b) Training of 'codebook' vectors using the LVQ algorithms so that the decision boundaries separating the classes are optimal (also dependent on the number of codebook vectors per class).

c) Classification uses the 1-Nearest Neighbour algorithm to find the nearest class representative 'codebook' vector to the input vector awaiting classification.

The algorithms are fully documented in Kohonen et al.(1992). During training, the codebook vectors are moved in feature space according to the following set of optimised-learning-rate-LVQ1 rules (OLVQ1):

m_c represents a codebook vector of class *c*.

α_c is the learning rate of the codebook vector.

x(t) is the training vector.

Rule 1: If *x(t)* is classified correctly move the codebook vector closer using

$$m_c(t+1)=m_c(t)+\alpha_c(t)[x(t)-m_c(t)] \qquad (2.20)$$

Rule 2:If *x(t)* is classified incorrectly move the codebook vector away using

$$m_c(t+1)=m_c(t)-\alpha_c(t)[x(t)-m_c(t)] \qquad (2.21)$$

Rule 3: If $i \neq c$ then do not modify the codebook vector

$$m_i(t+1)=m_i(t) \qquad (2.22)$$

The learning rate changes with time according to the following formula:

$$\alpha_c(t)=\frac{\alpha_c(t-1)}{1+s(t)\alpha_c(t-1)} \qquad (2.23)$$

where $s(t)=+1$ for correct classification and -1 for when the vector is misclassified.

The training and classification phases are illustrated in a computer generated simulation (Figure 2.10) with the (synthetic) training data set shown for reference. The two feature training set contains three classes of vector denoted by data points labelled A, B and C with each class having 10 cases in each. Before training the class representative (or 'codebook') vectors (labelled 1,2 and 3) are placed randomly in feature space. The OLVQ1 algorithm then adjusts the codebook vectors until a set number of training iterations is reached.

The classification phase requires selection of the nearest codebook vector (by the Euclidean metric) with the class it represents being the output decision. This example illustrates how the less complex decision surface formed by the class representative vectors is a reasonable generalisation of the decision surface formed by the original data points.

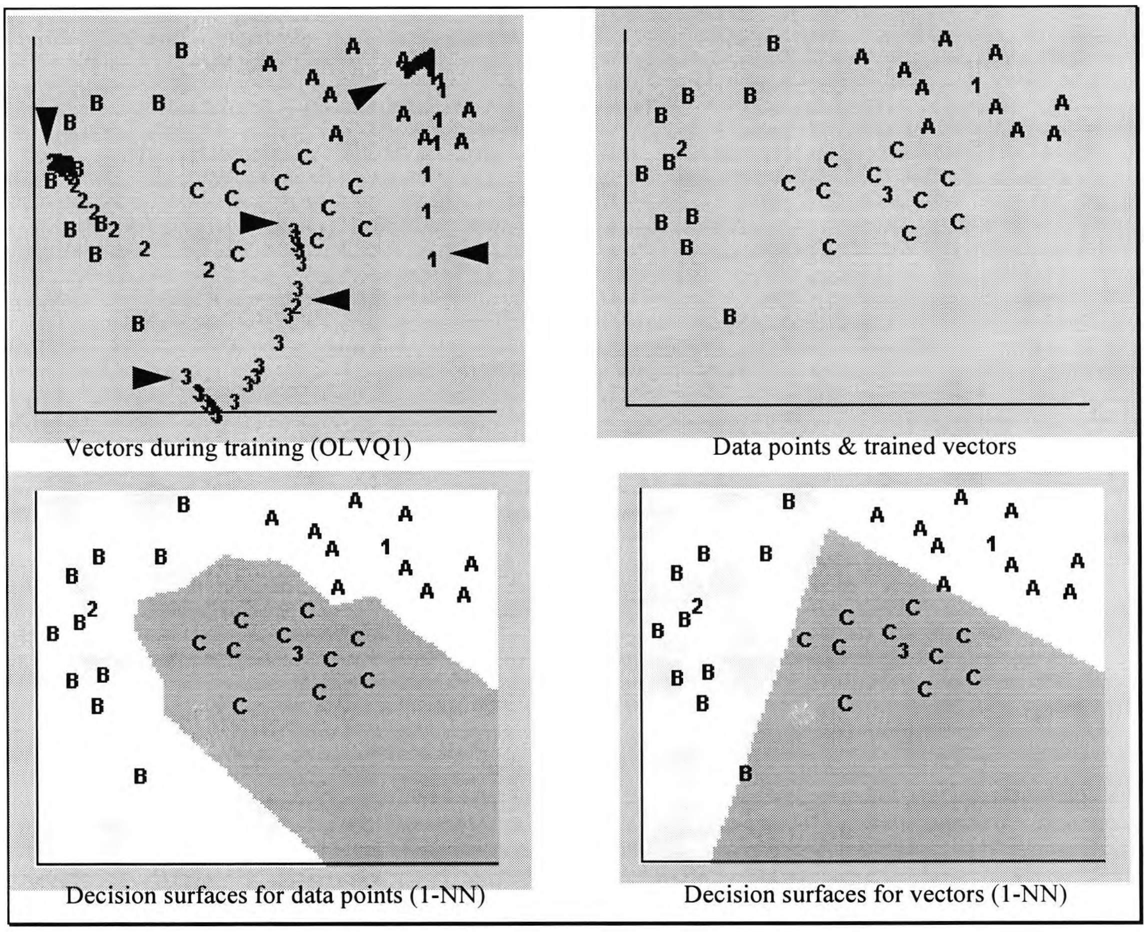


Figure 2.10 - Learning Vector Quantization classifier

Proof that the codebook vectors will converge to stable positions in feature space is important with the LVQ family of algorithms. The convergence of vectors trained by the basic LVQ1 algorithm is proved by Baras and LaVigna (1990). They note, however, that the algorithm can lead to divergence of the codebook vectors if, upon initialisation, they do not lie close to a locally asymptotic stable equilibrium. The initialisation of vectors is therefore important as convergence can only be guaranteed with vectors that lie close to the stable equilibrium points.

LVQ is included in the 'Neural Network' literature but it does not comply with the working definition of an ANN used in this thesis. It fails in that it is not a connectionist model having no set of processing units. The algorithm does appear to have great potential and is included in this review of notable algorithms due to its wide application in classification problems.

2.3.3.5 Unsupervised Networks

The discussion so far has centred on supervised learning networks. Another group of networks exists where the output of the network for a given set of inputs is not specified. The network is free to make its own representations. It is therefore not supervised in the response it makes to input data.

The problem domains presented in this thesis require known outputs for given inputs and are, therefore, supervised learning problems. Unsupervised learning is mentioned here, however, since it can be incorporated into a supervised learning model as a sub component or pre-processing of data.

A network that is permitted to generate outputs for various input patterns with no indication given to it as to the validity of those outputs would, at first sight, appear to be of little use. Indeed as Hertz et al.(1991:198) emphasize, *'unsupervised learning can only do anything useful when there is **redundancy** in the input data'*. This is more common in high dimensional problems (many inputs) where some unsupervised learning algorithms offer the possibility of reducing the redundancy by mapping the input space onto another representation (usually of a lower dimension). Given that they may aid in reducing redundancy and/or identifying features, to be of practical use they still need to be incorporated into a supervised learning architecture.

Since the networks make their own representations of the inputs, they are either used as an input pre-processing layer for a hybrid supervised network or as a separate pre-processing of the data before presenting to a supervised network. The number of layers, connectivity and learning rules in unsupervised networks are varied.
Simple one layer feedforward networks having *N* inputs and *P* outputs ($P<N$) with a modified Hebbian learning rule can be implemented to detect the *P* principle components of the inputs (Oja, 1989). Knowing the principle components of the data set allows a potentially successful reduction its dimension from *N* to *P* . The reduced dimension data set is then presented to a *P* input supervised network. It is important to note that although the network is unsupervised, its learning rule in this case means that it will attempt to learn the principle components of the data set presented to it.

Feature detection is another important use of unsupervised networks. In this instance, it is also possible to use unsupervised learning networks to effect clustering or feature mapping.

Notable models of this type are; Adaptive Resonance Theory (ART), Self Organising Feature Maps (SOFM), Willshaw and von der Marlsberg's Model and counter propagation.

Unsupervised models are not discussed in further detail as they are not to used in any experiments presented in thesis. The possible improvement of results by using unsupervised learning as part of a hybrid model or in pre-processing of data remains untested but is recognised as a potential enhancement to ANN experiments.

2.4 Summary

Artificial Neural Networks have been shown to be the result of interdisciplinary research, the vast majority of which has taken place in the past ten years. The models offer an alternative approach to classical AI methods in that they are objective, self-learning data based systems rather than subjective and knowledge based. They also offer the theoretical capability of universal computation although the trade off is all too often the interpretability of the resulting model. There are strong links between ANN models and existing methods, particularly in statistics and mathematics.

It has been shown that the current range of Artificial Neural Networks have potentially powerful computational capabilities. Feedforward ANNs in particular have well documented capabilities. It has also been shown that the theoretical computational capabilities can be difficult to achieve in practice due to the non-trivial problems of learning the correct weights and difficulties in setting model learning parameters and training issues (local minima, overfitting/generalization, speed of training). There are therefore two key issues pertinent to current ANN models and their application to problems; their *theoretical* computational capabilities and their *practical* learning capabilities. The latter issue is nebulous and far from well defined for most models. Using the feedforward ANN as an example model, Figure 2.11 illustrates the number of parameters that have to be defined for a solution to any given problem. These parameters are both architectural and algorithmic (parameters are shown in darker shading boxes). Hybrid networks introduce yet more parameters and algorithms into a single ANN model.

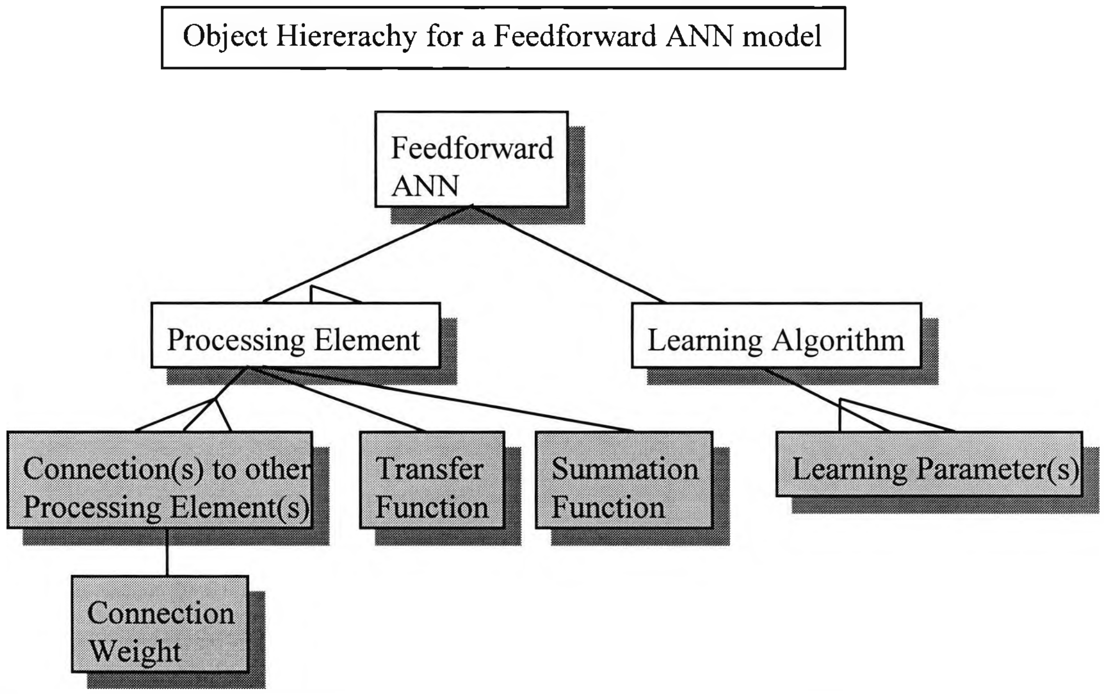


Figure 2.11 - Hierarchy of objects in a feedforward ANN model

An observation of algorithm and architecture development is that many of the parameters are becoming either better defined or set by algorithms automatically. A good example are the constructive learning algorithms such as cascade correlation. This algorithm removes the need to specify the number of nodes in a hidden layer by automatically adding nodes until the desired accuracy is reached (if possible). This should make the algorithm more consistent in application and easier to apply (given that such an algorithm has the theoretical capability to reach an optimum solution).

Other ANN models such as Kohonen's LVQ offer more transparent solutions with fewer parameters. The piecewise linear decision boundaries created by such models may yield solutions that are similar in accuracy to non-linear feedforward networks, but such comparative studies are difficult to find. The feedforward network remains the most broadly applicable type of model.

The knowledge of theoretical computational capabilities and learning capabilities will be applied to the problem domains of septicaemia and time series forecasting. The framework for analysis and experimentation in these domains is presented in chapter 3.

CHAPTER 3 - ANALYTICAL AND EXPERIMENTAL METHOD

3.0 Introduction

The novel analytical and experimental method developed for the research is discussed in this chapter. The method is generic and was applied to both domains of interest; classification and forecasting. The need for such a method arose from the findings of the literature review in chapter 2. The review drew a broad picture of the information and empirical evidence available; there were numerous texts that could be of use for any given application domain or model. The literature was characterised by:

- Large number of texts from a variety of academic disciplines
- A growing number of ANN models
- A wide variety of application oriented texts with associated results

In chapter 2 the prominent ANN models were presented alongside some of their associated theoretical capabilities. To select an appropriate model for a specific application and develop it successfully requires further empirical information about the model's performance. As was discussed in chapter 2, the theoretical results relating to ANNs give little indication of how well a model will perform in practice. The method presented in this chapter encompasses the whole application procedure and is structured to facilitate an objective, data based approach to model selection and performance evaluation.

The premise is that objective evaluation of the potential of ANNs requires measurement of performance using established metrics. Further to this, algorithms should be selected that are appropriate for a given task using objective metrics to aid the decision making wherever possible. It is therefore very important that the experimental process uses well defined metrics so that results can then be used to provide benchmarking that builds upon or facilitates comparison with that available in the literature.

The method is structured around the following process:

- The data set and model are characterised and performance metrics defined.
- The most appropriate ANN models are selected for the given task. Appropriate established algorithms are selected for comparison. Selection of all algorithms should be data based and objective where data and knowledge base information allows.
- The algorithms are applied to the domain data and the experimental results recorded.
- The performance of the ANN and established algorithms are measured using defined metrics.
- The performance results are assessed and if satisfactory the model can be applied to the domain problem.

The process is discussed in detail in the following sections.

3.1 Proposed Experimental Method

Figure 3.1 shows a conceptual model of an 'ideal' ANN model selection process in which inputs are taken from an ANN theory database, the problem domain and information on established techniques used in the domain.

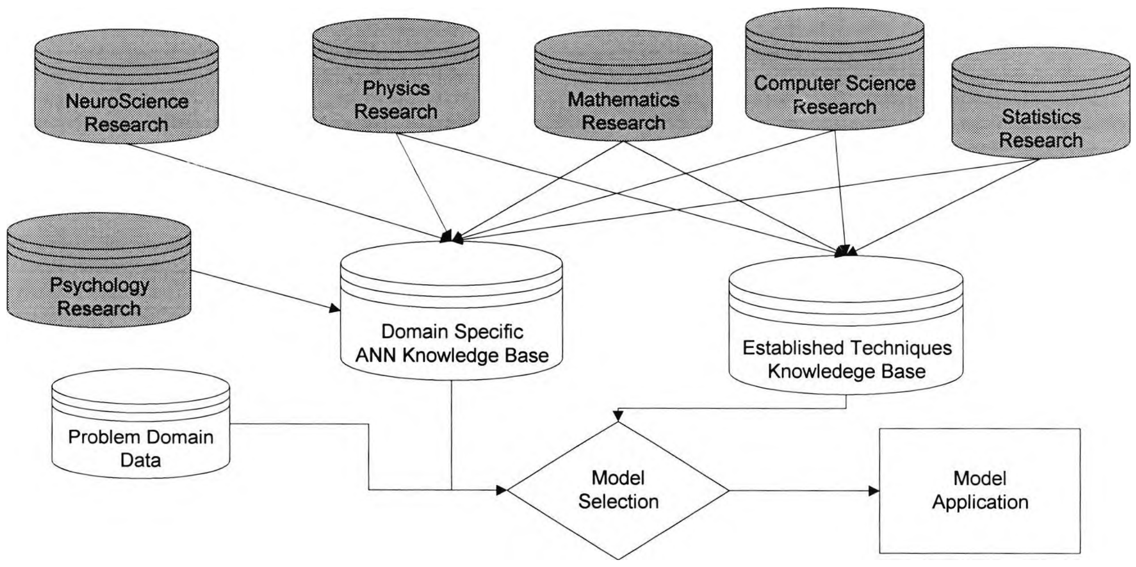


Figure 3.1 - Schematic of research activity and theory

The schema shows contributions from the varying research disciplines to the knowledge bases. The ANN knowledge base is structured to facilitate appropriate model selection for a given problem domain; forecasting or classification in this thesis. The established techniques (e.g. statistics, machine learning) knowledge base is also shown in the schema to illustrate how, together with knowledge of the problem domain, models are selected based on structured information.

Given that nearly all the published information on ANN's does not fit into an ideal framework pictured in Figure 3.1, a practical framework had to be proposed.

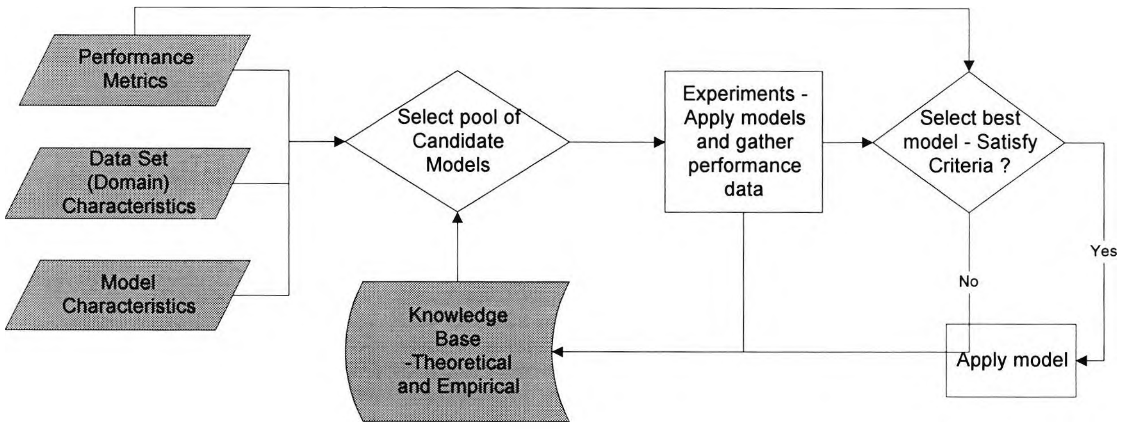

Performance Metrics
Data Set (Domain) Characteristics
Model Characteristics
Select pool of Candidate Models
Knowledge Base -Theoretical and Empirical
Experiments - Apply models and gather performance data
Select best model - Satisfy Criteria ?
Yes
No
Apply model

Figure 3.2 - Generic method for classification and forecasting applications

Figure 3.2 illustrates the novel structured method for ANN application development using a graphical representation of the experiment process. The prerequisite information required for the method are data set characteristics, model characteristics and performance metrics. These information are derived from the problem domain and are used in conjunction with the knowledge base to select suitable models for experiment. Models are tested and their performance measured using the metrics specified before the experiments begin.

Performance metrics are selected in the first stage to ensure that all models subsequently selected can be assessed using the specified metrics (a model would not be selected if its performance could not be evaluated against the metric(s)). If, after testing, the most successful model is satisfactory it can be applied to the domain. If no model is successful the process can be re-iterated with the performance of the tested models adding to the knowledge base.

In the period ('91-'93) when experiments were conducted for this research no texts were discovered that followed a comparative framework generating a knowledge base like that proposed above.

Since then Michie et al.(1994) published the results of large scale experiments which had been conducted in the classification domain using various algorithms with a range of 'real' data sets (with measured characteristics). With these data they were able to construct a basic rule based system for predicting the best classifier algorithms for a new data set using only the data set characteristics in a similar way to that proposed here.

The Statlog library data sets and algorithm results have not grown since the end of the research programme in 1994. Of the few papers trying to extend the concepts of such benchmarking, Ripley (1995) presented a set of statistical measures for selecting ANN models. Prechelt (1996), however, carried out a quantitative survey of some 400 ANN journal articles and concluded that research practice still exhibited poor benchmarking of solutions.

A more recent (and growing) benchmark collection has been introduced by the DELVE development group Rasmussen et al (1996). The group maintains an archive of data sets (including separate sets for developing and subsequent testing of algorithms) and software for statistically analysing the performance of algorithms. Importantly there is also a procedure for collecting benchmarking results from new learning methods to create a growing database of performance results.

3.1.1 Model Selection Knowledge base - form and function

The quality and breadth of the knowledge base information will dictate to what extent an objective selection of potentially successful models can be made. Knowledge bases for the experiments had to be built from the heuristic and quantitative results that could be gleaned from the literature. Where possible simple rules would be constructed from information available in the literature.

3.1.2 Data set characteristics - purpose and form

The distinction between 'real' and 'synthetic' data sets is important. Real data is defined to be that which is collected from monitored variables of a system whereas the term synthetic refers to simulated or computer generated data. The data sets used in this thesis are, therefore, real as the variables are from actual systems (medical and financial). Creating synthetic data sets enables the experimenter to test and define what characteristics data sets have (e.g. clustered, normally distributed, linear etc.). With a library of synthetic data sets each having differing characteristics it is possible to evaluate the performance of algorithms against these control data sets. By using the information gathered from the control experiments, a 'real' data set can be characterised and an algorithm selected which is known to perform well with data sets of similar character.

There will inevitably be complexity when attempting to characterise a real world data set. It would be surprising to see real world data readily exhibit, for example, a perfectly normal distribution. Typically, one would expect the data set to exhibit a *degree* of a given characteristic. The task is therefore to select characteristic measures that maximise the discrimination between data sets. The method can be viewed as a mapping process relating sets of data sets to sets of suitable algorithms. Two possible mapping schemes are illustrated in Figures 3.3 and 3.4.

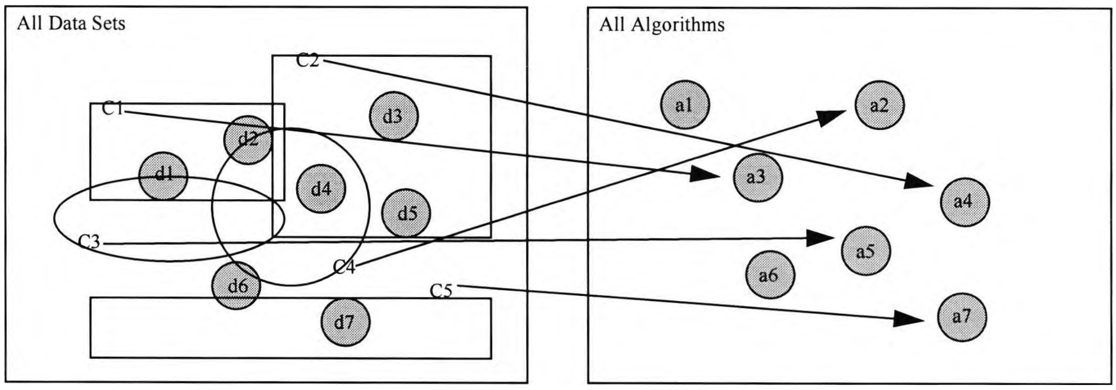


Figure 3.3 - Simple characteristics-algorithms mapping

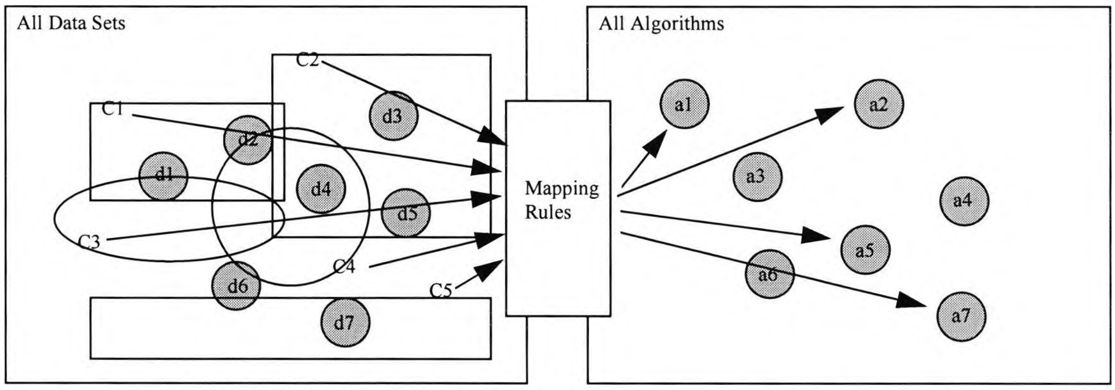


Figure 3.4 Complex characteristics-algorithms mapping

In both examples there are seven data sets (d1..d7) which are characterised by five measures (C1..C5). The data sets are subsets of the set of all possible data sets. There are seven algorithms available (a1..a7) in this example, each mapped to a data set by either one or several combinations of characteristics.

Figure 3.3 shows a simple mapping for each characteristic measure to the best algorithm for this measure. This may prove acceptable for a limited range of algorithms where the characteristic measures readily discriminate between data sets. A more sophisticated approach is illustrated in Figure 3.4 where all of the characteristic measures are used to provide a mapping. Figure 3.4 is a more likely scenario for real data where it is expected that data sets will exhibit varying degrees of most characteristics. This allows many more mappings for the same number of characteristic measures using an appropriate degree of complexity in combining the characteristic measures. Michie et al.(1994), for example, use a rules based approach for the combination of characteristics.

3.2 Example of method using synthetic data sets

To illustrate the use of the method, a simple example with models to fit time series data is described. Two synthetic, one dimensional data sets, d1 (a line) and d2 (a sinusoid), are shown in Figure 3.5. Only one data set characteristic measure, C1, is used in the example. C1 is a mean absolute deviation from the mean of the first derivative;

$$C1 = \frac{1}{n}\sum_{i=1}^{n}|x_i - \bar{x}| \tag{3.1}$$

where x_i are the differentials of the $n+1$ points in the data series with mean difference $\bar{x}$. This result is easily interpreted since, for a straight line C1=0 and for any non-linear series C1>0. The characteristic measure is therefore a simple discriminant between the two data series. The results for the time series shown below are $C1_{d1}=0$ and $C1_{d2}=0.057$. In both series n=31.

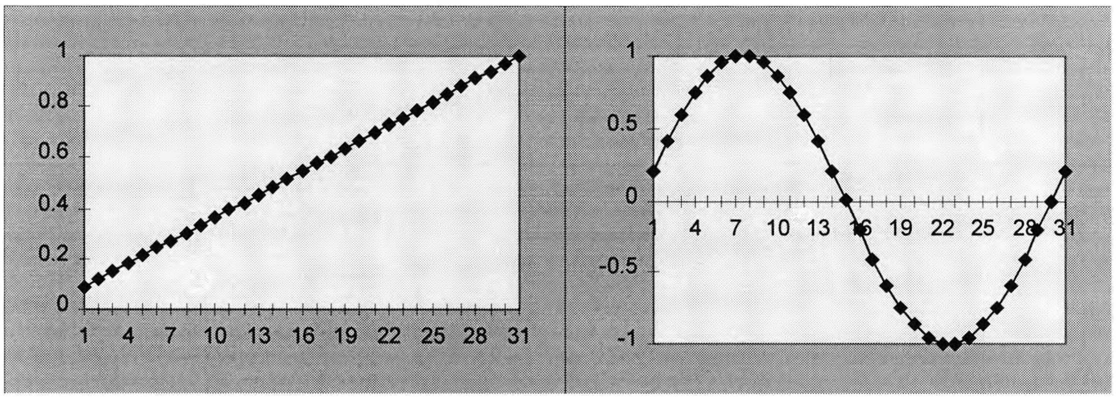


Figure 3.5 Time series data; d1 (line) and d2 (sinusoid)

In this example two algorithms are available; *a1* being linear regression and *a2* Box Jenkins. Model *a1* is appropriate to linear data sets (capable of fitting a line without error) and *a2* is widely used to model cyclical time series. A crude mapping rule can be constructed to relate the characteristic measure C1 to the algorithms *a1* and *a2*. This is shown in Figure3.6.

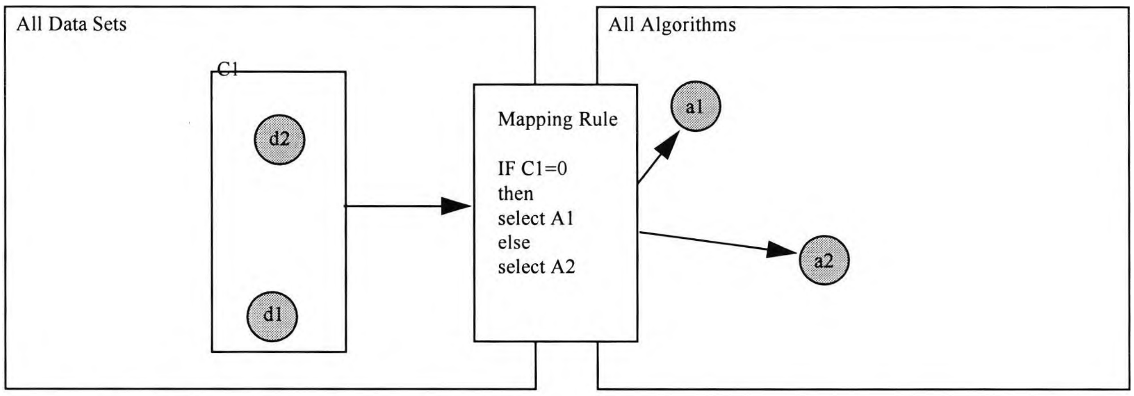


Figure 3.6 Schematic of mapping rule

The mapping rule is indeed crude since only an exact line will generate a characteristic measure of C1=0. If a third data set, d3, is introduced (Figure 3.7 - a line with added noise) we can see that the characteristic measure is now greater than 0 (C1=0.036), and the decision rule will select, in this case, an inappropriate algorithm (the Box Jenkins algorithm being suited to cyclical data).

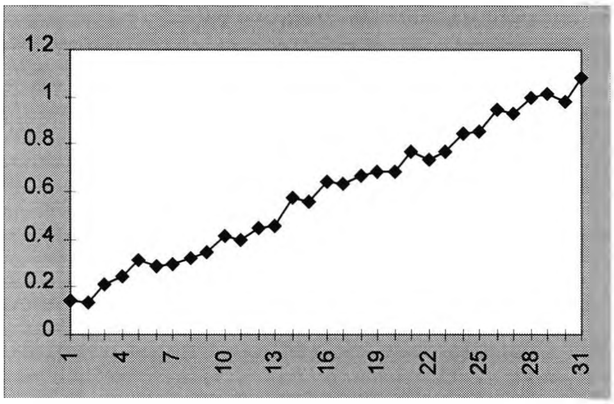


Figure 3.7 Time series d3 - (line with noise)

In this instance there are two options; modify the decision rule or introduce a new characteristic measure that is a better discriminant. By modifying the rule to *'If C1<0.04 then select a1 else select a2'*, the rule will function correctly for the control data sets.

3.3 Expected benefits and drawbacks of a structured method

There are some foreseeable problems as well as benefits to be gained by using a structured, objective analytical and experimental method. This section briefly discusses the issues which will be re-addressed in chapter 6 in light of the experimental results.

[1] Selection and maintenance of data set characteristic measures - The characteristic measures must be sufficiently discriminating between data sets to enable model selection for new data sets to be accurate. As new data sets and algorithms are added it may be necessary to add further measures and/or modify existing mapping rules.

[2] Number of data sets - As the number of discernibly different (by characteristic measures) data sets tested against all the models increases, the greater the applicability of the approach. If only a few data sets have been used, the amount of empirical evidence for selecting an appropriate algorithm may be insufficient to make a selection

decision with any degree of confidence. For some models, theoretical results may be applicable and can be translated directly into mapping rules.

