TRUST REGION ALGORTIHMS AND NEURAL NETWORKS FOR FINANCIAL FORECASTING

Zhu Xiaotian

NATIONAL UNIVERSITY OF SINGAPORE
2003

TRUST REGION ALGORITHMS AND NEURAL NETWORKS FOR FINANCIAL FORECASTING

Zhu Xiaotian

A THESIS SUBMITTED
FOR THE DEGREE OF MASTER OF SCIENCE
DEPARTMENT OF INFORMATION SYSTEMS
SCHOOL OF COMPUTING
NATIONAL UNIVERSITY OF SINGAPORE
2003

Acknowledgements

My sincere gratitude goes to my supervisor Professor Paul Kang Hoh Phua, for all his guidance and constant encouragement during all phases of this thesis. I enjoy the many lively discussions that we had about the theory and practice in financial forecasting and artificial neural networks, which contributes to both the managerial and technical insights in this thesis. He has taught me what a good researcher is and has shown me the importance of writing well. He is both a great mentor and good friend.

I would like to thank Professor Chengxian Xu and Dr. Weidong Lin for their tremendous help, especially in development of the financial forecasting software based on trust region neural networks. They have provided me many invaluable suggestions which greatly improve the efficiency of this software. I always turn to them whenever I encounter problems in programming and algorithm. I am grateful to their kindness and assistance.

Parts of this article have been presented at an Information System Workshop at National University of Singapore and the IEEE International Joint Conference on Neural Networks. I would like to thank participants for their comments and valuable discussions.

I thank my research group co-workers for their advice and assistance on my thesis: Xiaohua Wang, Chung Haur Koh and Wei Yu. I thank the Information Systems Research Lab students, Wan Wen, Li Yan and Yang Fan, for their kindness and for their making my stay at the university a pleasant experience. To my parents and sister, I would like to express my thanks, for their love and their encouragement to help me grow in all these years of my life.

Table of Contents

List of Figures

List of Tables

List of Formulae

Summary

This thesis presents a study of using artificial neural networks in predicting stock index increments. The data of five major stock exchange indices, DAX, DJIA, FTSE-100, HSI and NASDAQ, are applied to test our network model. Unlike other financial forecasting models, our model directly uses the component stocks of the index as inputs for the prediction. For the neural network training, a trust region dogleg path algorithm is applied. For comparison purposes, other neural network training algorithms are also considered, in particular, optimization techniques with line searches are applied for solving the same class of problems. Computational results from five different financial markets show that the trust region based neural network model obtained better results compared with the results obtained by other neural networks. In particular, we show that our model is able to forecast the sign of the index increments with an average success rate above 60% in all the five stock markets. Furthermore, the best prediction result in our applications reaches the accuracy rate of 74%. Another major contribution of the thesis is the development of artificial neural network models, including component-based input selection, internal architecture and preprocessing of the sample data. Based on individual and interactive sensitivity analysis on the three major factors in network modeling, our results generalize some valuable recommendations on neural network constructions.

The novel features of the model are the component-based prediction scheme and the introduction of trust region learning algorithms for the network training,

both of which are becoming the key issues in the neural network based financial forecasting. This research may be helpful for both the stock market practitioners and investors.

Chapter 1

Introduction and Thesis Overview

In Chapter one, we provide the motivation for this research and define its scope. Specifically, it addresses the following questions:

1. Are financial markets predictable?
2. What are the currently available technologies for financial market prediction?
3. What are the advantages of artificial neural network in financial forecasting?
4. What is the scope of this thesis?

1. 1 Predictability of Financial Markets

Financial time series forecasting continues drawing considerable attention both within the academic community and from the financial market practitioners. Whether financial market is predictable has been a hot research topic for many years. Generally, there are two main schools of thought in terms of the ability to profit from the equity market. The first school believes that no investor can achieve above average trading advantages based on the historical and present information. In another words, the financial market is unpredictable. The major theory includes the Random Walk Hypothesis and the Efficient Market Hypothesis. The Random Walk Hypothesis states that price on the financial market wanders in a purely random and unpredictable way. Each price change

occurs without any influence by past prices. The Efficient Market Hypothesis states that the markets fully reflect all of the freely available information and prices are adjusted fully and immediately once new information becomes available. If this is true then there should not be any profit for prediction, because the market will react and compensate for any action made from this available information. In the actual market, some people do react to information immediately after they have received the information while other people wait for the confirmation of information. The waiting people don't react until a trend is clearly established. Because of the efficiency of the markets, returns follow a random walk. If these hypotheses come true, it will make all prediction worthless. While, Taylor provides compelling evidence to reject the random walk hypothesis and thus offer encouragement for research into better market prediction [48]. In fact, even the stock market price movements of United States and Japan have been shown to confirm only to the weak form of the efficient market hypothesis. Also, Solnik studied 234 stocks from eight major European stock markets and indicated that these European stock markets exhibited a slight departure from random walk [6]. My research conducted here would be considered a violation of the above two hypotheses for short-term trading advantages in financial markets. The second school's view is that the security prices cannot adjust rapidly to new information. In another words, the current price of a security in financial markets can't fully reflect all the information currently available about the security, thus it's possible to get excess profit above average market return by financial forecasting under technique or fundamental analysis.

1. 2 Artificial Neural Networks for Financial Forecasting

Over the past four decades, the field of artificial intelligence has made great progress toward computerizing human reasoning. Nevertheless, the tools of AI have been mostly restricted to sequential processing and only certain representations of knowledge and logic. A different approach to intelligent systems involves constructing computers with architectures and processing capabilities that mimic the processing characteristics of the brain. The results may be knowledge representations based on massive parallel processing, fast retrieval of large amount of information, and the ability to recognize patterns based on experience. The technology that attempts to achieve these results is called neural computing, or artificial neural networks (ANN).

As an emerging and challenging computational technology, neural networks offer a new avenue to explore the dynamics of a variety of financial applications. Primarily offering time series forecasting and pattern-recognition capabilities, neural networks complement algorithmic, statistical and other artificial intelligence approaches for supporting financial decision-making and problem solving. Their ability to model non-linear dynamics, to deal with noisy data and their adaptability are potentially useful for a wide range of financial decision-making. In recent years, numerous financial applications based on a neural network approach have been developed in various areas such as stock market forecasting, foreign exchange market forecasting, bankruptcy prediction, credit scoring, investment screening and loan underwriting. My

research work is mainly focused on the application of artificial neural networks on financial time series forecasting.

In many financial decision making areas, ANN are supplementing or taking the place of statistical and conventional expert systems (ES) approaches, as the artificial neural networks approach provides features and performance advantages not available in the other types of systems.

The non-linear characteristics of neural networks make them a promising alternative to traditional linear and parametric methods. Generally, one chooses a non-linear model over a linear model when the underlining relationships between the variables are either known to be non-linear, or are not known. Conventional linear techniques cannot capture non-linear patterns and trends in the relationships between and within stock and bond price movements as well as cannot distinguish between random noise and non-linear relationships. Financial markets, such as stock and foreign exchange markets, are affected by many highly interrelated economics, political and even psychological factors, and these factors interact with each other in a very complex manner. Therefore, the movements of financial markets are nonlinear and full of noisy and complex relationships between the variables. So, when comparing with conventional linear statistical models, neural networks may provide a better model to capture the underlying relationships between the financial ratios and the dependent variables.

Neural networks also provide many advantages when comparing with the conventional non-linear parametric models, such as multiple regression and ARIMA: (1). Distributional assumptions are required for error terms for all the parametric models, and regression model in particular. However, as non-parametric models, neural networks can easily incorporate multiple sources of evidence without simplifying assumptions concerning the functional form of the relationship between output and predictor variables. When such statistical assumptions (distribution, independence of multiple features, etc.) are not valid, NNs that do not rely on these assumptions provide better generalization properties and seem to be better suited to handle small sample problems. (2). Parametric statistical models require the developer to specify the nature of the functional relationships between dependent and independent variables. NNs use the data to develop an internal representation of the relationship between the variables so that a priori assumptions about underlying parameter distributions are not required. (3). Most parametric statistical models require that the input variables be linearly separable. When financial ratios and aggregate account balance are used as inputs, this requirement can be easily violated. So, in financial applications, NNs are more suitable to be used than conventional statistical models. (4). Within a parametric model, outliers in a data set influence the size of the correlation coefficient, the average value for a group, or the variability of scores within a group. Those multivariate outliers are even harder to detect since the value for each individual variables are within bounds. There are numerous aspects of NNs that make them more robust with respect to outliers. Research has shown that NNs are more robust than

regression when outliers are present in the data (Marques et al., 1991; Subramanian et al., 1993) [35, 51]

In summary, NNs applied as non-parametric models are not constrained by distribution-related requirements as most traditional statistical models. The non-parametric NNs model may be preferred over traditional parametric statistical models in those situations where the input data does not meet the assumptions required by the parametric model, or when large outliers are evident in the dataset.

Artificial neural networks outperform expert systems in the following four aspects: (1). An expert system (ES) depends on the representation of the expert's knowledge as a series of IF-THEN conditions or rules, known as the rule based approach. The extracting knowledge and rules from the experts presents a very serious bottleneck. While, neural network systems do not exhibit these same shortcomings, primarily they do not require a predefined knowledge base. (2). Furthermore, once the expert system is functional, making even minor changes to the knowledge base can be a complex and expensive process because of the intricate relation between the rules forming the knowledge base. Thus expert systems are generally cost effective only for frequent recurring problems of a very narrow scope that can be solved by a knowledge base that is essentially static. While for neural networks, changes in the problem do not require reprogramming; the system simply retrains itself based on the new information by adjusting nodal weights. Best of all, neural network is fundamentally a dynamic, rather than a static system. The ability of

neural networks to self-organize and to function without a pre-programmed knowledge base gives it an important additional advantage in financial applications – protection of sensitive information. (3). Another problem with expert systems is that ES can't really deal with erroneous, inconsistent, or incomplete knowledge because most ES rely on rules that represent abstracted knowledge of the domain and thus the ES are not able to reason from basic principles. It is also unable to perform effectively when the input information is incomplete, ambiguous (noisy), or partially erroneous. It is in this area that neural networks may offer the clearest advantage over expert systems. Much of the information in real world financial market is noisy, incomplete, and full of error. Neural networks, however, can work with noisy and incomplete inputs and produce the correct output by using the particular ability of generalization. (4). Neural networks are also capable of abstraction – i.e., inferring the "ideal set" from a non-ideal training set. This process involves determining the most prominent characteristics of the training set, then using those characteristics to construct an internal representation of the idea or archetypical pattern. In fact neural networks, unlike ES, can potentially exceed the ability of human experts.

Though neural networks have many advantages that make them outperform other conventional methods, they also have some disadvantages on which we must also pay much attention. (1) A major and inherent problem of artificial neural network is that the internal structure of the neural network makes it difficult to trace the steps by which the output is reached. In other words, NNs can't tell the user how it processes the input information or reach a conclusion.

The output cannot be decomposed into discrete steps or series operations, as would be possible with an ES rule base or any conventional statistical methods. The only way to test the system for consistency and reliability is to monitor the output. (2). The absence of a clearly identifiable internal logic could be a severe stumbling block in the acceptance of neural networks, at least for some applications. Many important business decisions made by human suffer from the same shortcoming. (3). On the other hand, the NNs learning process requires a large number of training examples, hence can involve substantial time and effort. For most conceivable financial applications, especially financial forecasting, ample training examples would be readily available, so relatively little time or effort would be involved in data collection. Furthermore the time and effort required to train NNs would be much less than that required to extract and translate an expert's knowledge base for an ES, as well as less than required to set up a suitable conventional nonlinear statistical model. So, for my particular research area, this disadvantage of NNs seems to be their advantages compared to other methods, for time series data is very convenient to available for financial markets applications.

1. 3 Current Securities Markets for Forecasting

Securities markets, such as stock markets and bond markets, are where buyers and sellers are brought together to transfer securities. Capital market instruments are fixed-income obligations that trade in the secondary market. Bond is the major category of capital market instruments. Common stock represents ownership of the listed firms on the equity markets. Owners of the

common stock of a firm share in the company's successes and failures. On the other hand, many other economics, industrial, political and even psychological factors may also affect the stock prices in an interactive and very complex manner. Thus stock markets are relatively more unpredictable and more risky to invest compared with fix-income securities markets. From this point of view, the research of financial forecasting in the stock markets will be of great significance for both of the market investor and practitioners.

This thesis is mainly focused on the five major stock exchange markets in the world, including four major National Security Exchanges of New York Stock Exchange (NYSE), London Stock Exchange (LSE), Frankfurt Stock Exchange (FSE), Hong Kong Stock Exchange (HKSE) and the largest over-the-counter (OTC) security market in the world, NASDAQ. Security market indices are used to track performance of segments of the market and are commonly used as benchmarks to measure portfolio performance. A good prediction of the indices may do great help for the prediction of the performance of the segment of the corresponding stock markets. The details about indices corresponding to the above five major stock exchange markets are listed in Table 1.

Figure 1 shows the five major stock market indices (including DAX, DJIA, FTSE-100, HSI and NASDAQ) daily close prices from 04-Jan-1994 to 30-Sep-2002. Figure 2, shows the return of these indices daily close prices during the same period. Of this period, the range of daily returns of the five different indices is shown in Table 2. From both of the Table 2 and Figure 2, it's obvious for us to discover that HSI and NASDAQ indices have greater volatilities in

daily return fluctuations than those of DAX, DJIA and FTSE 100. Averagely, both HSI and NASDAQ have about 30% volatilities in returns during the past 8 years, while FTSE 100 having only about 10% during the same time. For stocks and securities that move together with their corresponding index, a reliable predictor of that index would benefit investors and financial institutions in many aspects.

Index Code	Index Full Name	Stock Exchange	Market Location	Market Type	Dominant Weighting Schemes	Number of Component Stocks
DAX	Deutsche Aktienindex Index	Frankfurt Stock Exchange	Germany	National Security Exchanges	Value-Weighted Series	30
DJIA	Dow Jones Industrial Average Index	New York Stock Exchange	United States	National Security Exchanges	Price-Weighted Series	30
FTSE100	Financial Times Stock Exchange 100 Index	London Stock Exchange	Great Britain	National Security Exchanges	Value-Weighted Series	100
HSI	Hang Seng Index	Hong Kong Stock Exchange	Hong Kong	National Security Exchanges	Value-Weighted Series	33
NASDAQ	National Association of Security Dealers Automated Quotation	/	United States	Over-the-Counter Market	Value-Weighted Series	100

Table 1, Details of Major Stock Indices for Forecasting

	DAX	DJIA	FTSE100	HSI	NASDAQ
From	-9.13144%	-7.18304%	-5.43548%	-13.7004%	-9.8574%
To	7.845208%	6.348753%	4.998544%	18.82361%	18.77132%
Volatility	16.97665%	13.53179%	10.43402%	32.52405%	28.62871%

Table 2, Daily Returns Range of DAX, DJIA, FTSE-100, HSI and NASDAQ

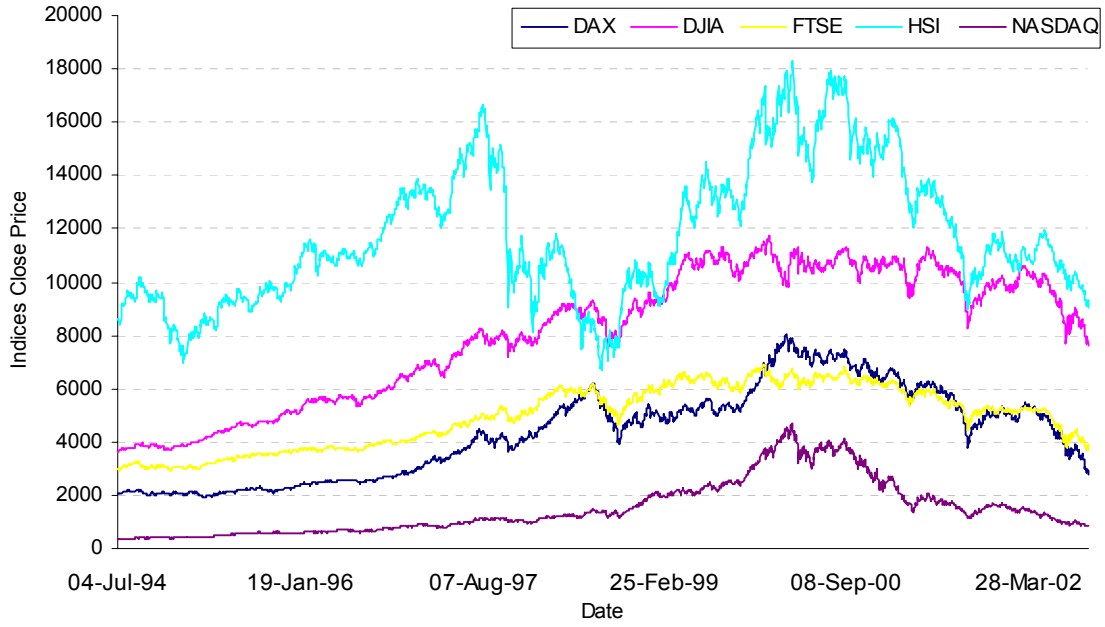


Figure 1, Five Stock Markets Indices Daily Close Prices from 4/1/1994 to 30/9/2002

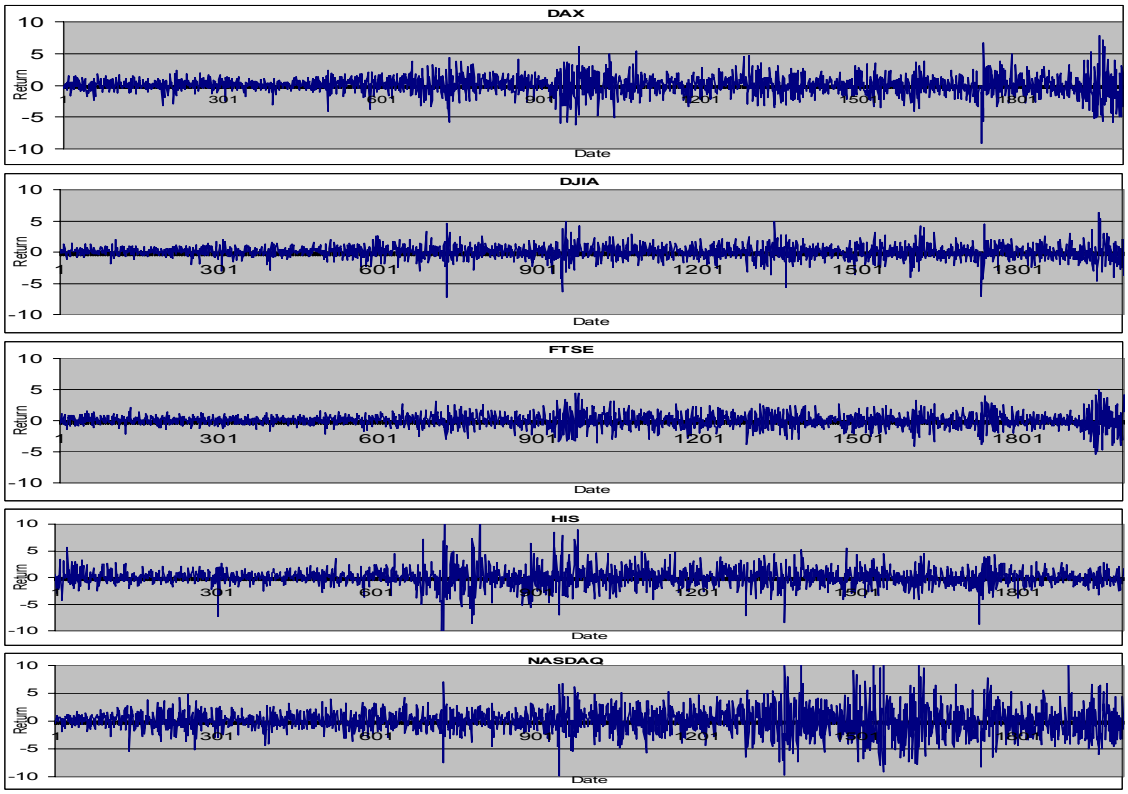


Figure 2. The Returns of the Indices Daily Close Prices from 04-Jan-1994 to 30-September-2002 (From Up to Down the Returns are: DAX, DJIA, FTSE, HSI, NASDAQ)

1. 4 Univariate & Multivariate Models

Practically, financial markets are normally predicted based on fundamental and technical analysis. Neural networks are often trained by both technical and fundamental indicators to produce trading signals. Fundamental and technical analysis could be simulated in neural networks. For fundamental methods, retail sales, gold prices, industrial production indices, and foreign currency exchange rates, etc. could be used as inputs. For technical methods, the delayed time series data could be used as inputs. Depending on what kind of prediction indicators are used, technical, fundamental or both, the existing neural network models could be classified as univariate or multivariate models. A univariate model uses only the technical indicators or past value of the time series for building a forecaster. The disadvantage of the univariate model is that it doesn't consider the environmental effects and interactions among different factors other than outputs. A multivariate model uses both of the technical and fundamental indicators as inputs. In another words, besides the past value of the time series, additional relevant information such as financial ratios/leverage of the company, other securities / foreign exchange market indices, interest rate, etc. are all used to build a forecaster. The disadvantage of the multivariate model is that the selection of inputs has always been a difficult task [24]. To overcome the above difficulties, our research constructs a simple univariate model, which uses only the past values of the component stocks' price time series to forecast the corresponding stock market index increments.

1. 5 Scope of the Thesis

In this thesis, we would investigate the impact of prediction scheme, training algorithm, input selection method, network internal architecture and pre-processing of the sample data on the neural network performance in order to construct a better model for financial forecasting. Specifically, we seek to address the following research questions:

(i) Is it a good way to forecast the stock index increments by directly using the past value of its selected component stocks' price time series?

(ii) What kind of optimization algorithms is more suitable for the neural network training in financial forecasting, the trust region optimisation algorithms or the line search based optimisation algorithms?

(iii) What is the individual impact of each of these three major factors in neural network construction on the network performance?

(iv) Are there any interrelationships between these major factors in network construction? If such interrelationships exist, how do they affect the impact of each factor on the network performance?

To provide some insights into these questions as well as to obtain better prediction accuracy in financial forecasting, we build up a component-based neural network model training by trust region optimization algorithms and

compare it with neural network models training by other kinds of optimization algorithms.

Our proposed model is novel in some aspects. First, we introduced a component-based univariate neural network model to predict the stock index increments. The idea is prompted by the fact that no matter whether the stock index is calculated by value-weighted or price-weighted methods, its price changes are heavily affected by its component stock price's changes. Thus, using the past value of its components stock's time series in addition to its own to build a forecaster is an innovative way for stock index prediction. Secondly, we use a class of trust region algorithms to train the neural networks. Unlike other trust region algorithms, this class of curvilinear search algorithms are applied to solve the trust region problems arising from the unconstrained optimization.

This research aims to make some contributions in the following aspects:

- In addition to a comprehensive survey in neural network applications in finance, we provide an integrated review of the theory and practice in two streams of literature: 1) training algorithms for neural networks, 2) financial market prediction analysis scheme.
- Introducing a component-based neural network forecasting model. Unlike other financial forecasting models, our model directly uses the component

stocks of the index as inputs for the prediction. We show that impressive results could be obtained by this kind of model.

- Applying a class of trust region dogleg path algorithms for neural network training process. Computational results from five different financial markets show that the trust region based neural network model obtained better results compared with the results obtained by neural networks training by other kinds of algorithms. In particular, we show that our model is able to forecast the sign of the index increments with an average success rate that is statistically significant.
- Investigating the impact of three major factors on the network performance by individual and interactive sensitivity analysises. We also generalize some valuable recommendations on the artificial neural network constructions.

This thesis is organized as follows. In chapter 2, we review the literature in neural network based financial forecasting, particularly in the two streams of learning algorithms and prediction schemes. We also give a brief introduction on trust region optimization algorithms and conventional gradient decent optimization methods. In chapter 3, we begin with the basic feedforward neural network model, which is the most commonly used neural network model for a variety of applications in finance and accounting [25]. We develop the model into a component-based financial forecasting model by directly using the component stocks of the index as inputs for the prediction. In chapter 4, we determine the optimal network topology for the purposed model by plenty experiments in five stock markets. Variable sensitivity analysises are also

conducted and some recommendations on neural network constructions are generalized. Computational results and comparisons are given in chapter 5 and it follows by conclusions in chapter 6.

Chapter 2

Literature Review

In chapter 2, we first provide a comprehensive survey on the literature of neural network applications in finance over the last decade. Furthermore, we make a survey on two major streams of literature related to the neural network based stock index forecasting: selection of learning algorithms for network training and stock index prediction analysis scheme.

2.1 General Review of Neural Network Applications in Finance

Artificial neural network is an information processing technology inspired by studies of the brain and nervous systems. After falling into disfavor in the 1970's, the field of neural networks experienced a dramatic resurgence in the late 1980s. The renewed interest developed because of the need for brainlike information processing, advances in computer technology, and progress in neuroscience toward better understanding of the mechanisms of the brain. It was the development of back-propagation in 1986 that enable neural networks to solve everyday business, scientific, and industrial systems and from then on neural networks have been widely applied to many real-world situations. Since the 1990's, the drastic breakthrough of the computing technology has led to an increasing amount of neural network research in the specific field of financial functional applications.

In order to understand the current research situations, as well as the future research trend in neural network applications in finance, we did a comprehensive survey of research works conducted in this field during the last decade (1988 ~ 2002). There are about 253 research articles (123 journal articles, 121 conference proceedings and 9 working papers and doctor dissertations) included in this survey. Of the total of 54 international journals surveyed, five journals published the most papers on neural network applications in finance: Journal of Management Science, European Journal of Operational Research, Decision Support System, IEEE Transactions on Neural Networks and Computer and Operational Research. On the other hand, the proceedings included in this survey are all come from IEEE international conferences (See Appendix A and B for details). A classification of these articles by year reveals that an increasing amount of neural network research has been conducted for a diverse range of financial applications over the last decade. Most of these research findings point out that neural network technology could be successfully used in finance and most of the time is superior to other techniques or technologies.

Figure 3 shows the distribution of articles published by year in the last decade. Overall the amount of research has been increasing in the last decade. It was noted that the number of research studies has increased significantly from 1988 to 1994 and has slightly decreased from 1994 to 1997. On the other hand, after the significant drop in 1998, the applications began to increase continuously again after 1999. The possible explanation for the suddenly drop in 1998 is that researchers are beginning to have more interest to conduct

research in other artificial intelligence techniques, such as genetic algorithm and fuzzy logic. And the new trend of increase appeared after 1999 may be caused by the newly developed interest to integrate neural networks with other techniques such as other artificial intelligence techniques, conventional statistics methods, and Grey theory or Chaos theory.

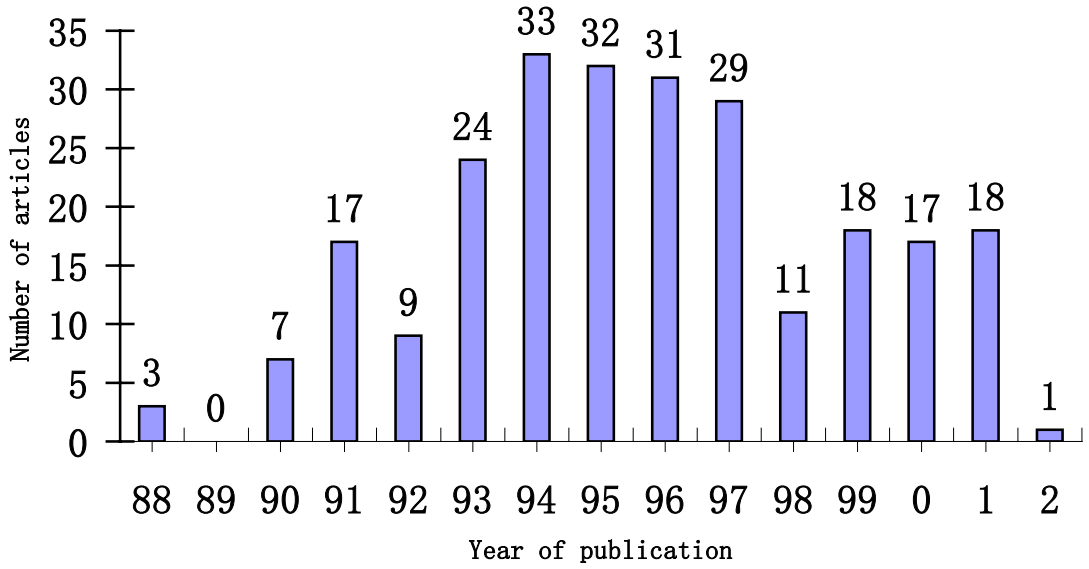


Figure 3 Distributions of Articles by Year

In our research, financial applications of neural networks are classified into 7 main categories: bank management, corporation finance, financial markets, insurance, real estate, risk management and financial regulation. Our survey disclosed that, over the last decade, almost two thirds of the financial applications by neural networks are conducted in the particular field of financial markets forecasting (see figure 4). Furthermore, figure 5 shows that stock market application is the most popular application field among all the six sub

categories in financial market applications, as well as the most popular application field among all the sub applications in finance area.

