Neural Networks For Financial Forecasting

Siew Lan Loo

A thesis submitted for the degree of

Doctor of Philosophy in Computer Science

University of London

May 1994

Department of Computer Science
University College London
Gower Street
LONDON WC1E 6BT

Abstract

Neural networks demonstrate great potential for discovering non-linear relationships in time-series and extrapolating from them. Results of forecasting using financial data are particularly good [LapFar87, Schöne90, ChaMeh92]. In contrast, traditional statistical methods are restrictive as they try to express these non-linear relationships as linear models.

This thesis investigates the use of the Backpropagation neural model for time-series forecasting. In general, neural forecasting research [Hinton87] can be approached in three ways: research into the weight space, into the physical representation of inputs, and into the learning algorithms. A new method to enhance input representations to a neural network, referred to as model sN*x*, has been developed. It has been studied alongside a traditional method in model N. The two methods reduce the unprocessed network inputs to a value between 0 and 1. Unlike the method in model N, the variants of model sN*x*, sN1 and sN2, accentuate the contracted input value by different magnitudes. This different approach to data reduction exploits the characteristics of neural extrapolation to achieve better forecasts. The feasibility of the principle of model sN*x* has been shown in forecasting the direction of the FTSE-100 Index.

The experimental strategy involved optimisation procedures using one data set and the application of the optimal network from each model to make forecasts on different data sets with similar and dissimilar patterns to the first.

A Neural Forecasting System (NFS) has been developed as a vehicle for the research. The NFS offers historical and live simulations, and supports: a *data alignment facility* for standardising data files with non-uniform sampling times and volumes, and merging them into a spreadsheet; a *parameter specification table* for specifications of neural and system control parameter values; a *pattern specification language* for specification of input pattern formation using one or more time-series, and loading to a configured network; a *snapshot facility* for re-construction of a partially trained network to continue or extend a training session, or re-construction of a trained network to forecast for live tests; and a *log facility* for recording experimental results.

Using the NFS, specific pattern features selected from major market trends have been investigated [Pring80]: *triple-top* ("three peaks"), *double-top* ("two peaks"), *narrow band* ("modulating"), *bull* ("rising") and *recovery* ("U-turn"). Initially, the *triple-top* pattern was used in the N model to select between the *logarithmic* or *linear* data form for presenting raw input data. The selected *linear* method was then used in models sN1, sN2 and N for network optimisations. Experiments undertaken used networks of permutations of sizes of input nodes (*I*), hidden nodes (*H*), and tolerance value. Selections were made for: the best method, by value, direction, or value and direction, for measuring prediction accuracy; the

best configuration function, $H = I\phi$, with ϕ equal to 0.9, 2 or 3; and the better of sN1 and sN2. The evaluation parameters were, among others, the prediction accuracy (%), the weighted return (%), the Relative Threshold Prediction Index (**RTPI**) indicator, the forecast error margins. The **RTPI** was developed to filter out networks forecasting above a minimum prediction accuracy with a credit in the weighted return (%). Two optimal networks, one representing model sNx and one N were selected and then tested on the *double-top*, *narrow band*, *bull* and *recovery* patterns.

This thesis made the following research contributions.

- A new method in model sNx capable of more consistent and accurate predictions.
- The new **RTPI** neural forecasting indicator.
- A method to forecast during the *consolidation* ("non-diversifying") trend which most traditional methods are not good at.
- A set of improvements for more effective neural forecasting systems.

Acknowledgements

This thesis is jointly sponsored by the Science And Engineering Research Council and Barclays de Zoete Wedd, London. To this end, I thank both sponsoring bodies for their financial support, and Barclays de Zoete Wedd for providing the research data.

The thesis would not have materialised, but for the efforts of Prof. P. Treleaven, Dr. P. Rounce and Dr. D. Gorse.

I also benefited from John Taylor, without which it would not have been possible to carry out all the experiments discussed.

The supports of Dr. D. Saunders of Queen Mary and Westfield College and Dr. E. Tsang of the University of Essex have been pivotal as well.

I owe a debt of gratitude to Graham Simister formerly of Nomura Bank International. This thesis has benefited from the support and knowledge gained whilst I was working under him.

Contents

List Of Figures

List Of Tables

Chapter 1

Introduction

This chapter presents the motivations, aim, and contributions of this thesis. Initially, it states the properties of neural networks that motivate this research. There is then a survey of financial neural forecasting, emphasising research systems applied to "real-world" data. Next, it presents the aim of the thesis, the objectives of the experiments undertaken, and the choice of specific pattern features chosen for the experimental data sets. Finally, it gives an overview of the thesis contribution and organisation.

1.1 Motivations

Neural networks have great potential for discovering the underlying structure of non-linear time-series and extrapolating these time-series to the future [LapFar88, ShaPat90, TanAlm90, MarHil91, SriLoo91]. In contrast, traditional statistical methods [HanRei89] are restrictive as they try to express these non-linear relationships as linear models. Dutta and Shekar [DutShe88] confirmed this in their comparison studies, neural networks consistently out-perform multi-regression models for predicting bond ratings: the former averaging at 80% accuracy against 60% by the latter. In addition, Chakraborty et al [ChaMeh92] showed their neural multi-index model approximates flour prices significantly better, providing a better fit for the test data than Tiao and Tsay's auto-regressive moving average model. Latterly, these optimistic views have been reinforced by Schönenberg [Schöne90] who obtained 90% accuracy for forecasting stocks.

Neural networks are known especially for their adaptive features and massive parallelism. With the impetus of early work like that of Carpenter and Grossberg [CarGro88], Rumelhart, McCelland and Hinton [RumMcC86, RumHin86], Sejnowski [SejRos86], and Kohonen [Kohone88], the research momentum escalated with reports of results from numerous and varied application domains for signal processing, natural language processing and pattern recognition. The potential benefits of this programming paradigm have also attracted a lot of interest from the financial sector. Some of its many applications are for fraud detection, mortgage underwriting [ReiCol90], extracting information from accounting reports [TriBer90], bond rating [ShaPat90, SurSin90], bankruptcy prediction [OdoSha90, RagSch91], and forecasting [KimAsa90, RefAze92, ChaMeh92, TanKam92, JanLai93].

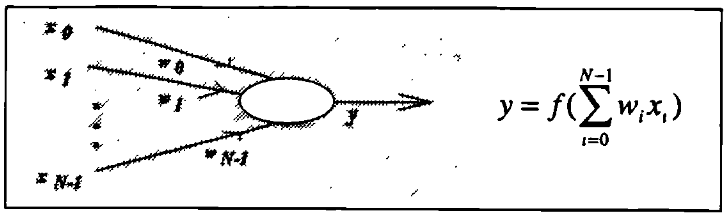


Figure 1.1 Computation Of A Simple Node

Artificial neural networks are mathematical models of simulated neurons based on our present understanding of the biological nervous system. The characteristics of the well-studied models, for example the Backpropagation model, ART, the Perceptron, Self-Organising Maps are well documented [Lippma87, Wasser89, Dayhof90, HerKro91]. Typically, a neural network like that of a Backpropagation model is composed of a number of processing elements (nodes) that are densely interconnected by links with variable weights. Unlike conventional sequential processing, all the adjacent nodes process their outputs in parallel. Each node delivers an output, y according to an *activation* rule. In its simplest form, the rule for a non-linear output of a node is a sum of its N weighted inputs as shown in Figure 1.1. The transition function, f, normally has a binary, linear, sigmoid or hyperbolic tangent characteristic. As a result, a neural model is made unique by the specifications of the topology and dimension of the network, the characteristics of the nodes including the type of transition function used, and the learning algorithm.

1.1.1 Forecasting Research Approaches

In his report on neural learning procedures, Hinton [Hinton87] classified neural research into three main categories: search, representation and learning. The investigative procedure carried out in the first category searches the weight space for an optimal solution to the network mapping in a constraint-satisfying manner. In the second, the research focuses on identifying the physical representation of the inputs that best represents the salient features of the domain. Finally, in the learning category, the research estimates a learning algorithm modelling the relationships of all the network elements to present a mapping solution in some numeric form.

On a similar level, forecasting techniques have been classified into two categories: qualitative methods and quantitative methods [HanRei89]. Qualitative methods cover a spectrum of non-science techniques, most of which are well-documented, tried and tested explanations and judgements. In contrast, quantitative research covers well-defined models for either time-series or econometric forecasting. Time-series forecasting predicts the future value by discovering the pattern of the historical series and extrapolating the value to the future. Econometric forecasting is, on the contrary, designed to predict a dependent variable by discovering the form of a cause-effect relationship using one (univariate) or more (multivariate) independent variables and to use the learnt interdependencies for prediction.

From the point of view of neural computing, neural forecasting research could, therefore, be viewed as a concentration of efforts in any one of the three neural categories applied to either one of the forecasting methods using a specific set of financial data for analysis. Of the three choices, the most popular approach has initially been the exploitation of the unique learning ability of a neural model on a specific forecasting application. To date, the Backpropagation model has always been the model of choice, mainly because it has been widely researched and the supervised learning strategy is well-suited for the task.

For instance, Schönenberg [Schöne90] carried out time-series forecasting research on German stocks by experimenting with four different neural models, Adaline, Madaline, the Perceptron and Backpropagation. In his approach, he had taken to optimising the models' networks with regard to learning the features of each of the different types of stocks. He was able to isolate and differentiate the behaviour of the models by fine-tuning neural parameters such as the size and the types of input of the input vectors, the ways of splitting the input vectors and the network configuration.

Varfis and Versino [VarVer91] like Schönenberg used a Backpropagation model for univariate forecasting. One of the areas on which they concentrated was the input data and the structure of the input vectors. Their main concern was the simulation of the underlying features of seasonal behaviour and time lags in an otherwise "flat" network.

Unlike Schönenberg, or Varfis and Versino, who used different time-series information derived from one specific type of price index for inputs to the neural model, there are others like Kimoto and Asakawa [KimAsa90], and Windsor and Harker [WindHar90] who have used a neural model for multivariate forecasting. Kimoto and Asakawa, for example, developed a prediction system for the Japanese stock index, TOPIX, using clusters of Backpropagation networks. Each cluster in the network is responsible for a batch of indices and macro-economic data, and all the outputs are combined to produce a weighted average for the weekly returns of TOPIX. In addition to optimising the network, they also modified the learning algorithm to improve the speed of training. The supplementary learning process is one such enhancement. It is a control applied to each training cycle to ensure a training pattern would not be unnecessarily presented for further training once the pattern has been sufficiently learnt. The learning criteria is based on the dynamic minimum network error achieved in the mapping.

Windsor and Harker, on the other hand, adapted a Backpropagation model to simultaneously predict the annual movements of a range of London Stock Exchange Indices. In particular, a steepness threshold was imposed onto the sigmoid transition function to control the interdependencies and correlations of the indices. This was to force the network to predict other chosen indices in addition to the index to be predicted while interpreting the movements of the indices as a whole. Apart from that, they also investigated ways to bring out the structure of the training data. To do so, they focused on methods for representing the data. They transformed the actual input values to fit logarithms onto a regression line using an equation which they derived. As a result, their system predicted the deviations of the Index from the exponential growth instead of its absolute value.

The papers by Refenes [Refene91], and Jang and Lai [JanLai93] are examples of approaches by learning. Refenes' CLS+ model forecasts the Deutschemarks by dynamically updating the dimension of the hidden nodes using the constructive learning procedure. The model is supported by a *linear classifier* which can be trained to optimise the errors between its output and the target. As it is, a training session always commences with a one hidden node network. This is followed by a look ahead test procedure to establish whether there would be a reduction in the errors by an additional *linear classifier*. An affirmative result would have the dimension of the hidden layer increased and the weights of the *linear classifier* frozen. The test procedure is repeated for further additional classifiers until an optimal configuration is obtained.

Jang and Lai, who found the fixed-structure Backpropagation model too rigid, developed the DAS net to synthesise and adapt its topology for Taiwanese stocks. The net is a hierarchy of networks representing both the short- and long-term knowledge of a selection of technical indices. The short-term knowledge is used for short-term prediction, principally for trading decision making. In addition, it supplements the knowledge for the long-term view as the time window advances. Together, these two levels of knowledge control the continuous self-adjustment of the network's assessment of curve-fitting.

Yet another variant of the learning approach is to explore the characteristics of the nodes as [Casdag89] and [JonLee90] have done. They, like Jang and Lai [JanLai93], redressed the fixed-structure problem with radial basis functions (RBF). The research on this alternative tool is concerned with the representation of the hidden layer and its evaluation. The composition of an RBF network is a number of RBF nodes evaluating a Gaussian-like kernel function, and because it is designed to use a minimum hidden layer dimension, it is claimed to offer a better network generalisation. Another of its advantages is that the layers of the network can be trained independently on a layer by layer basis. This is a sharp contrast to the usual method of computing each node and expanding it across the layers of the network, or incrementally like CLS+.

The Backpropagation model also has weaknesses. The network is often described as having a fixed structure, with its inability to accommodate changes in time window sizes, and being "flat", lacking a third dimension. In addition, training is often slow, with a likelihood of getting caught in a local minima. As a result, derivatives of the model, like those highlighted, have been researched to overcome some of these problems. A recurrent theme that is often encountered in financial applications is the simulation of the time dimension of a

2-dimensional "flat" network. This is an inconvenience which Varfis and Vesino tried to simulate by aligning data from more than one recording time alongside each other as inputs to a 2-dimensional input vector. Others like Kamijo and Tanagawa [KamTan90], and Wong [Wong91] tried to redress the issue explicitly by adapting the framework of the network itself.

In the case of Tanigawa and Kamijo [KamTan90, TanKam92], the authors used a recurrent neural network for the recognition of candle chart patterns (discussed in Chapter 2). Effectively, a recurrent network [WilZip89] is a generalised Backpropagation model with a new learning algorithm capable of retaining temporal information. The recurrent learning procedure uses the information that has been accumulated in the memory which can either have a fixed or indefinite historical span. In [TanKam92], the proposed extended Backpropagation network had two hidden layers. Each of these was divided into two sets of units, for holding the stock prices and for the corresponding temporal information about its previous activity. Its successes in recognising a *triangle* chart pattern resulted in a Dynamic Programming matcher being incorporated into a new version of the system. The matcher is designed to resolve non-linear time elasticity (a feature of charting where the pattern could be formed but could have variable window sizes).

Wong [Wong91] introduced time by the addition of a third dimension to a neural model. Unlike [TanKam92]'s 2-dimensional temporal network which simulates time by splitting the hidden layers into two halves, the NeuroForecaster simulates time orientation by concatenating duplicates of an entire 2-dimensional network to form a single network. This extensive network is burdened by long network training. Subsequently, the FastProp learning algorithm [Wong91] was introduced to improve the rate of convergence. It trains sections of the network and clusters them according to the ranking of the Accumulated Input Error index. The index is built with a strategy to rank the data from an entire time-series to several clusters according to the dynamic causal relationship between the input data and the output.

Despite the various efforts made to enhance the standard Backpropagation model for forecasting with non-linear inputs, Deboeck [Deboec92], being a financial player, called for attention to an approach that is feasible but yet often overlooked. He asserted that in order for financial systems to benefit from neural processing, it is more important to concentrate on the basics of the application domain rather than the network paradigms. Areas suggested are those relating to the dynamics of the domain such as the pre-processing of input data, risk management and trading styles.

Following a survey of research approaches for financial forecasting, the approach taken for this thesis follows Deboeck's point of view. Specifically, it concentrates on Hinton's representation category, focusing on the method of transformation of input data to a Backpropagation network.

1.2 Thesis Aim

The AIM of this thesis is to investigate the use of the Backpropagation neural model for time-series forecasting. A new method to enhance input representations to a neural network, referred to as model sN*x*, has been developed. It has been studied alongside a traditional method in model N. The two methods reduce the unprocessed network inputs to a value between 0 and 1. Unlike the method in model N, the variants of model sN*x*, sN1 and sN2, accentuate the contracted input value by different magnitudes. This approach to data reduction exploits the characteristics of neural extrapolation to achieve better prediction accuracy. The feasibility of the principle of model sN*x* has been shown in forecasting the direction of the FTSE-100 Index.

The experimental strategy involves optimisation procedures using one data set and the application of the optimal network from each model to make forecasts on different data sets with similar and dissimilar pattern features to the first.

Experimental Objectives

The objectives of the experiments for optimisation were:

(1) `To select a suitable raw data form.`

Either the *ogarithmic* or *linear* (absolute value) data form is selected as more suitable for inputs to an **NFS** (explained in Section 1.3) configured network. This also serves to justify the research approach for this thesis. The better data form was applied to the rest of the experiments.

(2) `To select a suitable prediction accuracy measurement.`

A prediction is accurate if the direction of a prediction correlates with the tracking data. A data correlation can be defined by *value*, *direction*, or *value* and *direction.* A forecast value for tomorrow is correct according to interpretation by: *value,* if the movement of the forecast value compared with today's tracking value is the same as the movement of the tracking data, from today to tomorrow; *direction*, if the direction of the forecast movement from today to tomorrow correlates with the movement of the tracking data at the same period; *value and direction*, if the interpretations for each of the separate elements are combined. The best interpretations are subsequently used for evaluations of other objectives.

(3) `To select a suitable configuration function.`

A configuration function relates the dimension of the hidden layer, H, to the dimension of the input layer, I, by $H = I\,\phi$. Three values of ϕ were used: 0.9, 2 and 3.

Each of the three function values were combined with a set of five different *I* values to configure a group of networks as contenders for an optimal network to represent models N and sN*x*. The test cases were cross-validated with the tolerance value.

(4) To select the better version of model sN*x*.

The better version, either sN1 or sN2, forecasting with a more consistent manner across the group of test cases, is selected.

An optimal network selected for each model, sN*x* and N, is like the selection in (4), characterised by consistency, a gently rolling prediction accuracy landscape across the section of the evaluation window without intermittent spurts of exceedingly good accuracy.

The objective of the experiments applying the optimised networks was:

(5) To observe the optimised network accommodating to new data sets.

The *double-top* pattern (explained in the section on Experimental Data below) was applied, in part, to test the networks' ability to adapt to a new data set similar to one on which it was optimised. Like the other experimental patterns, it was used to test the optimised network's performance on specific pattern features.

The evaluations for objectives (3), (4) and (5) were based on a combination of performance statistics, among them being, the test correlation (%), the corresponding weighted return (%), the **RTPI** indicator (explained in Section 1.3) and the forecast error margins. The test correlation percentage is the prediction accuracy percentage measured for the test data set. The weighted return percentage is the final total of the number of Index points made or lost, as points amounting to the size of the movement are added whenever a forecast correlates and subtracted when it does not. It is an indirect indication of a network's ability to forecast the salient movements of the test pattern. The **RTPI** indicator is developed to combine these two measurements to differentiate the forecasting performances of networks. A minimum accuracy of 50% was assigned to the interpretation of a "weak" performance and a "weak" network is differentiated by a negative **RTPI** value as opposed to the positive valued "good" network.

The effectiveness of a neural forecasting tool is discovered by appraising the neural forecasting results in a practical situation, highlighting its usability and areas for improvement.

Experimental Data

The experimental data sets were selected from historical periods showing three data trends: *consolidation* ("non-diversifying" pattern), *bull* ("rising" pattern) and *recovery* ("U-turn" pattern). The *consolidation* trend was chosen for the optimisation procedures because its data values fluctuate consistently within a relatively smaller range of values, therefore being particularly suitable for training and testing. As a result, three different *consolidation* patterns were used, *triple-top*, *double-top* and *narrow band*, in addition to the *bull* and *recovery* patterns. Each of these patterns have distinctive observational features:

(a) `triple-top.`
Its signature is three almost identical peaks. The data values across the time window lie neatly within fixed upper and lower bounds. The test pattern comprised a peak formation. It was used for the optimisation procedures.

(b) `double-top.`
Its signature is two rather than three peaks. Some of the data values in the second peak are slightly outside the upper bound of the first. This section is part of the test pattern.

(c) `narrow band.`
Its signature is data modulating within a tight range. An additional and interesting feature is a *breakout*, a sudden diversification with a rising pattern at the end. The major objective is forecasting the breakout section whose data values are outside the range on which the network has been trained.

(d) `bull.`
Its signature is a rapidly rising pattern followed by a loss in momentum with the trend "moving sideways". The emphasis is on the behaviour of a network trained on a rising pattern and tested on a *consolidation* pattern trend, with some of the data values outside the training range as well.

(e) `recovery.`
Its signature is a rising pattern followed immediately by a sharp plunge and a gradual recovery from the dip. The recovery test pattern is similar to the rising pattern of the training pattern, but they do not lie within the same range of values.

1.3 Thesis Contributions

The contributions are presented on three levels of merit. The first represents primary achievements for common problems of time-series forecasting. The second represents achievements secondary to those above, as they are specific to a problem, data or experimental pattern. They are contributions which could be useful for future work on the specific problem using the same data or pattern. The third presents the features of the customised neural forecasting system, a tangible achievement suitable as a vehicle for undertaking further research, or for use in a practical environment.

The thesis contributions may be listed as follows:

(a) `A good input representation for data reduction.`

It has been proven that the new method of enhancing input, reducing it to values between 0 and 1, is better than the commonly used formula. The method in model sN*x* triumphs over model N in the following ways:

- ♦ a more stable prediction accuracy, offering more consistent percentages across a section of training cycles.
- ♦ a better corresponding weighted return percentage, indicating the network's ability to forecast better defined data movements.
- ♦ smaller forecast error margins, overall.
- ♦ better adaptation to time and data changes, maintaining equal or higher prediction accuracy, with usually a slightly better return percentage and generally smaller forecast error margins.

(b) `A neural forecasting indicator, the RTPI.`

The Relative Threshold Prediction Indicator (**RTPI**) differentiates the forecasting capability of a network. A network's forecasting capability is measured by its percentages of prediction accuracy and weighted return. The latter is the final total number of Index points made on the basis that if a forecasts correlates correctly, the size of the movement is added to the subtotal but it is deducted if it is incorrect. The **RTPI** indicator value differentiates a network forecasting below a specified prediction accuracy percentage by a negative indicator value and a positive value if it is above. Proofs of the principle in simulations showed the indicator is particularly useful in neural forecasting research as test cases are quite often large.

(c) `A solution to forecasting during the` *`consolidation`* `trend.`

The *consolidation* trend occurs frequently and more persistently than other data trends. It is a situation where traditional methods, such as moving averages, the RSI indicator, or Elliot Wave Theory are not good as forecasts are "flat" and do not provide any lead. The principle of neural processing offers flexibility in designing neural outputs. Simulation studies using three different pattern formations in a *consolidation* trend show it is possible to predict the direction of data movement, a small piece of information that could be exploited during a low risk period.

(d) `A set of improvements for a more effective neural forecasting tool.`

References to claims of exceptional neural forecasting results have totally disregarded their effectiveness. The statistics reported have been measured under pseudo-conditions: the dynamic events surrounding time-series are measured under stationary rather than time-varying conditions. An effective neural forecasting system should be able to do the following:

♦ Handle out-of-range problems.

An effective neural forecasting tool for non-linear data should not only forecast within a specific range of data values. It must be able to handle forecasting data values that are beyond the boundaries of the training data set.

♦ Handle forecasting reliability.

The confidence on a neural forecast is unknown and it is not easily measured. This problem can be resolved by two approaches: to research a confidence indicator and to research a more reliable and stable neural forecasting model across non-linear data.

♦ Handle intra-time-series information.

A neural forecast on the direction of a data move is not informative. It requires information about the forecast and the other data in the time-series. A neural forecast could be augmented by endogenous information. Information, such as projections of the following day's high and low data values, provide cues for implementing a better investment strategy.

♦ Handle structured design for neural forecasting models.

Neural forecasting has been studied on an ad hoc basis using snapshots of a static time window. An effective neural forecasting model should handle the "generators" in the data that are responsible for the non-linear data movements. The implementation of the dynamics of the non-linear data forms a principled basis for designing an effective neural forecasting network.

In addition, the thesis has established the suitability of forecasting the FTSE-100 Index, on the Neural Forecasting System (the **NFS** is explained below), with regard to:

(e) The raw data form.

It confirmed a *linear* rather than a *logarithmic* data form is more suitable for forecasting, at least on the Backpropagation model of the **NFS**.

(f) The measurement of prediction accuracy.

It confirmed the best method to measure that a forecast has correctly correlated with the tracking Index is by the measurement method of *value* as opposed to *direction* or *value and direction*. This is valid for input patterns formed and loaded into a network as reduced values of absolute Index values.

(g) The dimension of the hidden layer, *H*.

It confirmed that *H* for a noisy data pattern, the *triple-top*, should be based on the configuration function with ϕ equal to 2.

Finally,

(h) The Neural Forecasting System.

The **NFS** is specially developed to facilitate neural forecasting research and operational use. The system facilities are:

♦ Raw data integrity.

The problems of those raw data files which might have data with various sampling intervals and differing data volumes are solved by standardising to a daily sampling interval and merging by date into a spreadsheet for subsequent use.

♦ A neural parameters specification table.

The table allows easy access to all the modifiable neural and system control parameters as a system file. It contains: names of log files, learning parameters, training control parameters and the specification language.

♦ A forecasting model specification language.

The language allows the specification for time-series, single-index or multi-index forecasting. It has parameters to specify: a weighting for the input value according to the arithmetic operator specified; a time lag or advance in relation to the target node input; the omission of specific time slices of data forming successive input patterns.

♦ Reconstruction of network.

A snapshot facility records an image of the training network at regular intervals. A recorded image allows the re-construction of a partially trained network for continuation or extension of a training session, or the re-construction of a trained network for forward simulation.

♦ Options for forecasting simulations.

It offers options for historical and forward simulation which are accessible directly via menu selections. The historical simulation option undertakes to train and test a prototype neural model specified in the parameter specification table. The forward simulation option delivers a forecast value for one day ahead using a re-constructed trained neural network. The forecast is for testing on live data.

♦ User support.

A user manual is also delivered with the **NFS** for subsequent user support.

1.4 Thesis Organisation

Following the Japanese New Information Processing Technology research programme [MITI90], there is a proposal for the next generation of intelligent systems using "soft" logic and "hard" logic. Both models of computation embrace a spectrum of state-of-the-art intelligent techniques. There are, on the "soft" side, techniques like expert systems, neural networks, genetic algorithms, fuzzy logic [TreLoo90, TreGoo92]. The complementary tools of "hard" logic such as parallel computers and silicon chips are usually used to support the computations of the "soft" techniques.

Chapter 2 exemplifies the significance of "soft" logic for financial systems. Initially, it surveys the classical forecasting methods to highlight their shortcomings. Following that, there is a description of the practical use of the "soft" neural network for sub-symbolic processing. Financial adjectives like *increasing* and *bull* which are not easily measured are qualified from time-series data. They are then asserted as facts in the knowledge-base for evaluation in a financial rule-based system.

The description of the research commences with Chapter 3, which reports the development of the customised Neural Forecasting System. Initially, it describes the system specifications, with outlines of general requirements, such as the operating platform, the user interface and the system characteristics. Following this are the technical requirements with details of the Backpropagation model and the research facilities needed to facilitate neural forecasting. Lastly, it explains the system design, with outlines of the system processes mapping these specifications.

Chapter 4 describes the translation of the system processes designed to five 'C' - ***Menu***, ***Align***, ***I/O***, ***Load*** and ***Backpropagation*** - Modules. First, it gives an overview of the system and explains the functions of these Modules. Following this, it explains the techniques used to implement system facilities for: the alignment of non-uniform raw data; the specification and tuning of parameters; the specification of input pattern formation and loading; and the re-construction of a neural network for further processing.

Chapter 5 reports the strategy used to validate the data reduction method in model sNx. Initially, it explains the notations used, including the **RTPI** indicator, the method of extrapolation, and the choice of data sets. Thereafter, it describes the experiments - to select a suitable raw data form, optimisation of network configuration and neural parameters to identify a suitable prediction accuracy measurement, and testing the optimised networks on time complex data sets - and evaluation criterion.

Chapter 6 evaluates time-series forecasting by models sNx and N. It covers the optimisation of networks for each model and compares their application to new data sets having similar or dissimilar train and test patterns. The optimisation procedures select a suitable raw input data form, prediction accuracy measurement and configuration function, and the better of models sN1 and sN2. The evaluation discusses, amongst other things, the test correlation (%), its weighted return (%) and the **RTPI** indicator.

Chapter 7 assesses the feasibility of neural forecasting, reviewing their effectiveness to deliver consistently good results and as a usable tool. First, it examines the **RTPI** indicator to justify its achievements and relevance. Thereafter, it examines the measurement of network generalisation and the relationship between network learning and extrapolation. Next, it examines the usability of neural forecasts to form investment strategies, and, finally, it compares model sNx with related work and other forecasting tools.

In conclusion, Chapter 8 summarises the research, its contributions and ways to improve the effectiveness of neural forecasting. It begins with a synopsis of the features of the customised neural forecasting system, the choice of experimental data, and the objectives of all the experiments undertaken. This is followed by an appraisal of the research with a presentation of the achievements resolving specific time-series forecasting problems. It ends with suggestions for technical and applied research to suit a range of time-scales and interests.

Chapter 2

Soft Logic And Finance

This chapter exemplifies the significance of "soft" logic for financial systems. Initially, it surveys the classical forecasting methods to highlight their shortcomings. Following that, there is a description of the practical use of the "soft" neural network for sub-symbolic processing. Financial adjectives like "increasing" and "bull" which are not easily measured are qualified from time-series data. They are then asserted as facts in the knowledge-base for evaluation in a financial rule-based system.

2.1 Financial Forecasting

Due to the Darwinian nature of operations in the financial markets, every investment strategy is made with a sound money management policy. However, the volatile atmosphere dictates that the decision of a strategy is made preceding the decision of the policy, following a thorough evaluation of the market. A speculator will aspire to make a profit by taking advantage of the results of the evaluation, which is, speculating the movements of the market data. Apart from speculating to be successful, an even greater temptation for the speculator is the prospect of capital gains by a "low risks and high rewards" approach. Consequently, they are increasingly turning to the latest technology for competitive solutions to their optimisation and forecasting problems.

Principally, the proponents of forecasting believe that, in spite of the temperament of the financial market, its direction can be anticipated, and that the anticipation can be effectively exploited for capital gains. Research into forecasting follows the axiom that the future is the result of an extension of the salient characteristics of past data. The procedures adopted are of either a qualitative or quantitative nature. On one extreme, there is a purely qualitative approach that does not require any overt manipulation of data. Rather, it is judgmental, where judgements made are based on the results of mental manipulations of past data. On the other extreme, there is a purely quantitative approach that does not require any input of judgement. It relies on researched mathematical and engineering formulæ to derive quantitative estimates from sources of financial data. Whichever approach is adopted, common-sense remains the most important requirement for intelligent forecasting to be effective.

In a quantitative approach, the manipulation and analysis of the data is carried out as an extrapolative (time-series) model or a causal (econometric) model. The choice is dependent on circumstances and the usage of the forecast. A time-series model is selected for prediction by applying the value of a variable and extrapolating the past values of the same variable. By contrast, a causal model is used for prediction by applying the value of a dependent variable and identifying its causal relationships with the values of other independent variables. In either case, the techniques used for forecasting fall into three analytical categories, fundamental analysis, technical analysis, and quantitative analysis.

Fundamental analysis is to find speculative opportunities from economic data by identifying potential major shifts in trends in the balance of the market supply and demand.

Technical analysis is to speculate on the supply and demand of the market based on the assumption that the market moves in cycles and there is repetition of its trend as patterns in price charts.

Quantitative analysis is to speculate by formally constructing a market model and computing an efficient frontier which would offer a maximum reward according to the amount of risk that is to be taken.

2.1.1 Fundamental Analysis

The core activity of fundamental analysis is to assess the trends of corporate profits and to analyse the attitudes of investors toward those profits with a focus on the supply and demand pattern. The information used is often derived from a range of marketing figures that are thought to influence the product or financial instrument being analysed. Amongst the figures normally studied are economic data - the gross national product, the balance of payment and the interest rates - corporate accounting reports, and currency rates. The methods used to uncover the supply-demand trend do not normally analyse the prices directly. Instead, there are analyses of the trends made by prices and those market variables that influence the direction of the trends. The variables incorporated in such analyses would often include seasonal influences, inflation, market response, government policies and international agreements. The opinions derived from these fundamentals are normally used to influence the medium to long term policies of future investments.

Old Hand Approaches

The most '"noble" approach to time the market is none other than using the coveted prowess of the old hand, informally evaluating an assorted combination of beliefs and anecdotes. Even then, it is often not unusual and against all odds, that the evaluated strategy is simply overridden by a mere "strong gut feel".

Tabular And Graphic Approaches

The tabular and graphic (TAG) approach is a systematic way to examine a balance table for the relationship between the supply and depreciation statistics of prices. To illustrate the approach, the balance table in Table 2.1 is a simple model to project hog prices. The columns of data reflect the relationship between the slaughter levels and the price of hog for contracts to be delivered in June for each of the years tabulated.

Year	Dec-May Slaughter (1000 head)		Dec-May
	Hog	Cattle	Avg Price June Hogs + PPI* (cents / lb)
1976	34,691	21,137	25.9
1975	37,854	19,324	26.2
1978	38,947	20,295	22.9
1977	39,435	20,641	20.5
1973	40,292	16,889	26.6
1974	41,184	17,230	27.0
[illegible]	[illegible]	17,264	[illegible]
1972	45,108	17,443	23.3
1981	47,479	17,063	17.9
1971	49,087	17,318	17.7
1980	49,286	16,284	15.3

*Product Price Index

Table 2.1 Hog prices vs Hog And Cattle Slaughter[Schwag85]

It is obvious from the table that there is a correlation between the high prices of hog and its slaughter levels. However, it appears that the prices in the shaded band, are relatively low when they are compared to the other years. An explanation for this might have been that prices were competitively set to offset an influx of alternative meat supply shown by the high increase of cattle slaughter for the same period. In addition, it is noticeable that for 1974 and 1979, there are similar levels of hog and cattle slaughters, and yet the prices of hogs are much higher in 1974. Again, there are such similar slaughter levels and prices differences for 1971 and 1980, and 1972 and 1979. This observation suggests that there is a trend for the price of hogs to fall in the latter years and it could possibly be due to a shift in the consumption of red meat.

This simple model highlights three possible variables that can determine the prices of hog: hog slaughter, cattle slaughter and time. In addition, it shows the projection of the market for hogs by applying marketing knowledge to the analysis of a selected set of historical information. Following this simple example analysis, it is clear that the TAG approach could not be made to work well if the explanations of price fluctuations are due to more than one factor. This view is also consistent with Remus' [Remus87] report on the relative impact of tabular and graphic displays for decision making. His investigations showed that tabular displays are, by far, more suitable for decision makers to weigh the appropriate factors in low complex problems. By contrast, graphical displays also play a significant role up to an intermediate level of environmental complexity. It is therefore, not surprising that in real situations where multiple factors can be involved, the TAG approach becomes a cumbersome forecasting tool. When such a situation arises, it is not unusual to formalise the TAG approach with a more efficient mathematical approach like regression analysis.

As an example, the TAG relationships for the December to May hog analysis can be translated to a linear equation like,

$$P = a + b_1H + b_2C + b_3T \qquad (2.0)$$

where P is the average price of June hogs divided by the PPI, H is the hog slaughter, C is the cattle slaughter, and T is the time trend. The regression equation (2.0) defines the price level, P that corresponds to any combination of hog and cattle slaughters and time. The values of the regression coefficients a, b_1, b_2, and b_3 can be determined by a multiple regression model and subsequently substituted into equation (2.0) to obtain a price forecast for hogs. A discussion on the mathematics of multiple regression can be found in [Schwag85 and HanRei89] for example. A linear regression model is also outlined in the next section on Technical Analysis.

2.1.2 Technical Analysis

Unlike fundamental analysis, technical analysis is a study on the reaction of the market players to their assessments of any relevant market factors that can have an impact on the prices. It is based on the belief that the price chart is an unambiguous summary of the net impact of all fundamental and psychological factors, and that the major market turning points are the result of the market players themselves unbalancing the supply and demand equilibrium. This subject had evolved over a period of time into five main ways of analysing the internal structure of the market. They are, classical charting techniques, statistical techniques, mathematical techniques, system trading techniques, and behavioural techniques. Generally, these techniques are highly dependent on the vagaries of human behaviour and judgements on the common-sense forecasting principles on which these models are based.

Classical Charting Techniques

The art of speculation can be traced as far back as the sixteenth century, to the Japanese rice markets. These rice merchants had devised a method which is similar to the Japanese Candle chart technique to obtain insights of the market by representing prices as picturesque descriptions and interpreting them using ideas which are often criticised by forecasting sceptics as similar to practising folklore.

The Japanese Candle chart is a price chart which represents the relationships between the open, close, high, and low prices as a candle with a wick. The body of a candle marks the spread between the open and close prices which is usually left with a space in between. The space is always filled as a red box if the open price is lower than the close price, but it is a black box if the situation of the prices are the reverse. Following the markings of the intra-day high and low beyond this range, they are joined to the body by a thin black line to give an impression of a wick. The specific importance of a chart is due to a combination of instances of candles with different shapes and sizes resulting in a variety of pattern formations. Each formation is recognised by a poetic description and any interpretation of its significance is based on myths surrounding the pattern. For example, the simple patterns on the top row of Figure 2.1 give a view of the general direction of the market based on the unique interpretation of the relationships of intra-day prices. They are then used as building blocks for more complex patterns like those along the bottom row. Each pictorial representation has a name which reflects its expected behaviour as it is told in folklore. The behaviour aspect is an important part of candle charting. It is an indication of the strategy to adopt in anticipation of the market behaving as predicted. For example, the formation of a window is to caution one to anticipate the "closing of the window" with an imminent change in the direction of the market. In the event of an unclosed window, it is a signal for the continuation of the present market trend.

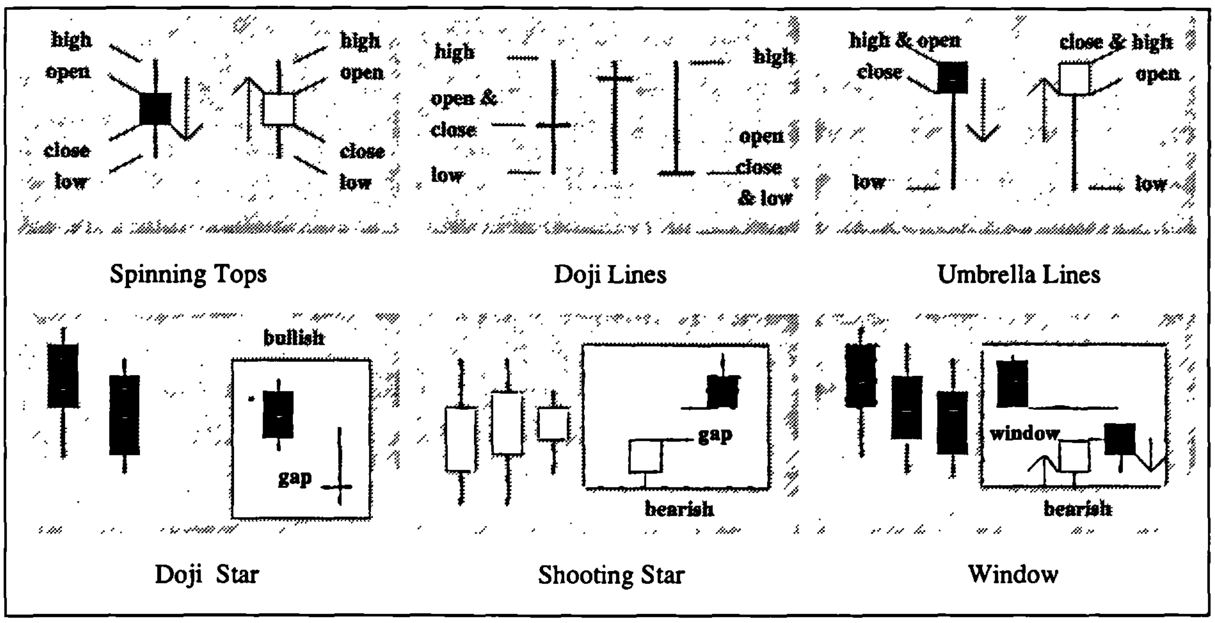


Figure 2.1 Some Basics Of Candle Chart Lines And Indicators [Nison91]

To obtain a current insight of the market, the price chart has to be reviewed constantly for the complete formation of any of the known candle patterns. In spite of this inconvenience, proponents believe that exploitation of the market is possible through the recognition of the known trends. As in candle charts, the western practise of charting is also concerned with the recognition of various descriptions of trend lines. Although the trend lines are derived from a different reasoning, they are equally mythical. Amongst the descriptions used are, *head and shoulders*, a *triangle*, an *island reversal*, and a *double top*. Sources for further reading on charting are in [Pring80, Schwag85, Murphy86] and [DunFee89, Nison91] for candle charts.

Charting techniques which rely on non-scientific general principles and which are entirely dependent on subjective interpretations do have their pitfalls. First of all, the patterns formed in a non-linear chart may not be clear, as they are likely to suffer some degree of distortion due to time warp. As a result, all interpretations demand craftsmanship and even then they can be inconclusive. Even if a view is correct, this method of analysis is reactive, that is, by the time a trend line is recognisable as a result of the formation of a pattern, the speculator would only be able to speculate by reacting to the market situation and not so much as to exploit the market by anticipation. Despite such a major shortcoming, this technique has remained very popular and continues to be used especially to reinforce the views of some of the other techniques to be discussed later.

Statistical Techniques

Following the popularity of computers, statistical techniques have also become popular as a means to automate what is seen as a key element of charting, pattern recognition. Many statistical methods have been studied to formalise the common sense principles used by players in the market. Amongst the list of formulæ studied to reflect the common principles used to explain the supply and demand of the market are: momentum, stochastic oscillator, moving averages, and Relative Strength Index (RSI). One of the goals of these methods is to turn an intra-day or daily price data into technical indicators that highlight certain aspects of the activity of the price. These indicators are to add precision and to eliminate some of the guesswork made to plot investment strategies.

The RSI indicator is a reading to compare the price at different times within the same chart. The calculation of the indicator to range from 0 to 100 uses the formula,

$$RSI = 100 - (100 / <1 + RS>) \tag{2.1}$$

where RS is the ratio of the exponentially smoothed moving average (EMA) of gains divided by the EMA of losses over a chosen time period, the tracking window. The dimension of the tracking window is one that is right and appropriate for viewing short, medium or long term charts.

In its simplest form, the indicator suggests an overbought market when the reading is above 70, and an oversold market when it is below 30. Such a reading merely indicates that the market trend is liable to see a short-term correction before too long. However, it is not a reliable indicator to signal buying or selling. At the most, an extreme RSI reading is a rather promising signal as it suggests that the market is likely to have to make a strong move to correct its present trend. When the correction has taken place, the main trend is also likely to follow the new direction for a while.

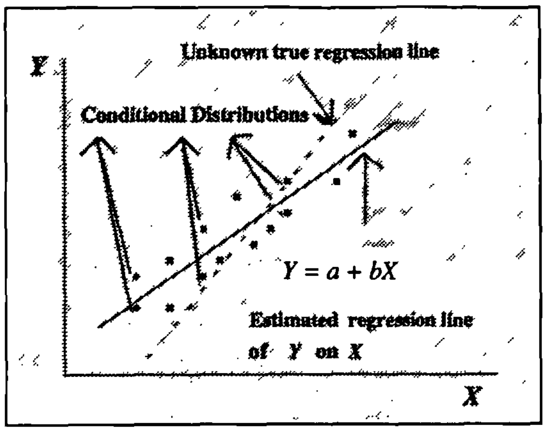
Despite its weak signal, it is possible to enhance the value of an RSI indicator. An example is to use it as a comparator for recognising divergence. When a market makes a new high or a new low but not a corresponding new high or low for the RSI, it indicates that the market is diverging. Furthermore, there is a positive divergence when there is a lower market low but not a lower RSI. The emergence of this market signal simply implies that there is a continuing supply by sellers which can mean that it is fairly probable that the price is likely to continue to go down. As a result, a positive divergence is interpreted as a buy signal and it has been found to be fairly reliable. In practice, the signal is always applied with consolidating evidence from other indicators. This is because the early onset of a divergence can indicate that the market can either diverge significantly to attain a lower low or it can also move but not too significantly from its current low position. A similar reasoning also applies to a negative divergence when the market is at its top end.

The reverse of a divergence, a convergence, is the situation when the RSI confirms the direction of the market by taking a new high or a new low. It is a sign to suggests that the market is likely to continue to follow its current trend, and like divergence, it is often applied with caution.

Like other technical indicators, the RSI has the ability to hint at an imminent market correction or a market trend. These signals are, however, often inconclusive and any potential misleading move has to be eliminated by other indicators and market opinions. In any case, making good decisions with it would require a substantial amount of practical experience. In the experimentation with other indicators, the RSI has been found to be useful with other techniques such as, the Elliot Wave Theory. A range of these popular techniques are sold as proprietary packages like Comtrend, TeleTrac and Tradecenter. They offer facilities for consultation with different methods within one price chart updated in real-time. It has the disadvantage for not allowing the technical indicators to be optimised as they are used in a model.

Mathematical Techniques

The development of mathematical methods for forecasting is initially aimed at econometric modelling of fundamental factors. Unlike time-series forecasting where it is normally used by both mid- and first-line management for short-term management, there is a strong association of mathematical methods with causal model forecasting to set the general course for medium- to long-term investment management. In most situations, it is the aim of management to predict the value of a variable (the dependent variable) given that there are many variables (the independent variables) with causal or explanatory relationships for evaluation. The range of methods used include regression analysis, Box-Jenkins' (ARIMA), auto correlation analysis, and Kalman Filters.



Y
Unknown true regression line
Conditional Distributions
$Y = a + bX$
Estimated regression line of Y on X
X

Figure 2.2 The Simple Linear Regression Model

The simplest form of regression analysis is the simple linear regression of two variables. It is an optimisation method that finds a straight line that can best fit all the data points in the sample set. For example, if the dependent variable is symbolised by Y and the independent variable by X, then the overall distribution of Y values is termed the marginal distribution. Then the distribution of Y for a given value of X is known as the conditional distribution for that particular value of X. In any simple regression problem, there is one marginal distribution and a number of different conditional distributions for each different value of X. As a result, to say that the relationship between Y and X is linear implies that the mean values of all the conditional distributions lie in a straight line which is known as the regression line of Y on X.

Referring to the example in the TAG approach, a simple regression model to express the measurement of the true regression line is,

$$P = a + bH \qquad (2.2)$$

where a is the estimate of the true intercept and b is the estimate of the true slope. The estimation of the best fit line usually uses the least square criterion to minimise the sum of the squares of the vertical deviations of each point from that line. The vertical deviations are in fact a measure of the errors in the dependent P, and it is squared to remove the relevance of points that lie above or below this best fit line. Minimising the squared vertical deviations in P produces the regression line P on H. Assuming that, (h_1,p_1), (h_2,p_2), . . . , (h_n,p_n) are the data points in the sample set, then the squared vertical deviations,

$$SD = \sum_{i=1}^{n}(y_i - a - bx_i)^2 \qquad (2.3)$$

is minimised with the regression coefficients, $b = \dfrac{n\sum xy - \sum x \sum y}{n\sum x^2 - (\sum x)^2}$ and $a = \dfrac{1}{n}(\sum y - b\sum x)$. By adapting the basic equations of (2.2) and (2.3), the method can be applied for multiple independent variables and non-linear relationships between H and P, just like the ternary equation of (2.0).

The sequential processing of this method of forecasting can use two assumptions. First, it can assume that the conditional distributions throughout the sample set have the same standard deviation, implying that it is sufficient to apply a regression which is based on an overall estimate of the conditional probability rather than one using different estimates for different values of *H.* Secondly, by using the assumption that the conditional distributions are normal, the model is more likely to become invalid when the trend changes. As a result, these assumptions are rather inflexible for non-linear applications.

System Trading Techniques

On frequent occasions when the market volatility is high, it is not uncommon for financial players to succumb to the undesirable weakness of difficulty in isolating emotion from rational judgements. Emotions like fear, greed or stress are major factors which can impair prudence to cause investment damages.