[3] Computational overhead - Some data set measures may prove to be more computationally expensive than many of the ANN algorithms. This would lead to a situation where it would take less time to run the experiments than to calculate the characteristic measures used to determine the most likely experiment to succeed. For example, the calculation of joint entropy values for variables that have a wide range of vales can be computationally expensive (see appendix C for formulae). Joint entropy values have in this instance been reserved for calculations with binary variables only.

[4] Experimental Data - Each algorithm will have to be evaluated against each data set for a given performance metric. As new data sets or algorithms are added to the database, the mapping rules must be re-evaluated.

The expected benefits of this approach are :

[1] Appropriate algorithm selection - New problem data sets can be characterised and the most appropriate algorithms selected for experimentation. This utilises previous (quantitative) experimental results with algorithms that are most likely to succeed being selected.

[2] Improved algorithm selection rules - As experimental data increases the model selection rules can be refined.

[3] Objective comparison of algorithms - New algorithms can be tested over the range of data sets to evaluate performance in comparison to existing algorithms.

[4] Redundancy or significant overlap between algorithms can be identified - In the review by Michie et al. (1994) one algorithm consistently performed poorly compared to similar methods over a variety of data sets.

3.4 Summary

An analytical and experimental method has been presented that will support the research objective and test the hypotheses. The method is intended to avoid making subjective decisions wherever there is sufficient data to make an objective decision. Performance, data set and model characteristics are defined before a model is selected and are used to evaluate the experimental results. As the number of experiments and results sets increase it is expected that the utility of the algorithm selection process will increase.

CHAPTER 4 - FORECASTING

Forecasting is like driving down a winding mountain road, blindfold, with someone looking out of the rear window giving directions - Unattributed

4.0 Introduction

Forecasting events is a frequent human activity. Forecasting the behaviour of one or several variables is a common function carried out by everybody whether it be applied to a business problem, the weather, or judging the best time to beat the rush hour traffic. The ability to forecast is usually based on knowledge of the history of the variable(s) of interest and related factors.

If the history of a variable (time series) or the history of related factors is analysed we may find patterns, trends, cycles or other recognisable features. In the London rush hour traffic, for example, one would expect to see predictable peaks in the morning and evening when people arrive and depart from work. If such patterns are evident then it is possible to forecast with some confidence based on the history of the variable alone.

Some systems may not be so readily forecastable. Accurate prediction of weather behaviour requires radar, satellite and weather station information which are input to complex models requiring mainframe computing power. Despite the use of such sophisticated technology, prediction can still be dramatically erroneous as the violent storms that hit the south of England 'without warning' in October 1987 demonstrated.

A key factor in forecasting is that of unforeseen changes to the system of interest which seriously affect the forecastability of the variable. Rush hour traffic, for example, will be severely affected by a major traffic accident. In forecasting terms this is known as a shock factor. Such shock factors are often seen in financial markets and medicine.

Weigend and Gershenfeld (1993; p.2) describe three aspects of time series analysis; forecasting, modelling and characterisation. If a time series is modelled accurately then its behaviour can be described and predicted in the long term. Forecasting in contrast attempts to provide accurate descriptions of the short term behaviour. Characterisation of the time series yields properties such as degrees of freedom or randomness. An important distinction can be made:

"forecasting and modelling are not necessarily identical: finding governing equations with proper long-term properties may not be the most reliable way to determine parameters for good short-term forecasts and a model that is useful for short-term forecasts may have incorrect long-term properties".

(Weigend and Gershenfeld, 1993)

The analysis carried out in this chapter is concerned with time series forecasting; specifically discrete time series of fixed time intervals. ANN models are an attractive paradigm for forecasting since they offer good function approximation properties (see chapter 2) which appear suitable for time series modelling.

The forecasting process is described below, followed by methods for characterising time series data sets. A critical review of forecasting methods and ANNs for forecasting then follows. Benchmarking studies are introduced to enable some performance comparisons of the various techniques. Finally the application domain and data set are outlined and experimental results given.

4.1 The forecasting process

The generic experimental and analytical method outlined in chapter 3 is applied here for the forecasting process. Before any models are selected for evaluation, the data set, performance metrics and potential algorithms are characterised or defined. The process is shown below in Figure 4.1 and can be compared to Figure 3.2; a simple application specific (forecasting) overlay of terms has been employed.

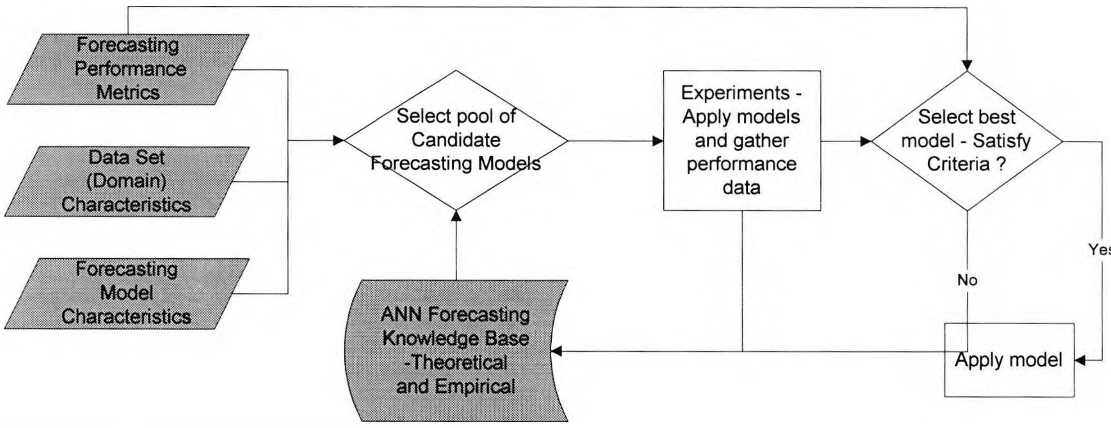

Forecasting Performance Metrics
Data Set (Domain) Characteristics
Forecasting Model Characteristics
Select pool of Candidate Forecasting Models
ANN Forecasting Knowledge Base -Theoretical and Empirical
Experiments - Apply models and gather performance data
Select best model - Satisfy Criteria ?
Yes
No
Apply model

Figure 4.1 - The forecasting process

To facilitate the discussion within this chapter it is useful to define an artificial time series data set and associated notation that can be used for exposition. Figure 4.2 shows a synthetic discrete time series with points in a chronological series x_{t-19} to x_t. A hypothetical prediction and the actual time series are shown as p_{t+1} to p_{t+5} and x_{t+1} to x_{t+5} respectively. This notation will be used throughout the remainder of this chapter.

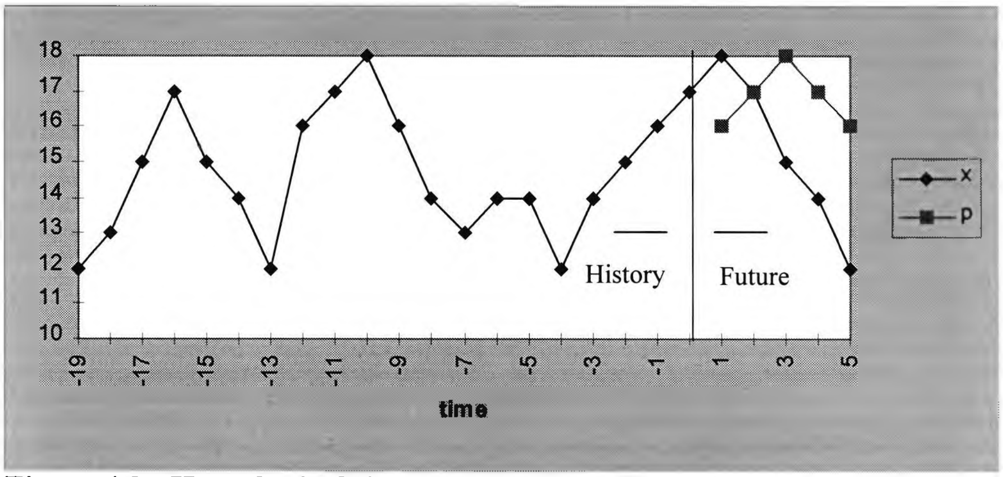


Figure 4.2 - Hypothetical time series and prediction

4.1.1 Data set characteristics

An elementary technique for characterising a time series data set is by visual inspection. Trends and cycles, for example, are often easy to identify simply by looking at the series. Quantitative and visual methods for describing and inspecting data exist, and a summary of those of potential use is presented in table 4.1 below. The measures and investigative techniques described here are appropriate for discrete data sets. The measures may be appropriate for model selection, analysis or both and are indicated as such in table 4.1 by the following notation: [A] - Analysis, [S] - model Selection, [A/S] - both Analysis and model Selection.

Measure/Technique	Symbol	Description
Standard Deviation [S]	σ	A measure of the spread of values around the mean of a time series.
Mean [S]	x	Average of time series values.
Stationarity [S]	x and σ^2 over t	If the mean and variance of a time series do not change with time the underling process is said to be stationary.
Autocorrelation Function[A/S]	$ACF(\{x_t, x_{t-\tau}\})$	A measure of how correlated a time series is with its past. For example it may be found that the autocorrelation value is high when τ is small but decays rapidly as τ increases.
Histogram [A/S]	Visual Tool	Shows the distribution and frequency of data values.
Phase diagram [A/S]	Plot (x,dx/dt) Visual Tool	A transformation of the time series into a phase space where recognisable features may emerge
Embedding diagram [A/S]	Plot $(x_t, x_{t-\tau})$ Visual Tool	A transformation of the time series into a lagged two dimensional plot.

Table 4.1 - Data set characteristic measures and visual tools

The measures and visual tools described above can be used as indicators of data set characteristics only when combined with other results (e.g. standard deviation and variance used together to investigate stationarity). There are no quantitative tools presented here that can be used unilaterally to characterise a data set and confidently select an appropriate prediction algorithm. An investigation using all available techniques gives some 'feel' for the underlying dynamics of a system and hence possibilities of selecting appropriate models for prediction. Some of the techniques are computationally intensive.

The visual tools described above offer insights into the time series by transforming the data into another state space. Phase diagrams and embedding diagrams are visual tools associated with the investigation of dynamical systems. They can be useful for detecting the existence of chaos in what, at first inspection, may appear to be a random series. The histogram is a simple yet useful visual tool for ascertaining the distribution of the time series variable. The autocorrelation function can be displayed graphically by plotting its value for a series of time lags.

The interpretation of a solitary visual tool is inadvisable but using several may give sufficient insight into a system to allow the selection of potentially successful models. In addition, functions such as autocorrelation and embedding diagrams may indicate the number of tapped delay line inputs that will be potentially useful in a forecasting model. A detailed account of the measures and visual tools described above is given in Appendix A.

4.1.2 Forecasting performance metrics

Performance metrics can be very varied. One of the simplest metrics is to evaluate how accurately a model will predict a time series *{x_t}* for *n* time intervals into the future from a point *t* in time. The accuracy of predictions can then be measured according to an error function operating on the set *{x_t,p_t}* where *{p_t}* is the predicted series. A cost function may be added to the error term (e.g. to penalise low forecasts more than high or to emphasise the accurate forecasting of turning points). Common error metrics and associated methods are detailed below. The error measure for a prediction can be defined as:

$$E=\frac{\sum_t (x_t-p_t)^2}{\sum_t (x_t)^2} \text{ where the notation is taken from Figure 4.2.} \qquad (4.1)$$

The error measure, E, takes values $0 \leq E$ with optimal prediction as $E \rightarrow 0$. E will be used as the standard error measurement.

A useful reference error measure for forecasting can be made if the series is assumed to have the random walk property. With such a series the best prediction that can be made is the last observation. The forecast provided by a model can be compared to this measure by using a ratio such as that defined below (Weigend and Gershenfeld, 1993; p.40) :

$$\text{Random Walk Ratio}=\frac{\sum_t (x_t-p_t)^2}{\sum_t (x_t-x_{t-1})^2} \qquad (4.2)$$

If the predictor under evaluation is more accurate than predicting no change (random walk), the ratio will yield a value less than 1.0. Values greater than this indicate that the predictor is worse than predicting no change. The measure is easily calculated and offers a simple way of gauging the prediction adequacy of a model under test.

A final, and for financial applications, very important measure is what will be termed here the 'Direction Correct' (DC) value. The value indicates the percentage of predictions that correctly gave the direction of movement (irrespective of magnitude). Formally the metric is defined as follows:

$$\text{Direction Correct} = \frac{\sum_{t=1}^{n} IIF(SIGN(p_t - x_t) = SIGN(x_t - x_{t-1}), 1, 0)}{n} \times 100 \qquad (4.3)$$

Optimum prediction is reached when DC = 100 and a prediction rate that is the same as chance would be DC=50.

4.1.3 Forecasting algorithm characteristics

For any given application domain there may be constraints on the type of solution that is acceptable. Such constraints can be the speed of operation of the forecasting method, the degree of explanation provided by an algorithm or implementation issues such as the memory requirements. Such *qualitative* constraints are grouped under the algorithm characteristics.

4.2 Forecasting methods - A critical review

Since forecasting is so widely applied (in domains such as business, finance, medicine etc.) there are numerous computational methods in existence. The techniques discussed here are restricted to the case of univariate forecasting and are split between what are termed here 'established techniques' and 'ANN models'.

Time series problems frequently use a representation technique known in engineering terms as a 'tapped delay line' (Weigend et al., 1992; p.397). It is simply a way of sampling the time series at set points which can then be used by a model to predict a future value. Using the notation associated with Figure 4.2, a tapped delay line of five consecutive points from x_{t-1} would be the set $\{x_{t-1}, x_{t-2} ... x_{t-5}\}$. This technique is also known as ‘state space reconstruction’ in physics and, ‘embedding’ in the dynamical systems literature. It is also possible to ‘compress’ the previous values of the time series using averaging techniques or to differentiate the time series to remove trends. Some of these methods will be discussed in this section.

4.2.1 Established forecasting techniques

The techniques discussed here are split into linear and non-linear forms. The term linear refers to the linear combination of parameters in which there are no second or higher order terms within the models. The non-linear models can be regarded as pseudo linear or piece-wise linear. They yield non-linear solutions by combining multiple local linear models.

4.2.1.1 Linear time series models

The two main families of linear time series models are moving average (MA) and auto regressive (AR) models. In moving average models a sample of the previous *N* points from a time series (a tapped delay line) are scaled linearly and combined to predict the next point in time. The model is therefore appropriate for trending a time series. Finite Impulse Response (FIR) filter is the term often attached to this model since an impulse in a time series is guaranteed to decay to zero after *N* time steps. Tapped delay line approaches require that the time series is stationary (Weigend et al., 1992; p.398). Formally using the notation given in example 4.2 we have for a MA(N) model:

$$p_{t+1}=\sum_{i=0}^{N}a_i x_{t-i} \qquad (3)$$

MA(N) models are widely used in currency markets. Pairs of MA(N) models (short and long lag) are used as buy and sell indicators. As the trend changes the shorter lag MA (with smaller N) will change direction more quickly, resulting in a crossing of the two functions which signals a buy/sell situation dependent on the direction of the movement.

Auto Regressive models use the previous predictions of the model to predict the next point. Another additive term is included to allow the input of data. Again, using the notation associated with Figure 4.2 we obtain:

$$p_{t+1}=\sum_{i=1}^{N}a_i p_{t-i}+x_t \qquad (4)$$

This model will not necessarily decay to a zero output after an impulse input (e.g. setting term x_t=1 for one iteration). The model is therefore often referred to as an infinite impulse response (IIR) filter. This model has internal memory whereas the moving average model has external memory in the form of the N tapped delay line points.

Fitting a linear model to a given time series requires the appropriate evaluation of the model parameters (a_i coefficients in each of the models described above). Box and Jenkins (1976) describe methods for calculating the coefficients of MA and AR models. Implementations of the algorithms are widely available in statistical packages such as MINITAB.

4.2.1.2 Non-linear time series models

Non-linear time series models described in texts are frequently composed of numerous localised linear models, thereby giving a global non-linear behaviour. The text by Weigend and Gershenfeld (1993) presents the best[1] modelling solutions to a range of time series problems and discusses the approaches available. The discussion of non-linear models is restricted to *only* those using multiple localised linear models such as the threshold autoregressive model (TAR) suggested by Tong and Lim (1980).

The Multivariate Adaptive Regression Splines (MARS) model introduced by Friedman (1991) , can be adapted for univariate time series modelling and has strong links to the Classification and Regression Tree (CART) technique used in classification problems. It reflects a representation of the forecasting problem as a classification problem. The univariate time series with a tapped delay line of *n* points and a predictor value of *p* can

[1]The best models emerged from an open competition where competitors were given six data sets and the freedom to apply the forecasting approach of their choice.

be formulated as an n dimensional feature space in which values of p can be categorised by thresholding. The CART system would yield discontinuities which may be unacceptable in forecasting and these issues are addressed by the MARS methodology which allows overlapping of regions. The ASTAR models use the MARS approach for a univariate series with a tapped delay line (Lewis and Stevens, 1991) and have in one study offered a performance comparable to an ANN approach (Weigend et al., 1992).

Non-linear systems theory and specifically 'chaos' (Gleick, 1987) have recently exerted much influence on forecasting. Currently the paradigm offers insights into non-linear systems using tools such as phase diagrams, embedding diagrams and fractal dimension measures. Constructing non-linear, deterministic models of real world systems is , however, notoriously difficult and invariably impractical for systems where the degrees of freedom are large (e.g. greater than 3 can be regarded as a pragmatic limit in finance, Economist, 1993; p.17).

4.2.2 ANNs for forecasting

There are numerous papers documenting the application of ANN models to a wide variety of predictive tasks using time series data. White (1988) outlines a stock price predictor of time series which was found to be better than linear regression but no better than chance on everything but the training set. Friesleben (1992) describes stock market prediction (the German FAZ-Index) using variants of the backpropagation model with multivariate data.

The tapped delay line approach is widely used but it is not the only technique available for building a memory capability into a network. Two notable approaches are exponential trace memory and gamma memory described by Mozer (Weigend and Gershenfeld, 1993).

The most widely applied model is the feedforward network using a tapped delay line approach. It has been used by White (1988) to attempt to predict share prices and has been proposed in a variety of forecasting applications before and since(e.g. Kimoto et al., 1990; Freisleben, 1992). The recurrent neural network model has emerged in recent years as an alternative approach to embedding time series memory into a network. The recurrent form can be represented by a large feedforward model (Pineda, 1987). Mozer (Weigend and Gershenfeld, 1993:256) draws attention to research indicating that the standard recurrent architecture is inadequate for some relatively simple temporal processing and forecasting tasks.

A constructive learning approach has been proposed by Refenes (1991) which yields a feedforward network solution by starting with a minimal configuration and increasing the number of processing elements until a specified output accuracy is achieved.

4.3 Benchmarking studies

The literature is littered with papers describing the performance of old, modified and novel algorithms on various data sets. There are however few studies that utilise a structured comparative method advocated in chapter 3; contrasting the performance of forecasting methods across a range of data sets.

Weigend and Gershenfeld (1993) describe the results of a forecasting competition based around six time series data sets. In each of the six cases, the algorithm performing best of those submitted is described. A breakdown of solutions is offered, investigating prediction and characterisation of the data sets. The competition was open to all and was carried out blind (the continuation of the time series were released after the competition closed). All but one of the data sets were 'real world', taken from finance, medicine, physics, astronomy and music. The most successful technique for predicting the currency exchange data set was a neural network model, although its 'success' was modest.

4.4 The Forecasting problem

The forecasting problem used in this study is taken from the domain of financial markets; specifically the currency exchange markets. The foreign exchange market trades money in huge quantities (frequently referred to as 'volume'). A central bank survey in 1989 estimated that the daily net turnover was $650 billion. A more recent estimate valued daily foreign exchange turnover (including derivatives) at about $900 billion (Economist, 1992a) which at the time was a mere $50 billion short of the total reserves of all IMF members. The quantity traded daily is now so large that the central banks, even working in unison, do not have the reserves to maintain a currency value against prevailing market forces. This was dramatically illustrated during a recent exchange rate crisis when European banks attempted to maintain the British pound at a value against market forces. One (in)famous speculator (George Soros) is estimated to have made $1 billion from the attempt by central banks to prevent the slide of the pound.

4.4.1 Foreign Exchange Markets

A vast number of time series are generated from currency markets and many forecasting studies have been made. The markets and their participants, the motivations for forecasting and the data available are discussed in this section. There are three readily identifiable groups that buy and sell currencies; exporters/importers, foreign investors and speculators (Copeland, 1989; p.10).

4.4.1.1 Exporters, importers and foreign investors

Individuals or companies wishing to buy or sell goods in a foreign country will invariably require foreign currency for the transaction. Similarly foreign investment also normally requires foreign currency. Currency for these transactions will normally be purchased from a bank. Some of the larger international companies have found it viable to operate their own foreign exchange dealing operation (e.g. British Petroleum, Allied Lyons). The time scale and quantity of currency conversions vary, with a range of currency exchange instruments existing to support business needs. It is possible, for example, to hedge exchange risk by purchasing options to buy a currency at a future date at an agreed price.

4.4.1.2 Speculators

Speculators are by far the largest group of traders in foreign currency and they attempt to make money from the process of buying and selling foreign currencies. Included in this definition of speculative trading is the capital movement by such entities as pension-funds. Estimates vary as to the speculative percentage of the total foreign exchange volume traded, but all estimates are consistently over 90% (Economist, 1992a). The speculative aspect of trading and its financial reward often draws criticism but is widely considered a necessary feature to ensure that the markets are as ‘efficient’ and ‘fluid’ as possible. It is the speculative group of traders that is of interest in this thesis.

4.4.1.3 The market mechanisms and driving forces

The motivation for forecasting foreign exchange is enormous. For speculative traders successful prediction of currency movements will allow profitable ‘positions’ to be taken. A position is taken when a dealer buys or sells enough currency so that his net balance will be positive or negative, respectively, in a given currency. If a dealer starts in a position of holding no Yen, for example, buying ten million US dollars worth of Yen would be taking a ‘long’ position in Yen. Similarly selling ten million dollars worth of Yen would be termed a ‘short’ position. Dealers taking such positions are relying on their predictions of a future upward or downward currency movement (relative to another currency) to make a profit in the price difference. It is possible to consider exchanges between multiple currencies but this study is restricted to currency pairs.

The operating principles of the traders described above apply to the relatively simple ‘spot’ currency market where the transaction is effectively completed at the time of the trade. More complex derivative financial instruments such as ‘options’ and ‘futures’ are available which allow the transaction time to be spread or delayed and therefore require more complex pricing strategies and analysis. There are numerous other exchanges that deal in a myriad of instruments but the underlying principle of all is that participants exchange objects for other objects in some ratio. The objects can be can be, for example, currency, coffee, gold or pork belly, and the ratio is usually called a price. The economic concepts determining pricing in these markets is ‘supply and demand’ (Begg et al., 1984; p.44). The variation in the supply of, and demand for, objects causes a change in the equilibrium ratio (price) for which they are exchanged. Another concept from economics, the efficient markets hypothesis (EMH) (Begg et al., 1984;p.313) also has important implications for the prediction of prices in these markets. If the market is efficient, the hypothesis states that the price of objects will reflect all 'currently available' information. Only unforeseen information will yield a change in the price. If this theory holds, then the market price should follow a 'random walk'; tomorrow's price change cannot be predicted by today's (or previous day's) price. The basic hypothesis comes in further forms known as the 'weak', 'semi-strong' and 'strong' hypotheses, which attempt to categorise the degree to which information is available to participants in a market contrasting with the simplistic assumption of the basic hypothesis that information is open to all. Empirical evidence alluding to the nature of currency markets is conflicting (Economist, 1993;p.6). If a market is assumed to be efficient, the hypothesis can be used to provide a simple measure of accuracy of any forecast by using the random walk prediction as a base line measure (see performance metrics - 4.1.2).

One factor that all forms of the EMH ignore is the *time* taken for information dissemination to market participants. In a heavily traded spot currency pair such as the dollar-deutshemark, price updates can frequently be received at 8 times per second. A reasonable hypothesis is that some market participants will have fast and efficient mechanisms to receive information and act upon it more quickly than others. Increasing speed of information processing and decision making therefore remains a way in which to 'beat' the market. The investment in computer networks, real time news feeds and analysis/charting software made by treasury dealing rooms is evidence to support this assertion.

Currently three routes exist that enable currency dealers to 'spot' exchange; direct transactions, electronic exchanges and broking services (Figure 4.3). Direct dealing is common between large merchant banks, where one dealer contacts a counterpart directly by telephone. The predominant electronic exchange system used in the London market is operated by Reuters ('Dealing 2000/2'). Numerous broking houses also offer a currency exchange facility (along with additional services). It is difficult to estimate the quantity of direct interbank trading but it is believed to be larger than the other mechanisms. Of the other two mechanisms, at present larger volumes are traded via broking services but the quantities traded electronically are increasing. Brokers and electronic exchange providers earn money for each currency transaction they make whereas speculative dealers earn (or lose) money by taking many currency positions (over time) in a market with a moving exchange rate. A 'flat' market (little or no change in exchange rates) lacks money making opportunities for speculative traders whereas fluctuating market conditions bring opportunities for taking speculative positions.

The currency flow through the market is shown in Figure 4.3. The diagram shows the core market participants who are required to make the exchange of currency possible with the dealers who 'drive' the currency movements. A currency exchange between two dealers takes place, facilitated by the market. The market is the nebulous grouping of mechanisms that allow the transactions to take place and is constantly evolving, driven by technology and business needs. The pricing mechanism, which is of prime importance is discussed next.

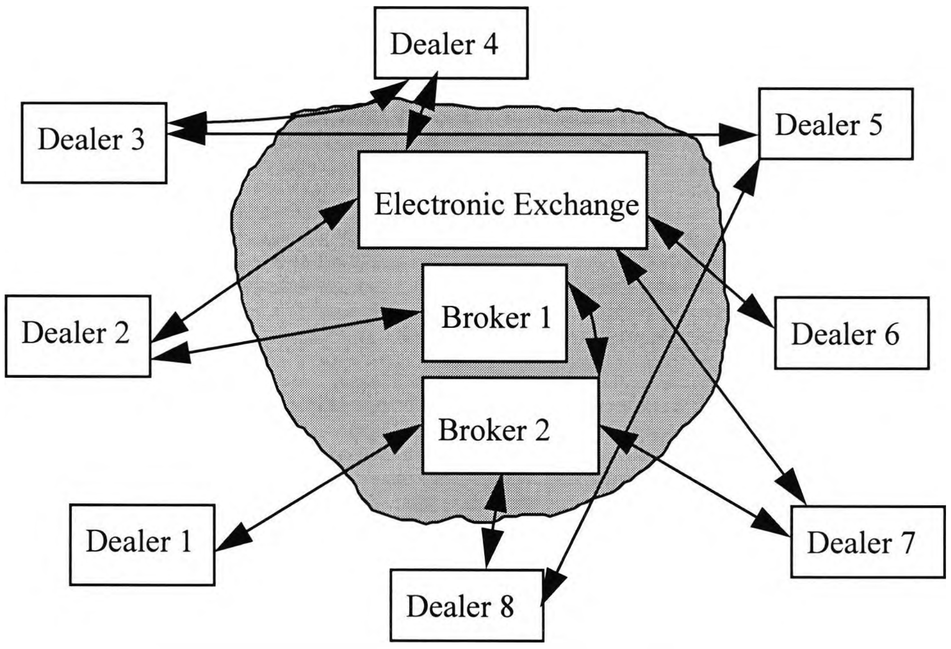


Figure 4.3 - A Simple Currency Exchange Market Model

The mechanics for pricing currencies is also based on the mechanism for transactions, but is slightly more complicated. There are several data feeds that provide price information but only the transaction records between the trader/broker or trader/exchange provide a history of the actual price that currencies are exchanged at. There are, however, *indicative values* of currency price available which are those supplied by news providers such as Reuters or Telerate. The time series values obtained from these systems are in turn obtained from frequent sampling of dealers in the market. The indicative rates are a method by which the dealers can advertise their prices but the actual price at which a trade is completed will be arranged by telephone and known only to the participants. The indicative rates will, therefore, lag the market slightly. It is important to note that it is the time series of indicative rates that are recorded and are available for forecasting, not the actual transaction rates.

4.4.1.4 Potential for forecasting

The currency markets are often described as 'efficient'. The economic theory of efficient markets states that in such markets the price will reflect all the information available. As was discussed earlier, however, the speed of information flow and volume of data in the currency spot markets means that the necessary prompt analysis of information is heavily reliant on the use of information technology. It therefore seems reasonable to attempt to forecast in the very short term where the information 'overload' may yield unused information. As the time available to traders for analysis increases, it is more likely that market participants will spot patterns and trends. In the spot market, traders have very short investment horizons, conduct many trades, and hence have little time for complex or detailed analysis. A survey conducted by Taylor and Allen (1992) looked at the use of technical analysis and fundamental approaches to forecasting over varied time frames in the foreign exchange markets. They concluded that there was a skew in reliance on technical analysis tools for shorter investment horizons, adding that many respondents viewed them as '*self-fulfilling*'. This suggests that market participants view technical tools as potentially worthwhile in the short term and given the wide belief of self-fulfilment, potential may well exists for an adaptive ANN approach to identify patterns.

The success of technical or computer based trading tools remains an unanswered question. As Lequarré (Weigend and Gershenfeld, 1993; p.137) points out, successful algorithms in the financial environment are kept secret, making research frustrating especially for *'the ones who come from academia'*. Claims have been made about computer forecasting systems yielding impressive returns on capital (Economist

1993;p.19; Economist, 1992b ; Refenes, 1991) but without independent analysis and verification, that is how they remain - claims.

The data sets shown in Figure 4.4 are five daily closing prices for currencies quoted against the US dollar. The time series cover a four year period and illustrate the very different characteristics of currencies. The two European currencies exhibit roughly similar behaviour. The Italian Lira was within the European exchange rate mechanism (ERM) for some of this period and the Finnish markka was outside. This mechanism attempted to fix member country exchange rates to within a well defined bandwidth. The Japanese Yen exhibits very different characteristics. The three currencies discussed so far exhibit very small, incremental changes in exchange rate, whereas the less fluid Chinese and Argentinean rates show pronounced, larger movements.

The scale of currency movements and the long term behaviour of currency pairs are typically very different if the exchange rates are free floating. If the exchange rates are fixed or floating within limits then the currencies will be correlated to a higher degree.

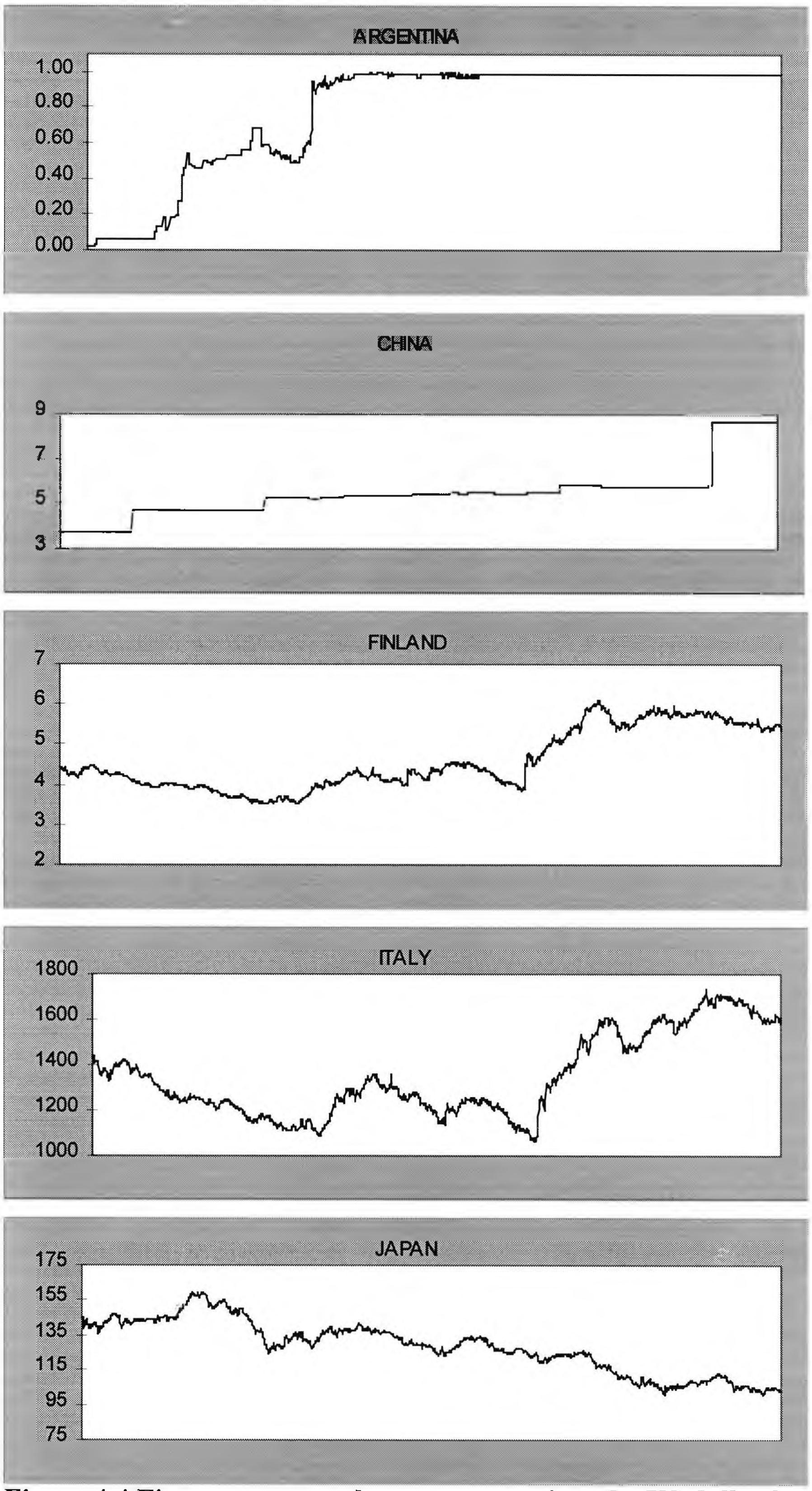


Figure 4.4 Five currency exchange rates against the US dollar between 6/90 and 6/94 with 1307 data points in each series[2].