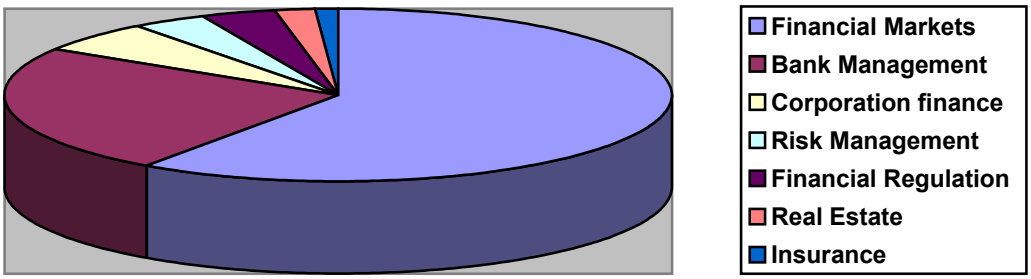


Figure 4 Distributions of Articles by Seven Major Application Areas in Finance

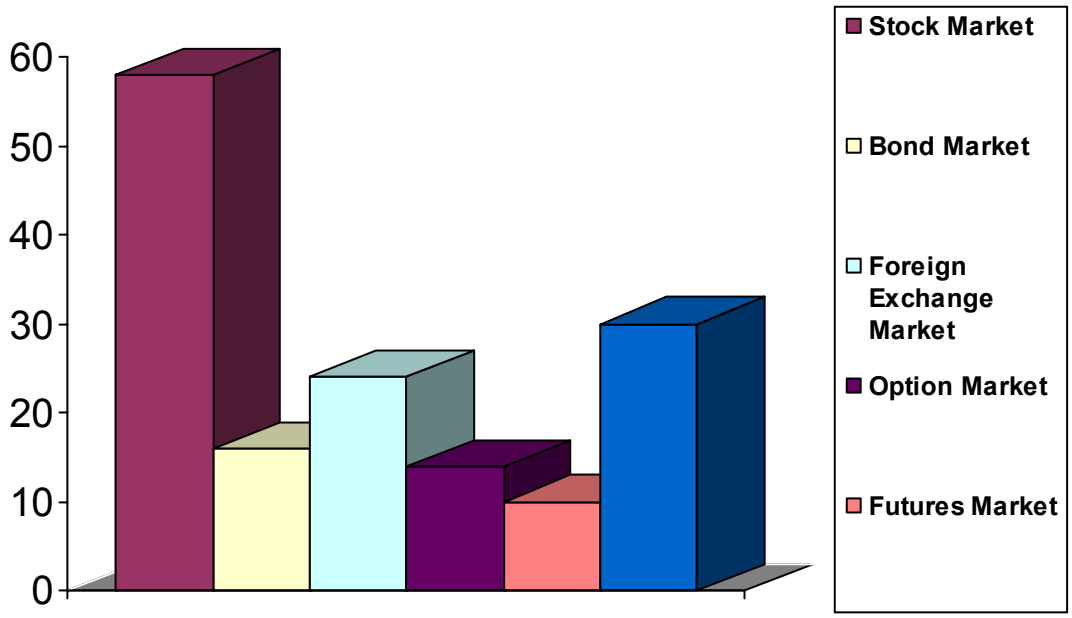


Figure 5 Distribution of Articles by 5 Major Application Fields in Financial Markets

Neural network applications in the field of finance are mainly focused on either time series forecasting or classification, while time series forecasting is more popular in neural network financial applications over last decade. Time series forecasting models assume that there is an underlying process from which data are generated and that the future values of a time series are solely determined by the past and current observations. Neural networks are able to capture the autocorrelation structure in a time series even if the underlying law governing the series is unknown or too complex to describe. Numerous neural network models have been proposed and used for forecasting (Zhang et al., 1998). The most popular and successful model in financial forecasting is the feedforward multilayer network or the multilayer proceptron (MLP). Our research work also makes financial time series forecasting using the feedforward multilayer network.

It seems that the development of neural networks application in finance experienced two main development stages over the last decade. The first stage ranges from 1988 to 1997; during which period, most of the financial application areas are established. Articles published in this period mainly focused on introducing neural networks as a new practical approach in each potential area in financial field, as well as to demonstrate that neural network techniques improve the accuracy or effectiveness of the application, superior to other conventional statistical technologies or at least give some insight in some new/potential areas. In short, researches conducted in this stage are mainly focused on whether neural network technique on its own is superior to other conventional techniques. All the neural networks properties, advantages

and disadvantages are discussed and research in detail. Regression analysis, discriminant analysis, human judgement, logit and ARMA/ARIMA model are the five most common techniques compared with neural networks in financial applications. A very large portion of research has confirmed that the performance of a neural network is better than that of other techniques. The second stage starts from 1998 to now and the trend is still going on. Researches conducted in the second stage are mainly focused on setting up new architectures or training algorithms for the neural networks or even integrating neural networks with various other techniques in the financial applications. For example, networks of many sub neural networks in various structures, fuzzy neural networks and genetic adaptive neural networks are all developed in this period. Though improved accuracy could be obtained by integrating neural networks with other techniques, the hybrid forecasting models or systems are normally too complex or impractical to use. A rule of thumb for obtaining good generalization from a forecasting system is to use the simplest model that will fit the data (Reed, 1993). Under this consideration, our research work introduces a new training algorithm as well as a new scheme for index forecasting in order to improve the network performance instead of setting up a more complex hybrid system.

2.2 Learning Algorithms in Finance Applications

In neural network literature, learning algorithm has attracted considerable attention. Once the decision to use neural network is made, the researcher

must decide which learning algorithm to use. Neural networks effectively filter input to produce output. More specifically, a neural network looks for patterns in a set of examples applied to the input layer of the network, and learns from those examples to produce new patterns, the output. Knowledge within the neural networks is maintained in the weights. The process of learning is implemented by changing the weights until the desired response is attained at the output nodes. In a NN with linear transfer functions, the weights can be derived using matrix manipulation. In a NN with non-linear transfer functions, two learning mechanisms can be used to derive the weights: unsupervised learning and supervised learning. Unsupervised learning is analogous to a cluster analysis approach and is mainly used in classification applications of NNs. Supervised learning accepts input examples, computes the output values, compares the computed output values to the desired output values or target values, and then adjusts the network weights to reduce the difference. The learning process is repeated until the difference between the computed and target output values are an acceptably low value.

The most common supervised learning algorithm is back-propagation (BP) (Rumelhart, 1986). Back-propagation employs a gradient-descent search method to find weights that minimize the global error from the error function. The error signal from the error function is propagated back through the network from the output layer, making adjustments to the connection weights that are proportional to the error. The process limits overreaction to any single, potentially inconsistent data item by making small shifts in the weights. The commonly cited disadvantages of the back-propagation learning algorithm are

that the training time usually grows exponentially as number of nodes increases and there is no assurance that a global minimum will be reached. To overcome the convergence problem, a very small step size or learning rate must be used to guarantee asymptotic convergence to a minimum point. But small learning coefficients lead to slow learning. The momentum factor was added to back-propagation algorithm to act as a low-pass filter on the weight adjustment terms. It allows a low learning coefficient with faster learning. But incorrect specifying of learning rate and momentum term can lead to either extremely slow convergence or to oscillatory behaviour without convergence. Thus further researches are conducted to improve the back-propagation algorithms under two directions; either speed up the converging procedure or assist in constructing globally convergence.

The majority of the accelerating techniques can be classified as conjugate-gradient (CG) methods or quasi-Newton (QN) algorithms. Leonard and Kramer improved the BP algorithm with conjugate gradient methods in 1990 [32]. This algorithm uses the second derivative of the error function to exploit information regarding both the slope and curvature of the response surface. When compared to BP, the conjugate-gradient algorithm has been shown to produce comparable results with much faster training times (Charitou and Charalambous, 1996; Wong 1991) [12, 53]. Ballo also speedup the training procedure of BP algorithm by using a nonlinear least-square optimization algorithm enhanced by a quasi-Newton (QN) algorithm in 1992 [8]. QN methods were criticized because they require more computation time and memory space to update the Hessian matrix (Watrous 1987; Nahas 1992).

Various modified QN methods are therefore proposed recently for training neural networks in order to speed up the rate of the convergence and/or to reduce the required memory space (Sectiono and Hui 1995; Robitaille 1999; Denton and Hung 1996; McLoone and Irwin 1999). Most recently, Phua and Ming (2000) proposed a class of parallel nonlinear optimization techniques based on QN methods to improve the rate of convergence of the training procedure for neural networks. Besides accelerating the convergence speed, computational results also show that this parallel QN method outperforms other existing methods by far. Our research work will taken this class of parallel QN methods as benchmark to compare with our proposed learning algorithms in neural network training process for real financial time series forecasting.

A major drawback of the gradient descent algorithm is that there is no way of determining in advance whether the architecture and selected methods will converge. Real error surfaces, with multiple weight dimensions, tend to have very complex features including dents and ravines. Although the gradient-descent method always follows the steepest path towards the bottom of the error surface, it may get a stuck within a large dent or ravine on the side of the surface. Numerous methods are available to compensate for this tendency to find local minima. Some methods adjust the weight derivation process to maintain the momentum established by previous adjustments. Other methods involve starting from a different point on the error surface by using a different set of initial weights and ensuring that the results are similar. Still other

methods involve the dynamic adjustment of the network architecture by trimming modes or connections between nodes.

Evolutionary programming is a stochastic optimization technique that has been used in finance problems as an alternative to the conventional gradient methods. Evolutionary programming involves two processes – mutation and selection. The algorithm starts with an original population of weight sets that are evaluated by examining the corresponding outputs. Random mutation of the parents creates new solutions. Specifically, a Gaussian random variable with mean zero and variance equal to mean squared error of the parent perturbs each weight and bias term. Each offspring is evaluated, and the n 'best' solutions are retained as parents for the next iteration.

Genetic algorithms extend this mutation and selection process by adding a recombination phase to create the child notes that are evaluated. Each child is formed as a cross between two parents. Goldberg (1994) cites numerous advantages of genetic algorithms: they can easily solve problems that have many difficult-to-find optima, they are noise tolerant, and they use very little problem-specific information. They work with the coding of parameter set, not the specific value of the parameters. The major disadvantage of cited for genetic algorithm is the difficulty in specifying the optimal parameter settings for the model. A genetic algorithm was used by Huang (1994) to predict financially distressed firms. Levitan and Gupta (1996) applied a genetic algorithm to the cost driver optimization problem in activity-based costing. By

far, genetic algorithms have been demonstrated as possible techniques to aid in the development of the neural network model (Back, 1996; Hansen 1998).

Optimal Estimation Theory has also been applied to finance problems as an alternative learning algorithm to back-propagation. Optimal Estimation Theory introduced by Shepanski (1988), uses a least squares estimator to calculate the optimal set of connection weights for the presented training set. This method significantly reduces the time required to train a network, but it is not known whether it achieves similar performance (Boucher 1990; Boritz 1995)

Our review of the previous literature suggests that much of the literature on neural network learning algorithm in last decade are mainly focused on line search based optimization algorithms such as BP, modified BP, CG and QN methods. To our knowledge, there has been no research introducing the class of trust region optimization algorithms in the neural network training process for financial forecasting. As an alternative to the conventional line search based gradient methods, trust region methods are a class of algorithms for the solution of nonlinear nonconvex optimization problems that covers both unconstrained and constrained problems. In this study, we seek to bridge the gap between the literature on the trust region optimization algorithm and the financial forecasting by neural networks. Performance comparison will be conducted between the neural network models training by trust region algorithms and conventional gradient descent algorithms.

2.3 Stock Index Forecasting with Neural Networks

Reference [3] indicates that conventional statistical techniques for forecasting have reached their limitation in applications with nonlinearities in the dataset. Artificial neural network, a computing system containing many simple nonlinear computing unites or nodes interconnected by links, is a well-tested method for financial analysis on the stock markets [55]. Neural networks have been shown to be able to decode nonlinear time series data which adequately describe the characteristics of the stock markets [2]. In the past decades, neural networks have been explored by many researchers for financial forecasting. Among them, some researches are conducted particularly on forecasting the value of a stock index [1, 3, 5, 23, 29, 39, 40, 44-46, 49, 50, 52, 54].

Wittkemper (1996) conducted a comparative study between seven traditional forecast models based on regression and averages with two different types of neural network models in forecasting the systematic risk as well as the market index of German stock markets. In the analysis, 67 most traded stocks in the German markets are considered. For each stock the daily stock market data or yearly financial statements from the period 1967 to 1986 are used. From the financial statements of the companies, totally 32 financial statement variables, such as operation/financial leverages, debt/equity ratios, growth rate, etc, are considered as the inputs for the neural network forecasting. Neural networks outperformed all the traditional models in the one-step-ahead forecasting. For the general regression neural network models, the one using the historical price data as the only input variable performed better than the other one using all fundamental financial variables. But, when genetic algorithms are used to

choose the optimal inputs from the fundamental variables for the network model, the resulting model outperformed all other existing models in comparison.

Antonio and Claudio (1996) applied neural networks to forecast the general index of share prices at the Santiago de Chile Stock Market. Time series with daily values of the index and of total amount of transactions were used to train the neural networks. A complex multilayer architecture containing two kinds of memories was used to design the neural network. Compared to traditional simple architectured neural network model as well as other statistical methods, this model produced better results in stock index forecasting. They also show that a time delay of ten labor days was sufficient to forecast.

Yao and Poh (1998) reported the results for Kuala Lumpur Stock Exchange indices forecasting by popular used backpropagation neural networks. Time series of both stock index value and technical indicators were used as the inputs for the neural network forecasting. Based on the out-of-sample results, they found that for daily data, neural networks were much better than conventional ARIMA models. However, if weekly data were used, the neural networks did not show much improvement over the ARIMA model. The experiment shows that useful predictions can be made without the use of extensive market data or knowledge.

Steven and Chun (1998) examined the out-of-sample performance of feedforward, recurrent and probabilistic neural networks in forecasting the

Singapore Stock Exchange index. Besides stock index price, total return index, dividend yield, turnover by volume and price/earnings ratio are all considered as the inputs for the network forecasting. The daily values consisting of 3056 observations were used in their investigation; with the last 186 data points retained as out-of-sample testing periods. Their results showed that the arrayed probabilistic network tended to outperform recurrent and back propagation networks. However, case based reasoning tends to outperform the arrayed probabilistic network as well as the other techniques when mistakes were taken into consideration.

Renate and Joaquin (2000) assessed the short-term predictive ability of the feedforward time delay neural networks in forecasting the Standard & Poor's 500 index. The S&P 500 index data used in this study covers 22 years, from 1973 to 1994. Different time delayed time series of stock indices are used together as inputs for the neural network forecasting. This study suggested that there are no short-term correlations in this stock market time series, which is consistent with conventional statistical analysis.

In order to increase the forecastability in terms of profit earning, Yao and Tan (2000) developed a profit based adjusted weight factor for backpropagation network training. Instead of using the traditional least squares error, they add a factor which certains the profit, direction and time information to the error function. Four major Asian stock market indices, Hong Kong Heng Seng Index, Malaysia Kuala Lumpur Composite Index, Japan Nikkei-225 and Singapore Straits Times Industrial Index together with the world economic benchmark

American Dow Jones Industrials Index are applied to this profit based adjusted network model. For each stock index, only the time series of historical index price are fed into the neural network model to make the one-day-ahead index forecasting. The results show that this new approach does improve the forecastability of neural networks.

In a recent time series prediction application, the data sets used were series of S&P 500, NASDAQ and Dow Jones Industrial Average Indices for the period of 1990-2000. Filippo (2000) reported the experience of forecasting the price value increments of these time series with backpropagation neural networks. One-step-ahead forecasting was made by feeding only the delayed index price time series to the network model. They show that a neural network able to forecast the sign of the price increments with a success rate slightly above 50% can be found.

Liu and Yao (2001) developed an evolutionary neural network approach for Hong Kong Heng Seng stock index forecasting. In this approach, a feedforward neural network is evolved using an evolutionary programming algorithm. Both the weights and architectures are evolved in the same evolutionary process. The network may grow as well as shrink. Only the historical time series data of Heng Seng index are used for the one-step-ahead index forecasting. Experimental results show that the evolutionary neural network approach can produce very compact neural networks with good prediction.

Several observations are warranted. First, none of the previous studies in stock index forecasting considered the component stock prices as the inputs for the neural network based index forecasting. Besides the delayed value of the stock index itself, fundamental indicators (such as various financial statement variables, total trading volume of the markets and general economic indicators, etc) and technique indicators are often considered as the additional inputs for the network index prediction in the previous researches. Our study would show that component-based prediction scheme could also produce impressive results in the neural network based stock index forecasting. Second, both multi- and one-step-ahead forecasting methods with different forecast horizons are examined in the literature. Multistep forecasts are useful for long-term forecasting and for the identification of major turning points in the stock index data. Single-step predictions are desirable for making short term buy or sell decisions. Single-step forecasting is also a good instrument for evaluating the adaptability and robustness of a forecasting technique (Refenes, 1993). Finally, the neural network out-of-sample performance is mixed, though most of previous studies show that neural network outperformed other techniques in dealing with nonlinear time series forecasting. We also notice that in the literature that there is no universally agreed upon set of performance measures. Both absolute and relative forecasting measures of performance have been employed.

Chapter 3

Component-Based Forecasting Models

In this chapter, we begin with the basic feedforward neural network model. We outline the basic structure, the computational scheme as well as the objective function of this model. In the consideration that indices are always directly or indirectly affected by its component instruments even under different kinds of stock market indicator weighting schemes, we further extend the basic model to a component-based financial forecasting model. It different from the basic model in several ways: First, this model is particularly designed for financial indices forecasting. Second, we directly use the component instruments of the index to forecast the future index value. Third, a class of trust region algorithms that can solve both definite and indefinite optimisation problems are used for the network training.

3.1 Basic Feedforward Neural Network Model

For the purpose of conducting experiments, we choose a three-layered feed forward neural network architecture as the basic financial forecasting model, which is most commonly used in finance and accounting applications. The Figure 6 shows the structure of a basic feedforward neural network model. The input, hidden and output layers are noted as {X, H, Z} respectively. Here the input layer X has (m+1) neurons with X_0=1, the hidden layer H have (n+1)

neurons with H_0=1, and the output layer Z has k output values. Let $w^{(L)}$ denoted the weights at Level L.

Let (X_p, Y_p), $p=1,2\ldots P$ be the set of given input/output vectors for training the neural network. For each input X_{pi} $(i=0,1\ldots m)$, the neurons H_{pj} and Z_{pk} are calculated according to the following equations:

$$H_{pj}(w^{(1)}) = f^{(1)}(\sum_{i=0}^{m} X_{pi} w_{ij}^{(1)}), \qquad \text{j = 1, 2, …, N} \tag{1}$$

$$Z_{pk}(w^{(1)}, w^{(2)}) = f^{(2)}(\sum_{j=0}^{n} H_{pj}(w^{(1)}) \bullet w_{jk}^{(2)}), \quad \text{k=1,2,…, K} \tag{2}$$

Here, the transfer function $f^{(1)}$ from the input to the hidden layer is the sigmoid function $y=1/(1+e^{-x})$, and the transfer function $f^{(2)}$ from the hidden layer to the output layer is the linear function $y=x$. The training of the neural network is done by feeding the set of input-output vectors (X_p, Y_p) to the neural networks and by minimizing the following objective function:

$$g(w) = \frac{1}{P \times K} \sum_{p=1}^{P} \sum_{k=1}^{K} [Y_{pk} - Z_{pk}(w)]^2 \tag{3}$$

where $w=w^{(1)} \cup w^{(2)}$, represents the weights of the neural network. The error function g defined by (3) is the Mean-Squared Error (MSE).

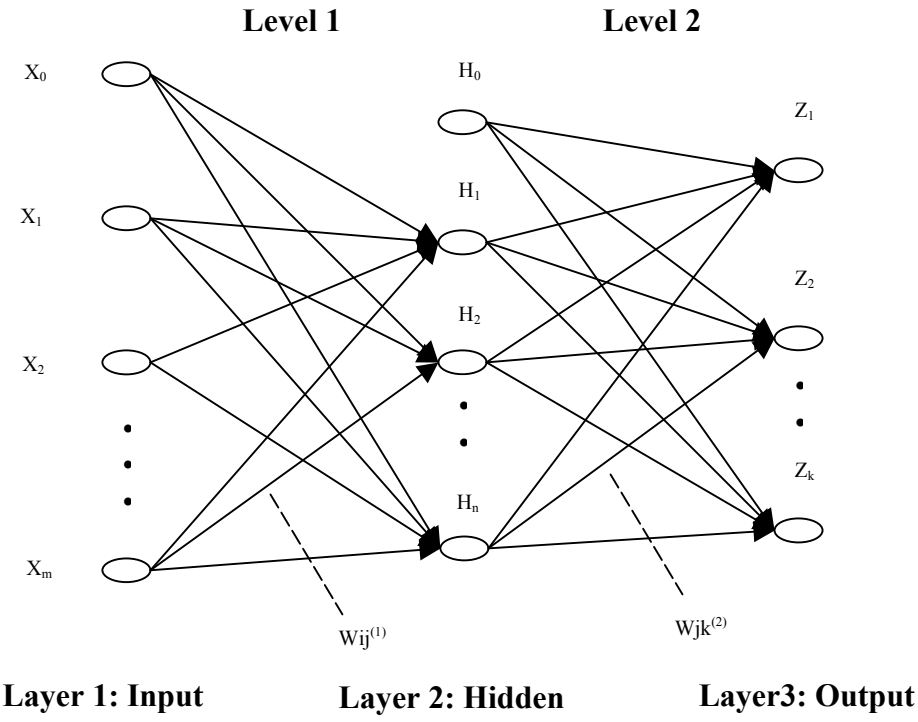


Figure 6. The Basic Structure of the Feedforward Neural Networks

3.2 Types of Stock Market Indices

The three predominant stock market indicator series weighting schemes are: price weighted, value weighted, and unweighted.

1). Price-Weighted Series. A price-weighted series is an arithmetic average of current prices; thus index price movements are directly influenced by the differential prices of the components. Computationally, a price-weighted index adds together the market price of each stock in the index and then divides this total by the number of stocks in the index. Because the index is price-weighted, a high-priced stock carries more weight than a low-priced stock. The

divisor must be adjusted for stock splits and other changes in the portfolio. Let $I_{t(PW)}$ be the price-weighted stock market index at time *t*, and *m* be the number of component stocks of $I_{t(PW)}$. Then $I_{t(PW)}$ can be computed by

$$I_{t(PW)} = \sum_{j=1}^{m} w_j p_{tj} \tag{4}$$

where P_{tj} is the price of the component stock *j* (C_j) at time *t*, and w_j is the price weighting coefficient for C_j. The major price-weighted index is the Dow Jones Industrial Average index (DJIA).

2). Market Value-Weighted Series. A market value-weighted series is calculated by summing the total value (current stock price times the number of shares outstanding) of all the stocks in the index. This sum is then divided by a similar sum calculated during the selected base period. This ratio is then multiplied by the index's base beginning value. Let $I_{t(VW)}$ be the market value-weighted stock market index at time *t*, and *m* be the number of component stocks of $I_{t(VW)}$. Then $I_{t(VW)}$ can be computed by

$$I_{t(VW)} = \frac{\sum_{j=1}^{m} N_j P_{tj}}{\sum_{j=1}^{m} N_j P_{Bj}} \times I_B \tag{5}$$

where P_{tj} is the price of the component stock *j* (C_j) at time *t*, P_{Bj} is the component stock *j* (C_j) at base year, and N_j is the number of shares outstanding at time *t*. I_B is the base beginning value of the index. DAX, FTSE100, HSI and NASDAQ are all the major market-weighted indices.

3). Unweighted Price Indicators Series. In an unweighted price indicator series, all stocks carry equal weight regardless of their price or total market value. A $20 stock is just as important as a $400 stock and a small-size company is just as important as a large-sized company. Here it is assumed the investor makes and maintains an equal dollar investment in each stock in the index. In effect, you are working with percentage price changes. The price of an unweighted index may be calculated using two methods of arithmetic average or geometric average.

3.3 Component-Based Feedforward NN Forecasting Model

It's obviously to notice that no matter what kind of indicator series weighting scheme are used, stock indices are always computed directly or indirectly from the values of their component instruments. So, when internal/external information related to a particular component stock is perceived, the price of that stock will change, and this will cause the corresponding index to change as well. Taken the National Association of Securities Dealers Automated Quotation System (NASDAQ) index as example, figure 7, shows the relationship between the index and its component stocks. Hence it is natural and logical to predict a market index by considering the prices of its component stocks. For predicting a general financial market index, we propose the following component-based forecasting model:

$$I_{t+1} = f(C_{1t}, C_{2t}, ..., C_{mt}, I_t) \quad (6)$$

where C_{1t}, C_{2t},…, C_{mt} are the closing prices of the component stocks C_1, C_2, ….., C_m at time t, while I_t and I_{t+1} are values of the market indices computed at time t and t+1, respectively. The component-based forecasting model based on the basic structure of feedforward neural networks is called the Component-Based Feedforward Neural Network Forecasting Model. Such model directly uses the past value of the selected component stock prices time series as inputs for the feedforward neural network model to predict the corresponding index. Figure 8 illustrates the structure as well as the particular input scheme for this model.

As an application of the component-based neural network forecasting model, we apply the one-day ahead prediction for the five different stock market indices: DAX, DJIA, FTSE-100, HSI and NASDAQ. For each market index, we consider all stocks that served as component stocks of that index during the period of 4 January 1994 to 30 September 2002. Although some stocks served as component stocks for the corresponding index for only a part of the period, we also treat these stocks as potential candidates of inputs to our model.

In this thesis, our proposed neural network model is trained by applying the Trust Region Indefinite Single Dogleg Path (TRISDP) algorithms proposed in [28]. Unlike other trust region algorithms, this class of curvilinear search algorithms are applied to solve the trust region problems arising from the unconstrained optimization. The curvilinear paths set by this algorithm are dogleg paths, generated mainly by employing Bunch-Parlett factorization for general symmetric matrices, which may be indefinite. These algorithms are

easy to use and they are shown to be globally convergent. It is proved that these algorithms satisfy the first- and second-order stationary point convergence properties and that the convergence rate is quadratic under some common conditions [27]. We will show that this kind of trust region algorithm is robust and efficient in solving all the test problems. We refer the neural networks training by this class of algorithms as Trust Region Neural Networks (TRNN).

For comparison purposes, we adopt another class of training algorithm for MLP neural networks, known as Self-Scaling Parallel Quasi-Newton (SSPQN) algorithms, proposed in [39], for solving the same set of test problems. Computational results are presented in Chapter 5.