Technical trading systems are serious attempts to simulate expert trading using a set of rigorously tested decision rules. They are often regarded as useful systems as they allow users to avoid using human judgements directly and the trading decisions can help to dampen the users' emotions in stressful times. To automate any trading tactics, any suitable technical methods that can generate valid technical indicators are applied and they are then optimised to give valid trading signals. These are then incorporated as a set of favourable conditions to meet in the conditional parts of decision rules that are specifically for recommending a "BUY" or a "SELL" signal.

A technical indicator that is widely used is the moving average (MA). It computes the forecasts by taking a fixed number of periods back from the current period and uses the arithmetic mean of the values in these periods as the next forecast value. Assuming that P is the tracking window, the value of the FTSE-100 Index, for example, X_i of the current period i, then the forecast for the future period, X_{i+1} is,

$$X_{i+1} = [X_i + X_{i-1} + \ldots + X_{i+1-P}] / P \tag{2.4}$$

For any value of P chosen by the user, equation (2.4) smoothes out variations in the values of X. By the nature of the calculation in the equation, it is unfortunate that the resulting forecast for the next period lags behind by one time slice. As a result any significant changes in the values of the time-series can only be detected after a time-delay. Despite this sensitivity to P, the MA can be suitably adapted to monitor the trend of the day-to-day fluctuations of prices. In addition, it is particularly useful as a screening device where a number of charts can be efficiently scanned to spot sudden price changes, momentum variations, new highs or lows, or variations in relative performances.

In the technical trading system described in [Loo89], the MA is used in a rule set to qualify a couple of financial adjectives from a stream of data signals. After processing, these MA deduced qualifications are dynamically asserted as facts and are used to match with the condition parts of decision rules as in Figure 2.3. In this particular example, a *bear* market is qualified when the price of the unit is less than the long-term MA. Conversely, it is a *bull* market. In the rule set, the calculation of the long-term MA is to measure the underlying trend of the unit, while the calculation of the short-term MA, over a suitable smaller tracking window, is to measure the recent price movement of the unit seen over that time window. Given that the system has to trigger the "BUY" signal with a certain amount of intelligence but without any judgmental input, an estimation based on twice the *volatility* is used as a conservative threshold to eliminate any false market moves.

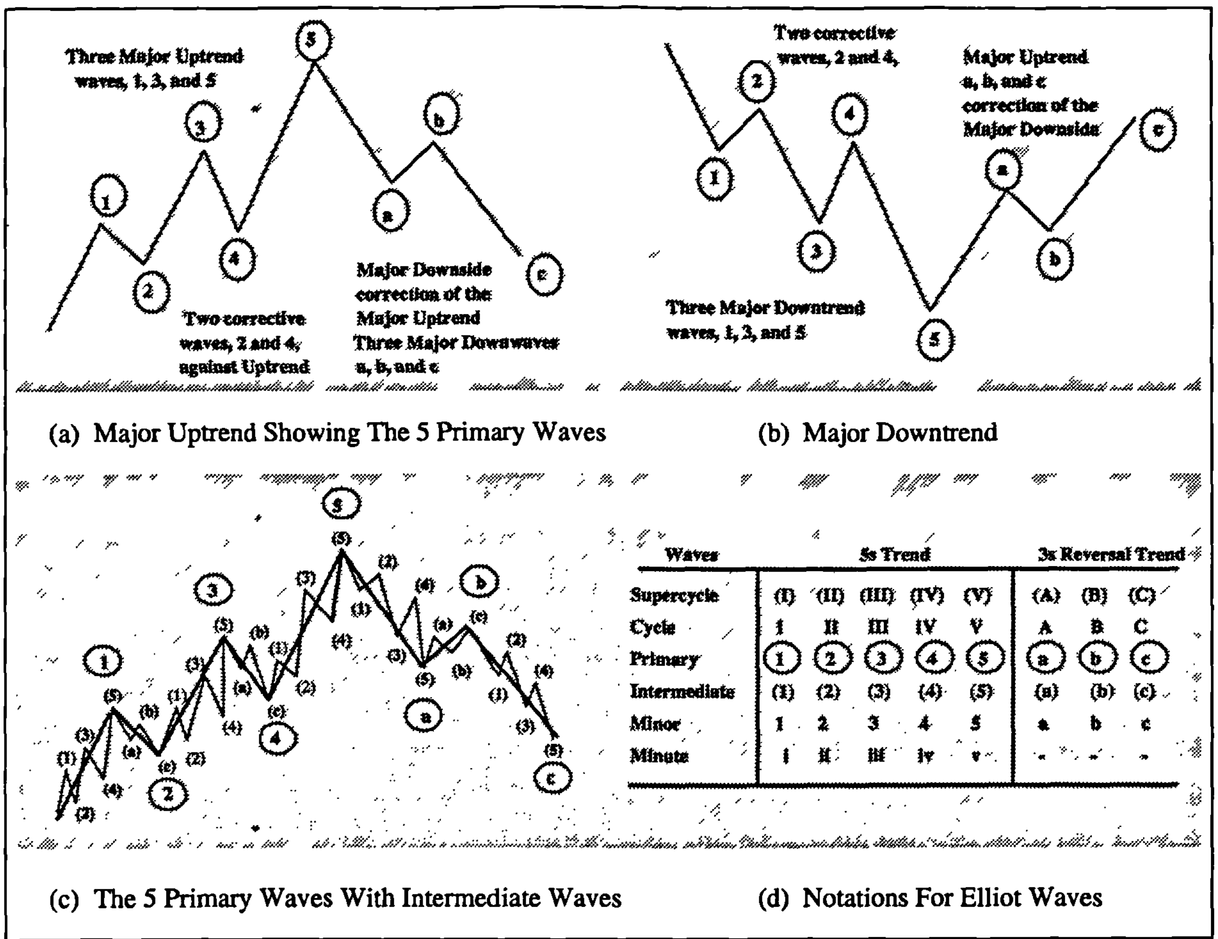
```
Rule R1
  IF the market is bear
  AND the price crosses the short-term MA from below
  THEN flag a 'BUY' signal but do not activate until
          the price penetrates the short-term MA by
          twice the volatility

Rule R1.1
  IF the 'BUY' signal has been flagged
  AND the price has not penetrated the MA by
          twice the volatility for 10 minutes
  THEN cut the 'BUY' signal
```

Figure 2.3 Rule Sets For A Technical Trading System

The example outlined highlighted three weaknesses using a statistical measurement, like MA for technical trading systems. Firstly, as it has been mentioned earlier, the nature of the formula is such that the programming can only allow the system to perform based on information from the last market cycle. Consequently, a MA oriented trading system is unable to anticipate. It is reactive rather than proactive. Secondly, MA forecasts with a time lag, and is therefore insensitive to ranging markets. The fact is, markets are more often ranging then trending. As a result, the system can only perform very well for short periods when the market is more trending as opposed to ranging. Thirdly, the decision rules lack a dynamic learning ability to adapt to the current market sentiment and the state of the user's balance account. This is partly due to the limitations of current technology and partly to the difficulty in computing some of the market factors in real-time. Even if it were possible to include some tangible factors (like a *bear* market) into the rules, their definitions are difficult and often inadequately defined for non-linear situations, so as to not allow the system to be flexible enough to perform satisfactorily. For instance, the conservative measure of twice the *volatility* may well result in many good opportunities going amiss.

Behavioural Techniques

Behavioural techniques of timing the market work around the outdated aphorism that "the market is made by men but the market is a woman". In practice, the principles used to exploit the human herd instinct spot opportunities on the basis that the time to buy cheap is when every one is selling. Their application is made on the understanding that a stampede of sellers would naturally shift the supply and demand equilibrium to force a price decline. Speculators who have benefited from exploiting the crowd psychology of the market players have evolved and developed theories like Elliot Wave Theory, Gann and Cycle Theories.


Three Major Uptrend waves, 1, 3, and 5
Two corrective waves, 2 and 4, against Uptrend
Major Downside correction of the Major Uptrend
Three Major Downwaves a, b, and c
(a) Major Uptrend Showing The 5 Primary Waves
Two corrective waves, 2 and 4,
Major Uptrend a, b, and c correction of the Major Downside
Three Major Downtrend waves, 1, 3, and 5
(b) Major Downtrend
(c) The 5 Primary Waves With Intermediate Waves
(d) Notations For Elliot Waves

Waves	5s Trend					3s Reversal Trend		
Supercycle	(I)	(II)	(III)	(IV)	(V)	(A)	(B)	(C)
Cycle	I	II	III	IV	V	A	B	C
Primary	①	②	③	④	⑤	ⓐ	ⓑ	ⓒ
Intermediate	(1)	(2)	(3)	(4)	(5)	(a)	(b)	(c)
Minor	1	2	3	4	5	a	b	c
Minute	i	ii	iii	iv	v	[illegible]	[illegible]	[illegible]

Figure 2.4 Elements Of Elliot Wave Theory

Through years of refinement, they have become particularly effective especially when it is used in conjunction with other technical indicators and observations from classical charting.

The most widely used technique to visualise the movements of the data according to market psychology is the Elliot Wave Theory. It analyses the market using a number of axioms on three chart elements: the pattern, ratios and time. The first axiom states that the market moves in the direction of the overall trend in a series of five waves (5s) followed by three waves (3s) and that this series of 5s and 3s will fluctuate against the general trend as they evolve. In theory, the impulse waves of Figure 2.4a, referenced by the uptrend waves 1, 3, and 5 and the corrective waves 2 and 4 are representative of a pullback from the uptrend. Subsequent correction to this 5s uptrend is represented by the 3s waves, a, b, and c.

The second axiom states that a formed 5s-and-3s cycle will itself be repeated recursively to a diminishing degree within a wave of the cycle. In the illustration in Figure 2.4c, the impulse wave 1 followed by a subsequent corrective wave 2 will break down into 5s-and-3s of one lesser degree, which again will break down into even smaller cycles of 5s and 3s. The depth of the cycles will bear a different degree of significance and for recognition purposes a naming system (Figure 2.4d) is a characteristic of the Theory. Due to the long duration taken to form the Supercycle and the impracticality of the minuette and subminuette cycles, their labels are often left out, in practice.

Once these waves are recognised, there are a number of maxims that can be practised, from time to time, to exploit some distinctive characteristics that are peculiar to them. That is, of course, if they exist. In the first instant, it is a belief that the middle impulse wave, wave 3, will never be the shortest impulse wave, rather it is normally the longest. The characteristic of its formation is usually an upsurge in volume and an extension of its breadth in comparison to the other waves in the cycle. In addition, it is also a belief that the bottom of the second correction wave, wave 4 will not fall below the top of the first wave 2. Also, there is a rule which says that the simplest form of a correction will consist of 3 waves moving 'down-up-down' and that it will retrace to the region of wave 4 of the previous bull move, often going below to reach its low point.

The third axiom states that the 5s-and-3s cycle will conform with the Fibonacci Summation Series (1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89,..). That is to say, the 5s waves and the 3s waves which make a complete Elliot cycle of 8 waves in total, are all Fibonacci numbers. In fact, as it is (Figure 2.4c), the sums of the waves spanning the cycles of lesser degrees will also obey the rule of the Fibonacci series.

The fourth axiom states that the lengths of the waves will also be related to the Fibonacci ratio, 1 to 0.618 or 1 to 1.618. There are three ways to obtain these numbers. It can be a division of a number like 55 by its successor, 89 to give 0.625. Alternatively, it can be a division by its predecessor, 34 to give 1.60. Lastly, the ratio can be obtained by dividing alternate numbers such as 1.618 by 0.618 to give 2.618 for special cases, as when the wave is thought to have undergone an extension.

Although the Elliot Theory follows a common theme on the relationships amongst the patterns, ratios and time, there are many variations to these basic axioms and exceptions to the rule to accommodate the non-linearity in the price charts. Consequently, its beneficial use often requires a trained eye, and even then, the recognition of some of the wave formations has remained a creative art.

2.1.3 Quantitative Analysis

Quantitative analysis is a scientific approach to forecasting that is becoming increasingly fashionable with the advances in technology. Unlike technical analysis, quantitative methods use economic theories to build a market model that measures the expected risk and reward equilibrium of the market in a systematic manner for rational decision making. An example of such a rigorously analysed method is the Modern Portfolio Theory (MPT). It is centred around the Efficient Market Hypothesis which says that given the large number of players in the market, their speed of reaction to any new piece of information that might affect a share's price will result in the market becoming overbought or oversold too quickly to allow the players to make any profit out of it. The MPT refutes this Hypothesis. Instead, it views the market as not efficient and with adequate management of the market uncertainty, rewards can be reaped.

The framework of the MPT is about the measurement of two investment criterion: the reward and the risk. It introduces the idea of a potential reward, the expected rate of return, for investing in a security or portfolio over a holding period. The reward expected is defined as the sum of all the products made for all the securities in a portfolio, the projected profit to be made from the projected price appreciation of a security and the probability of its occurrence. In association with this expected return there is also the uncertainty of potential risk. This variation is the standard deviation of the normal probability distribution function. As a result, the task of selecting a portfolio involves making a projection followed by a selection procedure. The selection is made either because a portfolio offers the greatest expected return for the smallest degree of risk involved, or it is according to the risk threshold that an investor is prepared to bear. Due to this dual selection criteria, the measurement of the risk exposure preferred by the investor uses a concept such as an indifference curve. The curve is derived by calculating the relationships of the expected return of all the portfolios against their standard deviations, resulting in a list of portfolios lying along its edges. It is possible to deduce from the curve, the expected level of risk a exposure of a portfolio as there will be a list of portfolios lying along its edges with its expected reward known.

The construction of a portfolio such as the one in Figure 2.5, involves the computation of a set of all the possible expected rewards and risk combinations corresponding to all the securities in an approved list. Running along the edge of this attainable set is the *efficient frontier* which are sets of portfolios that give the highest expected return for a given level of risk running along it. Such is the case that in practice, selections are made on those portfolios lying along the frontier rather than those that are lying further inside the attainable region. To obtain the optimal portfolio, the *efficient frontier* is intersected with a set of the indifference curves plotted on the risk-reward region. The optimal portfolio for an expected risk lies at the point where the curve with that expected risk just touches the *efficient frontier*. To understand the aggregation of the financial implications of all the portfolio selections,

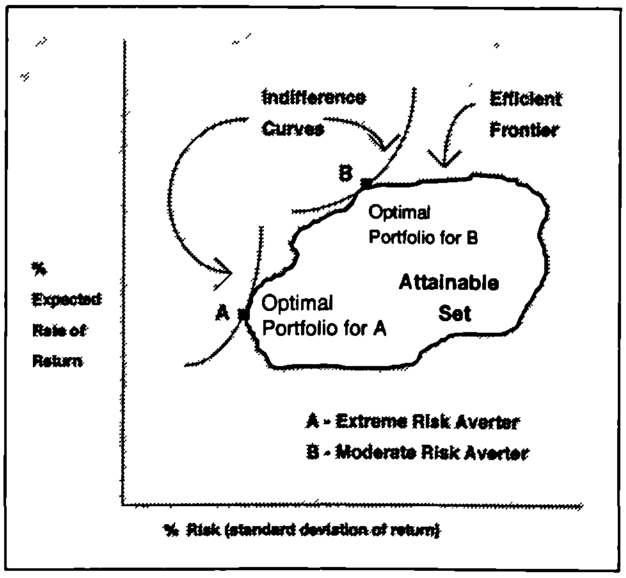


Figure 2.5 Portfolio Selection By MPT

the Capital Market Theory (CMT) has been introduced to describe the effects of portfolio decisions on the market. In particular, it focuses on the relationship between the expected return of the security and the total risk involved.

The MPT investment concept decomposes the total risk into,

$$\textit{Market risk} = \textit{Systematic risk} + \textit{Residual risk} \tag{2.5}$$

The definition of the total risk is derived from their sources of origin. It interprets the *systematic risk* as those risks that are often tied onto the uncertainty of the performance of the market portfolio. On the other hand, it views the *residual risk* as those risks that evolve from any other sources. From the risk relationship of (2.5), research on the portfolio risk applies three main assumptions which are influenced by the concepts of CMT. Firstly, the *systematic risk* is the only source of uncertainty for the rate of return of an efficient portfolio. As a result, an efficient portfolio does not have a *residual risk* and finally, a portfolio or a security with a *residual risk* does not lie in the capital market line [pp 23 of RudCla88].

Primarily, the core research in the MPT is the origins of the *residual risk*. It is,

$$\textit{Residual risk} = \textit{Specific risk} + \textit{Extra market covariance} \tag{2.6}$$

The first component of the residual risk, the *specific risk* is an element of risk that is exclusive to the security itself. The second, is an element of risk that is a result of the sum total of the securities held together in the portfolio, but which has no relations to the market. For example, a security on entertainment can have a *specific risk* like the risk of an unexpected bankruptcy and an *extra covariance risk* like the risk of poor consumer expenditure during a recession. In summary, the MPT is a formal framework to embody the assumption that there are several causal factors that influence prices and consequently, the returns of the securities.

The research efforts taken to measure these different aspects of risk explicitly rely mostly on statistical methods like Bayes Theorem and regression analysis. As earlier, sequential processing of information has its setbacks, and the power and novelty of the MPT is by no means not hindered by the limitations of computing techniques.

2.2 Soft Logic For Knowledge Programming

In Anderson's research [Anders82] on the verbal protocols of investment decision-making, he reported that the reasoning process often used a number of operators. These he observed, are usually qualitative characteristics of data provided in the information set and they are used in situations where the data is not clear cut. Amongst such descriptions are, "very large" and "volatile changes".

Due to the abstract and subjective nature of these descriptions (adjectives), it is rather unlikely that they are incorporated into financial decision systems. To redress part of this difficulty, there is a proposal to exploit the adaptive and parallel processing nature of "soft" logic [MITI90, TreLoo90 and TreGoo92] and to use them in an integrated hybrid system. By adopting a synergy of these "soft" processing techniques, it is possible to have Knowledge Programming as a prime feature in the next generation of financial decision systems.

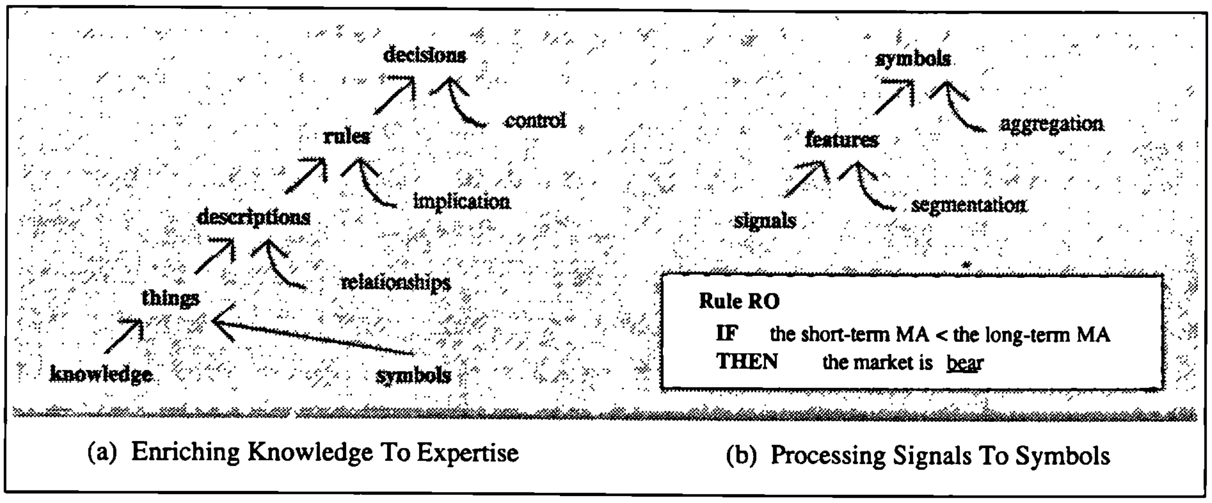
Knowledge Programming is a programming practice incorporating a hybrid of "soft" techniques for sub-symbolic processing of knowledge to deliver expertise which are intangible or unclear and to use them for symbolic reasoning. Its motivation is to extend the horizon of symbolic reasoning by extending the choice and quality of facts like those described by Anderson for evaluation. The Programming involves two phases: Soft Information Processing followed by Symbolic Reasoning.

The Soft Information Processing Phase is to extend the traditional idea of data analysis as a repetitive number crunching exercise to include the sub-symbolic processing of parts of the knowledge that are frequently used in making investment decisions. Specifically, it aims to transform raw data into knowledge by manipulating streams of numeric data to extract two types of knowledge. The first is to derive numeric facts from quantitative calculations. It is normally carried out for noise reduction, forecasting, or numerical analysis. An example of this knowledge extraction is the extrapolation of T-bond prices using a smoothing function like moving averages for a statement like, `the price of T-Bonds`. The second type, categorised as higher order facts, are qualifications that are moulded from sets of numeric data. Typical examples are those adjectives described by Anderson, and market descriptions like *bull, bear* or *stable*. By dynamically qualifying these "difficult to quantify" adjectives and represent them as facts, future systems are potentially more robust given that it is then possible to include statements like, `the price of T-Bond is` *`rising`* for evaluation.

Following from the Soft Information Processing Phase is the Symbolic Reasoning Phase to model the reasoning mechanisms of human decision makers. It uses a pool of knowledge which includes those that are static in the knowledge-base, qualified in the Soft Information Processing Phase, and any that have since been deduced dynamically as a result. By applying a "soft" technique in the Soft Information Processing Phase to qualify those "difficult" adjectives, it will be possible to extend the scope of the next generation of systems to allow the addition of decision strategies such as, `IF the price of T-Bond is` *`rising`* `THEN <recommendation>` into its existing rule sets. The rule is an example of a commonly used management reasoning statement but which will not be implemented simply because of qualifying the adjective, *rising*.

2.2.1 Soft Information Processing

Just as previous generations of computing manipulate data that are of a numeric form or are conditional probabilities from a database, the generic element to Symbolic Reasoning is knowledge. In its most primitive form, the logical representation of knowledge of real-world objects are symbols residing as facts in the knowledge-base of a system. Prior to their assertions in the knowledge-base, the basic elements that constitute a symbol are data signals. By an explicit and progressive linking of a stream of these signals, these raw elements can be enriched to form non-numeric knowledge structures representing concepts and relationships, constituting the elementary symbols of expert knowledge. The epistemology for the conceptualisation of signals to expertise is a taxonomy shown in Figure 2.6a [Fox84].



Rule RO
IF the short-term MA < the long-term MA
THEN the market is bear

(a) Enriching Knowledge To Expertise
(b) Processing Signals To Symbols

Figure 2.6 Processing Knowledge For The Qualification Of Expertise [Fox84]

As a great proportion of the raw data received for real-time applications are data signals, they have to be sampled at a high frequency as their evolution is dependent on the changes of exogenic factors. As a result, it is common to apply signal processing techniques like Kalman filters to remove some of the noise before the aperiodic data series is analysed. Although this particular filter discovers the hidden periodic pattern in the data series, it is like other types of filters, good for increasing the signal to noise ratio, but unfortunately it is not good for extracting knowledge about the signals. In order to achieve the latter, it is possible to use knowledge-base techniques to process signals to map our visualisation of these signals to a suitable symbolic form. The qualification of these signals to symbols follows the schema outlined in Figure 2.6b.

To translate this schema into practice, the processing of rule **R1** (from Figure 2.3) is examined. In the conditional part of the rule, it is possible to formulate the symbol, *bear* dynamically by applying the knowledge about the relationships of the stream of data signals that exhibits such a market pattern. A rule of thumb that can be used to formulate the notion is rule **RO** (Figure 2.6b), as it was used for the trading system described by Loo [Loo89].

Initially, the qualification process (Figure 2.6b) involves the segmentation of the data signals to a sub-event that can be pre-encoded symbolically and the polling of this sub-event to match with the conditions of rule **R1**. To encode a sub-event, a smoothing function like the MA equation, (2.4) can be applied to smooth the data signals over a suitable tracking window to form a continuous stream of segmented signals. The segmentation process uses two tracking windows to exploit the discontinuities of the signals at two time horizons. They are to form two features from the signals: the short-term and long-term MAs. Following these calculations, the aggregation process applies these two segments of knowledge onto the relationship in the conditional part of rule **R0** to deliver the encoded symbolic value, *bear*. It is then entered into the knowledge-base as a fact for pattern matching on rules like **R1** in the Symbolic Reasoning Phase. It is discussed in Section 2.2.2.

Neural Networks

The choice of using MAs to process technical information about a stream of data signals so as to enrich them into meaningful knowledge (as the symbol *bear*) has its shortcomings, as it has been highlighted in the section on Technical Analysis. Such an application for processing signals is inflexible because it is not appropriate or even possible to qualify a market description like *consolidation*, *whipsaw* or *stable*. Subsequently, it may only be possible to implement decisions regarding certain states of the market and not others.

Neural networks are a more suitable programming tool for the processing of segmented non-linear data signals by the schema of Figure 2.6b. This is because the dynamic adaptation of the "soft" neural networks can not only overcome the limitations of MA, it can also improve the quality of the information extracted from a stream of non-linear financial data signals: they are prone to time warp and distortions. In addition, the networks can eliminate common problems that are embedded in the signals. For instance, the neural networks can associate and therefore, define a pattern description before a pattern subscribing to a qualifier is fully formed. In addition, it can discover regular pattern features and avert problems such as sampling errors and spurious data.

The potential use of a "soft" neural network for knowledge extraction is discussed in conjunction with an example system to manage risk in a volatile market in the Symbolic Reasoning Phase. In the system, the recommendation of a strategy to avert risk exposure requires a preliminary assessment of the profit trend of the market players. At any given time, their profits can assume any one of the three states: *increasing, reducing* or *stable*. The views on the trend is specific to a time window which itself is dependent on the investment horizon being looked at: short-, medium-, and long-term. To qualify these adjectives from a time-series of the profits, it is proposed that a Backpropagation neural network is trained to recognise samples of the three main profit trend descriptions.

Given that the qualified symbol is to be used in a volatile condition, the notion of short term in a day-to-day trend analysis suggests that an evaluation is made every half hourly over a four hour period. By that token, the neural network can be configured according to the duration of evaluation. That is, the network may have 16 input nodes, 8 hidden nodes and one output node for loading patterns formed from a series of quarter hourly profits. The training pattern is a selection of classic samples of the patterns shown within various 16 time slices of time-series data.

To help the network to learn more rapidly, the patterns to be loaded are ordered so that it has an early exposure to the two extreme profit patterns: *increasing* and *reducing*. Following these are versions of a *stable* description. Corresponding to each of the three, the target pattern can be set to 0.01 for *increasing,* 0.99 for *reducing* and 0.5 for *stable.* The training process involves repetitively cycling each of the patterns through the network using a batch training strategy until the network converges to an average error of 0.25, if not better. Following the successful training of each of the patterns, the network output is fed to a fuzzy rule set to transform the numeric output to one of the three symbols representing the three profit trends. The trained network is then tested on new test patterns, the results of the qualification confirmed manually.

The same technique can also be applied for the qualification of the market trends, *bull, bear* and *consolidation* replacing the method using MAs on the technical description of the pattern such as the one described by rule **R0** in Figure 2.6b.

2.2.2 Symbolic Reasoning

Symbolic reasoning is the method of manipulation of knowledge (as symbols) that is uniquely the hallmark of AI. Like an AI system, the success of the Symbolic Reasoning Phase for Knowledge Programming is dependent on the total effectiveness of an appropriate knowledge representation tool that can map real-world objects accurately into symbols in the knowledge base, and an appropriate inference mechanism that uses an efficient search strategy to perform pattern matching of the knowledge within the knowledge base so that the knowledge can be linked together to form new knowledge representing skills, decisions and expertise. In AI, some facts are dynamically deduced by matching a set of static facts with a set of rules. Sometimes, facts that are not easily deducible are also used. For example, "is fairly hot" is usually measured indirectly by matching a user's impression of hotness to a range of numeric inputs corresponding to a range of descriptions of temperature. Unlike AI, Symbolic Reasoning in Knowledge Programming is not only able to reason with runtime deduced facts like these but it is also able to reason with machine learned facts like the qualification of financial adjectives described for the Soft Information Processing Phase.

To illustrate the significance of this phase, a potential application of a rule-based risk management system is described. The experiment has been tested on an intelligent hybrid environment, ONNEX [RocKhe93] by Khebbal [KheTre93]. The strategies used for monitoring a volatile market situation (Figure 2.7), in particular, the use of the financial adjectives, *reducing*, *stable*, and *increasing* reflect the infeasibility of the system if it is to be implemented using conventional computing methods. The application shows two important points: the potential role of "soft" logic to upgrade a financial decision-making system to a better level of functional competence, and it emphasises the significance of using a synergy of techniques for the success of future intelligent financial systems.

The architecture of a hybrid environment for this experiment should ideally be a network of hardware (processor) one of which is a master processor to schedule communications with the other slave processor(s) [Loo89]. Each of the slave processor has a specific processing function to deliver processed knowledge or data, which is then polled by the master processor at regular intervals.

```
Rule SR1a
  IF network_output is between 0.0 to 0.35
  THEN profit_trend is reducing

Rule SR1b
  IF network_output is between 0.36 to 0.65
  THEN profit_trend is stable

Rule SR1c
  IF network_output is between 0.66 to 1.0
  THEN profit_trend is increasing

Rule SR2a
  IF the profit_trend is increasing
  AND the position is making money
  THEN recommend increase position by 50%

          . . .
Rule SR2x
  IF the profit_trend is reducing
  AND the position is losing money
  AND the reduction is > 10% of loss limit
  THEN recommend reduce position by 25%

Rule SR2y
  IF the profit_trend is reducing
  AND the position is losing money
  AND the reduction is > 25% of loss limit
  THEN recommend reduce position by 50%
```

Figure 2.7 Rule Sets With Symbols Processed Dynamically

The set of rules, **SR2a-SR2y** in Figure 2.7 suggests that it is essential to have two slave processors operating concurrently with a master processor: one for tracking the profit trend and the other for assessing the current position of the market player. Like the qualification exercise in Figure 2.6, the qualification of the market player's profit trend also requires a processor that is dedicated to neural processing. The master processor polls the network outputs and performs pattern matching at two levels. First, there is the translation of the neural outputs to symbols according to rules **SR1a-to-SR1c**. Secondly, there is the recommendation of a suitable investment strategy according to the rules set by **SR2a-to-SR2y** using a combination of a neural qualified knowledge and knowledge that is deduced in the other slave processor.

A similar method of reasoning with neural qualified *bull* and *bear* descriptions of the market can also be applied to the trading system rule sets, rule **R1** (Figure 2.3) as it has been discussed earlier in the section on Technical Analysis.

2.3 Summary

This chapter illustrates the exploitation of "soft" logic to widen the boundaries of financial decision making systems. The organisation is in two parts. In the first it surveys the traditional forecasting techniques and the programming practices used to automate trading systems to highlight their strengths and weaknesses. In the second it introduces Knowledge Programming to complement and overcome some of the weaknesses discussed. Knowledge Programming is a programming practice incorporating a synergy of "soft" logic techniques, allowing sub-symbolic processing of knowledge to deliver new knowledge, those which are rather subjective and not clearly defined, for subsequent evaluation in symbolic reasoning.

The survey covers techniques that are commonly used in the three analytical categories: fundamental analysis, technical analysis, and quantitative analysis.

Fundamental analysis - the old hand approach and the tabular and graphic approach - is to speculate from economic data by identifying the shifts in the market trend to balance the supply-demand equilibrium. Technical analysis - charting techniques, statistical techniques, mathematical techniques, system trading techniques, and behavioural techniques - on the other hand, is to speculate on the balancing of the supply-demand equilibrium based on the belief that the market moves in cycles and that salient features from the past will be repeated in future prices. At the extreme end, a quantitative analysis method like the Modern Portfolio Theory, is specifically to speculate in the capital market by building a conceptual market portfolio to focus on the risk-reward equilibrium predicting the level of reward in relation to the level of risk exposure to be taken.

Knowledge Programming is motivated to exploit "soft" logic to simulate human reasoning by computing knowledge that have been conventionally difficult to qualify and to include these "new" knowledge as facts in decision-making systems. The computation involves the Soft Information Processing Phase followed by the Symbolic Reasoning Phase.

The Soft Information Processing Phase is primarily for the sub-symbolic processing of a stream of data signals to extract knowledge and translate them to symbols. An example use is described to qualify the profit trend as, *increasing, reducing* or *stable* from a time-series of profits. These financial adjectives are dynamically qualified using the "soft" neural network and fuzzy logic. The qualified symbols are then used as facts for pattern matching in a rule-based risk management system in the Symbolic Reasoning Phase. With the inclusion of these previously non-feasible facts for deduction, the recommendations offered by the system are more refined and sharp.

Chapter 3

System Specifications And Design

This chapter reports the development of the customised Neural Forecasting System. Initially, it describes the system specifications, with outlines of general requirements, such as the operating platform, the user interface and the system characteristics. Following this are the technical requirements with details of the Backpropagation model and the research facilities needed to facilitate neural forecasting. Lastly, it explains the system design, with outlines of the system processes mapping these specifications.

3.1 General Specifications

The development of a customised Neural Forecasting System (NFS) was for use as a vehicle to accomplish the goals of this thesis. In addition, it is to provide end-users the flexibility to undertake a variety of research on other forecasting applications with minimum inconvenience. The procedure for system development was to establish the general requirements to specify the environmental relationships of the proposed NFS with users. These special needs were then used as guidelines to establish the technical requirements on which the system design was based.

The general environment issues addressed were, the development language, the operating platform, and an appropriate user interface for direct access to those system facilities that have been identified as essential for research and optimisation. Following an assessment of the current facilities that were already in use at the end-user's site, there was a decision to exploit the portability of the 'C' language on a Unix platform, and to adopt menus as the standard interface with the NFS.

In so far that the neural technology is readily accepted by the end-users, the following three attributes influenced the system design.

Complete The NFS has provisions for all the system functions and utilities needed for undertaking neural research for both time-series and econometric forecasting. There are NFS functions to resolve data inconsistencies, and utilities to support historical and forward simulations.

Compatible All the relevant NFS input/output operations can read data files which have the unique formats of the Datastream proprietary source, and write outputs onto files for loading to other software packages such as a Lotus spreadsheet.

Easy-to-use A menu interface is used and simple terms and clear explanations given whenever applicable. The tedium associated with neural forecasting research is alleviated by research facilities which are directly accessible as system functions or embedded in these functions. A user manual is also provided for support.

3.2 The Backpropagation Model

It has been said in the discussion on quantitative forecasting methods in Chapter 2, that forecasting can be approached by extrapolative (time-series) or causal (econometric) models. For completeness, the **NFS** supports both forecasting options and makes a one-step forecast for one data recording time (time-slice) ahead.

The Backpropagation neural model for the **NFS** follows closely the original which was first described in [RumHin86]. The two-layer (exclusive of the input layer) network is fully connected and the network configuration is open to user specifications.

If the series of training patterns constructed from *tot_num* historical data is,

$$(I_1,T_1), (I_2,T_2), \ . \ . \ . \ , (I_{tot\_pat}, T_{tot\_pat}) \qquad (3.0)$$

Each time-ordered *(I,T)* pair comprises an input pattern, *I*, with a corresponding target value, *T*. The pattern, *I*, is made up of time-slices of raw data which have the same dimension as the dimension of the input layer of the neural network (further details are in Chapter 5). In a training session, a pair of *(I,T)* input pattern is presented to a network and the procedures for learning repetitively carried out for a specified number of epochs (cycles) to satisfy a sufficient mapping of the extrapolation of pattern *I* to the value *T*.

The learning algorithm for the **NFS** has two nested levels of control [RefAze92]. The outer control monitors the number of cycles to execute the learning procedure, the forward pass and the backward pass, for the *tot_pat* training patterns. The inner control which lies within the outer control also monitors the number of cycles but is only for the pattern loaded to be trained. At this level, a test is made to ensure that a training pattern requires training (see first enhancement). A pattern that requires one is then loaded and trained based on the on-line strategy: both the forward and backward pass are executed consecutively on a pattern-by-pattern basis up to the required maximum number of cycles. The control can be overridden and training ceases (see second enhancement below) if the error difference between the mapping is below a threshold value. On completion, control returns to the outer level and a new training cycle commences. For every training cycle commenced, all the training patterns are subjected to the same sequence of events within the inner level.

Prior to loading an input pattern each of its raw data, β_i' is scaled to β_i, to lie between 0 and 1 by,

$$\beta_i = (0.8\,((\beta_i' - min\_val) / (max\_val - min\_val))) + 0.1 \tag{3.1}$$

The values of *min_val* and *max_val* are the lower and upper bounds of the *tot_num* historical data. This equation is discussed further in the section on Notations in Chapter 5.

The forward pass calculates the latest mapping of the current *(I,T)* pair, applying the activation rule layer wise forwards, sequentially updating the nodes from the hidden layer. The new internal representation (state) of node x_i of layer, l, at training cycle t, is the summation of the products of each of the states of the nodes and their corresponding connection weights, w, as equation (3.2).

$$x_i^l(t) = \sum_j w_{ij}^l x_j^{l-1}(t) \tag{3.2}$$

where j ranges over all the nodes in the previous layer of l, which node x_i lies. The sigmoid transition function, f applied to (3.2) is,

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

The backward pass calculates the error between the network output and the target value, propagates the error measure backwards, and adjusts the weights of all the connections in the network. Starting at the output node and working back to the hidden layer, the weight adjustment during a training cycle, t is,

$$w_{ij}(t+1) = w_{ij}(t) + \eta\delta_j x_i + \mu(w_{ij}(t) - w_{ij}(t-1)) \tag{3.4}$$

where $w_{ij}(t)$ is the weight from a hidden node or an input node, i to node j, x_i is either the output for node i or is an input, η is the gain term (the `Learn rate`), δ_j is the error term for node j, and μ is the momentum term. If node j is an output node, then the error term is,

$$\delta_j = x_j(1 - x_j)(T_j - x_j) \tag{3.5}$$

where T_j is the expected target value of node j and x_j is the network output. However, if node j is a hidden node, then the term is calculated as,

$$\delta_j = x_j(1 - x_j)\sum_k \delta_k w_{jk} \tag{3.6}$$

where k ranges over all the nodes in the layers above node j.

In addition to these learning algorithms, enhancements are added to improve the speed of convergence. The first enhancement allows the network to converge to a transitional target based on the current achievement of the transformation of I to T. This strategy uses dynamic values of `Tolerance`, `Tolerance_Step` instead of `Tolerance` as a convergence criteria. This modification averts any unnecessary computation set by a fixed `Tolerance` value and which may also be difficult or impossible to achieve. The `Tolerance_Step` is dependent on ε, the average value of the sum of all the δ_j at that cycle. The calculation is made at every cycle of the outer control loop using,

$$Tolerance\_Step = Tolerance + (\varepsilon - Tolerance) * 0.25 \tag{3.7}$$

The second enhancement is the control used in the supplementary learning process of TOPIX [KimAsa90]. This modification is used in conjunction with the dynamic use of `Tolerance_Step`. The control applies the strategy that a input pattern will only be subjected to the learning routine if the error between the pattern output and the target value is greater than the current value of `Tolerance_Step`. It takes advantage of the assumption that as the training progresses, the network is converging and more patterns are likely to pass the network error test. Adopting such a strategy, therefore, reduces unnecessary computations.

3.3 Research Facilities

The process of neural forecasting research involves a series of repetitive fine-tuning experiments that are not much different, if not more tedious, than using statistical methods. The core simulation studies that can be made on the **NFS** are either historical simulations where the forecast is tested with historical data which "has already evolved", or forward simulations where the forecast is to be tested with data which "has yet to evolve". In either case, they can be applied to forecasting prototypes using time-series or econometric forecasting models.

The routines essential for research and optimising a neural prototype are,

(1) aligning "cleaned" data from a selection of data files for simulation studies.
(2) specifying and tuning the forecasting parameters for historical simulation.
(3) training and testing a neural prototype in historical simulation.
(4) applying a trained network for forward simulations (forecast for testing on live data).

To support these routines, there are research facilities,

(i) standardising and merging data files with non-uniform recording times and volumes.
(ii) modifying neural and system control parameters values for historical simulation.
(iii) specifying input pattern formation using one or more time-series data and loading to a neural network.
(iv) re-constructing a partially-trained network for continuation or recovery of training, or a trained network for forward simulation.
(v) recording experimental data and information for evaluation of neural prototype.

The details of the facilities are discussed in the following sections.

3.3.1 Raw Data Standardisation

The NFS can be used for a variety of forecasting applications applying one or multiple types of raw data. The data are retrieved from Datastream, the on-line proprietary data source, but the nature of release of these data could result in irregularities across the data files. The Datastream data files contain non-relevant information and there are outstanding problems to resolve even after the files have been "cleaned". The occurrence of one or both of the following irregularities in any data file is not uncommon:

(i) **Non-uniform data recording times.**

This problem is unavoidable and it is especially prevalent in economic indicators. The definitions of the indicators imply that the intervals of the data release and consequently, the minimum recording time are far longer than other data, such as currency prices. For example, the Retail Price Index is released monthly and the Gross National Product quarterly. The NFS supports three types of recording times: daily, monthly and quarterly.

(ii) **Non-uniform data volumes.**

This problem can easily arise by default or by accident. By default, the volumes of data files differ when they have different recording times: like a file for the Retail Price Index and another for currency prices. The problem can also arise in files with the same recording times. This is when there is no record of the data on the chosen date of collection but at some later date. For example, the S&Poors 100 Index is only available on Datastream from March 1984 but the equably popular Dow Jones Industrial Index stretches back to the 1970s. It is likely that a group of data files for research will have a common subset of data bounded by a common time frame.

The integrity of the data for research is important. To resolve the above problems, a facility for aligning one or multiple raw data files which could have a different data recording times or possibly different data volumes is essential.

3.3.2 Parameter Optimisation

One of the key factors influencing the success of neural forecasting is the training and optimisation of the neural prototype. The parameters for optimisation are categorised in the following groups:

(i) **Configuration.**

The physical size of a fully connected two-layer Backpropagation neural network is controlled by the,

(a) *number of input nodes* and,

(b) *number of hidden nodes.*

(ii) Learning variables.

The speed at which a network is trained and the accuracy of its forecasts are affected by the,

(a) *momentum.*
It represents the value of μ in equation (3.4).

(b) *learning rate.*
It represents the value of η in equation (3.4).

(c) *tolerance.*
It represents the average distance measure (see below) within which a training network is expected to converge.

(d) *distance measure.*
It describes the method to measure the distance between the data fit against the target value. There is a choice of the least mean square error or the maximum error measurement.

(e) *transition function.*
It describes the function to squash the outputs of the activation function. There is a choice of a sigmoid or tangent output characteristic.

(f) *data scaling.*
It describes the method to scale the raw data. There is a choice of applying the logarithm of the data or the absolute values in the linear method. If the latter is used the absolute value is scaled to between 0 and 1 by equation (3.1).

(iii) Control variables.

These parameters controlled the processing, and reading and writing of NFS outputs.

(a) *number of training cycles.*

(b) *proportion of the data set for training.*
It describes the ratio of the data set training. There is a choice of dividing the data set into, half, two-thirds, three-quarters, or ALL for training. The remainder of each of the ratio, except the last, are used for testing.

(c) *number of input factors.*
It describes the number of different types of raw data used for the formation of input patterns. A value of, 1 is time-series forecasting, 2 is single-index econometric forecasting, and greater than 2 is multi-index econometric forecasting.

(d) *frequency of saving a snapshot of network.*

It describes the interval between which a snapshot of the training network is recorded. The duration is measured by the number of training cycles. The details of the rècording are discussed in the section on Data Recording .

(e) *frequency of display of the status of the training.*

It describes the interval between displaying the current status of the training network. The information is a duplicate of that recorded for the item on *a history of network errors* which is discussed in the section on Data Recording .

3.3.3 Input Patterns

An important routine in neural forecasting simulations is the formation and loading of input patterns to the network. As the **NFS** supports both time-series and econometric forecasting, a user's preference is expressed through the number of different types of raw data used for the formation of input patterns. An extension to the intention is an expression to include a simple transfer function [Jenkins87] to each of the inputs to the nodes in the network. The specifications of the following parameters apply to each of the nodes in a network.

(a) *type of index.*

It identifies the column number in the spreadsheet which contains the raw data for the formation of a pattern.

(b) *time lag or time advance.*

It describes the number of time-slices of data to lag behind or to advance with reference to the date of the time slice of data for the target node. This parameter is sometimes necessary to account for the time differences between the release of the figure and the time taken for the figure to be effective. An indicator like "the volume traded per day" correlates but lags behind other indicators by a day due to the way the data is recorded and released.

(c) *data steps.*

It describes the number of time-slices of data to omit in the formation of a continuous series of training input patterns. This reduces the number of patterns used in simulations but preserves the underlying trend of a large train data set.

(d) *operator.*

It describes any one of the four arithmetic operators to support the weighting.

(e) *weighting.*

It describes the weighting to be attached to the input data to express the implicit impact a figure might project on the other input figures in a pattern.

3.3.4 Network Re-construction

The re-construction of a previously configured network involves the creation of the previous execution environment of a neural prototype which includes the internal representations of the network and all the parameters used. It is a convenience to avoid repeating sections of a training session that has been carried out in a previous session. Alternatively, it is a necessity for forward simulations. This facility is dependent on the latest record of the log file for *a snapshot of the training network* which is discussed in the section on Data Recording below. The circumstances for applying the facility are:

(i) **Continuation.**

This recovers a trained network and continues the routines in historical simulation to meet with a new target for training cycles.

(ii) **Recovery.**

This recovers a partially trained network and continues the routines in historical simulation to achieve an existing target for training cycles.

(iii) **Application.**

This recovers a trained network to perform a one-step forecast in forward simulation.

3.3.5 Data Recording

It has been established that a facility to reconstruct a network is vital for the forecasting routines in either historical or forward simulations. It is, therefore, essential to improvise a facility to store the necessary information and parameter values in log files. In addition, this facility is useful for recording experimental data for analysis at the end of the simulation.

The following recording categories apply during historical simulations:

(i) **a snapshot of the training network.**

The file records two categories of information: house-keeping data, and the internal representation of the training and trained network. The first category covers the names of the log files discussed in this section and those parameters in the section on Parameter Optimisation. The data in the second category include the,

(a) *states* of all the nodes in the network.
(b) *weights* of all the connections in the network.
(c) *errors* propagated to each of the weight connections.
(d) *last weight changes* used to update the connections in the network.

The snapshot of the training network and the trained network are kept in separate files so that they can be used for different purposes in historical or forward simulations. The recording is controlled by the *frequency of saving a snapshot of the network* parameter which is described in the section on Parameter Optimisation above.

(ii) **a history of the network errors.**

A record of the network errors allows the landscape to be analysed for curve-fitting. Like the above, a record is made at regular intervals and it is controlled by the same parameter. The information recorded are the,

(a) *training cycle.*

(b) *failures.*

It is the ratio of the total number of patterns that has failed to converge below the tolerance value and the total number of patterns being trained.

(c) *average network error.*

It is the ratio of the total error differences between the network output and the target, and the total number of patterns being trained.

(d) *maximum network error.*

The maximum of the error differences between the network output and the target.

(iii) **performance statistics after testing a trained network.**

The forecasting potential of a neural prototype being researched is based on performance statistics calculated for the size of the test data set. The statistics are the,

(a) *pattern number.*

(b) *tracking Index value, IndexT.*

(c) *forecast Index value, IndexF.*

(d) *forecast error margin, IndexE.*

It is based on, *IndexE = IndexF - IndexT.* (3.7)

(e) *correlation.*

The correlation measures the accuracy in forecasting the direction of movement of the tracking Index value, *IndexT.* The direction of movement from time *t* to *(t+1)* is defined by the rules below:

IF *IndexT(t+1)* > *IndexT(t)* THEN direction is UP

IF *IndexT(t+1)* < *IndexT(t)* THEN direction is DOWN

IF *IndexT(t+1)* = *IndexT(t)* THEN direction is EQUAL

Figure 3.1 Definition Of The Direction Of Movement

On identifying the movement of *IndexT*, the correlation of the forecast $IndexF_{(t+1)}$ with $IndexT_{(t+1)}$ is determined. Three methods of measurement have been investigated.