[2]The price shown (y axis) is the quantity of the currency that can be purchased for one US dollar. Time (x axis) is not shown for clarity but has the same range (6/90$\rightarrow$6/94) and interval (1 day) for each series.

4.5 ANN forecasting experiments

Two foreign exchange time series of different sampling frequencies were used for the forecasting experiments; the Lira daily closing price (1307 prices) and the Lira five-minute closing price series (2223 prices). Both time series are of the Lira - US dollar exchange rate.

Pursuing the analytical framework established in chapter 3 and using some of the analytical techniques described earlier in section 4.2, the information described in the following section was collected as a precursor to experimentation. In addition, three artificial time series were generated to help with the interpretation of the visual and quantitative analytic techniques used to characterise the data sets (appendices A and B contain function details and graphs of the data).

The discussion covers the daily and five minute time series in turn. For each series, an analysis of the characteristics is presented before consideration of the ANN algorithm and architecture selected for evaluation.

4.5.1 Daily time series: characteristics

Dynamics

The time series did not appear to have any discernible structure in the underlying dynamics (evidenced by investigation of phase and embedding plots) and appears to be non-linear as evidenced by the complex phase diagrams (Figures 4.5 and 4.6).

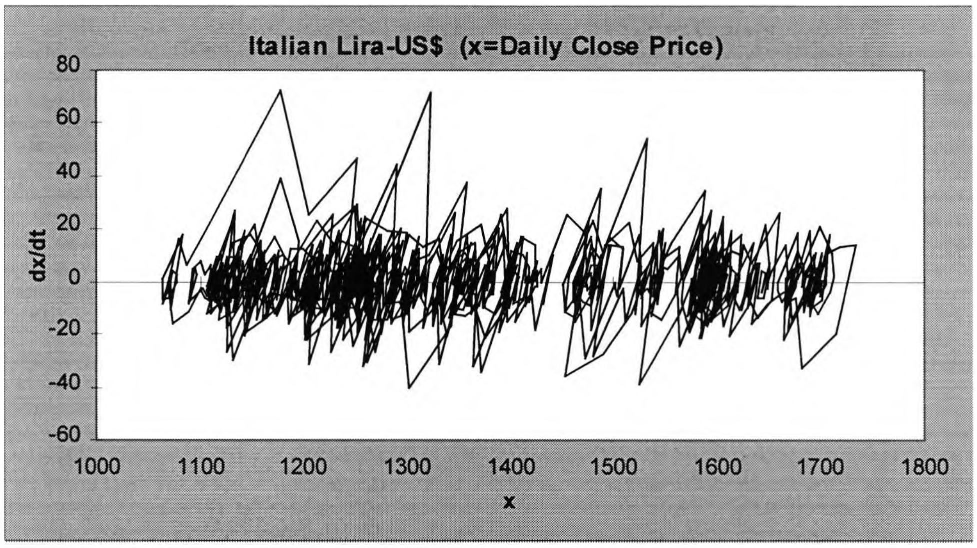


Figure 4.5 - Phase diagram of Italian Lira (x vs dx/dt)

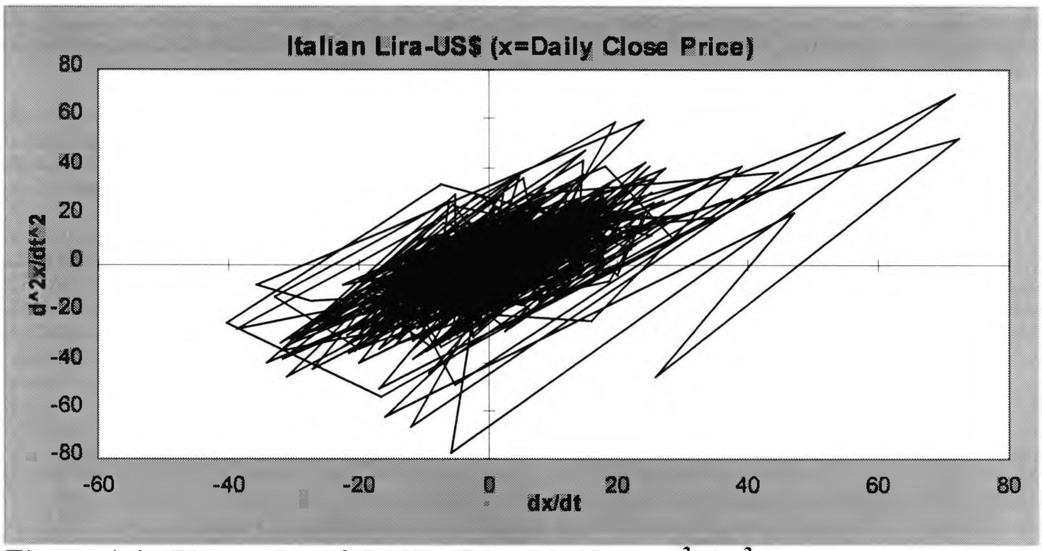


Figure 4.6 - Phase plot of Italian Lira (dx/dt vs. d^2x/dt^2)

Stationarity

The series is non-stationary (mean and variance vary with time) as can be seen from the time series plot (trends) and from the autocorrelation functions showing a linear decrease with time.

Distribution

The time series does not have a normal distribution. (evidenced by the histogram plot; Figure 4.7).

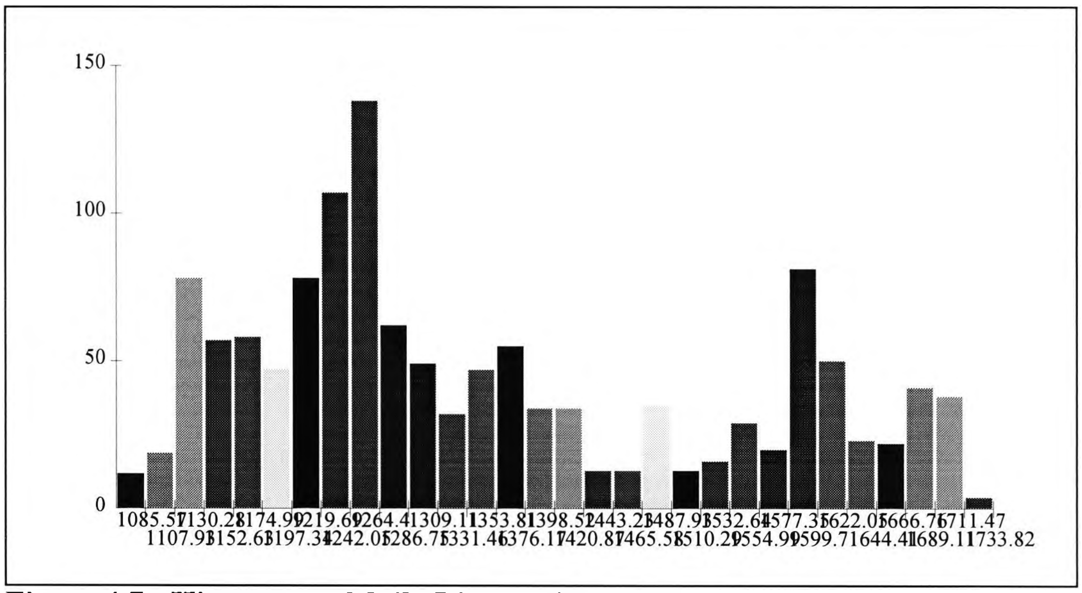


Figure 4.7 - Histogram of daily Lira closing prices.

Autocorrelation

The autocorrelation functions (ACFs) for the three artificial time and Lira series are shown in Figures 4.8 and 4.9. Calculations were made in the MINITAB statistical package with each ACF series covering 150 time lags. The Lira has a very similar ACF to the artificial generated 'line' time series, indicating that there is a strong trend component in the data. If an ACF of the Lira's first differential is taken, however, then a very different series is evident (Figure 4.9). This indicates that the price movement (the variable of interest) is very similar to the random time series. Using the differential 'uncovers' the day to day dynamics, which in the case of the Lira, are dwarfed by the absolute price value. An absolute price value used as an input to an ANN would vary very little (0-5%) therefore providing minimal chance for the learning algorithm to associate changes in input with output. The differential value has a full value range (0-100%) thereby increasing the chances of learning through weight modification.

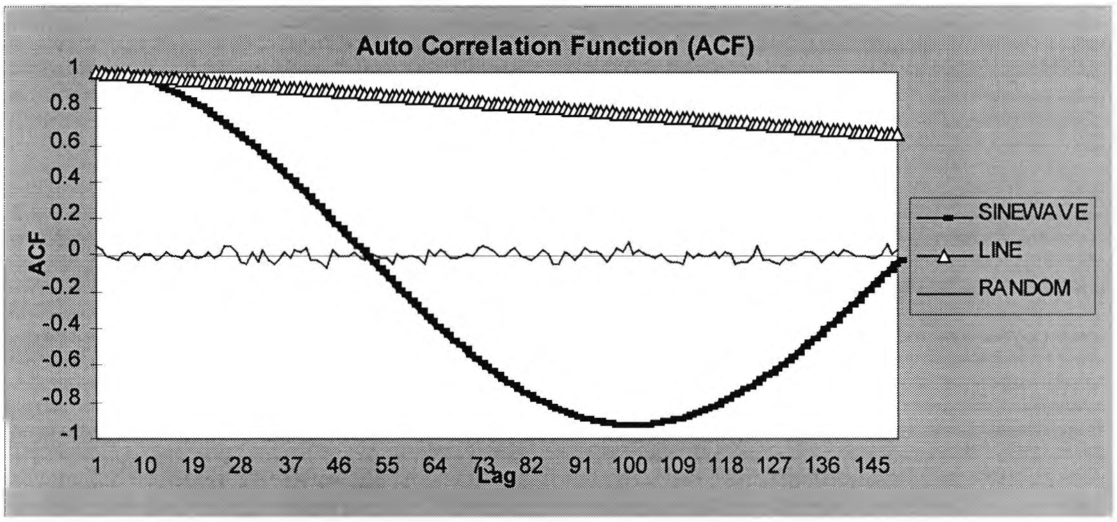


Figure 4.8 ACF functions for synthetic time series

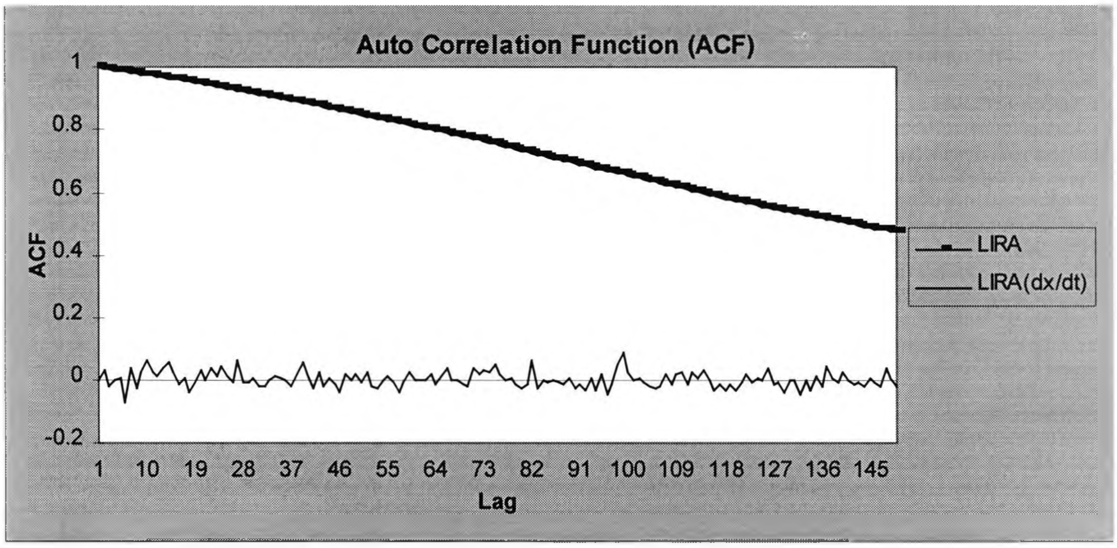


Figure 4.9 ACF functions for Daily Lira and first differential

The ANN and learning algorithm selected for forecasting the daily time series are the feedforward architecture and backpropagation rule. The problem is presented to the ANN as a tapped delay line with the difference between successive prices as inputs and a price difference as the forecast value. The interpolative properties of the feedforward network architecture would seem appropriate to the form of prediction required.

The (absolute) daily price time series and the first differential are both non-stationary in character. The series therefore offer little chance of successful forecasting using ARMA methods. ARMA requires that the time series to be modelled has a constant mean and deviation. To verify this conclusion, an attempt to predict was made using the Box-Jenkins ARMA model implemented in MINITAB but all models failed to fit, as expected.

To investigate the performance of the ANN model two error measures are taken. As the data set is difficult to characterise and appears to offer little hope for accurate forecasting, the random walk ratio is adopted as the primary measure of success. If models yield a performance error ratio lower than 1.0 it would indicate that the model under evaluation offers a prediction better than chance. A measure of the correct prediction of price movement (e.g. up, down or none) is also given. From a trading perspective this is probably the most significant measure of performance.

A computational model to predict daily prices is not constrained by the operational performance requirements needed for minute by minute prediction. It is likely, however, that there will be retraining of the model on a regular basis (perhaps daily). There is therefore an emphasis on training time being as short as possible. It would be useful for a model to provide an explanation of prediction or some form of reasoning, but this is not necessary since its utility will be measured by its accuracy alone. If the system were to be implemented it would be in an 'advisory' role leaving traders with the option to act upon its predictions.

4.5.2 Daily time series : experiments and results

The experimental data is drawn from two data series of 1307 closing prices for each currency (Lira and Yen). The Yen series was readily available and is included for comparison purposes only. Tapped delay lines are used to provide a six dimensional state space problem representation to the ANN for each currency. The selection of six values is somewhat arbitrary as no indication of the underlying dimension of the series could be ascertained from the embedding or phase diagrams (see Figures 4.5 & 4.6). Training is carried out with 1,200 data samples from the available 1307, leaving 100 or 99 cases for testing, depending on the representation made to the ANN. In instances where price differences are used, the 6 input tapped delay line uses the 7 previous data points. In all experiments the random walk ratio is used to test that the performance of the ANN forecasting model is better than chance.

Currency	Configuration	Format	Training	Testing	Error Function
Lira	Price,Price,n=6	6,1	N=1200	N=100	Rnd. Wlk. Ratio
Lira	Diff,Price,n=6	7,1	N=1200	N=99	Rnd. Wlk. Ratio
Lira	Diff,Diff,n=6	6,1	N=1200	N=99	Rnd. Wlk. Ratio
Yen	Price,Price,n=6	6,1	N=1200	N=100	Rnd. Wlk. Ratio
Yen	Diff,Price,n=6	7,1	N=1200	N=99	Rnd. Wlk. Ratio
Yen	Diff,Diff,n=6	6,1	N=1200	N=99	Rnd. Wlk. Ratio

Table 4.2 Configurations for Lira and Yen (daily) experiments

Three different data presentations were attempted for each currency; (i) previous prices input and next price output (ii) previous price differences with last price as input with next price output and (iii) previous price differences input and next price difference as output.

All of the experiments (for both currencies) proved unsuccessful with the backpropagation learning algorithm using the NeuralWorks Professional II package running on a SUN Sparc 2 workstation. There were no indications of convergence (reduction in output error), despite training runs in excess of one million iterations and using numerous variations of the algorithm parameters and network topologies. Learning parameter settings were varied for different run lengths and momentum terms were included in an attempt to attain convergence. It was concluded that backpropagation could not be trained for these daily data sets.

4.5.3 Five-minute time series: characteristics

Dynamics

Due to the non-contiguous nature of the data sets, selected examples from the 26 continuous time series are viewed using phase and embedding diagrams. The plots have no discernible structure evidencing no identifiable low order in the underlying dynamics. In the absence of any indication of dynamic order, the selection and number of tapped delay line points remains open to experiment.

Stationarity

As with the daily lira prices, the time series are non-stationary as can be seen from the trends in the time series and standard deviation plots (Figure 4.10). The lack of stationarity indicates that an ARMA model would be unsuccessful in predicting in this instance.

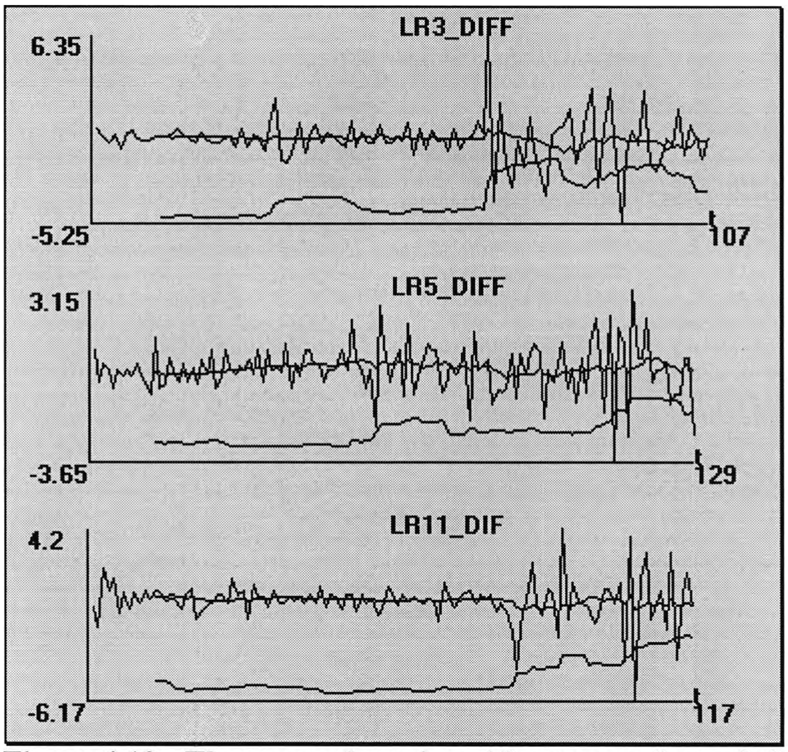


Figure 4.10 - Three sample series with mean and a scaled SD (x5)

Distribution

The time series selected for example have exhibited reasonably normal distributions. The overall time series resulting from joining the contiguous data sets is, however, not normally distributed.

Autocorrelation

Autocorrelations were calculated for selected series and found to be similar to the random series (e.g. serially uncorrelated). This indicates that there is little evidence of periodicity in the data.

4.5.4 Five-minute time series: experiments and results

The eventual database of cases for training and testing, following the processing described in appendix B, contained 2223 records. All experiments adopted the train-and-test method of data sampling. The first two experiments used training and test samples that were contiguous. Experiments 3 through to 6 used interspersed training and test samples. Test data sets contained no cases that were used to train the networks. The experimental results are presented in tabular form in Table 4.3.

Feedforward ANNs with varying numbers of hidden layers were applied. The 'model' column in table 4.3 indicates how many inputs (6 in all cases), hidden nodes & layers (varied) and outputs (1 in all cases) were used in each experiment. The networks were all trained using the backpropagation learning algorithm encoded in the Neural Works Professional II software package and running on a SPARC 2 workstation.

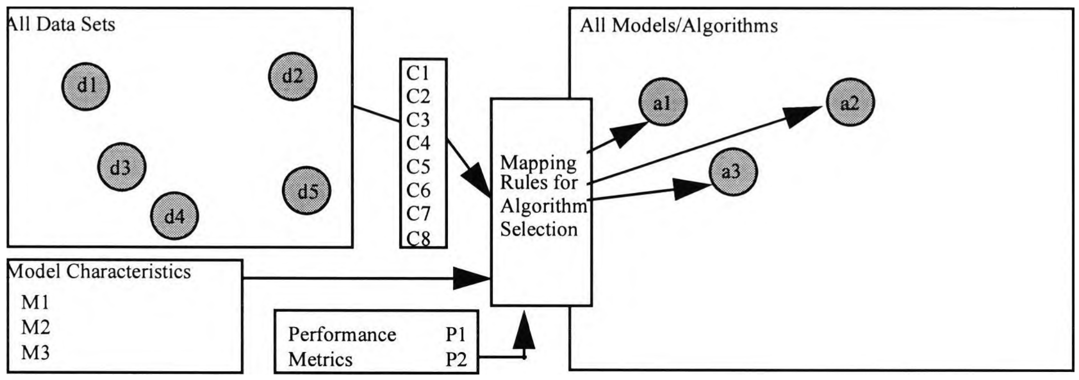
Exp.	Random Walk Ratio	Direction Correct (%)	Model	Training Iterations	Training Set / Test Set
E1	0.92	**55.6 (±0.21)**	6,10,6,1	121,307	1779 / 444
E2	0.91	**57.0 (±0.21)**	6,10,6,1	304,544	1779 / 444
E3	0.91	**55.6 (±0.24)**	6,10,1	100,907	1803 / 420
E4	0.89	**58.6 (±0.24)**	6,10,1	102,245	1803 / 420
E5	0.89	**59.3 (±0.24)**	6,10,1	140,592	1803 / 420
E6	0.89	**59.5 (±0.24)**	6,10,1	1,029,285	1803 / 420

Table 4.3 - Experimental results for 5 minute lira data.

Note: The Random Walk Ratio (equation 4.2) is calculated using the difference time series and not the actual price time series. This yields a more sensitive error measure compared to a calculation based on absolute price.

4.6 Discussion of results

The experimental method developed in chapter 3 is used as a template to summarise the experimental work presented in this chapter.



Characteristic Measures
C1=Number of Cases (examples)
C2=Mean
C3=Standard Deviation
C4=Histogram Distribution
C5=ACF Distribution
C6=Phase Diagram Interpretation
C7=Embedding Diagram Interpretation
C8=Contiguous series

Models/Algorithms
a1=Linear Regression
a2=ARMA
a3=Artificial Neural Network (ANN)

Data Sets
d1=Daily Lira (dx/dt)
d2=5-minute Lira (dx/dt)
d3=Random
d4=Line
d5=Sine Wave

Model Characteristics
M1=Forecasting Speed
M2=Training Time
M3=Reasoning Capability

Performance Metrics
P1=Direction Correct
P2=Random Walk ratio

Figure 4.11 - Summary of experimental method : forecasting

The mapping rules and heuristics for algorithm selection were gained from relevant published proofs and texts documenting algorithm performance. The mapping rules are an important component to the experimental method. As is evident from the algorithm review presented in this chapter, no rules were evident that applied to the data set characteristics. For the methodology to be of value, development of mapping rules is necessary. For the heuristic rules used to be of value, they will need to be validated by larger quantities of empirical evidence.

The time scale of prediction is a very important consideration in the modelling process. ARMA models appear to offer a longer term prediction capability than is required for the experiments discussed here. The parametric methods (a1, a2) do not perform as well in this short term prediction study (1 time step ahead) as the ANN (a3) method.

The selection of an appropriate tapped delay line length for optimum model performance was difficult to judge. Models a1, a2 and a3 all require a definition of such inputs for the problem representation adopted in this study. The number and spacing of delay line 'taps' was set heuristically. Since the currency time series (d1 and d2) showed no underlying order in the phase and embedding plots and had ACF's similar to a random series, characteristic measures C1 to C8 shed no light on the problem of selecting delay line length. An instructive characteristic measure indicating potentially successful configurations would be of great benefit to the modelling process. Refenes et al.(1992) discusses the use of such windows but at the time of writing there are no clear guidelines or rules to select appropriate sizes.

The number of cases in the data sets (Daily, 1307; 5-minute, 2223) are sufficiently large for all the models considered in this evaluation. For a generally applicable method, however, rules would have to be developed for specifying minimum (and possibly maximum) data set sizes for each of the models under consideration. Such rules may well be dependent on other factors such as number of delay line taps, variance of the data etc.

The selection of data sets for training/testing may have a strong influence on the accuracy of model prediction. In the latter 5-minute data experiments (Table 4.3; E3 to E6) it was noted that the prediction performance improved as the sample and test data became interspersed. It would be instructive to use a cross validation technique to investigate the bias of forecasts due to different sampling sets.

This study is restricted to univariate time series forecasting. A logical extension to the method is to include other variables of interest; the multivariate case. For daily data it is likely that additional factors such as interest rates would assist in forecasting. The very short term 5 minute data is, however, characterised more by short term 'market behaviour' and is more suited to the univariate case.

The results for the ANN prediction of 5-minute data are encouraging when considered in terms of the metrics chosen. In particular the higher DC values obtained (59%) are better than chance. Caution is required, however, since a true evaluation of the forecasting capability in this particular case would require the application of a trading strategy which included the cost of transactions. The ultimate outcome would be the system's ability to be consistently profitable over time. An appropriate cost function would have to be developed to simulate trading rules and discount the cost of transactions.

The analytical method has proved of value in the analysis of forecasting by defining the process as a series of identifiable components. Many issues have been raised and some have been discussed in this section. The potential for producing a focused (considering only univariate time series with tapped delay line sampling) forecasting method based on empirical analysis is the natural progression for this systematic approach. The framework supports the expansion of characteristic measures and new algorithms which are evaluated using a range of data sets to yield refined or extended algorithm selection rules. A natural habitat for an evolving system of inter linked measures, rules and models is computer software. To facilitate the research, numerous programs were developed, primarily to integrate the various characteristic measures. A frustrating observation is that, to perform many of the measures requires the use of various software packages. Application and research will benefit from more focused algorithms (e.g. for short term, univariate forecasting) that integrate the stages involved applying rule based selection (or advice) for selecting algorithms.

CHAPTER 5 - CLASSIFICATION

5.0 Introduction

Classification is a process carried out in many domains from finance to medicine. Whether it be a human physiological system or a currency market, attempts are made to classify based upon observations. An expert such as a doctor in medicine or market maker in financial markets is adept at making such classifications. They do this based on either rules, experience or a combination of both. The rules based approach is appropriate if the observed variables can be readily analysed to form a rules base. In reality few 'real world' systems are so readily deterministic and we therefore have a seemingly complex classification process. Further complexity is added to the situation by the often large number (increasingly so) of variables open to observation. It is within this context of data rich environments that this chapter explores the potential of data based approaches to classification. This is in contrast to the knowledge based approach.

There already exist a number of machine learning and statistical algorithms to tackle data based classification problems. Artificial Neural Networks have recently joined the ranks of such algorithms.

In this chapter the classification procedure 'will be applied to a continuing sequence of *cases*, in which each new case must be assigned to one of a set of pre-defined *classes* on the basis of observed *attributes* or *features*'. (Michie et al., 1994; p.1). The term 'feature space' is used to denote the 'input' or 'sample' space found in other texts. The dimension of feature space will be *n* with the feature vector *X* having components $[x_1,x_2,...x_n]$.

A generic outline of the classification process is presented first. This is followed by a critical review of the literature covering established techniques and Artificial Neural Networks (ANNs). Studies that have attempted to compare classifier performance are then reviewed before the data set is introduced.

5.0.1 Hypothesis

Artificial Neural Network models are capable of solving data based classification problems with comparable performance (by defined metrics) to established statistical techniques.

5.1 The classification process

The analytical and experimental method described in chapter 3 is applied here. Appropriate performance metrics, data set characteristic measures and classifier models are discussed and defined in this section.

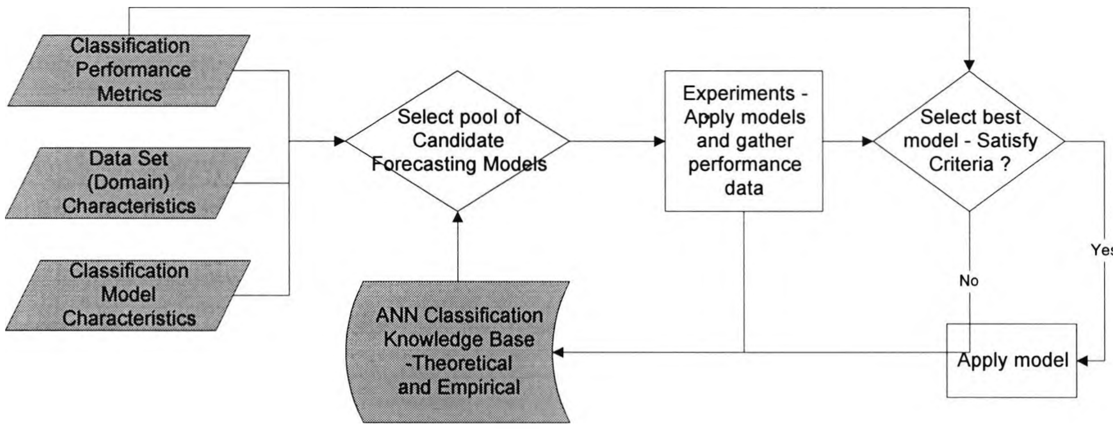

Classification Performance Metrics
Data Set (Domain) Characteristics
Classification Model Characteristics
Select pool of Candidate Forecasting Models
ANN Classification Knowledge Base -Theoretical and Empirical
Experiments - Apply models and gather performance data
Select best model - Satisfy Criteria ?
Yes
No
Apply model

Figure 5.1 - The classification process

5.1.1 Data set characteristic measures

Perhaps the most important component of any classification task is the data set. It is proposed in the method presented here that by analysing the data before any attempts are made to classify can lead to a better solution. Performing characteristic measures has an added benefit in the classification domain; it can lead to data set reduction. High dimensionality is a problem in classification (Hand, 1981) and it is beneficial to reduce the number of features if the resulting *information* loss is minimal. It is a measure of the information contained in feature variables and more specifically their discriminating ability between classes that is of particular value in the initial analysis of data sets.

The second purpose of the measures is to facilitate the selection of an appropriate algorithm for classification. The criteria for selecting measures here is more difficult to define. The table below describe the measures reviewed and their purpose (A-Analysis of data, S-Selection of algorithms).

Measure	Symbol	Description
Sample Size - [S/A]	n	Total Sample size
Features - [S/A]	f	The total number of features in the feature vector
Binary Features - [S/A]	f_b	The number of binary features in the feature vector
Ordinal Features - [S/A]	f_o	Ranked Features in the feature vector
Classes - [S/A]	c	The number of classes
Entropy - [A]	$H(f_b)$	For binary features a measure of the information content
Joint Entropy - [A]	$H(f_b, c)$	Joint information measure for binary features and classes
Av. J. Entropy - [S]	$\overline{H}(f_b,c)$	Average of joint entropy across all classes and binary features

Table 5.1 - Classification data set characteristic measures.

A detailed description of the measures in table 5.1, covering formulae, ranges and qualitative meaning of the measure values can be found in Appendix C.