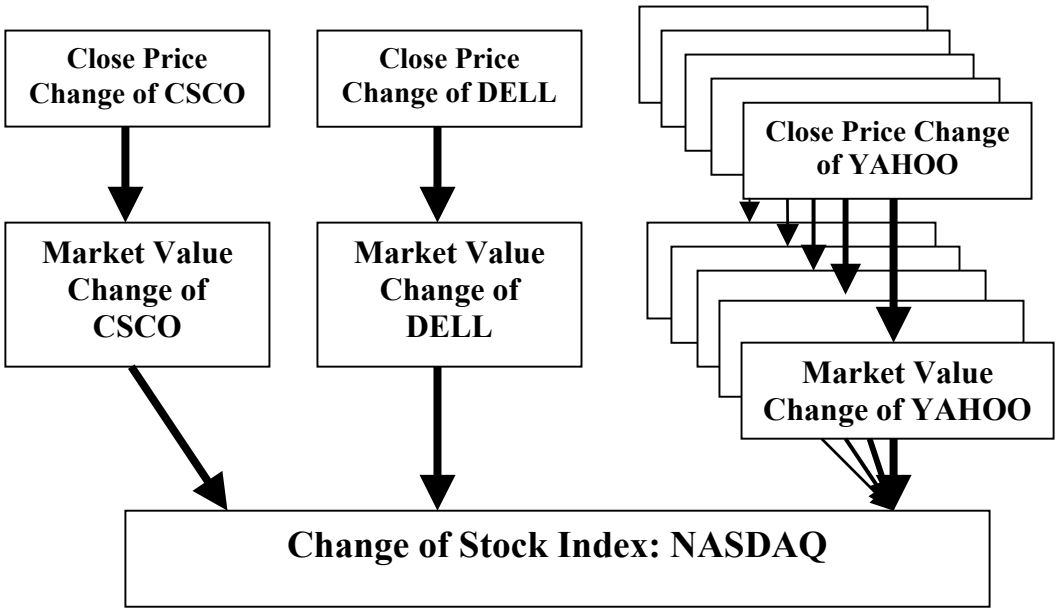


Figure 7, Relationship between the NASDAQ Index and its components

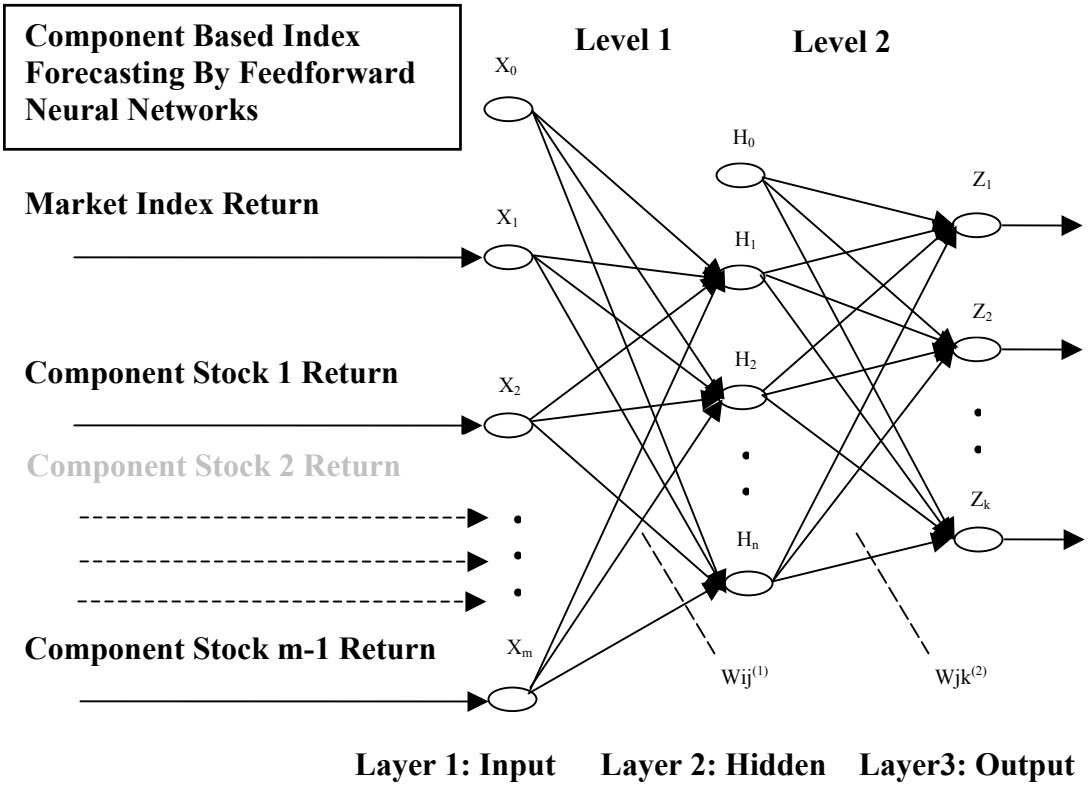


Figure 8, Structure of the Component-Based Feedforward Neural Networks Forecasting Model

Chapter 4

Determining Optimal Network Topology

Once the decision to use a neural network model is made, the researcher faces numerous decisions in the network constructions. In this Chapter, we first determine the optimal number of iterations on each stock market under different forecasting models and then determine the optimal network topology based on the variable sensitivity analysis in the network constructions.

Besides the training algorithms, three major factors that may have great impact on the neural network performance are carefully considered in the process of neural network construction: component-based input selection, internal architecture and pre-processing of the sample data. First, we would conduct the individual sensitivity analysis in order to learn the impact of each of these factors on the network performance. Then, interactive sensitivity analysises are introduced in order to learn whether interrelationships between these factors may affect their impact on the network performance. Based on the results getting from both of the sensitivity analysises, we would draw some general recommendations on the network constructions.

4.1. Determining optimal number of iterations

Besides considering MSE results, the secondary mark for measuring the model performance in our research is the Directional Symmetry (DS). Following Caldwell (1995), [11], the directional symmetry metric is defined as:

$$DS = \frac{100\sum_{p=1}^{P} d_p}{n} \tag{7}$$

Where

$$d_p = \begin{cases} 1 & if\ (Y_p)(Z_p) \geq 0 \\ 0 & Otherwise \end{cases} \tag{8}$$

As mentioned in Section 3.1, the variable Y_p is the set of given output vector for training the neural network. Z_p is the corresponding predicted return computed by the network. As the inputs we used are all stock or index returns, the DS shows the percentage of correctly predicted directions with respect to the stock index. It has more value than MSE in the application field of financial forecasting. Normally, institute investors and individual practitioners in financial markets have more interest and pay more attention on the accuracy of the direction prediction on the market index instead of MSE, because accurate direction prediction may do great help on making correct and profitable investment decisions. Most financial trading strategies in practice are mainly based on the prediction of direction up or down in the coming stage [17].

From our experiments, we noticed that the MSE (or DS) results for testing would reverse its trends from decreasing to increasing (or increasing to decreasing) at some particular iteration number in the training process. Thus the training process of our network will be terminated when the MSE (or DS) results for testing reverse its trends. In practical, 30 different number of

iterations (Iter = 5, 10,…, 95, 100, 110 , 120,…, 210) are considered for training in five different stock markets by TRNN and SSPQN component-based forecasting models. Thus in total 300 experiments (30×5×2) are conducted to determine the optimal number of iteration for different markets and neural network models.

Experiment results from TRNN model in figure 9 (a)-(d) show how the MSE and DS vary for different number of iterations both in the training and testing process. Theoretically, if the iteration number is sufficiently high and there are enough hidden neurons, the MSE on training data could be very low and even reach zero. Research suggests that an architecture with n input data streams will require at most (2n+1) processing nodes per hidden layer to achieve the desired accuracy. It is also possible to approximate a continuous function that may achieve the desired accuracy with a single hidden layer (Cybenko, 1989; Hecht Nielsen, 1990; Hertz et al., 1991; Hornik et al., 1989). Generally speaking, as the number of hidden nodes in a network is increased, the number of variables and terms are also increased. If the network has more degrees of freedom (the number of connection weights) than the number of training samples, it's easy for the network to accurately simulate the training samples. This is similar to fitting a small number of points by a high-order polynomial. Training of the neural network involves propagating the error to adjust the set of weights to minimize the error function. The Trust Region Dogleg Algorithm we proposed in this thesis guarantees that total error in the training set will continue to decrease as the number of iterations increases and this method is globally convergent. With each iteration, the weights are

modified to decrease the error on the training patterns. As training processes, the amount of change in the error function becomes smaller. Training with repeated applications of the same data set may result in exactly fitting the limited set of points. So, in short, the MSE on the training data can finally go to zero if the iteration number is sufficiently high and there are enough hidden neurons. While, in practice, researchers or practitioners always set a desired accuracy for the training process as a stop criterion in order to avoid the phenomenon of overfitting or overtraining. That is, when change in the error function is less than a specified threshold or when the error function value reaches the desired accuracy, the convergence occurs and the training process will be stopped. Simply pursuing zero MSE or error function results in training process has no meaning for the prediction or testing process. A good balance between accurately fitting the training set examples and still providing a reasonable good interpolation capability should be determined by experiments. As shown in Figure 9(b), our experimental results show that as iteration number increases in training process, the MSE results decrease continually. While we also noticed that with the increment of iteration number, the testing MSE results not decrease continually, instead, it always reverse its trend from decreasing to increasing at some particular point of training iterations (see Figure 9 (a)). Overtraining occurs when neural network attempts to exactly fit the limited set of points and loses its ability to interpolate those points. In the first stage of training, the network learns the underlying relationships in the data samples and with the increase of iteration number, in the second stage of training, the network begins to learn the noise in the training samples, which will lead to exactly fitting the limited sample data while

losing the capacity for accurate prediction for out-of-sample data in testing process. Thus, it's necessary to stop the training process after the MSE results reverse its trends in testing process. As shown in figure (c) and (d), DS results for training always increase. However, DS results for testing have some fluctuations. For all the 5 stock markets, the DS results for testing usually increase first, and then decrease after certain number of iterations, which is different for different markets. Our experiments show that the best iteration number for testing MSE results is usually not the best one for testing DS results for the same stock market.

By experiments, we find out the optimal iteration number for both the neural network models under the two criteria of MSE and DS. Table 3 shows the results of the optimal iteration numbers. From the results, we can see that on average the optimal iteration numbers for TRNN model are larger than those for the SSPQN model under both the MSE and DS criteria. These results may show that, on average, SSPQN neural networks model have faster convergence speed than the TRNN model. The following experiments to determine the optimal network topology in this study are all conducted with these optimal numbers of iterations for corresponding models and stock markets. For all the MSE results shown in the following many figures in Chapter 4, the number of iterations in training process is based on the corresponding optimal iteration number, which is pre-determined in this section. As both the training sample size and network architecture will determine the optimal iteration number in training process, we averaged the optimal results for different combination of these variables by experiments to

determine the optimal iteration number for each stock market under different models. So the optimal iteration numbers listed in Table 3 are all the averaged optimal results. A very interesting finding is that the optimal iteration numbers show great difference for different network model as well as for different stock markets. It seems that the optimal iteration number is very sensitive on the sample data and training algorithms for neural network based financial forecasting. Why the experimental results show so much difference would a very interesting research issue in our future research.

Table 3. Optimal Number of Iteration for Each Market and Neural Network Model Based on Two Criteria

Markets	**TRNN**		**SSPQN**	
	MSE	**DS**	**MSE**	**DS**
DAX	65	10	100	10
DJIA	150	180	120	95
FTSE-100	140	95	95	15
HSI	150	120	65	75
NASDAQ	140	55	55	160
Average Result	129	92	87	71

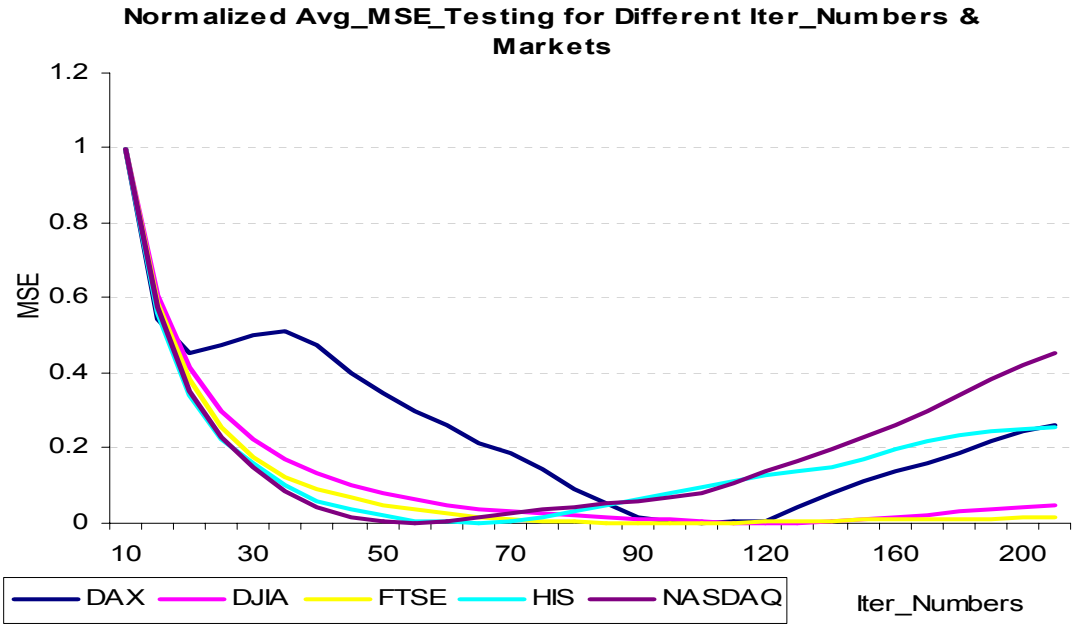


Figure 9(a), MSE Results in Testing Process for 5 Markets during the Increase of Iterations

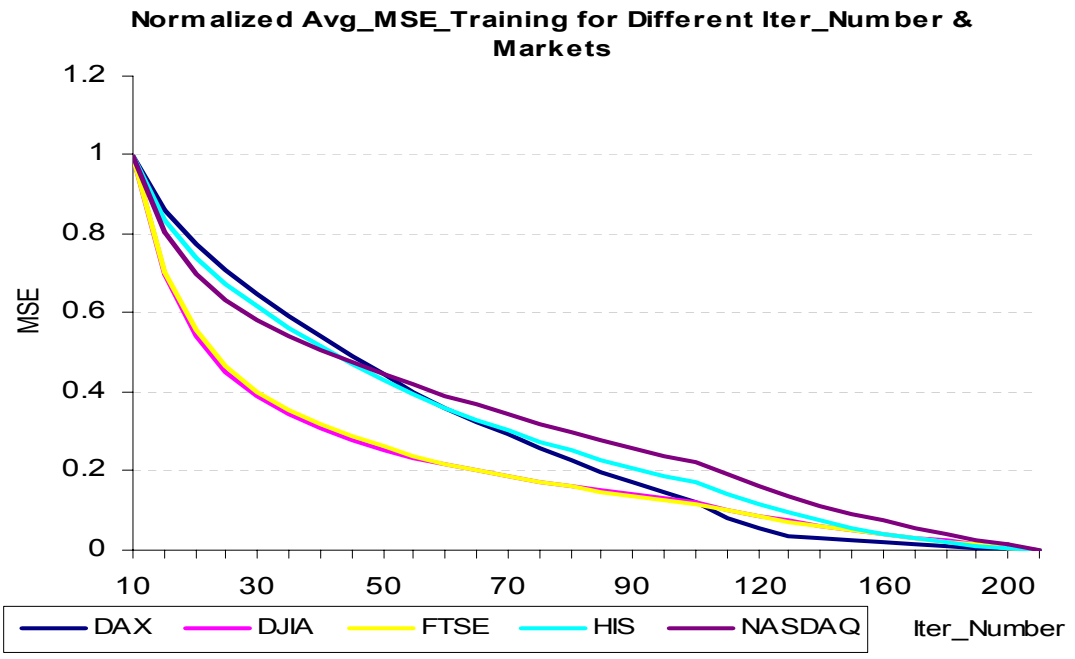


Figure 9(b), MSE Results in Training Process for 5 Markets during the Increase of Iterations

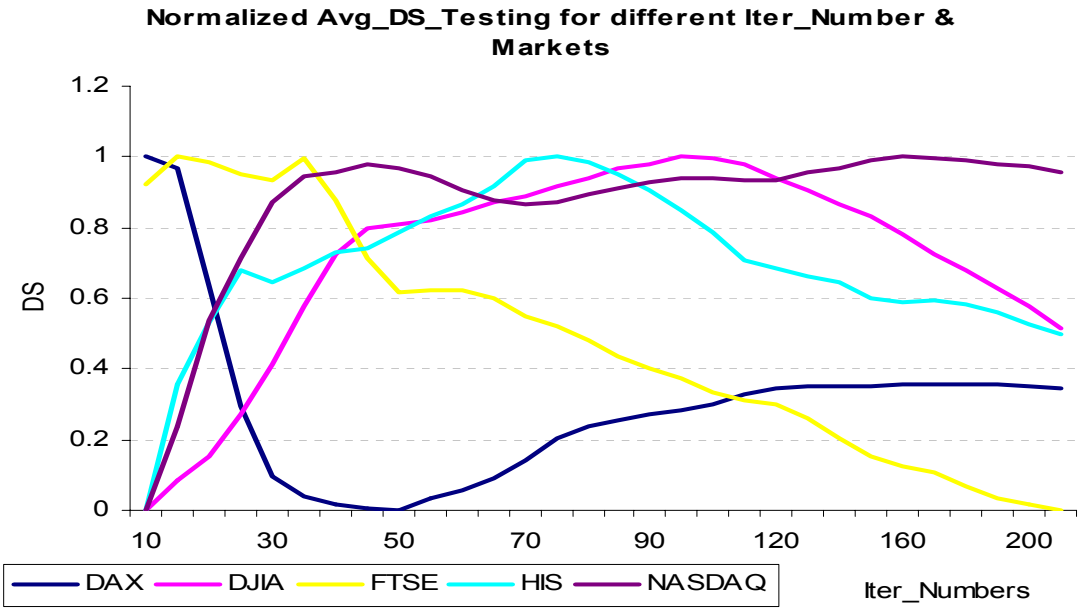


Figure 9(c), DS Results in Testing Process for 5 Markets during the Increase of Iterations

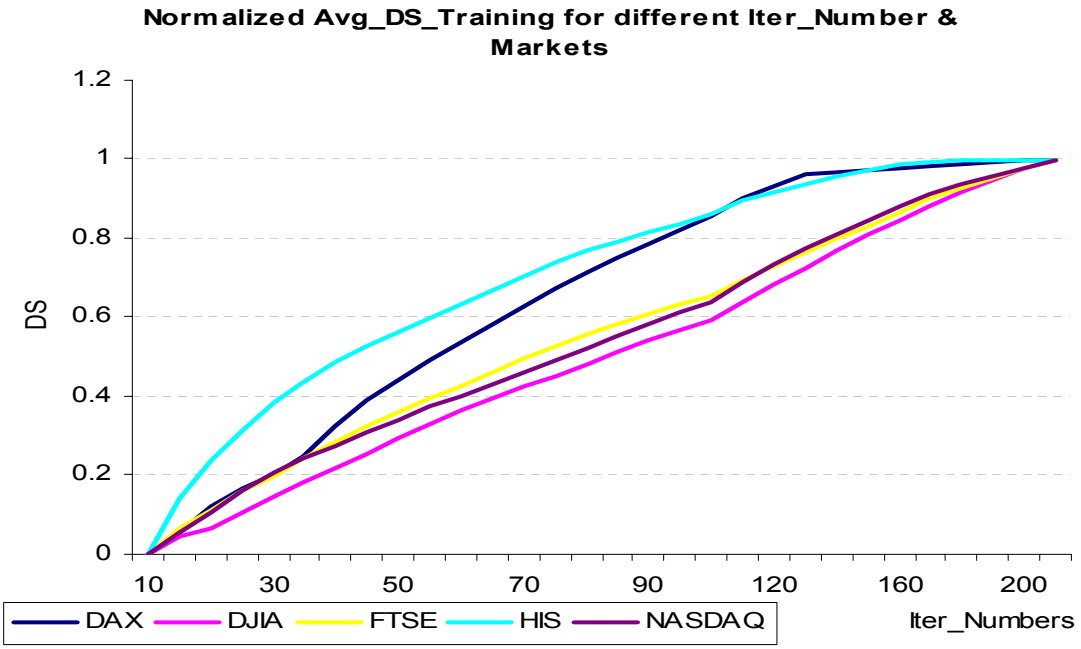


Figure 9(d), DS Results in Training Process for 5 Markets during the Increase of Iterations

4.2. Determining the optimal network architecture

To train the proposed neural network model, about nine years of daily trading data are captured for the period of 4 January 1994 to 30 September 2002, which give us a total of 2204 training patterns. To avoid the problem of over-training, we study the effects of employing different sizes of dataset in network training. We use the date of 30 September 2002 as the ending point, five different datasets are chosen for training our network models; these datasets include: 600, 800, 1000, 1500 and 2000 daily trading data. For each dataset, the latest 100 patterns are used for testing and the remaining patterns are used for training. As shown in Figure 10, five different sizes of samples are used in the neural network model based financial forecasting for the same period of 9 May 2002 to 30 September 2002.

To better understand the effects of the number of input neurons on the training and testing results, three kinds of component stock selection methods are considered in our experiments. To select the inputs for the neural network, we choose the component stocks whose correlation coefficient with their corresponding index ranks within the highest 5th, 10th and 15th respectively. However, even for the same particular market, the component stocks' correlation coefficients with the index depend on different sample sizes. For different sample sizes, the combination of those component stocks whose correlation coefficient ranks within the highest 5th, 10th or 15th may be

different. Thus we have to re-calculate as well as re-rank the correlation coefficient of the component stocks with their corresponding indices when sample sizes are different. So, a total of m component stocks (m = 5, 10, 15) are selected under these criteria for each particular market in conjunction with some specified sample size. (see Table 4.)

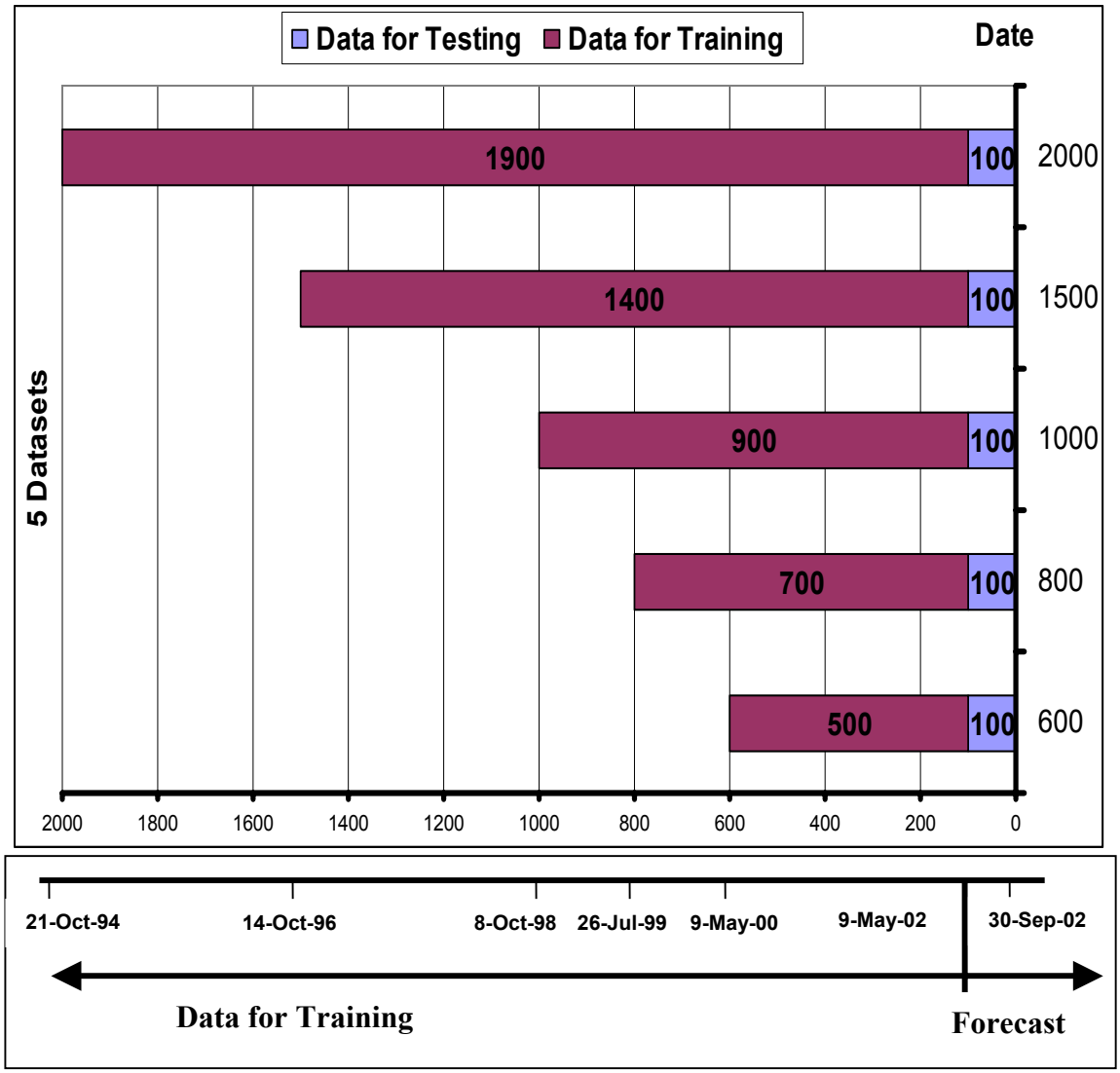


Figure 10 Five Different Datasets for Training and Testing

Table 4 Component Stocks with the Highest Correlation Coefficient with the Corresponding Indices for Each Market and for Different Dataset Size.

S=600	**DAX**		**DJIA**		**FTSE**		**HSI**		**NASDAQ**	
Rank	Code	Score	Code	Score	Code	Score	Code	Score	Code	Score
1	DBKGN	0.810	C	0.747	HSBA	0.563	941.000	0.782	CSCO	0.831
2	SIEGN	0.805	GE	0.739	III	0.559	13.000	0.770	MXIM	0.824
3	ALVG	0.782	AXP	0.718	AVZ	0.553	1.000	0.751	LLTC	0.823
4	DCXGN	0.752	JPM	0.698	VOD	0.552	5.000	0.738	XLNX	0.796
5	MUVGN	0.721	MMM	0.686	RTR	0.544	16.000	0.690	AMCC	0.789
6	DTEGN	0.711	UTX	0.672	BP	0.536	12.000	0.664	QLGC	0.789
7	HVMG	0.708	GM	0.659	RBOS	0.515	267.000	0.617	PMCS	0.784
8	VOWG	0.705	CAT	0.651	BARC	0.513	17.000	0.613	VRTS	0.784
9	BAYG	0.693	DD	0.640	LLOY	0.507	4.000	0.601	ALTR	0.780
10	BASF	0.687	HD	0.623	HBOS	0.505	20.000	0.597	SEBL	0.776
11	IFXGN	0.665	HON	0.605	PRU	0.504	11.000	0.593	AMAT	0.775
12	SAPG	0.640	MSFT	0.605	BSY	0.500	83.000	0.587	MOLX	0.772
13	EPCGN	0.637	AA	0.600	SDRT	0.488	992.000	0.574	FLEX	0.760
14	CBKG	0.636	IBM	0.587	SGE	0.487	23.000	0.540	INTC	0.756
15	BMWG	0.608	IP	0.580	OML	0.478	179.000	0.522	KLAC	0.754
S=800	**DAX**		**DJIA**		**FTSE**		**HSI**		**NASDAQ**	
Rank	Code	Score	Code	Score	Code	Score	Code	Score	Code	Score
1	SIEGN	0.783	C	0.743	HSBA	0.545	941.000	0.797	CSCO	0.825
2	ALVG	0.725	GE	0.735	VOD	0.535	13.000	0.777	MXIM	0.801
3	DBKGN	0.722	AXP	0.709	AVZ	0.526	1.000	0.759	LLTC	0.797
4	DTEGN	0.720	JPM	0.681	III	0.506	5.000	0.699	XLNX	0.793
5	DCXGN	0.685	MMM	0.650	PRU	0.495	16.000	0.665	AMCC	0.781
6	MUVGN	0.668	UTX	0.639	RTR	0.477	12.000	0.594	VRTS	0.776
7	BAYG	0.655	GM	0.623	SGE	0.466	20.000	0.583	ALTR	0.771
8	HVMG	0.647	CAT	0.610	OML	0.465	11.000	0.581	AMAT	0.771
9	VOWG	0.633	HD	0.604	BSY	0.460	17.000	0.580	PMCS	0.767
10	SAPG	0.619	DD	0.593	BARC	0.454	83.000	0.580	QLGC	0.762
11	IFXGN	0.616	HON	0.589	CW	0.449	23.000	0.543	SEBL	0.762
12	BASF	0.613	WMT	0.561	RBOS	0.447	4.000	0.540	INTC	0.753
13	CBKG	0.598	AA	0.558	STAN	0.445	267.000	0.509	KLAC	0.749
14	BMWG	0.527	INTC	0.554	LLOY	0.444	14.000	0.501	BRCM	0.748
15	TKAG	0.505	IP	0.553	BP	0.439	992.000	0.501	JDSU	0.746
S1000	**DAX**		**DJIA**		**FTSE**		**HSI**		**NASDAQ**	
Rank	Code	Score	Code	Score	Code	Score	Code	Score	Code	Score
1	SIEGN	0.624	C	0.730	HSBA	0.561	941.000	0.735	CSCO	0.829
2	DTEGN	0.613	GE	0.727	AVZ	0.529	1.000	0.697	MXIM	0.787
3	ALVG	0.596	AXP	0.712	VOD	0.528	13.000	0.694	XLNX	0.777
4	DBKGN	0.562	JPM	0.675	PRU	0.496	5.000	0.661	LLTC	0.777
5	MUVGN	0.537	UTX	0.623	BARC	0.493	16.000	0.613	INTC	0.754
6	DCXGN	0.532	MMM	0.602	III	0.492	17.000	0.546	VRTS	0.753
7	SAPG	0.511	GM	0.593	STAN	0.473	12.000	0.545	ALTR	0.748
8	VOWG	0.493	HD	0.590	LLOY	0.471	20.000	0.542	SUNW	0.747
9	BAYG	0.482	HON	0.577	RBOS	0.467	83.000	0.533	AMCC	0.745
10	HVMG	0.478	DD	0.572	RTR	0.466	11.000	0.524	AMAT	0.744
11	CBKG	0.453	WMT	0.561	BT	0.457	14.000	0.486	PMCS	0.737
12	BASF	0.421	CAT	0.560	CW	0.455	4.000	0.478	SEBL	0.735
13	BMWG	0.412	INTC	0.544	AV	0.448	23.000	0.471	JDSU	0.731
14	TUIG	0.393	MSFT	0.541	LGEN	0.437	363.000	0.461	KLAC	0.723
15	LHAG	0.374	IBM	0.539	SGE	0.434	19.000	0.460	QLGC	0.720
S1500	**DAX**		**DJIA**		**FTSE**		**HSI**		**NASDAQ**	
Rank	Code	Score	Code	Score	Code	Score	Code	Score	Code	Score
1	ALVG	0.628	GE	0.744	HSBA	0.586	1.000	0.529	CSCO	0.824
2	SIEGN	0.621	C	0.720	VOD	0.532	13.000	0.524	INTC	0.763

3	DBKGN	0.599	AXP	0.711	LLOY	0.526	5.000	0.518	MXIM	0.754
4	DTEGN	0.592	JPM	0.675	PRU	0.517	16.000	0.483	LLTC	0.751
5	MUVGN	0.579	UTX	0.631	BARC	0.514	12.000	0.454	XLNX	0.739
6	DCXGN	0.578	HD	0.597	STAN	0.509	17.000	0.447	SUNW	0.729
7	SAPG	0.543	MMM	0.589	RBOS	0.489	20.000	0.439	MSFT	0.724
8	VOWG	0.534	GM	0.588	III	0.483	11.000	0.413	AMAT	0.720
9	BAYG	0.527	HON	0.586	CW	0.469	291.000	0.411	ALTR	0.716
10	HVMG	0.512	DD	0.582	LGEN	0.469	101.000	0.392	DELL	0.698
11	CBKG	0.504	WMT	0.580	AV	0.467	83.000	0.390	KLAC	0.695
12	BASF	0.484	CAT	0.571	RTR	0.460	23.000	0.375	VRTS	0.694
13	BMWG	0.476	IBM	0.564	SDRT	0.458	267.000	0.372	PMCS	0.685
14	LHAG	0.444	MSFT	0.545	ANL	0.449	14.000	0.369	VTSS	0.678
15	TUIG	0.426	INTC	0.534	BT	0.448	4.000	0.367	JDSU	0.673
S2000	**DAX**		**DJIA**		**FTSE**		**HSI**		**NASDAQ**	
Rank	Code	Score	Code	Score	Code	Score	Code	Score	Code	Score
1	ALVG	0.620	GE	0.736	HSBA	0.590	1.000	0.516	CSCO	0.814
2	SIEGN	0.610	C	0.702	BARC	0.519	13.000	0.507	INTC	0.763
3	DBKGN	0.592	AXP	0.693	VOD	0.516	5.000	0.495	LLTC	0.726
4	DCXGN	0.575	JPM	0.655	PRU	0.514	16.000	0.467	MSFT	0.725
5	VOWG	0.531	UTX	0.622	STAN	0.505	12.000	0.451	SUNW	0.715
6	BAYG	0.526	HON	0.577	LLOY	0.498	17.000	0.439	XLNX	0.715
7	HVMG	0.503	MMM	0.575	RBOS	0.484	20.000	0.424	MXIM	0.711
8	CBKG	0.499	DD	0.575	III	0.480	11.000	0.396	AMAT	0.706
9	BASF	0.491	HD	0.572	AV	0.469	101.000	0.383	ALTR	0.701
10	BMWG	0.474	GM	0.566	LGEN	0.469	291.000	0.381	KLAC	0.676
11	LHAG	0.443	CAT	0.556	CW	0.463	267.000	0.369	ORCL	0.672
12	TUIG	0.426	WMT	0.549	RTR	0.463	83.000	0.368	DELL	0.668
13	TKAG	0.411	IBM	0.543	ANL	0.457	97.000	0.367	NVLS	0.647
14	EONG	0.397	MSFT	0.525	BP	0.454	14.000	0.366	PMCS	0.645
15	MANG	0.385	INTC	0.512	SDRT	0.449	4.000	0.365	ATML	0.644

To reflect the gains and losses of investors, the daily prices of component stocks are converted to their respective daily returns. In fact, we compute the daily returns R_{it} of each component stock C_i as follows:

$$R_{it} = \frac{C_{it} - C_{it-1}}{C_{it-1}} \times 100 \tag{9}$$

where C_{it} and C_{it-1} are close prices of the component stocks C_i for day t and day $t-1$, respectively. Similarly, the daily returns RI_t of the stock indices are calculated from index prices of I_t and I_{t-1} in the same way.