The first, VCorr compares the value of $IndexF_{(t+1)}$ with the value of $IndexT_{(t)}$. It is defined by the following rules:

IF the direction is UP **AND** *IndexF(t+1) > IndexT(t)* **THEN** TRUE
IF the direction is DOWN **AND** *IndexF(t+1) < IndexT(t)* **THEN** TRUE
IF the direction is EQUAL **AND** *IndexF(t+1) = IndexT(t)* **THEN** TRUE

Figure 3.2a Correlation Of Forecast By VCorr Definition

The second, DCorr compares the direction of *IndexF(t)* to $IndexF_{(t+1)}$ with the direction of $IndexT_{(t)}$ to $IndexT_{(t+1)}$. Its definition is described in Figure 3.2b.

IF the direction is UP **AND** *IndexF(t+1) > IndexF(t)* **THEN** TRUE
IF the direction is DOWN **AND** *IndexF(t+1) < IndexF(t)* **THEN** TRUE
IF the direction is EQUAL **AND** *IndexF(t+1) = IndexF(t)* **THEN** TRUE

Figure 3.2b Correlation Of Forecast By DCorr Definition

The third, VDCorr is a combination of the definitions for VCorr and DCorr taking into consideration the value and direction of movement of $IndexF_{(t+1)}$. It is defined as:

```
IF the direction is UP    AND IndexF(t+1) > IndexT(t)
                          AND IndexF(t+1) > IndexF(t) THEN TRUE
IF the direction is DOWN  AND IndexF(t+1) < IndexT(t)
                          AND IndexF(t+1) < IndexF(t) THEN TRUE
IF the direction is EQUAL AND IndexF(t+1) = IndexT(t)
                          AND IndexF(t+1) = IndexF(t) THEN TRUE
```

Figure 3.2c Correlation Of Forecast By VDCorr Definition

(f) *number of correct forecasts.*

It measures the current total number of forecasts that have accurately predicted the direction of movement of the Index. The number is the accrued total of the number of times the correlation rules have been satisfied as the test patterns are tested.

(g) *unrealisable weighted return.*

It is a measure to indicate how well the forecasting model has navigated the major data movements of the test data set. The weighting is the accrued total of the number of unrealisable Index points made as the test patterns are tested. Index points are credited whenever *IndexF* correlates with *IndexT*, they are debited if they do not correlate. The unrealisable points (unrealisable weighted return) are calculated according to equations (3.8) and (3.9).

$$\textit{Unrealisable points} = IndexT_{(t+1)} - IndexT_{(t)} \qquad (3.8)$$

$$\textit{Unrealisable points} = -1\ (IndexT_{(t+1)} - IndexT_{(t)}) \qquad (3.9)$$

They are applied for each of the three methods of correlation measurements. The rules for application are:

IF the correlation is TRUE **AND** the direction is UP **THEN** equation (3.8)
IF the correlation is TRUE **AND** the direction is DOWN **THEN** equation (3.9)
IF the correlation is TRUE **AND** the direction is EQUAL **THEN** equation (3.8)

IF the correlation is FALSE **AND** the direction is UP **THEN** equation (3.9)
IF the correlation is FALSE **AND** the direction is DOWN **THEN** equation (3.8)
IF the correlation is FALSE **AND** the direction is EQUAL **THEN** equation (3.8)

Figure 3.3 Definition Of Unrealisable Weighted Return

When all the test data have been tested, the final total is expressed as a percentage of the maximum total number of Index points that could be gained. The percentage gives an indication as to how well the trained network has forecast, or rather, tracked the major movements of the Index.

In forward simulations, a record is made of:

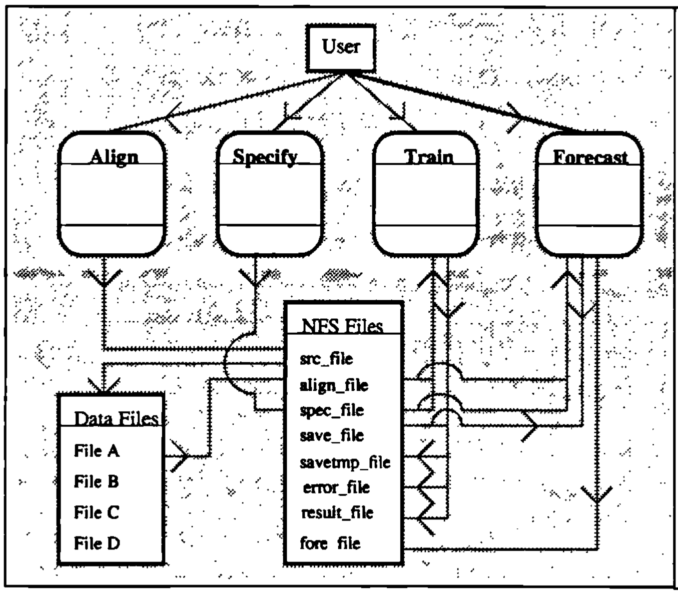
(iv) a history of the forecasts.

The absolute value of a *forecast* is appended to a log file for evaluation at a later date.

The data for these files have to be exported to an external source like a Lotus spreadsheet and plotted for analysis. As such, the data in the log files have formats which allow them to be converted for access by external software packages.

3.4 Process Design

The technical guidelines addressed are translated to system processes using a top-down design approach. They correspond to the top-level **NFS** functions identified as necessary for forecasting research. Within these processes there are implementations of the other research facilities and technical requirements which have been identified. The **NFS** framework comprises the - **Align, Specify, Train** and **Forecast** - Processes. Each of these processes operate independently but they communicate indirectly via parameters that are contained in system files (anchor files) specific to them. Each of these **NFS** anchor files have specific input/output functions to read, write, and concatenate data that are private to each Process. As a result,·a file may grant specific input/output function(s) for a particular Process but not to another. The set of private input/output functions is dependent on the purpose of the Process accessing the file contents. A summary of the anchor files and their respective input/output functions relating to each of the Processes are shown in Figure 3.4.



Process	Anchor File	I/O	Editable
Align	src_file	r	Yes
	align_file	w	Yes
Specify	spec_file	w	Yes
Train	spec_file	r,w	No
	align_file	r	No
	save_file	w	No
	savetmp_file*	r,w	No
	result_file	w+	No
	error_file	w+	No
Forecast	spec_file	r	No
	align_file	r	Yes
	save_file	r,w	No
	fore_file	w+	No

+append *always named after save_file

Figure 3.4 The NFS Processes With Their Anchor Files And I/O Functions

3.4.1 The Align Process

The **Align** Process standardises all the raw data files which may have different - daily, monthly or quarterly - data recording intervals to a standard daily recording time, and writes a subset of the data, having a common time frame, to a spreadsheet in the anchor file, align_file. The data are retrieved from all the files listed in the anchor file, src_file and aligned by the date of recording. The align_file is subsequently accessed by the **Train** Process and the **Forecast** Process for historical and forward simulations respectively.

The Process,

(i) establishes the names of the data files from src_file for merging.

(ii) identifies the interval for recording the data in each of these named data files.

(iii) establishes a common start and end dates from all the data files to resolve the possibility of non-uniform data recording times and data volumes.

(iv) standardises all the data to a daily recording interval to resolve the possibility of non-uniform data recording times.

(v) merging the set of data to align_file by their date of recording.

The Process is described in the structure diagram, Figure 3.5 in Appendix A.

3.4.2 The Specify Process

The **Specify** Process provides a medium for parameter specifications and tuning, which is necessary prior to training and testing a neural prototype as for historical simulations. These parameters are written in a table in the anchor file, spec_file. The parameter values are accessed by the **Train** Process and the **Forecast** Process to construct the neural prototype and the input patterns for simulation.

The Process writes the following in spec_file:

(i) the default names of the three log files to record, a snapshot of the training network, a history of the network errors, and the performance statistics at the end of the training.

(ii) the default values of the neural parameters and the control variables.

(iii) the specification language for specifying the formation of input patterns to the network.

This Process is described by the structure diagram, Figure 3.6 in Appendix A.

3.4.3 The Train Process

The **Train** Process performs historical simulation optimising a neural model by learning a portion of the data set in a time-series and testing it on the remainder portion. The optimised neural model and the parameters associated to it are recorded in save_file and savetmp_file. The first file allows the **Forecast** Process to re-build a trained network to forecast for testing with live data. The second allows the **Train** Process to re-build a training network so that training can be continued or it is to recover and continue training a partially trained which was abnormally terminated. The simulation environment is based on the user specifications in spec_file and the data set in align_file.

The procedure of the Process,

(i) reads the log file names, the neural and control parameter values.

(ii) re-builds a neural network or configures a new network according to specifications.

(iii) parses the pattern specification language to establish the method of formation of input patterns.

(iv) scales the raw input data down to between 0 and 1 and scales them up again.

(v) loads input patterns from the train or the test data set suitable for either a time-series or econometric forecasting.

(vi) performs the neural learning procedure to train the network to extrapolate the time-series in the train data set.

(vii) performs the neural procedures to test the trained network on the test data set.

(viii) calculates performance statistics after testing the trained network.

(ix) records the latest snapshot of the training neural model and the current network errors, the trained neural network, and the performance statistics in log files.

This Process is documented in Figure 3.7 in Appendix A.

3.4.4 The Forecast Process

The **Forecast** Process performs forward simulation to deliver a one-step forecast for one data recording period ahead. The forecast, a future value (without a tracking value for comparison), is an extrapolation of a time-series. Like the **Train** Process, it is dependent on the specifications in spec_file and the data in align_file. Unlike the **Train** Process, it re-builds the neural network using the information in save_file rather than savetmp_file and it does not carry out the procedures for training. Instead, it only carries out the procedures for testing but only on the last input pattern constructed.

The procedure of this Process is similar to the **Train** Process. It,

(i) re-constructs the optimised neural network with the trained network parameters.

(ii) scales the input values of the latest input pattern down to between 0 and 1 and scales the forecast value up again.

(iii) forms and loads the latest input pattern according to specifications for either a time-series or econometric forecasting.

(iv) performs the neural test procedures once, to extrapolate the time-series.

(v) records the latest snapshot of the optimised neural model to the log file.

(vi) records the absolute value of the forecast in a log file.

A structure diagram which describes the Process, Figure 3.8, is included in Appendix A.

3.5 Summary

This chapter reports the specifications and design for the development of the customised Neural Forecasting System (**NFS**). The **NFS** is for investigating the objectives set for this thesis and to enable forecasting research in other areas of investments. The description of the system specifications covers both the general and technical requirements. Following this, the description of the system design describes the processes mapping these specifications.

The procedure for system specifications was initially to establish requirements that are related to the system's environment. It established that the 'C' language be used for implementation and the system mounted on a Unix operating platform. In addition, the **NFS** is: ***complete*** with essential functions and facilities for, the alignment of data, historical and forward simulations to support both time-series and econometric forecasting; ***compatible*** with other software packages for reading and writing data; and ***easy-to-use*** adopting menu interfaces, simple terminology, and clear messages. The user has long-term support from a user manual.

The technical requirements covered the precise specifications of the Backpropagation neural model and the system facilities needed to facilitate neural forecasting research. The facilities are: to standardise and merge a pool of raw data which might have non-uniform data recording times and possibly compounded by non-uniform data volumes; to facilitate the specification of neural and control parameters, the formation and loading of input patterns to the network, file names to log performance histories and statistics for future evaluation; and finally, to re-construct a network for the continuation or recovery of training, or to use a trained network for forward simulation.

The system design translated the technical specifications as four system processes. Each of the - **Align, Specify, Train** and **Forecast** - Process represents an independent system function which has been established as essential to the **NFS**. They are, therefore, accessible as top-level **NFS** functions. The **Align** Process standardises the raw data to a uniform a daily recording time and merges those within a common time frame to a spreadsheet. The **Specify** Process provides a specification table - in a file - for the specification of the names of log files, the neural parameters of a neural forecasting prototype, the control parameters and the method of forming and loading of input patterns. The **Train** Process allows training followed by testing of a specified forecasting prototype using historical data which have been aligned earlier. Finally, the **Forecast** Process carries out a one-step forecast using a re-constructed trained neural network.

Chapter 4

Implementation Of Neural Forecasting System

*This chapter describes the translation of the system processes designed to five 'C' - **Menu, Align, I/O, Load and Backpropagation** - Modules. First, it gives an overview of the system and explains the functions of these Modules. Following this, it explains the techniques used to implement system facilities for: the alignment of non-uniform raw data; the specification and tuning of parameters; the specification of input pattern formation and loading and; the re-construction of a neural network for further processing.*

4.1 System Overview

The development of the Neural Forecasting System (NFS) followed from the conceptual model comprising of the **Align**, **Specify**, **Train**, and **Forecast** Processes. The implementation of the NFS was to translate these processes to the ***Menu***, ***Align***, ***I/O***, ***Load***, and ***Backpropagation*** 'C' Modules.

In mapping the functions of these Processes, the task-oriented Modules are mutually reinforcing whilst contributing their specific roles. In the program design, these Modules communicate amongst themselves under the control of one or more control variables. An overview of the system is illustrated by the entity diagram in Figure 4.1.

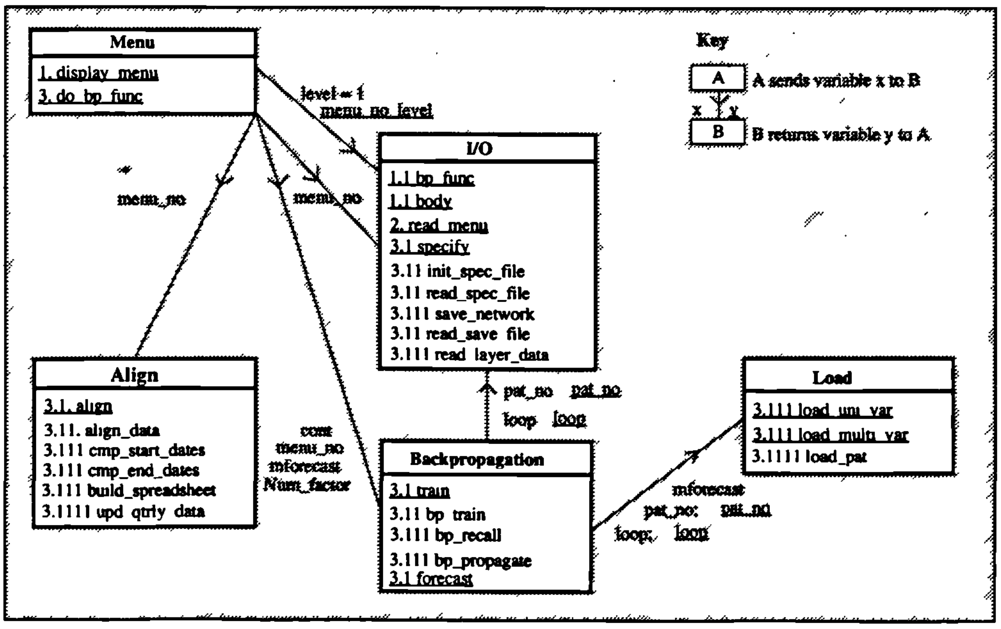


Figure 4.1 Program Modules With Procedure Calls And Control Variables

4.1.1 The *Menu* Module

The ***Menu*** Module provides a medium for the user to interact with the **NFS** and schedules communications amongst the other Modules.

The Module is invoked from the 'C' main routine soon after the **NFS** has been started by the `bp` command. After the execution environment has been initialised, the display_menu routine receives a call with values for `level` and `mforecast` passed. The value of `level` determines one of two types of menus for display: a value of `1` triggers bp_func for a menu of forecasting functions, and a value of `2` triggers body for a menu of all parameters and their values specified in `spec.dat`. These displays are made with calls to the ***I/O*** Module.

The consequence of a menu display by display_menu is the scheduling of communications with other Modules: the ***Align*** Module following a selection of `1` to call for the alignment of data files; the ***I/O*** Module following a selection of `2` to call for the creation of `spec.dat` for parameter and input pattern specifications; the ***Backpropagation*** Module following selections `3` and `4` to call for historical and forward simulations respectively. These direct links also result in indirect communications with other Modules to call on other supporting sub-routines.

The body routine is invoked to allow changes to the parameter values set earlier in `spec.dat`. This allows the user to confirm the specifications of the parameters prior to a call to the train subroutine to perform a historical simulation.

On completion of any of the forecasting requests, all the parameters used in the execution environment are restored to the same state as it was in main. The Module remains in control of the **NFS** until the menu option to terminate is explicitly requested.

4.1.2 The *I/O* Module

The ***I/O*** Module provides read and write routines to service the operations in the ***Menu*** and ***Backpropagation*** Modules.

The Module is first invoked by the ***Menu*** Module to display menus on the screen. Subsequently, it is triggered to display for the user: information, such as the status of the training network; error messages; experimental results, such as the performance statistics after testing a trained neural network.

The Module consists of a group of routines to write: data, such as the parameters for specification in `spec.dat` and raw data in `align.dat`; information, such as the internal representations of the neural network to `save.dat` and `save.dattmp`, training network errors to `error.dat`; experimental data to `graph.dat`; forecasts to `fore.dat`.

Similarly, there are complementary routines to read: on-line inputs, such as menu selections; data, such as data file names from `src.dat`, raw data from `align.dat`, and specifications from `spec.dat`; information saved in `save.dat` and `save.dattmp`.

As the Module may not have a direct link with the ***Menu*** Module, control is returned to the calling Module whenever the called routine terminates.

4.1.3 The *Align* Module

The ***Align*** Module standardises data files with non-uniform recording times and possibly non-uniform data volumes to a standard daily recording, and merges the set with a common time frame to `align.dat`. This data preparation procedure is obligatory and it is carried out whenever the data set for simulations changes.

The Module is invoked directly by the ***Menu*** Module by a call to align_data. It consists of three subroutines to standardise data. They are to establish: all the data file names, the start dates and end dates for each file, and the common start date and end date. The procedure for merging consists of subroutines to: unfold non-daily data to daily data and writes to the spreadsheet in `align.dat` through the ***I/O*** Module.

On completion, the execution environment unfolds: the dynamic values of the control variables are re-instated to their static values and control returns to the ***Menu*** Module.

4.1.4 The *Load* Module

The ***Load*** Module constructs input patterns based on the specification for time-series or econometric forecasting models and loads them to the neural network. The specifications are parsed from the pattern specification language in `spec.dat`, and the formation could use one or more time-series data from `align.dat`, as specified.

The Module is always invoked by the ***Backpropagation*** Module to load_uni_var or load_multi_var routines for time-series and econometric forecasting respectively. The value of `Num_factors` which is the variable value of the control parameter, `Number of input factors`, differentiates these two forecasting models. In addition, the value of `mforecast` differentiates between historical and forward simulation options.

The Module loads a pattern to a network by the load_pat routine. The method is dependent on the type of forecasting and the simulation option used. The subroutine relies on the value of `Num_factors` to load the pattern as for an extrapolative forecasting model, a single-index or multi-index causal forecasting model, and `mforecast` to load only the last pattern formed for forward simulation.

The execution environment unfolds back to the ***Backpropagation*** Module after loading.

4.1.5 The *Backpropagation* Module

The ***Backpropagation*** Module supports routines for Backpropagation learning and utilities for forecasting: to train and test a neural prototype using historical data and to make a one-step forecast for live test.

The Module is invoked at the top-most level by menu selections for the two forecasting functions. The pre-requisites to making a selection are an up-to-date specifications in spec_file (defaulted to `spec.dat`) and a suitable data set in align_file (defaulted to `align.dat`). Following a selection, the main routines corresponding to these functions, the train and forecast routines, are triggered.

A selection for historical simulation invokes the train routine. The procedures in this routine initially calls the bp_train subroutine which executes bp_recall and bp_propagate repeatedly until all the train data set have been cycled through a specified number of times. Thereafter, the bp_recall routine is invoked again to test the data of the test data set.

In the case of a selection for forward simulation, the forecast routine, after re-constructing the trained network and re-instating the execution environment, interfaces with the ***Load*** Module and calls bp_recall once to deliver a forecast.

For either routines, the ***Load*** Module is invoked with the value of `mforecast` passed to differentiate them. The pattern number, `pat_no`, and the training cycle number, `loop`, are used to index an input pattern (the last one in the case of the forecast routine) for loading to the neural network. The ***I/O*** Module provides the supporting routines to read `spec.dat` for the re-construction of a partially-trained network in savetmp_file (defaulted to `save.dattmp`) or of a trained network in save_file (defaulted to `save.dat`), reads data from `align.dat` for the ***Load*** Module, and logs information and experimental data in log files such as error_file, graph_file, fore_file.

After the execution of the routines for either forecasting functions, the execution environment and control variables are restored to the state at the ***Menu*** Module.

4.2 Data Alignment

The integrity of the raw data for simulation is crucial to the outcome of any methods of forecasting. It has been stressed that simulations on the NFS could use data set comprising daily, monthly and quarterly data releases. The objectives to uphold data integrity are standardising the heterogeneous data recording in different data files to a homogenous daily recording across all the files used, and merging the whole set of data in a new spreadsheet.

The following sub-sections explain the techniques used to resolve data non-uniformity.

4.2.1 Common Time Frame

The first procedure to resolve data non-uniformity establishes a common time frame of all the data files listed in `src.dat`. The elements of a common time frame are a common start date and end date which are calculated from these data files. These dates correspond to the latest start date and the earliest end date of the data files. They are the default dates for the aligned data in the spreadsheet to be created.

```
RuleStart1.1
     IF        all the data files listed in src_file are daily data files
     THEN      the common Start Date is the latest start date of all the data files
RuleStart1.2
     IF        there are monthly and daily data files listed in src_file
     THEN      the common Start Date is the latest start date of the daily data files
               which coincides with the expanded start date of the monthly data files
RuleStart1.3
     IF        there are quarterly and daily data files listed in src_file
     THEN      the common Start Date is the latest start date of the daily data files
               which coincides with the expanded start date of the quarterly data files
RuleStart1.4
     IF        there are quarterly, monthly and daily data files listed in src_file
     THEN      the common Start Date is the latest start date of the daily data files
               which coincides with the expanded start date of the monthly data files
               the later also lies within the expanded quarter of the quarterly data files
```

Figure 4.2 Definition Of Default Start Date

In the calculation of these dates, precedence is given to the daily data files as there has to be at least one data file listed with a daily recording time. The rules in Figure 4.2 define the intersection of all the start dates from all the data files listed to obtain the latest start date.

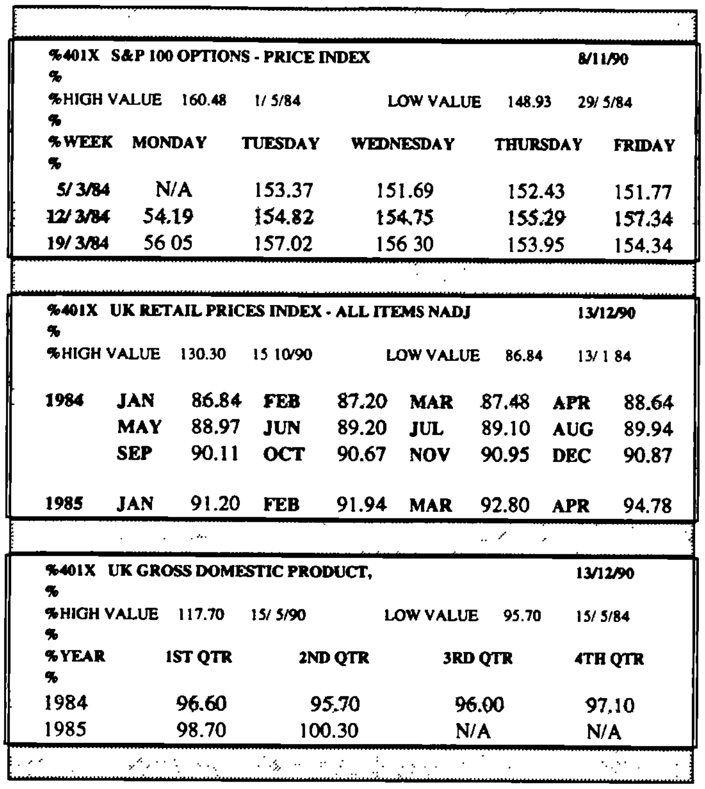
```
RuleEnd1.1
     IF        all the data files listed in src_file are daily data files
     THEN      the common End Date is the earliest End Date of all the data files
RuleEnd1.2
     IF        there are monthly and daily data files listed in src_file
     THEN      the common End Date is the earliest end date of the daily data files
               which coincides with the expanded end date of the monthly data files
RuleEnd1.3
     IF        there are quarterly and daily data files listed in src_file
     THEN      the common End Date is the earliest end date of the daily data files
               which coincides with the expanded end date of the quarterly data files
RuleEnd1.4
     IF        there are quarterly, monthly and daily data files listed in src_file
     THEN      the common End Date is the earliest end date of the daily data files
               which coincides with the expanded end date of the monthly data files
               the later also lies within the expanded quarter of the quarterly data files
```

Figure 4.3 Definition Of Default End Date

Similarly, the rules used to calculate the default end date for the spreadsheet is defined in Figure 4.3. They represent a set of criterion in which to intersect the set of end dates to obtain the earliest end date.

4.2.2 Common Recording Time

The second procedure implements a common recording time which is a daily recording. The procedure identifies the recording times from each of the data files and expands those data with non-daily data releases to daily "constants".



```
%401X  S&P 100 OPTIONS - PRICE INDEX                              8/11/90
%
%HIGH VALUE   160.48   1/ 5/84        LOW VALUE   148.93   29/ 5/84
%
%WEEK   MONDAY    TUESDAY    WEDNESDAY    THURSDAY    FRIDAY
%
 5/ 3/84  N/A      153.37     151.69       152.43      151.77
12/ 3/84  54.19    154.82     154.75       155.29      157.34
19/ 3/84  56 05    157.02     156 30       153.95      154.34
```

```
%401X  UK RETAIL PRICES INDEX - ALL ITEMS NADJ                    13/12/90
%
%HIGH VALUE   130.30   15 10/90       LOW VALUE   86.84    13/ 1 84

1984   JAN   86.84   FEB   87.20   MAR   87.48   APR   88.64
       MAY   88.97   JUN   89.20   JUL   89.10   AUG   89.94
       SEP   90.11   OCT   90.67   NOV   90.95   DEC   90.87

1985   JAN   91.20   FEB   91.94   MAR   92.80   APR   94.78
```

```
%401X  UK GROSS DOMESTIC PRODUCT,                                 13/12/90
%
%HIGH VALUE   117.70   15/ 5/90       LOW VALUE   95.70    15/ 5/84
%
%YEAR       1ST QTR      2ND QTR      3RD QTR      4TH QTR
%
1984        96.60        95.70        96.00        97.10
1985        98.70        100.30       N/A          N/A
```

Figure 4.4 File Layouts Of Recording Times

To identify the temporal irregularity of the data files and to maintain an easy-to-use spirit, the identification procedure exploits the unique Datastream file formats for each of the three data recording times. The file formats for the three types of "cleaned" files · are shown in Figure 4.4. The recording time of a file is detected by scanning the file for the different types of format. On its establishment, its start date and end date are read. When all the files have been scanned and read, the information is applied to the rule sets in Figure 4.2 and Figure 4.3 to obtain a common time frame.

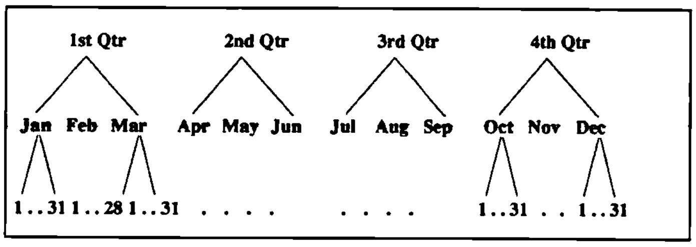


Figure 4.5 Calendar For Unfolding Data

Following the identification of the recording times of the data files and the calculation of the default start and end dates for the spreadsheet, the procedure for unfolding the non-daily data refers to the financial calendar in Figure 4.5. The days bounded by the date for a full month or a full quarter are filled with daily "constants", to fill in for that month's or quarter's data.

4.2.2 Merging Data

The third procedure aligns the heterogeneous set of data with homogenous recording time by their date stamps. The output in `align.dat` has a format such that: all the data in a row are aligned by date, all the data in a column are from the same data file and the number of columns is equal to and indexed by the order of the file listings in `src.dat`.

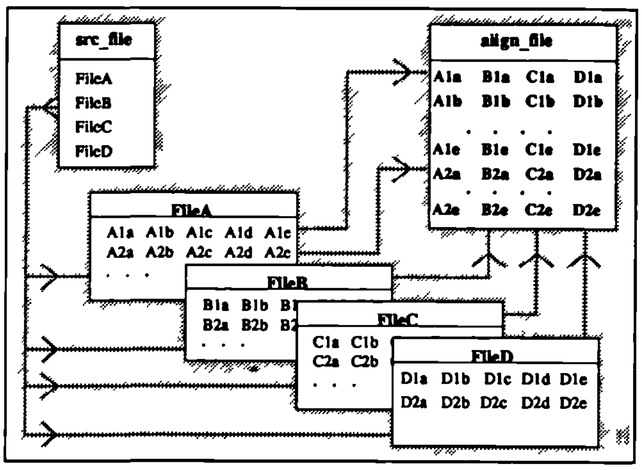


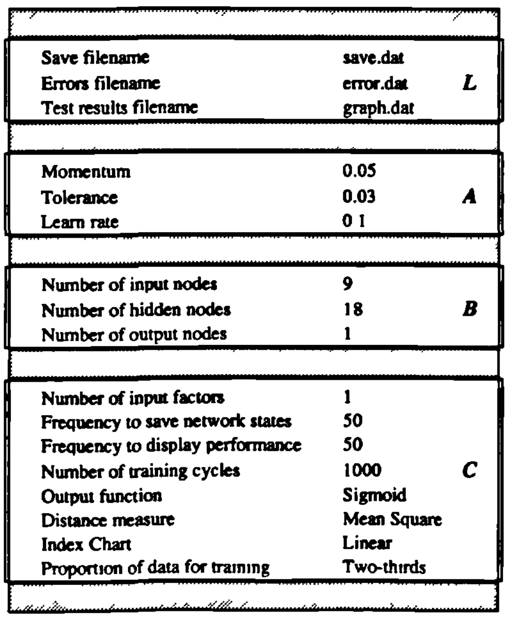
Figure 4.6 Merging Data To The Spreadsheet

The process of building the spreadsheet is a recursive operation which involves the reading and writing of batches of data from the default start date to the default end date. For each operation, it reads a weekly batch of daily data from each of the data files, align the data in the order shown in Figure 4.6, and writes to the spreadsheet created. For each batch of weekly daily data, there are non-daily "constants" - if non-daily data files are used - which are the result of unfolding those non-daily data files to smooth the irregular data volumes. When the terminating condition of the building process is met, any data remaining in any of the data files are ignored.

4.3 Parameter Specifications

The research and optimisation of a neural forecasting prototype is made simple by the parameter specification table which has been designed to be easily accessed and modified to requirements. The table is contained in the **NFS** anchor file, `spec.dat`.

The first facility within the table is for the specification of log file names for recording: a snapshot of the training and trained neural prototype, a history of the training network errors, and the performance statistics of testing a trained neural prototype. The corresponding log files save_file, error_file and result_file default to `save.dat`, `error.dat` and `graph.dat` respectively. The names are written in the section labelled *L* in Figure 4.7. These default names apply until they are changed by the user.



Save filename	save.dat	
Errors filename	error.dat	*L*
Test results filename	graph.dat	
Momentum	0.05	
Tolerance	0.03	*A*
Learn rate	0 1	
Number of input nodes	9	
Number of hidden nodes	18	*B*
Number of output nodes	1	
Number of input factors	1	
Frequency to save network states	50	
Frequency to display performance	50	
Number of training cycles	1000	*C*
Output function	Sigmoid	
Distance measure	Mean Square	
Index Chart	Linear	
Proportion of data for training	Two-thirds	

Figure 4.7 Contents Of `spec.dat`

The second facility is for the specification of the parameters of the neural prototype. There are three separate parameter lists, labelled as sections *A*, *B*, and *C* in Figure 4.7. Each of the sections correspond to the list discussed for, Configuration, Learning variables and Control variables in the Parameter Optimisation section in Chapter 3. Like the log file names, all the parameters are initialised to a default value and they remain in use until they are modified.

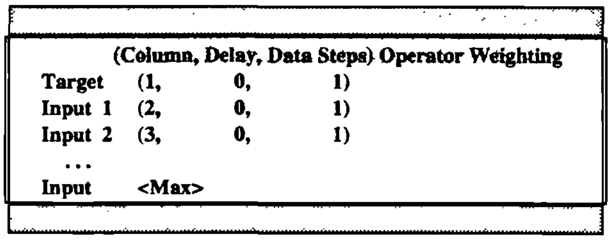
The specification of the parameters can be made on a new `spec.dat` which is created with the parameters assigned to default values. Alternatively, an existing file can be accessed and revised.

4.4 Pattern Specification Language

The requirement to support both time-series and econometric forecasting models in a system which has to be easy-to-use necessitates a facility to specify the formation of input patterns and loading to the neural network. To enable such a facility to operate, there are two elements to specify: the specification of the intention to use one of the two possible forecasting methods and the specification for the input pattern itself. As such, the pattern specification language is designed specifically for neural forecasting.

The specification of the forecasting model makes use of the variable value of the parameter, `Number of input factors`: a value of `1` implies time-series forecasting using one type of raw data for both the target and the input nodes; a value of `2` implies single-index econometric forecasting using one type of raw data for the target node and another for the input nodes; a value greater than `2` implies multi-index forecasting using one type of raw data for the target node and a different one for each of the input nodes. The ratio of the number of raw data for the target node to those for the input nodes is referred to as a *Self-to-Self*, *One-to-One* or *One-to-Many* relationship respectively. They are crucial to the control of the language parser.

The specification for the formation of input patterns is made through the pattern specification language. As it is shown in Figure 4.8, the table is initialised and appended at the bottom of the parameter specification table in `spec.dat`. The syntax of the language follows the specification of the parameters in the section on Input Pattern in Chapter 3. The default parameter values suggest that: the target node has data from the first column, the first input node the second column, the second from the third column of the spreadsheet; the data for each of the nodes do not have any time adjustments; only one time-slice of data is omitted between each successive pattern formation; weightings are not attached to the data.



```
         (Column, Delay, Data Steps) Operator Weighting
Target    (1,      0,      1)
Input 1   (2,      0,      1)
Input 2   (3,      0,      1)
  ...
Input     <Max>
```

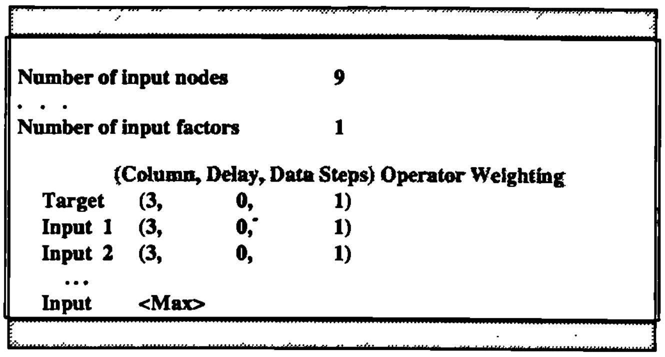
Figure 4.8 Pattern Specification Language

A top-down, left-to-right parser is used to translate the semantics of the language. The translation is dependent on the variable value, `Number of input factors` or more precisely, the interpretation to one of the three relationships of the ratio of raw data used.

The parser uses a 2-dimension control for both a *Self-to-Self* and *One-to-One* relationship, and a 3-dimension control for a *One-to-Many* relationship.

4.4.1 Time-Series And Single-Index Forecasting

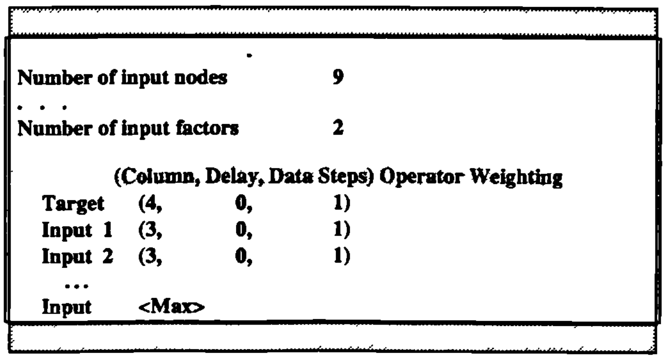
The parser applies the 2-dimension control when a *Self-to-Self* or a *One-to-One* relationship is established. A 2-dimensional control equates to parsing the pattern specification table from the target node up to the first input node and discards any specifications thereafter. This strategy exploits the fact that it is sufficient to apply the information parsed from the target node and the first input node, given that the other input nodes apply the same information from the first node.



```
Number of input nodes          9
. . .
Number of input factors        1

       (Column, Delay, Data Steps) Operator Weighting
  Target   (3,     0,      1)
  Input 1  (3,     0,      1)
  Input 2  (3,     0,      1)
  ...
  Input    <Max>
```

Figure 4.9 Time-Series Pattern Specification

The pattern specification in Figure 4.9 invokes the 2-dimension control through the parameter value of `Number of input factors`. After parsing, data from the third column of the spreadsheet is used to form input patterns for the time-series forecasting model and the specifications for the second input node is ignored. In a time-series pattern formation, the value of `Delay` is irrelevant, but subsequent pattern formations could depend on the value of `Data Steps`. If the time-series in the third column is $I^3_1, I^3_2, I^3_3, , , , , , I^3_{tot\_num}$, then the first pattern in this case is made up of $I^3_1, I^3_2, I^3_3, I^3_4, I^3_5, I^3_6, I^3_7, I^3_8, I^3_9$ with I^3_{10} assigned to the target node. As the default value of `Data Steps` is `1`, the next pattern is made up of, $I^3_2, I^3_3, I^3_4, , , , , I^3_9, I^3_{10}$ with I^3_{11} for the target node. If the value of `Data Steps` is `4` instead, the second next pattern is $I^3_5, I^3_6, I^3_7, , , , , I^3_{12}, I^3_{13}$ with I^3_{14} for the target node. Four time-slices of data are omitted in the formation of the second pattern and for all succeeding patterns until the time-series data is depleted.



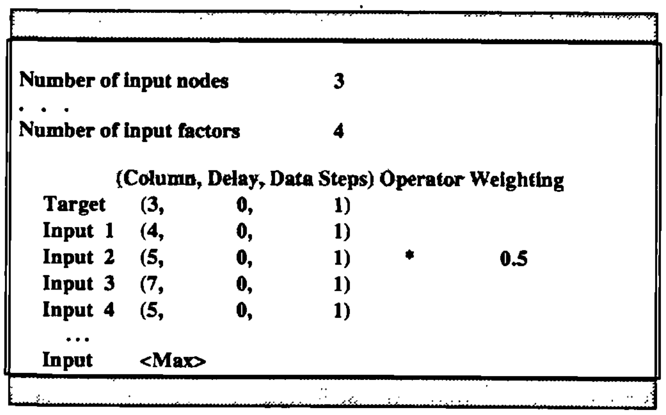
```
Number of input nodes          9
. . .
Number of input factors        2

       (Column, Delay, Data Steps) Operator Weighting
  Target   (4,     0,      1)
  Input 1  (3,     0,      1)
  Input 2  (3,     0,      1)
  ...
  Input    <Max>
```

Figure 4.10 Single-Index Pattern Specification

The specification for a single-index forecasting model is shown in Figure 4.10. The language used is parsed in the same manner as time-series forecasting. The pattern formation is also similar, apart from the source of the data for the target node. In the example, it is retrieved from the fourth column of the spreadsheet instead of the third column, as it is in the previous example.

4.4.2 Multi-Index Forecasting

The parser applies the 3-dimension control when a *One-to-Many* relationship is established. The parser reads the pattern specification table from the target node down to the input node that is equal in dimension to the value of `Number of input nodes` . The rest of the specifications after this are discarded. The objective in parsing the specification for a multi-index forecasting is to obtain information exclusive to each of the nodes in the network.



```
Number of input nodes          3
. . .
Number of input factors        4

        (Column, Delay, Data Steps) Operator Weighting
  Target  (3,     0,     1)
  Input 1 (4,     0,     1)
  Input 2 (5,     0,     1)      *        0.5
  Input 3 (7,     0,     1)
  Input 4 (5,     0,     1)
  ...
  Input   <Max>
```

Figure 4.11 Multi-Index Pattern Specification

The example in Figure 4.11 is a specification for multi-index forecasting. The target node has data from the third column, the first, second, and third input nodes have data from the fourth, fifth and seventh columns respectively. The data used for all the nodes are in their original form, that is, they are not temporally re-aligned and there is no omission of time-slices of data between the formation of each pattern. Apart from that, the data for the second input node is weighted to reflect its impact relative to the data for the other nodes. The first pattern is made up of the first time-slice of data from the three columns specified for the input nodes, $I^4{}_1, I^5{}_1, I^7{}_1$ with $I^3{}_1$ assigned to the target node. Thereafter, succeeding patterns are made up of succeeding time-slices of data from columns of input data. The time-slices are adjusted if the value of `Data Steps` is overridden.

4.5 Snapshot Facility

The process of undertaking historical simulations can, at some stage, require the use of a partially trained neural prototype. Similarly, a trained prototype for forward simulations. A checkpoint routine which records a snapshot of the training or trained network is used as a backup to support these simulation functions. The network re-construction facility restores the internal representations of the network and the parameter specification used for the network. These are recorded in log files, `save.dat` and `save.dattmp`.

Training Continuation

A network is restored for the continuation of training when a previously trained network is required to extend its training to a larger number of training cycles, with all the other neural parameter values unchanged. This facility is commonly used to economise on training and testing a neural prototype in historical simulations.

A preparation procedure is required to revise the values of the parameter controlling the number of training cycles and the new names of log files. It is common, although optional, to have new file names so that the new set of experimental data is kept separate for future evaluation. Following this, the contents of `save.dattmp` (or its equivalent name for `save.dat` in the previous version of `spec.dat`) is renamed to that specified in `spec.dat` as the **NFS** would be reading from the new file named in the new training session. It is also common to rename `error.dat` (or its equivalent name used in the previous version of `spec.dat`) so that the network errors calculated are appended to the history accumulated in a single file. The network reconstruction involving the recovery of all the internal network representations and parameter values from `save.dattmp` (or its equivalent name) is carried out after the user confirms that a continuation of a training session is required following a menu selection for historical simulation.

Training Recovery

A partially trained network is reconstructed when a training session has abnormally terminated and it is necessary to continue with the training. Unlike the continuation of the training session, the specifications in `spec.dat` remain intact, as the partially trained network has been trained for less than the maximum number of cycles specified. Unlike the above, a preparation procedure is not required. Simply, the same method to initiate the routine applies, and similarly, the contents of `save.dattmp` (or its equivalent name for `save.dat`) accessed.

Network Application

A trained network is recovered from the latest snapshot saved in `save.dat` (or its equivalent name used in `spec.dat`) for a one-step forecast. This routine makes the following three assumptions. First, the data in `align.dat` were previously used for training. Second, the training had used the same specifications in `spec.dat`, and finally, the trained network is recorded in `save.dat` (or its equivalent name used in `spec.dat`). This routine is implemented as a forecasting function that is initiated through a menu selection. Before this facility is used, the time-series in `align.dat` is updated with the latest set of data.

4.6 Summary

This chapter reports the implementation of the customised Neural Forecasting System (NFS). It follows the translation of the four - **Align, Specify, Train**, and **Forecast** - processes which have been designed to model the NFS to five - ***Menu***, ***I/O***, ***Align***, ***Load*** and ***Backpropagation*** - programming Modules.

The first section gives an overview of the program design, showing the relationships of the Modules, the main subroutines, and the control variables. The ***Menu*** Module provides menu interfaces for the user and schedules the interactions amongst the other Modules; the ***I/O*** Module handles all input-output functions between the Modules, files and the screen; the ***Align*** Module standardises all data files which might have non-uniform data recording times and volumes, and merges them to a new spreadsheet; the ***Load*** Module handles the formation of input patterns from one or more time-series data and loading them for the ***Backpropagation*** Module; and finally the ***Backpropagation*** Module provides the neural learning procedures and utilities for historical and forward simulations.

Following this, the second section explains the implementation of the research facilities. The first, the alignment of raw data files covers: the calculation of a common time frame by the intersection of start dates and end dates; the detection of daily, monthly and quarterly data files by their file formats and unfolding those non-daily dates and filling them with daily "constants"; and merging of the data bounded by the common time frame to the spreadsheet. The second, parameter specification table covers the facility to ease repetitive parameter modifications for network optimisations. The list of parameters were categorised under log file names, neural and control parameters. The third, input pattern specification covers the semantics of a pattern specification language to allow the specifications of pattern formations and loading for time-series, single-index and multi-index forecasting. Lastly, the snapshot facility covers the reconstruction of a partially trained network from `save.dattmp` or a trained network from `save.dat`. The former is to extend training of a trained network or to recover from an aborted training session in historical simulation. The latter is to apply the network to make a one-step forecast in forward simulation.

Chapter 5

Experimental Strategy

This chapter reports the strategy used to validate the data representation method in model sNx. Initially, it explains the notations used including the RTPI indicator, the method of extrapolation, and the choice of data sets. Thereafter, it describes the experiments - to select a suitable raw data form, optimisation of the network configuration and neural parameters to identify a suitable prediction accuracy measurement, and testing the optimised networks on time complex data sets - and evaluation criterion.

5.1 Neural Time-Series Analysis

A neural network learns by comparing characteristic input patterns, identifies and stores a fingerprint pattern that distinguishes a particular input characteristic in its memory [RumHin86, Lippma87]. The same network recognises a new input pattern by referencing the library of characteristic patterns that it has already learnt to identify in its memory.

A supervised Backpropagation neural model learns by comparing input patterns with desired characteristics in corresponding target patterns. Neural time-series forecasting extrapolates by learning continuous input patterns extracted from a historical time-series to project target values which are succeeding data of the input pattern series. A forecast for the future is made from its memory of past forecasts.

The investigations to test the feasibility of the new method of representing input data in model sN*x* compared to the tenable representation in model N, used experimental data obtained from different historical periods of the FTSE-100 Index. An explanation of the notations used and the choice of experimental data are given.

5.1.1 Notations

The time-series neural simulations have adopted the following formalisms to reference the Index.

(i) The structure of a sampled set of a time-series, X is,

$$X_{(t)}, X_{(t+t)}, X_{(t+2t)}, \ldots, X_{(t+(T_{D-1})t)} \qquad (5.0)$$

The series specifies the autonomous evolution of X within a chosen time window. Within this window is an embedded dimension T_D number of time stamped slices of X sampled at a time interval, τ, the tick-time. It says that each discrete state of X holds for a tick-duration of $(t+T_{(p+1)\tau}) - (t+T_{p\tau})$ units of time where $0 \le T_p \le T_{D-1}$.

(ii) The structure of segments of X with (I,T) training pattern pairs.

A training pattern series such as series (3.0) described earlier in Chapter 3, is

$(I_1,T_1), (I_2,T_2), \ , \ , \ , (I_{tot\_pat},T_{tot\_pat})$

where each (I_m,T_m) pattern pair, $1 \le m \le tot\_pat$, consists of a time-series, I, made up of num_in time slices of input data, X:-

$$X_1, X_2, \ , \ , X_{num\_in} \tag{5.1}$$

The dimension of num_in is equal to the dimension of the input nodes of the configured network. To load the input data into the network, each node, N_n, $1 \le n \le num\_in$, of the input nodes,

$$N_1, N_2, \ , \ , N_{num\_in} \tag{5.2}$$

receives a corresponding time slice, X_n of data from the series in (5.0). The expected result of extrapolating the series in (5.1) is the value of $X_{(num\_in+1)}$, the learnt value for T.