5.1.2 Classifier performance metrics

To measure the performance of a classifier it is necessary to define the accuracy or error metric and how the data available can be sampled to compute such a metric. Potential metrics and data sampling are described in the following text.

5.1.2.1 Data Sampling

Three readily identifiable methods are available for the sampling of data to enable training and testing of classifiers; cross validation, train-and-test and bootstrapping. The three techniques are discussed below and summarised in table 5.2.

Cross validation is a widely used technique where the data available is divided into *n* samples and each sample is tested against the classifier created (trained) with the remaining *n-1* samples. At its extreme (*n*=sample size) it becomes the leave-out-one method which is only practically possible with small data sets. The method is computationally intensive as each cycle requires retraining of the classifier. The method is therefore suited to classifier algorithms where training time is short.

The train-and-test technique selects two random samples from the database of available cases. One sample is used to construct a classifier and the other is used to test its performance. This is suited only to large test sets where leaving out training samples will not yield too small a training set.

The Bootstrap procedure is more involved than the two techniques detailed above. In bootstrapping, the database of cases is replicated many times by random sampling with replication. The generated sample is used as a training set and cases not selected from the database are used as the test set.

Technique	**Data set size**	**Advantages/Disadvantages**
n-fold Cross validation	Medium, Large	Requires *n* training cycles (-) Reduces biasing in samples (+)
Train-and-test	Large	Only one training cycle (+) Requires large data sets (-) May yield biased samples (-)
Bootstrap	Small	Technique least likely to be biased (+) Computationally intensive (-)

Table 5.2 - Data sampling techniques

A more detailed and referenced discussion of these methods can be found in Michie et al. (1994, p.108).

5.1.2.2 Accuracy / Error measures

To evaluate or compare the performance of classifiers it is necessary to define appropriate metrics to measure the accuracy or error rate in testing. Potential functions are discussed below.

One of the simplest measures of accuracy is what will be termed here the true positive (TP) function which can be calculated for each class and can be averaged to provide a measure of classification accuracy over all classes. The formula for this measure is:

$$TP_x = \frac{\sum Correctly\ Classifed\ x}{\sum Class\ x} \times 100 \text{ where the summation is made over all cases.} \qquad (5.1)$$

The measure described above indicates the number of correct classifications. It does not indicate, however, the cases that are incorrectly classified as Class x. A measure providing this information is known as the false positive (FP) function. In a two class problem (*x,y*) one measure can be simply calculated from the other (FP_{xy}=100-TP_x). In problems with more than two categories they must be calculated explicitly. The false positive function is defined as:

$$FP_{xy} = \frac{\sum Class\ x\ incorrectly\ classified\ y}{\sum Class\ x} \times 100 \qquad (5.2)$$

These functions can be broken down for categories and displayed in matrix form. The matrix for a four class problem is shown in table 5.3. It is instructive to view the accuracy in this way as it helps identify classes which the classifier has difficulty in separating.

Class	A	B	C	D
A	TP_A	FP_{AB}	FP_{AC}	FP_{AD}
B	FP_{BA}	TP_B	FP_{BC}	FP_{BD}
C	FP_{CA}	FP_{CB}	TP_C	FP_{CD}
D	FP_{DA}	FP_{DB}	FP_{DC}	TP_D

Table 5.3 - A true and false positive matrix

5.1.3 Classifier model characteristics

There are some characteristics of classifier solutions that are not entirely quantitative but need to be included in the selection process. These are discussed below.

The speed of solutions can often be a critical limiting factor. Some classification algorithms require much computation. The computational demand of algorithms can be subdivided into two types; training (or learning) demand and operational demand. Backpropagation ANN models, for example, frequently demand intensive computing power in the training phase but are relatively simple and therefore less demanding in operation. In a continuous operating environment such a on-line patient monitoring in medicine, or feedback control in engineering the operation time of a model will have a limit whereas training time is not critical.

The model may be required to attain a certain accuracy (average) in operation. This value can be used with the accuracy metrics (section 5.1.1) to establish a minimum performance level by which any proposed model can be judged to be adequate.

An analysable model or evidence of 'reasoning is in many situations a desirable if not necessary aspect of any solution. In medical diagnostic systems and safety critical control systems there are obvious legal implications for users of model. A significant drawback of ANN models is that they are frequently difficult to analyse.

Henery (in Michie et al., 1994; p.7) defines four considerations in the evaluation of classifier performance; accuracy, speed of classification, comprehensibility of classifier and the learning/training time of the classifier. These issues can in turn, however, be subdivided; accuracy requires the use of some metric and there are many parameters involved from the selection of data for learning and testing to the error measure function. The scope of the comparisons possible in this thesis must be therefore limited to that which supports the hypothesis. Informative texts that expose more detail are referenced where appropriate.

5.2 Methods of classification

It is difficult due to the overlap of techniques to create useful subdivisions of the number of classification algorithms available. The groupings used here are nominal and may differ to other texts. The groups are :

- Statistical Methods

- Rule Structured Classifiers (a branch of Machine Learning)
- Artificial Neural Network Models

This section discusses the prominent classification algorithms available. It is against such algorithms that the Artificial Neural Network models must stand for comparison. To facilitate the description and comparison of these techniques a simple (synthetic) classification problem is used. The diagram below shows the two category data distributed in its two dimensional feature space. The feature variables are continuous with upper and lower bounds.

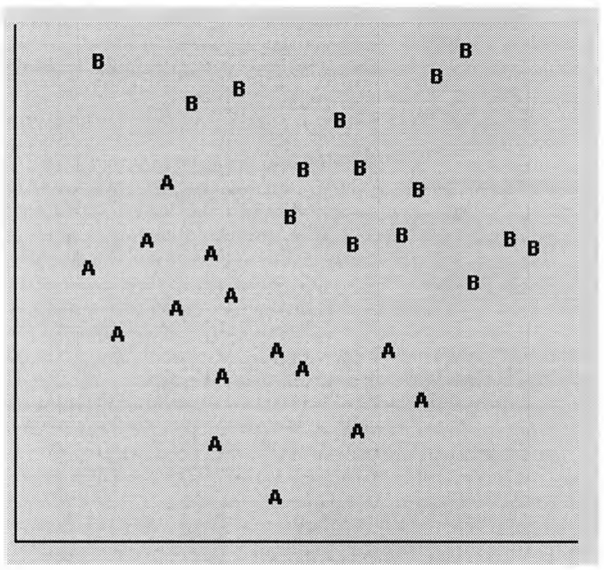


Figure 5.2 - Synthetic 2-D data set

5.2.1 Statistical Methods

Linear, Quadratic and Logistic Discriminant Functions

The Linear Discriminant Function (LDF) developed by Fisher (1936) is cited by Hand (1982) as the first method to be developed and most widely used in practice for discriminant analysis. The parametric technique results in a hyperplanar surface (a line in a 2 dimensional feature space) that separates the classes optimally according to an error metric (a function of estimated class means and covariance matrices). This method is effectively maximising the F-test value.

Fisher attempts to maximise the ratio of the difference between the means to the standard deviations of the two classes (Iris species) with a linear combination function of the form:

$f(X)=\sum_{j=1}^{n}\lambda_j x_j$ where λ_j are the coefficients of the n dimensional feature vector

components $x_1..x_n$. A limitation of this parametric approach is that the division of feature space will always be linear and with many problems this may prove to be unrealistic. A non-linear approach is to use the quadratic discriminant function which has a curved quadratic surface to partition feature space. Logistic regression separates the classes with a hyperplane (of the same type as the linear) but the error measure is of a different form. Fishers quadratic cost function is replaced with the maximisation of a conditional likelihood.

All three methods are analogous to regression analysis. Ripley (1992) gives a formal exposition of this link.

Bayes Rule (Non-Parametric)

A probabilistic approach can be taken using Bayesian methods. This requires that the Probability Density functions for the classes are known. For class probability density functions (pdf's)that can be defined by a continuous mathematical function a so called 'Bayes optimal ' classifier can predict the theoretical limit to optimal classification accuracies. This approach was used by Kohonen et al. (1988) for a benchmarking study of different classifiers. An advantage of Bayesian approaches is that *a priori* probabilities and cost functions can be included.

With 'real world' data sets it is often impractical to obtain mathematically precise functions for pdf's and some method of estimation is required. Techniques for such estimation are covered in the following section on Density Estimation.

The components included in a bayesian classifier are class likelihood functions, the *a priori* probability of the classes and the cost of misclassifying a class. For a two class problem Specht (1989) clearly shows the simplicity of the classification process provided that the above components are known.

Class A is selected if $h_A\, l_A\, f_A\,(X) > h_B\, l_B\, f_B\,(X)$

Class B is selected if $h_A\, l_A\, f_A\,(X) < h_B\, l_B\, f_B\,(X)$

where h is the class prior probability, l is the cost of misclassification and $f(X)$ is the class likelihood function. X is the feature vector of the case to be classified.

Using a simple line graph (Figure 5.2) it is easy to demonstrate the classification process in one dimensional feature space. The class probability density functions are both normally distributed with $\bar{x}_A = 6$, $\bar{x}_B = 14$ and $\sigma_A = \sigma_B = 3$. The decision surface is where the two curves intersect (x=10).

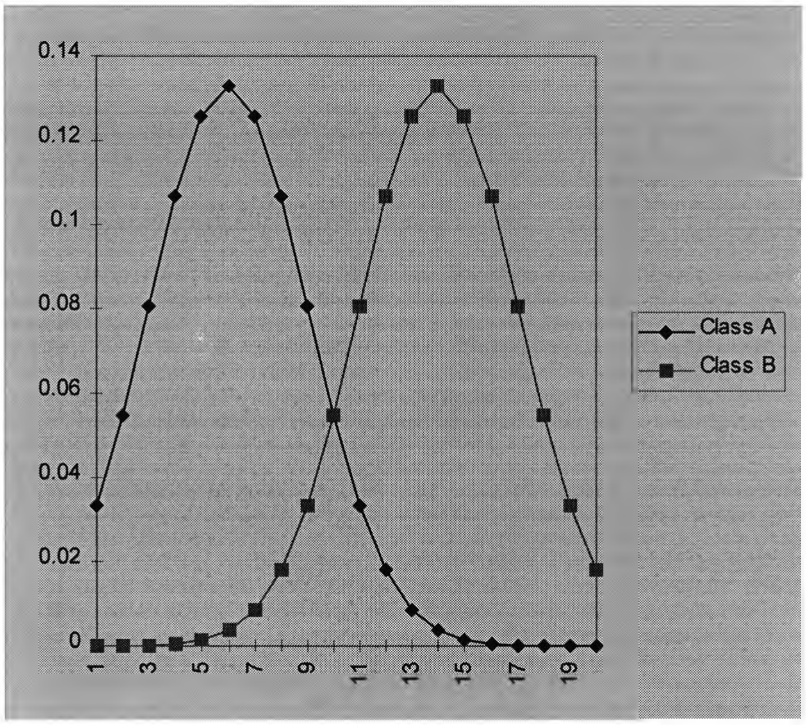


Figure 5.3 Class likelihood functions for classes A and B

A shift in the decision surface (a point) can be seen when class probabilities of h_A=0.2 and h_B=0.8 are included, modifying the probability distributions (Figure 5.4). The cost functions for both classes are equal (l_A=l_B). As can be seen the decision surface is now at approx. x=8.5.

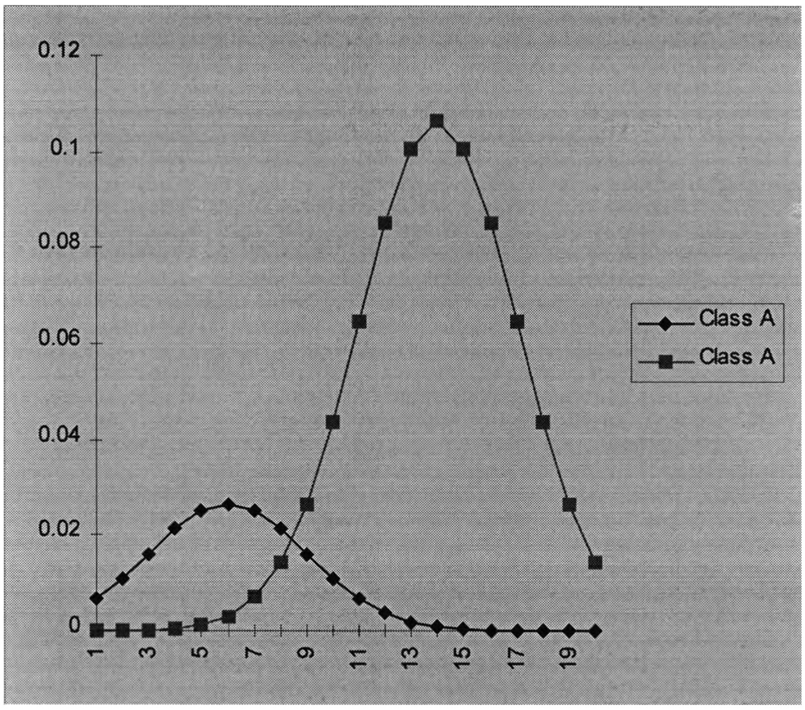


Figure 5.4 - Class A and B *posteriori* probability density functions

A Bayesian rules classifier is optimal, theoretically giving the lowest error rate. In practice it is usually impossible to obtain such a classifier without using synthetic data sets. The problem is that the probability density functions seen in Figure 5.2 require complete knowledge about the data distributions. With real world data sets, accurately obtaining these distributions in *n*-dimensional feature space for each class would be computationally impractical if not impossible. The alternative is therefore to estimate the density functions.

Multivariate Density Estimation

There are many algorithms that are associated with density estimation. An informative breakdown however, is to split the approaches into local density estimates and global density estimates. Roughly translated a local density estimator will use only points in feature space that are close to the new feature vector to be classified. Kernel and nearest neighbour methods are of this type. A global density estimator will use all the data points to classify the unknown vector. Hand (1982) presents an authoritative and well referenced account of kernel estimators. The earliest kernel methods traced by Hand; Fix and Hodges (1951), Rosenblatt (1956) and Parzen (1962) are recent due to 'their fundamental intractability compared to the parametric forms'. The ever rising computational power of modern computers has eroded this problem.

Many analogous problems encountered in ANN experiments such as convergence and choice of activation function have been studied in detail by statisticians working on kernel estimation. Hand also covers the concepts of bias and variance that are used in the argument put forward by Geman et al. (1992) that small bias and variance are the key to good classifier solutions. They argue that this in turn is a function of the data set and for practical artificially intelligent systems the data sets do not contain enough examples to learn the complexity within them. This must be 'hard wired'.

Multivariate Density Estimation allows construction of pdf's required for Bayesian classification. The kernel based approach allows construction of the overall pdf from multiple kernel functions in feature space. Parzen (1962) demonstrates a univariate approach to density estimation building up from kernel functions such as gaussians. His work was further extended by Coucallous (1966) to the multivariate case and his extension is used by Specht (1989) in a 'Probabilistic Neural Network' which approximates the Bayes optimal decision boundaries.

Kernel estimators require the selection of a kernel and a smoothing parameter. It is claimed that the smoothing parameter is more important than the actual kernel (Michie et al., 1994; p.31) and this is discussed in more detail by Specht (1990). Hand (1982) provides a detailed account and discusses the choice of kernel with respect to continuous and categorical variables in the feature vector.

k-Nearest Neighbour

The *k*-nearest neighbour (*k*-NN) algorithm is a simple yet often highly accurate classifier method (see performance comparisons in Michie et al. (1994), Ripley (1992)). Indeed the k-NN technique outperformed *all* statistical, machine learning and Artificial Neural Network techniques for satellite image classification in the comprehensive review described by Michie et al. (1994, p.143-45). The parameter *k* denotes the closest number of vectors from the data set that are used to classify the unknown vector (case) of interest. In the simplest case, where *k*=1, the class of the nearest vector to the unknown vector is the output of the classifier. The nearest neighbour decision rule has been examined in detail by Cover and Hart (1967). A review linking nearest neighbour techniques with other classifiers is given by Hand (1981).

The decision regions for 1 and 3 Nearest Neighbour algorithms using the synthetic data set example is shown in Figure 5.4. The software simulation uses a 150x150 resolution plot of the class regions and is sufficient to demonstrate the decision surface which is highly non-linear. The decision surface can be constructed from Voronoi tessellations and is piecewise linear. The nearest neighbour method can also be viewed as a kernel density estimation method with the smoothing parameter tending to zero. Specht (1990) discusses this and illustrates it graphically with different smoothing parameter values.

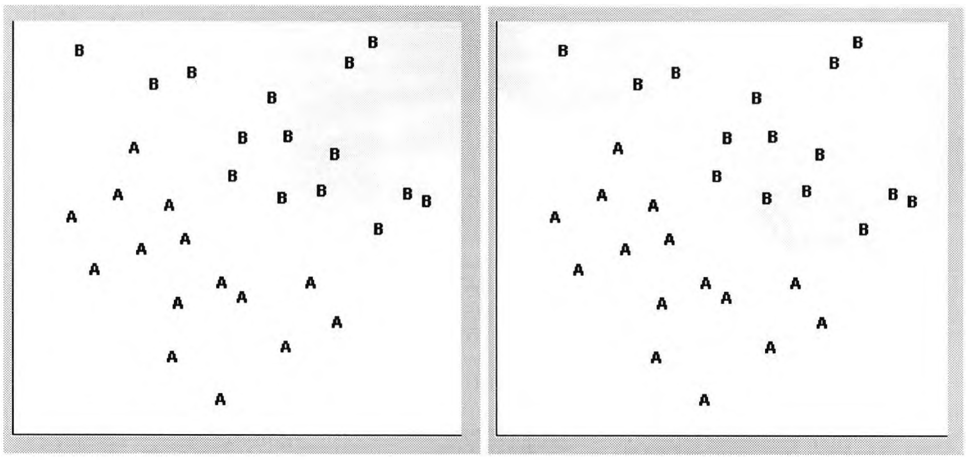


Figure 5.5 1-Nearest Neighbour (1-NN) & 3-Nearest Neighbour (3-NN)

5.2.2 Rule Structured Classifiers (Machine Learning)

The learning systems discussed here are symbolic learning systems. The solutions found by the algorithms are logical combinations of features. In the example below two dimensional feature space of age and work category is partitioned according to the following propositions:

class 1 if age>25 and work $\in$ { clerical, operator }

class 2 if age>25 and work = manual

class 3 if age<25

The partitioning defined by these propositions is rectangular and in stark contrast to (non-linear) decision boundaries generated by the nearest neighbour algorithm for example. The benefit is that they can be read.

Decision Trees

Decision trees partition feature space recursively into smaller and smaller rectangles with the desired outcome that the rectangle contains only cases from one class. A criticism is that they can be difficult to interpret as they become 'bushy'. They are, however, usually very fast in operation.

CART (Classification And Regression Tree)

CART (Breiman et al., 1984) is a binary decision tree algorithm which always yields two branches at each internal node. CART has an interesting index for evaluating the best way of partitioning the data at a node. The GINI index which defines a metric for the impurity of the subgroups chosen by a potential split (the ratio of correct classifications to incorrect). All possible splits are evaluated and the one of lowest impurity selected.

Following the construction of the tree CART performs a sophisticated pruning process which balances the two issues of obtaining the right size tree and getting accurate estimates of the probabilities of misclassification.

5.2.3 Artificial Neural Networks

A broad range of ANN models have been discussed in the literature review (Chapter 2) and selected models are expanded here in the context of classification only. The feedforward class of ANNs has been widely applied to classification problems, however, the less commonly applied LVQ class of models are also capable of performing well on classification tasks. Feedforward and LVQ models have been selected as they are very different in architecture yet have had documented success in tackling classification problems.

Feedforward Network

The Multi-Layer Perceptron (MLP) has been a widely used ANN model in classification tasks (Gorman & Sejnowski, 1988; Shu et al., 1991; Le Cun et al., 1990). The computational capabilities of the feedforward class of ANNs have been well documented (see chapter 2). In the context of classification problems is known that a feedforward ANN of appropriate size and architecture can theoretically compute complex decision surfaces to the desired accuracy. A key problem with implementing the Feedforward architecture remains the selection of an appropriate network topology and setting of learning rule parameters. Some learning algorithms have attempted to overcome these problems, notably in the setting of the network topology dynamically during learning. The successful application of models is, however, very dependent on skill of the experimentor.

Learning Vector Quantization (LVQ)

The LVQ algorithm introduced in Chapter 2 is a relatively simple ANN model. It has, however, had proven success in the field of classification (Kohonen 1988c). In a broader comparative study of classification algorithms (Michie et al., 1994) it was found that LVQ performed strongly on data sets containing large numbers of cases with a high proportion of binary attributes.

5.3 Benchmarking studies

This benchmarking review attempts to address the problem of classifier performance and in particular a means for quantitatively assessing the capabilities of a range of classifiers on deterministically different data sets. It was Minsky and Papert (1969) using a methodological approach that led to the infamous 'x-or problem'. The literature review therefore centres around texts that applied various classifier techniques to data set problems and distinguishes between synthetic and real life problems.

A structured experimental approach is that taken by Thrun et al.(1991) in comparing the accuracy of 25 classifier algorithms on three synthetic data sets (the third data set being a noisy data set). Advocates of each technique were asked to apply their chosen algorithm on the data sets; further increasing the chance of obtaining a credible comparison of techniques. The classifiers were the *AQ* learning algorithms (constructive induction), *Assistant Professional* (Inductive learning of decision tree), *mFOIL* (inductive logic program generator), *ID5R,IDL,ID5R-hat & TDIDT* (Decision Tree algorithms), *ID3, ID5R,AQR,CN2 & CLASSWEB 0.1-0.20* (Inductive Learning Programs), *Prism* (Inductive Learning Algorithm), *ECOBWEB* (Unsupervised clustering algorithm), *Backpropagation & Cascade Correlation* (Artificial Neural Network technique). A problem with the design of the experiment from an application perspective is that the data sets are too artificial to be useful examples outside of the academic exercise.

The data sets themselves were six parameter logic problems with the following number of categories per parameter $x_1..x_6$: [3,3,2,3,4,2] yielding 432 possible combinations. The test sets were created with one of the disjunctive normal form -'*supposedly easily learnable by all symbolic learning algorithms*' (Thrun et al., 1991:p2) and the second a parity type problem. From a connectionists viewpoint the results are particularly encouraging in that both the ANN techniques achieved 100% accuracy on all but the noisy data set (97.2%). A strong point for many of the alternative techniques is that they produce comprehensible rules for classification from the data. The ANN techniques do not generate any interpretable rules.

Ripley (1991) also compares a variety of techniques and performs a similar survey of the literature as to that conducted here. His experimental data is based upon tsetse fly distributions and is therefore a 'real life' data set rather than synthetic. The data set has 14 parameters that can be used to classify the presence or absence of tsetse flies. This does however yield the problem that the comparison of techniques cannot be evaluated from known rules or distributions (although canonical variate plots were used to view the data for discriminating clusters). Ripley uses a simple two variable plot for the well known iris problem to show how the structure of the data allows a simple classifier to perform well. This is exactly the kind of insight that is lost in higher dimensional problems where visualisation is impossible. The point of analysing data before applying algorithms is quite pertinently made.

Michie et al. (1994) document the most comprehensive comparative study found in the literature. The text is a summary of 3 years work from the 'StatLog' project funded under the EC ESPRIT programme. The study uses 21 data sets grouped under the headings 'Images', 'Datasets with costs', and 'other datasets' and evaluates the performance of a similarly large number of classifiers (22) under the groupings of 'Statistical Techniques', 'Machine Learning Rules and Trees' and 'Neural Networks'.

In addition to the comprehensive range of techniques and data sets there are well defined metrics for evaluating the performance of the algorithms. The comparative trials are carried out using Cross-Validation and Train-and-Test.

An important aspect of this work is the analysis of the relationship between dataset types and algorithm performance. If this relationship can be established then it should be possible to develop selection criteria and predict the optimal algorithm.

5.4 The classification problem

The data set used in the classification experiments is taken from a large database of Septicaemia episodes that has been compiled by the Department of Microbiology, St. Thomas Hospital, London. The database contains in excess of 5000 episodes spanning over 50 categories of infecting micro-organism (Gransden et al., 1990). Each episode (*case*) contains information on the patient (observed *features*) and the clinically detected micro-organism(s) (*classes*) causing the infection. The source database for experimental work detailed here contained only cases of single organism infection.

5.4.1 Septicaemia

Septicaemia ('sepsis of the blood') is the clinically significant occurrence of micro-organisms in the blood stream. The patient shows characteristic evidence of the sepsis which include fever, rigors, mental confusion, tachycardia and hypotension. The disease is severe with considerable morbidity and mortality. There are two forms of Septicaemia as defined in Sleigh and Timbury (1990, p.267), with the second form being more common:

'a) A basic feature of some generalised or 'septicaemic' infectious disease, e.g. brucellosis, enteric fever.
b) A complication of more localised infections, e.g. pneumonia. The organisms spread from the focus of the infection into the blood stream.'

Prompt treatment of septicaemia is vital. Patients evidencing septicaemia require the administration of an appropriate antibiotic therapy immediately. The chances of survival depend on the age of the patient, the underlying condition and the treatment given. The overall mortality rate for septicaemia varies between 15 and 35 %. (Shanson, 1989). To plan and administer an appropriate therapy the following information is required:

a) Identification of infecting organism(s).

b) The antibiotic susceptibility of the micro-organism(s)

c) The focus of the localised infection that has lead to septicaemia.

With this information the clinician and microbiologist can design an appropriate therapy for the patient. All of this information is not available, however, in the critical first 24 hours after the clinician has identified the condition of Septicaemia and requested blood and other relevant cultures. It will take 18 hours incubation time (typically) in the laboratory for detection of bacteria in the blood sample. In this period the clinician must 'best guess' at an appropriate therapy. This can mean using a 'blanket cover' of antibiotics which increases cost and the chances of toxicity (the antibiotics can have strong side effects). In 94-95% of cases of septicaemia the infection is caused by a single organism (Sleigh and Timbury, 1990; Shanson, 1989).

It can take a further 24 hours from the initial bacterial detection before comprehensive results on antimicrobial sensitivities are available and a final therapy chosen.

The purpose of using a data based classifier is to improve the 'best guess' made by the clinician. The objective, data based approach is in contrast to the subjective, rules based approach that has been tried in MYCIN (Shortliffe, 1976). Any increase in the prediction accuracy of micro-organism in this critical time period will be beneficial to the patient outcome (death or survival).

5.4.2 Data Set Features

The training and test data sets each contained 1664 cases randomly sampled from the database. There are 51 features for each case as follows:

- Year of the episode (Interval Scale, Continuous)
- Age of the patient (Interval Scale, Continuous)
- Sex of the patient (Binary)
- Whether the infection was acquired in the hospital (Binary)

With binary encoded variables as follows

- e) Whether the patient is already on antibiotics (Binary)
- f) Medical speciality of the ward (Binary)
- g) The underlying diseases (if any) (Binary)
- h) The anatomical site of the infection (Binary)

5.5 Experiments

The experiments followed the classifier selection process detailed in section 5.2. Analysis of the data set follows:

5.5.1 Septicaemia data set: characteristics

The data set was investigated using the measures detailed in appendix C. Detailed calculations on which these findings are based can be found in appendix D.

Measure	Value	Description
Sample Size - [S/A]	1664	Total Sample size
Features - [S/A]	51	The total number of features in the feature vector
Binary Features - [S/A]	49	The number of binary features in the feature vector
Ordinal Features - [S/A]	2	Ranked Features in the feature vector
Classes - [S/A]	4	The number of classes
Entropy - [A]	App. D	For binary features a measure of the information content
Joint Entropy - [A]	App. D	Joint information measure for binary features and classes
Av. J. Entropy - [S]	App. D	Average of joint entropy across all classes and binary features

Table 5.4 - Characteristic measures for classification

The number of micro-organism classes in the data set were deemed to be too numerous for experimentation and analysis purposes. To reduce the number of classes the database was pre-processed to yield four categories from the over fifty available; (i) Escherichia coli, (ii) Staphylococcus aureus, (iii) Streptococcus pneumoniae and (iv) the remaining cases grouped as 'Other' micro-organisms. There are, therefore, four categories to be classified.

Only data records with complete feature vectors were selected (no missing values). The feature vectors contained 51 variables; two being continuous interval valued and the rest binary. The number of cases for the training and test sets was 1664, both being randomly selected from the database.

A range of measures were taken to characterise the data and are presented below in table 5.2. From these measurements it is possible make a more informed selection of potentially successfully algorithms.

Since the domain is that of medical decision making, the system should offer as much insight to the classification process as possible. This issue has been raised by Hart and Wyatt (1990) where they conclude that *'black box systems appear more likely to attract strict product liability, which would be applied no matter how much care was taken in development and testing...One possible role, therefore, for such systems is as indicators of the maximum attainable classification accuracy, to encourage those developing more transparent systems'*. Lipscombe (1989) surveyed the small number of AI systems that have actually progressed to the point of clinical usage and proffered that the way forward is for '*human-directed "decision support" systems*'.

5.5.2 Algorithms selected for experiment

The literature review uncovered the following two potentially successful ANN classifiers: Learning Vector Quantization (LVQ) and Feedforward Networks. The k-nearest neighbour (*k*-NN) algorithm has proved to be successful in many classification tests and is selected here as a representative statistical technique for comparison. Several characteristics of the septicaemia problem are included in the algorithm selection process; the large data set (N=1664), the high dimensionality of feature space (f=51) , a fast classification speed required and some comprehensibility of the classification process.

The train-and-test data sampling method is adopted for training and evaluation of resulting classifiers.

5.5.3 Feedforward Networks: experiment results

Experiments with the feedforward architectures prove unsuccessful. There is no evidence of weight values within the networks converging to useful values despite using several training algorithms (backpropagation and variants) and numerous topologies. It may be possible to uncover successful feedforward architectures using a more exhaustive search of the available parameters (network topology etc.). A viable method for achieving this would be to use Genetic Algorithms to generate the architectural variations.

5.5.4 Learning Vector Quantization (LVQ): experiment results

Details of the LVQ experiments can be found in Appendix F and a summary of the experiments comparing LVQ to k-NN can be found in Worthy et al. (1993). It was found that the accuracy of LVQ solutions varied for classes and the optimum solution yielded a balanced accuracy rate for all classes. It was particularly easy to obtain an unbalanced solution with the largest class having 100% accuracy and other classes with 0%. Figure 5.6 illustrates some of the better LVQ solutions and the 'trade off' between class accuracies. The twenty experiments are presented with class accuracies shown relative to the 1-NN accuracy rates (i.e. solutions showing better performance than the 1-NN algorithm have all four class accuracies greater than 0 on the vertical axis). The accuracy rates are calculated for the test data set..

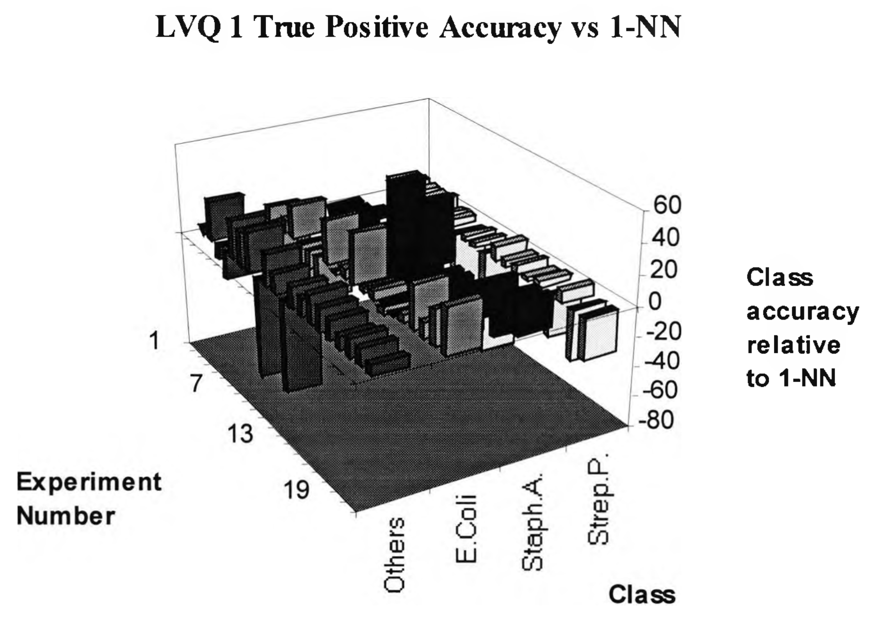


Figure 5.6 -Variation in class classification accuracy for LVQ solutions

Class	Others	Escherichia Coli	Streptococcus Pneumoniae	Staphylococcus Aureus	Std.	Avg.
1-NN	66.90	55.87	44.77	73.30	62.20	60.21
Best LVQ	73.45	85.92	66.19	46.03	69.53	67.90

Table 5.5 Class and average accuracies for Best LVQ solution and 1-NN.