In this study, the correlation coefficient, $r_i(I)$ of the component stock C_i, is computed as follows:

$$r_i(I) = \frac{\sum_{t=1}^{N}[(R_{it} - \overline{R_i})(RI_t - \overline{RI})]}{\sqrt{\sum_{t=1}^{N}(R_{it} - \overline{R_i})^2}\sqrt{\sum_{t=1}^{N}(RI_t - \overline{RI})^2}} \quad (10)$$

where $\overline{R_i} = \frac{1}{N}\sum_{t=1}^{N} R_{it}$, and $\overline{RI} = \frac{1}{N}\sum_{t=1}^{N} RI_t$.

Consequently, taking DJIA stock index as example, the one-day ahead prediction function for daily returns can be constructed as follows.

$$R_{DJIA}(t) = \Phi(R_{DJIA}(t-1), R_1(t-1), R_2(t-1), ..., R_m(t-1)) \quad (11)$$

where, $R_i(t-1), i=1,2\ldots m$ is the daily return of the component stock C_i computed at day *t-1*. The prediction function (11) will be generated by the proposed neural network models.

Random initial weights are generated to our network simulations. As we are going to determine the optimal neural network structure and optimal data size by experiments with minimal potential influence caused by initial weights, each experiment in this section is repeated five times with different random initial weights. Thus the results reported in each experiment in this section are in fact the average results obtained from 5 independent experiments.

Besides inputs selection, we also investigate the effects of the hidden neuron number on the training and testing results, so, four kinds of neural networks with NN=5, 10, 15 and 20 neurons at the hidden layer are considered in our experiments. Thus a total of twelve kinds of neural network architectures (3×4)

are chosen in this thesis for the optimization experiments and further study. Since the size of dataset also affects the training and testing results, each network architecture is tested with the above-mentioned five kinds of datasets. Besides datasets, experiments are conducted in five stock markets by two different neural network models and repeated five times with different initial weights. Thus, as shown in Figure 11, totally 3000 independent experiments (3×4×5×2×5×5) are performed in order to study the combined effects of the datasets and the neural network architectures in both the input and hidden layers.

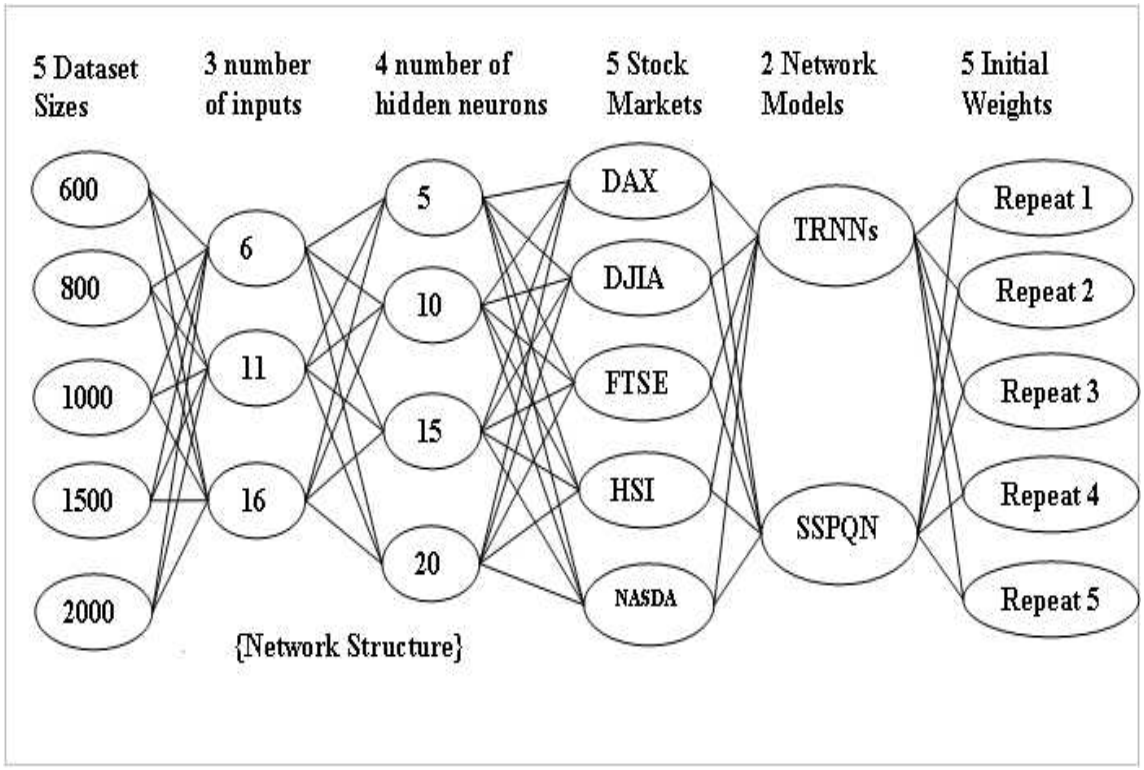


Figure 11, The Structure of the Experiments Conducted to Study the Combined Effects of the Network Structure & Dataset Sizes on Training and Testing

4.3. Variable sensitivity analysis on network modeling

4.3.1. Individual analysis

In our research, three important variables that have influence on the performance of the neural network models in both training and testing processes are considered: Dataset Size, Input Neural Number and Hidden Neuron Number. The sensitivity analysis of the neural network performance on these three major factors is a very important issue on network modeling. These major variables are noted as {D, I, H} respectively. As mentioned in Section 4.2, five different Dataset Sizes (M=5), three kinds of Input Neuron Number (N=3) and four Hidden Neuron Numbers (K=4) are considered for each of these variables in this study respectively. Table 5 illustrates the sensitivity calculation methods for each of these variables. $f(D_i, I_j, H_k), i = 1,..M,\ j = 1,..,N, k = 1,..,K$ represents the neural network performance function in MSE or DS results depends on three major variables. For a particular class of $\{D_i, I_j, H_k\}$ there is a unique $f(D_i, I_j, H_k)$ function results corresponding to it. In the case that some particular variable in the class changes, the performance function result will also change accordingly. The purpose of the study in this section is just to learn how sensitive the network performance is on the variance of these three major variables. Both the sensitivities of the neural network performance based on total variance and unit variance of these major variables are considered. Take the variable of Dataset Size as example, the sensitivity on total variance of Dataset Size

represents the largest possible influence of D, $Max_{D_i}\{\Delta f(D_i, I, H)\}, i = 1, ..., M$, on the network performance during D's variance from D_1 to D_M, while the sensitivity on unit variance of Dataset Size representing the averaged influence of D, $\frac{Max_{D_i}\{\Delta f(D_i, I, H)\}}{M-1}, i = 1, ..., M$, on network performance when D changed in unit space (changed from D_X to $D_{X\pm1}$).

The sensitivity results of the network performance on both total variance and unit variance of each variable are illustrated in table 6. The network performances in five stock markets by two models are evaluated in two criteria of MSE and DS. Take the influence of variable of Dataset Size on the testing results in Frankfurt stock exchange market as example, for there are totally 5 different sizes of dataset considered in our experiments (600, 800, 1000, 1500, 2000), "V=0.1635333" represents the largest possible change in the network performance during the variance of Dataset Size within its all possible values; while "ΔV=0.040883" (which is calculated by V/4 in this case) represents the averaged volatility effects on the neural networks performance by the unit variance of the variable (for example, the average influence on results when dataset size change from 600 to 800 or change from 800 to 1000). As obviously reflected in table 6, the sensitivity of the network performance on each of these three variables is different. Some variables may have greater influence on the performance of neural networks than other variables in some particular cases while having less influence in other cases.

It's a good idea to set M=N=K, especially when we compare the variable sensitivities based on the total variance of variables. But, there are two main considerations that make setting M=N=K in this thesis unnecessary. The first reason is for comparison with former researches. Many former researches in neural network based financial forecasting studied the different effects of training sample, input and hidden neuron number on the prediction performance and based on these analyses to determine the optimal network topology for final prediction. For a more objective comparison on the prediction results, we followed the way of former researches in determining the optimal network architecture and training samples for final prediction. For example, former researches considered five different kinds of sample sizes (600, 800, 1000, 1500 and 2000) for network training and considered four kinds of hidden neuron number (5, 10, 15 and 20) for proposed network architecture. Secondly, as we are more concerned with ΔV, the sensitivity on prediction based on the averaged unit variance of variables, the number of possible value for each variable (M or N or K) won't affect ΔV and thus won't affects the ranking results as well. Table 7, rank of variable sensitivity on network performance, is totally based on ΔV. From this point of view, the sensitivity ranking results in Table 7 based on averaged unit variance of variables do not depend on the number of N, M or K. Thus it's not necessary to set M=N=K, if we are just concerned with the sensitivity based on averaged unit variance of these variables.

Table 5 Calculation on the Sensitivity of Variables Based on Both the Total and Unit Variance of Variables

Variable Name	**Dataset Size**	**Input Neuron Number**	**Hidden Neuron Number**
Variable Note	**D**	**I**	**H**
Number of possible value for each variable	M=5	N=3	K=4
All possible values for each variable	D_i = 600,800,1000,1500, 2000 i=1,2,..,M	I_j= 6,11,16 j=1,.., N	H_k=5,10,15,20 k=1,2,..K
V, Sensitivity of neural network performance on each variable (based on total variance)	$V_D = Max_{D_i}[\frac{\sum_{j=1}^{N}\sum_{k=1}^{K} f(D_i,I_j,H_k)}{N\times K}] - Min_{D_i}[\frac{\sum_{j=1}^{N}\sum_{k=1}^{K} f(D_i,I_j,H_k)}{N\times K}], i=1,2,..M$ (12-1) $V_I = Max_{I_j}[\frac{\sum_{i=1}^{M}\sum_{k=1}^{K} f(D_i,I_j,H_k)}{M\times K}] - Min_{I_j}[\frac{\sum_{i=1}^{M}\sum_{k=1}^{K} f(D_i,I_j,H_k)}{M\times K}], j=1,2,..N$ (12-2) $V_H = Max_{H_k}[\frac{\sum_{i=1}^{M}\sum_{j=1}^{N} f(D_i,I_j,H_k)}{M\times N}] - Min_{H_k}[\frac{\sum_{i=1}^{M}\sum_{j=1}^{N} f(D_i,I_j,H_k)}{M\times N}], k=1,2,..K$ (12-3)	
Δ**V** , Sensitivity of neural network performance on each variable (based on unit variance)	$\Delta V_D = \frac{Max_{D_i}[\frac{\sum_{j=1}^{N}\sum_{k=1}^{K} f(D_i,I_j,H_k)}{N\times K}] - Min_{D_i}[\frac{\sum_{j=1}^{N}\sum_{k=1}^{K} f(D_i,I_j,H_k)}{N\times K}]}{M-1}, i=1,2,..M$ (13-1) $\Delta V_I = \frac{Max_{I_j}[\frac{\sum_{i=1}^{M}\sum_{k=1}^{K} f(D_i,I_j,H_k)}{M\times K}] - Min_{I_j}[\frac{\sum_{i=1}^{M}\sum_{k=1}^{K} f(D_i,I_j,H_k)}{M\times K}]}{N-1}, j=1,2,..N$ (13-2) $\Delta V_H = \frac{Max_{H_k}[\frac{\sum_{i=1}^{M}\sum_{j=1}^{N} f(D_i,I_j,H_k)}{M\times N}] - Min_{H_k}[\frac{\sum_{i=1}^{M}\sum_{j=1}^{N} f(D_i,I_j,H_k)}{M\times N}]}{K-1}, k=1,2,..K$ (13-3)	

Table 6, Average Individual Effects of 3 Variables on the NN Performance

Based on Testing Results		**Trust Region Neural Networks**				**SSPQN Neural Networks**			
		MSE		**DS**		**MSE**		**DS**	
Markets	Variables	**V***	**ΔV***	**V**	**ΔV**	**V**	**ΔV**	**V**	**ΔV**
DAX	Dataset Sizes	0.164	0.041	4.865	1.216	0.127	0.032	5.498	1.375
	Inputs Number	0.411	0.206	0.202	0.101	0.468	0.234	1.005	0.503
	Hidden Neurons	2.123	0.708	1.564	0.521	2.263	0.754	1.019	0.340
DJIA	Dataset Sizes	0.036	0.009	1.565	0.391	0.024	0.006	1.437	0.359
	Inputs Number	0.420	0.210	0.478	0.239	0.441	0.220	0.681	0.341
	Hidden Neurons	1.980	0.660	1.456	0.485	2.098	0.699	0.855	0.285
FTSE	Dataset Sizes	0.097	0.024	3.963	0.991	0.052	0.013	5.793	1.448
	Inputs Number	0.395	0.198	0.516	0.258	0.453	0.226	0.409	0.205
	Hidden Neurons	2.124	0.708	0.972	0.324	2.267	0.756	0.812	0.271
HSI	Dataset Sizes	0.056	0.014	5.013	1.253	0.031	0.008	7.135	1.784
	Inputs Number	0.441	0.220	1.725	0.863	0.468	0.234	1.891	0.946
	Hidden Neurons	1.883	0.628	0.927	0.309	1.992	0.664	1.404	0.468
NASDAQ	Dataset Sizes	0.106	0.026	1.880	0.470	0.165	0.041	2.737	0.684
	Inputs Number	0.350	0.175	0.654	0.327	0.408	0.204	0.907	0.454
	Hidden Neurons	2.086	0.695	0.589	0.196	2.194	0.731	0.428	0.143
Based on Training Results		**Trust Region Neural Networks**				**SSPQN Neural Networks**			
		MSE		**DS**		**MSE**		**DS**	
Markets	Variables	**V***	**ΔV***	**V**	**ΔV**	**V**	**ΔV**	**V**	**ΔV**
DAX	Dataset Sizes	0.547	0.137	2.082	0.520	0.492	0.123	2.614	0.653
	Inputs Number	0.424	0.212	0.148	0.074	0.475	0.238	0.290	0.145
	Hidden Neurons	2.123	0.708	0.524	0.175	2.253	0.751	0.430	0.143
DJIA	Dataset Sizes	0.390	0.097	1.231	0.308	0.335	0.084	1.964	0.491
	Inputs Number	0.448	0.224	0.215	0.108	0.477	0.239	0.342	0.171
	Hidden Neurons	2.120	0.707	0.417	0.139	2.268	0.756	0.133	0.044
FTSE	Dataset Sizes	0.330	0.082	0.554	0.138	0.286	0.071	0.437	0.109
	Inputs Number	0.467	0.234	0.306	0.153	0.509	0.254	0.297	0.148
	Hidden Neurons	2.150	0.717	0.598	0.199	2.277	0.759	0.354	0.118
HSI	Dataset Sizes	1.859	0.465	2.199	0.550	1.851	0.463	2.592	0.648
	Inputs Number	0.455	0.228	0.414	0.207	0.475	0.238	0.127	0.064
	Hidden Neurons	2.101	0.700	0.756	0.252	2.226	0.742	0.289	0.096
NASDAQ	Dataset Sizes	4.709	1.177	1.254	0.314	4.658	1.164	1.995	0.499
	Inputs Number	0.481	0.240	0.178	0.089	0.511	0.255	0.219	0.110
	Hidden Neurons	2.300	0.767	0.594	0.198	2.388	0.796	0.228	0.076

* V represents the sensitivity of neural network performance on total variance of the variables
* ΔV represents the sensitivity the neural network performance on unit variance of the variables

Table 7 Rank of Variable Sensitivity on Network Performance

For Training	The Extent of Variables Influencing the Performance of Neural Networks in Financial Forecasting		
MSE Criteria	**Most sensitive variable**	**More sensitive variable**	**Not sensitive variable**
DAX	Hidden Neuron Number	Input Neuron Number	Dataset Size
DJIA	Hidden Neuron Number	Input Neuron Number	Dataset Size
FTSE	Hidden Neuron Number	Input Neuron Number	Dataset Size
HSI	Hidden Neuron Number	Dataset Size	Input Neuron Number
NASDAQ	Dataset Size	Hidden Neuron Number	Input Neuron Number
DS Criteria	**Most sensitive variable**	**More sensitive variable**	**Not sensitive variable**
DAX	Dataset Size	\	\
DJIA	Dataset Size	\	\
FTSE	\	\	Dataset Size
HSI	Dataset Size	Hidden Neuron Number	Input Neuron Number
NASDAQ	Dataset Size	\	\
For Testing	**The Extent of Variables Influencing the Performance of Neural Networks in Financial Forecasting**		
MSE Criteria	**Most sensitive variable**	**More sensitive variable**	**Not sensitive variable**
DAX	Hidden Neuron Number	Input Neuron Number	Dataset Size
DJIA	Hidden Neuron Number	Input Neuron Number	Dataset Size
FTSE	Hidden Neuron Number	Input Neuron Number	Dataset Size
HSI	Hidden Neuron Number	Input Neuron Number	Dataset Size
NASDAQ	Hidden Neuron Number	Input Neuron Number	Dataset Size
DS Criteria	**Most sensitive variable**	**More sensitive variable**	**Not sensitive variable**
DAX	Dataset Size	\	\
DJIA	\	\	\
FTSE	Dataset Size	Hidden Neuron Number	Input Neuron Number
HSI	Dataset Size	Input Neuron Number	Hidden Neuron Number
NASDAQ	Dataset Size	Input Neuron Number	Hidden Neuron Number

As shown in table 7, the sensitivities of the network performance on three variables in each particular market are ranked in three levels: the most sensitivity, more sensitivity and least sensitivity. We should note that, only when the two ranking results obtained from both network models are

consistent; the consistent ranking results could be concluded. If the ranking results for the same particular market obtained from different models are different, then no conclusion could be drawn and in this case the results are reflected by the mark of "/". The variable that ranks as the "most sensitive" should be given more attention by researchers in network forecasting for it has greater influence on the network performance than other two variables. It's very interesting that, based on testing MSE results, the ranking orders for all the five markets are absolutely consistent: that the variance of Hidden Neural Number gives the most influence on the network performance, while the variance of Dataset Size gives the least influence on it, leaving the variance of Input Neural Number at the middle point. This results shows that in neural network based financial forecasting, the prediction accuracy depends more on the specification of the network architecture than on the sample data selection. Based on training MSE results, the ranking orders for DAX, DJIA and FTSE are also consistent with what is concluded from the testing MSE results. But for the HSI and NASDAQ indices, the ranking orders are different: in both implementations the Dataset Size ranks with higher sensitivity than it does in other cases, leaving the ranking orders between the two architecture variables remain unchanged. Comparing with other three stock market indices, HSI and NASDAQ indices data are obviously noisier, which may be the main reason that causes the difference in ranking orders in the training process. Tough, many ranking results based on DS obtained from two different network models are not consistent; we still can draw some conclusions on the importance of some variables on the influence of DS performance of neural networks. One most impressive finding under the DS criteria is that sample data selection

impacts the network DS performance more than network architecture variables do in all consistent cases except for London stock exchange market both in training and testing processes. As we have introduced in chapter one, the FTSE index data is the least noisy one among all the five indices. In the FTSE case, the network DS performance is least sensitive on the variance of Dataset Size.

Based on the individual sensitivity analysis on the three major factors in neural network modeling for financial forecasting, some important conclusions could be drawn from the findings:

(1). The network prediction accuracy evaluated under MSE criterion depends more on the specification of network architecture than on the sample data size, while such relationship will reverse when prediction accuracy is evaluated under DS criterion.

(2). In the aspect of network architecture, the Hidden Neural Number usually has more impact on the network performance than Input Neuron Selection.

(3). The sensitivity of the network performance on sample data size is positively correlated with the extent of noise or volatility of the sample dataset being studied. The noisier or more volatility the sample dataset is, the more attention should be paid on determining the optimal sample data size in order for a good network performance. In the case that the sample data is not noisy, sample dataset selection is comparatively less important than specification of network architecture in the aspect of the impacts on network performance.

As shown in Table 7 that under MSE evaluation criterion, the entire ranking results in five different stock markets are consistent by different network models both in training and testing processes, while under DS evaluation criterion, many ranking results are inconsistent when different network models are used. For example, under DS criterion, the network performance is more sensitive on the Input Neurons than on the Hidden Neurons when TRNN model is used, while the network performance is more sensitive on the Hidden Neurons than on the Input Neurons if SSPQN model is used. As we know that evaluation criteria of MSE and DS are just different ways to measure the neuron network performance on prediction accuracy. Why under different performance evaluation criteria, the consistency of the sensitivity ranking results under different models is different? A main possible reason may lead to such difference, that is the difference between the definitions of these two criteria themselves. Mean Squared Error (MSE) is the average of the square of the difference between the desired response and the actual system output (the error), while, in this thesis, Directional Symmetry (DS) reflects the percentage of correctly predicted directions with respect to the stock index return. In our research, only MSE is used as the error function for both of the network models. Thus, the learning process is implemented by changing the weights in order to reduce the MSE value. The learning process is repeated until the MSE between the computed and target output values are at an acceptably low value. Though, a low (or high) MSE value often in conjunction with a corresponding high (or low) DS value, their relationship is not always consistent. Low (or high) MSE value cannot guarantee a high (or low) DS

value, because mean squared error cannot reflect the prediction accuracy in the aspect of direction. Thus a sensitivity ranking based on MSE criterion may not always consistent with the ranking results based on the DS criterion. Also, in the case when MSE based sensitivity rankings are consistent for different models, it still possible for the inconsistency in the ranking results for different models when DS evaluation criterion is used. The key point here is that the error function for the network training is MSE, thus we cannot guarantee the DS value to be optimized steadily during the process of minimizing MSE value in the training process, especially when different training algorithms are used.

The sensitivity analysis, especially the ranking results can be regarded as a valuable reference on neural network modeling for financial forecasting both in training and testing processes. Particularly, the analytical results for the five major stock markets are more valuable for the future researches conducted in these particular markets based on neural networks.

4.3.2. Interaction analysis

The above research analysis is mainly focused on ranking variable influences on the network performance individually without paying much attention on the interrelationship between these three factors as well as how these interrelationships affect the network performance. Understanding more about the interrelationships of these major factors between each other and how they

affect the network performance may generalize more valuable recommendations on network modeling, especially in financial forecasting.

In this research work, we classify the influence of one variable on the impact of other variable on the network performance into three levels: “High”, “Medium” and “Low” influences. If the impact of variable A on the network performance (either MSE or DS result) is not sensitive on the variance of variable B, then variable B is said to have “Low” influence on the impact of variable A on the network performance. On the contrary, if the impact of variable A on network performance is obviously sensitive on the variance of variable B, then such influence is said to have “High” effect on the impact of A on network performance. In the case that the extent of influence of variable B on A ranks between “High” and “Low”, such influence is said to be “Medium”. In this thesis, we use the chart analysis method to identify the three levels of interrelationships between the major variables as mentioned above. In the chart analysis (see figure 12), if the shape of the chart that reflects the relationship between variable A and MSE (or DS) results keeps consistent in all cases of variable B, then it was said that there is a “Low” influence of variable B on A in the aspect of the impact of A on network performance. While, when the chart shape changes for each case of variable B, the influence of B on the impact of A on network performance is classified into the category of “High”. The influence in situations that ranks between “High” and “Low” are classified into “Medium”, that is when the chart shape only changes in some cases of variable B, instead of changing in all cases. Under this classification method, the interrelationships between these variables are

ranked into three levels of “High”, “Medium” and “Low”. The summarized results reflected under this way are shown at the end of this subsection.