The first input pattern, (I_m,T_m) is a continuous series, $X_{(t)}, X_{(t+1)}, \ , \ , X_{(t+num\_in-1)}$. The neural model learns to extrapolate the pattern to a future value, $X'_{(t+num\_in)}$ target to correlate as closely to the tracking value, $X_{(t+num\_in)}$ as possible. The succeeding pattern, $X_{(t+1)}, X_{(t+2)}, \ , \ , X_{(t+num\_in)}$ has a corresponding target value of $X_{(t+num\_in+1)}$. The succeeding pattern pair, (I_{m+1},T_{m+1}) coincides with the next series from $X_{(num\_in+2)}, \ , \ , X_{((num\_in*2)+2)}$. This procedure using a first in first out principle is repeated until the depletion of all the T_D time-slices of data.

(iv) The scaling of the input data β_i' to between 0 and 1, equation (3.1) in Chapter 3,

$\beta_i = (\varphi\ ((\beta_i' - min\_val) / (max\_val - min\_val))) + \Upsilon$

where min_val and max_val are the minimum and maximum values of the sampled time-series. The input representation in model sN1 has $\varphi = 0.9$ and $\Upsilon = 0.0$, and model sN2 has $\varphi = 0.8$ and $\Upsilon = 0.0$. The method in model N follows the constants named in Chapter 3, $\varphi = 0.8$ and $\Upsilon = 0.1$.

(iv) The dimension of the input layer, I for analysing X is expressed as,

$$I = sI + \sigma \tag{5.3}$$

where sI is a constant representing the minimum size of the tracking window, I, and σ allows fine adjustments to the overall dimension of I. As a result, σ controls the dimension of I, or rather, *num_in* as in (5.1) and (5.2). Throughout the experiments, sI was assigned to 5 and a range of σ used to indicate input layer sizes.

(v) The dimension of the hidden layer, H is,

$$H = I\,\phi \quad . \tag{5.4}$$

where H is the result of increasing I by multiples of ϕ [Lippma87]. Three values of ϕ, were investigated: `0.9`, `2`, and `3`. They refer to as `RuleS`, `RuleD`, and `RuleT` configuration function respectively.

(vi) The **Relative Threshold Prediction Index, RTPI**, which is,

$$RTPI = \frac{R}{C}\|C - TP\| + (C - TP) + \frac{R}{C} \tag{5.5}$$

is an index to indicate the prediction accuracy of a network above a minimum prediction accuracy percentage, TP. Networks with a prediction accuracy below the minimum expectation of TP have negative values. The index is a summation of three performance indicators based on the unrealisable weighted return (%), R, made over the percentage of accuracy achieved by the network, C, the test correlation (%). The first is a cost function calculating the unrealisable weighted return (%) for the percentage value between the median at TP and the percentage at C. The second measures the percentage difference between TP and C. Finally, the third measures the unrealisable weighted return achievable per percentage of accuracy achieved in C. TP was conservatively set to 50% in this thesis.

5.1.2 Experimental Data

One of the factors instrumental to the success of applying neural networks for financial use is the choice of training and test patterns. At different periods of any financial data, the trend is described as *consolidation*, *bull*, or *bear*. They describe a "non-diversifying"; "rising" and "falling" trend respectively. As the *bull* and *bear* are pattern-wise, reflections of each other, the *bear* was replaced by the transitional *recovery*, "U-turn", trend. The experimental data sets used, therefore, show a *consolidation*, *bull* and *recovery* data trend.

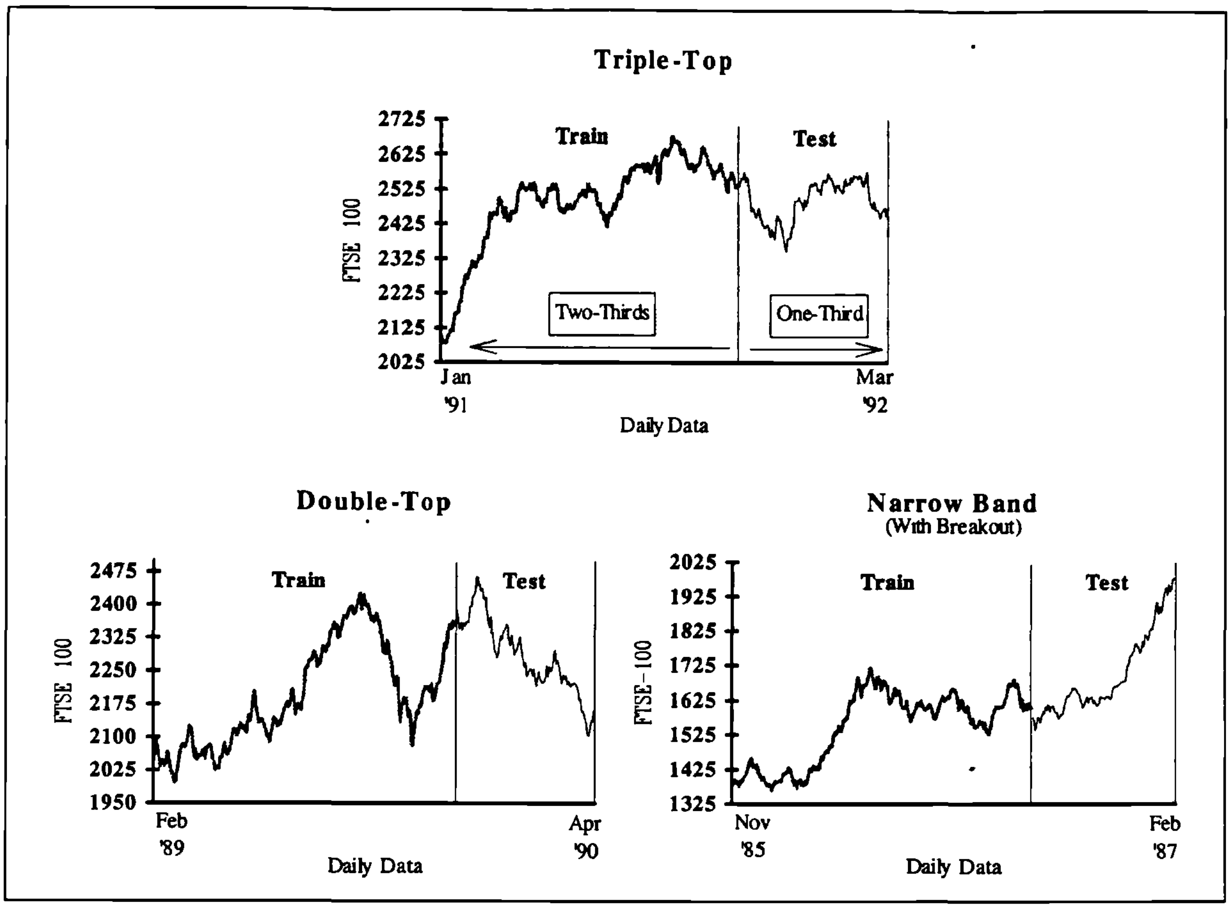


Figure 5.1 Types Of Pattern Formations In A *Consolidation* Trend

A *consolidation* trend can assume various pattern formations, such as a *triple-top*, *double-top* and *narrow band*, or merely modulating [Murphy86]. A *triple-top*, such as the example in Figure 5.1, is as the name suggests, a pattern formation which has three almost identical "tops" bounded by an upper and lower range of values. On a similar note, a *double-top* pattern has two instead of three "tops". A *narrow band* which is the flat section of the pattern in Figure 5.1, is a special case of a modulating *consolidation* trend. It has no specific pattern formation and modulates within a relatively narrow range of values.

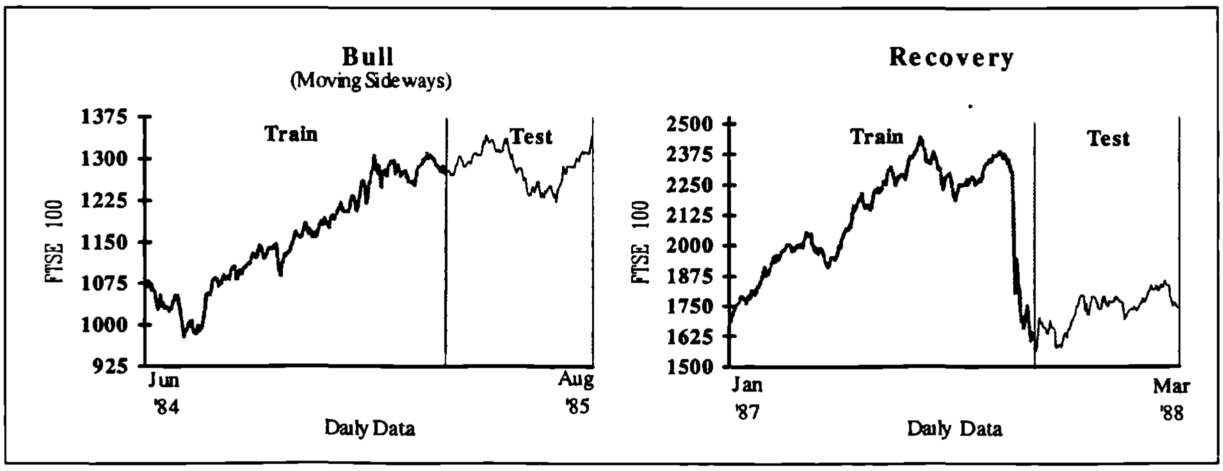


Figure 5.2 A *Bull* Trend

Figure 5.3 A *Recovery* Trend

On the contrary, a *bull* trend is an asymmetrical "rising" pattern formation. As a result, the upper and lower range of values in the test data set is out of the range of those of the train data set. A *recovery* trend is also an asymmetrical pattern but the "U-turn" formation has the upper and lower bounds of the test data set within the range of those of the train data set.

The *consolidation* trend is especially chosen for the main parts of the research because the train and test data sets are bounded within the same limits and the pattern formation on both sectors could be symmetrical or are reflections of each other.

The patterns selected for the experimental data sets have either symmetrical, asymmetrical, or reflections of patterns between the train and the test data sets.

(i) the *triple-top* pattern formation is symmetrical and has identical patterns in the train and test data sets. It was used for the optimisation experiments to study the neural networks' capability to extrapolate by "recognising" an identical pattern - learnt in the train set - in the test data set. The three "tops" distribute almost proportionately when the data set is divided into two-thirds for training and one-third for testing.

(ii) the *double-top* pattern formation is rather similar to a *triple-top*. It was used to test the forecasting ability of the optimised networks of the two models in an almost similar pattern but which is from a different historical period.

(iii) the *narrow band* pattern is also symmetrical and has the pattern in the train data set reflected in the test set. As such, both data sectors do not lie in the same range of values. The pattern was used to test the optimised networks' capability in the event of a *breakout*, a transitional pattern formation with *bull* pattern features.

(iv) The *bull* pattern in Figure 5.2 is asymmetrical and does not have identical patterns in both the train and test data sets. It was used to test the optimised networks' capability to learn an easy to train pattern and to forecast a test pattern which is almost within the range of the train data set but is described as moving *sideways*.

(v) The *recovery* pattern is also asymmetrical but both the train and test data sectors have identical patterns. It offers the situation to observe the models' behaviour in forecasting a pattern which is similar to one learnt but lies in a different range of values.

Forecasting simulation studies require input pattern formations in the form of the *(I,T)* pairs discussed earlier. The method of formation applies to the training and test patterns. There is, however, a subtle difference in their interpretation. For historical simulations, the patterns from the train set are used for optimisation and during training they are continuously assessed as to how closely the forecasts delivered, *T*, fit the tracking values of $X_{(num\_in+1)}$. Like the training patterns, the test patterns, which have not been seen by the network before, also use the values of *X* ,to track the values of *T*. There is, however, a more significant difference in forward simulations. The forecast values of T are for the time being the projected values for X, well before its evolution. Consequently, assessments cannot be made until after a time lapse equal to X's `tick-duration`, the interval expected of X to evolve to.

The pattern specification language in `spec.dat` was filled out as explained for Figure 4.9 in Chapter 4. The assignment of `Number of input factors` to `1`, also in `spec.dat`, allows the **NFS** to load the network for time-series training and testing.

5.2 Data Form Of The Index

A series of three-tier experiments were designed to accomplish the aim of this thesis. Each tier was set objectives to steer the direction of the next level. The objective of the first tier was to choose a suitable physical representation for the raw data as inputs to the network.

In the initial stages of using neural technology for forecasting, the assortment of network parameters and the range of values that these parameters can be assigned are often bewildering. The uncertainty is compounded by the time taken to train a network. In any case, even if some reasonable constants are assigned to the parameters, a question that often arises when analysing non-linear time-series is the physical forms of the input data.

Experiment 1 Data Form

The objective of Experiment 1 was to resolve the ambiguity over a *logarithmic* or a *linear* data form, as more suitable for applying the Index on the **NFS**.

In the extraction of facts from financial data, it is mandatory to convert the plethora of raw data into useful graphs before analysing: often, a *logarithmic* or a *linear* arithmetic conversion is used. The method chosen is suitable for a specific forecasting application or a specific type of data. In practice, a method is applied simply because it has been tried and proven, or merely based on subjective inclination.

The specifications for the experiment are in Figure 5.4.

Experiment 1	: A *Logarithmic* against a *Linear* Data Form.
Constants	: Model = N, Configuration = `RuleD`,
	`Momentum` = 0.05, `Tolerance` = 0.02, `Learn Rate` = 0.1
	`Proportion of data for Training` Two-Thirds
Data	: *Consolidation* data set with $^2/_3$ used for training and $^1{}_3$ for testing .
Variable	: $\sigma = 1, \ldots, 5$
	`Number of Training Cycles` 1000 to 7000
	`Index Chart` Linear, Log

Figure 5.4 Specifications For Experiment 1

The evaluation takes into account the time taken to train the network, the percentage of forecast accuracy calculated for the test data set (test correlation (%)) for the three methods of correlation measurement, VCorr, DCorr and VDCorr (see Chapter 3), and the forecast error margins.

5.3 Optimisation Of Neural Forecasting Models

The investigations of the second tier of experiments were to establish an optimised network for models sNx and N on the *triple-top* pattern. The procedures undertaken allow comparison studies on the forecasting potential of model sNx with the tenable model N, and to choose the better variant of sNx, sN1 or sN2.

Experiment 2a Network Configuration

The objective of Experiment 2a was to establish for models sNx and N, the relationship of the dimension of the hidden layer, H to the input layer, I, and a set of σ (the controlling variable for I) that has the potential to forecast the Index with fairly consistent results. The relationship was investigated under the RuleS, RuleD, and RuleT configuration functions defined in equation (5.4). The notion of forecasting accuracy was also investigated using three, VCorr, DCorr and VDCorr, methods of measurement (see Chapter 3 for their definitions).

The size of Number of Input Nodes for I was uncertain at this stage. The choice of σ used ranged from 1 to 5 and the training was carried out in steps of 7*1000 training cycles. The details of the experiment are in the worksheet in Figure 5.5.

Experiment 2a3(iii) : Model N with RuleT Configuration

Experiment 2a3(ii) : Model N with RuleD Configuration

Experiment 2a3(i) : Model N with RuleS Configuration

Experiment 2a2(iii) : Model sN2 with RuleT Configuration

Experiment 2a2(ii) : Model sN2 with RuleD Configuration

Experiment 2a2(i) : Model sN2 with RuleS Configuration

Experiment 2a1(iii) : Model sN1 with RuleT Configuration

Experiment 2a1(ii) : Model sN1 with RuleD Configuration

Experiment 2a1(i)	: Model sN1 with RuleS Configuration
Constants	: Momentum = 0.05, Tolerance = 0.02, Learn Rate = 0.1 Proportion of data for Training Two-Thirds
Data	: *Consolidation* data set with $^2/_3$ used for training and $^1/_3$ for testing.
Dependent	: Index Chart established from Experiment 1
Variable	: $\sigma = 1, \ldots, 5$ Number of Training Cycles 1000 to 7000

Figure 5.5 Specifications For Experiments 2a1(i)-2a1(iii) To 2a3(i)-2a3(iii)

Experiment 2b Tolerance

The objective of Experiment 2b was to check the possibility that the performances of any of the networks, for all of the set of σ configured over the three configuration functions, can be improved by variables of the tolerance level.

The scaling method in model sN*x* extrapolates a time-series and accentuates the extrapolated value. The tolerance is one of the three learning parameters, the others being the learning rate and the momentum. The learn rate was not cross-examined because a small learn rate of 0.1 was used in the experiments and its suits the strategy of the scaling method. Similarly, the momentum term was omitted as it improves the speed of convergence [RumHin86], which is not the main priority of this study.

The experimental details are in Figure 5.6.

Experiment 2b3(iii)	: Model N against Tolerance
Constants	: RuleT Configuration, Momentum = 0.05, Learn Rate = 0.1

Experiment 2b3(ii)	: Model N against Tolerance
Constants	: RuleD Configuration, Momentum = 0.05, Learn Rate = 0.1

Experiment 2b3(i)	: Model N against Tolerance
Constants	: RuleS Configuration, Momentum = 0.05, Learn Rate = 0.1

Experiment 2b2(iii)	: Model sN2 against Tolerance
Constants	: RuleT Configuration, Momentum = 0.05, Learn Rate = 0.1

Experiment 2b2(ii)	: Model sN2 against Tolerance
Constants	: RuleD Configuration, Momentum = 0.05, Learn Rate = 0.1

Experiment 2b2(i)	: Model sN2 against Tolerance
Constants	: RuleS Configuration, Momentum = 0.05, Learn Rate = 0.1

Experiment 2b1(iii)	: Model sN1 against Tolerance
Constants	: RuleT Configuration, Momentum = 0.05, Learn Rate = 0.1

Experiment 2b1(ii)	: Model sN1 against Tolerance
Constants	: RuleD Configuration, Momentum = 0.05, Learn Rate = 0.1

Experiment 2b1(i)	: Model sN1 against Tolerance
Constants	: RuleS Configuration, Momentum = 0.05, Learn Rate = 0.1
	Proportion of data for Training Two-Thirds
Data	: *Consolidation* data set with $^2/_3$ used for training and $^1/_3$ for testing.
Dependent	: Index Chart from Experiment 1
Variable	: $\sigma = 1, \ldots, 5$
	Number of Training Cycles 1000 to 7000
	Tolerance 0.03

Figure 5.6 Specifications For Experiments 2b1(i) - 2b1(iii) To 2b3(i) - 2b3(iii)

All the experiments carried out were evaluated for a suitable method of measuring the correlation of a prediction. The test correlation (%) and the corresponding weighted return(%) calculated for VCorr, DCorr, and VDCorr definitions for measuring correlation were compared.

The evaluation for a suitable configuration function was based on consistent test correlation (%) of 50% an above, and a corresponding positive weighted return (%) across the 4000-7000 training cycles sampled. The **RTPI** indicator was also used in the evaluation and the forecast error margins were compared.

In addition to selecting an optimal N network, a representative optimal network of model sN*x*, the better of sN1 and sN2, was also identified. The criterion for selection were in addition to those used for the configuration function, the size of the network based on the configuration function.

5.4 Adaptation Of Optimised Networks

The objective of the third tier of experiments was to test the adaptation of the optimised sNx and N networks on different data patterns which have been obtained from different historical periods. The distinctive pattern features of a *double-top, narrow band, bull* and *recovery* patterns were tested.

The single most sought after characteristic of any forecasting tool is its ability to adapt to changes to the non-linearity of an evolving time-series. Ideally, the tool should have the capability to adjust to changes to time grains and pattern cycles as and when they occur. The main obstacle to the implementation of such a tool is that current computing techniques are inadequate: they are only good at solving problems viewed within a static time window while those problems are themselves dynamic processes. As such, the experiments in this tier are simply to observe the flexible application of the optimised networks to a different time domain, on different chart patterns which have identical or non-identical train and test data sets. It is not aimed to test their dynamic adaptation to non-linear changes as such.

Experiment 3a Optimised Networks And A *Double-Top* Pattern

The objective of Experiment 3a was to test how the optimised networks of the models perform on in a *double-top* pattern and how the accuracy of the forecasts vary.

The details are outlined in Figure 5.7.

Experiment 3a(ii) : Model sNx and *Double-Top* Pattern

Experiment 3a(i)	: Model N and *Double-Top* Pattern
Constants	: Momentum = 0.05
Data	: *Double-Top* pattern data set
Dependent	: Index Chart from Experiment 1
	Number of Hidden Nodes from Experiment 2a and 2b
	Number of Input Nodes σ optimised from Experiment 2a and 2b
	Tolerance optimised from Experiments 2b
Variable	: σ if necessary else see above
	Tolerance if necessary else above
	Number of Training Cycles 1000 to 7000
	Proportion of data for Training Two-Thirds

Figure 5.7 Specifications For Experiments 3a(i) And 3a(ii)

Experiment 3b Optimised Networks And A *Narrow Band* Pattern With A *Breakout*

Experiment 3c Optimised Networks And A *Bull* Pattern *Moving Sideways*

Experiment 3d Optimised Networks And A *Recovery* Pattern

Experiments 3b, 3c and 3d are similar to Experiment 3a, except for the patterns of the data set. Consequently, the specifications are similar to those in Figure 5.7.

The evaluation of this study, on the adaptive nature of the optimised networks representative of the two models was based on their performances on, the test correlation (%), the weighted return (%), the **RTPI** values, and forecast error margins on specific features of the pattern.

5.5 Summary

The **NFS** was used to verify the forecasting viability of the new method of representing data in network input patterns in models sNx. It was compared to a tenable representation method in model N. The experimental data sets showed: *consolidation* ("non-diversifying" patterns, *triple-top*, *double-top*, and *narrow-band* with *breakout*), *bull* ("rising" pattern) and *recovery* ("U-turn" pattern) data trends. The experiments were organised into three-tiers.

The objective of the first tier of experiments was to establish a suitable data form for using the FTSE-100 Index on the NFS. Networks of the tenable model N were used for historical simulations using - *logarithmic* and *linear* - raw data forms of a *triple-top* pattern. The selection was based on the forecast error margins.

The objective of the second tier of experiments was to optimise the sNx and N networks on the *triple-top* pattern. In addition, it established the most suitable correlation measurement (`VCorr`, `DCorr`, and `VDCorr` as described in Chapter 3) for measuring the prediction accuracy. The data form chosen was applied to the input data representation methods in models sNx (studied as sN1 and sN2) and N. The parameters optimised on networks of these models were:

(i) the number of input nodes, I, studied as a function (equation 5.3) of σ.
A set of σ ranging from `1` to `5` were used.

(ii) the number of hidden nodes, H, as three configuration functions (equation 5.4) of I. These functions (`RuleS`, `RuleD`, and `RuleT`) were applied to the set of σ to configure networks of the models.

(iii) the tolerance level.

All the networks configured to the above specifications were optimised over variable tolerance values: 0.2 and 0.3.

The most suitable correlation measurement established after (i) was subsequently used to measure the percentage of prediction accuracy, the test correlation (%). The corresponding weighted return (%) was also used for assessment.

The optimisation procedure established a suitable configuration function which relates H with I for configuring a network. The criterion for evaluation were a combination of: the test correlation (%), the corresponding weighted return (%), both combined together as the **Relative Threshold Prediction Index** (**RTPI**, see equation (5.5)), and the network sizes in terms of σ and the configuration functions. The **RTPI** is a new indicator introduced in this thesis to filter out networks forecasting above a chosen threshold and a credit in the corresponding weighted return (%).

The objective of the third tier of experiments was to test the adaptation of the optimised networks of sNx - the better of sN1 and sN2 - and N models. It was to test the networks' capacity to accommodate to new pattern formations from different historical periods. The behaviour and the prediction capability of the networks on patterns - *double-top, narrow band, bull* and *recovery* - which may have similar or dissimilar train and test data sets were compared and contrasted. Their evaluations were based on the test correlation (%), the weighted return (%), the **RTPI** indicator values, and the forecast error margins at specific regions of each pattern.

Chapter 6

Performance Analysis

This chapter evaluates time-series forecasting by models sNx and N. It covers the optimisation of networks for each model and compares their application to new data sets having similar or dissimilar train and test patterns. The optimisation procedures select a suitable raw input data form, prediction accuracy measurement and configuration function, and the better of models sN1 and sN2. The evaluation discusses amongst other things, the test correlation (%), its weighted return (%) and the RTPI indicator.

6.1 Raw Data Form

The strategy to identify an NFS compatible raw input data form compares the *logarithmic* and *linear* arithmetic methods of transforming raw data to meaningful structures for analysis.

Identical experiments using both data forms studied networks of model N which were configured using the `RuleD` configuration function with input layer sizes, σ, ranging from `1` to `5`. The notations, `RuleD` and σ, are discussed in equation (5.3) and (5.4) in Chapter 5.

The evaluation criterion are, the prediction accuracy (test correlation (%)), the trained network's capability to forecast the major movements of the test data set by the corresponding unrealisable weighted return (%), the number of training cycles delivering these percentages, and the forecast error margins. The definitions of three test correlation measurements and their return (%) are discussed in the Data Recording section of Chapter 3.

The raw *linear* inputs from the *triple-top* pattern were scaled down to between `0` and `1` before they were loaded into a network and tested. The *logarithm* of the same set of input data were loaded and similarly tested. The percentage of prediction accuracy recorded for all the networks tested showed, for both the *logarithmic* and *linear* input data form that the best set of results are from the `VCorr` method of measurement. In addition, it showed networks using *logarithmic* inputs converged much more rapidly. For example, a network with σ equal to `1` and achieving 53.85% accuracy required only 1000 training cycles to bring the average mean square error down to `0.006`. This contrasts considerably with the performance of the *linear* data form, where the same network delivered 51.92% accuracy but required 4000 cycles to bring the average mean square error down to only `0.33`. However, the prediction accuracy for the *logarithmic* inputs remained unchanged for the next 2000 training cycles. As a result of this uncharacteristic behaviour, the focus of evaluation was shifted to the forecast error margins. The graphs in Figure 6.1 illustrate forecasts of a network with σ equal to 4 using *linear* inputs, and two networks using σ equal to 1 and 4 for *logarithmic* inputs. The outputs from the latter inputs are almost indistinguishable and they are remarkably poor compared to the *linear* inputs. These test cases show the *logarithmic* data form is not suitable for the **NFS**, the prediction accuracy measurements merit further investigation. More importantly, the conclusion justifies the investigation of model sNx. The conclusion, however, do not indicate that the *logarithm* is unsuitable for other systems or Backpropagation models.

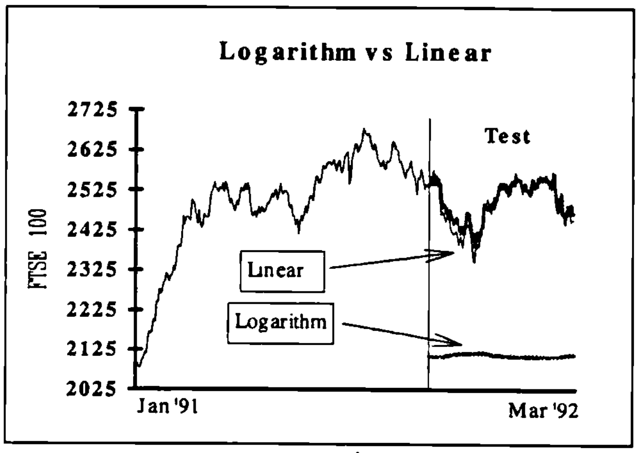


Figure 6.1 *Logarithmic* vs *Linear* Inputs

6.2 Network Optimisation

The strategy investigating the validity of model sNx was to optimise networks of models sNx (sN1 and sN2) and N exhaustively over five input layer sizes for all the three network configuration functions studied. Following that, all experimental cases were cross-tested against the learning coefficient, `Tolerance`.

Identical experiments with input layer size, σ, assigned from 1 to 5 were configured with hidden nodes by the `RuleS`, `RuleD` and `RuleT` "input nodes to hidden nodes" ratio (see equation 5.4 of Chapter 5). The `Tolerance` was assigned to 0.2 and 0.3. The prediction accuracy, the correlation of the forecast with the direction of movement of the tracking value, were studied by the `VCorr`, `DCorr` and `VDCorr` interpretations of prediction accuracy. The unrealisable weighted return as per prediction accuracy measured, measures the generalisation of the trained network. Statistics from both performance measurements were used to calculate the Relative Threshold Prediction Index (**RTPI**). This new indicator (discussed in equation 5.5 in Chapter 5) classifies a network to a positive or negative value taking into account the network's prediction accuracy in relation to a desired threshold level (set at 50% for this thesis) and its weighted return (%).

The methodology for obtaining optimal networks of models sNx and N was establishing:

(1) a suitable prediction accuracy measurement, the best of: `VCorr`, `DCorr` and `VDCorr`.

(2) a suitable configuration function, the best of: `RuleS`, `RuleD` and `RuleT`.

(3) and an optimum model sNx, the better of: sN1 and sN2.

6.2.1 Correlation Measurement

That a prediction, by the requirement of this thesis, is correct, depends on the interpretation of the correlation of the prediction with the direction of the tracking data. The term, correlation, can be defined in three ways: by value, by direction, or by value and direction. The first, which is VCorr, states that a prediction for tomorrow is correct if its movement compared with today's tracking value is the same as the tracking Index movement, from today to tomorrow (Figure 3.2a). The second, DCorr, requires the forecast movement, from today to tomorrow, to be in tandem with that of the tracking Index (Figure 3.2b). The third, VDCorr, the most stringent, is a combination of the definitions of VCorr and DCorr (Figure 3.2c). These measurement methods are important issues to neural forecasting research, as each has positive interpretative features which are beneficial, in the real world, for applying the prediction.

A sample of these three measurements which have been recorded for Experiments 2a(i) to 2a(iii) in Table 6.1, are in Appendix B. It is representative of the performance trend of each of the methods across the spectrum of experiments carried out. A subset of Table 6.1 is shown in Table 6.2. It records the statistics for each method of measurement in two columns pairs, the correlation (%) and the weighted return (%) for both the train and test sectors.

Model	Cycle	VCorrelation		VReturn		DCorrelation		DReturn		VDCorrelation		VDReturn	
		(Train %	Test %)	(Train %	Test %)	(Train %	Test %)	(Train%	Test%)	(Train %	Test %)	(Train %	Test %)
N	4000	80.48	49.04	88.78	9.39	69.05	43.27	60.81	-17.14	57.14	12.50	52.82	-75.93
	5000	79.52	45.19	87.25	6.58	67.62	42.31	60.69	-15.57	56.67	13.46	51.87	-76.49
	6000	74.29	46.15	79.50	6.96	69.05	41.35	60.68	-17.48	54.29	14.42	46.28	-73.79
	7000	74.29	46.15	78.66	6.96	70.00	44.23	61.77	- 4.53	54.76	17.31	46.66	-60.84
	8000	75.71	46.15	82.32	6.96	67.14	43.27	58.11	- 6.38	52.38	16.35	45.02	-63.51
sN1	4000	80.48	55.77	89.67	9.62	75.24	50.96	75.36	- 3.26	68.57	36.54	71.61	-34.44
	5000	70.00	45.19	73.00	- 3.01	64.76	50.00	55.66	- 3.72	52.86	25.96	42.54	-46.92
	6000	72.86	50.00	76.83	1.43	68.57	50.00	64.67	- 4.62	60.00	32.69	56.13	-37.88
	7000	73.33	47.12	78.26	- 3.6	69.52	50.96	63.85	- 0.58	58.57	27.88	53.36	-49.86
sN2	4000	68.57	49.04	59.56	- 0.82	70.00	45.19	54.06	1.55	53.33	26.92	29.76	-41.08
	5000	67.14	45.19	64.64	3.81	66.67	47.12	52.37	-13.50	51.43	24.04	33.25	-55.16
	6000	65.24	47.12	58.65	8.79	67.14	46.15	45.85	- 9.06	44.76	20.19	15.39	-56.10
	7000	63.33	44.23	58.89	- 2.82	66.19	47.12	49.47	- 6.68	44.76	19.23	21.70	-55.07

Table 6.2 Prediction Accuracy And Weighted Return By VCorr, DCorr And VDCorr

The objective of this analysis was to choose the measurement method that delivers consistent and always, in relative terms, the best positive valued statistics. The evaluation considers the test data set statistics rather than those of the train data set because they give a better indication of the training network's capability to extrapolate. An initial comparisons of the test correlation (%) show that measurements by VCorr deliver the best set of percentages, followed closely by DCorr, and significantly worst by VDCorr. Further, the performance order applies for the weighted return (%). A combination of performance by the correlation and the weighted return eliminates DCorr. The conclusion for VCorr is justified by the consistently similar performance trend throughout the other experimental recordings.

6.2.2 Prediction Accuracy

The singular statistic, often sought after in forecasting, is the prediction accuracy. Being an ambiguous term, the first objective was to identify a definition suited to the *consolidation* data trend. This trend dictates: the range of values between the highest and lowest data value is small; the movements are also usually small; and the direction of the movements are consistent for a very short-term, but possibly never conforming to any specific direction.

The evaluation for prediction accuracy - or rather the test correlation - according to the definitions of VCorr, rates networks by their correlation percentages across a window of between 4000 to 7000 training cycles. A minimum expectation of 50% prediction accuracy was used as the threshold percentage. A network is rated: *4Y* if it delivered test correlation (%) above the threshold for each of the 4*1000 training cycles; *3Y* for three out of four cycles; *2Y* for two out of four cycles; *Y* for only one out of four cycles; and *N* for anything less. This rating system is applied to summarise the prediction accuracy of all the networks tested. They are charted in Appendix C (Figure 6.2a and 6.2b), Appendix D (Figure 6.3a and 6.3b), and Appendix E (Figure 6.4a and 6.4b), and summarised in Table 6.3a to 6.3c. To visualise the ratings scored, shading intensities are used.

Models: sN1, sN2 And N

The sN1 networks configured by the RuleS configuration function deliver better test correlation (%) with a Tolerance of 0.2 rather than 0.3 (Table 6.3a). In addition, networks with σ equal to 5 are unsuitable, as they consistently deliver forecasts below the 50% threshold. The networks with σ equal to 4 and 2 with a Tolerance of 0.2 and 0.3 respectively, are the better performers. The rest of the networks have an average performance, where only two out of the four training cycles have percentages above the threshold value, resulting in a less consistent stable test correlation (%) landscape. The RuleD configuration function tends to perform better with a Tolerance of 0.3. It is especially suited to networks with σ equal to 4 and 5, which by coincidence

Model	Tol	σ Nodes	RuleS	RuleD	RuleT
sN1	0.2	1	Y	N	3Y
		2	Y	2Y	4Y
		3	Y	Y	2Y
		4	4Y	3Y	2Y
		5	N	2Y	N
	0.3	1	Y	N	Y
		2	3Y	Y	N
		3	Y	N	4Y
		4	N	4Y	N
		5	N	4Y	3Y

Table 6.3a sN1 Networks And Accuracy Grades

is unsuitable with the RuleS function. For a Tolerance of 0.2 RuleS and RuleD are rather similar, with RuleS having a slight advantage. The RuleT configuration function with a Tolerance of 0.2 tends to be the most consistent and stable, although the network with σ equal to 5 is unclassified. On the whole, a RuleD function with a Tolerance of 0.3 appears to be the best performing sector. This is because two of its networks deliver *4Y* accuracy grades.

Model	Tol	σ Nodes	RuleS	RuleD	RuleT
sN2	0.2	1	N	2Y	Y
		2	4Y	Y	3Y
		3	N	3Y	3Y
		4	4Y	3Y	Y
		5	2Y	N	N
	0.3	1	2Y	2Y	2Y
		2	N	N	Y
		3	N	2Y	N
		4	4Y	N	N
		5	2Y	Y	Y

Table 6.3b sN2 Networks And Accuracy Grades

The scoring contours for sN2 networks (Table 6.3b), suggest better prediction accuracy are obtained from networks using the RuleS function. Although it has more instances of non-classified networks, it also has more instances of those scoring over 50% for all the four cycles. The networks using the RuleD and RuleT functions have almost similar scoring contours, especially with a Tolerance of 0.2. The networks from both configuration functions did not manage to have all the four cycles scoring above 50%. On the whole, a RuleS function using, σ equal to 2 or 4, and a Tolerance of 0.2 is the best combination.

Model	Tol	σ Nodes	RuleS	RuleD	RuleT
N	0.2	1	N	3Y	N
		2	N	2Y	N
		3	N	2Y	N
		4	2Y	4Y	3Y
		5	4Y	2Y	N
	0.3	1	2Y	3Y	2Y
		2	3Y	N	N
		3	Y	N	N
		4	Y	2Y	N
		5	N	N	N

Table 6.3c N Networks And Accuracy Grades

The N networks (Table 6.3c), tend to perform best with a RuleD function. It is not suited for a RuleT function. It is only suitable with a RuleS function for σ equal to 4 and 5 in the case of a Tolerance of 0.2 , or the smaller values of σ for a Tolerance of 0.3. On the whole, a RuleD function using σ equal to 4 and a Tolerance of 0.2 is the best combination.

Configurations: RuleS, RuleD Or RuleT

An analysis of the test correlation (%) landscape suggests for model sN*x* that any one of the three configuration functions can be used for prediction. This is based solely on the incidence of consistent percentages above 50% across the 4*1000 training cycles evaluation window. Networks of model sN1, in particular, tend to deliver pockets of consistent percentages on all the three configuration functions. The determinants are the input layer size and the Tolerance level. There is a marginally better spread of *2Y* and *3Y* ratings towards the RuleD and RuleT functions. In the case of model sN2, a RuleS function with a Tolerance of 0.2 offers pockets of *4Y* ratings, outperforming the other two configuration functions.

It is, however, clear for model N, that a RuleD function with a Tolerance of 0.2 is the only suitable configuration, eliminating both the RuleS and RuleT functions.

6.2.3 Curve-Fitting

A statistic often under quoted, but important to non-linear forecasting is a measure of network generalisation as it extrapolates. A trained network might extrapolate satisfactorily, according to statistics of forecasting correctly the number of chances tried, but predictions might only be made correctly at minor rather than major data movements. Naturally, there is preference for trained networks capable of the latter, as per correct prediction calculated.

The evaluation for network generalisation is the weighted return (%), the calculation of which is dependent on the correlation method used. Each group of 1000 training cycles is graded by the bands of weighted return (%): 0-5%, 5-10%, 10-15%, 15-20%, 20-25%, and unclassified for negative returns. These bands are used to summarise the weighted return (%) of the same set of networks rated earlier for their accuracy. The values are charted in Appendix F (Figure 6.5a to 6.5e), Appendix G (Figure 6.6a to 6.6e), and Appendix H (Figure 6.7a to 6.7e). They are summarised in Table 6.4a to 6.4c, and again the bands graded are highlighted by shading intensities.

Models: sN1, sN2 And N

The shading intensities of the sN1 networks (Table 6.4a) configured with the RuleS function appear to perform better when σ is equal to 2 to 4 for a Tolerance of 0.2 and, σ is equal to 2 and 3 for Tolerance of 0.3. The performance faired better for all the networks with the RuleD function, particularly, using a Tolerance of 0.2. The results of networks using a Tolerance of 0.3 appear to be clearly defined: σ equal to 4 and 5 performed well, especially 4. The RuleT function also appears to deliver relatively acceptable weighted return (%) and gently rolling weighted return (%) landscapes with a Tolerance of 0.2. It is also slightly better than the RuleS function. In general, the best network delivering a consistent and high weighted return (%) is a RuleD function with σ equal to 4 and a Tolerance of 0.3.

The definitions of the bands of weighted return (%) are more defined for networks of model sN2 (Table 6.4b). These networks generalise better with a Tolerance of 0.2, in particular a RuleS function with σ equal to 2 and 5. The networks with a RuleD function also perform satisfactorily with σ equal to 4 or, a RuleT function with σ equal to 2. On the whole, the best network has a RuleS function with σ equal to 2 and a Tolerance of 0.2. However, the level of return is not as good as those made by networks of model sN1.

The weighted return (%) delivered by networks of model N (Table 6.4c) are lesser than those of model sN1 and sN2. In this model, the weighted return (%) landscape are

Model	Tol	σ Nodes	Cycles	RuleS	RuleD	RuleT
sN1	0.2	1	4000	5-10	0-5	10-15
			5000		5-10	5-10
			6000	0-5	0-5	0-5
			7000		0-5	10-15
		2	4000	15-20	5-10	10-15
			5000	5-10		10-15
			6000	5-10	5-10	15-20
			7000	5-10	15-20	15-20
		3	4000	5-10	10-15	
			5000	0-5	15-20	
			6000	0-5	10-15	5-10
			7000	5-10	10-15	15-20
		4	4000	10-15	10-15	10-15
			5000	10-15	15-20	10-15
			6000	10-15	10-15	
			7000	10-15		
		5	4000		0-5	5-10
			5000		0-5	0-5
			6000		10-15	5-10
			7000		10-15	5-10
	0.3	1	4000			
			5000	5-10		15-20
			6000	0-5		5-10
			7000			5-10
		2	4000	10-15		
			5000	10-15	5-10	
			6000	10-15	5-10	
			7000	10-15	10-15	
		3	4000	0-5		
			5000	10-15		0-5
			6000			10-15
			7000	5-10		10-15
		4	4000		20-25	
			5000		20-25	
			6000		15-20	
			7000		15-20	
		5	4000		5-10	0-5
			5000		10-15	5-10
			6000		15-20	10-15
			7000		10-15	5-10

Table 6.4a sN1 Networks And Curve-Fitting Grades

Model	Tol	σ Nodes	Cycles	RuleS	RuleD	RuleT
sN2	0.2	1	4000			20-25
			5000	0-5	10-15	
			6000	5-10	10-15	
			7000			
		2	4000	10-15	5-10	10-15
			5000	15-20		10-15
			6000	15-20		10-15
			7000	15-20	10-15	5-10
		3	4000		0-5	5-10
			5000		5-10	10-15
			6000		10-15	0-5
			7000		5-10	5-10
		4	4000	5-10	0-5	5-10
			5000	5-10	15-20	5-10
			6000	5-10	5-10	5-10
			7000	5-10	15-20	0-5
		5	4000	5-10	5-10	0-5
			5000	10-15		5-10
			6000	0-5		5-10
			7000	20-25		5-10
	0.3	1	4000			5-10
			5000	5-10	0-5	5-10
			6000	10-15	5-10	5-10
			7000	10-15	0-5	5-10
		2	4000			0-5
			5000	0-5	5-10	5-10
			6000	0-5		
			7000		5-10	
		3	4000			5-10
			5000			
			6000		5-10	5-10
			7000		0-5	5-10
		4	4000	10-15		
			5000	0-5		5-10
			6000	0-5		
			7000	5-10		
		5	4000	5-10		
			5000			
			6000		0-5	
			7000	5-10	0-5	

Table 6.4b sN2 Networks And Curve-Fitting Grades

Model	Tol	σ Nodes	Cycles	RuleS	RuleD	RuleT
N	0.2	1	4000	5-10	5-10	5-10
			5000	5-10	10-15	10-15
			6000	5-10	20-25	5-10
			7000	5-10	0-5	5-10
		2	4000	5-10	5-10	5-10
			5000			5-10
			6000		5-10	5-10
			7000		15-20	10-15
		3	4000		0-5	
			5000		5-10	
			6000		5-10	0-5
			7000		10-15	
		4	4000	10-15	5-10	10-15
			5000	10-15	5-10	0-5
			6000	5-10	5-10	5-10
			7000	0-5	20-25	15-20
		5	4000	5-10	10-15	
			5000	5-10	5-10	
			6000	5-10	10-15	
			7000	5-10		
	0.3	1	4000	10-15		10-15
			5000	15-20	10-15	15-20
			6000		10-15	10-15
			7000		15-20	
		2	4000	15-20		
			5000	15-20		0-5
			6000	10-15		
			7000			
		3	4000	15-20	0-5	
			5000	0-5	0-5	
			6000	5-10		
			7000			
		4	4000		0-5	5-10
			5000		0-5	
			6000		0-5	5-10
			7000		0-5	5-10
		5	4000	0-5		0-5
			5000	5-10		0-5
			6000	0-5		
			7000			

Table 6.4c N Networks And Curve-Fitting Grades

consistent and stable in the sectors where the `Tolerance` value is `0.2`, although there are pockets of fairly reasonable return (%) in the `0.3` sector. Amongst the three configuration functions, the `RuleD` function is capable of a better performance: networks with σ equal to `1` and `4` produce consistent and good return (%). However, the conclusion for the overall best network between them is divided.

Configurations: `RuleS`, `RuleD` **Or** `RuleT`

The analysis of the network generalisation capabilities offers evidence to augment and eliminate indecisive contenders for the optimum network which have been judged solely by the prediction accuracy of the network. Unlike the evidence from prediction accuracy, the contours of the weighted return (%) are more defined and, therefore, the decision line more clearly drawn.

The sN1 model favours a `RuleD` function, with σ equal to `4` and a `Tolerance` of `0.3`, whilst the sN2 model favours a network with a `RuleS` function using, σ equal to `2` and a `Tolerance` of `0.2`. The N model favours networks with a `RuleD` function using σ equal to `1` or `4`, which is, indecisive.

6.2.4 Optimum Network

An optimum network should be capable of consistent accurate predictions with stable credit winnings. The consistency in prediction accuracy judges the test correlation (%) over the 50% threshold across the time window of evaluation, all the four 1000 training cycles. The stability in credit worthiness judges, across the time window, the corresponding weighted return (%) for those predictions with greater than 50% test correlation.

The evaluation applies the **RTPI** indicator. The **RTPI** superimposes the weighted return (%) on its test correlation (%), eliminates those predictions below the threshold percentage accuracy level, and highlights those with a relatively good prediction accuracy - in relation to the expectation of the threshold percentage and the percentage accuracy of the other training cycles. The explanation for the indicator is in the Notations section of Chapter 5. The indicator is plotted against the test correlation (%) and the weighted return (%) for model sN1 in Appendix F (Figure 6.5a to 6.5e), model sN2 in Appendix G (Figure 6.6a to 6.6e), and model N in Appendix H (Figure 6.7a to 6.7e).

Model sN1

From the analysis of the prediction accuracy for model sN1, the networks of choice are, a `RuleD` or `RuleT` functions with a number of possible input layer sizes; from the analysis of generalisation, it is a `RuleD` function with σ equal to `4`. As it is difficult to draw any definition conclusion, the **RTPI** is used to assist in making a decision.

The charts in Appendix F gives a cross-section of the forecasting capabilities of the five input layer sizes, configured against the RuleS, RuleD and RuleT configuration functions. The **RTPI**, in particular, eliminates those networks forecasting below the 50% accuracy threshold but giving a positive-valued weight return (%). The outcome of the **RTPI** filter is five candidate networks vying for the position as the optimum network of model sN1, shown in Figure 6.8.