Class	Others	Escherichia Coli	Streptococcus Pneumoniae	Staphylococcus Aureus
Others	582	141	114	45
Escherichia Coli	142	195	10	3
Streptococcus Pneumoniae	100	11	107	7
Staphylococcus Aureus	46	2	8	151

Table 5.6 True and False positives for 1-NN solution.

The best result for LVQ yields an accuracy over 7% higher than 1-NN (Table 5.5). In some experiments it is found that certain classes can be around 30% higher than the 1-NN accuracy although there is always a 'trade-off' in this result by a reduction in the remaining class accuracies. This highlights the need to achieve a balanced result where the overall accuracy is achieved with no low individual class accuracies.

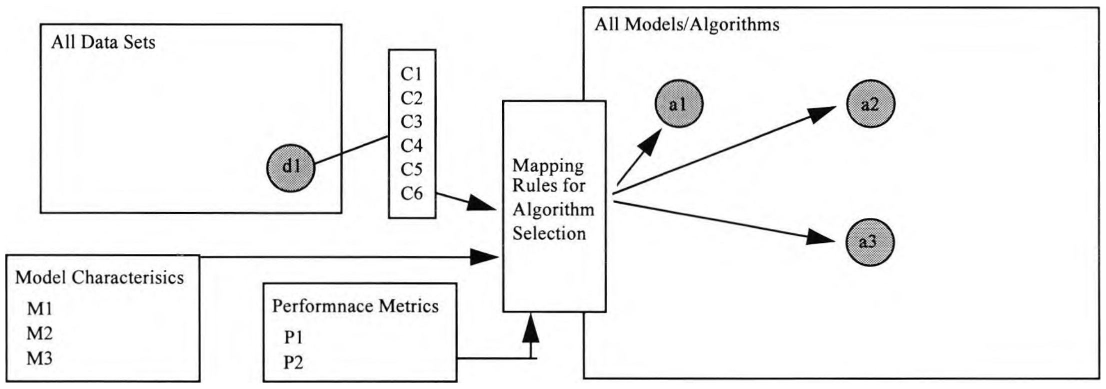
Table 5.6 shows the True/False positive matrix that was obtained for the 1-NN solution. The data to generate such a matrix was not obtained from the software implementation of the LVQ algorithm used in the experiments.

In classification mode the trained LVQ codebook vectors are used in exactly the same way as the 1-NN algorithm. The LVQ codebook vectors used for classification represent a data set reduction of 99.76% (1664 data set cases to 4 representative vectors) thereby drastically reducing computation and storage requirements of the classifier.

Repeatability of experimental results is difficult due to the numerous algorithm parameters and the stochastic methods used in training and codebook initialisation. It is possible to achieve different accuracy rates from the same initial conditions. Successful application of the algorithm therefore requires a degree of experimentation.

5.6 Summary

The classification experiments are represented in Figure 5.7 using the framework introduced in Chapter 3.



Classification Characteristic Measures

Data set
C1=Sample Size
C2=Number of Features
C3=% Binary Features
C4=% Ordinal Features
C5=Number of Classes
C5=Entropy
C6=Joint Entropy

Model
M1=Fast operation speed
M2=Descriptive as possible/Reasoning
M3=Training Time as low as possible

Performance Metrics
P1=Balanced class accuracies
P2=High Accuracy

Data Sets

d1=Septicaemia

Models/Algorithms

a1=k-Nearest Neighbour
a2=LVQ
a3=FeedForward ANN

Figure 5.7 Summary of experimental method: Classification

In contrast to the forecasting experiments in chapter 5 there is only one data set used in the septicaemia experiments. As a result no conclusions can be drawn as to the effectiveness of the data set characteristic measures in distinguishing between data sets.

The feedforward model failed to converge when presented with both a full featured and reduced featured data set. A range of topologies and learning parameters were used with no satisfactory evidence of learning in any case. This may be due to a limitation in the backpropagation algorithm on this particular data set or insufficient skill in applying the feedforward model.

The successful LVQ approach was found to be very sensitive to training parameters and could rapidly lead to an unbalanced solution where some classes were predicted with zero accuracy and others with 100%. The sensitivity of this technique makes the repeatability of results dependent on comprehensive knowledge of the technique. It was observed following the extensive series of experiments performed using LVQ that it can achieve impressive accuracy rates but the method cannot be applied without careful experimentation to achieve a balanced *and* accurate result.

ANN solutions would face significant ethical problems in being applied to automate medical decision making. There would, however, be a role for it as an advisory or decision support tool. Lipscombe (1989) asserts that there is a fundamental difference between expert systems (using a strong AI approach) and the application of formal knowledge by humans. The human possesses what he terms *"interpretative skills"* which modify the way in which knowledge is applied. The role of assistant is probably the best a machine can hope for in the medical environment where it is usually deprived of some of the factors and much of the complex knowledge required to make accurate diagnoses.

CHAPTER 6 - DISCUSSION

6.0 Precepts

In this chapter, the utility of the experimental and analytical method (chapter 3) is explored in light of the experimental results detailed in Chapters 4 and 5. The analytical method presented in chapter 3 provides a framework for conducting experiments and, importantly, comparing the performance of ANNs with established techniques. The experimental process has been defined as a sequence of stages with requisite attributes and methods in each. For example, the selection of the algorithm for experiment is one such component in the experimental process which requires characteristic measures of the data set and defined performance metrics before a selection can be made.

The choice of ANN algorithm used for experiment is based upon three sets of criteria; algorithm performance metrics, model characteristics and data set characteristics. Theoretical and heuristic knowledge are needed to select the most appropriate algorithm based on the defined criteria. The outcome of the experiment, however, is dependent on the following; (i) the quality of model selection criteria, (ii) the quality of knowledge base and (iii) the experimenters skill in applying the chosen model(s). Since these factors are dependent upon many variables it rapidly becomes apparent that reasons for the success or failure of an experiment can be difficult to identify and may well result from a combination of multiple variables, rather than a single, readily identifiable case. It can occur when applying ANNs, for example, that the experimenter will not see any convergence of weight values during training. The experimenter may conclude that the ANN is incapable of learning in this instance, but in reality his choice of learning parameters or initialisation of the network weights may be inappropriate. The application of the ANN has failed due to the lack of skill (or knowledge) of the experimentor with the particular algorithm.

Many documented ANN experiments choose not to justify the seemingly widely adopted premise "*ANNs are the most appropriate class of models for a given problem*".

Although ANN techniques have purposefully been applied to the forecasting and classification domains in this thesis, a selection procedure and comparison to existing techniques are always performed to verify the appropriateness of algorithms chosen for experiment. The generic framework for applying learning algorithms to forecasting and classification tasks is based upon a meta-method. An algorithm is not selected until an analysis of the data set has been performed. The rules for selecting an algorithm will often be heuristic but the approach provides the structure for rules to incorporate quantitative measures. Quantitative selection of an algorithm is an *objective* data based approach as opposed to a *subjective* heuristic approach. An advantage of an objective approach is that decisions are based on data from previous experimental results. As the database of cases becomes more substantial through experiment, it should support a richer set of selection rules based on a growing number of measures.

6.1 Research Objectives

The driving force for this thesis is to test the hypothesis presented in Chapter 1 (section 1.2). Two research objectives emerged from the hypothesis. The results from the forecasting experiments presented in Chapter 4 meet the first research objective put forward in Chapter 1;

'To evaluate... the performance of ANN Models compared to other potentially successful techniques in a forecasting problem'.

Foreign exchange data was used to investigate the learning ability of forecasting algorithms. The ANN selected for foreign exchange forecasting was a feedforward network trained by the classic backpropagation algorithm, which has been used so widely in the past decade. A better than chance forecast was achieved when the data was presented to the ANN using a differential rather than absolute price value. The most successful ANN solution gave a correct indication of the next price movement in over 59% of cases. The statistical techniques tested for comparison (ARMA & regression) could not model the data and the random walk model had to be used for comparison of performance. The forecasting problem is a notoriously difficult one since the foreign exchange markets have been noted as being one of the most efficient of the financial markets (and therefore practically impossible to predict).

The second research objective;
'To evaluate... the performance of ANN models compared to other potentially successful techniques in a classification problem'
is met by the classification experiment results presented in Chapter 5.

The septicaemia classification problem was of a high dimensional, predominantly binary character. The high dimensionality of the data set proved to be too demanding for most ANN techniques, resulting in only one algorithm, LVQ, performing adequately. The prominent, established technique chosen for comparison was the statistical k-NN algorithm. The best performing LVQ solution yielded a 7% greater accuracy than k-NN with the added advantage of faster operational performance from a 99% reduction in data set size and computational requirements. Although the best LVQ result offers an improvement over the k-NN performance it did require a significant number of experiments to achieve this result.

The results from both tasks evolved from a series of experiments following initial analysis of the data sets (pursuing the method described in chapter 3). This gives credence to those who claim that successful application of ANNs (and other methods) follows from careful analysis of the data before any experiments are carried out (Ripley, 1993).

The method of experimentation requires that the performance of any solutions attained be measured and compared to those of existing techniques. The method provides a framework which facilitates additional experimental data to improve the knowledge base. Further forecasting or classification problems can be tackled using the improved knowledge.

The performance of ANNs in the forecasting and classification experiments compares favourably with the performance of the established methods considered. The conclusion reached from this research is that ANNs are flexible self learning models that may be appropriate for a range of forecasting and classification tasks. A caveat remains, however, that well defined metrics and solution requirements should be specified before any pathway to a solution is explored.

The meta-method for algorithm selection is a crucial component to successfully applying any model to a given problem. The method described in chapter 3 is structured which, in the context of applying ANN models, could be criticised for being excessively rigid. ANNs have been lauded as powerful tools for discovering the hidden relationships in complex data, hence the structured method for selecting algorithms would appear to contradict the claimed ability of these techniques.

However, if ANNs are simply applied to a problem with no comparison to previous results or with no characterisation of the problem data set, it becomes apparent from the number of factors involved, that the reasons for the success or failure of the algorithm will be difficult to evaluate. It is very important that the performance of ANNs is compared to that of existing techniques in the chosen application domain. The success of any technique is *relative*.

6.2 Limits to ANN learning capabilities

Cullen Schaffer (1994) clearly and succinctly expounded the theoretical limitations of learning algorithms. Schaffer's Law states that there is no universal learning method capable of learning any concept. The tests performed in the process of researching this thesis (considering only ANN models) support the premise that there is no generic model applicable to all types of problem. This conclusion is further evidenced by the numerous studies comparing algorithm performance on differing problem types where a range of algorithms have shown superior performance. Considering a broader range of learning methods, evidence from large comparative studies such as those described in Michie et al.(1994) and Thrun et al.(1991) also provide evidence to support Schaffers Law. In these studies, some algorithms showed performance advantages with certain data sets or problem types but there was no algorithm that performed better than all over the whole range of problems. The ANN results presented in Michie et al. clearly indicate that ANNs have merit for some problems.

In the light of these findings it would seem appropriate that an experimental method, such as that adopted in this thesis, facilitates model selection for a given problem using defined metrics and performance measures enabling comparison to existing techniques. The knowledge used in model selection is derived from theoretical and experimental results which can be used objectively with data from the model selection metrics.

In tasks such as forecasting and classification, where a range of tools are available, it requires a meta-method to select a tool most appropriate to the task.

Assuming that the non-trivial task of model selection has been completed, a range of issues related to ANN learning remain. The limits to an ANNs learning capability can be viewed as a function of a) the architecture of the model and b) the learning algorithm. It is has been found in the experiments conducted for this thesis that it is rare to isolate either factor as the one limiting a successful solution.

The forecasting experiments in chapter 4 found the feedforward ANN model capable of learning some relationship between previous currency movements and the next. The representation of the problem, in this instance, was the most important factor affecting performance (see section 6.3). With the problem representation defined, however, it was found that by introducing an architecture constraint (limiting the size and layers in the ANN), improved prediction accuracy was obtained.

In chapter 5, the classification experiments resulted in a completely different ANN architecture being selected. In a parallel fashion, LVQ, performance was also found to be optimal when a restriction was made on the architecture. Namely, the number of class representative vectors was limited to one per class. With increasing numbers of class representative vectors the complexity of the decision surfaces separating categories can grow. The generalisation of the network is, therefore, at its highest with low numbers of class representative nodes.

Geman and Bienenstock (1992) explored the related issues of bias[1] and variance and concluded that *"the fundamental challenges in neural modelling are about representation rather than learning per se"*. They argue that the more bias put into a solution, the better the performance at the expense of generalisation. Conversely, the more variance, the better the generalisation. The bias, they argue, is introduced by using data representation and architecture constraints. Le Cun (1989) focused on feedforward networks and found that by constraining the network architecture, significantly better performance was achieved on an image recognition task. Bias was introduced into the design of the network by building feature extraction layers. As Le Cun notes, the ability to add bias into designs for a given problem *"is more easily said than done"*.

Some ANN models have been described as being "non-parametric" (Geman et al., 1992) in the sense that they do not require the user to specify an order of fit as would be required in a polynomial curve fitting solution, for example. In practical terms, however, the absence of parameters in the same sense of those required for curve fitting does not mean that the implementation of an ANN is parameter free.

Returning to the feedforward network for illustration, before a training run can be started the experimenter must select a host of attributes for both the network and the learning algorithm. The network must have the number of layers, nodes, transfer functions and initial weights selected. Even a basic learning algorithm will require the selection of parameters such as learning coefficient and momentum terms. All of these attributes and parameters can have an effect on performance as they change the learning capability of the network.

6.3 Problem representation and reasoning

It was observed whilst experimenting with two different types of problems, forecasting and classification, that in some instances the problems presented are very similar, if not interchangeable. Taking the foreign exchange forecasting problem as an example, the forecast required is that of a price or alternatively a direction of price movement. If a discrete rather than continuous valued range is used for the output, the problem can be represented as a classification task. Figure 6.1 illustrates the change in problem representation where a continuous valued range is converted to a discrete range having only three values; "up", "down" and "no change". More resolution can be introduced by breaking down the ranges still further. An example would be to subdivide the "up" range into "small up" and "large up". Value ranges can now be interpreted as discrete categories. The change in representation opens up a broader range of problem solving methods than would have been available if considering only forecasting algorithms.

There is therefore, the issue of *problem representation* to be considered. Taking the forecasting example given, there will be other issues to consider such as the minimum resolution of the forecast variable. A trader may require a more informative indication of forecast value than a simple "up", "down" or "no move". If that is the case then a greater number of subcategories will be needed to increase the resolution.

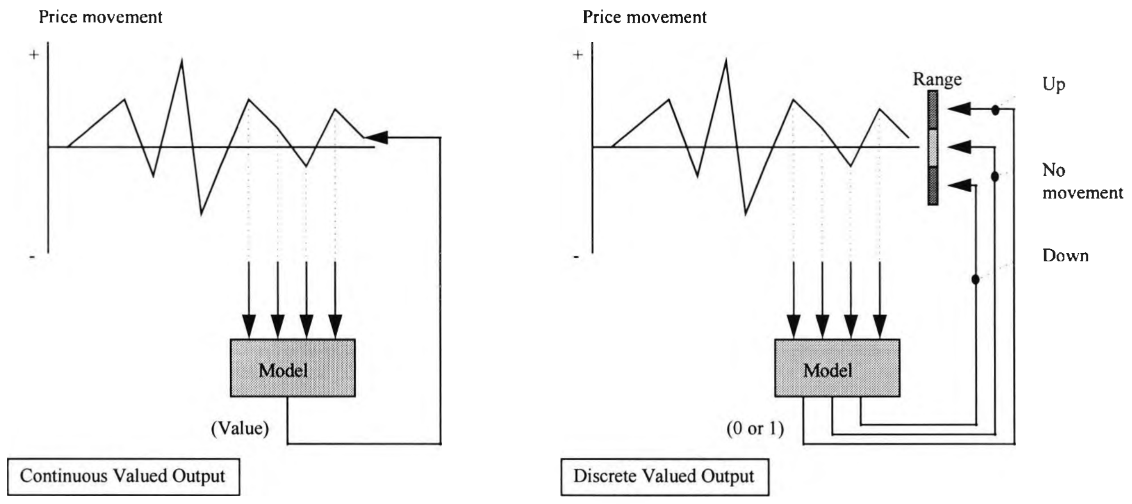


Figure 6.1 - Forecasting problem representation

Appropriate problem representation is noted as the key in achieving a good solution when using conventional Artificial Intelligence (AI) approaches (Winston, 1992). Representation is a factor not investigated in depth in the research work for this thesis,

[1] 'Bias' is not to be confused with the term used in control theory. Geman and Bienenstock use the term in the context of the 'bias \ variance dilemma' in model based or non-model based estimators. The dilemma is faced, for example, when attempting to fit polynomial functions to a curve with added noise.

but is likely to be of significant effect when applying ANN (or other) models to a given problem.

The ANN models developed and discussed in this thesis can be classified as sub-symbolic. The ANNs, in operation, do not provide any symbolic explanation of how they generate results. Analysis of the models does not yield any meaningful set of rules or relationships between output and inputs. This places ANNs at a definite disadvantage when compared to the deductive and explanatory capabilities of classic rules based AI systems.

The need to understand the internal representations of ANNs creates a paradox. Most ANNs are comprehensible only on the small scale when they have few layers and processing elements. Their computational capability is essentially proportional to their size. Increasingly complex computational tasks having a reasonable numbers of inputs and outputs will require networks that rapidly become analytically opaque. Symbolic processing systems are more scaleable from an understanding viewpoint in that they can provide an interpretable explanation of their 'reasoning'.

6.4 The utility of a meta-method

The process of applying ANNs in this thesis can be viewed as a framework or meta-method. The application of an ANN model is encompassed by the meta-method which provides mechanisms for algorithm selection and performance evaluation. This is analogous to the use of meta-languages which are used to discuss other languages or systems.

The benefit of a meta-method is that it adds understanding to the application of ANNs by allowing the problem data set, choice of ANN model and the eventual performance of the chosen ANN, to be described in terms of the metrics. As the number of experiments increases, sufficient quantitative and theoretical results can be amassed to create application rules based around the metrics.

The concept is illustrated below for a classification problem using a small set of metrics, algorithms and rules. In AI the meta-method would be termed a Frame Based (Winston, 1992:180) approach to gathering knowledge about the problem.

Metrics

[C1:No. classes]=4

[C2:No. cases]=1000

[C3:No. Features]=10

[C4:% Binary Features]=25

[M1:Fast Training Time Necessary]=True

Algorithms Available

[A1:k-NN]

[A2:LVQ]

[A3:Feedforward network using BP learning algorithm]

Rules

Each of the rules have either a heuristic (H*n*) or theoretical (T*n*) basis.

[H1] If ([No. cases] / [No. classes] > 10) and ([No. cases] / [No. Features] >10) then sample size is large enough to train and test an ANN.

Use [A1], [A2] or [A3]

[H2] If [Fast Training Time Necessary] and [N. classes]*[No. Features]>15 then do not use Feedforward model.

Do not use [A3]

[H3] If [% Binary Features]>80% then use LVQ or k-NN (It was found by Michie et al. (1994) that these two algorithms performed better than feedforward nets on data sets containing predominantly binary feature data).

Use [A1] and [A2] in preference to [A3]

A further useful extension to rules would be to include those that apply to specific ANN models. Again the rules would be a mix of heuristic and theoretical which is often not made explicit in the ANN literature. Typically, successful experiments are dependent upon the skill and knowledge of the experimenter.

It becomes obvious that some rules may require modification or further metrics as the number of algorithms is considered. Using rule H2 as an example, "Fast training time" has meaning in the context of the limited number of models available as it has been found that the feedforward model takes much longer during training than LVQ. If the number of models is increased a better rule would be to use a more meaningful metric. In this instance it would be more appropriate to use a maximum number of mathematical calculations allowed (measured in a unit such as FLOPs[2]) rather than the somewhat imprecise "Fast training time".

A principal drawback of a meta-method is that it adds another layer of complexity to what may already be a complex problem. For knowledgeable researchers this may be regarded as unnecessary and in certain cases a hindrance. This research set out to review ANNs from an applied perspective where the emphasis was to explore the capability of ANNs *compared* to other methods and on *real* data. The utility of the meta-method in this context lies in a) the capacity to add previous experimental and theoretical knowledge into the process in a structured way and b) the capability of selecting ANNs and assessing the performance in a more objective way.

[2]FLoating point OPerations (FLOPs) is a measure of the number of floating point calculations required by an algorithm on a serial computer. The power of serial computers is often described by the number of FLOPs they can process in a second. Typically the unit is the Mega-FLOP (MFLOP). The measure would be inappropriate if the algorithm is implemented on a parallel computing machine.

CHAPTER 7 - CONCLUSIONS

7.0 The objectives

A research objective was put forward in the introduction derived from the hypothesis and research aims. This last chapter draws conclusions on the objective, what contribution to knowledge has been made and areas of further work.

The objective was to devise a framework for selecting, applying and evaluating the performance of ANN models. The applicability of the framework was to be tested by applying it to two problem domains. The experiments followed the meta-method put forward in Chapter 3 which provided an objective framework for selecting and evaluating the performance of models on a given data set. As discussed in Chapter 6 the framework enabled the performance results of the ANN models in both problem domains to be compared objectively with established techniques. The framework also facilitated objective selection of algorithms for experiment as the development of rules from results demonstrated.

It can be concluded that the framework provides an objective meta-method for experiments and resulting performance results show that for the problem domains chosen, ANN models were indeed appropriate solutions. Further to this, given that established techniques were outperformed (on the chosen metrics) by the ANN models it can be concluded that ANN models in these cases were the best solution from the algorithms selected.

The hypothesis put forward at the start of this thesis is supported by the experimental results. There are, however, a number of issues that have been discussed in Chapter 6 that prevent firm conclusions being drawn;

(i) The clear definition of metrics before selecting models is key to evaluating the performance of ANNs applied to a given problem domain. In the absence of such metrics it is very difficult to consider the performance of an ANN (or other technique) objectively. Different performance metrics might yield different results and conclusions.

(ii) As important, characterisation of the domain data set enhances the evaluation of performance by giving insight into the properties of the data set which may make some models more disposed to successful application than others. A model can be selected where it's bias is suited to the data set.

(iii) The choice of the problem representation was found to significantly affect the results in one of the problem domains. Careful consideration of the solution requirements and how they and the data set can be represented will affect the overall choice and performance of algorithms. The meta-method used in this thesis has utility in comparing models applied using a similar problem representation, but does not provide any insight into how an optimum problem representation should appear.

The research illuminated important issues relevant to ANNs considered for general application to problem domains;

(i) The lack of transparency in ANN solutions compared to other classes of models (e.g. fuzzy systems and rules based systems) can be a problem in some domains. The financial application did not require explanation of the predictions made by a model which is in contrast to the ethical \ legal implications of using such a model in the medıcal domain.

(ii) Skill is required to apply the models successfully. Although only two types of ANN model were eventually applied to the problem domains, it was found that numerous learning and architecture parameters had to be set. None of the models used were self configuring and the knowledge needed to set parameters is achieved through experience.

Further research is required to confidently choose the most appropriate ANN (or other class) model for a given application.

This thesis makes a contribution to the application process in its provision of a generic meta-method for applying models to problem domains. Emphasis is on performance measures which facilitate objective comparison of models across classes. The work of Michie et al.(1994) was the only text that came to light at the end of the research where a similar method had been employed for classification algorithms.

7.1 Future work

Pursuing the benefits of the meta-method put forward in Chapter 3 and discussed in Chapter 6, future research could improve its utility. For each domain (classification and forecasting) in which the framework has been used as the knowledge building method;

(i) The number of models and classes of model applied to the data sets need to be increased and new performances figures calculated with the data already gathered.

(ii) The number of data sets should similarly be increased. The range of possible data sets an algorithm could be applied to is infinite but a wide range of library data sets will enable new data sets to be characterised and appropriate algorithms selected with increasing confidence.

(iii) Refine and develop the characteristic measures so that the meta-method has suitably rich and distinguishing terms with which to construct rules.

(iv) Where data or theory provides evidence, develop and extend the heuristic and theoretically based rules using the characteristic measures as inputs to those rules.

(v) The role of data manipulation prior to experiment (pre-processing) should be explicitly defined in the framework. This is closely related to the issue of problem representation discussed earlier and the use of data set characteristic metrics.

Outside of the scope of the framework, developments discussed in Chapter 6 that have an impact on the application of ANN models in real world problems should be explored in more detail;

(i) The creation of models with hybrid architectures. This can be pursued in both a modular (distinct functional components) and integrated fashion (merged models). The modular approach has already made an appearance on the commercial scene with products such as "Clementine"[1], a data mining tool. Clementine allows developers to analyse large databases by mixing machine learning and ANN modules for processing the data. Modular hybrid architectures use the most appropriate components for given functions within the architecture.

The integration approach has been pursued both within the ANN paradigm and in combining ANNs with other classes of models such as fuzzy systems and genetic algorithms. The strategy behind this design method is that the characteristics of different models can sometimes be combined if their architectures are merged into one functional unit. The result of the process is a new form of model.

A key advantage of fuzzy logic approaches compared to ANNs is the interpretability of the rules that determine the relationship between inputs and output(s) in the system of interest. ANNs, however, have an advantage over fuzzy systems as their learning algorithms enable them to develop a model of the system of interest through data based training as opposed to rules derived by a (human) expert.

[1] Clementine a data mining tool with ANN and rule induction components. It is produced by Integral Solutions Limited, Basingstoke, UK.

If ANN learning algorithms can be applied to architectures containing fuzzy representations then more comprehensible models may evolve. Attempts have been made to this end, producing fuzzy rule sets that provide a degree of reasoning (Kosko, 1991). The relative performance of such models compared to ANN models needs to be explored further.

(ii) AI or fuzzy rules to provide more complex reasoning than the simple rules constructed so far in the meta-method. This will require a larger database of model performance results on characteristically different data sets.

(iii) The development of reliable, modular code for models to facilitate easier experimentation. A frustration arising from the variety of ways in which algorithms are implemented is that much of the available code is stand-alone and difficult to integrate. The adoption of C as the *de facto* language in academic circles has helped but there is still some way to go before new models can be tested reliably in a simple 'modular' fashion. If this undoubtedly useful technology is to penetrate the world outside of academia then this will be a particularly important factor. The development of an application framework into which algorithm modules can be assembled would be of enormous benefit to experimenters. The situation is analogous to computer construction where integrated circuits (ICs) from different manufacturers can be combined to create solutions of choice. This capability is a direct result of IC manufacturers exposing the often complex functionality of their devices in standard component form. Although research into hybrid architectures where the functionality of differing models is combined would not benefit from this approach, it would be very productive for experimenters and application developers who build, test and evaluate the performance of ANN models.

APPENDIX A - Characteristic measures of data sets for forecasting

The characteristic measures for forecasting are defined in chapter 4, Table 4.1. This appendix details the equations, algorithms and implementation for these measures. Following the detailed definitions the results for the data sets investigated are displayed.

A.1 Standard deviation (SD) - σ

The standard deviation is calculated for a moving window along the time series. The result is a series of values which may be graphed. The graph allows analysis of the change in SD of the time series with time. The SD calculation and graphing are automated in a software utility developed and implemented in Visual Basic. The SD is calculated for a window size of 1+INT(0.1*(Time Series Length)). The SD graph therefore lags the time series plot by this number of time steps. Formally, SD is calculated as:

$$\text{SD}=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2} \quad \text{where } n \text{ is the window size.} \qquad (1)$$

A.2 Mean - $\bar{x}$

The means for a time series is calculated using the same recursive procedure outlined for the standard deviation. Formally, the mean is calculated as:

$$\text{Mean}=\frac{1}{n}\sum_{i=1}^{n}x_i \quad \text{where n is the window size.} \qquad (2)$$

A.3 Stationarity

Stationarity is an important characteristic of that is necessary for many time series models. A series is said to be stationary if its mean and standard deviation remain constant with time. The graphs of these measures are therefore used to judge the *degree* of stationarity.

A.4 Autocorrelation Function (ACF)

The autocorrelation function measures how correlated a time series is with itself through time. This is calculated by lagging the time series ($t-\tau$) and performing a standard correlation against the unlagged series (t). By using a series of lag values a graph can be plotted of the autocorrelation. The ACFs in this study are calculated by the MINITAB statistical package. Formally, the ACF is calculated as:

$$ACF=Cov(x_t, x_{t-\tau}) / [Var(x_t)*Var(x_{t-\tau})]^{1/2}$$

where Var() is the square of the standard deviation defined above and Cov() is the covariance which formally is calculated as:

$$Cov(x_t, x_{t-\tau})=\frac{1}{n}\sum_{i=1}^{n}(x_{t+i}-\bar{x}_t)(x_{t+i-\tau}-\bar{x}_{t+i-\tau})$$

A.5 Histogram

The histogram is a very simple analysis tool but gives invaluable insight into the frequency distribution of the data. It will for example show if the data is normally distributed which is an assumption made for many statistical measures (such as correlation coefficients).

A.6 Phase Diagram

The phase diagram is a graphical representation of the phase space of the time series under investigation. The series can be regarded as the output of a dynamical system where the amplitude of the signal x_t and subsequent differentials of the amplitude (velocity, acceleration and so on) represent the dynamics.

The two dimensional graph consists of a series of points where the co-ordinates are defined by x_t and Δx_t (e.g. the amplitude and velocity). This tool has been used to great effect in the investigation of chaotic systems (Gleick, 1988; p.134; Feigenbaum, 1983; p.34; Denton et al., 1990) and is a useful tool for investigating underlying order in seemingly random systems. The plots in this thesis are created using spreadsheet software.

A.7 Embedding Diagram

The embedding diagrams are two dimensional Cartesian plots of the delay space of a time series. A purpose designed software tool is used to display plots of x_t against $x_{t-\tau}$ where $\tau > 0$, and is the delay in units of time steps.

A.8 Synthetic data sets

Three synthetic data sets were generated for assisting time series analysis; a line, a sine wave and a series of random numbers. All three series contain 1306 data points so that they match the daily price currency data sets in length. The series are shown graphically below in Figure A1.

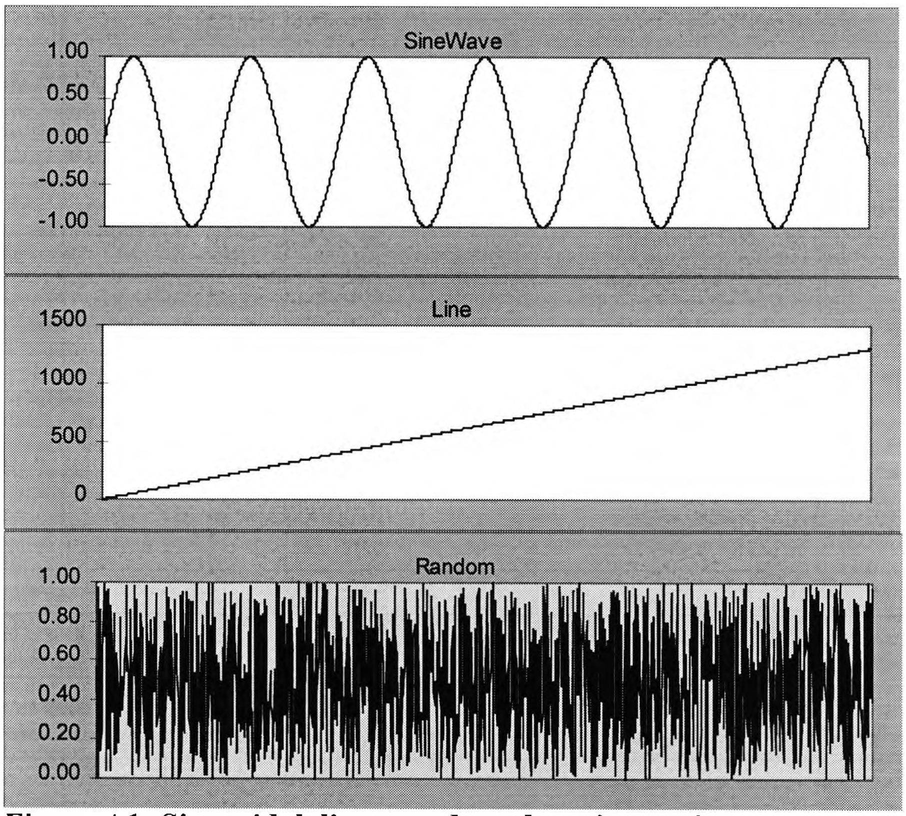


Figure A1- Sinusoidal, linear and random time series

The sine wave series is generated from a function with a period of 100 time steps. The linear time series is of the form $y=t$ where $1 \leq t \leq 1306$. The random series was generated by a computer based random number function.