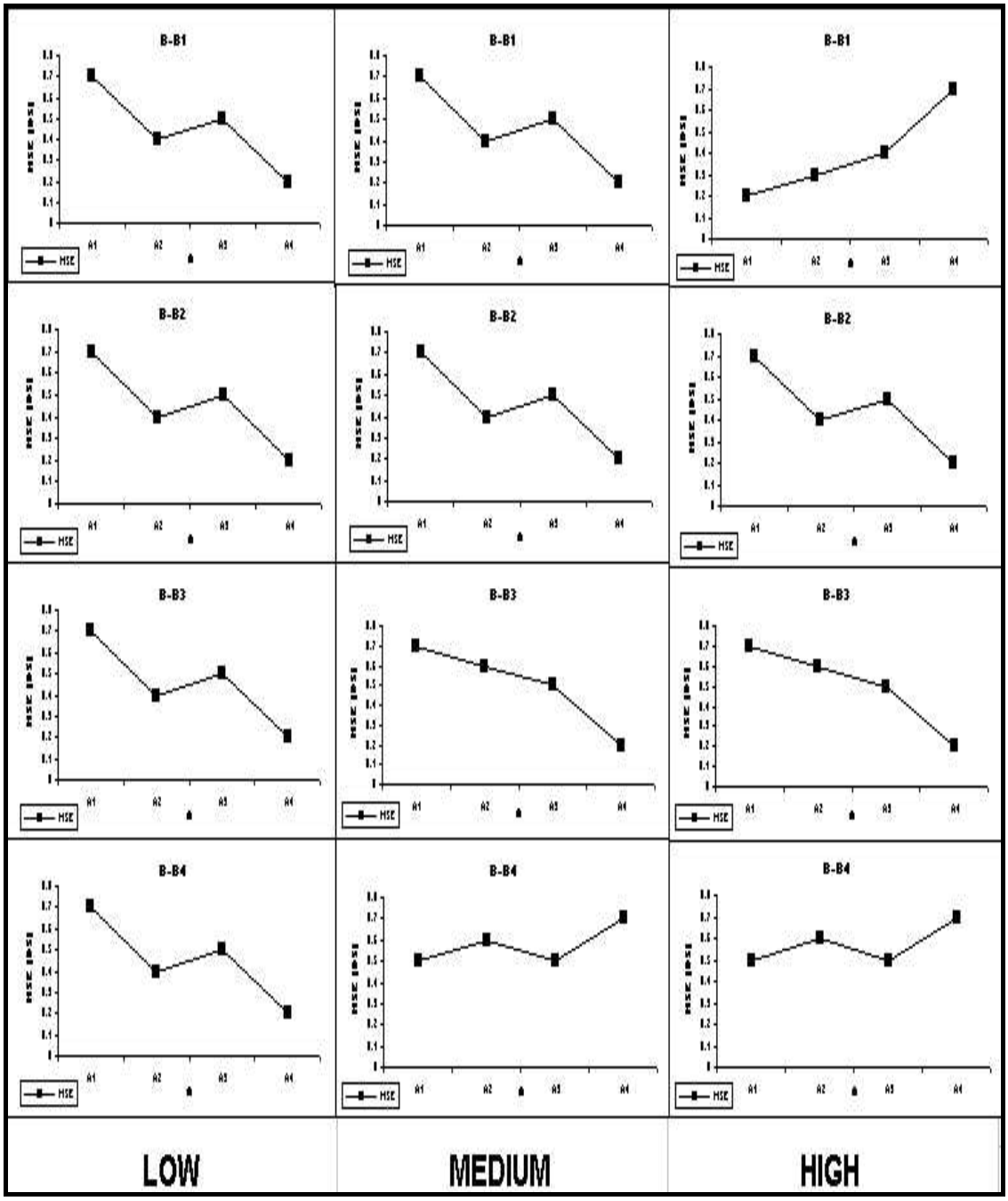
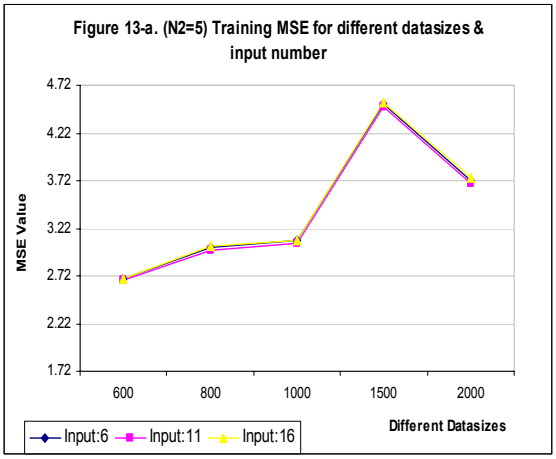


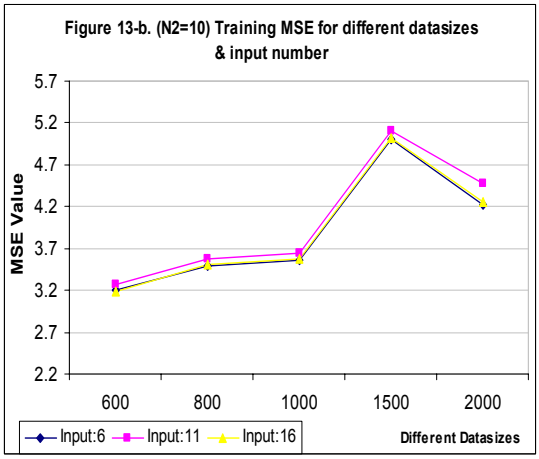
Figure 12. The Classification of Interrelationship between Two Variables Based on Chart Analysis

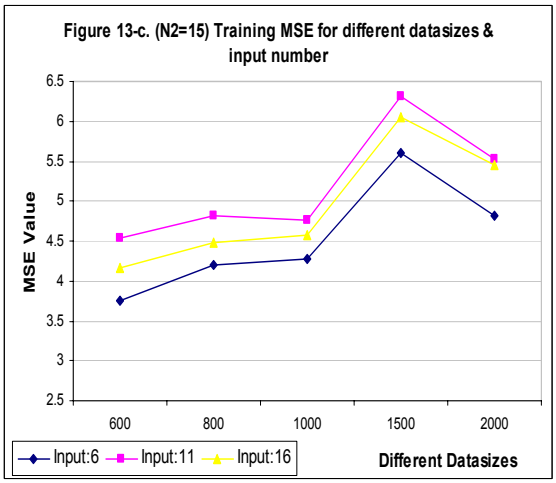
In this section, we will first take Hong Kong Heng Seng Stock Index (forecasting by TRNN model) as example to show how we analyse the interrelationships between these three variables in detail and then summarize the analyzing results obtained from all the five stock markets under two neural network models. Figure 13 to 20 analyse the interrelationships between these three variables in the training process based on both the MSE and DS results, while figure 21 to 28 mainly focused on the similar analysis in the testing process. For there are totally three major variables that may have influence on the network performance, in order to know more clearly about the interrelationship of the variables between each other, it's wise for us to fix one particular variable and see what's the relationship between the remaining two.

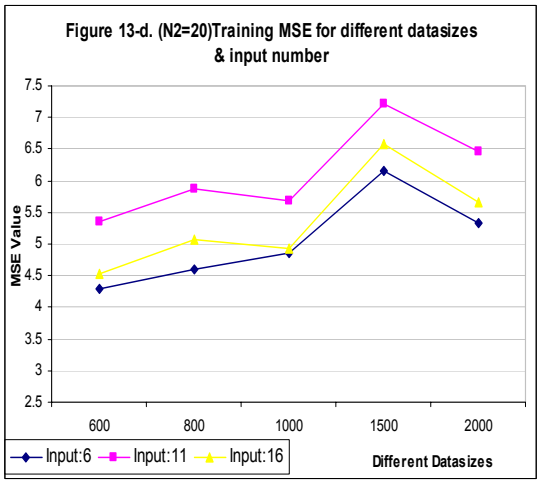
Under this consideration, figure 13 (a) to (e) illustrate the relationship between dataset size and input number when the number of hidden neuron is fixed at 5, 10, 15 and 20 respectively and the averaged results are also plotted for reference. As we can see that in all cases, the charts are in the quite similar shape for all different hidden neuron numbers. That is, the MSE result for training keeps rising gradually as the dataset size changes from 600 to 1500 and drops suddenly from 1500 to 2000, leaving the chart reaches its global top at 1500. This observation could demonstrate that the changes of input number and hidden neuron number have little or "Low" influence on the impact of dataset size on the training MSE results. On the basis of the above analysis, the best dataset size is obtained when dataset size is 600. Besides the obvious influence of dataset size on the training MSE results, we also could notice that as the hidden neuron number changes from 5 to 20 the impact of

input number on the training MSE results becomes more obvious. From figure (b) to (e) we could easy to find that the best MSE result was obtained when input number is 6 and the worst one was get when input number is 11. These phenomena could show that the hidden neuron number could influence the impact of input number on the training MSE results. Figure 14 (a) to (e) also fix the hidden neuron number to 5, 10, 15, 20 and average value respectively, but focused on illustrating the impact of input number on the performance of MSE results on training. When hidden neuron number fixed at 5, the variance of input number seems have no impact on training MSE. But, under remaining conditions when hidden neuron number is larger than 5 and on average, the impact of input number becomes obviously that for different dataset sizes the relationships between input number and training MSE are all in the shape of reversed “V”, that means the model gets the highest MSE result when input number is 11 and gets lower ones when input number being 6 or 16. On the other hand, the best training MSE is always obtained when input number is 6. These results demonstrate again that the hidden neuron number influences the impact of input number on training MSE results in “Medium” level.









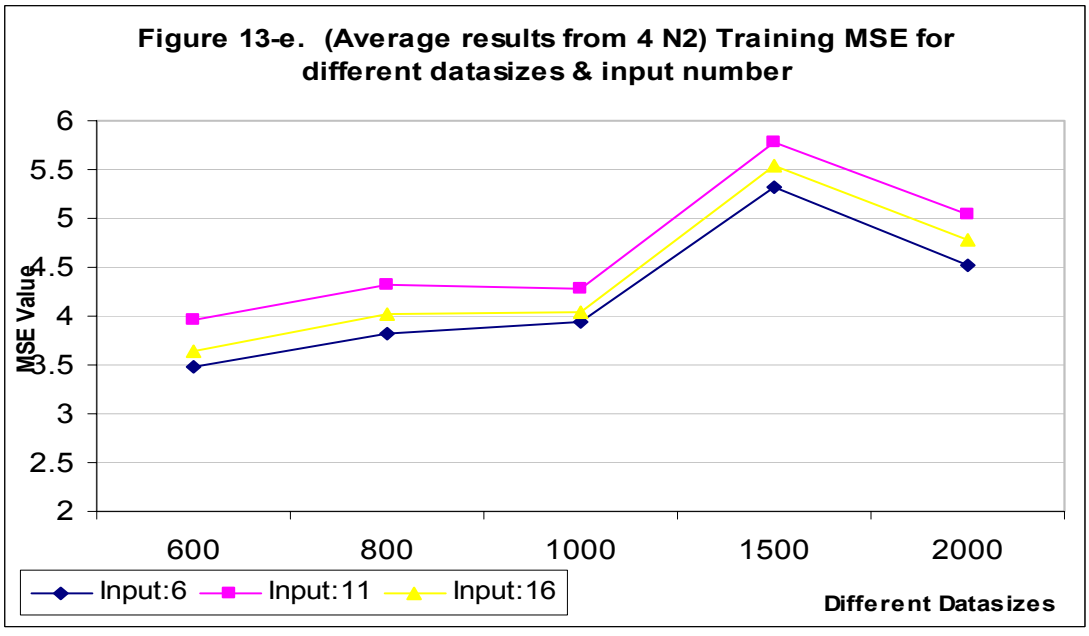
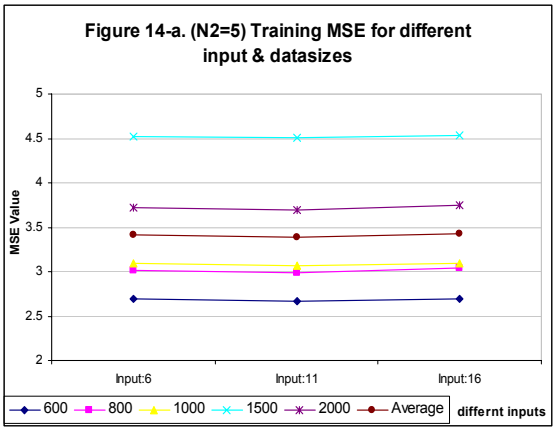
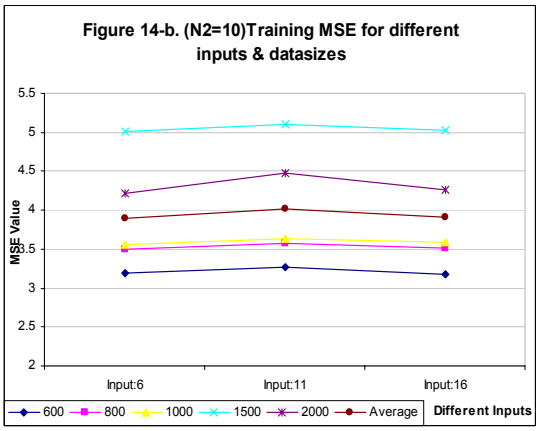
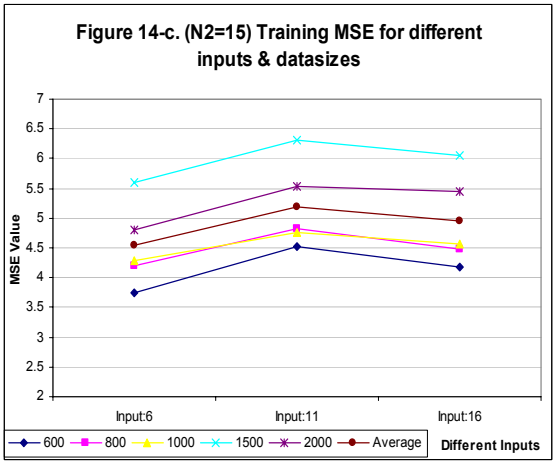
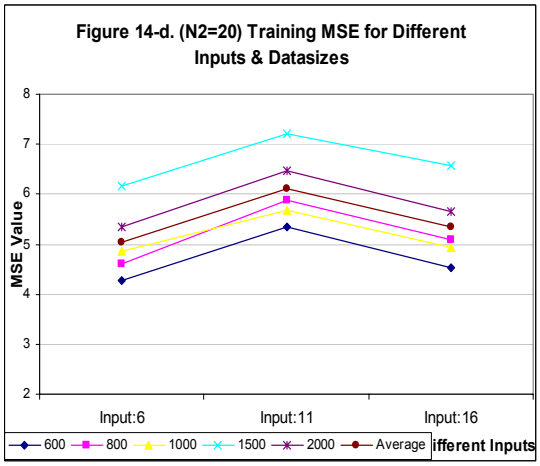


Figure 13 (a) to (e), Effects of Dataset Sizes as Measured by Average MSE for Training (TR) on HSI Experiments (Hidden Neuron Number fixed to 5, 10, 15, 20 and Average respectively)









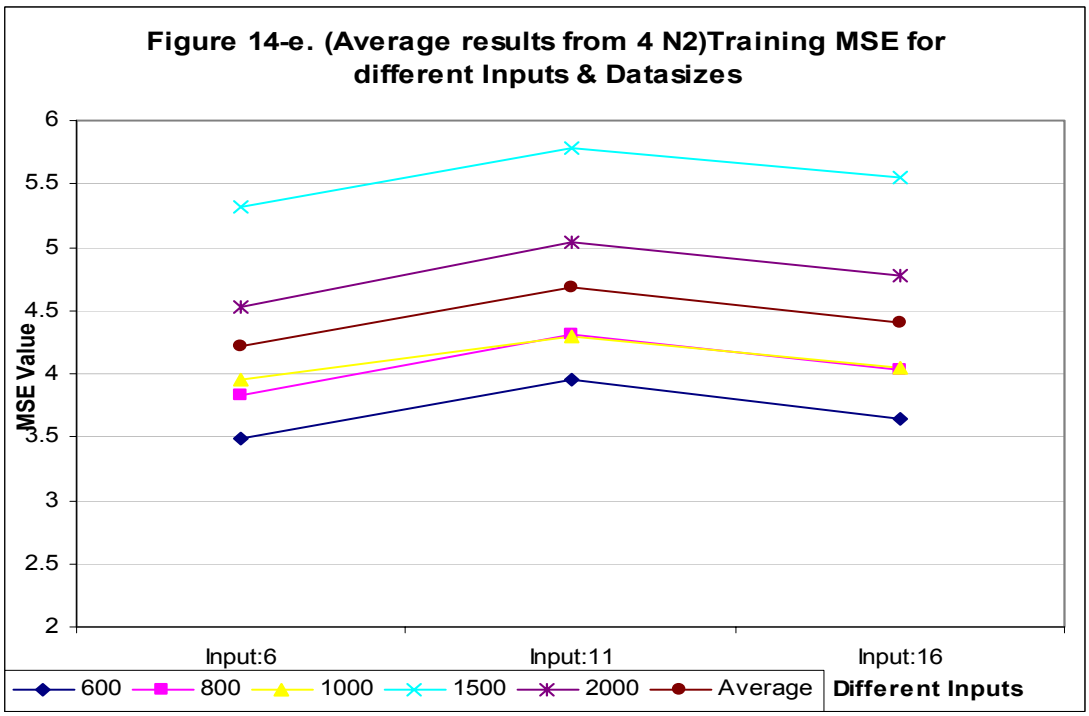
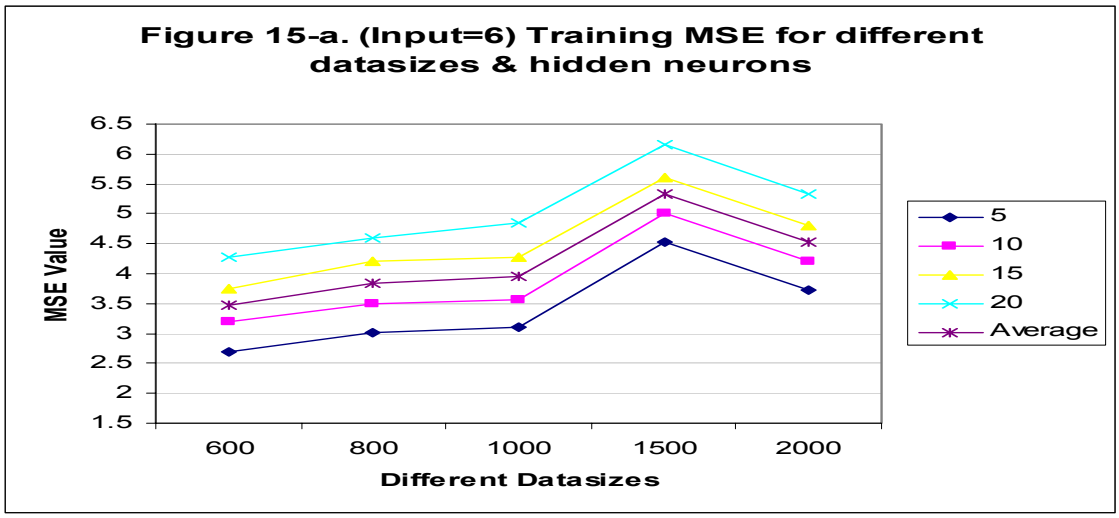
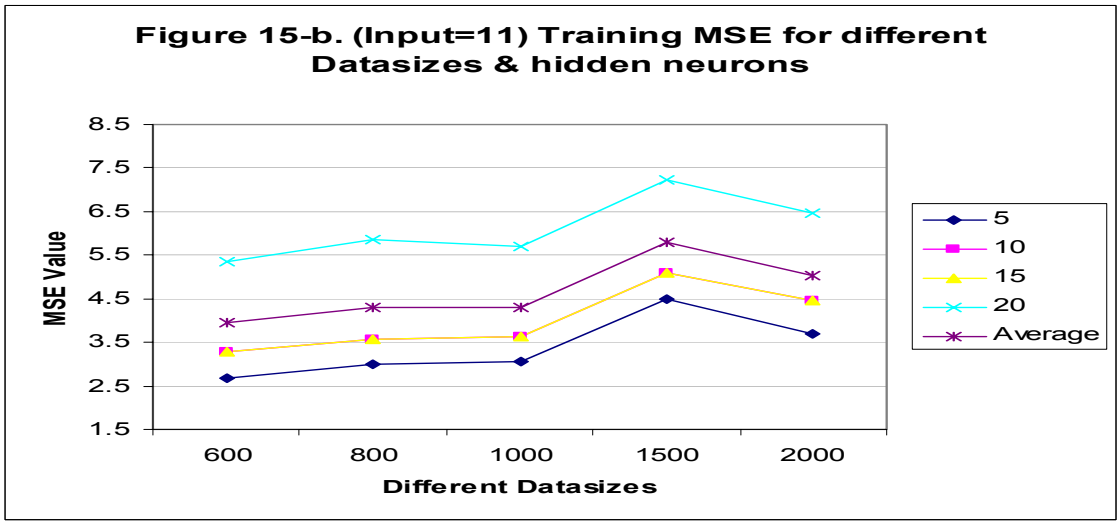


Figure 14 (a) to (e), Effects of Inputs Number as Measured by Average MSE for Training (TR) on HSI Experiments (Hidden Neuron Number fixed to 5, 10, 15, 20 and Average respectively)

After fixing the number of input neurons to 6, 11 and 16 respectively, the effects of dataset size on training MSE as well as the interrelationship between dataset size and hidden neuron number are illustrate in figure 15 (a) to (c). It is obviously that the impact of dataset size on training MSE is consistent in all cases. It seems that the neuron network structure has “Low” influence on the impact of dataset size on the training MSE results in neuron network based forecasting.

As for the number of hidden neurons, there are many studies in the literature to guide the architecture selection. Generally, too many nodes in the hidden layer produce a network that memorize the input data and lack the ability to generalize. However, besides that there is no general guidance that is suitable for all situations. Some methods are time-consuming and impractical, such as cascade-correlation method proposed by Fahlman [16] and pruning approach [4, 26]. While the others seems do not work well for all problems, which include some rule of thumb in the literature. In summary, the specification of the internal architecture involves tradeoffs between fitting accuracy and generalization ability and the best way to find the optimal number of hidden neurons for a particular application is through experiments [25]. The effects of hidden neurons on training MSE are illustrated in figure 16. It’s interesting to find that the training MSE result keeps rising gradually as the hidden neuron number increases from 5 to 20 and this trend keeps consistent for all different dataset sizes and input numbers. It demonstrates that the other two variables have “Low” influence on the impact of hidden neuron number on training MSE.





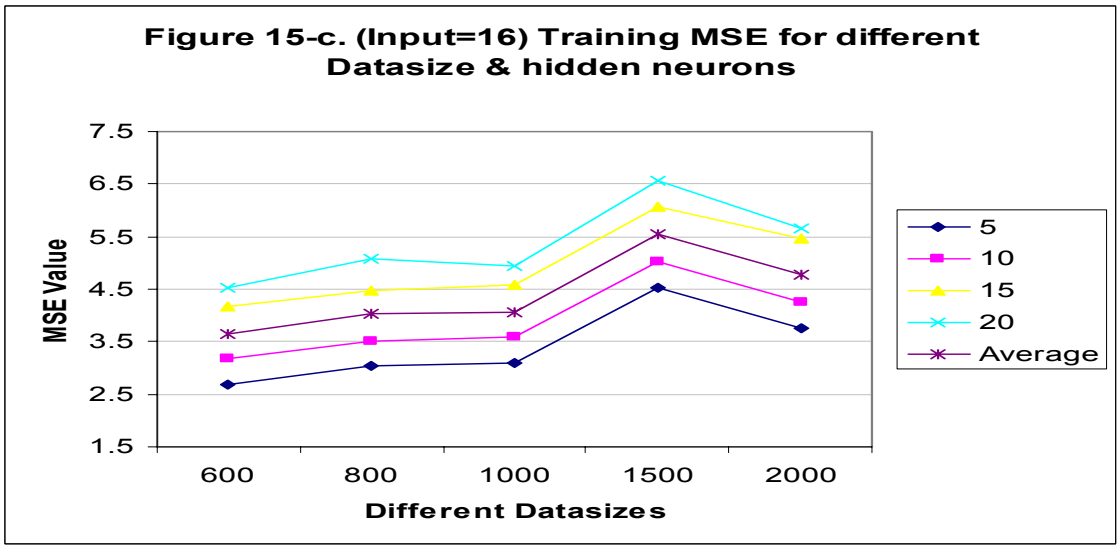
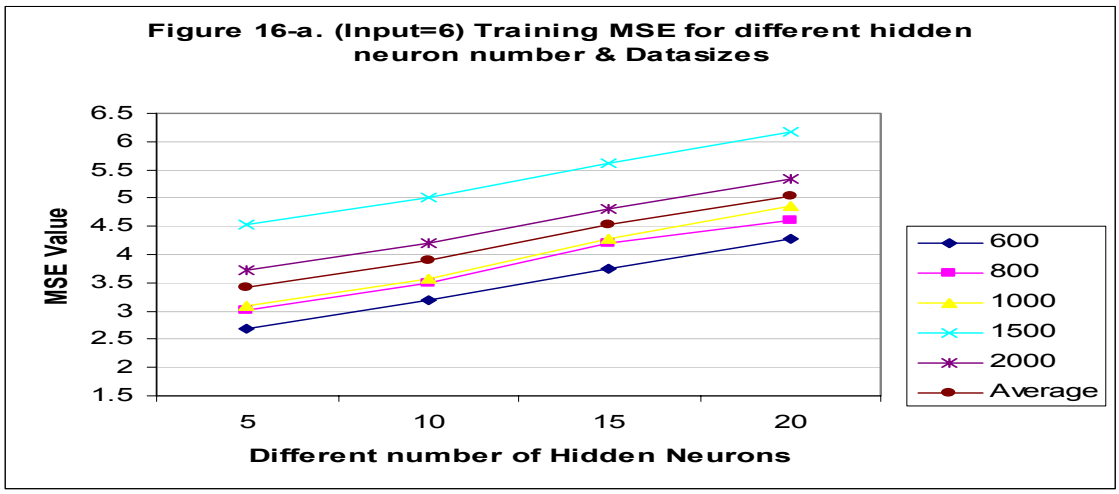
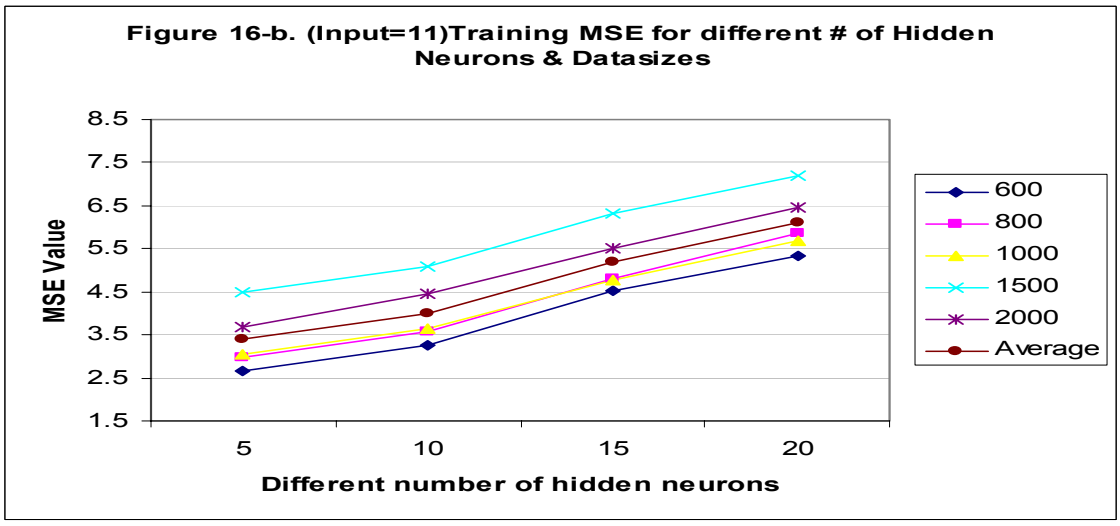


Figure 15 (a) to (c), Effects of Dataset Size as Measured by Average MSE for Training (TR) on HSI Experiments (Input Number fixed to 6, 11, 16 respectively)





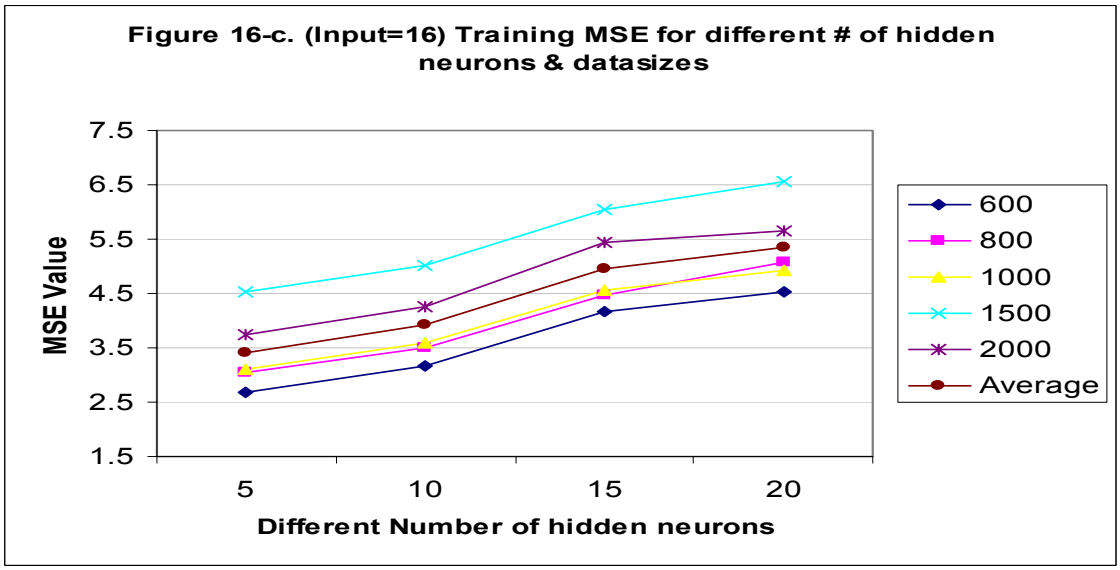


Figure 16 (a) to (c), Effects of Hidden Neuron Number as Measured by Average MSE for Training (TR) on HSI Experiments (Input Number fixed to 6, 11, 16 respectively)

In our research work, both of the criteria are considered and analysed in details. Figures 17 to 20 analyse the interrelationships between the three variables in the training process based on DS results. Figure 17 (a) to (e) illustrate the effect of dataset size on the training DS as well as the interrelationship between dataset size and input number by fixing the hidden neuron number to 5, 10, 15, 20 and average value respectively. On average, the training DS result first increases from 600 to 800 and, after reaching its high point at 800, it keeps decreasing after 800 until 2000. As shown in figure 17 (d), when hidden neuron number is 20, we could see that the relationships between dataset size and the DS results are obviously different under different input numbers. But, the impact of dataset size on training DS is not obviously sensitive to the variance of the other two variables in all other situations, thus we can say that hidden neuron number and input number have no obvious influence on the impact of dataset size on training DS. Their influence on the impact of dataset size on network performance could be regard as “Medium”. The effects of input neuron number on the training DS results are illustrated in the figure 18. It’s obviously that for each particular fixed hidden neuron number, the impact of input neuron number on training DS is different for different dataset sizes. Taken N2=5 as example, the DS results for all three input number have no obvious differences when dataset size is 800, while obvious difference could be observed when dataset size changes to 1000. These results show that the dataset size has “High” influence on the impact of input number on the training DS. On the other hand, it’s also easy to notice that for different number of hidden neurons the relationship between input number and DS result changes obviously. Thus, hidden neuron number can

also influence the impact of input number on training DS in “High” level. Figure 19 illustrates the effects of dataset size on training DS by fixing the input neuron numbers respectively. By the variance of input number, the interrelationships between the dataset size, hidden neuron number and the training DS seem consistent without obvious changes, which shows that input neuron number has “Low” influence on the impact of dataset size on training DS results. While for each particular fixed input number, when hidden neuron number changes, the corresponding relationships between DS and dataset size may also change, which shows that the hidden neuron number has influence on the impact of dataset size on network performance, but this influence is “Medium”. By the similar analysing methods, we discovered that there is no consistent relationship between the hidden neuron number and the training DS results. And on the other hand, both the input number and the dataset size have obviously “High” influence on the impact of hidden neuron number on the training DS, which is illustrated in the Figure 20. Based on the analysing results obtained under two criteria, we can see that the interrelationships between the three variables are more obviously reflected when DS criterion is used. While in the case of using DS criterion, the interrelationships between the three major factors become obviously and the variance of each variable may cause obvious influence on the impact of other variables on the network performance. Thus, we should pay more attention on the impact of interrelationship between the major variables on network performance when DS is used as the evaluation criterion.