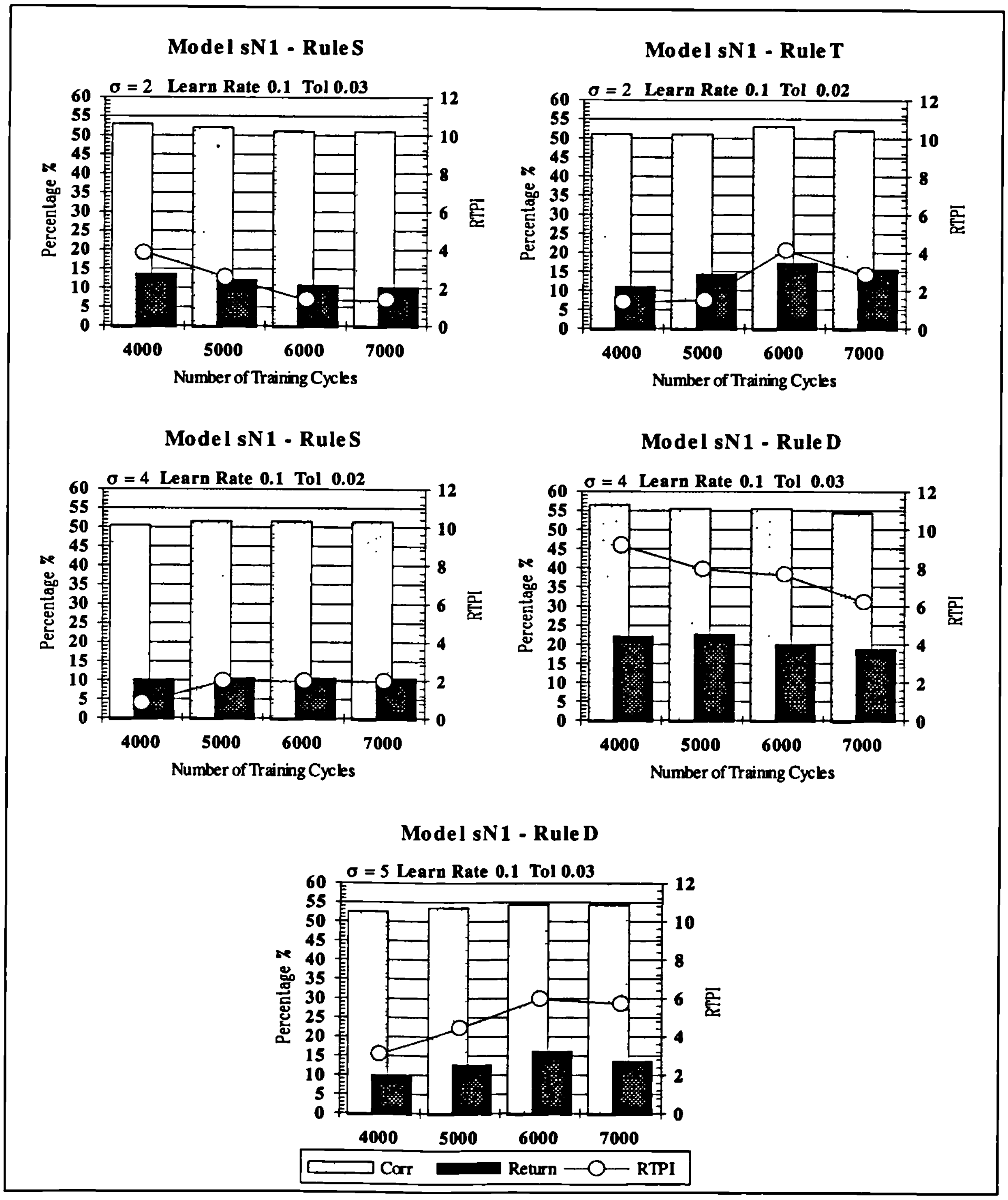


Figure 6.8 Candidates For Optimum Network Of Model sN1

The performance of the candidate networks show that model sN1 forecasts with a rolling test correlation (%) landscape, between 50% to 55%, with a corresponding weighted return (%) not exceeding 25%, averaging between 10% and 15%.

The outstanding network has a RuleD function using σ equal to 4, and offered a prediction accuracy and generalisation at the top end of their respective ranges.

Model sN2

A suitable sN2 network was according, to prediction accuracy, a RuleS function with σ equal to 2 or 4, and to generalisation, a RuleS function with σ equal to 2. The charts in Appendix G, with the help of the **RTPI**, justify the conclusions. The candidate networks are as shown in Figure 6.9.

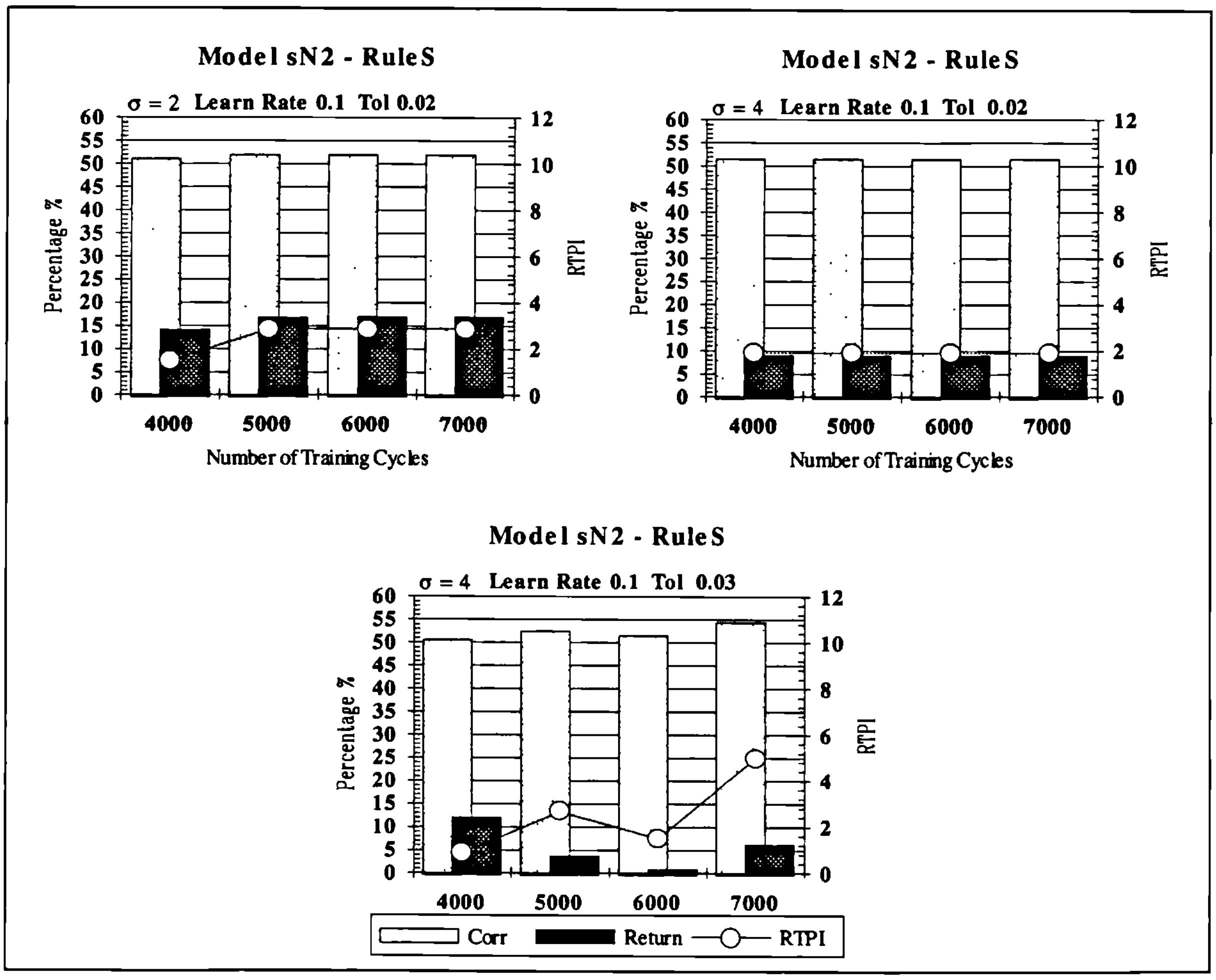


Figure 6.9 Candidates For Optimum Network Of Model sN2

Like the sN1 networks, these networks are also capable of prediction accuracy between the 50% to 55% range but they usually border on the lower end of the scale. In terms of generalisation capabilities, they also faired marginally below the standard of sN1. They ranged from between 10% to 15%, to between 0% to 5%.

Of the candidates, the network using a `RuleS` function with σ equal to `2` triumphs.

Model N

The conclusion for an optimum network for model N was by prediction accuracy, `RuleD` function using, σ equal to 4 and a `Tolerance` of `0.2`; and generalisation, again a `RuleD` function using, σ equal to 1 or 4, and a `Tolerance` of `0.2`. Like the selection of model sN*x*, the **RTPI** are charted alongside the test correlation (%) and weighted return (%) as shown in Figure 6.10.

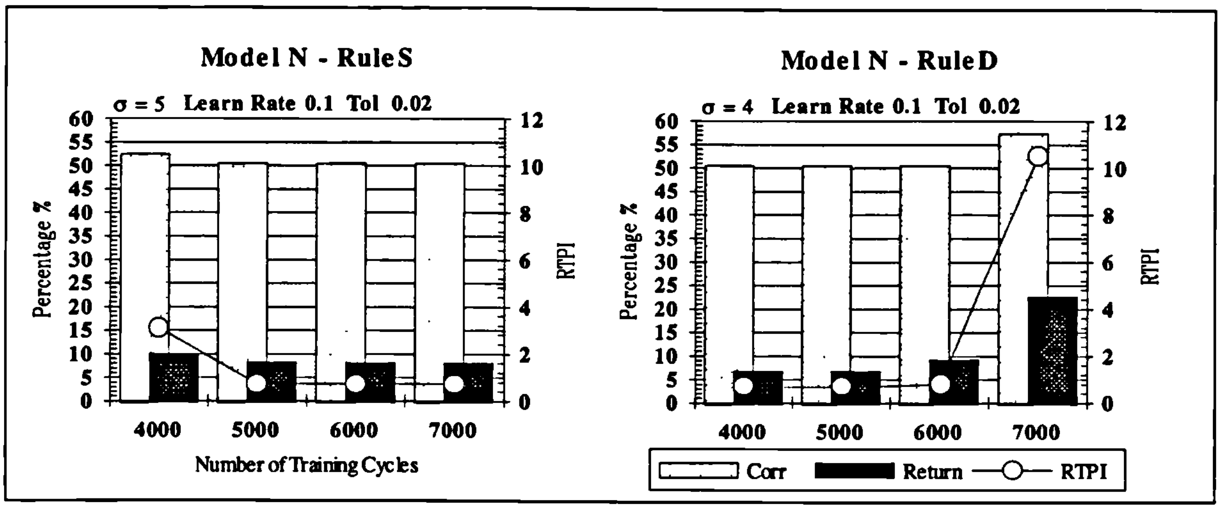


Figure 6.10 Candidates For Optimal Network Of Model N

The forecasting performances of the N networks, contrasting with those of model sN1 and sN2, are more defined. That is, there are a number of networks which do not forecast well at all and have consequently been abandoned. Furthermore, there are more clusters of those with poor prediction accuracy and generalisation. The contenders for optimum network are considered below.

The forecasting performance of the candidate networks show the prediction accuracy to lie between 50% to a maximum of 57%, and returns averaging from 5% to 10%, with a maximum at 22%.

The decision of the outstanding network is difficult to draw, as there is a choice to trade off network size for a marginally larger return. The network with `RuleD` function using σ equal to `4` is chosen because a `RuleD` rather than `RuleS` function is more suitable for a *consolidation* trend.

Model sN*x* And Model N

The final selection procedure compares-the representative networks from each model. The charts of the representative networks of models sN1 and sN2 show unanimously that the network of the former is the optimum for model sN*x*. The optimum performing network for models sN*x* and N are summarised in Figure 6.11.

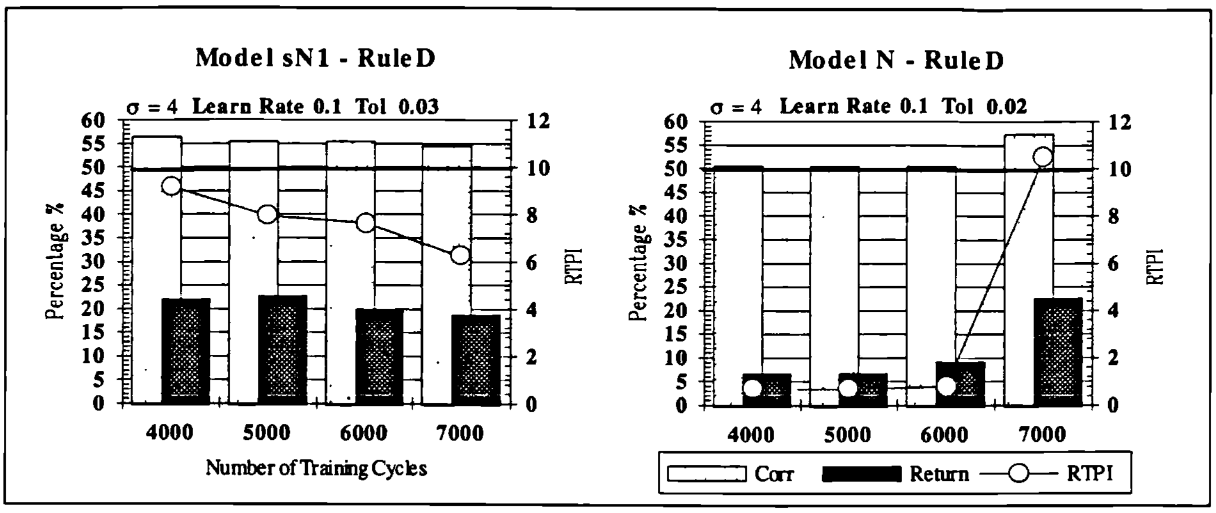


Figure 6.11 Performances Of Optimum Networks Selected For Models sN*x* And N

The first difference between the two optimum networks is the marginally better prediction accuracy percentages of the sN1 network. Its accuracy borders along the 55% margin, while model N borders along the 50% margin with the exception of its last cycle at 57%. The second, and more important, is the better generalisation capabilities of the sN1 network. It has consistent and better weighted return (%) at around 20%, while model N has between 5% to 10%, with the exception of the last cycle, at 22%.

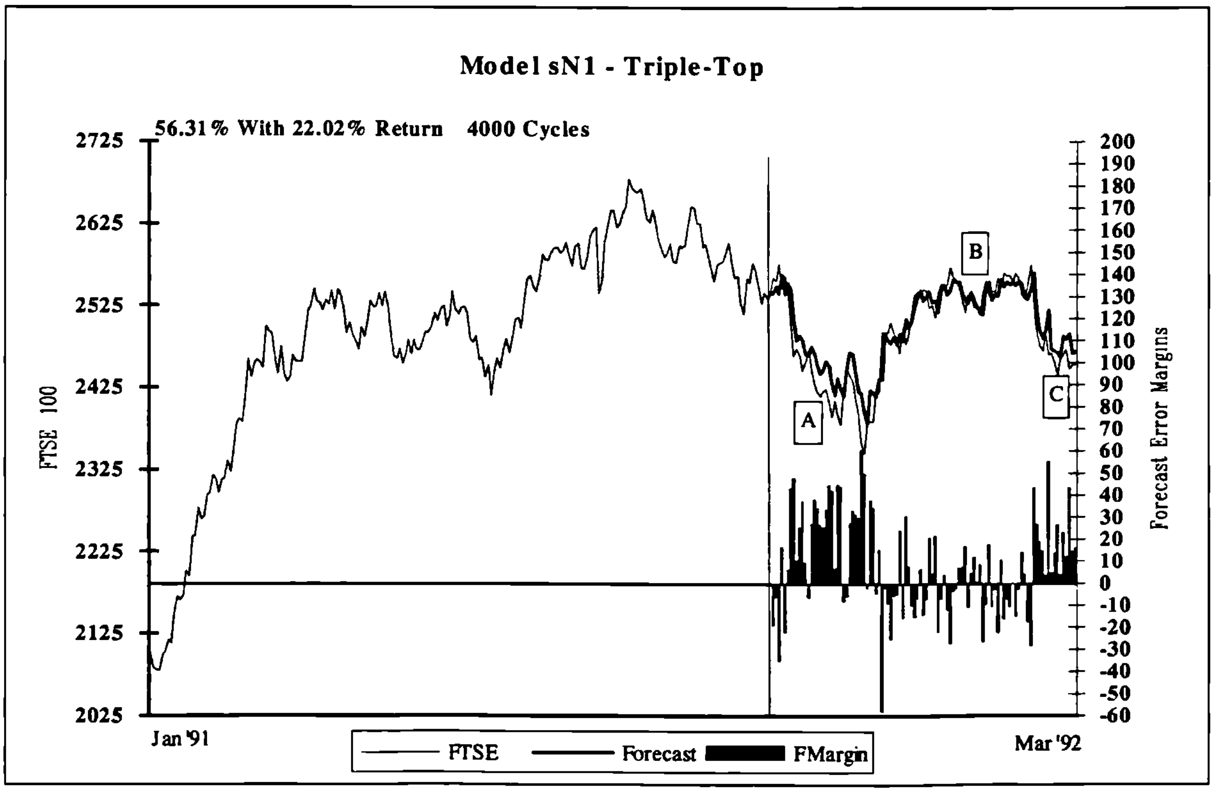


Figure 6.12a Forecasts On *Triple-Top* Pattern By Optimum sN1 Network

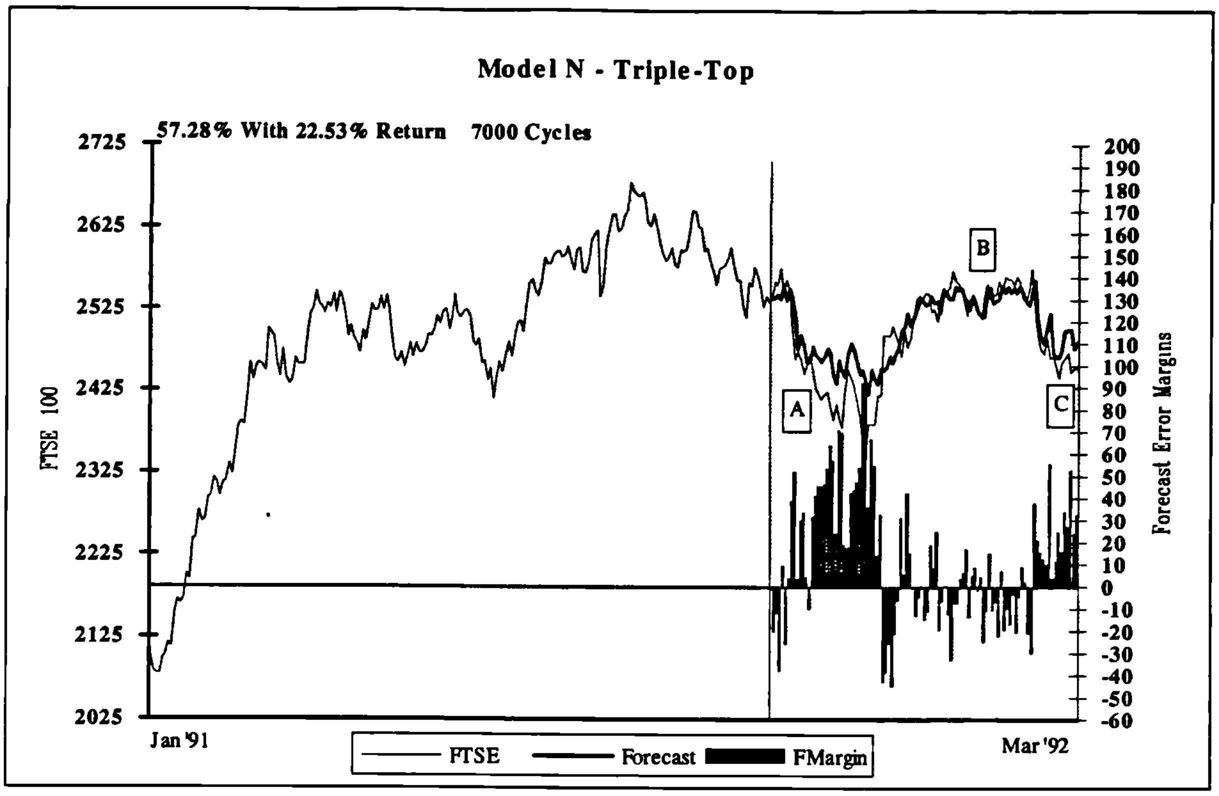


Figure 6.12b Forecasts On *Triple-Top* Pattern By Optimum N Network

The forecasts of the optimum networks, Figures 6.12a and 6.12b, show their capabilities to forecast the contours covering regions A, B and C. Both the models have especially forecast the last "top" remarkably well. Their fundamental difference is the slightly larger error margins of the N network, especially in regions A and C.

In region A, the sN1 network is shown to have a better ability to forecast the dip prior to the formation of the third "top" compared to the N network. Apart from smaller error margins, the error at the lowest point is about 70 Index points for the sN1 network but it is about 95 points for the N network. In region B, both the networks handle the "dome" of the "top" with almost identical curve fit, reducing the errors considerably. When the pattern reaches region C, the error trend increases but the sN1 network tended to have lesser error margins compared to the N network. A review of the error margins across the three regions suggests that both the networks are capable of forecasts within a narrow range of values, similar to those of the first and second "top".

In summary, the optimum network for model sN*x*, sN1, has a rolling test correlation landscape, consistently rolling above 55% prediction accuracy with a similar weighted return landscape with a minimum of 18.72% return. By contrast, the correlation and weighted return (%) landscape of model N faired less consistently by about 5% for accuracy and wider return margins. Finally, the best performing sN1 network, despite being 1% less accurate has an almost similar weighted return (%), and delivers by far better predictions with smaller error margins.

6.3 Adaptation

The strategy to test the usability of the optimised networks in other non-linear time-series periods uses specific observational pattern features of a *double-top*, *narrow band*, *bull* and *recovery* patterns for evaluation. In the main, the strategy is to optimise networks for each of the models on a *triple-top* pattern. Following the selection of a representative optimal network for each model, they are used on new data sets which have test patterns that are either similar or dissimilar from the training patterns.

6.3.1 Similar Pattern Trend

The notion of similar refers to the almost identical pattern forms on both sides of the vertical axis dividing the train and test data sets. The pattern forms selected refers to those formed in a *consolidation* trend.

The choice of a *double-top* pattern tests the optimal networks' capability to accommodate a data form that is almost identical to the *triple-top*, apart from a larger horizontal extension of each of the "top" formation.

On the contrary, the choice of a *narrow band* pattern tests the optimal networks' capability to accommodate a completely different pattern that does not have a specific pattern form. The data oscillates between a tight band and it breaks out from the band to reach an all time new upper bound.

Double-Top

The performance of forecasts on the *double-top* pattern, in spite of its similarity to the *triple-top* pattern, is poor but not altogether unexpected. The results merely reflect that the optimised network no longer applies in a new time period in a non-linear time-series.

The charts of the prediction accuracy, the return and the **RTPI** over the 4*1000 training cycles (right of Figure 6.13) show that it is debatable as to which optimised networks has a better forecasting performance. The test correlation and weighted return landscapes of the two networks are almost similar. Between them, model sN1 has the worst accuracy at 42.31% and the best at 50% and, also the extremes of return (%) from -11.05% to 15.18%.

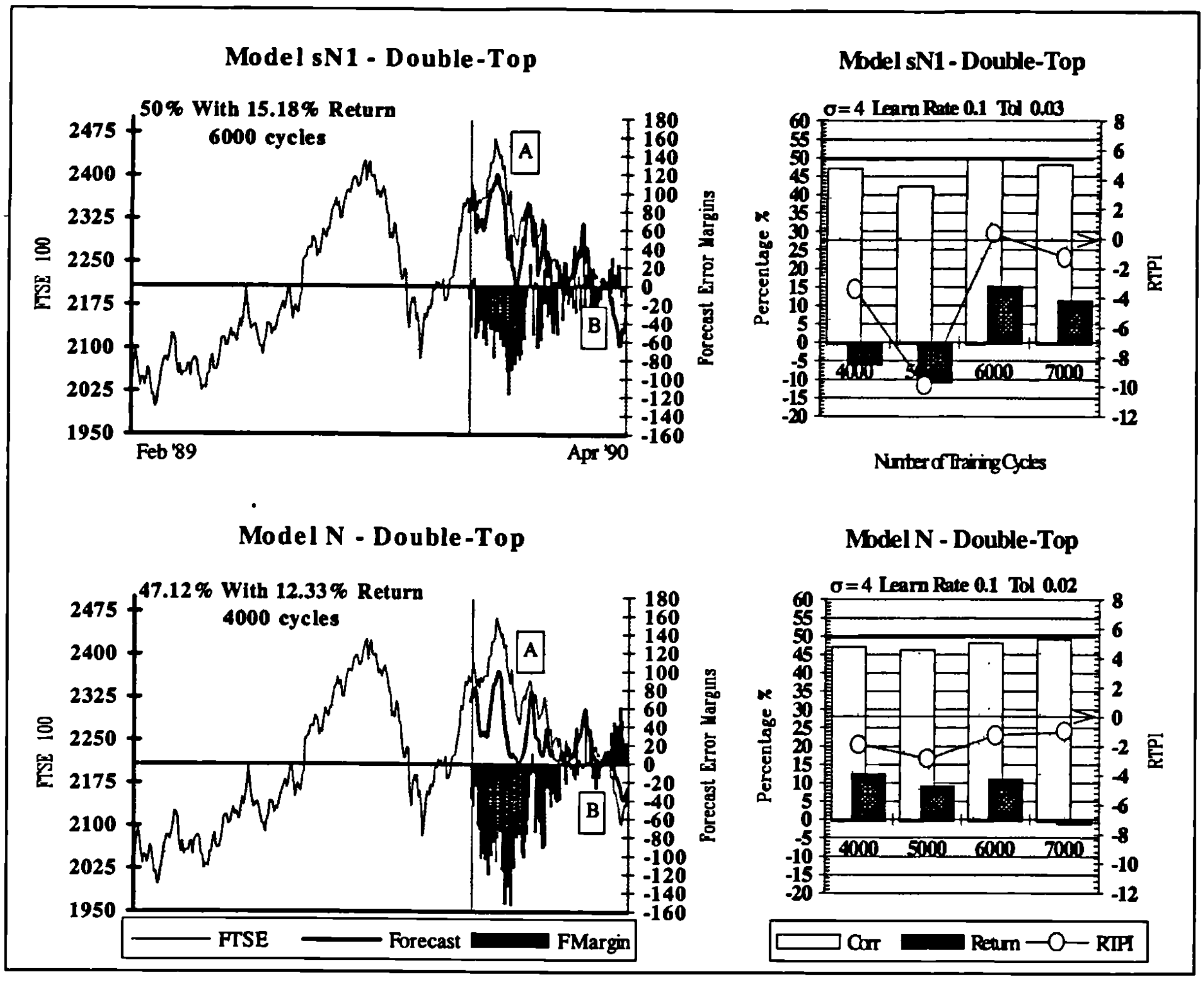


Figure 6.13 Optimum Networks Forecasting On *Double-Top* Pattern

An examination of the forecast error margins shows that the best of model sN1 has smaller error margins and the forecasts tend to fit the tracking Index value better, given that it is about 2.5% more accurate than the N network and has a slightly better return (%). The differences in forecast error margins is particularly obvious at the "top", marked A and in the downslide, at B. In region A, model sN1 has a maximum error margin of -120 Index points compared to -160 points for model N. The significant error margins suggests that both models cannot forecast well beyond the maximum value used by the trained network. Following from the "top", the forecasts recovers and they follow the contours of the test patterns, reasonably well with a reduction in the error margins. When the pattern reaches its lower bound at region B, the error margins becomes positive. This shows that the forecasts do not accentuate the curve on a downslide. The margins of the sN1 network at this region are comparatively small, with a maximum of about 30 Index points, implying a better curve-fit. By contrast, those in network N are between two to three times larger.

In summary, the optimised network do not appear to perform as well in a new time-period even though the data set is quite similar to that used for optimisation. The best performing network, a balance between the prediction accuracy and the weighted return (the network with the best **RTPI** value), from the two models has the sN1 network produce a marginally better prediction accuracy, smaller forecast error margins, and it has performed with a smaller error margins in those test data points which are out of the range of the train data.

Narrow Band

The forecasting performance on the *narrow band* pattern with a breakout, falls below the 50% threshold level. The performance consolidate the fact that the optimised networks do not adapt other types of patterns or new time periods well.

The charts of the prediction accuracy, the return (%) and the **RTPI** (to the right of Figure 6.14) show that the performance of the two networks are equally unacceptable. Their test correlation and weighted return landscapes are quite similar, although model sN1 is marginally better on both accounts.

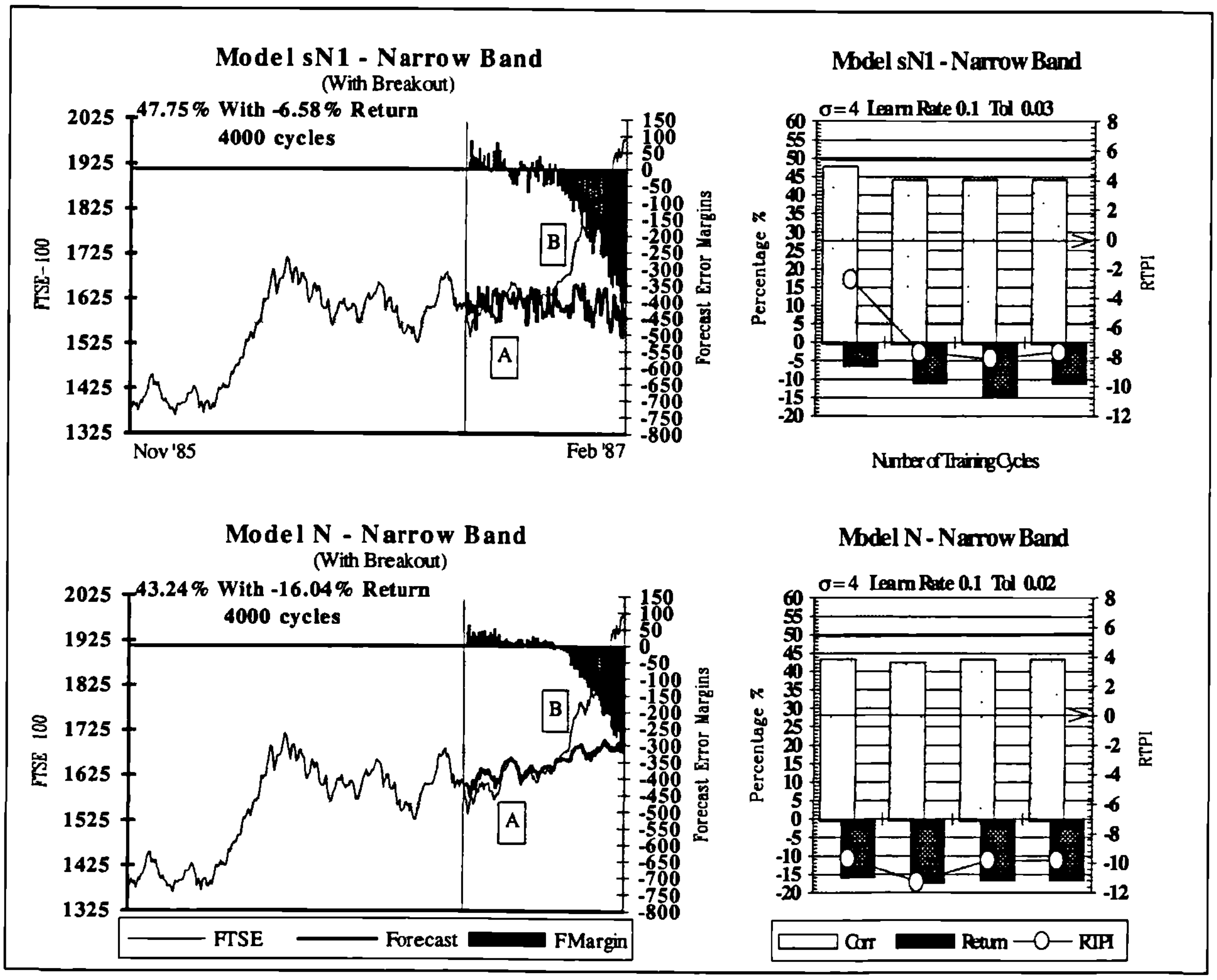


Figure 6.14 Optimum Networks Forecasting On *Narrow Band* Pattern

An examination of the forecast error margins, however, shows that both the networks tend to fit the tracking Index value relatively well in the narrow band region, marked A, but deteriorates rapidly from the breakout point in region B. Despite the overall poor performance, the best of model sN1 forecasts better based on the percentage of prediction accuracy. It is about 4.5% more accurate than the N network and it has a better return (%) as well.

The differences in error margin are particularly obvious at A and B. In region A, the sN1 network has larger error margins and oscillates rather than tracing the contours which the N network has done very well. As the trend pattern progresses, the sN1 network continues to forecast with the narrow range of values, thereby magnifying the error margins beyond those of the N network. At the maximum point, the sN1 network reaches an error of almost 425 points while the N network only manages to reach 300 points.

In summary, the optimised networks, especially network sN1, do not adapt to the *breakout*, forecasting with large error margins. Even then, the sN1 network has a slightly better prediction accuracy and generalises better. This shows that the quality of the forecasts are difficult to judge by the prediction accuracy and the generalisation measurements.

6.3.2 Dissimilar Pattern Trend

The notion of dissimilar refers to the non-symmetrical pattern forms on both sides of the vertical axis dividing the train and test data sets. The *bull* and *recovery* trend patterns selected are not formed during a *consolidation* trend.

The choice of a *bull* pattern tests the optimised networks' ability to adapt to a data form that is completely different to that being optimised and which has a completely different train and test pattern. In addition, it frequently tests the upper bound of the train data set as the rising trend reduces its momentum.

The second choice of a *recovery* pattern tests the optimum networks' ability to adapt to a completely different pattern where a part of the data trend is repeated in the test data set but the pattern is formed with a different range of data values.

Bull

The forecasting results on the *bull* pattern are the best recorded from all the experiments carried out. The best prediction accuracy for the sN1 and N networks are 56.73% and 54.81% respectively. The results are better than those for patterns formed in the *consolidation* trend. It can imply two sets of explanation. The first is simply that the optimised networks adapt to this new trend pattern. Alternatively, it cannot but because the rising pattern is easy to train and therefore, the results delivered for this particular pattern are not good, in relative terms that is, and could be much better.

The charts of the prediction performance (to the right of Figure 6.15) show that the performance of the two networks are equally identical and favourable, trundling along with 55% accuracy, with the exception of the 6000 cycle. However, the weighted return landscapes differ: the sN1 network consistently has positive values but the N network fluctuates between positive and negative values.

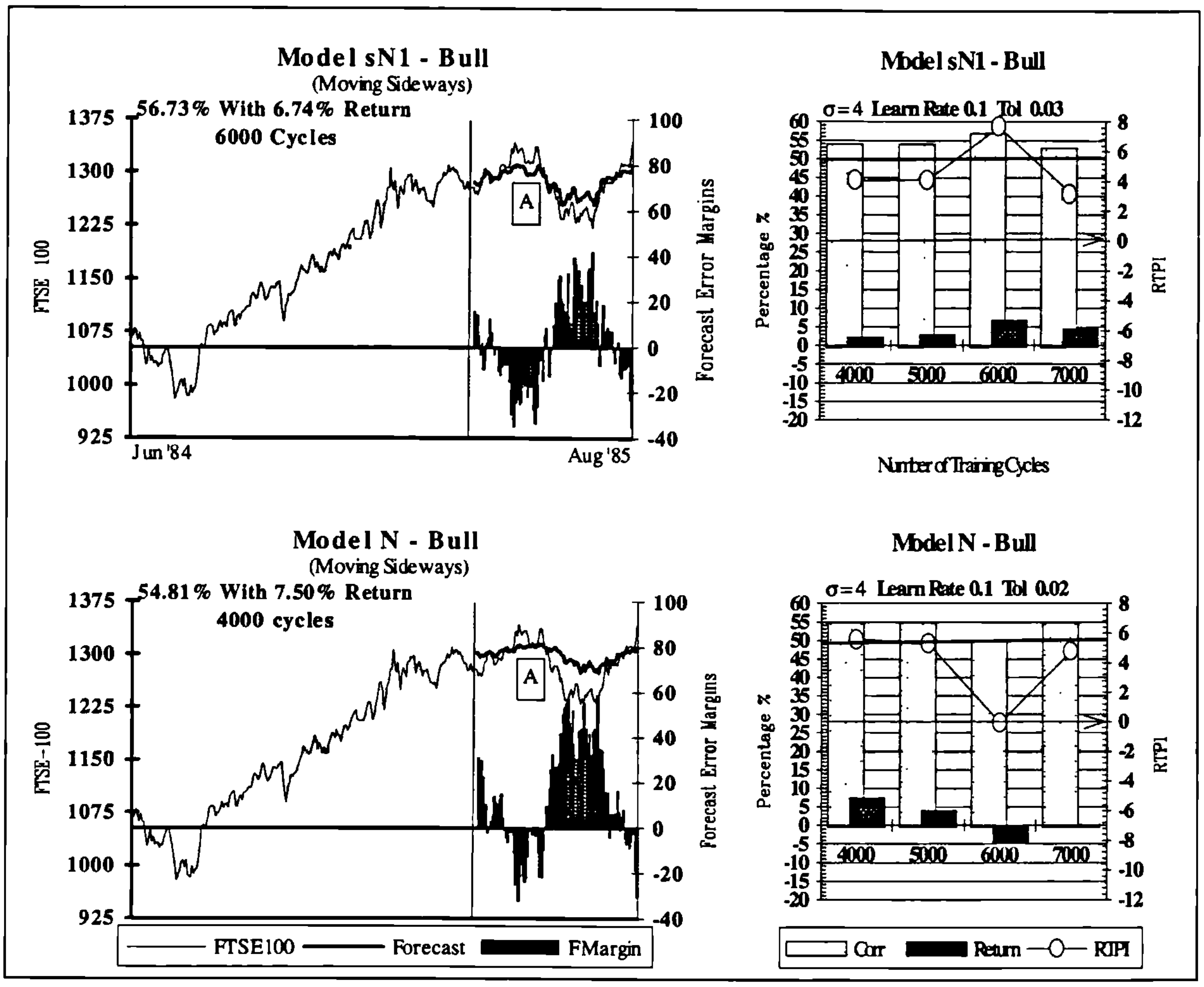


Figure 6.15 Optimum Networks Forecasting On *Bull* Trend Pattern

The charts of the forecast error margins re-iterate that a trained network cannot forecast above the maximum value to which it had been introduced during training. This is a likely explanation for the large error margins on the small "top" in region A. In addition, it shows that networks, especially the sN1 network, can adapt a different test pattern fairly well if it is formed within a range of values of the training data set. In doing so, the sN1 network picks out intricate pattern details better than the N network. This is highlighted by the lesser error margins to the left and right of region A.

In summary, the optimised network of both the models appear to forecast a test pattern which is different from that used for training reasonably well. The sN1 network has a slightly better prediction accuracy and consistent weighted return (%), and it has adapted the test data pattern better by not forecasting values that over simplify the data trend.

Although the prediction accuracy of N network is about 2% smaller than the sN1 network, it has a larger weighted return per percentage accuracy. In addition, the forecasts of the N network again showed that the quality of a network's forecast is difficult to determine from the performance results.

Recovery

The performance on the *recovery* pattern is consistent at 50%, with a positive return for all test recordings. Like the *bull* pattern, it can be interpreted that the optimised network can adapt to time or data changes or otherwise.

The prediction performance (to the right of Figure 6.16) show that both the models perform equably along the 50% threshold with minor exceptions in the N network. The weighted return landscape did not differ either: both the networks have consistent positive returns with the sN1 network always offering a slightly higher return.

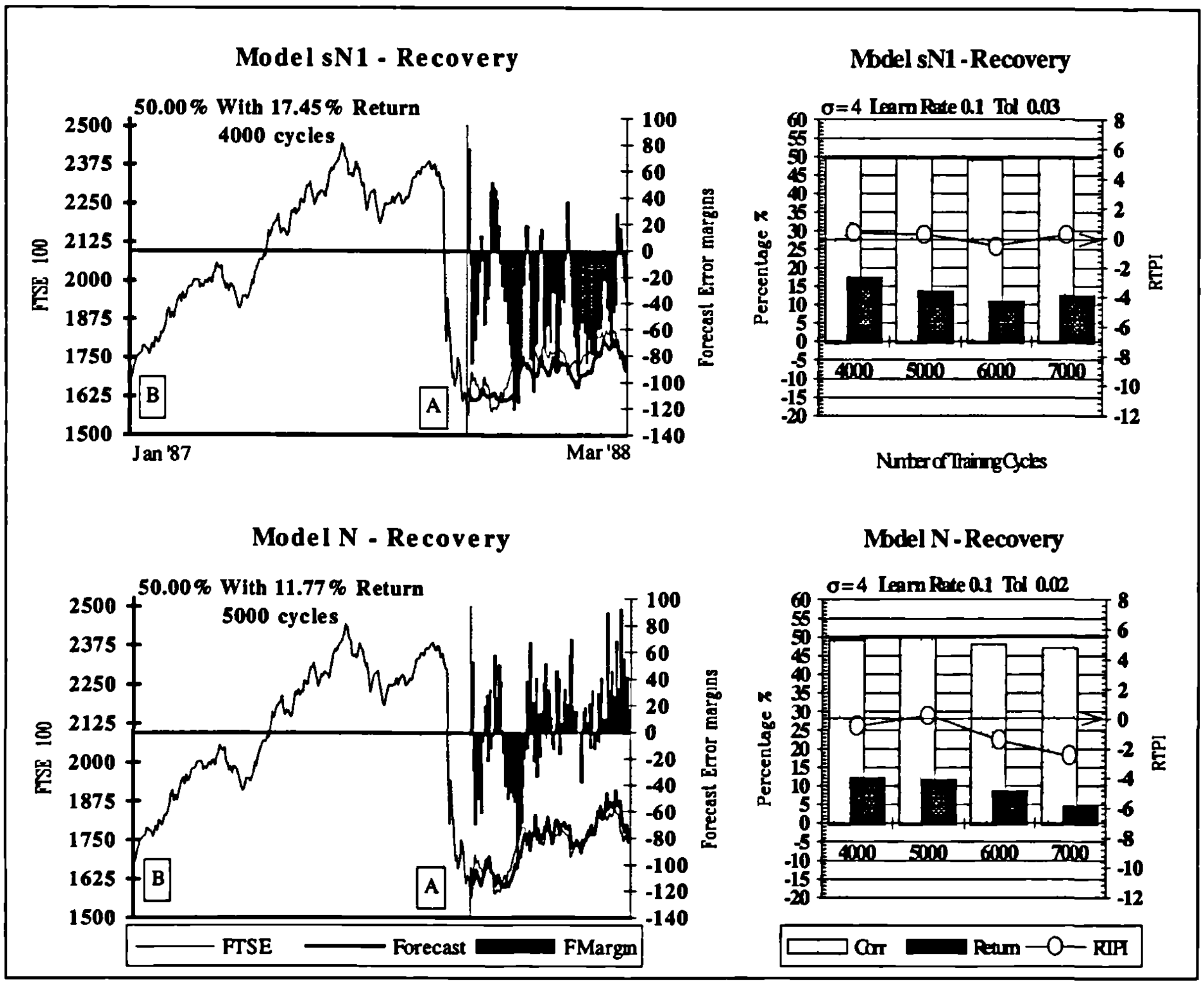


Figure 6.16 Optimum Networks Forecasting On *Recovery* Trend Pattern

The charts of the forecast error margins show that the predictions from both the models, in particular the sN1 network, have large error margins. The negative error values suggest that the forecast do not accentuate the tracking values. The N network, on the other hand, has a better set of forecasts. A possible explanation for the behaviour of model sN1 is that it is unable to offer a forecast value, beyond region A, which it has not seen before in the pattern in region B, even though the data trend is quite similar and the forecasts are within the range of the train data set. This is a contrast to the case of the *bull* pattern which appear to adapt to a new test pattern formed within the train pattern values.

In summary, the ability of the optimised network to adapt to a new data trend is again open to interpretation. In spite of the sN1 networks forecasting with appalling error margins, it is in terms of performance indicators, marginally better than the N network by its better weighted return (%).

6.4 Summary

This chapter reports time-series forecasting by models sNx and N. The experimental strategy was to optimise networks of both the models and to apply them to different data sets which have a similar or dissimilar training and test data pattern formations, and the test data values are within or outside the range of the training set. The optimisation procedures select a suitable raw input data form, prediction accuracy measurement and configuration function, and the better of model sNx, sN1 or sN2.

A *triple-top* pattern formed on a *consolidation* trend was used for optimisation. The optimised networks were applied to the *double-top* and *narrow band consolidation* trend patterns, and the *bull* and *recovery* data trends.

The evaluation compares and contrasts the performance of the two models against several criterion, such as a consistency in prediction accuracy, the network generalisation, a combination of both as the **RTPI** indicator, and the forecast error margins. The **RTPI** is a cost function developed to distinguish networks that meet the required minimum prediction accuracy percentage and a creditable network generalisation.

The first selection eliminates *logarithmic* inputs to **NFS** configured networks due to the fact that they produced forecasts resembling a flattened pattern trend with extremely large error margins. The conclusion justifies the data form selected, the *linear* (absolute value) form, to be applied to investigate the input representations in models sNx and N. This is because both the input models are based on the transformation of raw *linear* data.

The second selection picks the measurement of prediction accuracy based on the correlation of the forecast for tomorrow with today's tracking value, VCorr, as it consistently delivered the best set of test correlation percentages. This rating bettered the measurement based on the correlation of the direction of movement of the forecast with that of the tracking data, DCorr, and a combination of the previous two definitions, VDCorr. In all cases, with a few exceptions for the DCorr definition, the analyses of the corresponding weighted return (%) for each test correlation (%) are similarly ordered, reinforcing the choice for the vCorr method of measurement.

The third selection picks the RuleD configuration function as the most suitable network configuration for the *triple-top* pattern. The function relates the hidden layer size as twice the input layer size. The evaluation of a combination of test correlation (%) and weighted return (%) performance in conjunction with the **RTPI** indicator values showed the RuleD function delivered, overall, more consistent and good forecasts than the RuleS or RuleT functions.

The fourth selection picks the sN1 version of enhancing the input pattern values to the neural network as the better to represent model sNx. The conclusion uses observations from the *triple-top* pattern. The optimum network was based on a combination of evaluation results from the earlier selection procedures. An optimised sN1 network has the input layer size, σ equal to 4 using a RuleD function and a Tolerance of 0.03. By a similar method of selection, the optimised N network has a network configuration which is similar to the sN1 model, but uses a Tolerance of 0.02.

The performance of the optimised networks showed the input representation method in the sN1 network forecasts the *triple-top* pattern better than the N network. Within the evaluation window, between 4000 to 7000 training cycles, the test correlation (%) and weighted return (%) landscapes are consistently above 55% and 15% respectively. Based on these two criterion, the forecasts from N network are consistently less accurate, have lesser return (%) and larger forecast error margins.

The application of the optimised networks to test their abilities to adapt to changes to the data pattern and time period showed the following behaviour.

(i) neural networks do not adapt well to new data patterns or data trends.

(ii) the prediction accuracy of a neural network on its own gives insufficient or inaccurate indication of the forecasting ability of the network.

(iii) the weighted return (%) provides an important additional information indicating the trained network's ability to forecast the major data movements of the test data set.

(iv) the prediction accuracy and the network generalisation do not indicate the quality of the forecast value.

(v) the confidence of a forecast is not known.

(vi) a forecast value does not have an intrinsic value on its own.

(vii) neural networks predict with large error margins on test data points which lies outside the values on which the network has been trained.

Chapter 7

Assessment

This chapter assesses the feasibility of neural forecasting, reviewing their effectiveness to deliver consistently good results and as a usable tool. First, it examines the RTPI indicator to justify its achievements and relevance. Thereafter, it examines the measurement of network generalisation and the relationship between network learning and extrapolation. Next, it examines the usability of neural forecasts to form investment strategies, and, finally, it compares model sNx with related work and other forecasting tools.

7.1 Targets Review

The theme of this thesis has been to investigate time-series neural forecasting of the direction of the FTSE-100 Index on a Backpropagation neural model. In the main, it has studied a new method, as a group of models in sN*x*, to enhance the input representations to networks alongside a commonly used method in model N.

To achieve the target aim, experiments were carried out on the customised Neural Forecasting System (**NFS**), developed to facilitate the research. The experimentation was organised into a number of cascading test procedures, such that each has specific objectives for specific research results. The research results and contributions accomplished are assessed to finalise a conclusion to this work on whether neural networks are a viable tool for non-linear time-series forecasting.

To underline the support for neural networks, the judgements are also made with reference to other non-neural forecasting tools (Chapter 2 gives a comprehensive survey of existing practices). A neural forecasting system, such as the **NFS** has to satisfy logistical criterion. It has to have a potential:

- to deliver *consistent* and *good* forecasts and,
- to be *usable.*

The first criteria requires neural forecasts to be comparable to conventional methods, not necessarily the best, and consistently tolerating changes to time and data. The second criteria requires the forecasting tool to be usable in a dynamic non-linear application environment.