Appendix B - Analysis of forecasting data

Two data bases of time series samples were selected for analysis and experimentation; the Lira closing price every five minutes (intra-day) and the Lira closing price at the end of daily trading (inter-day). The analysis and transformation of these data bases uses the tools described in appendices A and E.

B.1 - The five minute closing price data

The complete data set of closing prices for the lira (five minute data) is shown in Figure B1. The series is not a time series as there are some discontinuities in sampling where no trading takes place (such as overnight and weekends). The representation is useful however for inspection purposes. The variance and mean are also displayed, although they will be affected by the discontinuities.

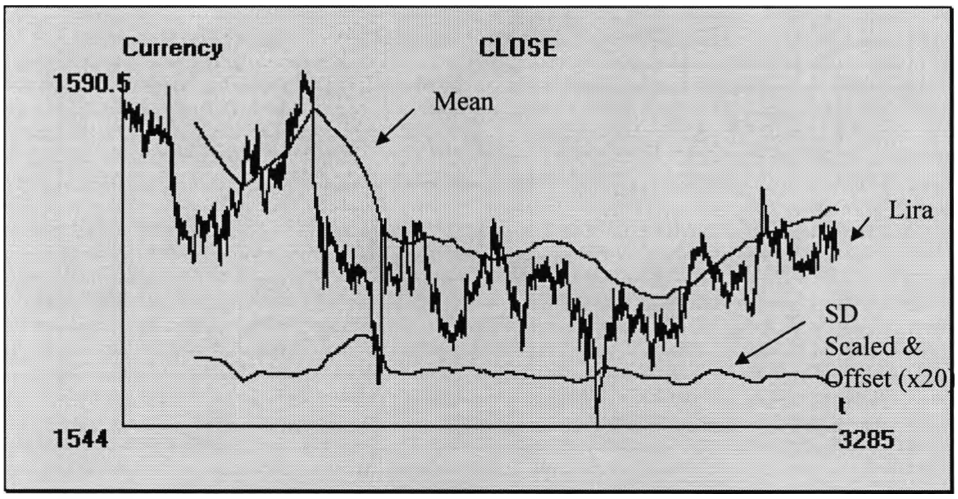


Figure B1 - Lira five minute closing price database (*not a* time series)

This data set is too discontinuous for both analysis and training purposes. The data is therefore split into multiple distinct series containing the consecutive trades found during trading days. The time series resulting from this breakdown are shown in Figure B2.

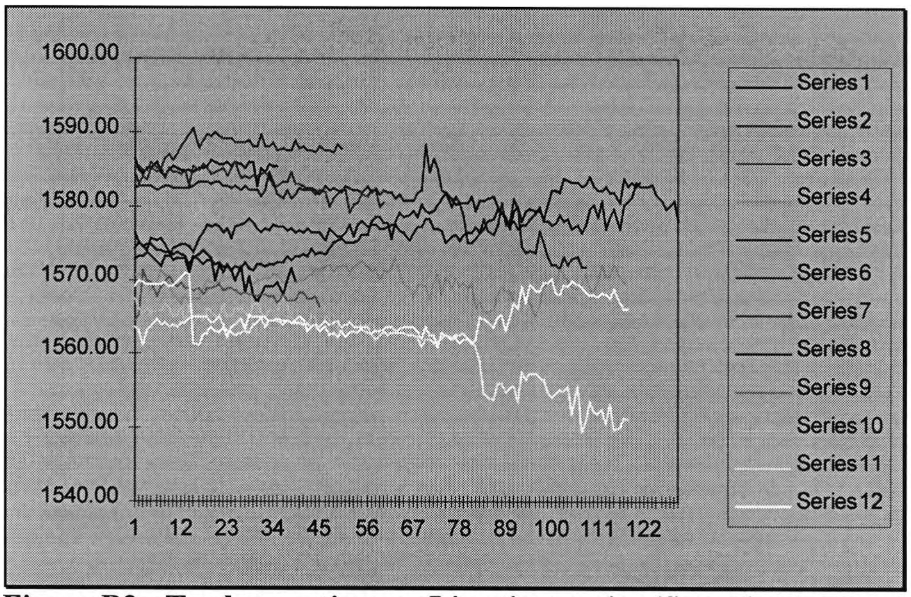


Figure B2 - Twelve continuous Lira time series (five minute samples)

With the database split into shorter series of continuous data, it is possible to attempt an analysis of the data for forecasting. The first transformation performed was to use the first differential of the series rather than the price values. The transformation yields the series shown in Figure B3.

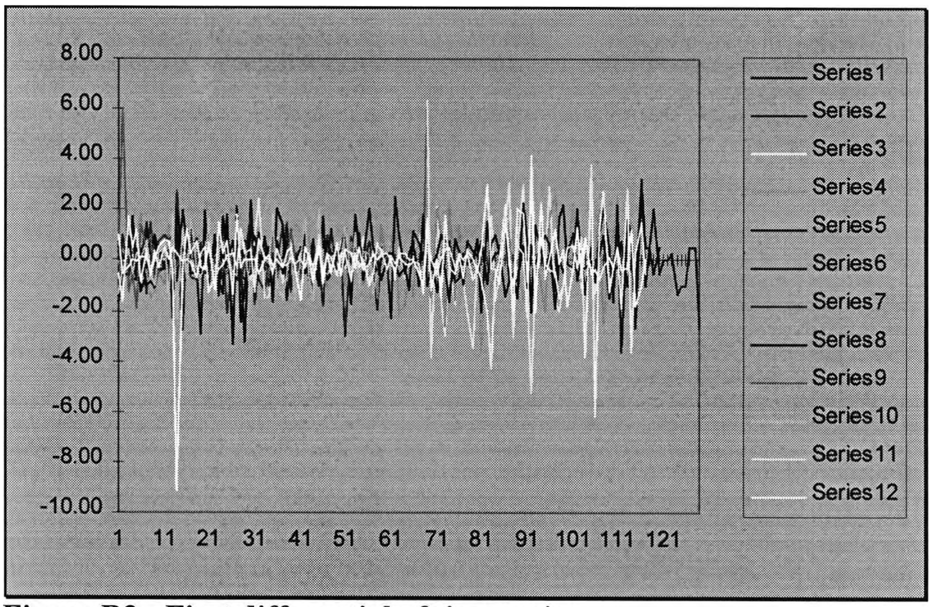


Figure B3 - First differential of time series

The series are now distributed around means that are close to zero and the differentials (i.e. the movements between successive prices) are now the large feature of the series. Investigations of three of the longer series are given in Figures B4.

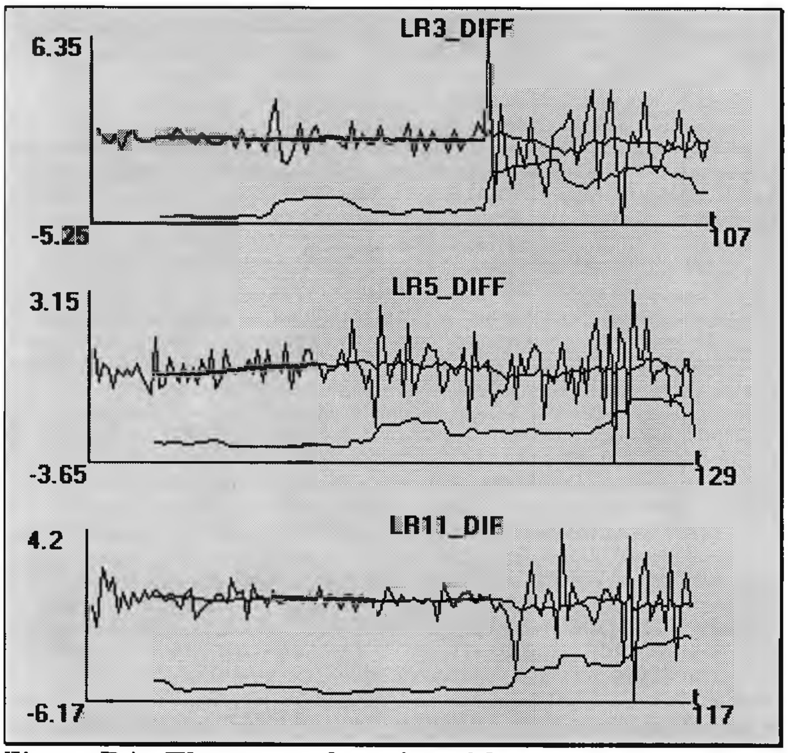


Figure B4 - Three sample series with mean and a scaled SD (x5)

The plots show that the nature of the standard deviations of the series can vary with time (varying volatility) with the mean now reasonably constant. The differentiation has, to a great extent, de-trended the time series. The series are still non-stationary. The histograms for the three series shown in Figure B.4 are displayed below in Figures B.5, B.6 and B.7.

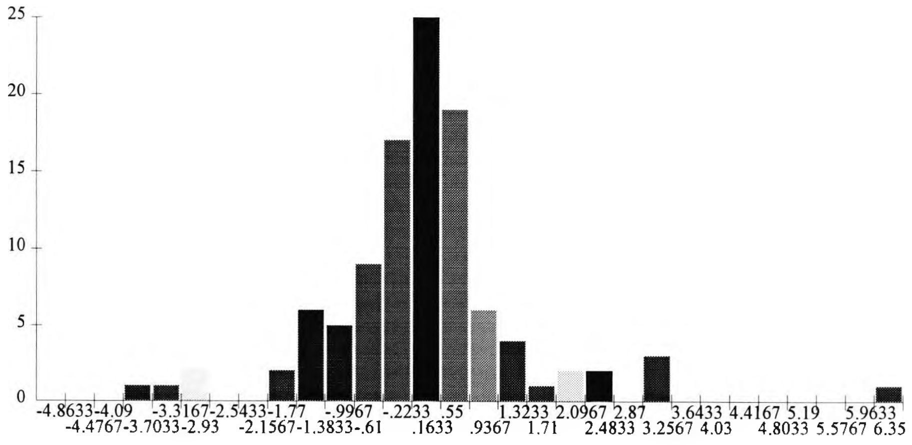


Figure B.5 - Histogram for series LR3_DIFF.

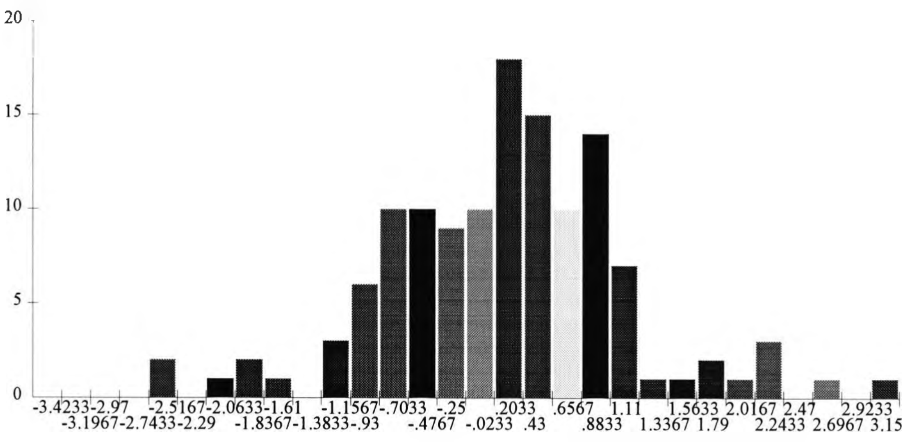


Figure B.6 - Histogram for series LR5_DIFF.

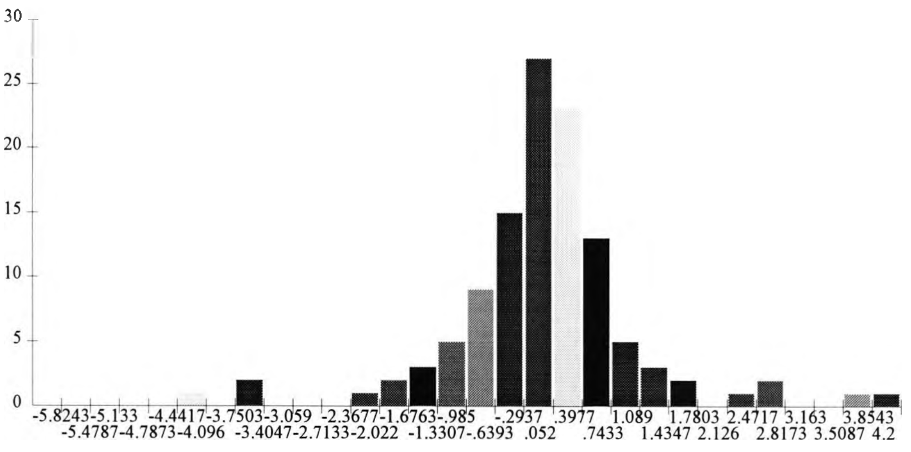


Figure B.7 Histogram for series LR11_DIF

B.2 - The daily closing price data

The daily closing price time series for the Lira is shown in Figure B8. The graph also shows the windowed mean (moving average) and an offset and scaled standard deviation (SD). The results of these basic measures can be compared to those for the artificial series shown in Figure B9.

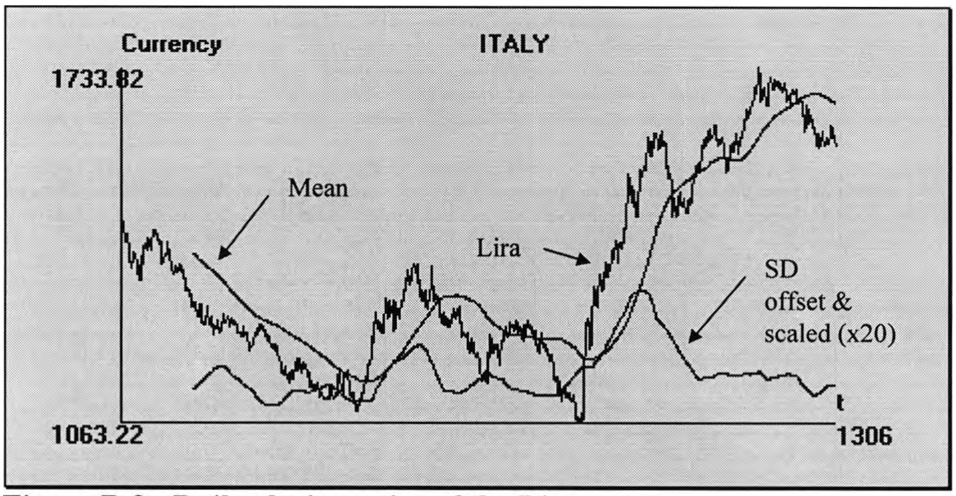


Figure B.8 - Daily closing price of the Lira

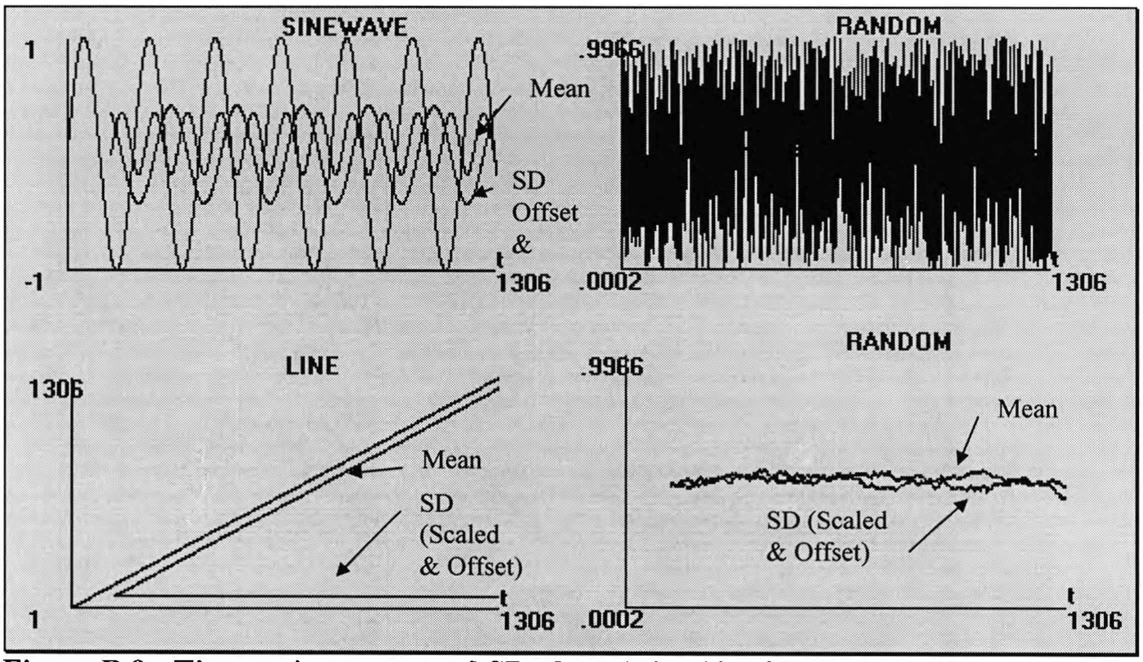


Figure B.9 - Time series, mean and SD plots (win=131, SD scaling x20)

The lira time series clearly shows that the data is highly non-stationary having large variations in the SD and mean with time. The data set was transformed to use the difference between successive values (the first differential). The time series plot for this transformation is shown in Figure B.10 with the phase plane plots in Figures B.11 and B.12.

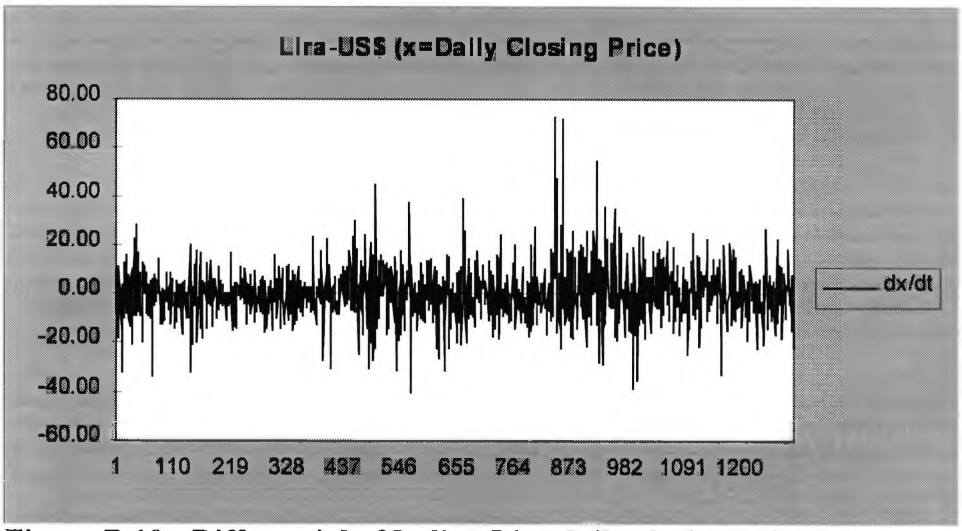


Figure B.10 - Differential of Italian Lira daily closing prices

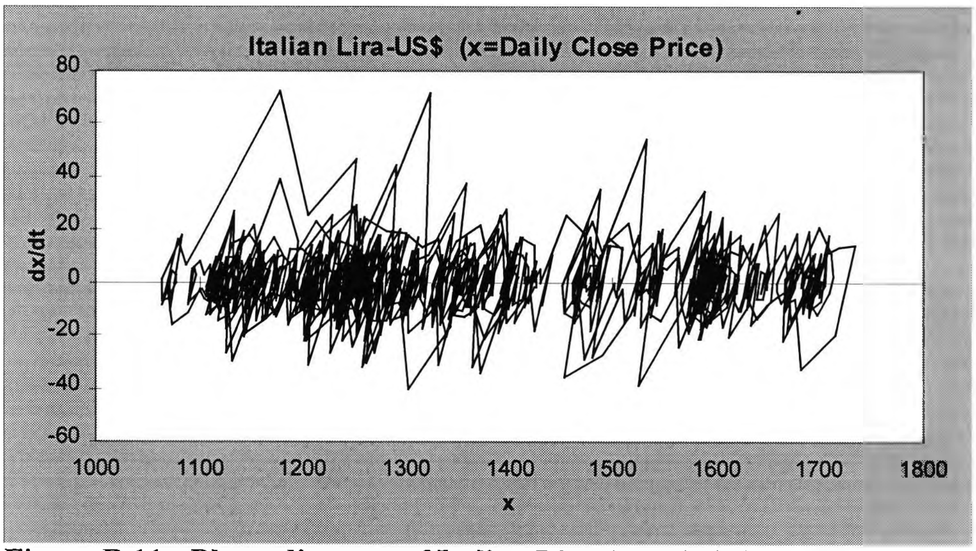


Figure B.11 - Phase diagram of Italian Lira (*x* vs d*x*/d*t*)

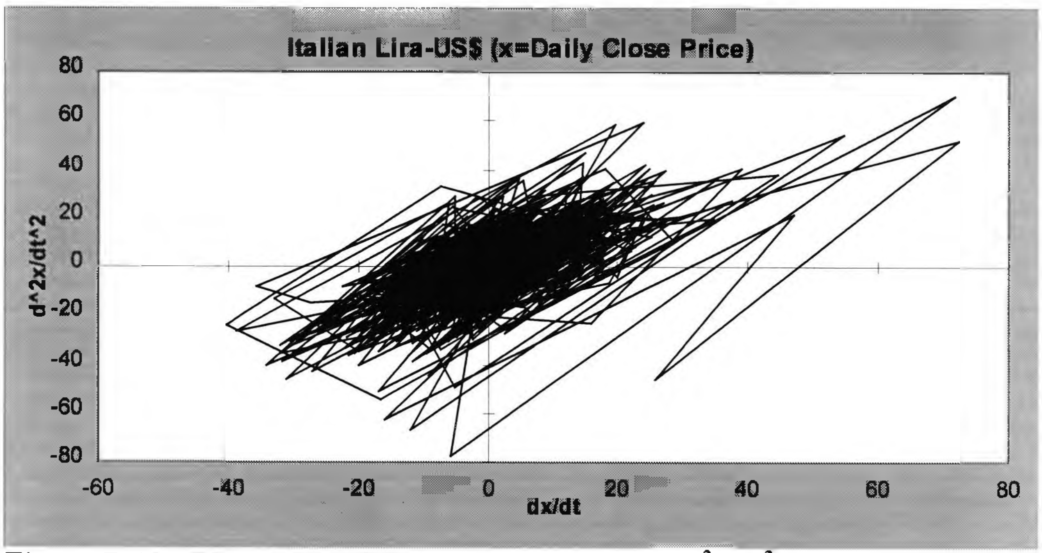


Figure B.12 - Phase plot of Italian Lira (dx/dt vs. d^2x/dt^2)

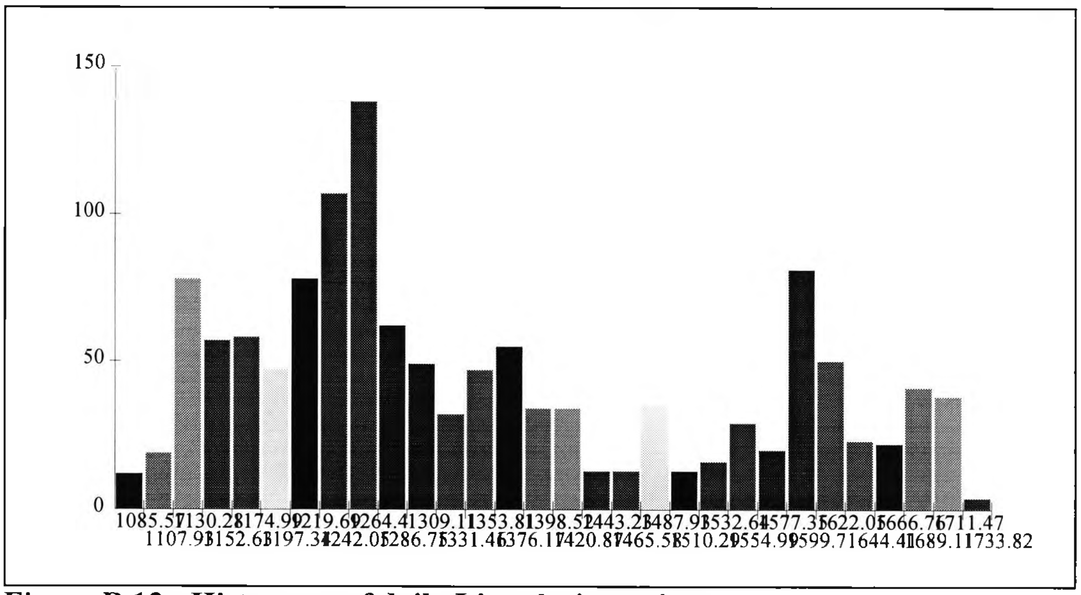


Figure B.13 - Histogram of daily Lira closing prices.

The distribution of prices for the Lira (Figure B.10) is clearly not a normal distribution.

B.3 Prediction Results (five minute data)

The following graphs show the prediction of two time series models on test data following training. The first graph is for experiment 1 and the second graph experiment 5 where improved prediction results were obtained (by both performance metrics; 'random walk ratio' and 'direction correct').

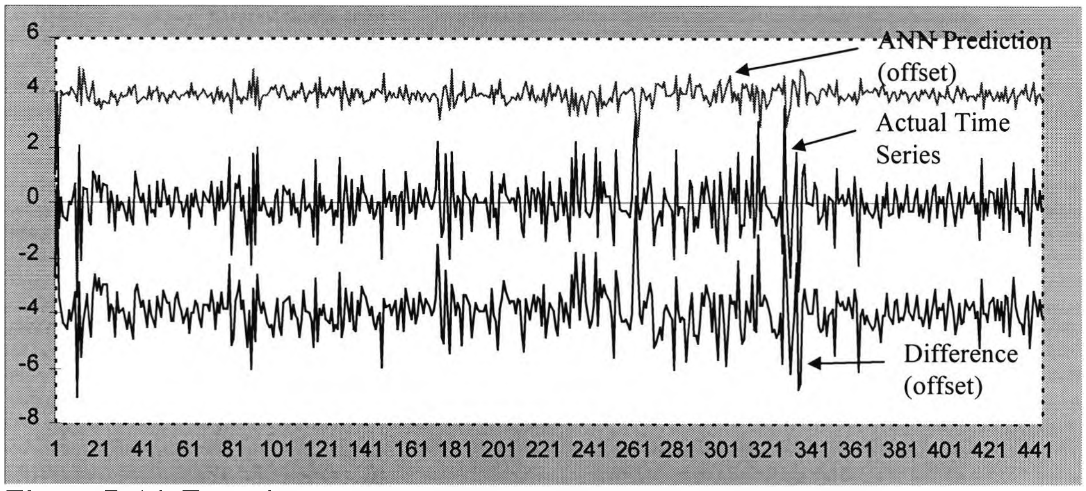


Figure B.14 Experiment 1 prediction (RWR=0.92, DC=55.63%)

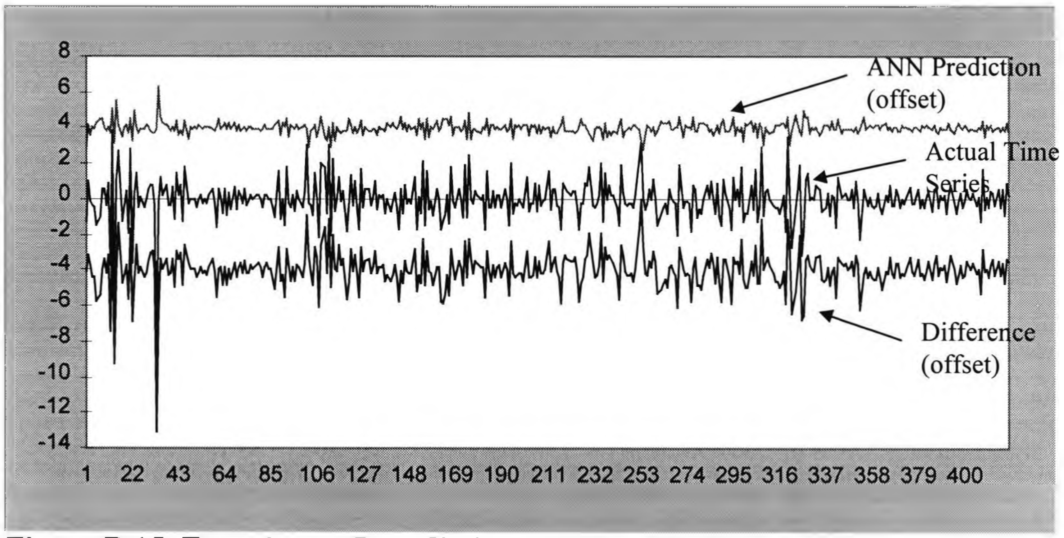


Figure B.15 Experiment 5 prediction (RWR=0.89, DC=59.29%)

APPENDIX C - Characteristic measures of data sets for classifiers

Table 5.1 lists the characteristic measures used for classification experiments. This appendix details the calculations necessary to arrive at these measures and the related Appendix D details the characteristic values of the experimental data.

C.1 Sample Size

The sample size, n, is the integer number of data samples without missing values that are available for experiment. Many evaluations will split the n available cases into two or more groups so that some examples are used for training and some are retained for testing the performance of algorithms.

C.2 Features

Features, f, is an integer measure of the number of features in each case of the sample data set of n cases. The sample set therefore contains $n.f$ data elements.

C.3 Binary Features

The integer value, f_b, is a measure of the number of binary features there are, from the total number of features f. There will be nf_b binary data elements in the data set.

C.4 Ordinal Features

The integer value, f_o, is a measure of the number of ordinal features, from the total number of features f.

C.5 Classes

The number of classes, c, is an integer measure of the total number of classifications that exist in the data set. A data set for human gender classification, for example, would have two classes (male and female).

C.6 Entropy of a feature

Entropy is a measure normally applied to physical systems. It is used here in the information theoretic sense, where it provides a measure of the probability of a value occurring for a given feature. Formally the calculation of entropy for a feature is

$$H(f) = -\sum_i p_i \log_2 p_i \qquad \text{(C.1)}$$

for the range of values, *i*, that the feature takes.

For binary features, the entropy value is easier to calculate (than continuous valued features) since it only requires totalling the occurrences of 0 and 1 (t_0 and t_1 respectively) in the data set of *n* values, and using the formula:

$$H(f_b)=-\frac{t_1}{n}\log_2\left(\frac{t_1}{n}\right)-\frac{t_2}{n}\log_2\left(\frac{t_2}{n}\right) \qquad \text{(C.2)}$$

A binary feature with little information content will have a low (close to zero) entropy value. If the feature has a continuous value of 1 or 0 then the entropy value will be 0. The highest entropy value is obtained when there is equal probability of 1 or 0 occurring, yielding:

$H(f_{equeal}) = 0.5\log_2(0.5)+0.5\log_2(0.5)=1.0$

The entropy value of a feature does not give much insight as to how useful the feature may be in aiding prediction of given class. To do this requires the calculation of joint 'feature-class' entropy values.

C.7 Joint Entropy of features and classes

Joint entropy provides a measure of the information content in combinations of features and classes. If the joint entropy of a combination of a feature and a class are high, it is an indicator that the feature may prove to be useful in a classifier. The calculation of joint entropy follows logically from the univariate case;

$$H(f_b,c)=-\sum_{ij}p_{ij}\log p_{ij} \qquad \text{(C.3)}$$

where probability p_{ij} is for the occurrence of the *i*-th value of feature *f* in class c_j.