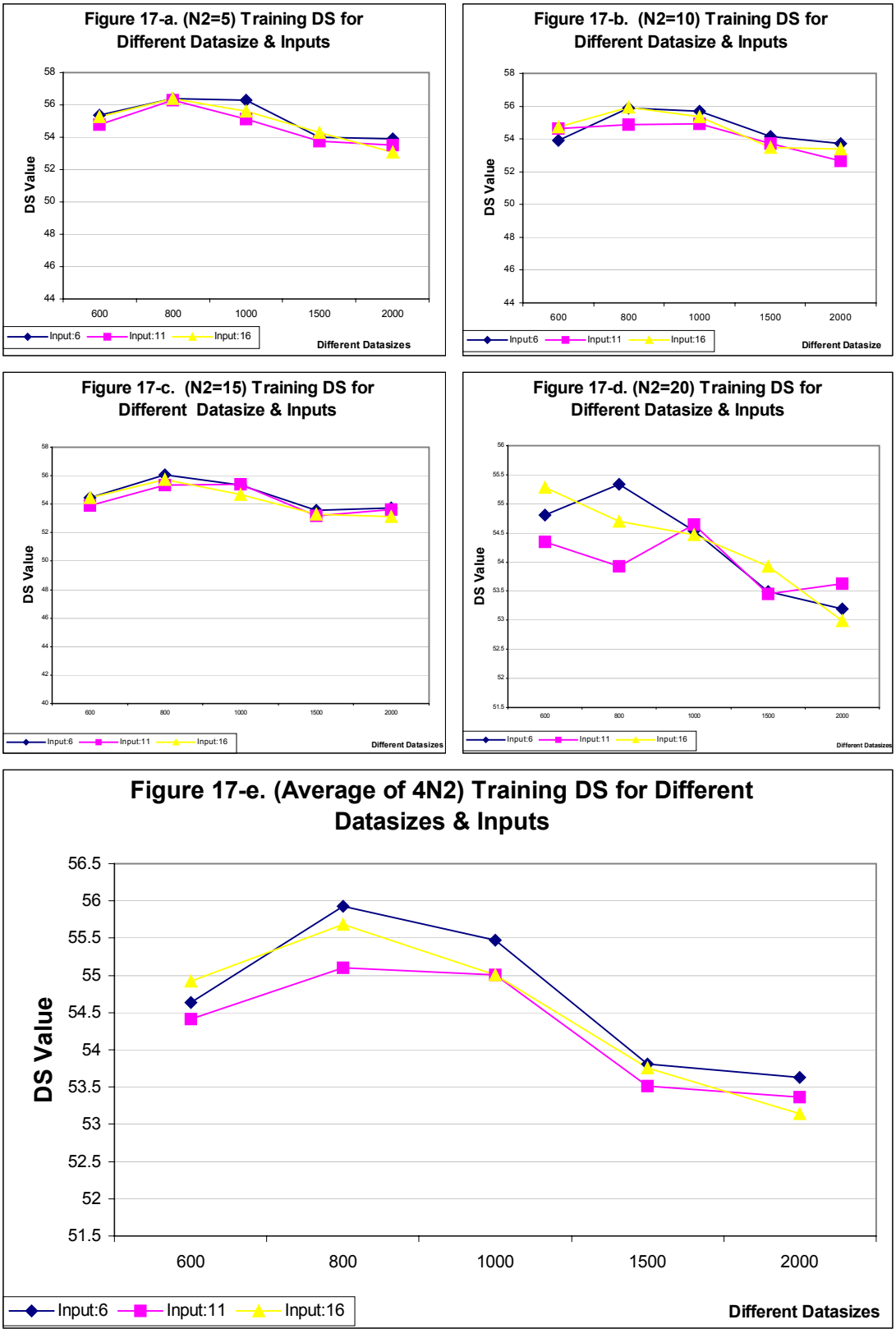
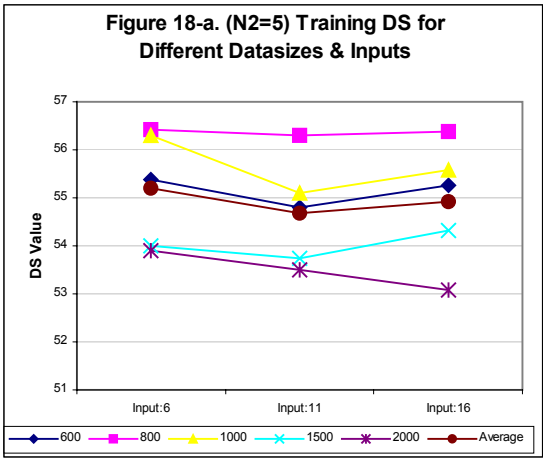
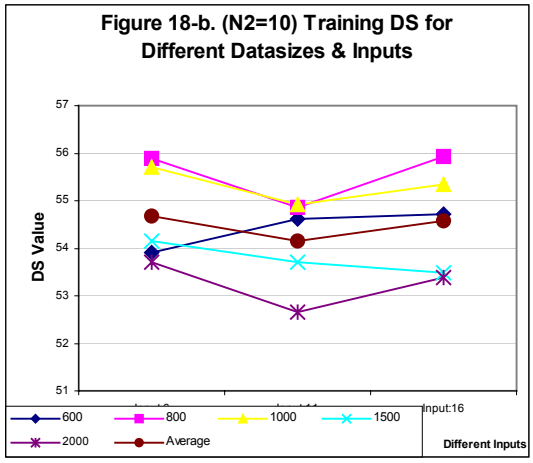
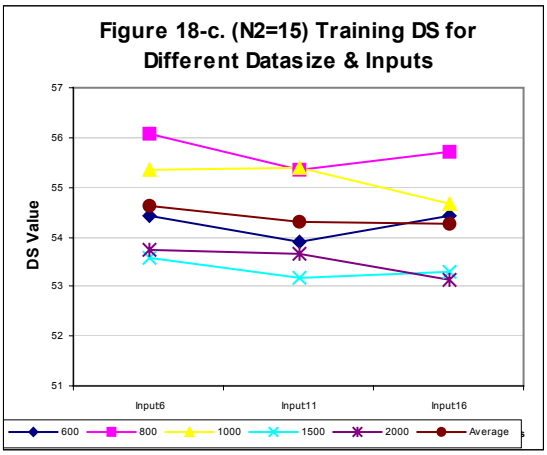
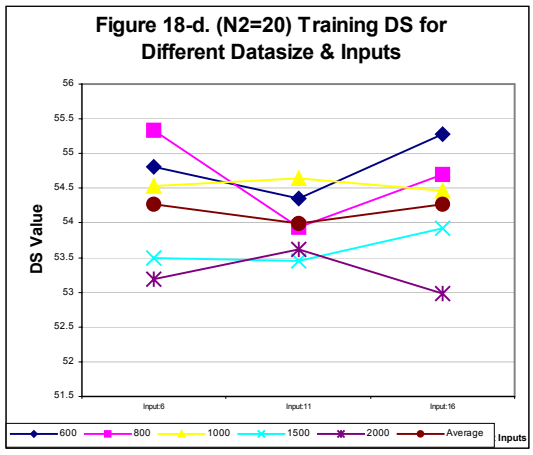


Figure 17 (a) to (e), Effects of Dataset Sizes as Measured by Average DS for Training (TR) on HSI Experiments (Hidden Neuron Number fixed to 5, 10, 15, 20 and Average respectively)









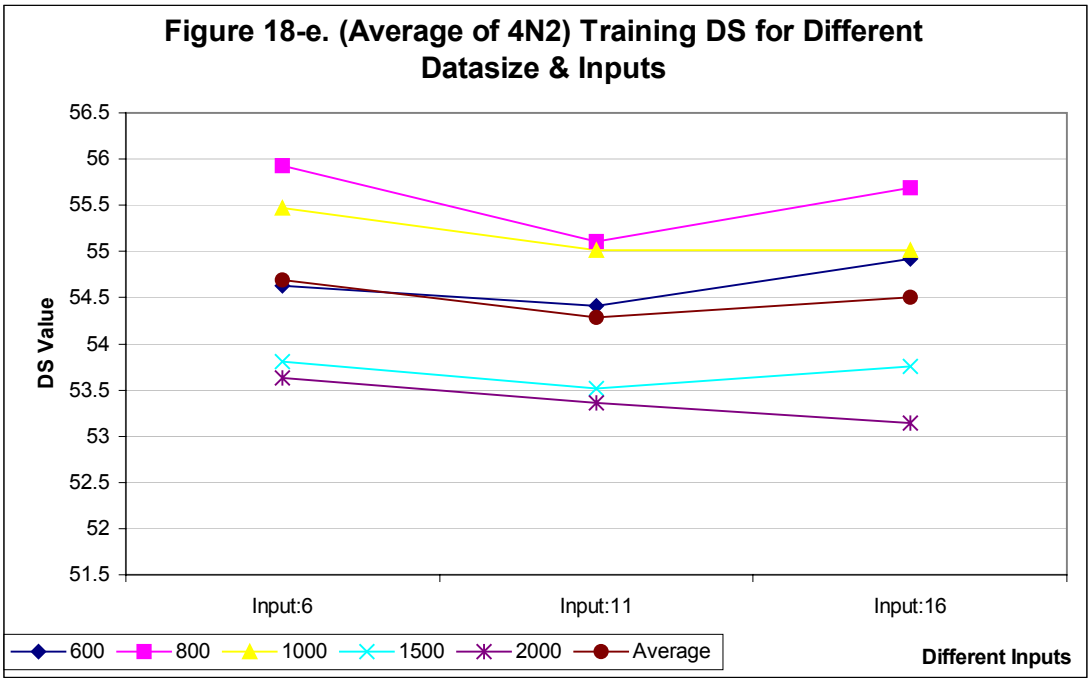
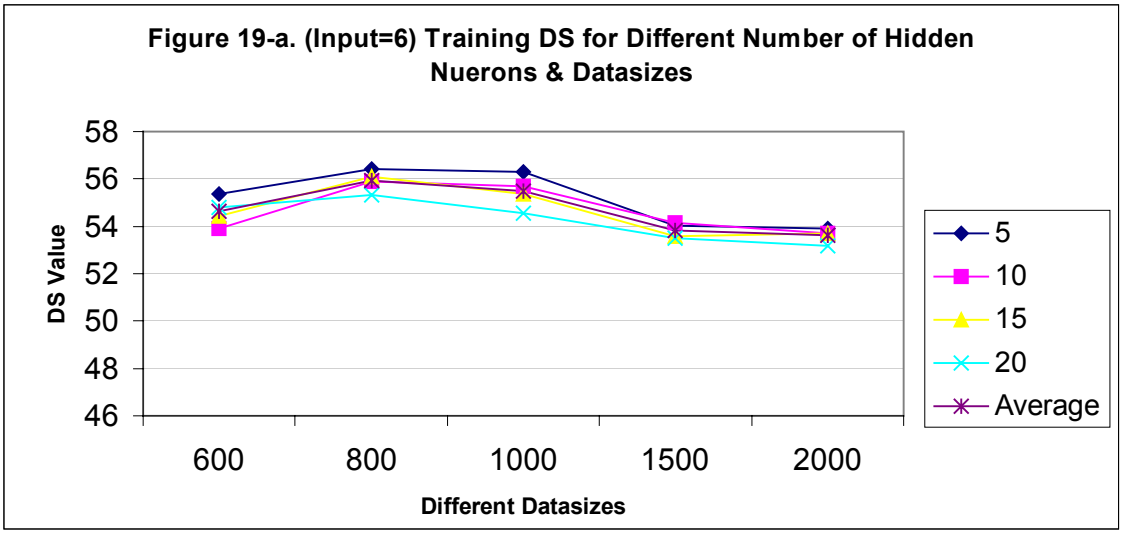
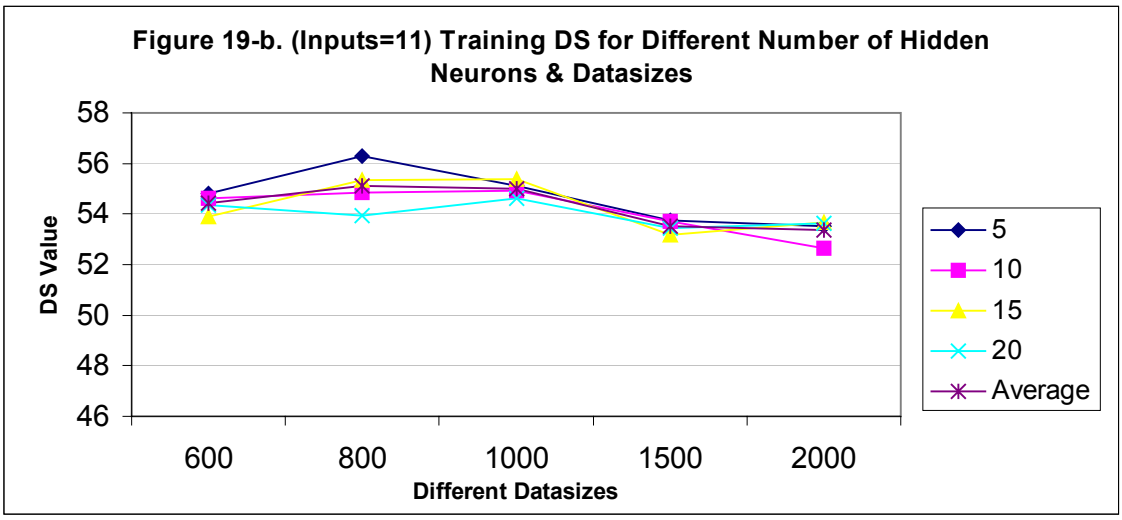


Figure 18 (a) to (e), Effects of Input Number as Measured by Average DS for Training (TR) on HSI Experiments (Hidden Neuron Number fixed to 5, 10, 15, 20 and Average respectively)





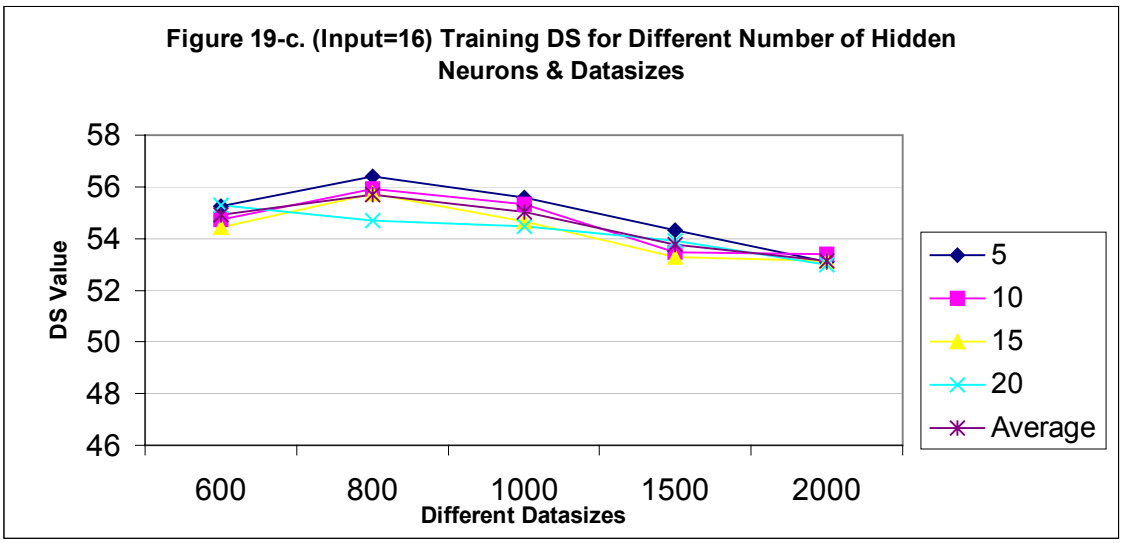
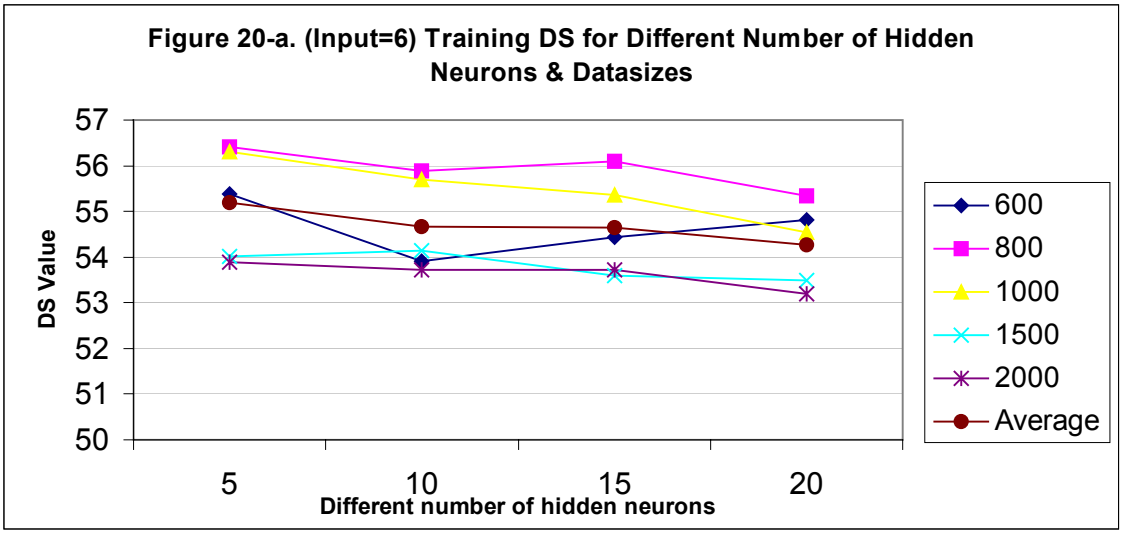
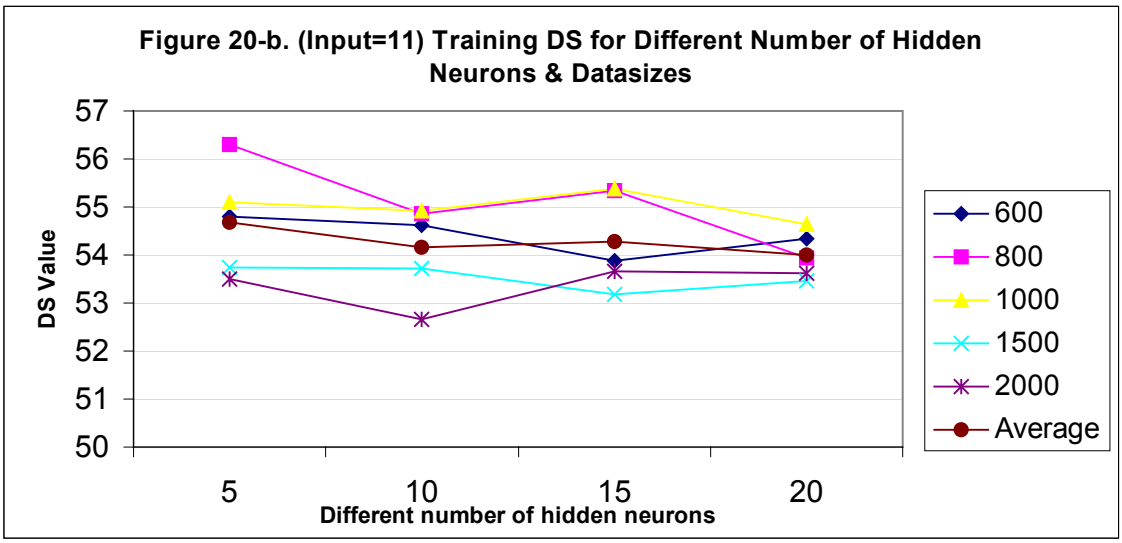


Figure 19 (a) to (c), Effects of Dataset Size as Measured by Average DS for Training (TR) on HSI Experiments (Input Number fixed to 6, 11, 16 respectively)





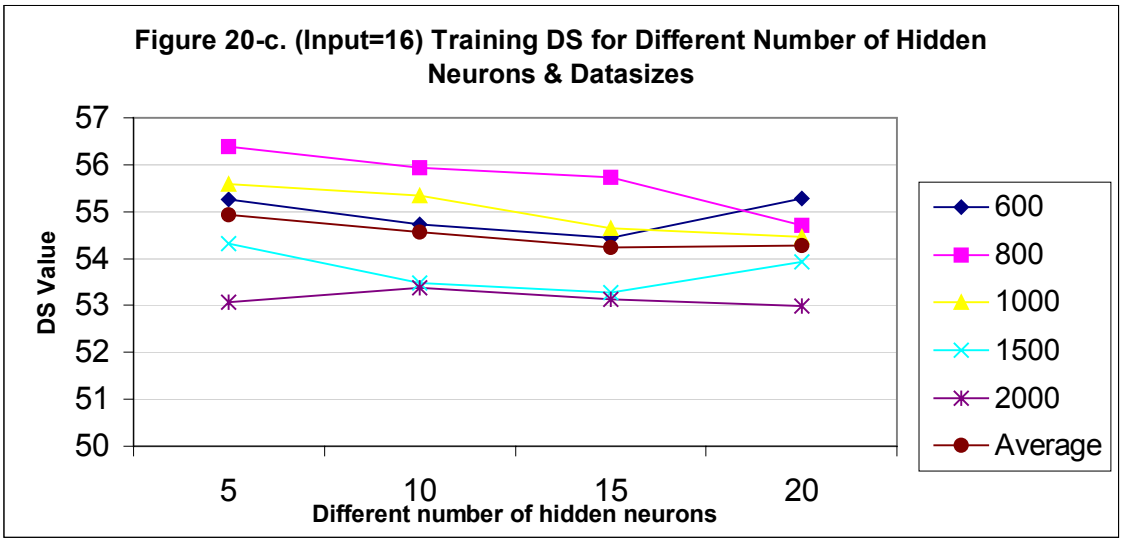


Figure 20 (a) to (c), Effects of Hidden Neuron Number as Measured by Average DS for Training (TR) on HSI Experiments (Input Number fixed to 6, 11, 16 respectively)

After analysing the effects and interrelationships of each variable based on both the MSE and DS results in the training process, we now move to the analysis in the testing process. Compared to the training process analysis, the analysis in the testing process is more important for us, because the performance of the neuron network model is mainly evaluated from the results in testing or real forecasting process. Figure 21 to 24 analyse the effects and interrelationships based on the MSE results in testing process, while Figure 25 to 28 based on the DS results in testing process. For the analysing methods used in testing process is the same with those used in training process, we just make a summary description for what are reflected from each figure in this process briefly. Figure 21 illustrates the effects of dataset size on the testing MSE results as well as the interrelationships between it and the other two variables. There is no consistent relationship exist between the dataset size and the testing MSE result. The relationship is influenced by other two variables obviously in “High” level. Figure 22 illustrates the effects of input number on the MSE results in testing process. For the conditions when hidden neuron number larger then 5, the inputs number has a relationship with the testing MSE results in the shape of reversed “V”. (When hidden neuron number is 5, the relationship is in the shape of “V”). Hidden neuron number obviously influences the impact of input number on the testing MSE result when hidden neuron number is small and influences weakly when hidden neuron number increases. On the other hand, dataset sizes have slight influence on the impact of input number on network performance.