This review assesses the following practical considerations for adopting neural forecasting a time-series.

(a) *The Relative Threshold Prediction Index (RTPI) indicator.*
It justifies its achievements and development, showing its ability to differentiate various borderline cases of network forecasting performance and generalisation abilities.

(b) *The measurements of network generalisation.*
It highlights finding the relationship between the global minimum error and the forecasting performance, and finding the relationship of network interpolation and extrapolation.

(c) *The task of forecasting and the act of using a forecast.*
It discriminates the gap between proofs of forecasting potential from a fixed evaluation window and the successful use of neural forecasts to form investment strategies for use on a dynamic time-series.

(d) *The position of neural forecasting with model sNx.*
It compares the model with related work and existing traditional forecasting tools.

The conclusion on the suitability of neural networks as a forecasting tool are examined on the behaviour of the neural models in the experiments carried out.

7.2 The RTPI Indicator

The **RTPI** indicates the extrapolation calibre of a trained network in terms of the prediction accuracy and the generalisation of the network for a fixed test data period.

The indicator is a cost function, equation (5.5) in Chapter 5,

$$RTPI = \frac{R}{C}\|C - TP\| + (C - TP) + \frac{R}{C}$$

where C is the test correlation (%), R the weighted return (%) and TP the threshold prediction percentage. As the TP has been assigned a constant value of 50%, the assessment of the **RTPI** indicator considers example forecast performance with combinations of C and R. The discussion refers to these tuple in the format, (C:R) **RTPI**.

(1) Case of test correlation (%) *less than* the threshold level.

(i) with positive valued *R*.

The first example covers networks with the same correlation percentages. The tuple, (45.19% : 6.58%) -4.66 and (45.19% : 3.81%) -4.73 (Table 7.1) from model N and sN2 highlight that networks forecasting below the 50% expectation have negative valued **RTPI**, and the values themselves can reflect the better network based on its network generalisation (the return (%)). Between the two networks, the sN2 network has a smaller return (%) which implies that it has a lesser ability to pick out the major movements of the test data set. This is consequently reflected by its smaller **RTPI** value compared to that of the N network.

Model	Cycle	VCorrelation (Train %	Test %)	VReturn (Train %	Test %)	RTPI
N	4000	79.52	45.19	87.25	6.58	-4.66
	7000	66.83	48.54	63.20	1.23	-1.40
sN1	5000	70.00	45.19	73.00	-3.01	-4.88
	6000	59.33	49.51	44.01	12.60	-0.11
sN2	5000	67.14	45.19	64.64	3.81	-4.73
	4000	74.64	41.75	79.16	-5.42	-9.45

Table 7.1 **RTPI** And Accuracy *Below* The 50% Threshold

The second example covers networks with different correlation percentages. The tuples, (45.19% : 6.58%) -4.66, and (48.54% : 1.23%) -1.4 from model N show that they are below expectations but even then, the better performing network is represented by a higher **RTPI** value.

(ii) with negative valued *R*.

The first example covers networks with the same correlation percentages. The tuples, (45.19% : 6.58%) -4.66 and (45.19% : -3.01%) -4.88 from models N and sN1 indicate that networks which are capable of the same percentage of accuracy are differentiated by their return percentages. For example, the sN1 network with a **RTPI** value of -4.88 has a debit return (%) while a similar N network with a **RTPI** value of -4.66 has a credit return (%).

The second example covers networks with different correlation percentages. The tuples, (45.19% : -3.01%) -4.88 and (41.75% : -5.42%) -9.45 from models sN1 and sN2 indicate that networks with different accuracy capability, even though below the threshold are appropriately represented. The **RTPI** value for the sN2 network represents a worst correlation and return percentages compared to those of the sN1 network.

In addition, a combination of positive and negative return(%), such as the tuples, (45.19% : -3.01%) -4.88, (41.75% : -5.42%) -9.45, (45.19% : 6.58%) -4.66, and (48.54% : 1.23%) -1.4 reflect through their **RTPI** values, the order of the networks' forecasting capability.

(2) **Case of test correlation (%) *equal to* the threshold level.**

(i) **with positive valued *R*.**

The capability of a network forecasting to the minimum required percentage accuracy is indicated by its generalisation capability. The tuples from the first row of each of the models (Table 7.2) with correlation(%) of 50%, show that their **RTPI** values are ordered in the same manner as the return(%): the highest at 0.23 for a return(%) of 11.27%, followed closely by 0.2 for 9.8% and 0.03 for 1.43%.

Model	Cycle	VCorrelation (Train %	Test %)	VReturn (Train %	Test %)	RTPI
N	7000	71.29	50.00	72.59	11.27	0.23
	6000	73.33	50.00	72.85	-4.77	-0.10
sN1	6000	72.86	50.00	76.83	1.43	0.03
sN2	7000	71.77	50.00	75.55	9.80	0.20

Table 7.2 **RTPI** And Accuracy *Equal To* The 50% Threshold

(ii) **with negative valued *R*.**

This particular case has been encountered only once in the experiments. The tuple is shown in the second row of model N in Table 7.2. It has a debit return (%) and it is reflected by the negative valued **RTPI**.

In so far as both the positive and negative return (%) cases are combined, their respective **RTPI** remain ordered, reflecting the networks' forecasting performance.

(3) **Case of test correlation (%) *greater than* the threshold level.**

(i) **with positive valued *R*.**

The first case covers networks forecasting with the same correlation percentages has, (50.49% : 6.71%) 0.69 and (50.49% : 8.10%) 0.73 from models N and sN2, and (51.92% : 3.88%) 2.14 and (51.92% : 16.72%) 2.24 from model sN1 and sN2. They show for these examples, the latter tuple which has a higher return(%) has a larger **RTPI** value. Another example is, (50.49% : 6.71%) 0.69 and (50.49% : -2.26%) 0.42 from models N and sN1. Each of the network has an **RTPI** value which also highlights its return (%) potential.

The second case covers networks with different correlation percentages. An example set of tuples are, (54.81% : 19.69%) 6.9, (51.92% : 3.88%) 2.14, and (52.88% : 18.42%) 4.23, from the first row of each of the models in Table 7.3. The order of the **RTPI** values, 6.9, 4.23, and 2.14 represent the order of forecasting accuracy, 54.18%, 52.88% and 51.92%.

Model	Cycle	VCorrelation (Train %	Test %)	VReturn (Train %	Test %)	RTPI
N	4000	60.95	54.81	45.16	19.69	6.90
	4000	70.19	50.49	69.89	6.71	0.69
	4000	62.98	51.46	55.65	-2.22	1.35
sN1	6000	69.52	51.92	69.87	3.88	2.14
	4000	71.29	50.49	72.98	-2.26	0.42
sN2	4000	56.94	52.88	38.46	18.42	4.23
	5000	68.90	51.92	66.61	16.72	2.24
	7000	67.46	50.49	69.22	8.10	0.73
	5000	67.15	50.49	56.18	-0.02	0.49

Table 7.3 **RTPI** And Accuracy *Above* The 50% Threshold

(ii) with negative valued *R*.

The first case covers those networks with negative valued return (%) for correlation (%) above the threshold level, regardless of their percentage value. The tuples are, (50.49% : -2.26%) 0.42 and (50.49% : -0.02%) 0.49, and (51.46% : -2.22%) 1.35, (50.49% : -2.26%) 0.42. These example cases of trained networks would normally be discarded given that the status of the network generalisations are considerably poor. In such circumstances, it would be appropriate for the **RTPI** to be a negative indicator, like those discussed in the first case. As the **RTPI** values in the examples are positive valued, they give spurious indications to the forecasting capability of the networks.

In spite of this weakness, these four **RTPI** indicator values have, however, not affected the selection decisions in this thesis and its effect has been small. The cases, (50.49% : 6.71%) 0.69 and (50.49% : -2.26%) 0.42 from model N and model sN1, discussed for Table 7.3, is an example of a disadvantage that is almost insignificant. One reason is, these cases, which are few and isolated, have occurred in correlation percentages which are close to the threshold level. At the most, they have been confusing and have under-estimated and over-shadowed the performance of those networks forecasting near or at the threshold level, such as those discussed for the second case. A second reason is that its occurrence is likely to be even less frequent in non-*consolidation* pattern trends. A third reason is that the threshold level used has been conservative and, if a higher expectation is used, which it should be, this problem would become even less possible.

In summary, assessments of cases of the **RTPI** over conditional combinations of the two performance parameters secure its validity as a pointer to the forecasting calibre of a network. There are two weaknesses to the method of calculation.

The first, highlighted in correlation (%) above 50% but having negative valued return (%) is not usually relevant in the selection of a high forecasting calibre network, but it has, of course, to be rectified. The second is the small weight placed on the weighted return in the equation (5.5). As a result, it is obvious, from the set of correlation : return percentages discussed for all the cases, that they reflect the generalisation for equable test correlation(%) but not those which are not. For example, the **RTPI** values for, (50.49% : 6.71%) 0.69 and (51.92% : 3.88%) 2.14 do not reflect the first pair of values as better than the second pair. As it is, equation (5.5) has to be improved to add more weight to the generalisation capability, but only if, the relative importance between the two criteria is addressed further.

7.3 Network Generalisation

Neural learning maps an input pattern to a desired target pattern and, more importantly, it encodes the salient features of all the input-output training pattern pairs. The procedure discovering the complex network internal representations is referred to as the generalisation problem. When the network eventually generalises, the learning ceases and convergence is said to have occurred.

The generalisation problem is a solution of curve-fitting. If f is the function of the curve representing the training samples and f' is the mapping function intersecting f, then the relationship, $f \approx f'$ holds when a network has generalised successfully. The key to a successful curve-fit is a balance of a minimal number of input nodes and the correct number of network connections. These translate to the values of σ and the configuration function.

The network's convergence is, on the other hand, solved by the dynamics of the learning process. This references the learning procedure, such as described for the Backpropagation model (in Chapter 3). The efficiency in achieving convergence is a combination of η the gain term to search for a global minimum distance between the curve data points and a hyper plane, to satisfy a minimum threshold distance, the tolerance level, and the number of training iterations.

7.3.1 Global Minimum Error

The solution to the curve-fitting problem is network optimisation. It is a procedure to select a suitable input layer size and configuration function which together allow the best global minimum to be retained. In the application of neural networks to non-linear data, the optimum parameter values are optimised for a specific data pattern. The optimum status could possibly falter when other data patterns are used.

The optimisation procedure selected a suitable set of parameters for the *triple-top* data pattern. The experiments eliminate contenders from different combinations of the number of input nodes and the number of hidden nodes. Throughout the optimisation process, the global minimum monitors the average mean square error (network error) of the training network. It was also monitored by the forecast performance: the test correlation (%) and the weighted return (%). A sample of these measurements are shown in Table 7.4.

The measurements in the four tables show that there is no relationship between the correlation (%) for the training data set and the test data set. The example of the *triple-top* pattern which records the optimised N network show that a test correlation (%) of 50.49% could be produced from a network trained to fit the curve to a train correlation (%) of 70.19%, 67.79% or 69.23%. In each of these cases, the average mean square error are 0.0421, 0.044 and 0.0465. As a result, their respective weighted return (%) differs even

Model	Cycle	VCorrelation		VReturn		Network Error	
		(Train %	Test %	(Train %	Test %	(Avg.	Max.)
N	4000	70.19	**50.49**	69.89	6.71	0.0429	0.1177
	5000	67.79	**50.49**	68.33	6.71	0.0440	0.1147
	6000	69.23	**50.49**	70.00	9.11	0.0465	0.1284
	7000	63.94	57.28	63.97	22.53	0.0456	0.1342
sN1	4000	69.23	56.31	63.15	22.02	0.0590	0.1595
	5000	68.75	55.34	67.98	22.66	0.0393	0.1226
	6000	70.19	55.34	63.20	19.99	0.0542	0.1479
	7000	71.15	54.37	74.41	18.72	0.0372	0.1135

(a) *Triple-Top* Pattern

Model	Cycle	VCorrelation		VReturn		Network Error	
		(Train %	Test %	(Train %	Test %	(Avg.	Max.)
N	4000	55.71	47.12	47.87	12.33	0.1339	0.3919
	5000	55.71	46.15	27.46	9.21	0.1856	0.5458
	6000	67.14	48.08	68.09	11.01	0.0814	0.2278
	7000	70.95	49.04	73.64	-1.41	0.0738	0.2550
sN1	4000	70.48	47.12	71.22	-6.31	0.0634	0.2440
	5000	72.38	42.31	75.53	-11.05	0.0652	0.2461
	6000	72.38	50.00	75.00	15.18	0.0563	0.2131
	7000	75.24	48.08	79.61	11.37	0.0515	0.1817

(b) *Double-Top* Pattern

Model	Cycle	VCorrelation ·		VReturn		Network Error	
		(Train %	Test %	(Train %	Test %	(Avg.	Max.)
N	4000	71.43	**54.81**	75.94	7.50	0.0335	0.0977
	5000	70.95	**54.81**	74.84	4.02	0.0593	0.1513
	6000	73.33	50.00	72.85	-4.77	0.0514	0.1534
	7000	72.86	**54.81**	76.80	0.13	0.0564	0.1460
sN1	4000	70.48	53.85	71.10	2.24	0.0493	0.1442
	5000	70.48	53.85	71.19	3.02	0.0497	0.1443
	6000	70.00	56.73	71.97	6.74	0.0372	0.1074
	7000	69.52	52.88	64.50	4.58	0.0532	0.1608

(c) *Bull* Pattern

Model	Cycle	VCorrelation		VReturn		Network Error	
		(Train %	Test %	(Train %	Test %	(Avg.	Max.)
N	4000	60.95	49.04	51.33	12.26	0.0738	0.2999
	5000	54.29	50.00	36.86	11.77	0.0132	0.3812
	6000	58.10	48.08	46.99	8.59	0.0773	0.2609
	7000	60.95	47.12	51.75	4.52	0.0717	0.2930
sN1	4000	65.71	**50.00**	62.10	17.45	0.0557	0.2433
	5000	59.05	**50.00**	42.93	13.66	0.1018	0.3379
	6000	55.71	49.04	41.92	10.87	0.1092	0.3258
	7000	50.48	**50.00**	30.64	12.45	0.1236	0.3345

(d) *Recovery* Pattern

Table 7.4 Average Network Errors And Forecast Performance

though the test correlation (%) are the same. This trend is repeated by the sN1 network on the *recovery* pattern.

In the *triple-top* pattern, the sN1 network has for 4000 and 5000 cycles, errors at 0.059 and 0.0393 with a weighted return (%) of 22.02% and 22.66%, which is a rather large error difference for a negligible return percentage. This feature is also seen in the other three examples.

The forecasting performance of the *triple-top* pattern show that the best performing network does not have a global minimum error. It may also apply to all the other patterns, although it is not proven since the results of the other patterns are not optimised but applied applications of an optimised network. This presents a problem in assessing the network generalisation of non-linear time-series. A popular method [Lake93] embeds a look ahead technique in the training procedure. It involves, testing the training network regularly to find the network error for the test data set, intersect the network errors from the train and test data sets and use the error at the point of intersection as the chosen global minimum. The method is reasonable for historical simulations. It can, however, be problematic in practise.

Firstly, a look ahead technique implies an ideal world with a set of test data. It is applicable for historical simulations, but it is difficult to apply for live tests. This is because there is no test data as such and, even if a portion of the historical data is used, it is not a proper solution for non-linear data.

Secondly, the minimum network error could depend on the test data pattern and the size of the non-linear test data. As a result, the global minimum deduced may not be the desired global minimum after all.

In part, the argument justifies the use of a combination of the test correlation and the weighted return for the evaluation of forecasting networks, and consequently, the relevance of the **RTPI** indicator. On the other, the argument highlights it being a problem in non-linear neural forecasting.

7.3.2 Neural Learning And Extrapolation

Neural processing is programmed to learn a sample data set by discovering and encoding knowledge about features of the data. Neural forecasting exploits neural processing to program the knowledge and forecasts new data based on the premise that the future data is a repetition of the past data trend.

In practical terms, a neural network learns a past trend by curve-fitting during network training and forecasts by extrapolating the curve. The performance of curve-fitting, as it was said, are measured by the test correlation (%) and the weighted return (%). The relationship of these percentages for the training and test data sets are shown in Table 6.1 (Appendix B) and Table 7.4.

The sample from both tables illustrate two important facts. The first is that a reasonably good train correlation (%) has no bearing on the degree of goodness of the test correlation (%). The second is that the same train correlation percentage value does not necessarily give the same test correlation percentage value. The test correlation percentage value is dependent on the parameters of the convergence problem: the size of the network, the number of training cycles and the tolerance value used.

The apparent lack of a coherent relationship between network interpolation and extrapolation, an extension of the difficulty measuring network generalisation, triggers a questions about the measurement of network learning and extrapolation of a trained network. Haley and Soloway [HalSol92] showed in their studies that a trained network can extrapolate but it is limited. In addition, Sarle [Sarle94] showed that the extrapolation of a sine curve is satisfactory but it can be improved at the cost of a slightly worse interpolation. From the point of view of this thesis, the extrapolation is dependent on the training and test patterns and the expectation of the life-span of the network. They are discussed in the following section.

7.4 Forecasting And Using A Forecast

Forecasting is a subject applied to a practical domain and there are an abundance of forecasting tools using, as it was described in the survey, diverse techniques. Consequently, introducing neural networks as an addition to the list justifies debate on its benefits. The question explores the characteristic of neural forecasts and its effect on making strategies aimed at maximising gains with low risks.

7.4.1 Evaluation Window: Static vs Dynamic

Neural networks have, in the laboratory, demonstrated great potential to forecast beyond expectations, especially beyond those of traditional forecasting methods. The potential is, it must be stressed, reflected by measurements made under laboratory conditions.

Neural forecasting uses time-slices of input data and extrapolates one-time slice of data for the future. The method of neural extrapolation forces the evaluation of network performance be made over a fixed and static evaluation window: simulation results are measured over an unspecified test data set which has a fixed and unspecified data size. A network which has a prediction accuracy of 56%, for example, implies 56% of the predictions are correct up to the last time-slice of test data. By contrast, the test data pattern, or rather, financial data evolve dynamically and non-linearly. As a result, the percentage value can sometimes not reflect accurately the actual forecasting potential. In a lot of cases, the percentage depends on the size of the evaluation window and the type of test pattern formation. This is illustrated by a history of the seemingly poor forecasts, in Table 7.5, of the optimum networks on the *double-top* pattern in Figure 6.13.

Model	Cycle	10 days	20 days	30 days	40 days	50 days	60 days	70 days	80 days	90 days	100 days	103 days
N	4000	**40.00%**	**35.00%**	**46.66%**	**45.00%**	**46.00%**	**46.66%**	45.71%	45.00%	44.44%	48.00%	47.12%
sN1	6000	**40.00%**	**35.00%**	**46.66%**	**45.00%**	**46.00%**	**46.66%**	47.12%	47.50%	46.66%	48.00%	50.00%
N	7000	50.00%	40.00%	50.00%	50.00%	48.00%	50.00%	51.43%	51.25%	50.00%	47.00%	49.04%
sN1	5000	40.00%	35.00%	46.66%	47.50%	42.00%	41.66%	41.43%	42.50%	43.33%	41.00%	42.31%

Table 7.5 Prediction Accuracy History Of *Double-Top* Pattern

The first two rows show the history of test correlation (%) for networks with the best weighted return (%) and the second pair, for the worst return (%). In both cases, they show that the prediction accuracy varies throughout the different evaluation windows. The networks with the best weighted return (%) have the same prediction accuracy for the first 60 days before they diverge. It showed that the forecast performance measured is dependent on the evaluation window and evaluation results are equally affected.

If the networks are judged solely by their prediction accuracy, that is, the second and third row, the N network has better percentages throughout all the evaluation windows. This particular N network has a high accuracy but also a high risk. It is because with its weighted return(%) at -1.41%, it is not good at picking out the major data movements.

The measurement of forecasting performance on non-linear data is not easy and neither is the assessment of the forecasting potential. The assessment should rightly reflect performance in a dynamic rather than a static evaluation window. A static window implies that the potential is as good as the size of the evaluation window. These problems also enforce the importance of supporting the prediction accuracy by the other forms of measurement, such as the weighted return (%) and the use of the **RTPI** indicator.

7.4.2 Investment Strategy Formation

A forecasting tool evaluates data making forecasts for sound investment strategies (the Chapter 2). A neural forecasting tool makes forecasts of time-series data on a time-slice basis. The future part of a one-step forecast consists of a window with one time-slice data, and the window moves along with time as the time-series evolves. For each forecast, a prediction either correlates or does not correlate with the movement of the data from a value today to a new value tomorrow.

It has been pointed out, for historical simulations, that the correlation of each forecast evaluates each prediction value at a time, but the prediction accuracy evaluates all the predictions together for a specific evaluation window size. The graphs in Figure 7.1 and 7.2 are typical examples of neural simulation outputs and results.

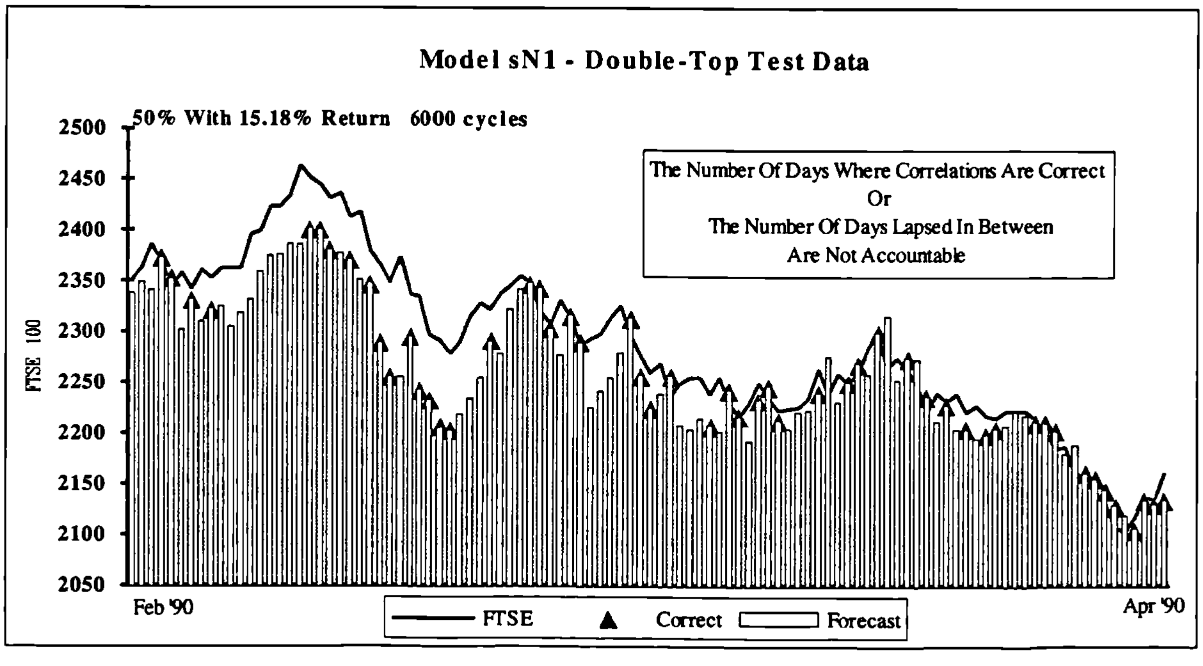


Figure 7.1 Forecasts And Correct Correlations On Test Data Of *Double-Top* Pattern

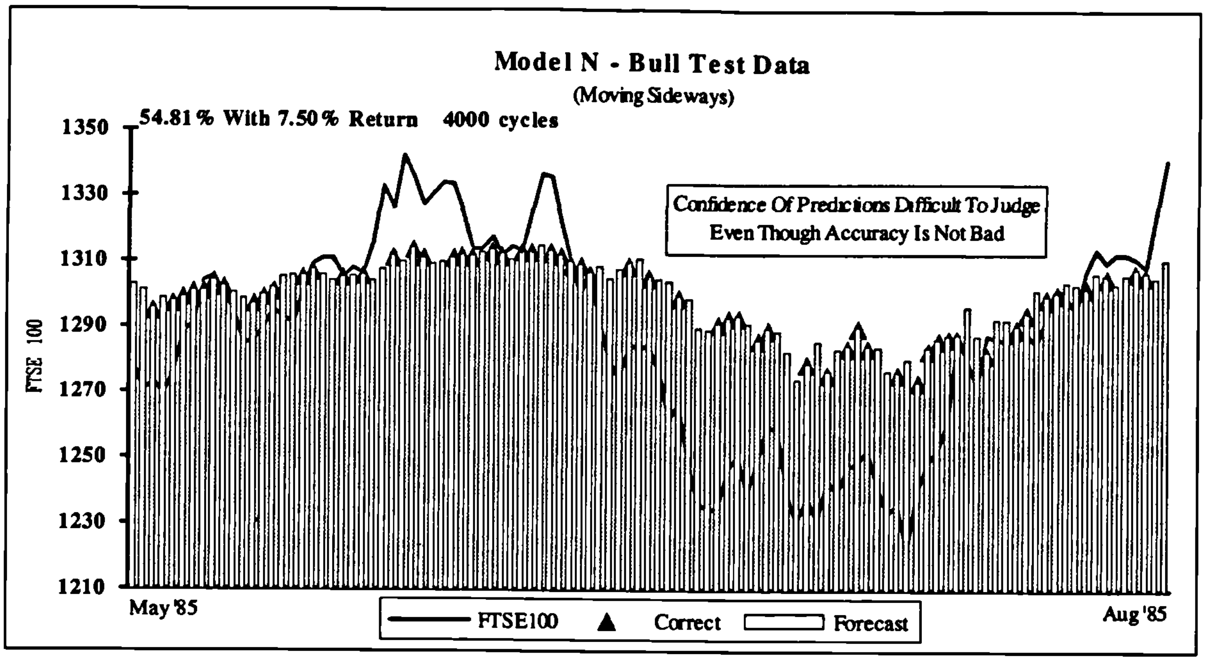


Figure 7.2 Forecasts And Correct Correlations On Test Data Of *Bull* Pattern

The graphs illustrate common neural forecasting features. The first is the random manner in which a forecast correlates correctly: the number of consecutive days it correlates or the number of days lapsed before it correlates again are unaccountable. The second is the relatively good percentage of accuracy for the *bull* pattern, even though the forecast error margins are large. This shows that a prediction value by itself does not contain much information.

Let us assume a trained network capable of forecasting with accuracy beyond expectations and which is known to have a good generalisation capability. Following a forecast, the prediction is used to make an investment strategy. A prudent decision is, however, difficult to come by from a prediction which is merely an isolated figure. By the example of the *bull* pattern, a coherent strategy is difficult to make as the intrinsic value of a forecast figure is difficult to judge, without investing in chance or instinct, to guess the confidence of the prediction. The difference between making guesses on predictions from a good performing network and a lesser one is the difference in the confidence to guess whether a forecast correlates or not. This, to some extent, defies the purpose of using a forecasting tool.

A neural prediction on the direction of the data movement lacks intra-time-series relationship. The prediction does not provide information on its bearing relationship with the rest of the data in the time-series. This isolated piece of information could be supplemented with endogenous information. The minimum knowledge required is the confidence of the forecast. It is a piece of information that is not readily measured in neural networks. Other forms of information can be used. Amongst the examples are the percentage of data movement made and the trend of the movements as defined by the DCorr prediction accuracy measurement.

It has been pointed out that it is difficult to assess the extrapolation capabilities of neural networks on non-linear data. The occasions when the confidence of a network's prediction is more assured, is when the data has a *bull* or a *bear* trend. Unlike a *consolidation* trend, these two occasions can raise a user's confidence in guessing the direction of the forecasts because the underlying direction of the data is known. Unfortunately, the experiments on particularly, the *double-top*, *bull* and *narrow band* patterns show that those regions (region A in Figure 6.13 and 6.15, and region B of Figure 6.14 respectively) of the test data set with data values which are out of the range of those in the train data set do not perform well. The poor network extrapolation beyond the range of the squashed outputs, (out-of-range problem) are limitations pointed out by White [White88] and lately, there have been reports on its limitation for extrapolation [HolSol92, HusHor93].

A neural forecasting system has to ensure that it will not falter when it encounters the out-of-range problem in a *bull* or a *bear* data trend, otherwise, neural forecasting will be confined for use on specific trend patterns. The solution to the problem is a hierarchy of networks, each of which is responsible for different time grains providing support whenever an out-of-range problem is encountered.

The problems highlighted concludes that neural forecasting can be viable in the real world, only if, there are endogenous information available to support using predictions to form investment strategies. A forecast value has to be enriched to be useful. As it is, the risk is high. In addition, neural forecasting has to be able to handle the out-of-range problem when the future value of the time-series evolve beyond the highest high or lowest low of the training data.

7.5 A Forecasting Tool

Following the reports of good neural forecasting potential, and the satisfactory performance on tested on a *triple-top* pattern, the next important question is whether a neural forecasting system has an edge over existing forecasting tools. This question relates to model sNx, related neural systems and other forecasting methods.

7.5.1 Model sNx vs Related Neural Systems

The survey of neural forecasting using real world data showed that the research is mostly approached from neural learning and network architecture. The approach in this thesis concentrates on the input representations to the network instead. As much as it is essential to compare the performance of model sNx with other neural systems, the comparison can be inconsistent and inaccurate. This is due to the uncertainty of neural networks and insufficient details, especially the percentage of accuracy, of these systems. It has been shown by the experiments in this thesis that results could vary due to the data pattern, the method of measuring correlation and the size of the test data horizon. Other influences include the type of data, and whether the prediction accuracy is measured for the test data or the training and test data. These details are, unfortunately, not discussed in reports.

Schöneburg [Schöne90] is one of the few who published details of the research - or rather, exploratory investigations - results. His work is one of the most quoted citing an astounding 90% prediction accuracy. The truth is that the Adaline neural model used offered a 90% accuracy for a miserly 10 days prediction period! More exactly, it was in the region of 60% accuracy with the best of 74% after 2500 training cycles. In fact, it was estimated that for the 10 to 20 days prediction period the average of a total of 300 predictions is 71.3%.

The size of the evaluation window has been shown to be a crucial parameter to the value of the performance results. This thesis presents tests on less ideal patterns where the *double-top*, *narrow band* with a *breakout*, and a *bull* trend *moving sideways* have some data in the test set with out-of-range problem. The history of prediction accuracy for the *double-top* pattern, Table 7.6, show the best at 50%. It is certainly much worse than Schöneburg had achieved. However, the application of the optimised sNx network on the *narrow band* pattern produced 80% and 75% accuracy for 10 and 20 days duration, even though it was 47.75% for the entire 111 days.

Refenes and Azema [RefAze92] have used a Backpropagation model to forecast the Deutschemarks on a data set which has, a *bull* train pattern and a combination of, *bear* and *recovery* test pattern. The test data set has values within the range of the train data set. The optimised network was quoted to achieve 68% accuracy for 60 days into the future. This same pattern was not studied in this thesis. The optimised sNx network consistently achieved 50% accuracy for 103 days and around 55% for 60 days duration when applied, not optimised, on the *recovery* pattern. Unlike the Deutschemarks the test data set has Index values which are on the edge of the out-of-range problem, and [White88] have reported that forecasts suffer under such a condition.

7.5.2 Neural Forecasting vs Traditional Forecasting

Traditional forecasting methods in technical analysis are fervently criticised as arcane practices. The strengths of model sNx or even model N are not directly comparable with such methods not because the approaches are arcane but, neural networks forecast and technical analysis methods mostly anticipate.

For example, the traditional Elliot Wave Theory uses a combination of anecdotes, a little geometry and acquired creativity, to "forecast", providing useful information, such as "what levels to get in", "the level to get out", "the level it is likely to turn around is", or "it is likely to reach level X in 3 months, failing that it will test level Y". The Theory offers essential cues to secure calculated chances to form an investment strategy. It does not measure forecasts as an isolated prediction on a time-slice basis and it does not allow accuracy to be measured like neural networks. The endogenous information, which no doubt are well studied conjectures, provides a useful and rather powerful edge that is lacking in neural forecasting. The prediction accuracy is measured after the anticipation has proven to be correct and a profit is made.

The inconsistency and confusion over claims of neural forecasting performance are well summarised in [TanAlm90]. For example, Sharda and Patil [ShaPat90] concluded from their time-series competitions between neural networks and the Box-Jenkins methodology that both techniques have equable abilities. Due to inadequate explanations of the conclusions, Tang, Almeida and Fishwick [TanAlm90] conducted different experiments and concluded that the Box-Jenkins is slightly better for short term forecasting and neural networks are better for long term forecasting.

7.6 Summary

The assessment of neural networks as a viable technique for non-linear time-series forecasting reviews its potential capability to forecast with consistent and good results, and to be a usable tool.

First, I have made an assessment of the **RTPI** indicator, to justify its achievements and development. The indicator filters out networks forecasting above a specified minimum accuracy and have shown to generalise relatively well by the percentages of their weighted return. Following tests on cases of prediction accuracy percentages lying above, at and below the minimum of 50% against positive and negative return percentages, the conclusion was that the indicator can differentiate the desired networks as positive indicator values and the rejects as negative values. The indicator is inaccurate when the prediction accuracy is just above the threshold level and has a negative weighted return (%). The problem was manifest on only four occasions and it does, therefore, not invalidate the evaluation of experimental results.

Second, I have examined the estimation of network generalisation for non-linear data. I underlined the problem of estimating a global minimum to indicate sufficient network learning to enable satisfactory network extrapolation. I concluded that a good fit on the training data does not guarantee good forecasting performance, but a less good curve-fit can. The difficulty is estimating the degree of curve fitting especially for forward simulations.

Third, I have examined forecasting and applying the forecast results in practice. I showed that forecasting performance is evaluated over a fixed data window, but the results are dependent on the size of the window, and in reality the future has a dynamic window. The uncertainty of the forecasting potential is compounded by the fact that the confidence on a forecast is not readily measured. Seemingly, it is necessary to have a strategy before a neural forecast can be used to make a prudent investment strategy.

Finally, I reviewed the prospect of a neural forecasting tool. I drew distinctions between model sNx, and related neural works and other forecasting techniques. I concluded that the uncertainty and the lack of consistent and clear published results makes comparison void. By contrast, one-step forecasts complement anticipation by charting and technical analysis.

The overall conclusion is that a neural network can be a viable forecasting tool. Presently, it is disadvantaged by its sensitivity to the type and patterns of the training and test data, time, the network size, and the number of training cycles, the lack of information about the prediction confidence, the difficulty monitoring network learning and extrapolation. As a result, neural networks can have inconsistent performance and an element of uncertainty on the forecasts persists.

A neural forecasting tool is viable as a complementary tool if it can handle forecast values that are out of the range of the training data set, resolve the lack of confidence of the prediction, provide endogenous information to supplement the prediction, and be less sensitive in order to perform more consistently across data and time changes. A system which does not is limited to a small set of pattern formations and a high risk is involved when the forecasts are used in real-time.

Chapter 8

Conclusion And Future Work

This chapter summarises the research, its contributions and ways to improve the effectiveness of neural forecasting. It begins with a synopsis of the customised neural forecasting system, the choice of experimental data, and the objectives of all the experiments undertaken. This is followed by an appraisal of the research with a presentation of the achievements resolving specific time-series forecasting problems. It ends with suggestions for technical and applied research to suit a range of time scales and interests.

8.1 Summary

The history of financial forecasting is a long list of "tried and tested but does not really work" modelling tools spanning across a spectrum of techniques. On one extreme, there are ad hoc methods relying on the vagaries of human judgements to interpret data formations by anecdotes. On the other extreme, there are mathematical techniques modelling data formations and their evolution with well-defined formulæ. In between these, there are simple mathematical techniques measuring specific data behaviour for use as signals. The monitoring of such volatile non-linear data series had remained unsatisfactory.

A simple neural network can outperform conventional methods, sometimes by orders of magnitude [LapFar87]. Consequent work applied to financial data [ChaMeh92, Schön90, SharPat90] reinforced this initial observation and confirmed rudiments of biological neural processing can forecast financial data with enviable results by normal standards.

The aim of this research has been to investigate non-linear time-series forecasting on the non-linear Backpropagation model. Principally, it has been to study two methods of enhancing the representations of input data to make one-step forecasts of the direction of the FTSE-100 Index. The methods in models sNx and N, reduce raw inputs to a value between 0 and 1. The two versions of the method in model sNx studied, sN1 and sN2, accentuate the structure of the data inputs by different magnitudes. The popularly used method in model N was used as a benchmark for comparisons.

The research commenced from the development of a customised Neural Forecasting System (NFS). The NFS was designed with features to facilitate the research and for future research to other forecasting models, such as econometric modelling. The NFS is a menu driven, and it supports both historical and forward simulations. The first allows a specified neural prototype to be trained and tested on a chosen period of historical data. The second allows a trained network to be applied for forecasting one data period into the horizon for testing with live data.

These forecasting options are supported by system facilities allowing the **NFS** to exist as an independent and complete forecasting tool. They are:

(a) *A data alignment facility.*
It standardises and merges data files with non-uniform recording times and volumes.

(b) *A parameter specification table.*
It allows modifications of neural and system control parameter values for use in historical simulations.

(c) *A pattern specification language.*
It allows specifications of input pattern formation, using one or more time-series data, and loading to a neural network .

(d) *A snapshot facility.*
It re-constructs a partially trained network for continuation or recovery of training, or reconstructed a trained network for forward simulation.

(e) *A log facility.*
It records experimental data for evaluation and information for retrieval.

The specifications of these facilities were documented as four - **Align**, **Specify**, **Train**, **Forecast** - Processes. At the implementation stage, they were translated to five interdependent 'C' - ***Menu***, ***Align***, ***I/O***, ***Load***, and ***Backpropagation*** - Modules.

The ***Menu*** Module offers a menu selection: to align the raw data files, to specify the neural parameters of a prototype network, to perform a historical simulation on the specified prototype network, to perform a one-step forecast on a trained network for tests on live data. The ***Align*** Module standardises data files with irregular data volumes and complicated by different data time stamps, and merges them onto a spreadsheet. The ***I/O*** Module writes images of training network and performance statistics from historical simulations onto specific log files, reads data from the spreadsheet and log files, and displays information for the user. The ***Load*** Module forms input patterns and loads them into the network by specifications for either time-series or econometric forecasting. Finally, the ***Backpropagation*** Module executes the Backpropagation learning routines and, controls the utilities for the snapshot facility, and schedules the procedural executions for historical and forward simulations.

Following its implementation, the **NFS**, which is accompanied by a user manual, has been used to test the validity of models sNx and N for neural forecasting. The experiments involved, establishing an optimal network of each of the models and testing each network on new data sets observing their abilities to adapt to time and pattern changes. The experimental data selected show patterns from *consolidation*, *bull* and *recovery* data trends. The *triple-top* pattern from a *consolidation* trend was chosen for the optimisation procedures because the data values in both the training and test set are within the range of each other. The tests on adaptation used two other pattern formations from the *consolidation* trend, the *double-top* and the *narrow band*. Each of the five experimental patterns were chosen for their specific pattern features (explained in Section 5.12 of Chapter 5).

Prior to the network optimisation, a simple experiment was carried out on model N to select a suitable:

(1) raw data form.
The selection procedure was to eliminate two popular raw data forms, *logarithmic* and *linear* absolute values as inputs to the network. The experimental results certified that absolute values are more suitable for neural learning. The conclusion also justifies the investigations on the representations of linear input data in models sN*x* and N.

Following that, optimisation procedures were carried out to select a suitable:

(2) prediction accuracy measurement.
A forecast is said to be correct, or more specifically to correlate with the movement of the tracking value by interpretations of its value, direction, or value and direction: a forecast correlates by value, VCorr, if the forecast value for tomorrow compared to today's tracking value moves in the same direction as the tracking data's movement from today to tomorrow; by direction, DCorr, if the movement of the forecast from today to tomorrow is the same as the movement of the tracking data in the same period; and by value and direction, VDCorr, if the definitions of VCorr and DCorr are combined.

(3) configuration function.
A configuration function relate the sizes of the input layer, I, to the hidden layer, H, as $H = I\,\phi$. I is described by, $I = sI + \sigma$, where sI is a constant equalled to 5 and I is minimum when σ is assigned to 5. RuleS, RuleD, and RuleT representing the functions with ϕ assigned to 0.9, 2 and 3 were contenders. Each of the functions were cross-validated with σ ranging from 1 to 5.

(4) representative model sN*x*.
The input representations of versions sN1 and sN2 which differ by the magnitude of data accentuation as a data value is reduced to lie between 0 and 1 is selected based on an overall better prediction accuracy and weighted return percentages across the combinations of the above optimisation parameters.

The methods of measuring prediction accuracy deliver in their respective interpretations a percentage value with a corresponding percentage of weighted return. The latter is a grand total of the number of Index points made or lost depending on whether the forecast is correct or wrong. Its value is an indicator of a network's capability to forecast the salient features of the test data correctly. It can also be interpreted as an indirect measurement of the network generalisation. The **RTPI** indicator was developed to combine these two forecast measurements. It is primarily used to filter out those networks that forecast below an expected percentage of prediction accuracy and which have poor network generalisation.

An optimal network, each representing models sNx and N were selected based on the network's consistency to deliver a prediction accuracy of over 50% and a positive return percentage across each of the 1000 training cycles, between 4000 to 7000 cycles evaluation window. The selection was aided by the **RTPI** indicator.

The tests on the optimised networks' adaptation carried out historical simulations using the other sets of experimental data in the same manner: testing the network for every 1000 cycles, again between 4000 to 7000 cycles window, and using the sameset of optimised parameter values.

8.2 Research Contributions

The successful accomplishment of the objectives of the thesis saw among the results and conclusions, research contributions that have a mixture of both technical and practical significance. Primarily, the results have delivered solutions to the following forecasting problems:

(a) *A good method to enhance input data representation.*

The method of reduction input data to within a range of 0 and 1 which at the same time accentuates the data points of the input pattern has shown to deliver a more stable prediction accuracy and weighted return (%) landscapes. In addition, it has also shown to deliver a better percentage accuracy, percentage return and forecast error margins when the optimised network is applied to other patterns of the same trend. The method exhibits smaller prediction accuracy differences and stable performances across a range of network sizes. This feature is a precursor to achieving a forecasting tool that forecasts well and is less sensitive to time and data changes.

(b) *A neural forecasting indicator.*

The **RTPI** indicator is a cost function developed to assist in the identification of networks forecasting above a pre-set minimum prediction accuracy and which have a fairly good network generalisation. The latter is indicated by its corresponding positive-valued weighted return percentage. The indicator value differentiates a good performing network from one that is not so good. It is especially useful when neural forecasting research requires an enormously large number of simulations.

(c) *A solution to forecasting during the "quiet" consolidation trend period.*

The *consolidation* trend is often a period when traditional forecasting methods are not good at forecasting. Typical techniques with this deficiency are, the moving average, the RSI, the Elliot Wave Theory. The research has drawn out the usefulness of neural forecasting, for one data period ahead, to complement conventional techniques which usually anticipate over a larger time horizon. A neural forecast will, of course, be forfeited if the strategy as how the prediction is to be used is not entirely clear.

(d) *A set of improvements for a more effective neural forecasting tool.*

One of the reasons for focusing on patterns of the *consolidation* trend is that this can expose far more the weaknesses of neural forecasting. The appraisal of the research results was made with a practical tone to transcend its limitations as an effective technique in reality. Neural forecasting can be improved to be more effective if it can:

- handle forecasting values out of the range of those used in training the network.
- resolve the lack of confidence of the forecast.
- offer intra-time-series information.
- offer a structured framework for building effective neural forecasting models.

In view of the uncertainties applying the technique, a neural forecasting tool can be an effective tool, even by current standards. Most importantly, it requires a well-understood strategy to formulate an investment strategy which can be used with good effect.

8.3 Future Research

In the course of undertaking the groups of experiments, the miscellany of observations and lessons learnt could be projected into new areas of research to exploit the prospects of neural learning or to improve the efficiencies of neural networks, or even to revolutionise the whole approach to neural forecasting research.

The proposals for future work cover interests of proponents of theoretical and empirical research with a variety of interests and those with short-, medium- or long-term commitments.

Neural Processing

Ostensibly, the first item in the agenda is to repair the out-of-range problems which have been frequently encountered in the experiments. As it could have alarming repercussions in tests with live data, the problem should be addressed within a short- and medium-term research time-scales. The solution in both cases is to incorporate more than one `tick-time` data into the training data set so as to cover different ranges of data values.

In the short-term, the **NFS** could be extended to a prototype comprising a hierarchy of networks. Each network would be responsible for learning data from a different `tick-time` and each is activated only when there is a need to make use of the knowledge that is held in that particular `tick-time`. These network of networks would be managed by a cluster of heuristics whose priority is to ensure that the integrity of the forecasting is maintained whenever an out-of-range problem arises.

In the medium-term, the approach to neural forecasting should be revolutionised by approaching such a time complex application domain through a temporal neural network. Its motivation would be to remove the tedium of the endless amount of network training and maintenance throughout the life-time of a conventional neural forecasting system. Specifically tailored for financial time-series, such a network incorporates time sensitivity into its architecture to remove the need to simulate time-lags and to be able to track intra-day data. Of special significance is an extensional Memory Component which has the responsibility to record the history of the processing elements in every layer of the feed-forward network. Its management would be the responsibility of the Heuristics Component whose principal role would be to ensure that the Memory Component is appropriately used to prevent the system from faltering at the edge of its knowledge (the out-of-range problems) in its capacity to forecast.

Additionally, and running in parallel with the temporal network, there should be research into the relevance of chaos theory in neural forecasting. This is because there are similarities and coincidences amongst those observations of the Efficient Market Hypothesis, the market momentum, the sensitivity of networks to the weights of the network connections and the behaviour of perturbations in chaos theory.

Another important area that could be looked at is the measurement of network generalisation. The usual approach to optimisation and model selection makes use of intensive cross-validation exercises on a fixed set of stationary data. Specifically, this project is to study the relationship of data interpolation and extrapolation to establish a systematic technique to develop an effective neural network structure capable of tracking a dynamic and noisy time-series. Current methods are ad hoc and the underlying element responsible for varying the data pattern are not taken into consideration.

One of the requirements of neural forecasting that could tip the balance of forecasting with greater or lesser success is the magnitude of the training patterns. A useful extension to the study of the physical requirements is to research a function for estimating the dimension of this parameter. It is one of the initial problems encountered in the process of getting the experiments off the ground and which is still carried out with a bit of inspired guesswork. This study would only require a slight modification to the existing **NFS** parameter, `Proportion of data for Training`.

Investments

By the standard of a speculator, the temptation of success is never too strong. The next generation of forecasting system from the **NFS** is a technical trading system. It is one that is hopefully capable of consistent timely recommendations that would eventually aggregate to a healthy balance sheet, devoid of human emotions.

The components for a trading system are a reliable forecasting system, good knowledge elicitation tool(s) for measuring both tangible, and if possible some intangible financial expertise, good speculation strategies, and shrewd money management policies. Overall, neural learning could be used as the forecasting component of the system. Perhaps the most problematic aspect of building such a system is the identification and the measurement of the strategies to be applied to a set of money management policies. In this respect, there is a requirement to quantify expertise which would often involve intangible qualifications and infallible information. An example of extending the horizon of the heuristics used for decision making is the application of neural networks for modelling financial lexicons like "the market is *bull*" (The outline for this proposal is in the Symbolic Reasoning section of Chapter 2). Of course, where numerical analysis is concerned, there is also the option of a symbiotic fusion with well researched mathematical techniques like Fourier transforms, Kalman filters and other mathematical derivatives.

Alternatively, the **NFS** could be tried for a range of applications involving multi-index forecasting. By switching onto the *One-to-Many* input pattern specification, for example, the **NFS** could be used for studying the underlying market trends as in fundamental analysis, using data as diverse as interest rates, macro-economic data, currencies, accounting reports, and derivatives of financial data. Along the same line, the *One-to-One* input pattern specification could be used to replace the unsatisfactory [pp 160 of RudCla88] statistical techniques which have been suggested for the prediction of β as it is described in Modern Portfolio Theory.