High joint Entropy values will again occur when the distribution of probabilities of feature and class combinations is equal. Taking a combination of one class and one feature, both having binary values and with equal probability of all four combinations occurring yields the following entropy value:

$H(fc_{equal})=-0.25\log_2(0.25)-0.25\log_2(0.25)-0.25\log_2(0.25)-0.25\log_2(0.25)$

$H(fc_{equal})=2.0$

APPENDIX D - Analysis of classification data

The tables below show the entropy values and rankings of features and classes for the classification data.

D1. Analysis of feature Entropy *H(f)*

The analysis of feature entropies (Table D2) exposes which features contain high levels of information. Any features that are low in value are likely to be poor in aiding classifier methods to distinguish between classes. The following features are high in entropy;
Sex
Ha
Onab
Spec4
Uld15
Focus14

D2 Analysis of class-feature entropy *H(c,f)*

The analysis of joint class-feature entropies reveals which combinations are high in information and therefore potentially useful in classifiers. The following table lists the features that score highly for each class.

Class1	Class2	Class3	Class4
SEX	FOCUS14	HA	FOCUS10
HA	HA	SEX	SPEC4
ONAB	SEX	FOCUS6	SEX
ULD15	SPEC4	SPEC4	ULD4
SPEC4	ONAB	ULD15	ULD2
FOCUS14	ULD15	SPEC3	ULD15
SPEC14	ULD6	ONAB	SPEC3
SPEC3	ULD5	FOCUS13	HA

Table D1 - Features with highest joint entropy ranked by class and magnitude

D.3 Entropy values for test classification data set

Field	Name	Low	High	Bin.	Entropy *H(f)*	Joint Entropy *H(c,f)*
1	YR	69	89	0	-	Not calculated
2	AGE	0	99	0	-	Not calculated
3	SEX	0	1	1	0.9919	0.9507 0.5553 0.4799 0.4271
4	HA	0	1	1	0.9946	0.9599 0.5681 0.4915 0.3877
5	ONAB	0	1	1	0.8455	0.8397 0.5336 0.4495 0.3836
6	SPEC1	0	1	1	0.1603	0.5592 0.4722 0.4229 0.0000

7	SPEC2	0	1	1	0.2730	0.5861 0.4877 0.4262 0.3786
8	SPEC3	0	1	1	0.5968	0.6936 0.4962 0.4503 0.3911
9	SPEC4	0	1	1	0.9535	0.8177 0.5553 0.4550 0.4320
10	SPEC5	0	1	1	0.0782	0.5093 0.4807 0.4175 0.0000
11	SPEC6	0	1	1	0.2182	0.5481 0.4924 0.4192 0.3824
12	SPEC7	0	1	1	0.0345	0.5011 0.4734 0.4175 0.0000
13	SPEC8	0	1	1	0.1302	0.5423 0.4746 0.4175 0.3806
14	SPEC9	0	1	1	0.2935	0.5627 0.4932 0.4355 0.3778
15	SPEC10	0	1	1	0.1505	0.5609 0.4746 0.4175 0.3778
16	SPEC11	0	1	1	0.1232	0.5191 0.4722 0.4287 0.0000
17	SPEC12	0	1	1	0.1916	0.5537 0.4746 0.4208 0.3857
18	SPEC13	0	1	1	0.0345	0.5093 0.4722 0.0000 0.0000
19	SPEC14	0	1	1	0.5519	0.7077 0.5104 0.4311 0.3786
20	SPEC15	0	1	1	0.4429	0.6548 0.5078 0.4249 0.3786
21	SPEC16	0	1	1	0.0190	0.5039 0.0000 0.0000 0.0000
22	ULD1	0	1	1	0.1946	0.5384 0.4976 0.4175 0.0000
23	ULD2	0	1	1	0.3254	0.5764 0.4860 0.4243 0.3950
24	ULD3	0	1	1	0.3603	0.5861 0.4969 0.4396 0.3786
25	ULD4	0	1	1	0.2884	0.5555 0.4788 0.4184 0.4003
26	ULD5	0	1	1	0.5469	0.6598 0.5116 0.4430 0.3867
27	ULD6	0	1	1	0.4166	0.5986 0.5153 0.4305 0.3830
28	ULD7	0	1	1	0.0939	0.5167 0.0000 0.4236 0.3786
29	ULD8	0	1	1	0.3134	0.5893 0.4877 0.4256 0.3872
30	ULD9	0	1	1	0.0244	0.5011 0.4722 0.0000 0.3778
31	ULD10	0	1	1	0.5124	0.6858 0.5110 0.4275 0.3799
32	ULD11	0	1	1	0.1916	0.5384 0.4852 0.4281 0.0000
33	ULD12	0	1	1	0.0782	0.5039 0.4788 0.4166 0.3806
34	ULD13	0	1	1	0.0244	0.4979 0.0000 0.4184 0.0000
35	ULD14	0	1	1	0.1405	0.5403 0.4807 0.4200 0.0000
36	ULD15	0	1	1	0.8404	0.8286 0.5283 0.4516 0.3911
37	FOCUS1	0	1	1	0.2544	0.5696 0.5018 0.0000 0.0000

38	FOCUS2	0	1	1	0.2239	0.5258 0.4778 0.4416 0.0000
39	FOCUS3	0	1	1	0.0295	0.5093 0.0000 0.0000 0.0000
40	FOCUS4	0	1	1	0.1916	0.5797 0.4768 0.4175 0.3778
41	FOCUS5	0	1	1	0.3847	0.6523 0.0000 0.4350 0.3799
42	FOCUS6	0	1	1	0.5368	0.6846 0.0000 0.4602 0.0000
43	FOCUS7	0	1	1	0.3955	0.6671 0.4947 0.0000 0.0000
44	FOCUS8	0	1	1	0.2095	0.5696 0.4768 0.0000 0.3862
45	FOCUS9	0	0	1	0.0000	0.0000 0.0000 0.0000 0.0000
46	FOCUS10	0	1	1	0.6223	0.6204 0.4746 0.4229 0.4388
47	FOCUS11	0	1	1	0.1052	0.5363 0.0000 0.0000 0.3812
48	FOCUS12	0	1	1	0.2677	0.5955 0.4722 0.4311 0.3786
49	FOCUS13	0	1	1	0.3847	0.6162 0.4825 0.4445 0.3778
50	FOCUS14	0	1	1	0.8480	0.7623 0.5966 0.4256 0.0000
51	FOCUS15	0	1	1	0.1161	0.5280 0.4757 0.4208 0.3793
52	ECOLIONLY	0	1	1	0.7364	0.0000 0.6233 0.0000 0.0000
53	STREPAONLY	0	1	1	0.6194	0.0000 0.0000 0.5107 0.0000
54	STREPBONLY	0	1	1	0.5469	0.0000 0.0000 0.0000 0.4459
55	OTHERCLASS	0	1	1	0.9995	1.0066 0.0000 0.0000 0.0000

Table D2 - Entropy values for features and classes

APPENDIX E - Software developed to support research

Software was developed to investigate data sets and simulate algorithms. The programs developed are discussed in this appendix. All of the software was developed using Microsoft Visual Basic 3.0 (Professional). Visual Basic facilitates rapid development of Windows compatible programs having highly graphical output. Most of the software developed and described here was for visualisation of data or algorithm behaviour, hence an important requirement was for a good graphical interface.

E.1 LVQ simulation

The LVQ family of algorithms (used in Chapter 5 for classification problems) was investigated using a software simulation program. The software is flexible in that it allows user defined data sets to be configured (having two features) with up to 5 categories. Vectors (and data points) can then be initialised for each category and can be repositioned using the cursor. This is an important feature since it has been found that the initial positioning of the class representative vectors strongly affects the results of training. The vectors are trained with LVQ or OLVQ1 (algorithms written into the software) using the equations presented in chapter 2 (section 2.3.3.4). During training the vectors can be monitored as they move in the 2D feature space and a trace of their paths is drawn. An example is shown is Figure E3.

The software also has the capability to generate 1-Nearest Neighbour or 3-Nearest Neighbour decision surfaces for the original data set and the class representative vectors. This capability is very useful as it shows how well a set of class representative vectors will model the original decision surface.

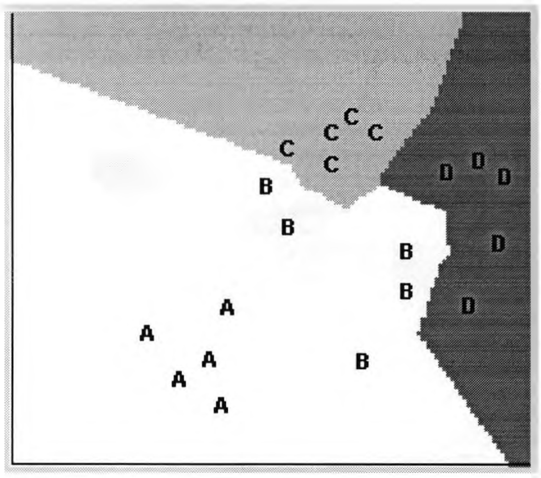


Figure E1

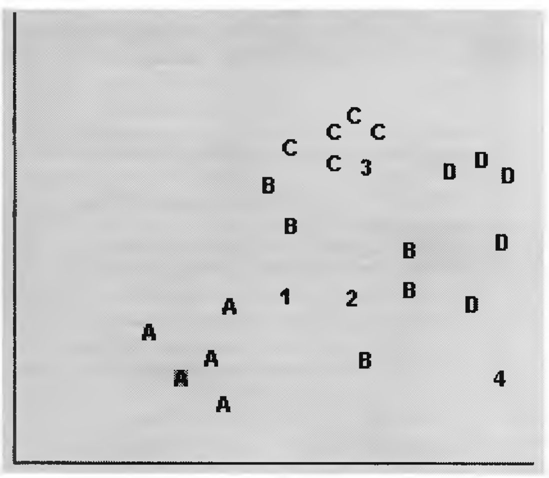


Figure E2

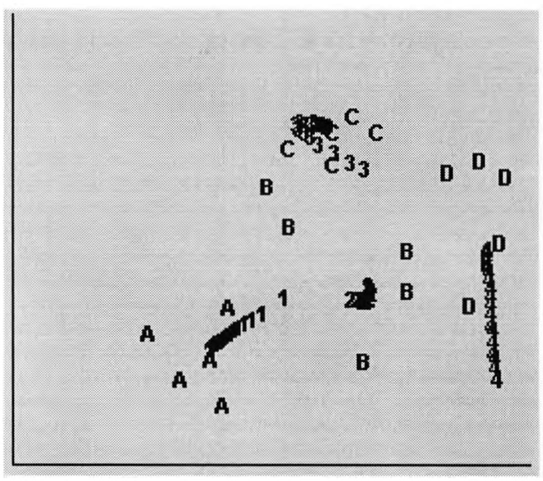


Figure E3

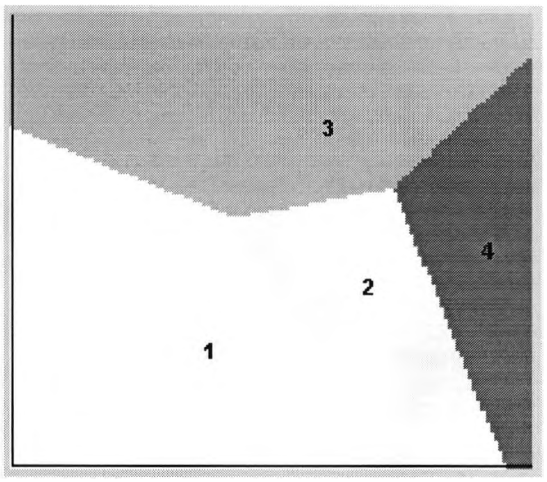


Figure E4

Figures E1- E4 Screen displays from LVQ simulation software. (E1) Synthetic four class data set with 1-NN class boundaries (E2) Data set with initial class representative vector positions (E3) vector movement during training with OLVQ1 with alpha=0.01 (E4) 1-NN class boundaries for trained vectors.

The stability of the OLVQ1 and LVQ algorithms is dependent on the initial placement of the class representative vectors. As Figure E3 illustrates, the vectors rapidly migrate to suitable positions in feature space if they are initially placed in a region majority populated by data points of the vectors class.

Figure E5 and E6 illustrate what happens when a vector is initially placed in a region populated mostly by data points of another class. As can be seen, the class B representative vector (labelled 2) is pushed away from the region by data point from the closest dominant class (which is class C). Similarly, the class C representative node is initially forced away but comes sufficiently close to the class C again so that it migrates towards a suitable position in feature space.

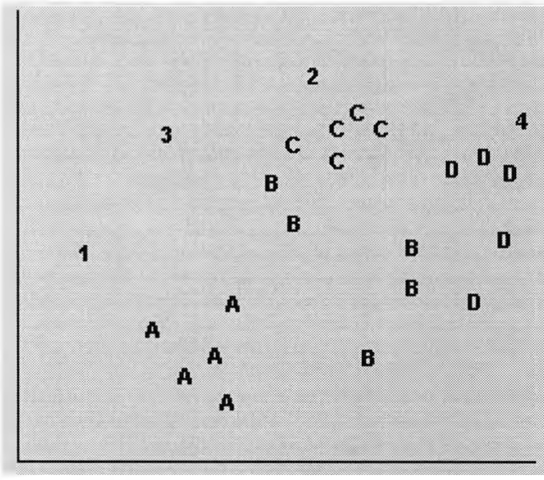


Figure E5

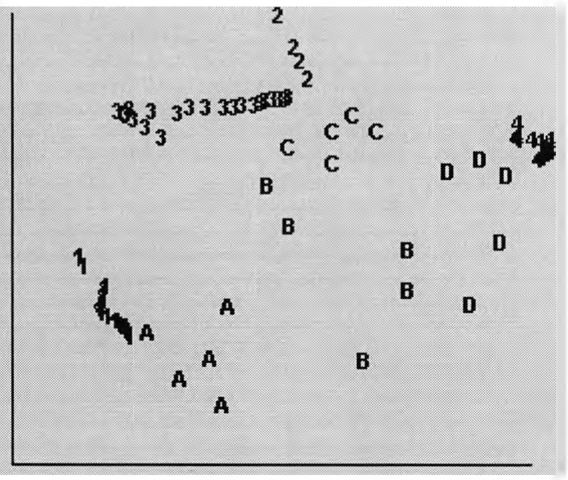
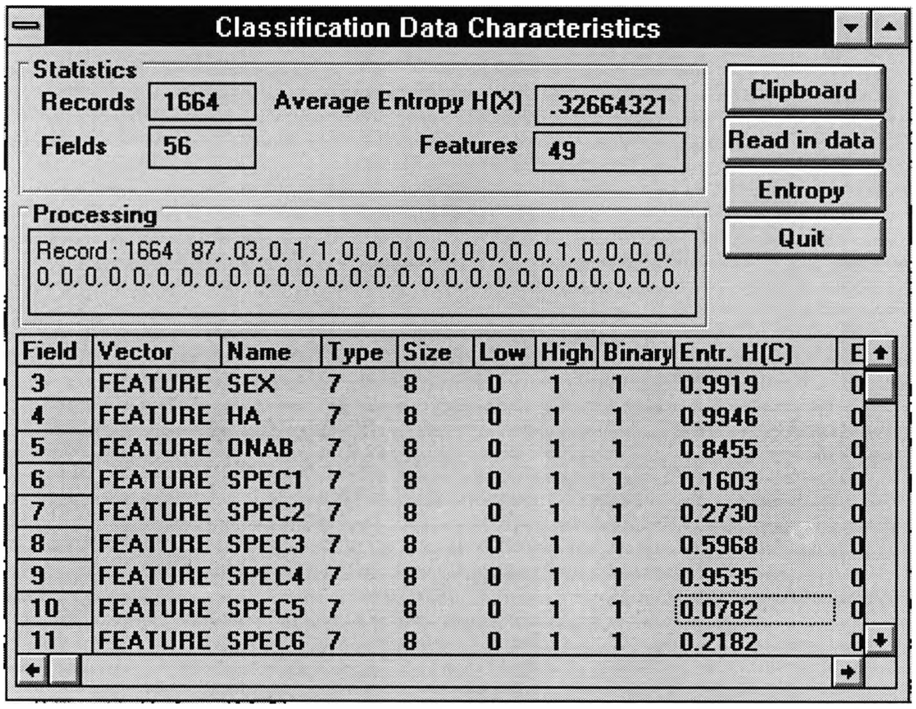


Figure E6

E.2 Classification analysis

Following the method proposed in chapter 3, data sets for classification problems require characteristic measures to be taken so that appropriate algorithms can be selected. A software program was developed to calculate the data characteristics described in Chapter 5 (the Septicaemia problem).



Statistics

Records 1664 Average Entropy H(X) .32664321

Fields 56 Features 49

Processing

```
Record : 1664  87, .03, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

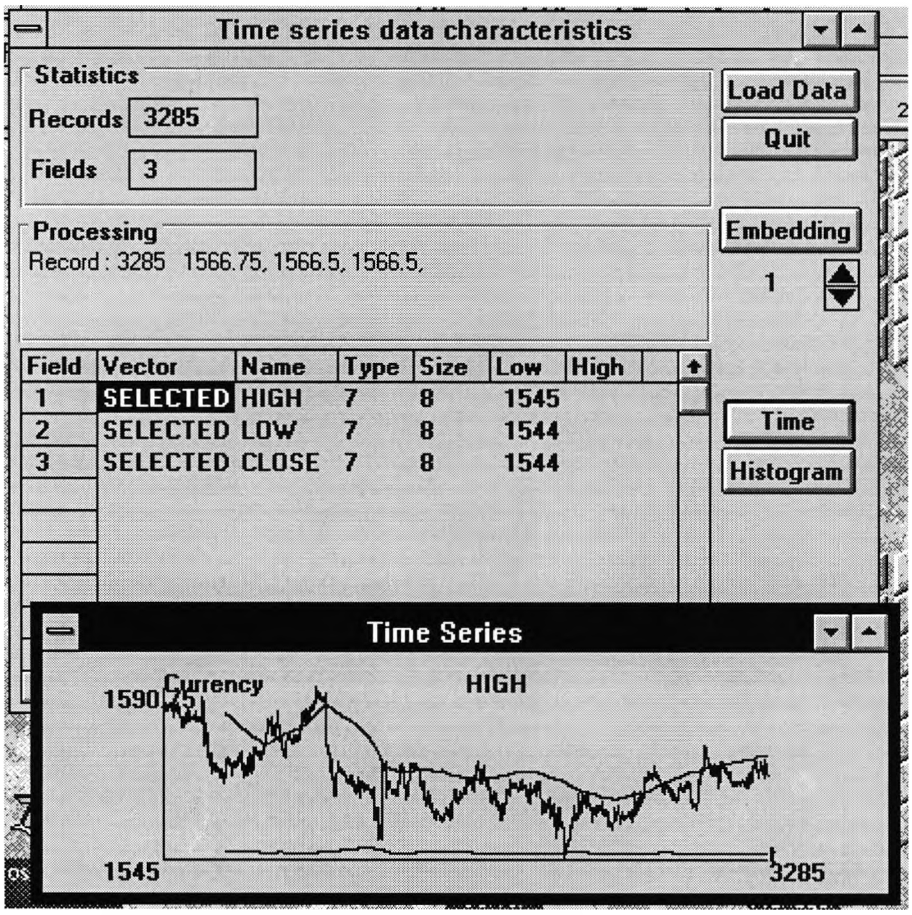
Field	Vector	Name	Type	Size	Low	High	Binary	Entr. H(C)	E
3	FEATURE	SEX	7	8	0	1	1	0.9919	0
4	FEATURE	HA	7	8	0	1	1	0.9946	0
5	FEATURE	ONAB	7	8	0	1	1	0.8455	0
6	FEATURE	SPEC1	7	8	0	1	1	0.1603	0
7	FEATURE	SPEC2	7	8	0	1	1	0.2730	0
8	FEATURE	SPEC3	7	8	0	1	1	0.5968	0
9	FEATURE	SPEC4	7	8	0	1	1	0.9535	0
10	FEATURE	SPEC5	7	8	0	1	1	0.0782	0
11	FEATURE	SPEC6	7	8	0	1	1	0.2182	0

Figure E7 - Software to calculate entropy values of classification data

The software calculates, for each feature, the highest and lowest value, whether the feature is binary, and for cases where the feature is binary; its entropy and joint class entropy with each class. The equations used for entropy calculations are those given in appendix C, sections C.6 and C.7.

E.3 Time series analysis

Chapter 4 describes a series of time series characteristics that help in identifying appropriate forecasting algorithms. A further piece of software was developed to help analyse time series data sets and calculate or display some of these characteristics.



Time series data characteristics

Statistics

Records: 3285

Fields: 3

Processing

Record : 3285 1566.75, 1566.5, 1566.5,

Field	Vector	Name	Type	Size	Low	High
1	SELECTED	HIGH	7	8	1545	
2	SELECTED	LOW	7	8	1544	
3	SELECTED	CLOSE	7	8	1544	

Figure E8 - Software to calculate time series characteristics

The software shown in Figure E8 calculates for each data series;

(1) The highest and lowest value

(2) Time series plot shown alongside the mean and variance (to investigate stationarity)

(3) Histogram (to investigate data distribution)

The mean and variance are calculated using a variable time window of length l;

$l=1+0.1*n$

where n is the number of data points in the data series. The time window the mean and variance to be calculated and charted along the data series thereby showing any change with time. A condition of stationarity is that the mean and variance of a time series do not alter with time.

A histogram can also be displayed for a given series. An example is shown in Figure E9 below.

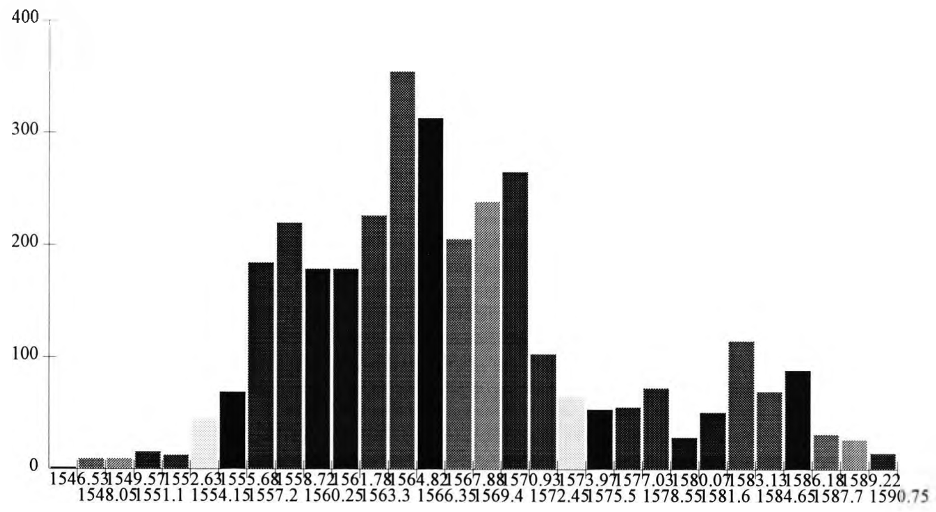


Figure E9 - Histogram of 5 minute lira time series

The time series range is divided into a sequence of 30 equal width 'bins' for which totals are generated and the histogram generated. The shape of the distribution is far from normal.

(4) Embedding diagram

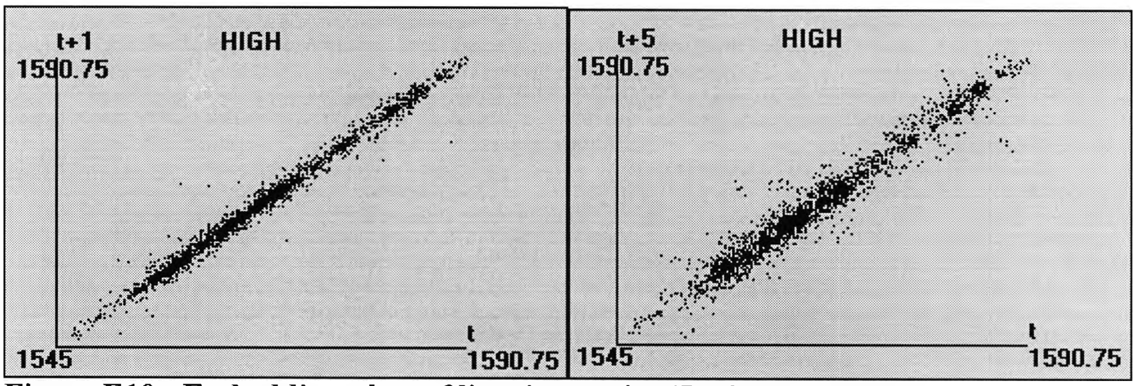


Figure E10 - Embedding plots of lira time series (5 minute sample)

An embedding diagram plots the relationship between lagged time points. The two plots in Figure E10 are from the same time series (Italian Lira 5 minute samples) each plot having x_t along the x axis and x_{t+n} along the y axis (where n=1 and 5 respectively). Embedding plots can be useful in identifying how many points are needed for a tapped delay line.

E.4 Chaos identification and simulation

Some useful concepts from chaos work have been incorporated into a software program to identify and compare the characteristics of chaotic and predictable systems. Screen shots of the software are shown below to illustrate its functionality.

The software will display the following for a given time series;

(1) A time series plot of data points and a linearly interpolated plot

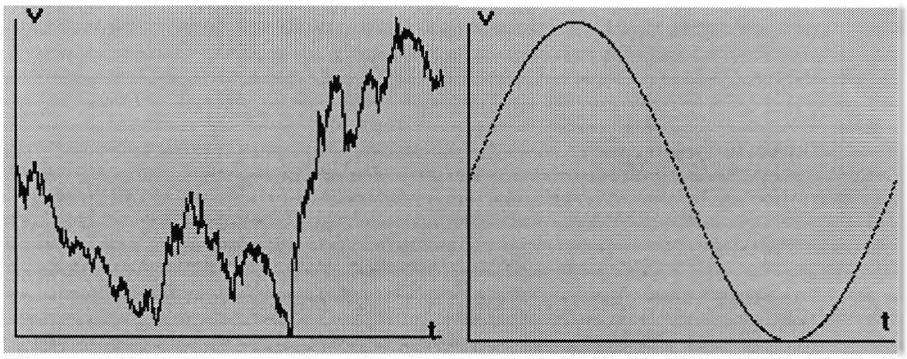


Figure E11 - Time series plot lira and sinusoid

(2) A phase diagram or phase portrait

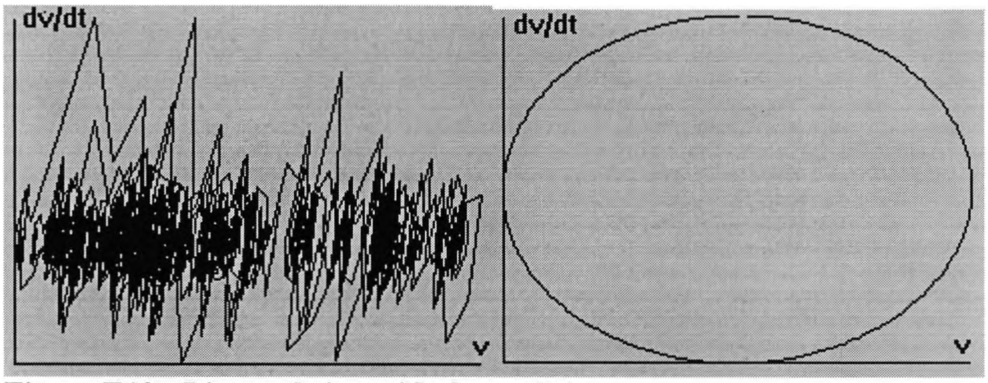


Figure E12 - Lira and sinusoid phase plot

A phase plane plot is a representation of the state space of a system. Commonly, the x-axis will be the value of the variable (amplitude) and the y axis will be the first differential of the variable (velocity). Formally for a variable v_t;

$$x_{axis}=v_t$$

$$y_{axis}=\frac{dv}{dt}$$

In systems that have cycles there will be repeating trajectories or orbits. In chaotic systems, the orbit will typically show trajectories close together at one point and further apart at others (Denton, 1990:1429;) with some form of attractor present.

The phase plane plot can be extended to any number of dimensions (more than 3 are hard to visualise) with each dimension allowing one more degree of freedom of the system being investigated to be plotted. Gleick discusses phase plane plots in two dimensions (Gleick, 1988:50).

APPENDIX F – LVQ Experiments

F.1 Software

This appendix documents the LVQ classification experiments. Two pieces of software were used:

1) Neural Works Professional (LVQ algorithm), version 4.03.
This is a commercial package offering a range of neural network paradigms with a Graphical User Interface to assist in creation of models. Graphical tools are available to monitor training in real time. The experiments conducted ran the software on a UNIX platform (Sun Sparc 2).

2) The 'Learning Vector Quantization Program Package', version 2.1 (LVQ team).
This software is freely available by anonymous ftp from the LVQ research team in Helsinki. The experiments conducted ran the software on a IBM PC platform (486DX processor running at 33MHz).

F.2 Data files

The training and test data files each have a random (mutually exclusive) population sample of 1664 episodes from the larger database of over 5000 episodes. Each episode has a normalised feature vector of 51 elements (mostly binary) to represent the following :

Variable	Type	
Year	Discrete	
Age	Discrete	
Sex	Binary(1)	M/F
Whether the infection was acquired in hospital	Binary(1)	Y/N
Whether the patient is already on antibiotics	Binary(1)	Y/N
Medical speciality of the ward	Binary()	
Underlying Disease (If any)	Binary()	
Anatomical Site of infection	Binary()	

Within the database there are episodes with approximately 55 different types of micro-organism causing the septecaemic condition. To reduce the complexity of the classification, the data set has been reduced to three common (statistically) microorganisms with the other microorganisms being grouped as a large category named 'others'. The test file population sizes of these categories is given below:

TEST FILE microorganism population sizes:

Micro-organism	(Episodes)
B - Escherichia coli	(349)
C - Streptococcus Pneumoniae	(239)
D - Staphylococcus Aureus	(206)
A - OTHERS	(870)

Filenames
TRAINING FILE : '*norm.csv*'
TEST FILE : '*normtest.csv*'

F.3 Results

The results are presented using the metrics chosen (average and standard error) for ranking. The results are compared to the nearest neighbour technique.

Results Summary for 1 CODEBOOK VECTOR PER CLASS :

CODEBOOK	A	B	C	D	Standard	Average
lvq_vect.cod	71.84	64.76	36.82	83.58	66.78	64.25
normr.cod	93.22	0.00	12.55	84.47	61.00	47.56
Hybrid1.cod	57.93	64.18	11.72	82.53	55.65	54.09
Hybrid2.cod	48.74	77.94	52.30	82.04	59.50	65.26
Hybrid2a.cod	92.64	0.00	17.15	83.98	61.30	48.44
Hybrid2b.cod	90.00	43.55	9.21	78.16	67.18	55.23
Hybrid2c.cod	94.25	27.79	0.84	78.64	64.96	50.38
Hybrid3.cod	0.69	84.81	89.54	0.00	31.01	43.76
Hybrid3a.cod	88.62	53.87	5.02	76.70	67.85	56.05
Hybrid3b.cod	80.46	62.18	25.94	76.21	68.27	61.20
Hybrid3c.cod	5.86	89.40	87.87	4.85	35.04	47.00
Hybrid3d.cod	81.38	59.03	16.32	83.01	67.55	59.94
Hybrid3e.cod	78.74	61.32	30.54	77.67	68.03	62.07
Hybrid3f.cod	86.67	51.00	13.39	79.13	67.73	57.55
Hybrid3g.cod	82.76	55.30	27.62	76.21	68.27	60.47
Hybrid3h.cod	80.00	61.03	34.73	78.00	69.29	63.48
Hybrid3i.cod	75.40	84.95	64.18	45.61	69.95	67.54
Hybrid3j.cod	79.08	65.62	33.89	81.07	70.01	64.92
hybrid3k.cod	78.28	79.13	65.62	40.17	70.25	65.80
hybrid3l.cod	73.45	85.92	66.19	46.03	69.53	67.90

1-NN	66.90	55.87	44.77	73.30	62.20	60.21
3-NN					66.81	66.07
5-NN					68.12	67.02

Results summary for VARYING CODEBOOK VECTORS PER CLASS

LVQ 1/Class	69.53	67.90	hybrid3l.cod
LVQ 2/Class(even)	70.73	66.22	2c_e6.cod
LVQ 2/Class(proportional)	71.27	67.54	2c_p8.cod
LVQ 3/Class(even)	71.21	67.85	3c_e10.cod
LVQ 3/Class(proportional)	70.43	67.36	3c_p3.cod
LVQ 4/Class(even)	70.19	67.04	4c_e5.cod
LVQ 4/Class(proportional)	69.77	67.53	4c_p5.cod

F.4 Experimental Data

• Experiment 1

Description
This network uses the 51 input vector and was trained on the Neural Ware software. For validation the generated code book vectors were transferred across to the Kohonen Software and the classification run again. The slight discrepancy in % classification rates (equivalent to 1 episode) is probably due to rounding errors during code book transfer (file was limited to 2 decimal places). The confusion matrix was generated using the Neural Ware test output file and a routine written in FOXPROW.

Software Commands
TRAINING FILE : '*norm.csv*', TEST FILE : '*normtest.csv*'
Command Line>> accuracy -din normtest.csv -cin lvq_vect.cod

Results

Confusion matrix :

	B	C	D	A
B	226	8	0	131
C	7	87	1	62
D	1	6	172	53
A	115	138	33	624

Classification accuracy:

	Accuracy A[1]	Accuracy B[2]
B	64.76%	64.76%
C	36.82%	36.83%
D	83.58%	83.49%
A	71.84%	71.72%
Overall	66.78%	66.72% (Standard)
Overall	64.25%	64.20% (Average)

CODE BOOK FILE GENERATED : *lvq_vect.cod*

• Experiment 2

Description
LVQ-1 codebook per class Network generated by the Kohonen software. Again it uses the 51 input vector with 4 classifications. The code book vectors were generated using

[1]Classification accuracy A is calculated from the codebook vectors using the Kohonen software 'accuracy.exe' with the file 'lvq_vect.cod' as the vector file and the test set 'normtest.csv'.

[2]Accuracy B is from the Neural Ware output file, from which the confusion matrix was also generated.

the Kohonen 'eveninit.exe' utility. Training took place with 'olvq1.exe' using the following parameters :
-rlen 30000, -cin norm.cod (generated by 'eveninit.exe'), -din norm.csv -cout normr.csv

Software Commands
TRAINING FILE : 'norm.csv' , TEST FILE : 'normtest.csv'
Command line>> accuracy -din norm.csv -cin normr.cod

Results

A (870)	93.22%
D (206)	84.47%
B (349)	0.00%
C (239)	12.55%
Overall	61.00% (Standard calculation)

GENERATED CODEBOOK FILE : 'normr.cod'

•Experiment 3

Description
This codebook vector file is the result of merging vectors A,C & D from 'normr.cod' and B from 'lvq_vect.cod'.

Software Commands
TRAINING FILE : N/A , TEST FILE : 'normtest.csv'
No training

Results

A (870)	57.93%
D (206)	82.53%
B (349)	64.18%
C (239)	11.72%

GENERATED CODEBOOK FILE : hybrid1.cod

•Experiment 4

Description
This codebook vector file is the result of merging vectors A & D from 'normr.cod' and B & C from 'lvq_vect.cod'.

TRAINING FILE : N/A , TEST FILE : 'normtest.csv'

Results

A (870)	48.74%
D (206)	82.04%
B (349)	77.94%
C (239)	52.30%

Overall	59.50% (Standard)
	65.26% (Average)

GENERATED CODEBOOK FILE : Hybrid2.cod

•Experiment 5

Description
This vector file was generated by using olvq1 on the hybrid2.cod file from experiment 4. The command line detailing parameters is given below :

Software Commands
TRAINING FILE : 'norm.csv' , TEST FILE 'normtest.csv'
Command Line >olvq1 -cin hybrid2.cod -cout hybrid2a.cod -din norm.csv -rlen 1664

Results

A (870)	92.64%
D (206)	83.98%
B (349)	0.00%
C (239)	17.15%
Overall	61.30% (Standard)
	48.44% (Average)

GENERATED CODEBOOK FILE : Hybrid2a.cod

•Experiment 6

Description
This solution was generated from the 'hybrid2.cod' file being further adapted by the lvq2 algorithm.

Software Commands
TRAINING FILE : 'norm.csv' , TEST FILE 'normtest.csv'
>lvq2 -din norm.csv -rlen 1664 -cin hybrid2.cod -cout hybrid2b.cod -alpha 0.05 -win 0.3

Results

A (870)	90.00%
D (206)	78.16%
B (349)	43.55%
C (239)	9.21%
Overall	67.19% (Standard)
	55.23% (Average)

GENERATED CODEBOOK FILE : hybrid2b.cod

•Experiment 7

Description
This solution again uses lvq2 but for another 1664 iterations.

Software Commands
TRAINING FILE : 'norm.csv' , TEST FILE 'normtest.csv'