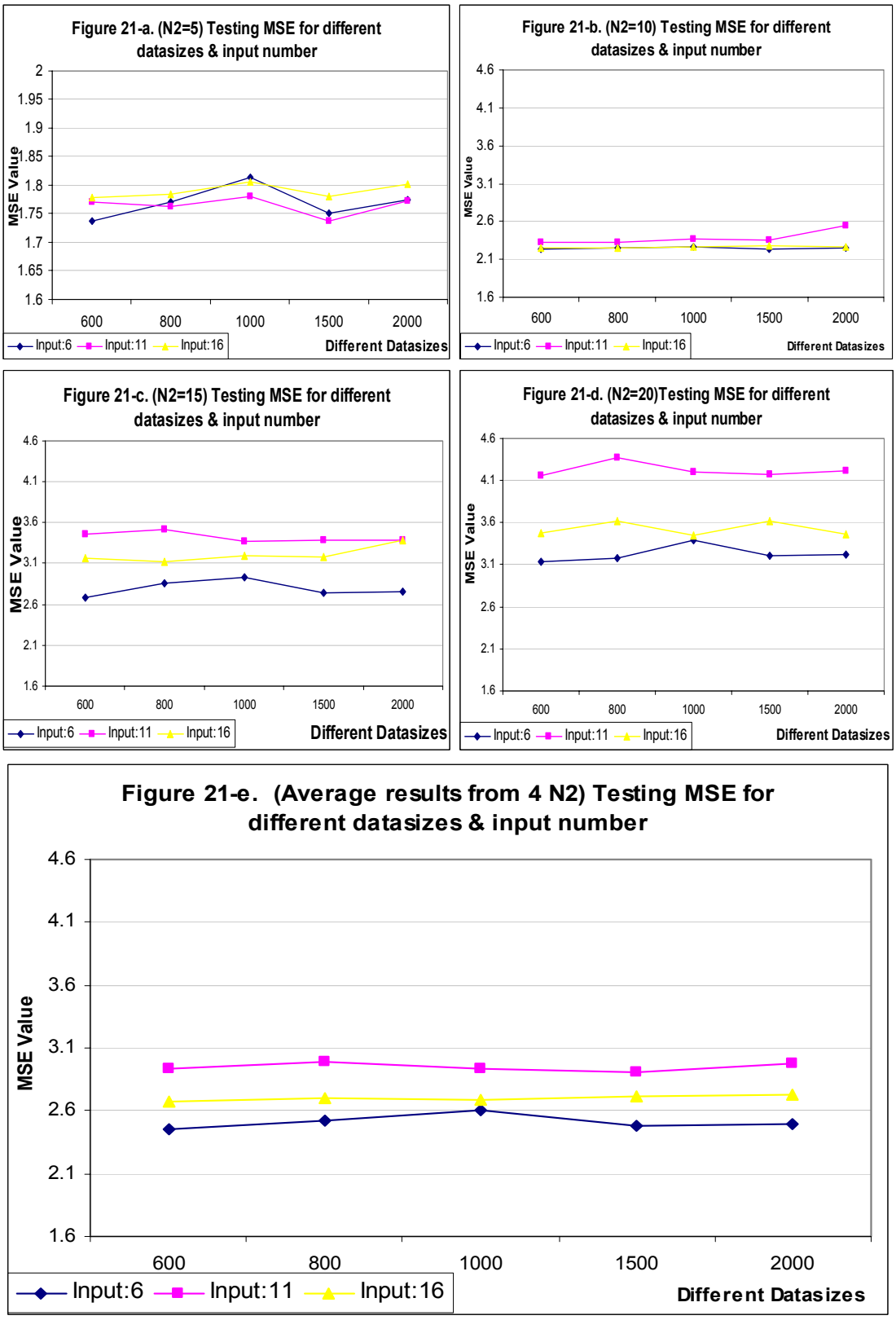


Figure 21 (a) to (e), Effects of Dataset Sizes as Measured by Average MSE for Testing (TR) on HSI Experiments (Hidden Neuron Number fixed to 5, 10, 15, 20 and Average respectively)

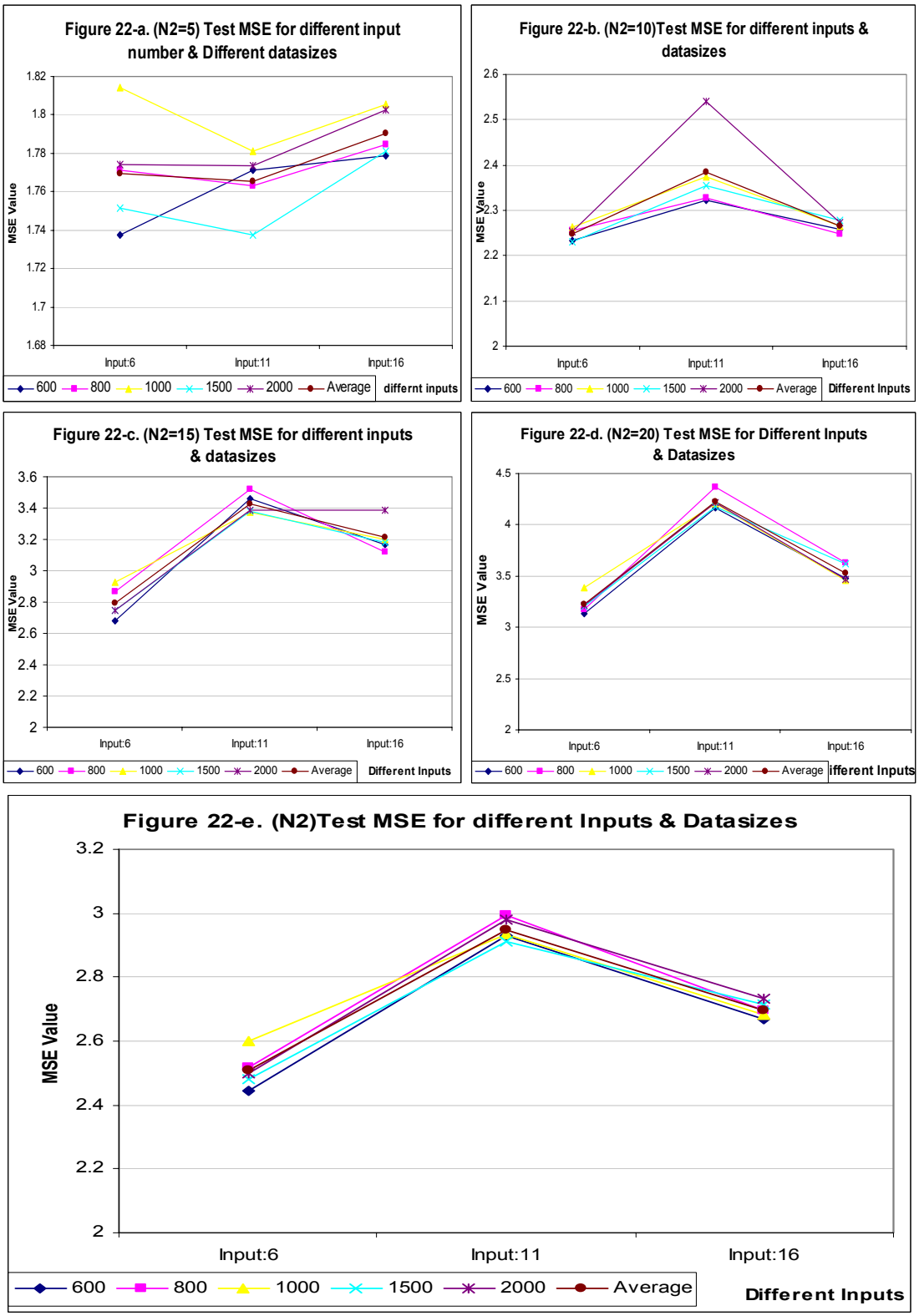
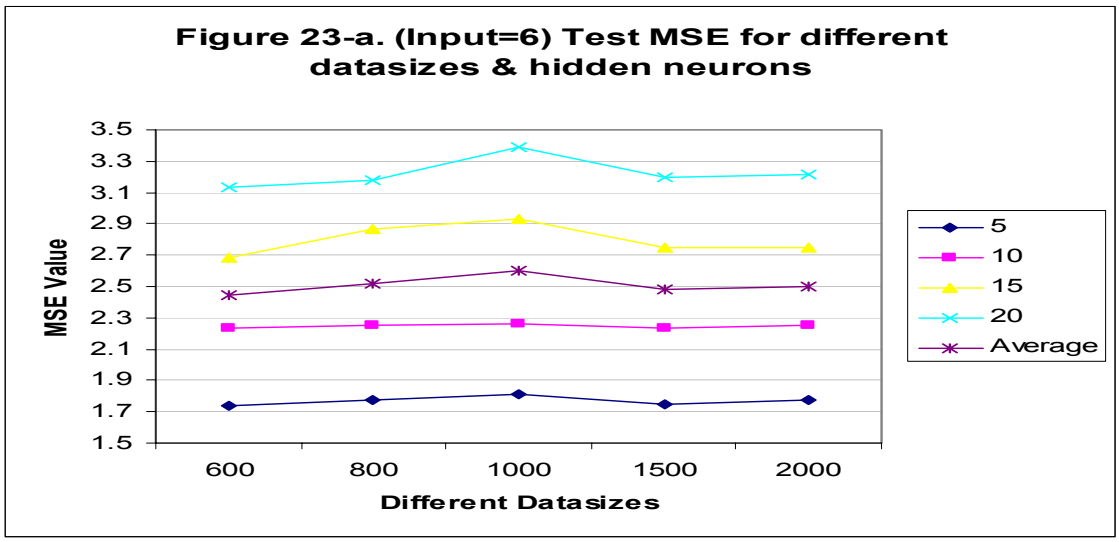
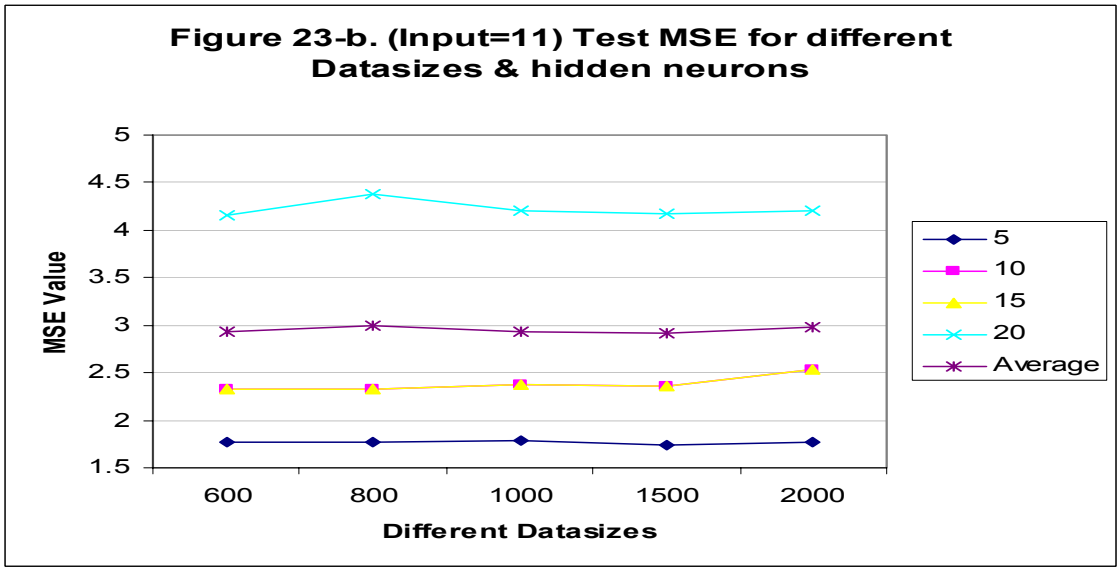


Figure 22 (a) to (e), Effects of Inputs Number as Measured by Average MSE for Testing (TR) on HSI Experiments (Hidden Neuron Number fixed to 5, 10, 15, 20 and Average respectively)

Figure 23 and figure 24 focus on illustrating the impact of dataset size and hidden neuron number on the MSE results in testing process. It is obviously that the dataset size has influence on the testing MSE results, but there are no consistent relationships between the dataset size and the MSE results. Both the input and hidden neuron number have “High” influences on the impact of dataset size on the testing MSE results. There are obvious relationship between the testing MSE results and the hidden neuron number and this relationship is consistent in all cases when other variables are different. The testing MSE result keeps rising gradually as the hidden neuron number increases, and the increase is very fast during the whole process. Thus it could be said that both the inputs number and dataset size have “Low” influence on the impact of hidden neuron number on testing MSE.

The effects of dataset size on the testing DS results are illustrated in figure 25. On the whole, the relationship between dataset size and DS results could be regard as consistent without obvious fluctuation. That is, the DS results for testing decrease gradually and reach the global minimum point as the dataset size changes from 600 to 1000. It increases from 1000 to 1500 and reverses to decrease again after 1500. Both the other variables have “Medium” influence on the impact of data size on network performance. As shown in figure 26 the input number has impact on the testing DS but this relationship is not consistent and is highly sensitive to the other two variables. But, on average, the best DS is obtained when input number is the smallest one.





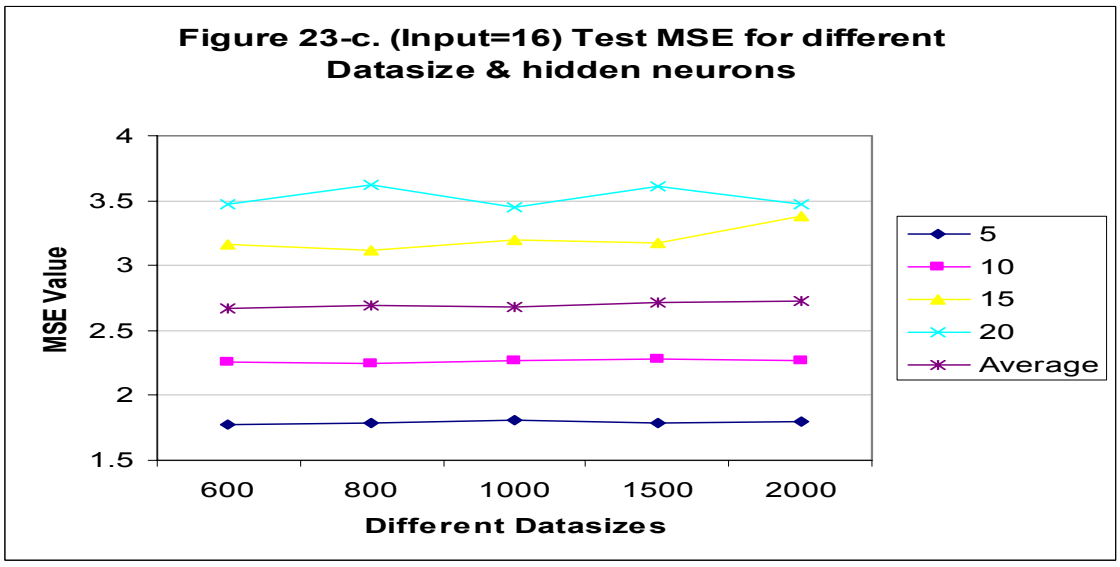
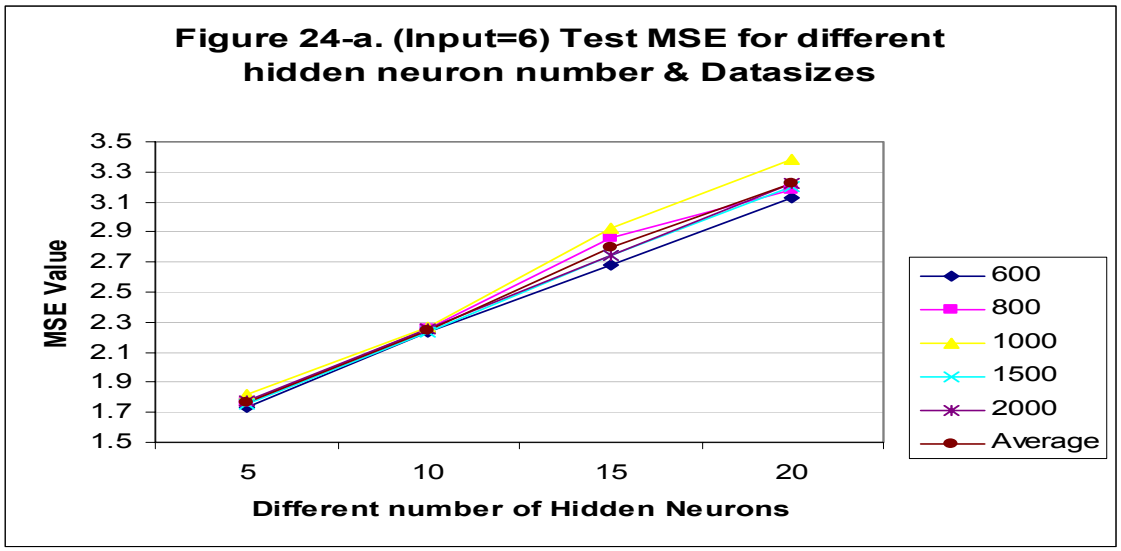
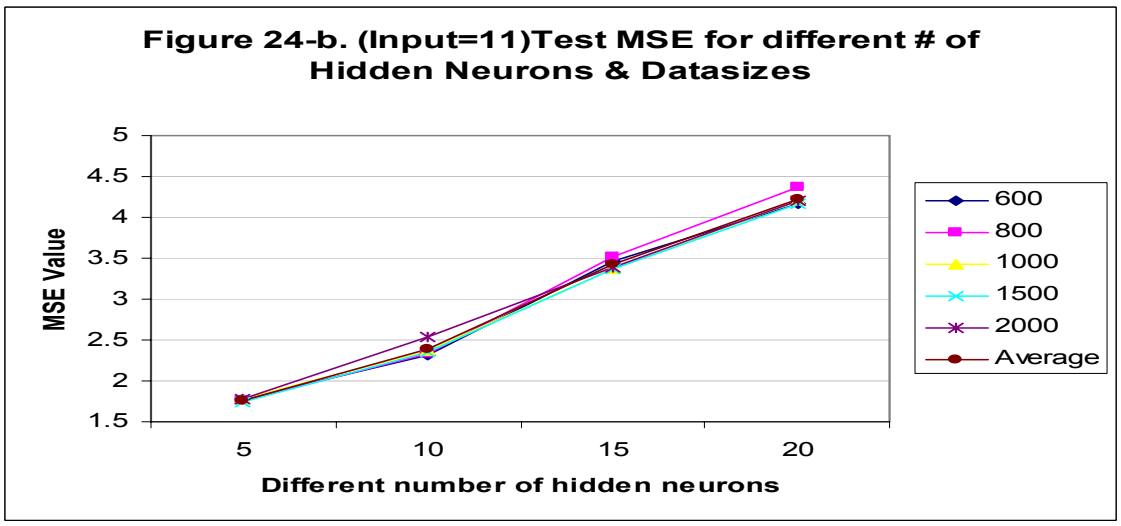


Figure 23 (a) to (c), Effects of Dataset Size as Measured by Average MSE for Testing (TR) on HSI Experiments (Input Number fixed to 6, 11, 16 respectively)





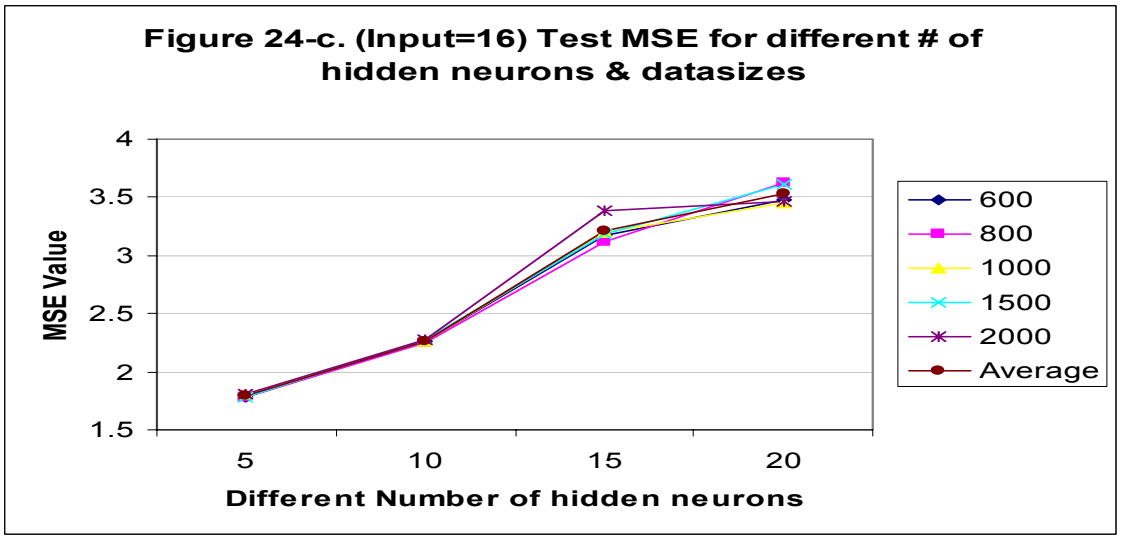
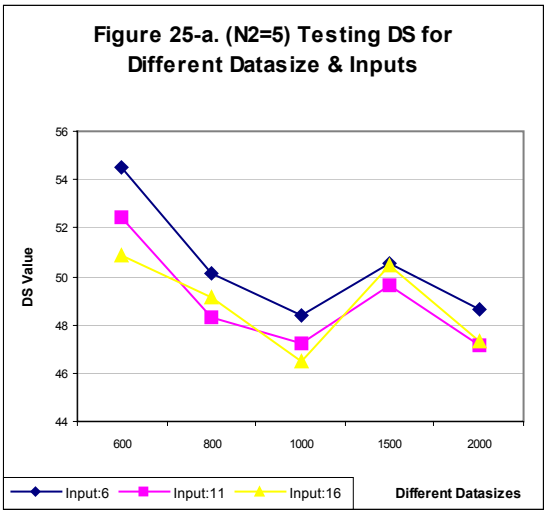
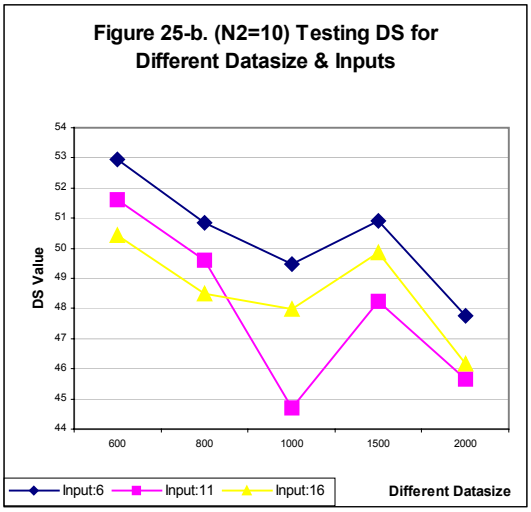
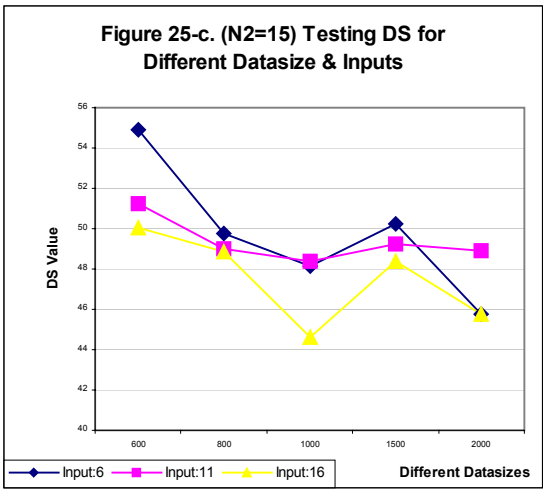
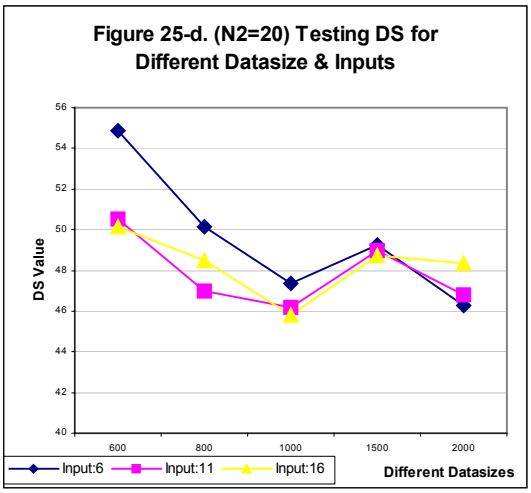


Figure 24 (a) to (c), Effects of Hidden Neuron Number as Measured by Average MSE for Testing (TR) on HSI Experiments (Input Number fixed to 6, 11, 16 respectively)









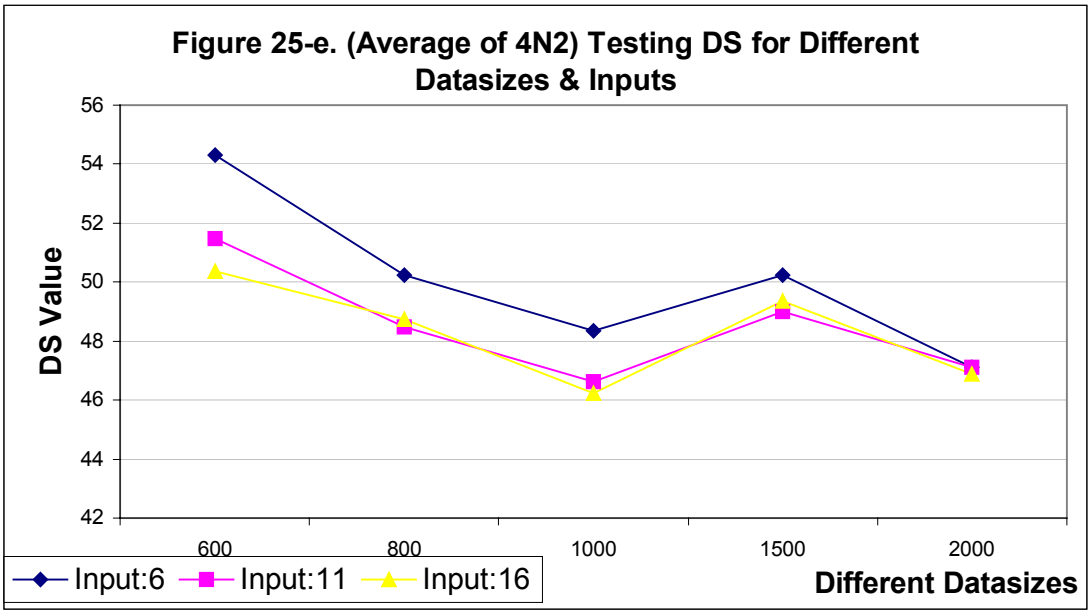
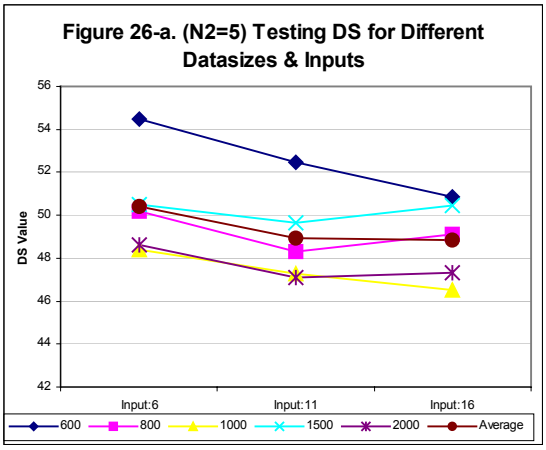
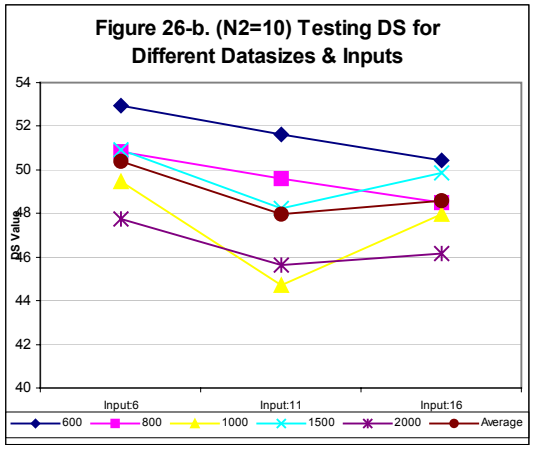
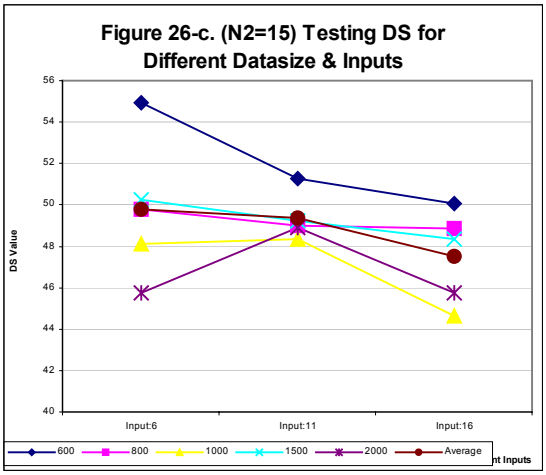
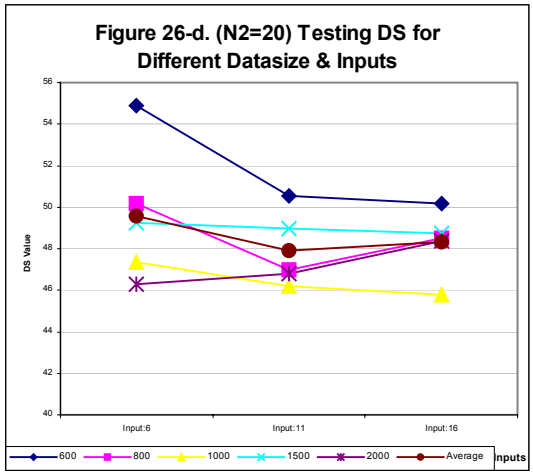


Figure 25 (a) to (e), Effects of Dataset Size as Measured by Average DS for Testing (TR) on HSI Experiments (Hidden Neuron Number fixed to 5, 10, 15, 20 and Average respectively)