The specification language and the robust nature of the system coding can also be exploited. The current version of the **NFS** can accept specifications of a network with an unrestricted number of nodes for each layer. This allows further research into other aspects of forecasting, such as providing endogenous information to supplement the forecast value.

The specification can be adapted to load various types of information of a time-series data to deliver apart from a forecast value, the direction of the move, the open, high, low, and close, to name a few.

A neural network can offer innovation to an endless number of investment problems. An application might not provide a total solution, but it could resolve part of the complex problem to be resolved provided, of course, the exact nature of the problem is carefully identified and the strategy to apply the results is known.

References

[Anders82] M.Anderson
"The Investment Decision: An Analysis Using Verbal Protocols", PhD Dissertation, 1982, Michigan State University, E.Lansing, Michigan.

[Angus89] J.Angus
"On The Connection Between Neural Network Learning And Multivariate Nonlinear Least Squares Estimation", Neural Networks, Vol.1, No.1, 1989, pp 42-46.

[CarGro88] G.Carpenter and S. Grossberg
"The ART Of Adaptive Pattern Recognition By A Self-Organising Neural Network", IEEE Computer, Vol. 21, March 1988, pp 77-88.

[Casdag89] M.Casdagli
"Nonlinear Prediction Of Chaotic Time Series", Physica D, Vol. 35, 1989, pp 335-356.

[ChaMeh92] K.Chakraborty, K.Mehrotra, C.K.Mohan and S.Ranka
"Forecasting The Behaviour Of Multivariate Time Series Using Neural Networks", Neural Networks, Vol. 5, 1992, pp 961-970.

[ColGho88] E.Collins, S.Ghosh and C.Scofield
"An Application Of A Multiple Neural Network Learning System To Emulation Of Mortgage Underwriting Judgements", Proceedings of the 1988 IEEE International Joint Conference on Neural Networks, Vol.2, 1988, San Diego, pp 459-466.

[Deboec92] D.Deboeck
"Pre-processing And Evaluation Of Neural Nets For Trading Stocks", Advanced Technology For Developers, Vol.1, No.2, Aug 1992.

[Duliba91] K.Duliba
"Contrasting Neural Nets With Regression In Predicting Performance", Proceedings of the 24th Hawaii International Conference on System Sciences, Vol.4, 1991, pp 163 170.

[DunFee89] C.Dunis and M.Feeney
"Exchange Rate Forecasting", Woodhead-Faulkner Ltd., 1989, ed. C.Dunis and M. Feeney

[DutShe88] S.Dutta and S.Shekhar
"Bond Rating: A Non-Conservative Application Of Neural Networks", Proceedings of the 1988 IEEE International Joint Conference on Neural Networks, Vol.2, 1988, San Diego, pp 443-450.

[FarSid87] J.D.Farmer and J.J.Sidorwich
"Predicting Chaotic Time Series", Physical Review Letters, Vol 59, No.8, 1987, pp 845-848.

[Fox84] J.Fox
"A Short Account Of Knowledge Engineering", The Knowledge Engineering Review, Vol 1, No.1, 1984, pp 4-14.

[GooKhe92] S.Goonatilake and S.Khebbal
"Intelligent Hybrid Systems", Proceedings of the 1st International Conference on Intelligent Systems, September 1992, Singapore, pp 207-212.

[GroWur90] C. de Groot and D.Wurtz
"Analysis Of Univariate Time Series With Connectionist Nets: A Case Study Of Two Classical Examples", Neural Networks of Statistical and Economic Data, December 10-11, 1990, Dublin, pp 95-112.

[HalSol92] P.Haley and D.Soloway
"Extrapolation Limitations Of Multilayer Feedforward Neural Networks", International Joint Conference on Neural Networks, Vol. 4, June 1992, Baltimore, pp 25-30.

[HanRei89] J.Hanke and A.Reitsch
"Business Forecasting", 3rd. edition, Allyn and Bacon, 1989.

[HerKro91] J.Hertz, A.Krogh and R.Palmer
"Introduction To The Theory Of Neural Computation", Addison Wesley, 1991.

[Hinton87] G.Hinton
"Connectionist Learning Procedures", 1987, Technical Report CMU-CS-87-115, Computer Science Department, Carnegie-Mellon University, Pittsburgh PA 15213

[HuaLip87] W.Huang and R.Lippmann
"Comparisons Between Conventional And Neural Net Classifiers", Proceedings of the 1st International Conference on Neural Networks, Vol. 4, IEEE June 1987, pp 485-493.

[HusHor93] D.R.Hush and B.G.Horne
"Progress In Supervised Neural Networks, What's New Since Lippmann", IEEE Signal Processing Magazine; Jan 1993, pp 8-39.

[JanLai93] G.Jang and F.Lai
"Dual Adaptive-Structure Neural Networks For Stock Market Forecasting", Technical Report, Department Of Electrical Engineering, National Taiwan University.

[JanLai91] G.Jang, F.Lai B.Jiang and L.Chien
"An Intelligent Trend Prediction And Reversal Recognition System Using Dual-Module Neural Networks", Proceedings of the 1st. International Conference on AI Applications on Wall Street, New York, 1991, pp 42-51.

[Jenkin82] G.Jenkins,
"Some Practical Aspects Of Forecasting In Organisations", Journal of Forecasting, Vol.1, 1982, pp3-21.

[JonLee90] R.Jones, Y.Lee, C.Barnes, G.Flake, K.Lee, P.Lewis and S.Qian
"Function Approximation And Time-Series Prediction With Neural Networks", Proceedings of the 1990 International Joint Conference on Neural Networks, Vol. 1, June 1990, San Diego, pp 649-665.

[KamTan90] K.Kamijo and T.Tanagawa
"Stock Price Pattern Recognition: A Recurrent Neural Network Approach" Proceedings of the 1990 International Joint Conference on Neural Networks, June 1990, Vol. 1, San Diego, pp 215-221.

[KheTre93] S.Khebbal and P.Treleaven
"An Object-Orientated Hybrid Environment For Integrating Neural Networks And Expert Systems", Proceedings of the 1st. New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems, IEEE Computer Society Press, Los Alamitos, 1993.

[KimAsa90] T.Kimoto and K.Asakawa
"Stock Market Prediction System With Modular Neural Networks", Proceedings of the 1990 International Joint Conference on Neural Networks, Vol.1, June 1990, San Diego, pp 1-6.

[Kohone88] T.Kohonen
"The 'Neural' Phonetic Typewriter", IEEE Computer, Vol.21, March 1988, pp 11-22.

[KosSon90] A.Koster, N.Sondak and W.Bourbia
"A Business Application Of Artificial Neural Network Systems", The Journal of Computer Information Systems, Vol.XXXI, 1990, pp 3-10.

[Lake93] G.Lake
"How To Make Money With Neural Networks", M.Sc dissertation, Department Of Electrical Engineering, Kings College, University Of London.

[LapFar87] A.Lapedes and R.Farber
"Nonlinear Signal Prediction Using Neural Networks: Prediction and System Modelling", Los Alamos National Laboratory Report LA-UP-87-2662, 1987.

[Lapede88] A.Lapedes and R.Farber
"How Neural Networks Work", Neural Information Processing Systems, Dana Anderson (eds) American Institute of Physics, New York, 1988, pp 412-456.

[Lippma87] R.Lippmann
"An Introduction To Computing With Neural Nets", IEEE ASSP Magazine, April 1987, pp 4-22.

[Loo89] S.Loo
"Collating Real-Time Market Indicators Through A Temporal Paradigm", Expert Systems, November 1989, Vol.6, No.4, pp 263-269.

[MarAnd82] S.Makridakis, A.Anderson, R.Cabone, R.Fildes and M.Hibon
R. Lewandowski, J. Newton, E. Parzen, E and R. Winkler, "The Accuracy Of Extrapolation (Time Series) Methods: Results Of A Forecasting Competition", Journal of Forecasting, 1, 1982, pp 111-153.

[MarHil91] L.Marquez, T.Hill, W.Remus and R.Worthley
"Neural Network Models As An Alternative to Regression", Proceedings of the 24th Hawaii International Conference on System Sciences, Vol.4, 1991, pp 129-135.

[MarHil92] L.Marquez, T.Hill, M.O'Connor and W.Remus
"Neural Network Models For Forecast: A Review", Proceedings of the 25th Hawaii International Conference on System Sciences, Vol.4, 1992, pp 494-498.

[Matsub91] I.Matsuba
"Application Of Neural Sequential Associator To Long-Term Stock Price Prediction", Proceedings of the 1991 International Joint Conference On Neural Networks, Vol. 2, November 1991, Singapore, pp 1197-1201.

[MitGed92] V.Mital, T.Gedeon and L.Johnson
"Neural Nets And Knowledge Systems In Finance", Chapman & Hall, 1992.

[MITI90] Japanese Ministry Of International Trade And Industry, MITI
"Report Of The Research Committed On The New Information Processing Technology", April 1990. .

[Murphy86] J.Murphy
"Technical Analysis Of The Futures Markets: A Comprehensive Guide To Trading Methods And Applications", Institute of Finance, New York, 1986.

[MusAhm92] M.Musavi, W.Ahmed, K.Chan, K.Faris and D.Hummels
"On The Training Of Radial Basis Function Classifiers", Neural Networks, Vol. 5, pp 595-603.

[Nison91] S.Nison
"A Introduction To Japanese Candle Charts", Market Technician, The Journal Of The Society Of Technical Analysts, Issue 8, 1990, pp 9-15.

[OdoSha90] M.Odom and R.Sharda
"A Neural Network Model For Bankruptcy Prediction", Proceedings of the 1990 International Joint Conference on Neural Networks, Vol.2, June 1990, San Diego, pp 163-168.

[Pring80] M.Pring
"Technical Analysis Explained: An Illustrated Guide For The Investor", McGraw Hill, 1980.

[RagSch91] W.Raghupathi, L.Schkade and B.Bapı
"A Neural Network Application For Bankruptcy Prediction", Proceedings of the 24th Hawaii International Conference on System Sciences, Vol.4, 1991, pp 147-155.

[Refene91] A.Refenes
"Constructive Learning And Its Application To Currency Exchange Rate Forecasting", Chapter 27, Neural Network Applications in Investment and Financial Services, E.Turban and R.Trippi, Probus Publishing, USA, 1991.

[RefAze92] A.Refenes, M.Azema-Barac and S.Karoussos
"Currency Exchange Rate Forecasting By Error Backpropagation", Proceedings of the 25th Hawaii International Conference on System Sciences, Vol.4, 1992.

[ReiCol90] D.Reilly, E.Collins, C.Scofield and S.Ghosh
"Risk Assessment Of Mortgage Applications With A Neural Network System: An Update As The Test Portfolio Ages", Proceedings of the 1990 International Joint Conference on Neural Networks, Vol.2, June 1990, San Diego, pp 479-482.

[Remus87] W.Remus
"A Study Of The Impact Of Graphical And Tabular Displays And Their Interaction With Environmental Complexity", Management Science, Vol.33, 1987, pp 1200-1205.

[RemHil90] W.Remus and T.Hill
"Neural Network Models For Managerial Judgement", Proceedings of the 23th Hawaii International Conference on System Sciences, Vol.4, 1990, pp 340-344.

[RemHil91] W.Remus and T.Hill
"Neural Network Models For Intelligent Support Of Managerial Decision Making", University of Hawaii Working Paper, 1991.

[RocKhe93] P.Rocha, S.Khebbal and P.Treleaven
"A Framework For Hybrid Intelligent Systems", Proceedings of the 1st New Zealand Two-Stream Conference on Artificial Neural Networks and Expert Systems (ANNES '93), November 1993.

[RoyCos90] J.Roy and J.Cosset
"Forecasting Country Risk Ratings Using A Neural Network", Proceedings of the 23th Hawaii International Conference on System Sciences, Vol.4, 1990, pp 327-334.

[RudCla88] A.Rudd and H.Clasing Jr.
"Modern Portfolio Theory: The Principles Of Investment Management", 3rd Edition, 1988, Andrew Rudd.

[RumHin86] D.Rumelhart, G.Hinton and R.Williams
"Learning Representations By Backpropagating Errors", Nature, Vol.323, No.9, October 1986, pp 533-536.

[RumMcC86] D.Rumelhart and McCelland
"Parallel Distributed Processing, Explorations In The Microstructure Of Cognition", Vol. 1, MIT Press, 1986.

[Sal94] W.Sarle
Personal Communication, SAS Institute Inc., SAS Campus Drive, Cary, NC 27513, USA.

[Schöne90] E.Schoneburg
"Stock Price Prediction Using Neural Networks: A Project Report", Neurocomputing, Vol.2, 1990, pp 17-27.

[SejRos86] T.Sejnowski and C.Rosenberg
"NETalk: A Parallel Network That Learns To Read Aloud", John Hopkins University Technical Report JHU/EECS-86/01, 1986.

[Schwag84] J.Schwager
"A Complete Guide To The Futures Markets: Fundamental Analysis, Technical Analysis, Trading, Spreads And Options", John Wiley & Sons, 1984.

[ShaPat90] R.Sharda and R.Patil
"Neural Networks As Forecasting Experts: An Empirical Test", Proceedings of the 1990 International Joint Conference on Neural Networks, Vol.2, January 1990, Washington D.C, pp 491-494.

[SriLoo91] S.Srirengan and C.Looi
"On Using Backpropagation For Prediction: An Empirical Study", Proceedings of the 1991 IEEE International Joint Conference on Neural Networks, Vol.2, November 1991, Singapore, pp 1284-1289.

[StoHub87] W.Stornetta and B.Huberman
"An Improved Three-Layer Backpropagation Algorithm", Proceedings of the 1987 IEEE International Joint Conference on Neural Networks, Vol.2, June 1987, San Diego, pp 637-643.

[SugMay90] G.Sugihara and R.M.May
"Nonlinear Forecasting As A Way Of Distinguishing Chaos From Measurement Error In Time Series", Nature, 344, 1990, pp 734-741.

[SurSin90] A.Surkan and J.Singleton
"Neural Networks For Bond Rating Improved By Multiple Hidden Layers", Proceedings of the 1990 International Joint Conference on Neural Networks, Vol.2, June 1990, San Diego, pp 157-162.

[TanAlm90] Z.Tang; C.Almeida and P.Fishwick
"Time Series Forecasting Using Neural Networks vs. Box-Jenkins Methodology", International Workshop on Neural Networks, February 1990.

[TanKam92] T.Tanigawa and K.Kamijo
"Stock Price Pattern Matching System: Dynamic Programming Neural Network Approach", Proceedings of the 1992 International Joint Conference on Neural Networks, Vol. 2, June 1992, Baltimore, pp 465-471.

[TreLoo90] P.Treleaven and S.Loo
"Novel Information Technologies For Rocket Science", Rocket Science Made Simple: Forecasting and Optimisation in Financial Services, IBC Technical Services Ltd., London 3-4th October, 1990, pp 1-12.

[TreGoo92] P.Treleaven and S.Goonatilake
"Intelligent Financial Technologies", Proceedings of the Workshop on Parallel Problem Solving from Nature: Applications in Statistics and Economics, Statistical Office of the European Communities (EUROSTAT), 1992.

[TriBer91] D.Trigueiros and R.Berry
"The Application Of Neural Network Based Methods To The Extraction Of Knowledge From Accounting Reports", Proceedings of the 24th Hawaii International Conference on System Sciences, Vol.4, 1991

[VarVer90] A.Varfis and C.Versino
"Univariate Economic Time Series Forecasting By Connectionist Methods", International Neural Network Conference, Paris, 1990, Vol.2, pp 342-345.

[Wasser89] P.Wasserman
"Neural Computing: Theory And Practise", Van Nostrand Reinhold, 1989.

[Wasser93] P.Wasserman
"Advanced Methods In Neural Computing", Van Nostrand Reinhold, 1993.

[WeiHub90] A.Weigend, B.Huberman and D.Rumelhart
"Predicting The Future: A Connectionist Approach", International Journal Of Neural Systems, Vol.1, No.3, 1990, pp 193-209

[White89] H.White
"Learning In Artificial Neural Networks: A Statistical Perspective", Neural Computation, Vol.1, No.4, 1989, pp 425-464.

[White88] H.White
"Economic Prediction Using Neural Networks: A Case Study Of IBM Daily Stock Returns" Proceedings of the 1988 IEEE International Conference on Neural Networks, Vol. 2, San Diego, pp 451-458.

[WilZip89] R.Williams and D.Zipser
"A Learning Algorithm For Continually Running Fully Recurrent Neural Networks" Neural Computation, Vol. 1, pp 270-280.

[WinHar90] C.Windsor and A.Harker
"Multi-variate Financial Index Prediction - A Neural Network Study", International Neural Network Conference, Paris, 1990, Vol.2, pp 357-360.

[WolMia90] D.M.Wolpert and R.C.Miall
"Detecting Chaos With Neural Networks", Proceedings of the Royal Society, London, B, 242, 1990, pp 82-86.

[Wong91] F.Wong
"FastProp: A Selective Training Algorithm For Fast Error Propagation", Proceedings of the 1991 IEEE International Joint Conference on Neural Networks, Vol.2, November 1991, Singapore, pp 2038-2043.

[WonTan92] F.Wong, P.Tan and X.Zhang
"Neural Networks, Genetic Algorithms And Fuzzy Logic For Forecasting", Proceedings of the 3rd. International Conference on Advanced Trading Technologies - AI Applications on Wall Street And Worldwide, New York, July 1992.

[YooSwa91] Y.Yoon and G.Swales
"Predicting Stock Price Performance", Proceedings of the 24th Hawaii International Conference on System Sciences, Vol.4, 1991, pp 156-162.

[Yourdo77] E.Yourdon
"Managing The Structured Techniques", 2nd. ed. Englewood Cliffs, N.J., Prentice Hall, Inc., 1977.

Appendix A

Processes Of The Neural Forecasting System

(Figures 3.5 to 3.8)

This page is deliberately left blank

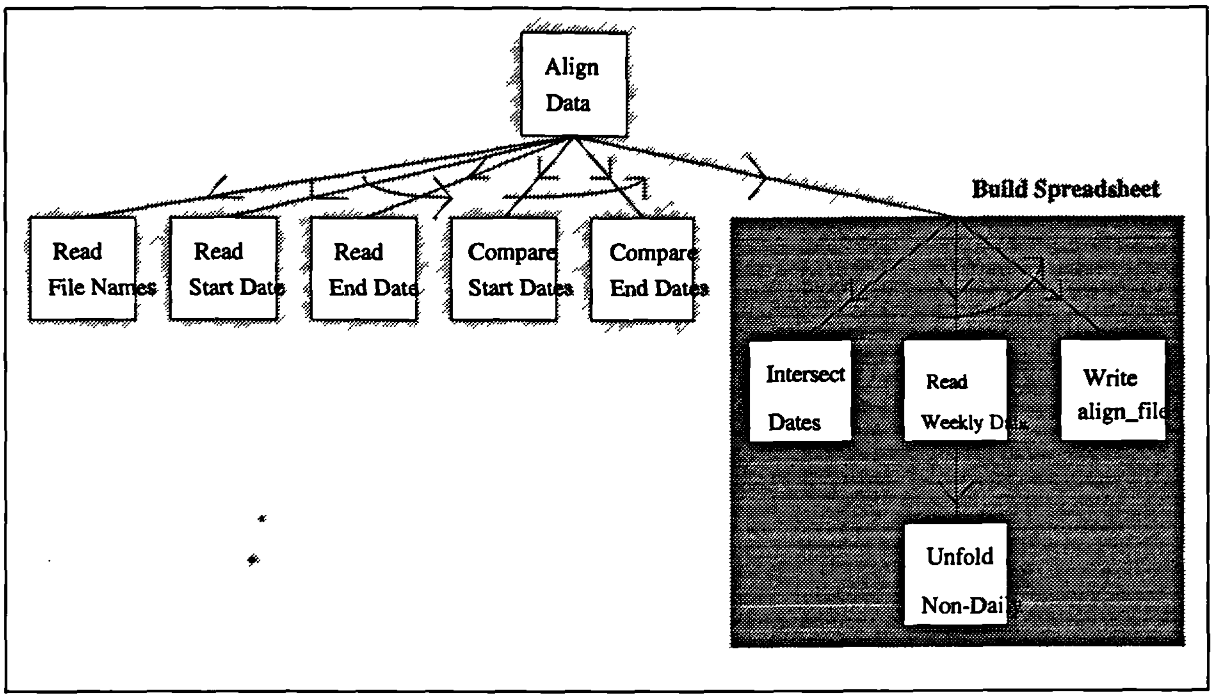


Figure 3.5 A Structure Chart For The **Align** Process

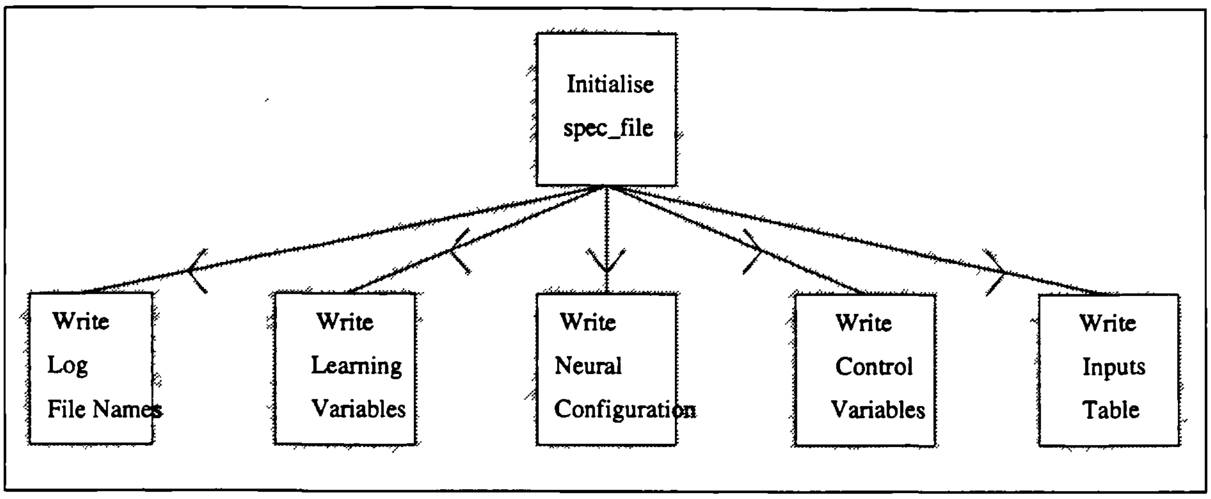


Figure 3.6 A Structure Chart For The **Specify** Process

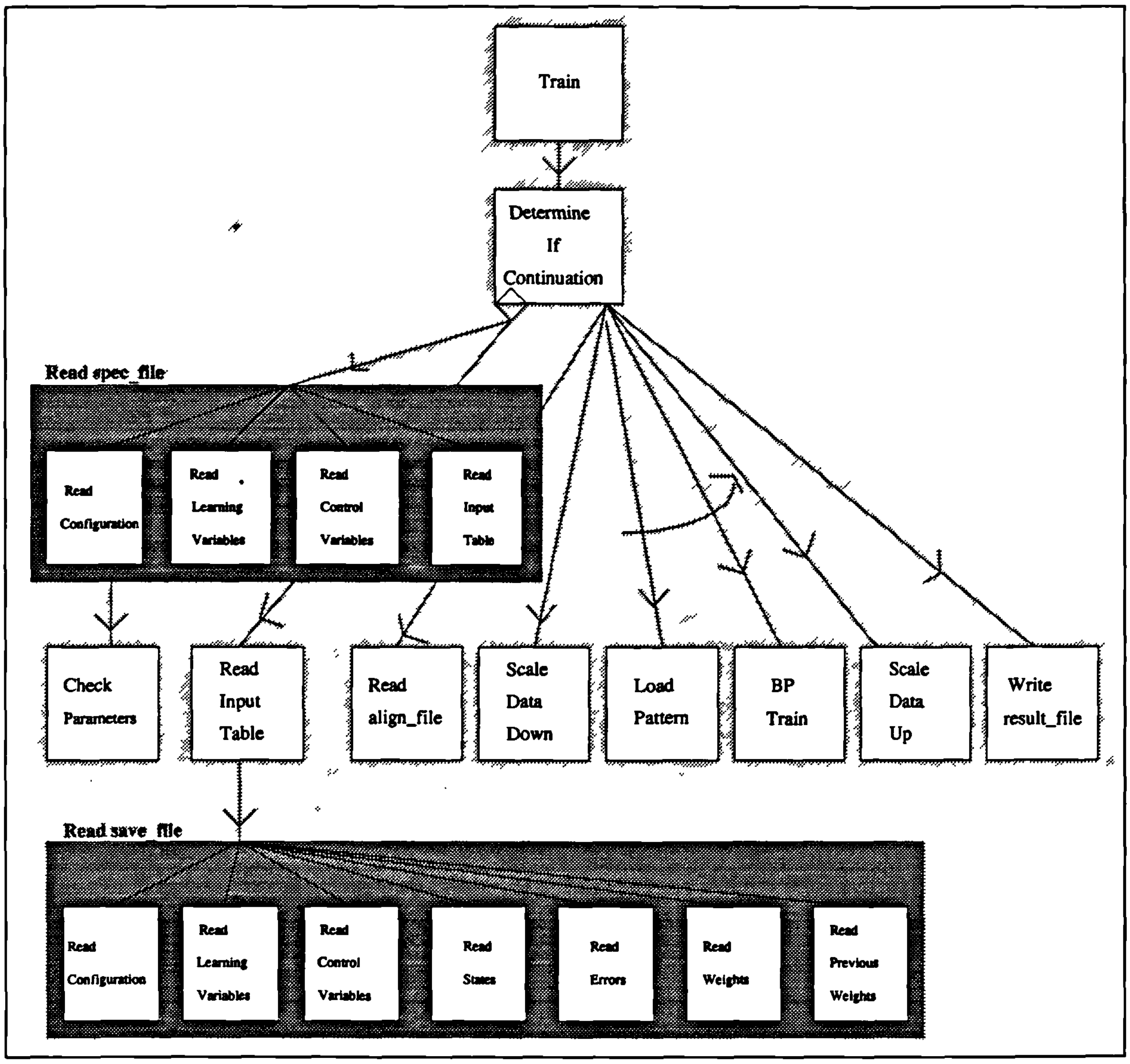


Figure 3.7 A Structure Chart For The **Train** Process

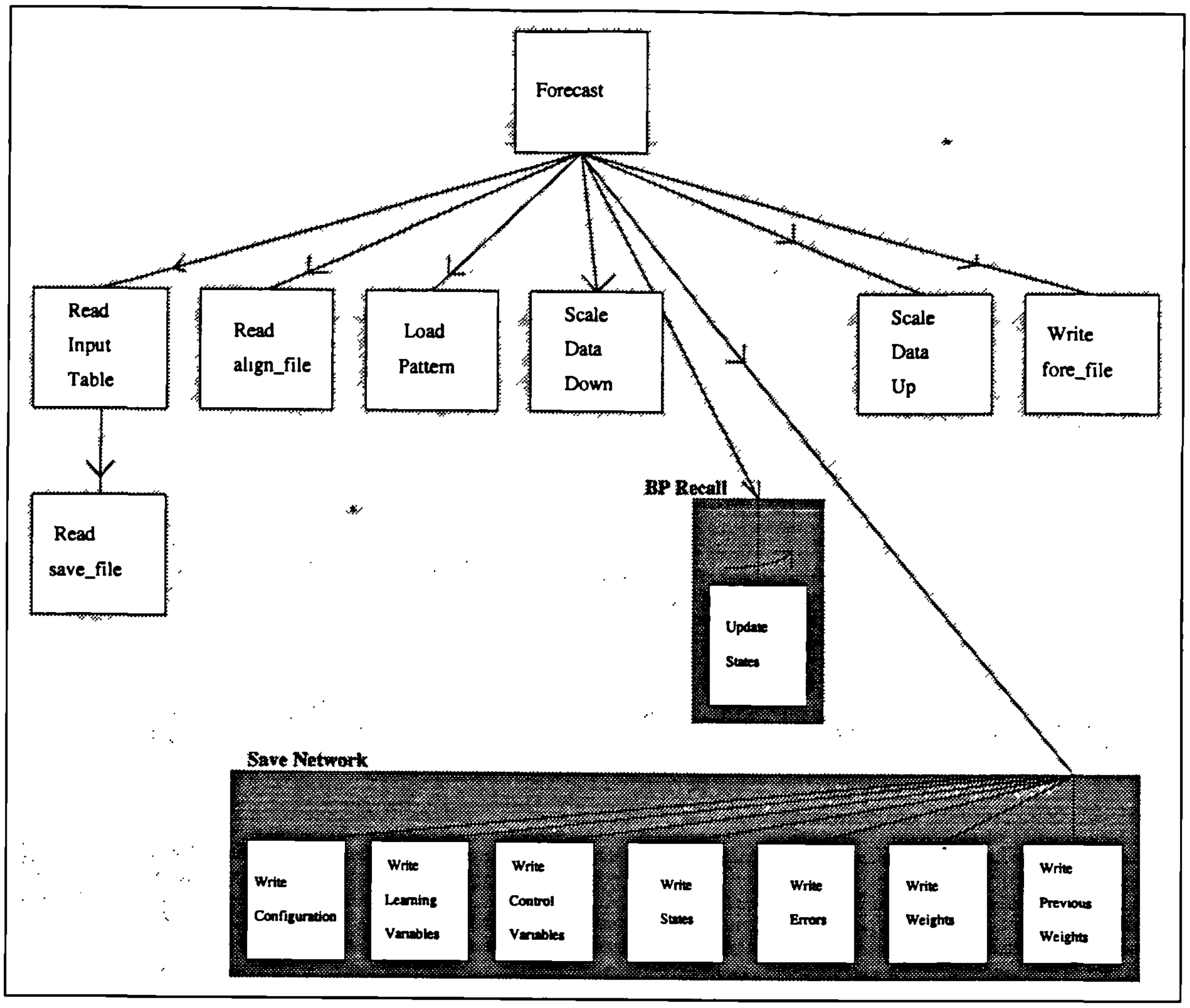


Figure 3.8 A Structure Chart For The **Forecast** Process

Appendix B

Correlation Measurements From Experiments 2a(i) To 2a(iii)

This page is deliberately left blank

Table 6.1

Optimisation Of Neural Forecasting Models - Experiments 2(a)(i) To (iii)

Aim	: Model N against Models sN1 and sN2
Data	: **Consolidation 1/1/91 to 31/3/92**, Two-thirds for training, One-third for testing
Constants	: Momentum = 0.05, Configuration = **RuleS**, Learn rate = 0.1, Tolerance = **0.02**
Procedure	:

σ	Model	Cycle	VCorrelation (Train %	Test %)	VP & L (Train %	Test %)	DCorrelation (Train %	Test %)	DP & L (Train %	Test %)	VDCorrelation (Train %	Test %)	VDP & L (Train %	Test %)	Network Errors Avg(10^3)	Max(10^3)	Tol Test No Fail	% Fail
1	N	4000	80.48	49.04	88.78	9.39	69.05	43.27	60.81	-17.14	57.14	12.50	52.82	-75.93	46.81	131.27	169	19.91
		5000	79.52	45.19	87.25	6.58	67.62	42.31	60.69	-15.57	56.67	13.46	51.87	-76.49	42.04	141.43	163	22.75
		6000	74.29	46.15	79.50	6.96	69.05	41.35	60.68	-17.48	54.29	14.42	46.28	-73.79	32.82	129.85	122	42.18
		7000	74.29	46.15	78.66	6.96	70.00	44.23	61.77	-4.53	54.76	17.31	46.66	-60.84	32.75	131.32	119	43.60
		8000	75.71	46.15	82.32	6.96	67.14	43.27	58.11	-6.38	52.38	16.35	45.02	-63.51	32.80	132.66	119	43.60
	sN1	4000	80.48	55.77	89.67	9.62	75.24	50.96	75.36	-3.26	68.57	36.54	71.61	-34.44	41.08	150.02	155	26.54
		5000	70.00	45.19	73.00	-3.01	64.76	50.00	55.66	-3.72	52.86	25.96	42.54	-46.92	45.29	157.75	145	31.28
		6000	72.86	50.00	76.83	1.43	68.57	50.00	64.67	-4.62	60.00	32.69	56.13	-37.88	45.15	160.10	141	33.18
		7000	73.33	47.12	78.26	-3.6	69.52	50.96	63.85	-0.58	58.57	27.88	53.36	-49.86	41.92	151.67	131	37.91
		8000	Abandon															
	sN2	4000	68.57	49.04	59.56	-0.82	70.00	45.19	54.06	1.55	53.33	26.92	29.76	-41.08	63.88	359.94	138	34.60
		5000	67.14	45.19	64.64	3.81	66.67	47.12	52.37	-13.50	51.43	24.04	33.25	-55.16	54.95	196.28	155	26.54
		6000	65.24	47.12	58.65	8.79	67.14	46.15	45.85	-9.06	44.76	20.19	.15.39	-56.10	57.40	228.78	152	27.96
		7000	63.33	44.23	58.89	-2.82	66.19	47.12	49.47	-6.68	44.76	19.23	21.70	-55.07	59.24	236.74	153	27.49
2	N	2000	75.60	48.08	81.59	8.37	68.42	42.31	63.17	-9.63	54.07	17.31	50.91	-62.57	47.63	162.22	158	24.76
		5000	77.99	43.27	81.72	-5.13	69.86	44.23	62.03	-5.21	59.81	17.31	51.90	-60.06	33.68	128.34	116	44.76
		6000	76.08	43.27	80.41	-5.13	71.17	47.12	66.95	-2.33	60.77	19.23	56.04	-58.47	33.58	128.11	118	43.81
		7000	76.08	43.27	80.35	-5.13	70.81	47.12	64.84	1.70	59.81	19.23	54.53	-54.44	33.12	125.94	118	43.81

	sN1	4000	74.16	55.77	75.83	19.23	70.33	42.31	59.58	-10.27	56.46	20.19	44.12	-54.24	40.12	138.45	134	36.19
		5000	71.29	50.00	74.64	7.05	64.11	50.00	54.28	-9.05	55.02	28.85	43.72	-50.10	65.90	206.69	159	24.29
		6000	71.29	47.12	72.70	8.60	68.42	51.92	56.03	1.44	54.55	25.96	40.69	-48.32	59.20	205.01	148	29.52
		7000	72.73	50.00	76.68	6.62	68.90	46.15	55.68	-16.45	56.94	25.00	45.03	-60.56	50.99	198.64	135	35.71
	sN2	4000	67.94	50.96	62.49	14.10	58.37	45.19	28.85	-10.12	40.19	21.15	4.52	-58.15	59.26	173.09	148	29.52
		5000	68.90	51.92	66.61	16.72	56.46	50.96	28.71	7.88	39.23	25.00	8.65	-46.79	54.12	162.51	152	27.62
		6000	70.33	51.92	66.44	16.89	56.94	53.85	27.34	12.29	39.71	25.96	7.43	-46.80	53.51	163.33	150	28.57
		7000	70.33	51.92	67.33	16.89	58.37	51.92	30.99	14.55	41.63	24.04	12.14	-44.54	52.96	163.94	153	27.14
3	N	4000	77.51	41.75	83.86	-5.42	73.21	44.66	67.32	-9.41	60.77	17.48	56.72	-66.42	43.90	136.62	164	21.90
		5000	74.64	41.75	78.73	-5.42	72.73	47.57	69.71	5.79	60.77	20.39	57.45	-50.50	36.45	118.04	124	40.95
		6000	74.64	41.75	78.73	-5.42	72.73	47.57	69.71	5.79	60.77	20.39	57.45	-50.50	36.45	118.04	124	40.95
		7000	74.64	41.75	78.48	-5.42	71.29	48.54	64.89	6.25	59.33	21.36	53.32	-50.04	36.77	115.69	128	39.05
	sN1	4000	75.12	48.54	82.74	5.91	67.94	43.69	54.36	-14.26	52.63	17.48	42.78	-69.29	59.24	173.80	178	15.24
		5000	68.90	49.51	71.18	4.29	64.11	44.66	47.47	-3.51	45.45	15.53	28.52	-63.68	48.93	147.53	153	27.14
		6000	68.42	49.51	71.32	3.18	61.72	45.63	43.55	2.54	44.50	16.50	26.09	-60.65	49.40	149.09	152	27.62
		7000	68.90	52.43	69.78	8.78	64.59	44.66	49.35	1.79	48.33	17.48	30.78	-57.21	49.89	150.57	151	28.10
	sN2	4000	74.64	41.75	79.16	-5.42	67.46	41.75	55.41	-11.67	55.98	15.53	45.61	-66.64	32.48	123.27	106	49.52
		5000	74.16	41.75	77.63	-5.42	66.03	41.75	51.67	-11.67	53.59	15.53	39.65	-66.64	32.25	120.90	108	48.57
		7000	Abandon															
4	N	4000	75.48	53.40	79.47	10.99	65.87	41.75	51.20	-4.11	51.92	20.39	39.06	-52.91	65.57	222.86	175	16.27
		5000	68.75	53.40	69.47	14.62	69.23	44.66	59.32	-3.27	52.40	22.33	39.79	-54.32	51.37	160.47	141	32.54
		6000	67.79	49.51	65.32	8.53	60.10	45.51	34.97	-1.23	41.35	19.42	13.09	-56.06	61.26	202.04	158	24.40
		7000	66.83	48.54	63.20	1.23	63.46	43.69	42.26	1.10	43.75	17.48	18.83	-60.00	57.77	188.65	158	24.40
	sN1	4000	75.96	50.49	84.54	10.15	73.08	42.72	67.70	-9.92	58.65	18.45	56.65	-62.15	49.71	152.23	162	22.49
		5000	73.08	51.46	76.64	10.56	63.94	44.66	46.06	-6.93	51.44	20.39	33.40	-57.61	52.79	164.18	153	26.79
		6000	71.15	51.46	74.26	10.56	62.98	42.72	44.37	-9.34	49.04	19.42	30.39	-58.61	50.89	161.55	151	27.75
		7000	71.63	51.46	74.43	10.56	62.98	42.72	42.35	-10.83	48.56	17.48	27.64	-63.78	49.84	158.77	153	26.79
	sN2	4000	70.67	51.46	74.17	9.02	62.50	41.75	45.22	-4.71	50.48	16.50	33.10	-59.16	36.88	123.13	135	35.41
		5000	69.23	51.46	70.47	9.02	61.54	41.75	44.18	-4.74	49.04	16.50	30.85	-59.16	37.35	125.06	137	34.45
		6000	68.75	51.46	70.00	9.02	62.02	41.75	48.22	-4.74	50.48	16.50	35.18	-59.16	36.95	123.53	135	35.41

		7000	69.23	51.46	70.68	9.02	62.50	39.81	45.95	-7.18	49.52	15.53	31.44	-59.61	37.38	123.88	134	35.89
5	N	4000	74.40	52.43	80.07	9.94	68.12	40.78	56.32	-13.98	55.56	20.39	46.61	-60.56	52.85	150.11	171	17.79
	£	5000	69.57	50.49	71.48	8.11	65.70	44.66	46.84	1.56	51.69	18.45	33.27	-53.69	42.42	125.30	144	30.77
		6000	68.60	50.49	70.07	8.10	62.80	41.75	41.33	-8.51	47.83	17.48	25.94	-61.28	41.07	127.78	142	31.73
		7000	66.67	50.49	66.78	8.10	61.35	39.81	39.67	-9.34	46.38	15.53	23.63	-62.11	40.82	132.25	143	31.25
	sN1	4000	78.26	41.75	85.66	-5.42	74.88	44.66	72.31	-12.12	63.77	16.50	63.71	-64.52	47.31	145.59	149	28.37
		5000	73.91	41.75	77.27	-5.42	74.88	45.63	71.77	-10.51	63.29	16.50	59.54	-64.52	41.49	128.60	131	37.02
		6000	75.36	41.75	80.99	-5.42	74.88	43.69	73.00	-11.49	64.25	16.50	62.85	-64.52	39.69	122.98	126	39.42
		7000	Abandon															
	sN2	4000	69.08	49.51	72.35	6.62	64.73	42.72	50.73	-10.85	49.28	15.53	34.46	-66.74	40.34	126.79	139	33.17
		5000	69.57	51.46	71.65	11.96	62.80	44.66	44.51	-0.86	45.89	18.45	27.83	-54.62	41.10	126.40	142	31.73
		6000	68.60	43.69	61.45	1.24	64.25	52.43	46.50	9.41	42.51	24.27	17.37	-43.06	55.20	185.64	139	33.17
		7000	68.12	54.37	63.97	20.76	63.77	52.43	47.98	28.23	43.48	27.18	22.96	-30.07	50.80	180.17	133	36.06

Appendix C

Prediction Accuracy Of `Rules` Networks

(Figures 6.2a And 6.2b)

This page is deliberately left blank

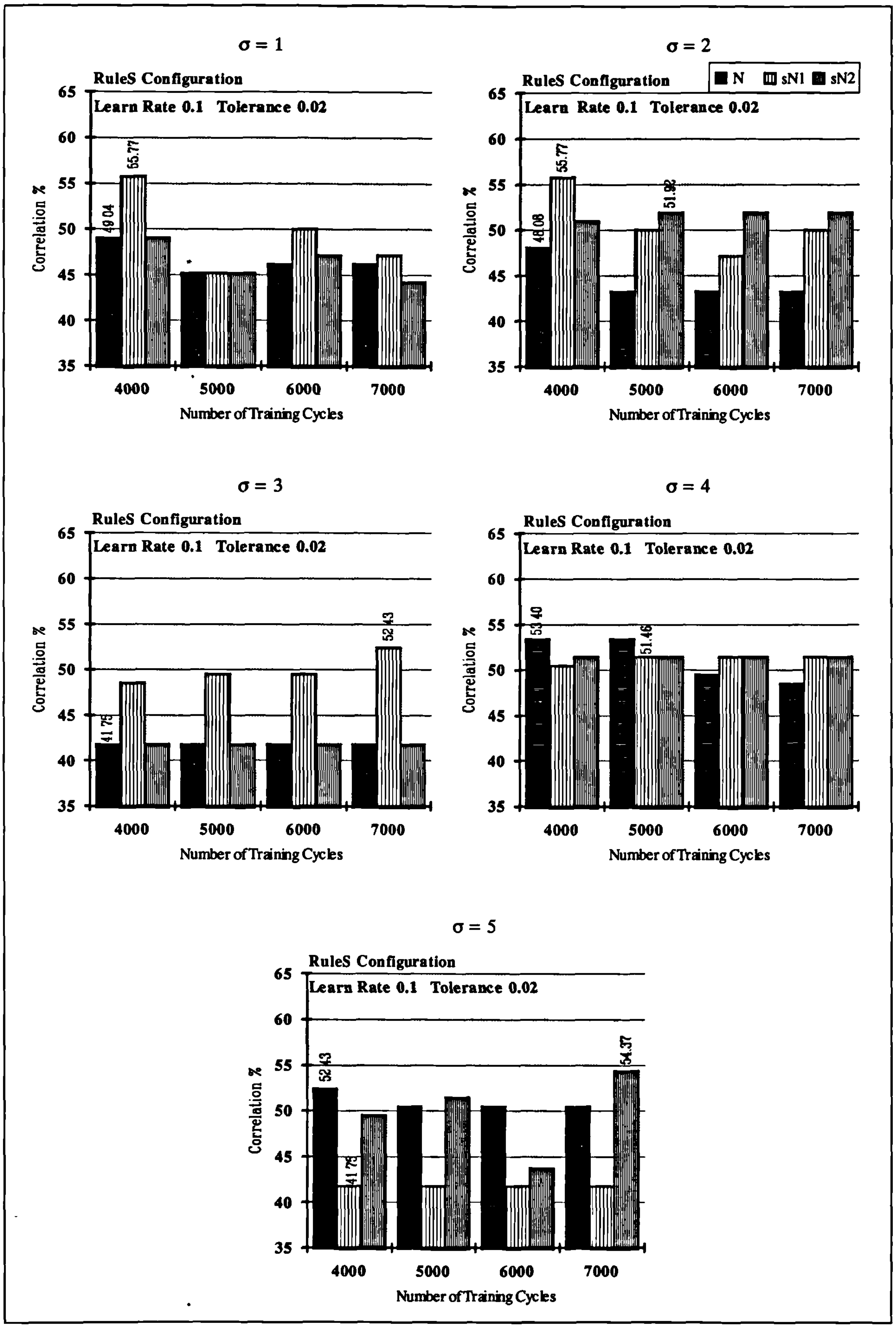


Figure 6.2a Test Correlation (%) For RuleS Configuration With Tolerance = 0.02

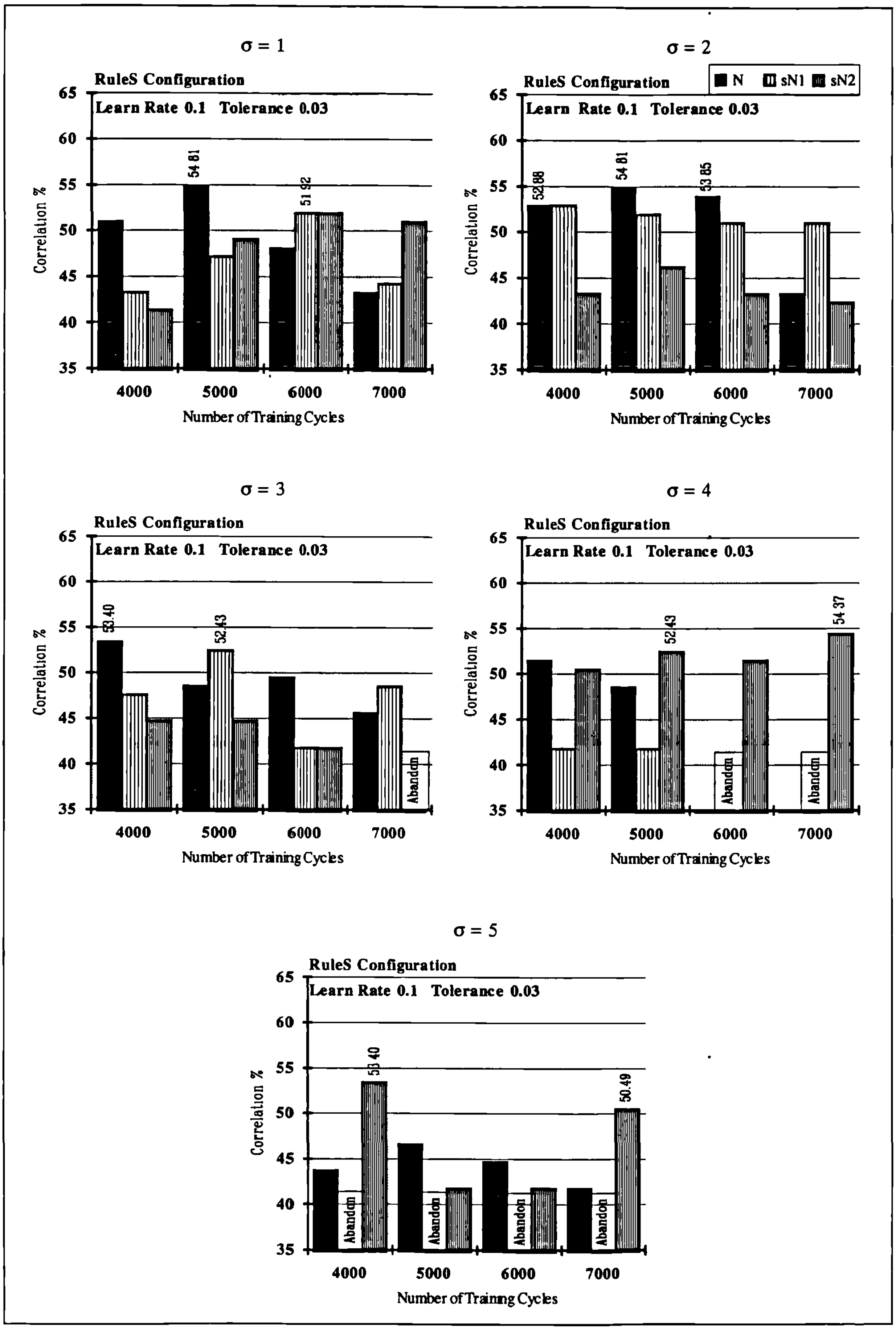


Figure 6.2b Test Correlation (%) For RuleS Configuration With Tolerance = 0.03

Appendix D

Prediction Accuracy Of RuleD Networks

(Figures 6.3a And 6.3b)