```
>lvq2 -din norm.csv -rlen 1664 -cin hybrid2b.cod -cout hybrid2c.cod -alpha 0.05 -
win0.3
```

Results

A (870)	94.25%
D (206)	78.64%
B (349)	27.79%
C (239)	0.84%
Overall	(Standard)
	50.38% (Average)

GENERATED CODEBOOK FILE : hybrid2c.cod

•Experiment 8

Description
This solution is from combination of vectors A & D from 'hybrid2b.cod'
and B & C from 'hybrid2.cod'.

TRAINING FILE : N/A , TEST FILE : 'normtest.csv'

Results

A (870)	0.69%
D (206)	0.00%
B (349)	84.81%
C (239)	89.54%
Overall	31.01% (Standard)
	43.76% (Average)

GENERATED CODEBOOK FILE : hybrid3.cod

•Experiment 9

Description
This solution uses 'hybrid3.cod' with lvq2 for 500 iterations.

Software Commands
TRAINING FILE : 'norm.csv' , TEST FILE : 'normtest.csv'

```
>lvq2 -din norm.csv -rlen 250 -cin hybrid3.cod -cout hybrid3b.cod -alpha 0.05 -win0.3
```

Results

A (870)	88.62%
D (206)	76.70%
B (349)	53.87%
C (239)	5.02%
Overall	67.85% (Standard)
	56.05% (Average)

GENERATED CODEBOOK FILE : hybrid3a.cod

•Experiment 10

Description
This solution uses 'hybrid3.cod' with lvq2 for 250 iterations.

Software Commands
TRAINING FILE : 'norm.csv' , TEST FILE : 'normtest.csv'

```
>lvq2 -din norm.csv -rlen 150 -cin hybrid3.cod -cout hybrid3b.cod -alpha 0.05 -win 0.3
```

Results

A (870)	80.46%
D (206)	76.21%
B (349)	62.18%
C (239)	25.94%
Overall	68.27% (Standard)
	61.20% (Average)

GENERATED CODEBOOK FILE : hybrid3b.cod

•Experiment 11

Description
This solution is based on 'hybrid3.cod' for 50 iterations of lvq2.

Software Commands
TRAINING FILE : 'norm.csv' , TEST FILE : 'normtest.csv'

```
>lvq2 -din norm.csv -rlen 50 -cin hybrid3.cod -cout hybrid3c.cod -alpha 0.05 -win 0.3
```

Results

A (870)	5.86%
D (206)	4.85%
B (349)	89.40%
C (239)	87.87%
Overall	35.04% (Standard)
	47.00% (Average)

GENERATED CODEBOOK FILE : hybrid3c.cod

•Experiment 12

Description
This solution is based on 'hybrid3.cod' for 1000 iterations of LVQ2 and a smaller window value.

Software Commands
TRAINING FILE : 'norm.csv' , TEST FILE : 'normtest.csv'

```
>lvq2 -din norm.csv -rlen 1000 -cin hybrid3.cod -cout hybrid3d.cod -alpha 0.05 -win
0.2
```

Results

A (870)	81.38%
D (206)	83.01%
B (349)	59.03%
C (239)	16.32%
Overall	67.55% (Standard)
	59.94% (Average)

GENERATED CODEBOOK FILE : hybrid3d.cod

•Experiment 13

Description
This code book vector set is based on 'hybrid3.cod' with training using lvq2 (2000 iter.) and small window size.

Software Commands
TRAINING FILE : 'norm.csv' , TEST FILE : 'normtest.csv'

```
>lvq2 -din norm.csv -rlen 2000 -cin hybrid3.cod -cout hybrid3e.cod -alpha 0.05 -win
0.2
```

Results

A (870)	78.74%
D (206)	77.67%
B (349)	61.32%
C (239)	30.54%
Overall	68.03% (Standard)
	62.07% (Average)

GENERATED CODEBOOK FILE : hybrid3e.cod

• PHASE 2 EXPERIMENTS

•Experiment 14

Description
This code book vector set is based on the hybrid3.cod set and trained with the lvq2 for 3000 iterations.

Software Commands
TRAINING FILE : 'norm.csv' , TEST FILE : 'normtest.csv'

```
>lvq2 -din norm.csv -rlen 3000 -cin hybrid3.cod -cout hybrid3f.cod -alpha 0.05 -win
0.2
```

Results

A (870)	86.67%
D (206)	79.13%
B (349)	51.00%
C (239)	13.39%
Overall	67.73% (Standard)
	57.55% (Average)

GENERATED CODEBOOK FILE : hybrid3f.cod

•Experiment 15

Description
This code book vector set is based on the hybrid3e.cod set and trained with the lvq2 for 1000 iterations with a much smaller 'window' of w=0.1.

Software Commands
TRAINING FILE : 'norm.csv' , TEST FILE : 'normtest.csv'

```
>lvq2 -din norm.csv -rlen 1000 -cin hybrid3e.cod -cout hybrid3g.cod -alpha 0.05 -win
0.1
```

Results

A (870)	82.76%
D (206)	76.21%
B (349)	55.30%
C (239)	27.62%
Overall	68.27% (Standard)
	60.47% (Average)

GENERATED CODEBOOK FILE : hybrid3g.cod

•Experiment 16

Description
This code book vector set is based on the hybridb.cod set and trained with the lvq2 for 100 iterations with w=0.1

Software Commands

TRAINING FILE : 'norm.csv' , TEST FILE : 'normtest.csv'

```
>lvq2 -din norm.csv -rlen 100 -cin hybrid3b.cod -cout hybrid3h.cod -alpha 0.05 -win
0.1
```

<u>Results</u>

A (870)	80.00%
D (206)	78.00%
B (349)	61.03%
C (239)	34.73%
Overall	69.29% (Standard)
	63.48% (Average)

GENERATED CODEBOOK FILE : hybrid3h.cod

•Experiment 17

<u>Description</u>
This code book vector set is based on the hybrid3b.cod set and trained with the lvq2 for 30000 iterations with a very low w=0.02.

<u>Software Commands</u>
TRAINING FILE : 'norm.csv' , TEST FILE : 'normtest.csv'

```
>lvq2 -din norm.csv -rlen 30000 -cin hybrid3b.cod -cout hybrid3i.cod -alpha 0.05 -win
0.02
```

<u>Results</u>

A (870)	75.40%
D (206)	84.95%
B (349)	64.18%
C (239)	45.61%
Overall	69.95% (Standard)
	67.54% (Average)

GENERATED CODEBOOK FILE : hybrid3i.cod

•Experiment 18

<u>Description</u>
This code book vector set is based on the hybrid3b.cod set and trained with the lvq2 for 30000 iterations with a very low w=0.02 and alpha=0.04.

<u>Software Commands</u>
TRAINING FILE : 'norm.csv' , TEST FILE : 'normtest.csv'

```
>lvq2 -din norm.csv -rlen 30000 -cin hybrid3b.cod -cout hybrid3j.cod -alpha 0.04 -win
0.02
```

<u>Results</u>

A (870)	79.08%

D (206)	81.07%
B (349)	65.62%
C (239)	33.89%
Overall	70.01% (Standard)
	64.92% (Average)

GENERATED CODEBOOK FILE : hybrid3j.cod

•Experiment 19

Description
This code book vector set is based on the hybrid3b.cod set and trained with
the lvq2 for 30000 iterations with a low w=0.03 and w=0.02.

Software Commands
TRAINING FILE : 'norm.csv' , TEST FILE : 'normtest.csv'
>lvq2 -din norm.csv -rlen 30000 -cin hybrid3b.cod -cout hybrid3k.cod -alpha 0.03 -win 0.02

Results

A (870)	78.28%
D (206)	79.13%
B (349)	65.62%
C (239)	40.17%
Overall	70.25% (Standard)
	65.80% (Average)

GENERATED CODEBOOK FILE : hybrid3k.cod

•Experiment 20

Description
This code book vector set is based on the hybrid3b.cod set and trained with
the lvq2 for 300 iterations with a very low alpha=0.01 and w=0.02.

Software Commands
TRAINING FILE : 'norm.csv' , TEST FILE : 'normtest.csv'
>lvq2 -din norm.csv -rlen 30000 -cin hybrid3i.cod -cout hybrid3l.cod -alpha 0.01 -win 0.02

Results

A (870)	73.45%
D (206)	85.92%
B (349)	66.19%
C (239)	46.03%
Overall	69.53% (Standard)
	67.90% (Average)

GENERATED CODEBOOK FILE : hybrid31.cod

•Experiment 21

The training set was used as the derivation set for the 1-NN algorithm using software written for the purpose (written in CLIPPER - a database language). The software generated a confusion matrix of the classification categories by iterating through the test set. (There are therefore 2,768,896 (1664^2) euclidean distance measurements to be made). This algorithm is computationally expensive.

CONFUSION MATRIX

	A	B	C	D
A	582	141	114	45
B	142	195	10	3
C	100	11	107	7
D	46	2	8	151

Results	
A (870)	66.90%
D (206)	73.30%
B (349)	55.87%
C (239)	44.77%
Overall	60.21% (Average)
	62.20% (Standard)

REFERENCES

[1.] Ackley D H, Hinton G E and Sejnowski T (1985). A learning algorithm for Boltzmann machines, *Cognitive Science*, **9**, pp.147-169.

[2.] Almeida L B (1987). A learning rule for Asynchronous Perceptrons with Feedback in a Combinatorial Environment. *IEEE First International Conference on Neural Networks* (San Diego 1987), eds M.Caudill and C.Butler, **Vol 2**, pp.609-618. New York. IEEE.

[3.] Anderson J A and Rosenfeld E (1988). *Neurocomputing: Foundations of research*, MIT Press, 1988, Cambridge MA.

[4.] Anderson J A, Pellionisz A & Rosenfeld E (1990). *Neurocomputing 2: Directions for Research*, MIT Press, 1990, Cambridge, MA.

[5.] Aristotle (ca. 400 BC). "De memoria et reminiscentia", *Aristotle on Memory*, Richard Sorabji (translated), Providence, RI. Brown University Press, 1972. This excerpt is reprinted in Anderson et al (1990).

[6.] Baras J S and LaVigna A (1990). Convergence of the Vectors in Kohonen's learning Vector Quantization. *Proceedings IEEE International Neural Network Conference*, Paris, July 9-13, 1990, **Vol 2**, pp.1028-1031.

[7.] Barlow H B (1988). Neuroscience:A New Era ?, *Nature*, **331**, (February 1988):571

[8.] Begg D, Fischer S and Dornbush R (1984). *Economics:British Edition*. London. McGraw Hill Book Company (UK) Ltd.

[9.] Box G E P and Jenkins F M (1976). *Time Series Analysis: Forecasting and Control*. 2nd Edition. Oakland, CA. Holden Day.

[10.] Brieman L, Friedman J H, Olshen R A & Stone C J (1984). *Classification and Regression Trees*, Monterey, CA. Wadsworth and Brooks.

[11.] Bryson A E and Ho Y C (1969). Applied Optimal Control. New York: Blaisdell.

[12.] Cacoullos T (1966). Estimation of a multivariate density. *Annals of the Institute of Statistical Mathematics*, Tokyo, **Vol.18(2)**, pp.179-189.

[13.] Copeland L S (1989). *Exchange Rates and International Finance*. Wokingham, England. Addison Wesley Publishers Ltd.

[14.] Cornwell J (1994). Is mind merely matter ? : The Culture Essay. *The Sunday Times*, 15 May 1994, Section 10, pp.4-6.

[15.] Cover T M & Hart P E (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, **IT-13**, pp.21-27.

[16.] Comp.ai.neural-nets.faq. Frequently Asked Questions file, Posted 29th August 1994.

[17.] Crick F (1989). Neural Edelmanism. *Trends in Neurosciences*, **12**, (July 1989), pp.240-48.

[18.] Crick F (1994). *The Astonishing Hypothesis: the scientific search for the soul.* London. Simon & Schuster.

[19.] Cybenko G (1988). Continuous Valued Neural Networks with Two Hidden Layers Are Sufficient. Technical Report, Department of Computer Science, Tufts University, Medford. MA.

[20.] Cybenko G (1989). Aproximation by Superpositions of a Sigmoidal Function. *Mathematics of Control, Signals, and Systems*. **2**. pp.303-314

[21.] Dennett D (1991). *Conciousness Explained.* London. Allen Lane: the Penguin Press, Originally published in the USA by Little, Brown and Company.

[22.] Denton T A, Diamond G A, Helfant R H, Khan S and Karagueuzian H (1990). Fascinating rythm: A primer on chaos theory and its application to cardiology. *American Heart Journal*, **Vol.120**, pp.1419-1435.

[23.] *The Economist*, (1992a). August 15th 1992. Finance Section, 'The last of the good times ?'. pp.65.

[24.] *The Economist*, (1992b). August 15th 1992. Finance Section, 'Technical Analysis:Tilting at chaos'. pp.70.

[25.] *The Economist*, (1993). October 9th. Frontiers of Finance: A survey.

[26.] Edelman G (1994). *Bright Air, Brilliant Fire : On the matter of the mind.* London, England. Penguin Books.

[27.] Fahlmann S E and Lebiere C (1990). The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems 2* (Denver, 1989), ed D.S.Touretzky, Morgan Kaufmann, pp.788-793.

[28.] Feigenbaum M J (1983). Universal Behaviour In Nonlinear Systems. *Physica*, 7D. pp.16-39. North Holland Publishing Company.

[29.] Fisher R A (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7, pp.179-188.

[30.] Fix E & Hodges J (1951). *Discriminatory analysis, nonparametric discrimination: consistency properties. Report no. 4*, project no. 21-49-004, USAF School of Aviation Medicine. Texas. Randolph Field.

[31.] Frean M (1990). The upstart algorithm : a method for constructing and training feedforward neural networks. *Neural Computation 2*, pp.198-209. Cambridge, Massachusetts. MIT Press.

[32.] Friedman J H (1991). Multivariate Adaptive Regression Splines, *Annals of Statistics*, 19, pp.1-142.

[33.] Friesleben B (1992). Stock Market Prediction with Backpropagation Networks. *5th International Conference of Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pp.451-460. Paderborn, Germany, June 1992. Springer Verlag.

[34.] Geman S, Bienenstock E & Doursat R (1992). Neural Networks and the Bias/Variance Dilemma. *Neural Computation*, **4**, pp.1-58. Cambridge, Massachusetts. MIT Press.

[35.] Gleick J (1988). *Chaos*. London, England. Abacus (a division of Little, Brown and Company Ltd).

[36.] Gorman R P and Sejnowski T J (1988). Analysis of Hidden Units in a Layered Network to Classify Sonar Targets. *Neural Networks 1*, pp.75-89.

[37.] Gransden W R , Eykyn S J and Philips I (1990). The computerized documentation of septicaemia, *Journal of Antimicrobial Chemotherapy, Supplement C*, 25, pp.31-39.

[38.] Gregory R L (1987). *The Oxford Companion to The Mind.* Oxford, England. Oxford University Press.

[39.] Grossberg S (1976). Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors. *Biological Cybernetics*, **23**, pp.121-134. Reprinted in Anderson and Rosenfeld (1988).

[40.] Hand D J (1981). *Discrimination and Classification*. Chichester. Wiley.

[41.] Hand D J (1982). *Kernel Discriminant Analysis*. Letchworth, Herts, England. Research Studies Press.

[42.] Hart A & Wyatt J (1990). Evaluating black-boxes as medical decision aids:issues arising from a study of neural networks, *Medical Informatics*, **15**, pp.229-236.

[43.] Hebb D O (1949). *The Organization of Behaviour* (Extracts : pp.xi-xix & pp.60-78.) New York. Wiley. Reprinted in Anderson and Rosenfeld (1988).

[44.] Hertz J, Krough A & Palmer R G (1991). *Introduction to the theory of Neural Computation*. New York. Addision Wesley.

[45.] Hopfield J J (1982). Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences*, **79**, pp.2554-2558. Reprinted in Anderson & Rosenfeld (1988).

[46.] Hopfield J J (1984). Neurons with graded response have collective computational properties like those of two state neurons, *Proceedings of the National Academy of Sciences*, **81**, pp.3088-3092. Reprinted in Anderson & Rosenfeld (1988).

[47.] Hospers J (1988). *An Introduction to Philosophical Analysis*, 3rd Edition, London. Routledge.

[48.] Hunt K J, Sbarbaro D, Zbikowski R & Gawthrop P J (1992). Neural Networks for Control Systems - A Survey, *Automatica*, Vol.28, No.6, pp.1083-1112. International Federation of Automatic Control (IFAC). UK. Pergamon Press Ltd.

[49.] Hush D R & Horne W G (1993). Progress in Supervised Neural Networks: What's new since Lippmann, *IEEE Signal Processing Magazine*, Jan.

[50.] Hecht-Nielsen (1989). Theory of the Backpropagation Neural Network. *Proceedings 1989 IEEE International Joint Conference on Neural Networks*, **Vol.1** pp.593-605. Piscataway, NJ. IEEE Press.

[51.] INSPEC (1994). *Key Abstracts: Neural Networks*. Published for the IEE/IEEE.

[52.] Kimoto T, Asakawa K, Yoda M & Takeoka M (1990). Stock Market Prediction System with Modular Neural Networks, *IEEE Neural Networks Conference Proceedings 1990.* Piscataway, NJ. IEEE Press.

[53.] Kohonen T (1986). *Learning Vector Quantization for Pattern Recognition*, Helsinki University of Technology, Department of Technical Physics, Laboratory of Computer and Information Science, Report TKK-F-A601, 1986.

[54.] Kohonen T (1988a). An Introduction to Neural Computing, *Neural Networks*, **Vol.1**, pp.3-16.

[55.] Kohonen T (1988b). Learning Vector Quantization, *Neural Networks*, Supplement 1, pp.303.

[56.] Kohonen T, Barna G and Chrisley R (1988c). Statistical Pattern Recognition with Neural Networks: Benchmarking Studies, *Proceedings IEEE International Conference on Neural Networks*, San Diego, CA, USA, July 24-27, 1988, Vol **1**, pp.61-68. Reprinted in Anderson et al (1990).

[57.] Kohonen T (1990). Statistical Pattern Recognition Revisited, *Advanced Neural Computers*, R.Eckmiller (Editor), Elsevier Science Publishers B.V. (North Holland).

[58.] Kohonen T, Kangas J and Laaksonen J (1992). *SOM_PAK: The Self-Organising Map Program Package*, Version 1.2 (Novermber (sic) 2, 1992). Helsinki University of Technology, Laboratory of Computer and Information Science, Rakentajanaukio 2 C, SF-02150 Espoo Finland.

[59.] Kosko B (1991). *Neural Networks and Fuzzy Systems*. Prentice Hall.

[60.] Le Cun Y (1989). *Generalization and Network Design Strategies*. Technical Report CRG-TR-89-4, Department of Computer Science, University of Toronto. Toronto Canada.

[61.] Le Cun Y, Bozer D, Denker J S, Henderson D, Howard R E, Hubbard W & Jackel L D (1990). Backpropagation applied to handwritten zip code recognition. *Neural Computation 1*: 541-551. Cambridge, Massachusetts. MIT Press. Reprinted in Anderson et al (1990).

[62.] Lewis P A W and Stevens J G (1991). Nonlinear Modeling of Time Series Using Multivariate Adaptive Regression Splines (MARS)'. *Journal of the American Statistics Association*, **87**,pp. 864-877.

[63.] Lipscombe B (1989). Expert Systems and Computer-Controlled Decision Making in Medicine, *AI & Society*, **3**, pp.184-197, Springer Verlag.

[64.] McCulloch W S & Pitts W (1943). A logical calculus of the ideas immanent in nervous activity, *Bulletin of Mathematical Biophysics*, **5**, pp.155-133. Reprinted in Anderson & Rosenfeld (1988).

[65.] Michie D, Spiegelhalter D J & Taylor CC (1994). *Machine Learning, Neural and Statistical Classification.* Chichester, West Sussex, England. Ellis Horwood.

[66.] Minsky M & Papert S (1969). *Perceptrons*. Cambridge, MA. MIT Press. Parts reprinted in Anderson & Rosenfeld (1988).

[67.] Montana D J and Davis L (1989). Training Feedforward Networks Using Genetic Algorithms. *Eleventh International Joint Conference on Artificial Intelligence*, Detroit, 1989. Editor Sridharan N S. pp.762-767. San Mateo. Morgan Kaufmann.

[68.] Moravec H (1988). *The Mind Children: the future of robot and Human Intelligence.* Cambridge, MA. Harvard University Press.

[69.] Oja E (1989). Neural Networks. Principal Components, and Subspaces. *International Journal of Neural Systems*, **1**, pp.61-68.

[70.] Parker D B (1985). Learning Logic. Technical Report TR-47, Center for Computational Research in Economics and Management Science. Cambridge, MA. Massachusetts Institute of Technology.

[71.] Parzen E (1962). On Estimation of a probability density function and mode, *Annals of Mathematical Statistics*, **33**, pp.1065-1076.

[72.] Penrose R (1989). *The Emporer's New Mind: concerning computers, minds, and the laws of physics*. London. Oxford University Press. ISBN 0-09-977170-5.

[73.] Peterson C and Anderson J R (1987). A Mean Field Theory Learning Algorithm for Neural Networks. *Complex Systems*, **1**, pp. 995-1019.

[74.] Pineda F J (1987). Generalization of Back-Propagation to Recurrent Neural Networks. *Physical Review Letters*, **59**, pp.2229-2232.

[75.] Prechelt L (1996). A Quantitative Study of Experimental Evaluations of Neural Network Learning Algorithms: Current Research Practice. *Neural Networks*, 9(3), pp.457-462.

[76.] Rasmussen C E, Neal R M, Hinton G E, van Camp D, Revow M, Ghahramani Z, Kustra R & Tibshirani R (1996). *The DELVE Manual*, Version 1.1: http://www.cs.utoronto.ca/~delve/

[77.] Refenes A N (1991). Constructive Learning and its Application to Currency Exhange Rate Forecasting. Chapter 27 in Turban E and Trippi R (Editors), *Neural Network Applications in Investment and Finance Services*. USA. Probus Publishing.

[78.] Refenes A N, Azema-Barac M and Treleaven P C (1992). *Financial Modeling using Neural Networks*. Technical Report UCL-CS, RN-92-94, Department of Computer Science, University College London.

[79.] Ripley B D (1993). Statistical aspects of neural networks, Published in *Networks and Chaos - Statistical and Probabilistic Aspects*, pp.40-123, Edited by Barndorff-Nielsen O E, Jensen J L & Kendall W S. London. Chapman and Hall.

[80.] Ripley B D (1996). Statistical ideas for selecting network architectures. Published in *Neural Networks: Artificial Intelligence and Industrial Applications*, pp.183-190, Edited by Kappen B & Gielen. Holland. Springer Verlag.

[81.] Refenes A N (1991). Constructive Learning and its Application to Currency Exchange Rate Forecasting. Appearing in Turban E and Trippi R (Editors): *Neural Network Applications in Investment and Finance Services*, Chapter 27. USA:Probus Publishing.

[82.] Rochester N, Holland J H, Haibt L H, Duda W L (1956). Tests on a cell assembly theory of the action of the brain, using a large digital computer. *IRE Transactions on Information Theory*, IT-2, pp.80-93.

[83.] Rosenblatt M, (1956). Remarks on some non-parametric estimates of a denisty function, *Annals of Mathematical Statistics*, Vol 27, pp.832-37.

[84.] Rosenblatt F (1958). The perceptron: a probabalistic model for information storage and organization in the brain, *Psychological Review*, **65**, pp.386-408. Reprinted in Anderson & Rosenfeld (1988).

[85.] Rumelhart D E, McClelland J L and the PDP Research Group (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1:Foundations*. Cambridge, MA: MIT Press.

[86.] Schaffer C (1994). A Conservation Law for Generalization Performance. *Machine Learning International Conference Proceedings; 11th Edition.* Cohen W W & Hirsh H. Sanfrancisco, CA. Morgan Kaufman.

[87.] Shanson D C (1989). *Microbiology in Clinical Practice*. Chapter 5, pp.138-150. London. Wright. 2nd Edition.

[88.] Shortliffe E H (1976). *Computer Based medical consultations: MYCIN*. New York. Elsevier.

[89.] Shu S D, Bliven S E & Belina J C (1991). Training Of Feedforward Neural Network Architectures For Feature Recognition of Abnormal ECG Waveforms. *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, **Vol.13**, pp.1395-1396. IEEE, New York.

[90.] Sietsma J and Dow R J F (1988). Neural Net Pruning - Why and How. *IEEE International Conference on Neural Networks (San Diego)*, **Vol.1**, pp.325-333. IEEE, New York.

[91.] Sleigh J D & Timbury M C (1990). *Medical bacteriology*, Churchill Livingstone London. (Medical Division of Longman Group UK Ltd). Third Edition.

[92.] Specht D F (1988). Probabalistic Neural Networks for Classification, Mapping or Associative Memory, *Proceedings International Conference Neural Networks 1988 (ICNN-88)*.

[93.] Specht D F (1990). *Probabalistic Neural Networks*, Neural Networks, pp.109-118, **Vol.3**. Pergamon Press.

[94.] Tang Z, Almeida C & Fishwick P (1991). Time series forecasting using neural networks vs. Box-Jenkins methodology. *Simulation*, November.

[95.] Taylor M P & Allen H (1992). The use of technical analysis in the foreign exchange market. *Journal of International Money and Finance*, 1992, **11**, pp.304-314. Butterworth-Heinemann Ltd.

[96.] Thrun S B, Bala J, Bloedorn E, Bratko I, Cestnik B, Cheng J, De Jong K, Dzeroski S, Fahlman S E, Fisher D, Hamann R, Kaufman K, Keller S, Kononenko I, Kreuziger J, Michalski R S, Mitchell T, Pachowicz P, Reich Y, Vafaie H, Van de Welde W, Wenzel W, Wnek J and Zhang J (1991). *The MONK'S Problems : A Performance Comparison of Different Learning Algorithms*, December, Carnegie Mellon University, Technical Report CMU-CS-91-197.

[97.] Tong H and Lim K S (1980). Threshold Autoregression, Limit Cycles and Cyclical Data. *Journal of the Royal Statistical Society*. B 42, pp.245-292.

[98.] Widrow B and Hoff M E (1960). Adaptive switching circuits, *1960 IRE WESCON Convention Record*, New York: IRE, pp.96-104.

[99.] Wilken P (1994). Psyche-D Usenet Discussion Forum. <PSYCHE-D%NKI.BITNET@uga.cc.uga.edu>.

[100.] Werbos P (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioural Sciences*. Ph.D. Thesis, Harvard University.

[101.] Weigend A S, Huberman B A and Rumelhart D E (1992). Predicting Sunspots and Exchange Rates with Connectionist Networks. Appearing in Casdagli M and Eubank S (Editors) *Nonlinear modeling and forecasting,* SFI Studies in the sciences of complexity. Reading, MA:Addison Wesley Publishing.

[102.] Weigend A S and Gershenfeld N A (1993). *Time Series Prediction: Forecasting the Future and Understanding the Past*. Reading, MA:Addison Wesley.

[103.] White H (1988). Economic Prediction using neural networks: the case of IBM daily stock returns, *Proceedings of the IEEE International Conference on Neural Networks, San Diego, 1988*, **Vol.2**, pp.451-459. Reprinted in Anderson et al. (1990).

[104.] Winston P H (1992). *Artificial Intelligence.* Third Edition. Reading, MA. Addision Wesley.

[105.] Worthy P J, Dybowski R, Gransden W R & Summers R (1993). 'Comparison of Learning Vector Quantization and k-Nearest Neighbour For Prediction of Microorganisms Associated With Septicaemia', pp.273-274, *Proceedings of the 15th*

Annual international Conference of the IEEE Engineering in Medicine and Biology Society, **Vol.15**, San Diego California, October 28-31.