This page is deliberately left blank

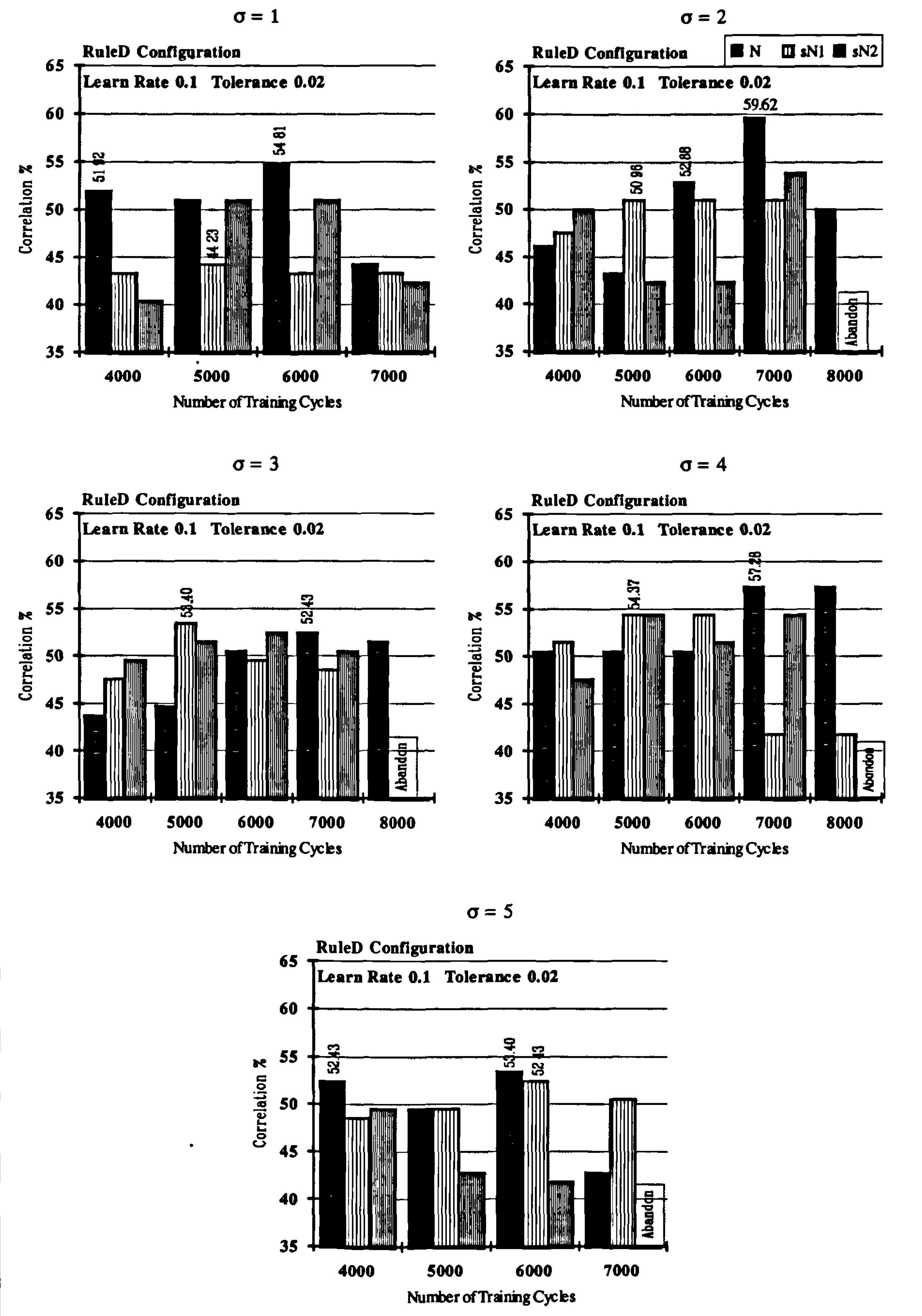


Figure 6.3a Test Correlation (%) For RuleD Configuration With Tolerance = 0.02

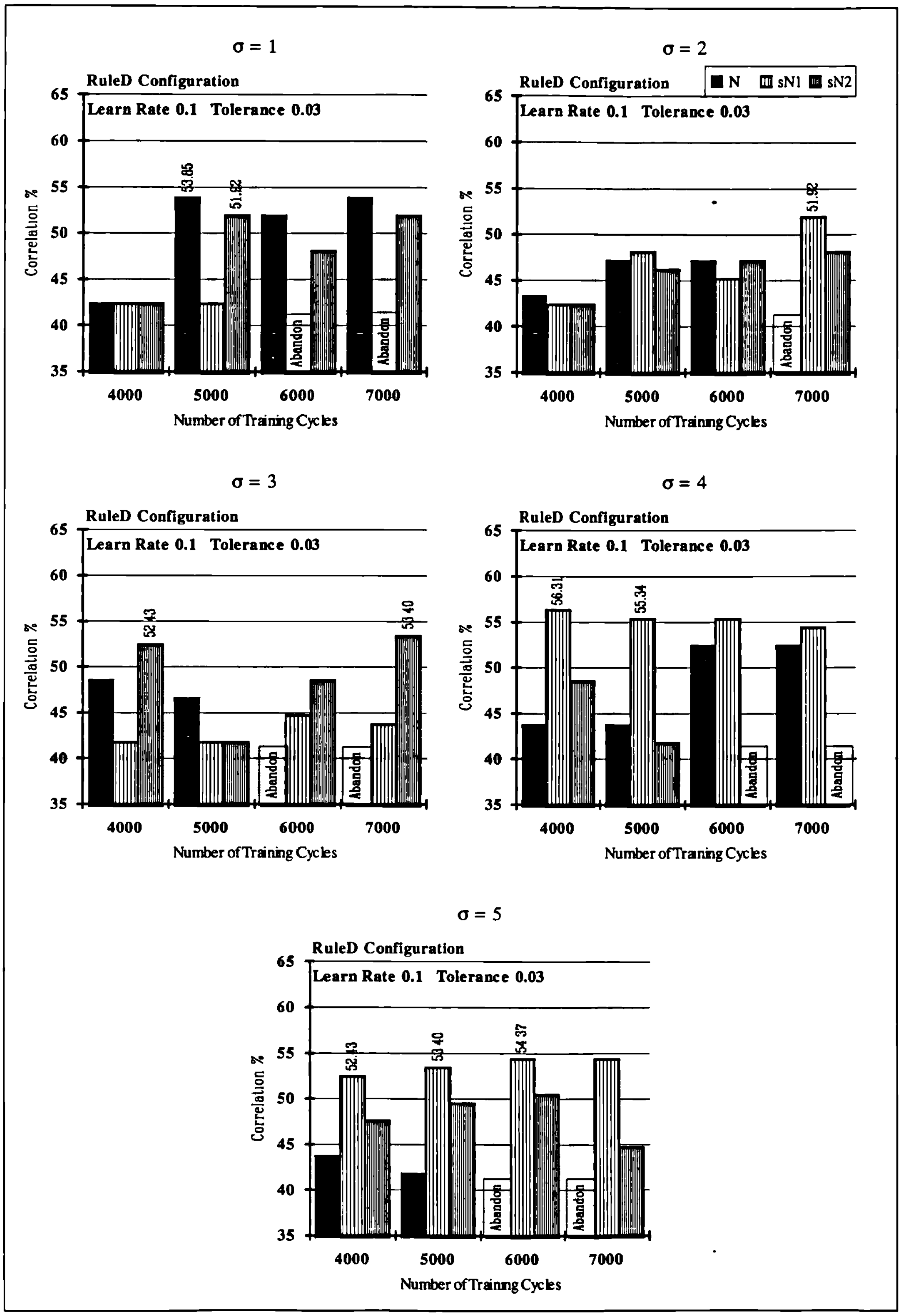


Figure 6.3b Test Correlation (%) For RuleD Configuration With Tolerance = 0.03

Appendix E

Prediction Accuracy Of `RuleT` Networks

(Figures 6.4a And 6.4b)

This page is deliberately left blank

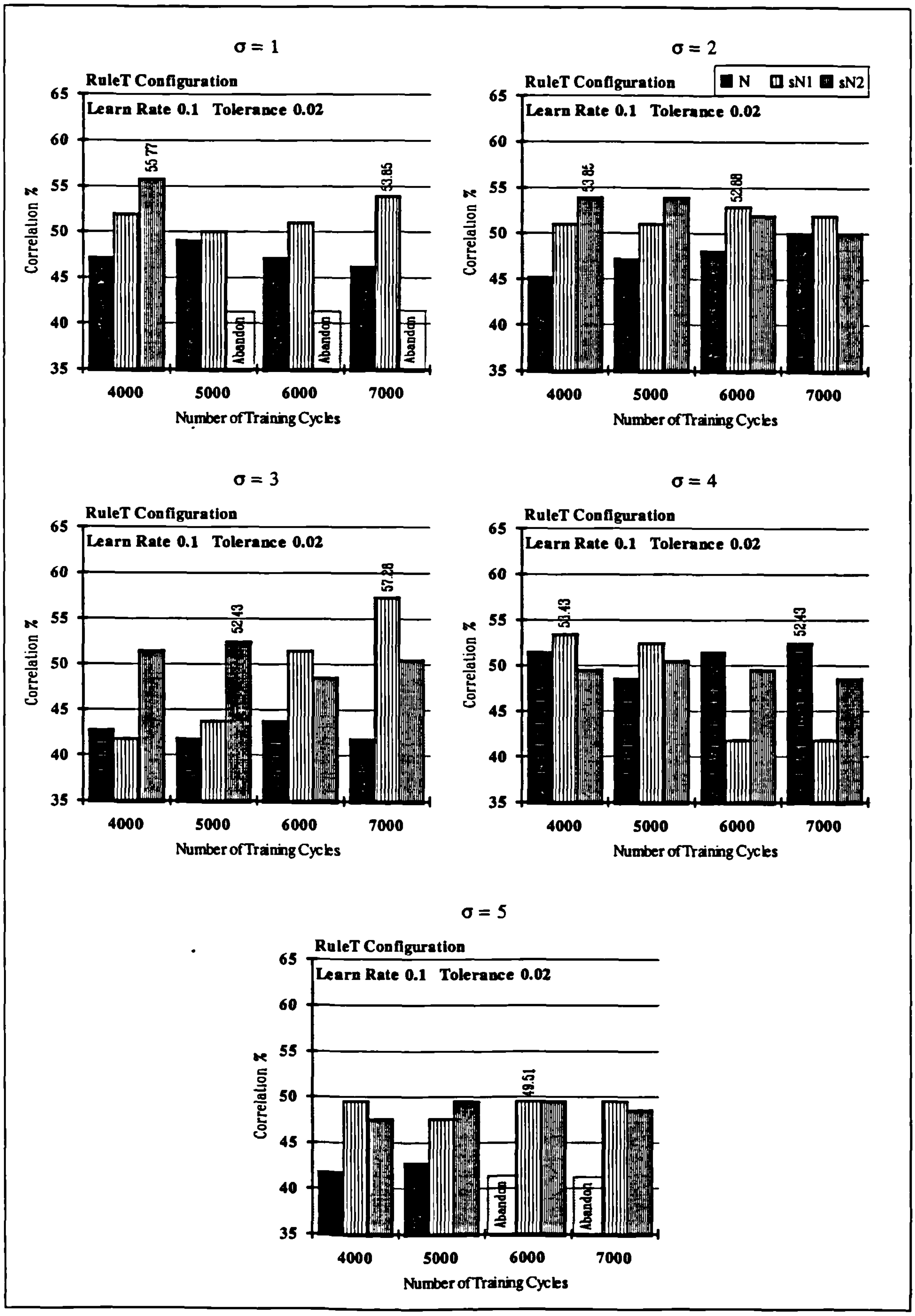


Figure 6.4a Test Correlation (%) For RuleT Configuration With Tolerance = 0.02

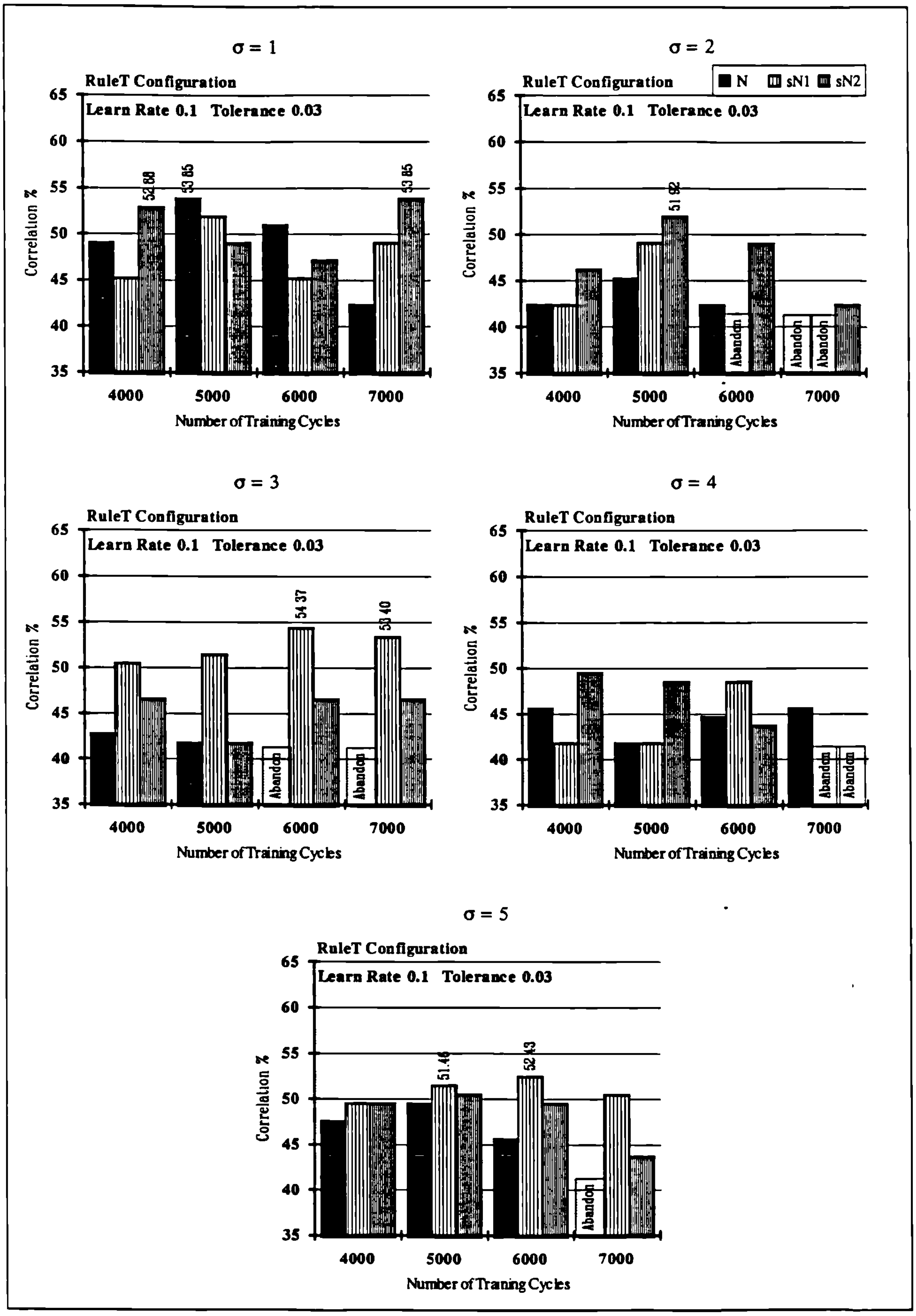


Figure 6.4b Test Correlation (%) For RuleT Configuration With Tolerance = 0.03

Appendix F

sN1 Networks And Relative Threshold Prediction Index

(Figures 6.5a To 6.5e)

This page is deliberately left blank

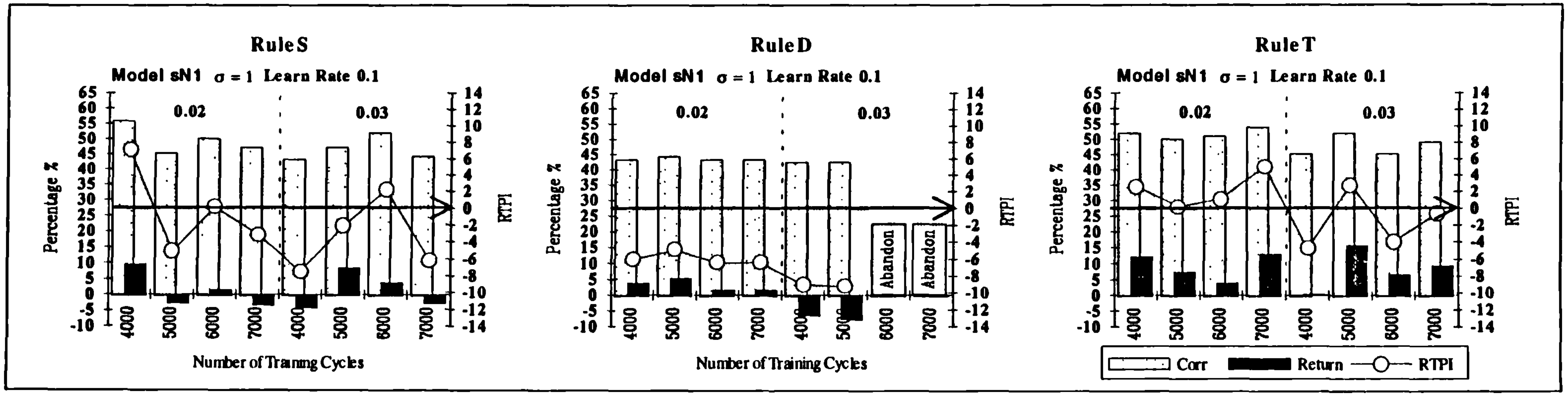


Figure 6.5a RTPI Of sN1 Networks With Configurations Using σ = 1

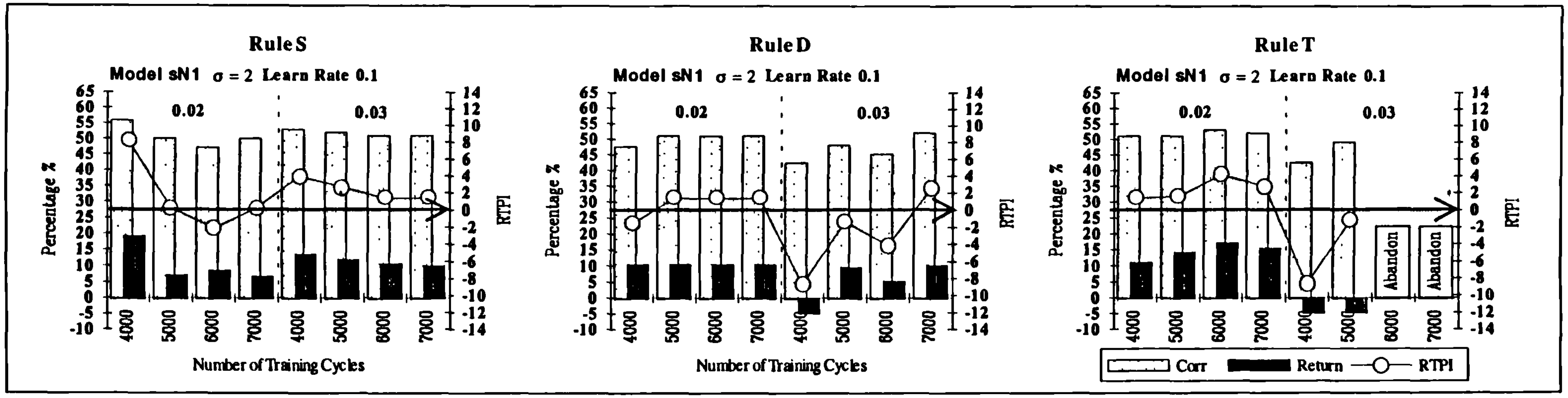


Figure 6.5b RTPI Of sN1 Networks With Configurations Using σ = 2

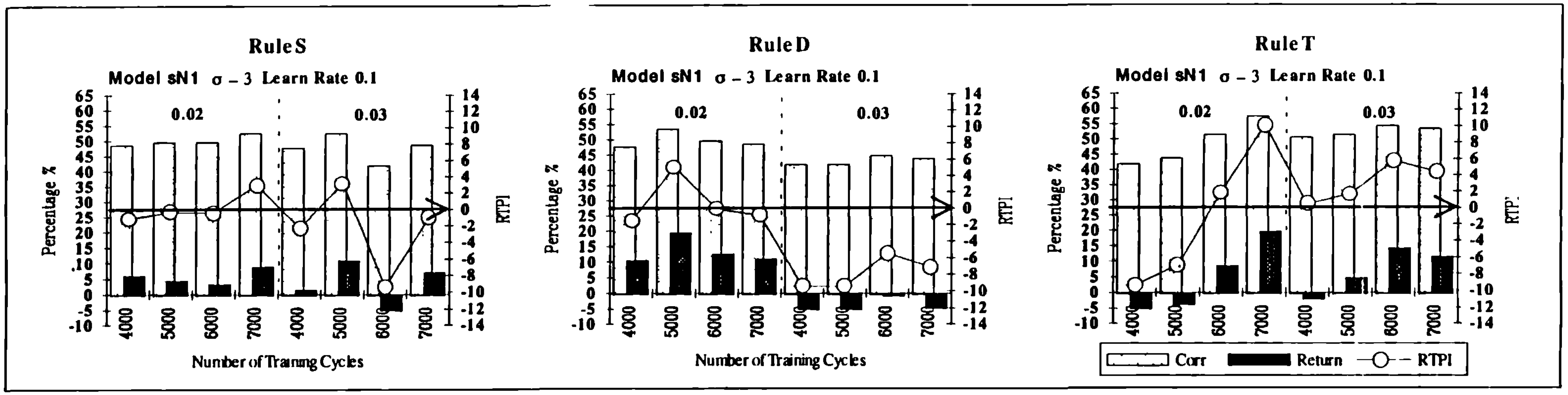


Figure 6.5c RTPI Of sN1 Networks With Configurations Using σ = 3

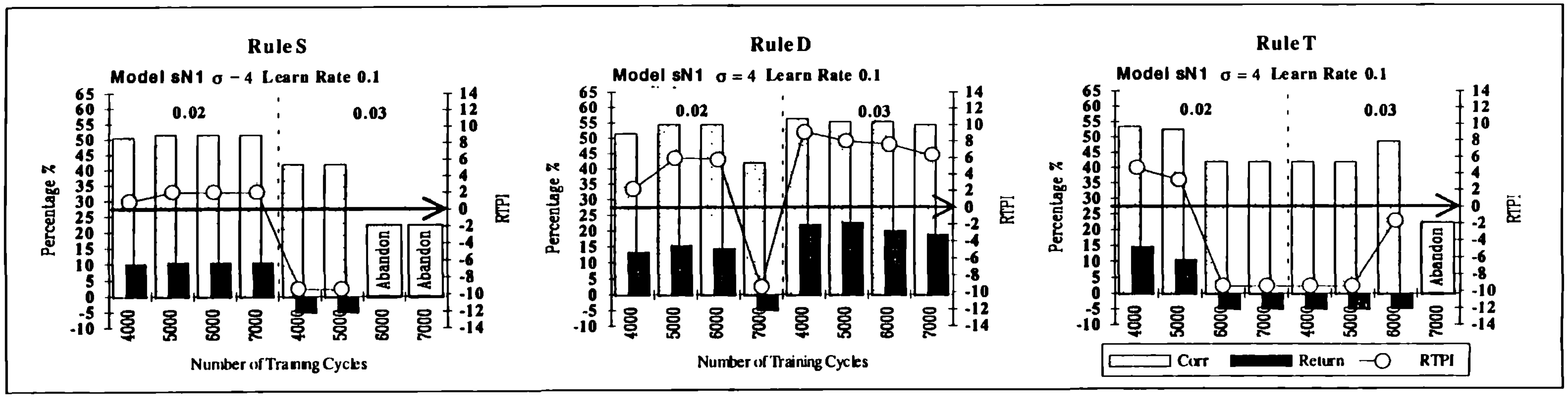


Figure 6.5d RTPI Of sN1 Networks With Configurations Using σ = 4

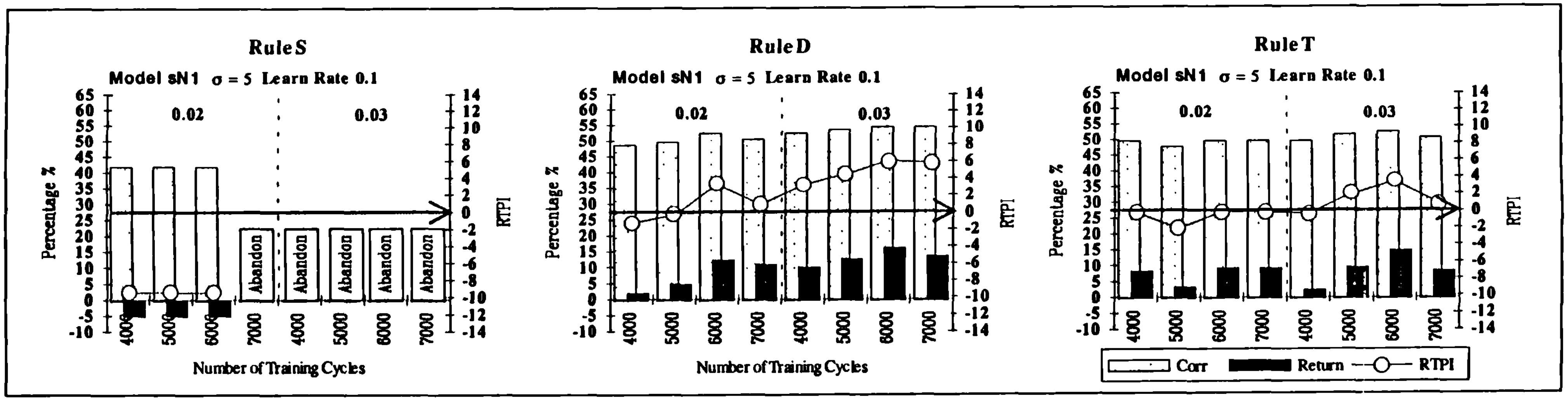


Figure 6.5e **RTPI** Of sN1 Networks With Configurations Using σ = 5

Appendix G

sN2 Networks And Relative Threshold Prediction Index

(Figures 6.6a To 6.6e)

This page is deliberately left blank

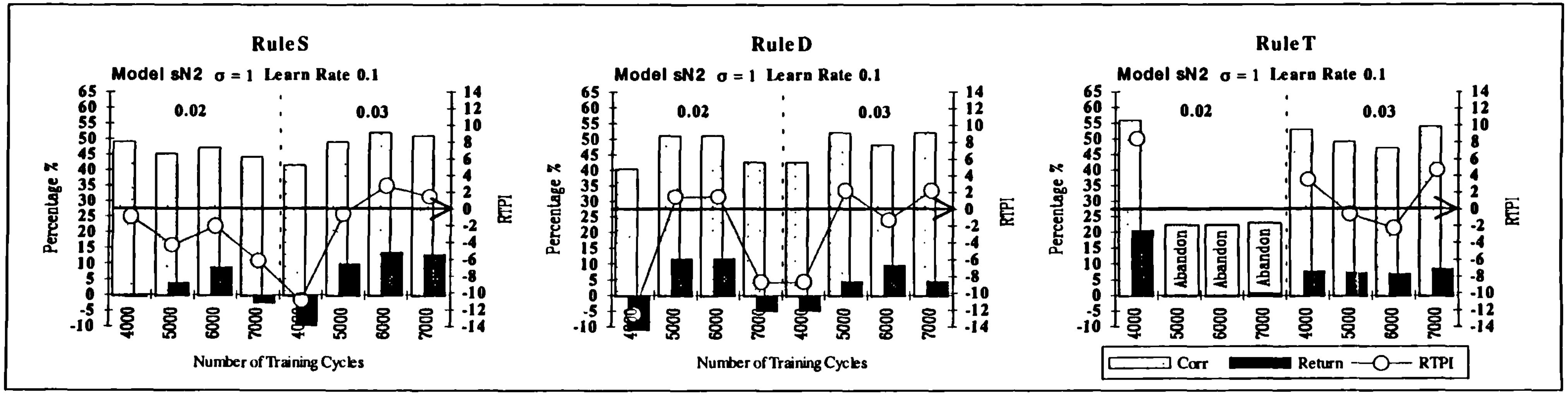


Figure 6.6a RTPI Of sN2 Networks With Configurations Using σ = 1

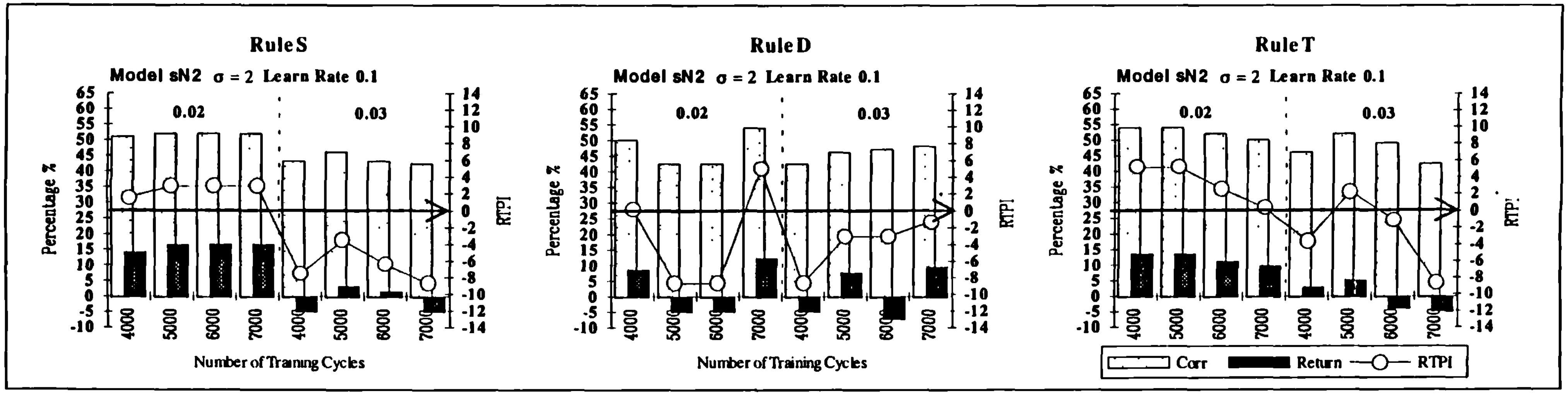


Figure 6.6b RTPI Of sN2 Networks With Configurations Using σ = 2

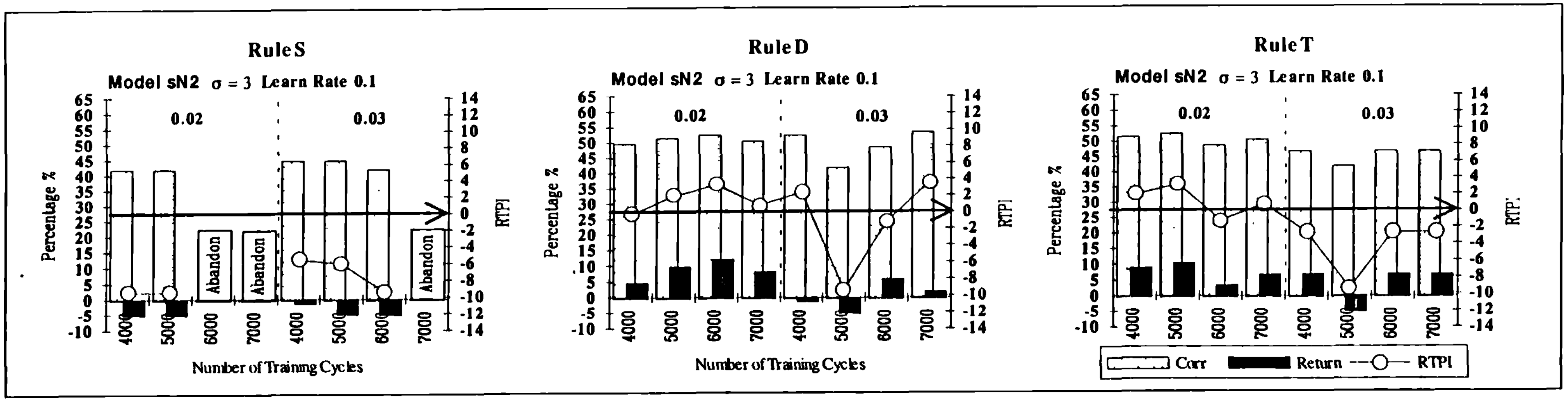


Figure 6.6c **RTPI** Of sN2 Networks With Configurations Using σ = 3

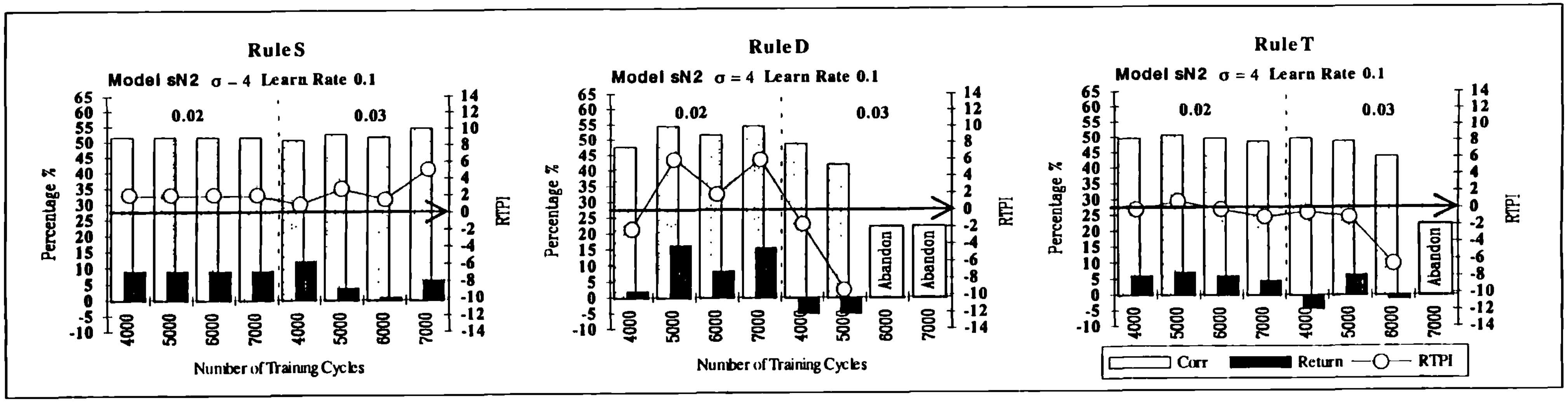


Figure 6.6d **RTPI** Of sN2 Networks With Configurations Using σ = 4

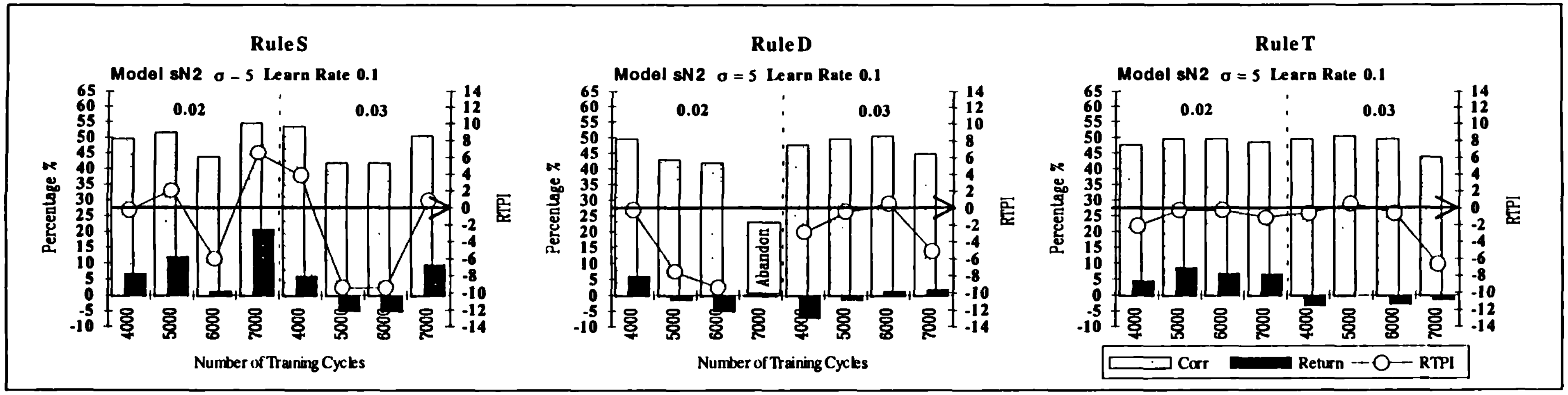


Figure 6.6e RTPI Of sN2 Networks With Configurations Using σ = 5

Appendix H

N Networks And Relative Threshold Prediction Index

(Figures 6.7a To 6.7e)

<u>This page is deliberately left blank</u>

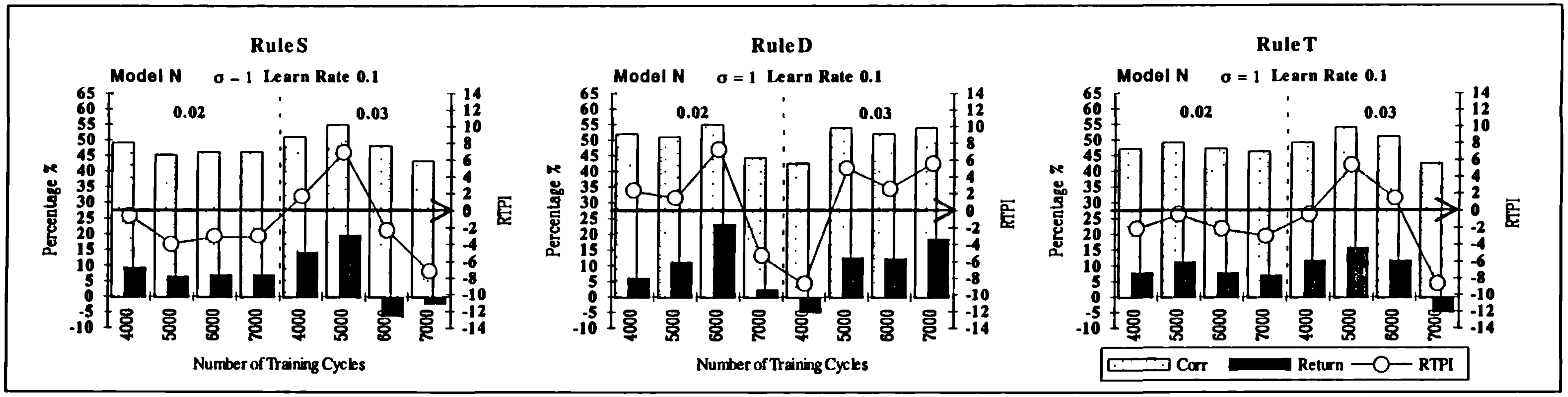


Figure 6.7a RTPI Of N Networks With Configurations Using σ = 1

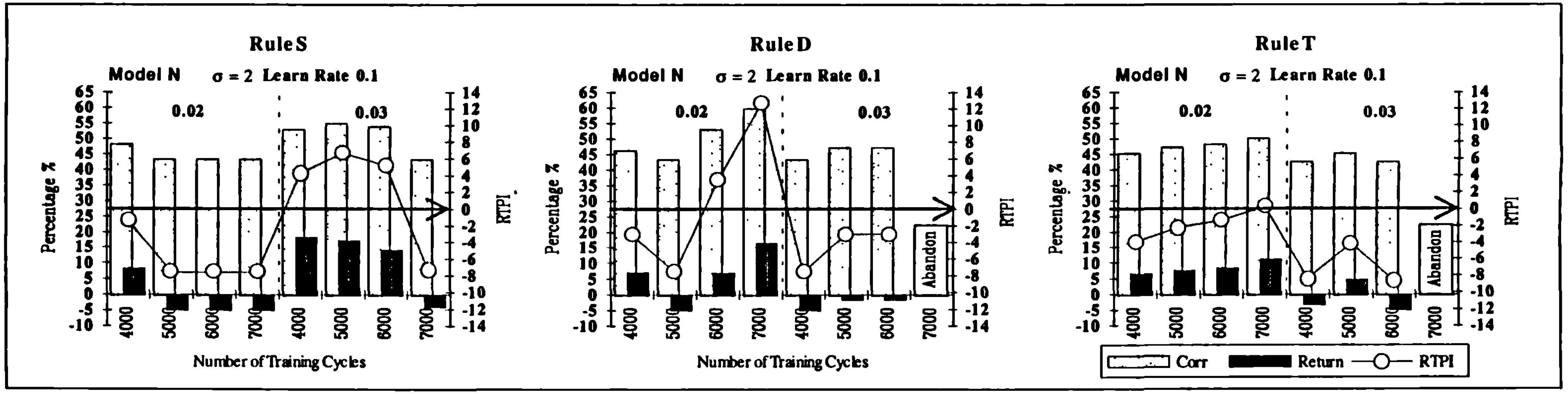


Figure 6.7b RTPI Of N Networks With Configurations Using σ = 2

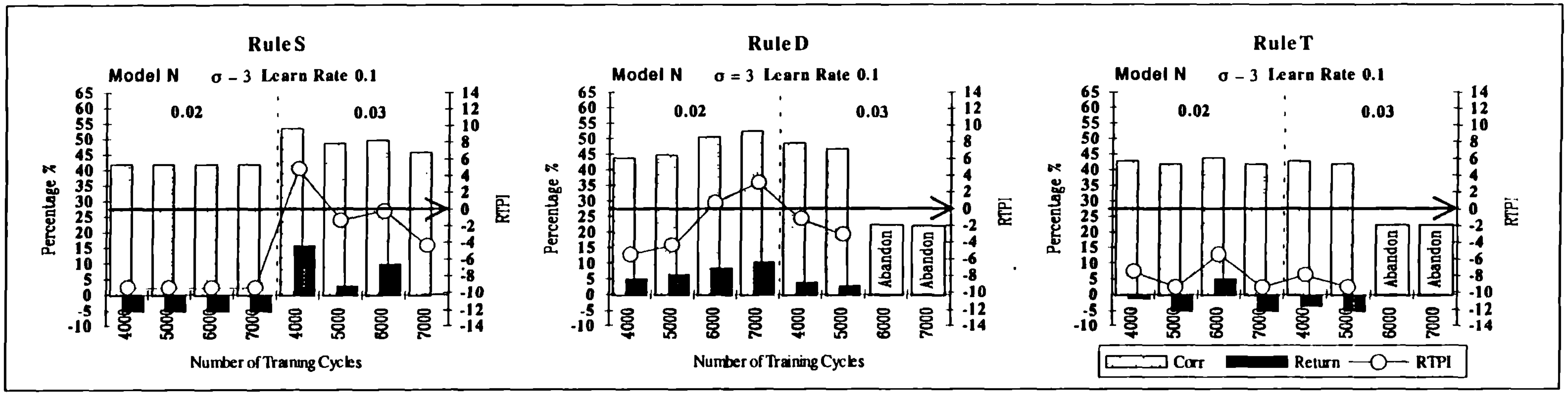


Figure 6.7c **RTPI** Of N Networks With Configurations Using σ = 3

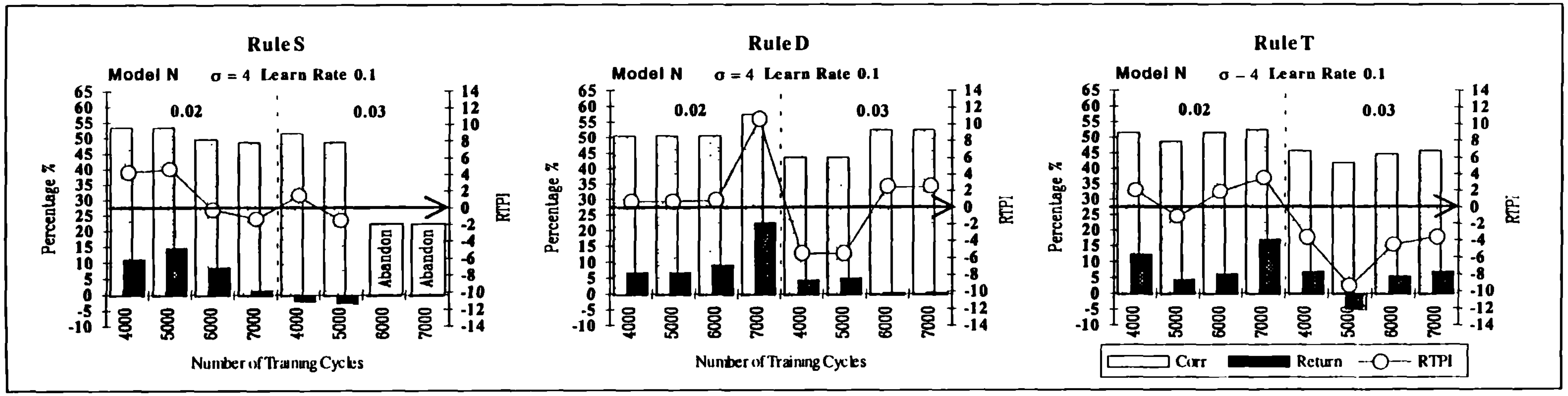


Figure 6.7d **RTPI** Of N Networks With Configurations Using σ = 4

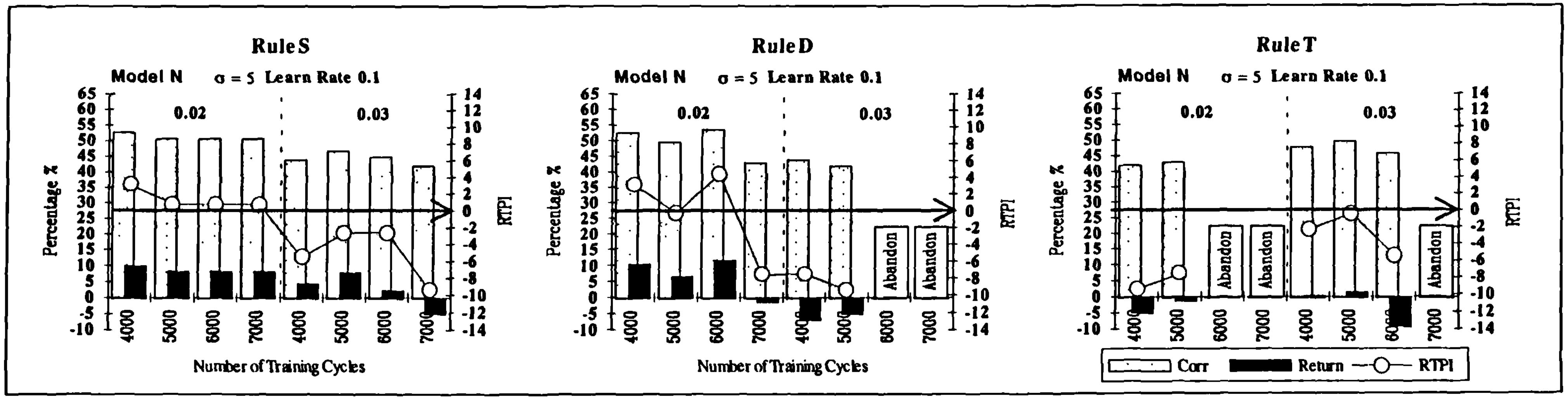


Figure 6.8e **RTPI** Of N Networks With Configurations Using σ = 5