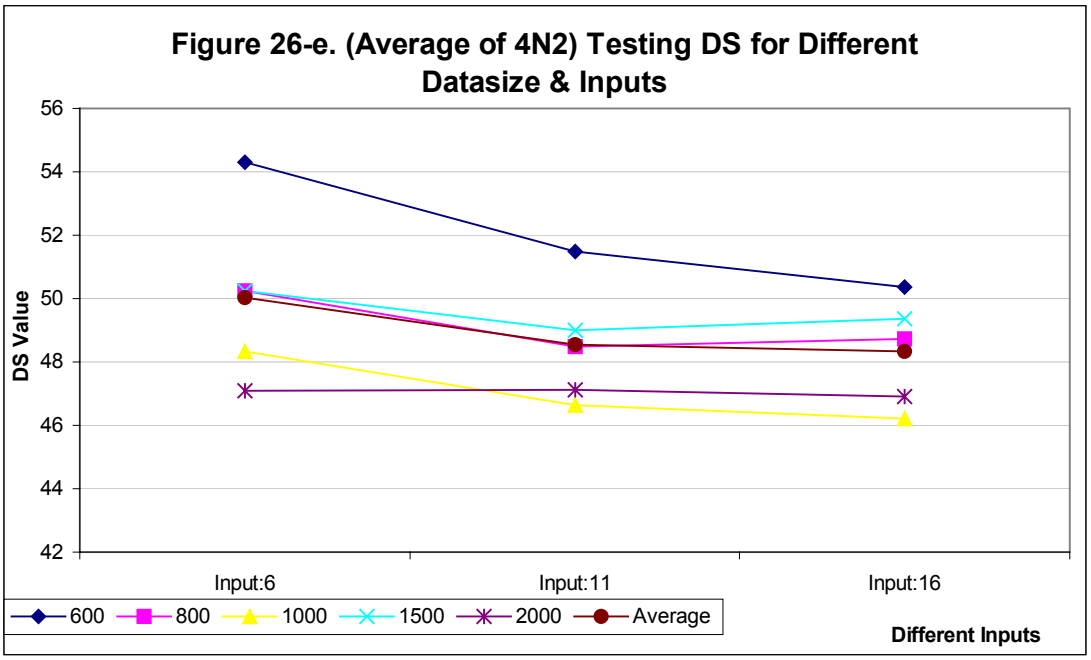
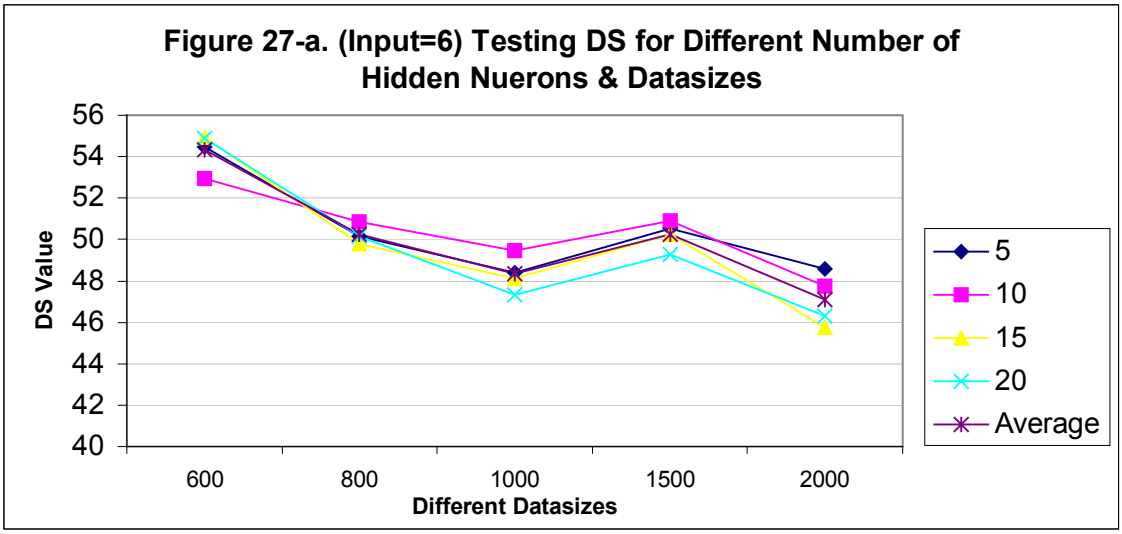
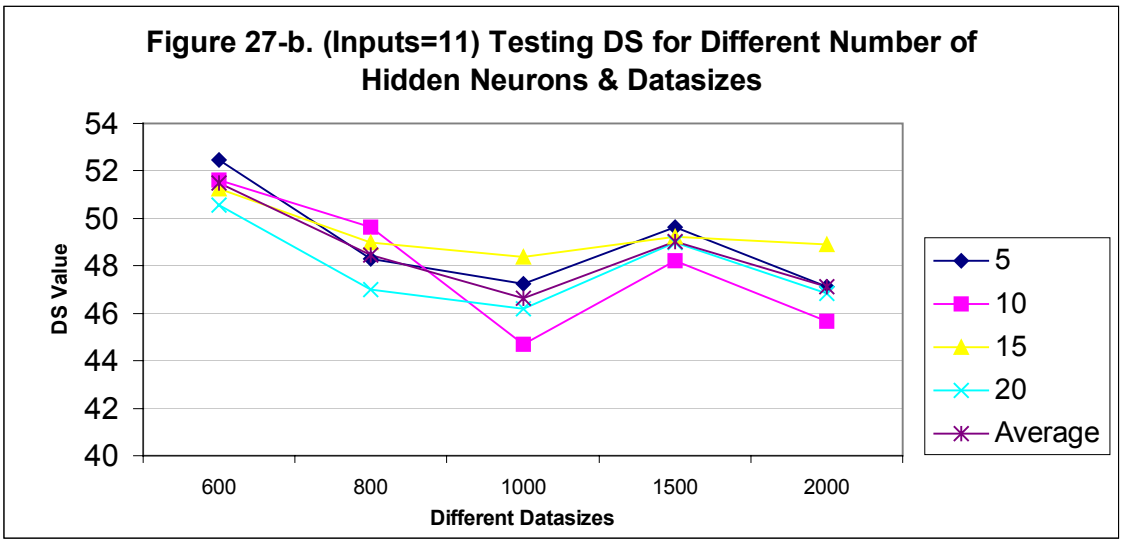


Figure 26 (a) to (e), Effects of Input Number as Measured by Average DS for Testing (TR) on HSI Experiments (Hidden Neuron Number fixed to 5, 10, 15, 20 and Average respectively)

Figure 27 and figure 28 illustrate the impact of dataset size and hidden neuron number on the testing DS results respectively. It's obviously that the dataset size has influences on the testing DS results and the relationship between dataset size and DS results are almost consistent in all cases. In another words, both the variables of input number and hidden neuron number give "Medium" influence on the impact of dataset size on testing DS results. The hidden neuron number has obvious effects on the DS performance of network model in the testing process, but the relationship is not consistent. As we can see clearly that for any particular fixed number of inputs, the effect of hidden neuron number on DS results changes substantially when dataset size changes. On the other hand, the relationship between hidden neuron number and DS results also changes substantially when the variable of input number changes. Thus it has no doubt that both the variable of dataset size and input number highly influence the impact of hidden neuron number on the testing DS results.

From the detailed analysis on the interrelationships between the three factors in neural network modeling, we notice that in most situations such interactive relationships may have obvious influence on the network performance. In some cases, particular variable may have great influence on other variables, while, in other cases, such influences may be weak.





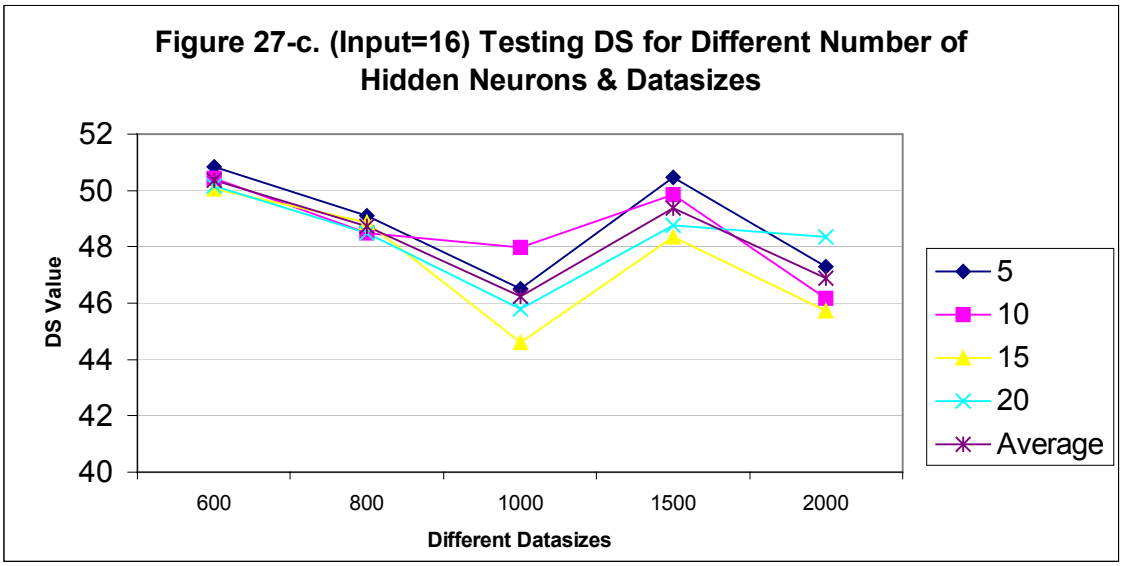
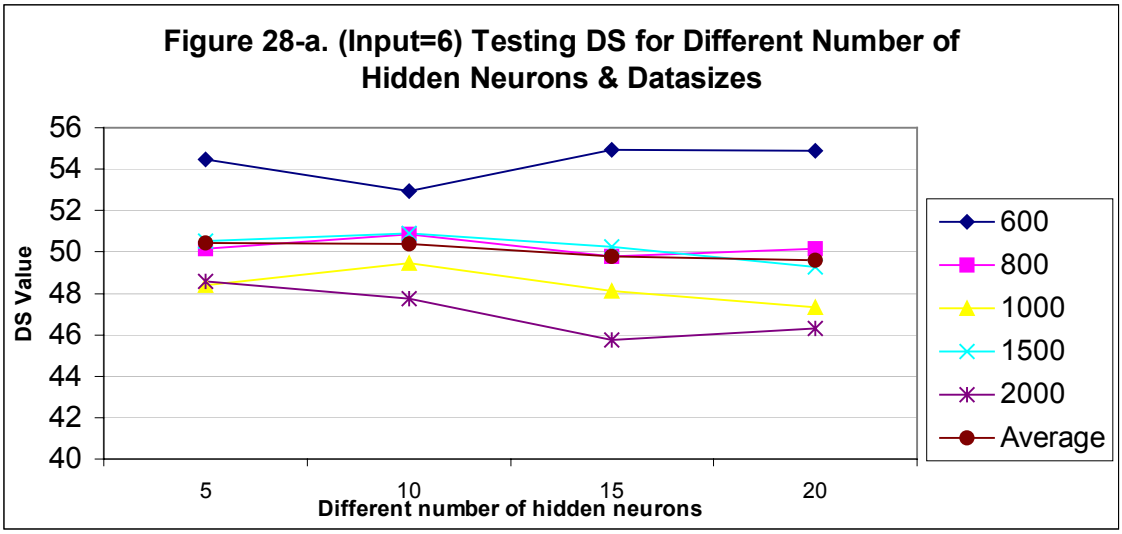
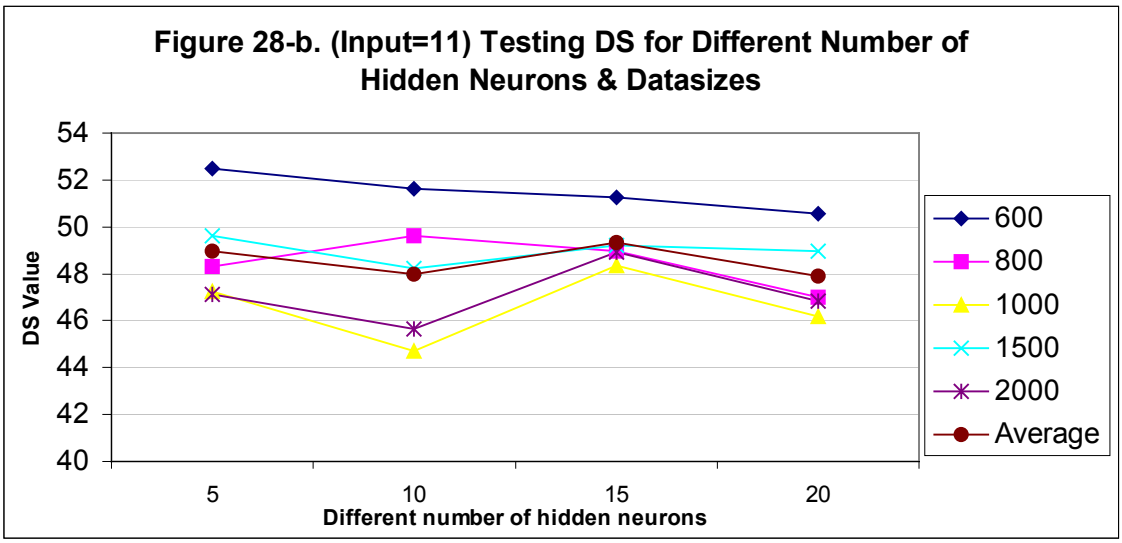


Figure 27 (a) to (c), Effects of Dataset Size as Measured by Average DS for Testing (TR) on HSI Experiments (Input Number fixed to 6, 11, 16 respectively)





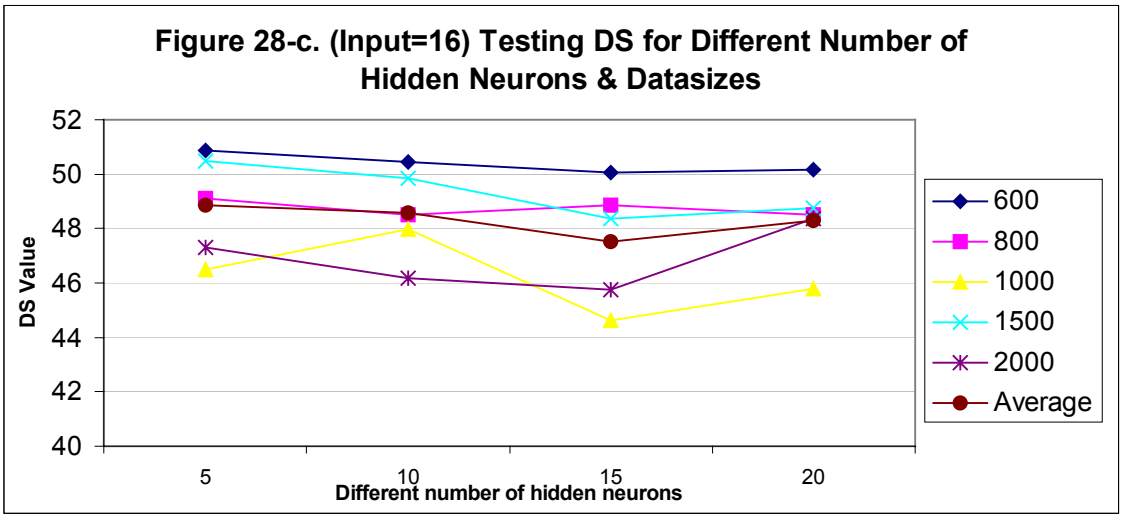


Figure 28 (a) to (c), Effects of Hidden Neuron Number as Measured by Average DS for Testing (TR) on HSI Experiments (Input Number fixed to 6, 11, 16 respectively)

Understanding such kind of effects of variables on each other in different stock markets may have great value for both the researchers and practitioners in financial markets. For there are totally 160 charts obtained from 5 different markets under two neural network models, we are unable to plot all the charts in this thesis even in the appendix, but we can briefly summarize all the important findings about the interrelationships between the three variables in all cases. Table 8 and table 9 summarize the interrelationships between these three major variables in five stock markets by TRNN and SSPQN models respectively. Though the analysing results getting from different neural network models are not consistent in all cases, some consistent general findings from both models could be drawn as below:

(1). On average, the interrelationships between these three variables measured under DS criterion are obviously stronger than those measured under MSE criterion in both training and testing processes. Most interrelationships under the DS criterion are “high” or “medium”, while most such relationships under MSE criterion are “Low”.

(2). All interrelationships between variables are “low” under training MSE results, besides Hidden Neuron Number having above average influence on the impact of Input Neuron Number on network performance.

(3). The interrelationships under testing MSE results are quite similar with those under training MSE results, besides, under testing MSE results, both of the network architecture factors have stronger influence on the impact of Dataset Size on network performance.

(4). The differences of the analysing results among different stock markets are smaller under MSE results and larger under DS results. In another word, the stronger the average interrelationships between the variables, the wider the differences between the results getting from different stock markets.

In this thesis, there are two kinds of criteria used for the network performance evaluation. The Mean Squared Error and Directional Symmetry. By comparing between Table 8 and Table 9, we can notice that under MSE evaluation criterion, the analysis results (hence the conclusions) by two different training algorithms are very similar. In all the 60 items under MSE criterion, only few analysis results by different training algorithms are different. Under DS criterion, more obvious differences are observed under different models. It seems that when variables like network topology and data set are same, the interrelationships between these variables should be consistent even when different training algorithms are used. But, there may be some slight effects of training algorithms on the interrelationships between those variables like network topology and data set. The slight effects maybe caused by the difference between different training algorithms, for example, the different stop criteria for the training process. The two training algorithms in this thesis use different convergence thresholds to stop the training process. Maybe the interrelationships between these variables are sensitive on the different stop criteria of the training algorithms. Whether such difference between different training algorithms may have some slight effects on the interrelationships between these variables is a very interesting issue. Maybe under some

particular training algorithms, the interrelationships could be more obviously, while under other training algorithms, the interrelationships would be less obviously. Anyway, training algorithm itself is also a very important factor that may affect the network performance. Different training algorithms may lead to different network performance. As we know, the network topology determines the number of weights (thus the number of variables and terms) for the network model and the data set determines the points the model simulates in the training process. For number of weights represent the complexity of the network model, and data set represent the property of the points to be simulated by the model, it's no strange that there are some interrelationship between these variables on the network performance. Training the neural network involves propagating the error to adjust the set of weights to minimize the error function. Different training algorithms involve different way to adjust the set of weights to accurately simulate the sample points. From this point of view, training algorithm is the link between these variables under the framework of network model, thus different kinds of algorithms may have different effects on the interrelationships between these variables. How and on what extent the training algorithm affects the interrelationships between the variables could be a very interesting topic in our further research.

In my opinion, though, the analysis results on training process may not give a direct reference on the later practices in financial forecasting, it provide us a deeper insight on the process of network training. By comparing the results from training and testing processes, we found some interesting findings. For example, our MSE results show that, in training process, the network

architecture variables (input and hidden neurons) have low influence on the impact of Data Size on network performance. While, in testing process, such influence become more obviously and the choice of network architecture obviously influence the relationship between data size and network performance. As training process is the repeated applications of the same data set, as long as the network architecture is complex enough for the desired accuracy, it won't have obvious effects on the impact of data set on network performance. It is possible to approximate a continuous function that may achieve the desired accuracy with a single hidden layer (Cybenko, 1989; Hecht-Nielsen, 1990; Hertz, 1991; Hornick, 1989). Specification of the internal architecture involves tradeoff between fitting accuracy and generalization ability. The architecture of the network in testing process is very important for determining the generalization ability of the network, thus affecting the data set impact on network performance.

Of course, the most important things we discovered from these results are those unique characteristics that each particular stock market has, which are different from each other. These unique characteristics are valuable for our further research in that particular market. Based on the interactive sensitivity analysis between these major factors in neural network modeling, we discovered that the network performance is not only sensitive to each of these major factors individually, but also affected by the interrelationships between these factors. From this study, we could see that in some cases such interrelationships may have obvious influence on the network performance, thus the issue on how such interrelationships will affect the network

performance must also be considered during the neural network constructing for financial forecasting. For example, in the case that Hidden Neuron Number has great influence on the impact of Dataset Size on network performance, we should consider both the direct and indirect roles that Hidden Neuron Number act in the neural network constructing. The general findings from two different models as well as the unique interrelationships between these three major variables under five different stock markets could be a valuable reference for both the academic researchers and the investment practitioners in neural network constructing for these particular markets.

Summary of the interrelationship between variables in each particular market by TRNNs						
(1).For Training Process						
Based on MSE results	**Dataset Sizes**		**Hidden Neuron Number**		**Input Neuron Number**	
Extent of other variables affecting the specified Variable	Hidden Neuron Number	Input Neuron Number	Dataset Sizes	Input Neuron Number	Dataset Sizes	Hidden Neuron Number
DAX	Low*	Low	Low	Low	Low	Medium
DJIA	Medium*	Low	Low	Low	Low	Medium
FTSE	Low	Low	Low	Low	Low	Medium
HSI	Low	Low	Low	Low	Low	Medium
NASDAQ	Low	Low	Low	Low	Low	Low
Based on DS results						
DAX	Low	Medium	Low	Medium	High	High
DJIA	Medium	Medium	High	High	High	High
FTSE	High*	High	High	High	High	High
HSI	Medium	Low	Medium	Medium	High	Medium
NASDAQ	High	High	High	High	Medium	Medium
(2).For Testing Process						
Based on MSE results	**Dataset Sizes**		**Hidden Neuron Number**		**Input Neuron Number**	
Extent of other variables affecting the specified Variable	Hidden Neuron Number	Input Neuron Number	Dataset Sizes	Input Neuron Number	Dataset Sizes	Hidden Neuron Number
DAX	High	High	Low	Low	Low	Medium
DJIA	High	High	Low	Low	Low	Medium
FTSE	Medium	Medium	Low	Low	Low	Medium
HSI	Medium	Medium	Low	Low	Low	Medium
NASDAQ	High	High	Low	Low	Low	Medium
Based on DS results						
DAX	Medium	High	High	High	High	High
DJIA	High	High	Medium	High	High	High
FTSE	Medium	Medium	High	High	High	High
HSI	Low	Low	High	High	Medium	Medium
NASDAQ	Medium	High	High	High	High	High
*H, *M and *L represents other variables has high, medium or low influence on the impact of target variable on network model performance respectively						

Table 8 Summary of the interrelationships between major variables in each particular market by TRNN model

Summary of the interrelationship between variables in each particular market by SSPQN						
(1).For Training Process						
Based on MSE results	**Dataset Sizes**		**Hidden Neuron Number**		**Input Neuron Number**	
Extent of other variables affecting the specified Variable	Hidden Neuron Number	Input Neuron Number	Dataset Sizes	Input Neuron Number	Dataset Sizes	Hidden Neuron Number
DAX	Low	Low	Low	Low	Low	Medium
DJIA	Low	Low	Low	Low	Low	Medium
FTSE	Low	Low	Low	Low	Low	Medium
HSI	Low	Low	Low	Low	Low	Medium
NASDAQ	Low	Low	Low	Low	Low	Low
Based on DS results						
DAX	Low	Medium	High	Medium	Medium	Medium
DJIA	Low	Medium	High	Medium	Medium	Medium
FTSE	Medium	High	High	High	High	High
HSI	Low	Medium	Medium	Medium	High	High
NASDAQ	Low	Medium	High	High	High	High
(2).For Testing Process						
Based on MSE results	**Dataset Sizes**		**Hidden Neuron Number**		**Input Neuron Number**	
Extent of other variables affecting the specified Variable	Hidden Neuron Number	Input Neuron Number	Dataset Sizes	Input Neuron Number	Dataset Sizes	Hidden Neuron Number
DAX	High	High	Low	Low	Low	Medium
DJIA	Medium	High	Low	Low	Low	Medium
FTSE	Medium	High	Low	Low	Low	Medium
HSI	Low	Low	Low	Low	Low	Medium
NASDAQ	Medium	High	Low	Low	Low	Medium
Based on DS results						
DAX	Medium	Medium	High	High	High	Medium
DJIA	Medium	High	High	High	High	High
FTSE	Low	Medium	High	High	High	Medium
HSI	Low	Medium	High	High	Medium	Low
NASDAQ	Medium	High	High	High	High	Medium
*H, *M and *L represents other variables has high, medium or low influence on the impact of target variable on network model performance respectively						

Table 9 Summary of the interrelationships between major variables in each particular market by SSPQN model

4.4. Proposed network topology

Table 10 illustrates the optimal neural network structures and optimal dataset sizes for the five different stock markets based on the training MSE and DS performances. A very interesting finding from MSE results is that, besides the optimal dataset size of HSI by SSPQN, the optimal network topologies (both in network structure and dataset size) for all the five markets are absolutely the same: the optimal number of input neuron, hidden neuron and optimal dataset size are 11, 5 and 2000 respectively for both models in the training process. These results seem demonstrate that, in training process, the larger the dataset size the better. Particularly, in the case of HSI index forecasting, the best dataset size is 600 in training process. For the hidden neuron number, there seems no doubt that the lesser the hidden neuron number the better the results in training process. Based on the training DS results, the optimal number of hidden neurons is still 5 in all cases. This finding re-confirm that the lesser the hidden neurons the better the training results. It's also no strange that based on training DS results, stock markets show some differences in the optimal number of input neurons and optimal dataset sizes. However, it is just such kind of difference that provides us a nice way to distinguish the particular financial time series characteristics among those different stock markets being studied.

Although the study of the optimal topology in training process may provide us some valuable references and hints on neural network modeling, what we really care about in this thesis is the optimal network topology for each particular market in the testing process. The optimal network topologies in the testing process will finally determine the ultimate proposed neural network

models for performance comparison in the next chapter. For the problem of overtraining exists in neural network training process, a good performance in training may not guarantee a comparable good results in testing process or in real forecasting, thus the optimal topologies for training are normally different from those for testing. Table 11 illustrates the optimal network topologies for different stock markets under both models based on the testing results, which will be the proposed network architectures for the stock index increments forecasting in next chapter. Based on the average testing MSE results, the optimal hidden neuron numbers in all cases are still 5. Thus our experiments show that for both the training and testing processes, the lesser the hidden neuron number the better the network performance in MSE results. On the other hand, the optimal numbers of input neurons are all 11 by TRNN model for all stock markets except DAX. The optimal dataset sizes are not consistent for different markets and models. There are no obvious conclusion could be drawn from the optimal topology based on testing DS results, besides that the best input neuron number in most cases by SSPQN model are 6, that is the smaller the input neuron number the better. Taken NASDAQ index as example, figure 29 illustrates the optimal network structures for one-day-ahead prediction by TRNN model under the criteria of both MSE and DS.

(1). Based on the Criteria of Average MSE on Training					
Optimal Structure for Trust Region Dogleg Method Based Neural Networks					
	DAX	**DJIA**	**FTSE**	**HSI**	**NASDAQ**
Optimal Number of Input Neuron	**11**	**11**	**11**	**11**	**11**
Optimal Number of Hidden Neuron	**5**	**5**	**5**	**5**	**5**
Optimal Dataset Size	**2000**	**2000**	**2000**	**2000**	**2000**
Optimal Structure for Parallel Quasi-Newton Method Based Neural Networks					
	DAX	**DJIA**	**FTSE**	**HSI**	**NASDAQ**
Optimal Number of Input Neuron	**11**	**11**	**11**	**11**	**11**
Optimal Number of Hidden Neuron	**5**	**5**	**5**	**5**	**5**
Optimal Dataset Size	**2000**	**2000**	**2000**	**600**	**2000**
(2). Based on the Criteria of Average DS on Training					
Optimal Structure for Trust Region Dogleg Method Based Neural Networks					
	DAX	**DJIA**	**FTSE**	**HSI**	**NASDAQ**
Optimal Number of Input Neuron	**6**	**6**	**6**	**6**	**16**
Optimal Number of Hidden Neuron	**5**	**5**	**5**	**5**	**5**
Optimal Dataset Size	**2000**	**2000**	**1500**	**800**	**600**
Optimal Structure for Parallel Quasi-Newton Method Based Neural Networks					
	DAX	**DJIA**	**FTSE**	**HSI**	**NASDAQ**
Optimal Number of Input Neuron	**16**	**6**	**16**	**16**	**16**
Optimal Number of Hidden Neuron	**5**	**5**	**5**	**5**	**5**
Optimal Dataset Size	**2000**	**2000**	**600**	**800**	**2000**

Table 10 Optimal Network Topology Based on Training Results for Each Market

(1). Based on the Criteria of Average MSE on Testing					
Optimal Structure for Trust Region Dogleg Method Based Neural Networks					
	DAX	**DJIA**	**FTSE**	**HSI**	**NASDAQ**
Optimal Number of Input Neuron	**16**	**11**	**11**	**11**	**11**
Optimal Number of Hidden Neuron	**5**	**5**	**5**	**5**	**5**
Optimal Dataset Size	**600**	**800**	**800**	**1500**	**600**
Optimal Structure for Parallel Quasi-Newton Method Based Neural Networks					
	DAX	**DJIA**	**FTSE**	**HSI**	**NASDAQ**
Optimal Number of Input Neuron	**16**	**6**	**11**	**6**	**16**
Optimal Number of Hidden Neuron	**5**	**5**	**5**	**5**	**5**
Optimal Dataset Size	**600**	**800**	**600**	**600**	**800**
(2). Based on the Criteria of Average DS on Testing					
Optimal Structure for Trust Region Dogleg Method Based Neural Networks					
	DAX	**DJIA**	**FTSE**	**HSI**	**NASDAQ**
Optimal Number of Input Neuron	**6**	**6**	**11**	**6**	**16**
Optimal Number of Hidden Neuron	**15**	**20**	**10**	**15**	**20**
Optimal Dataset Size	**600**	**800**	**800**	**600**	**800**
Optimal Structure for Parallel Quasi-Newton Method Based Neural Networks					
	DAX	**DJIA**	**FTSE**	**HSI**	**NASDAQ**
Optimal Number of Input Neuron	**11**	**6**	**6**	**6**	**6**
Optimal Number of Hidden Neuron	**5**	**20**	**20**	**5**	**20**
Optimal Dataset Size	**600**	**800**	**600**	**600**	**1000**

Table 11 Optimal Network Topology Based on Testing Results for Each Market

The optimal neural network internal architecture is an important issue that affects the network performance in forecasting. Despite the importance, there is no standard criterion on the number of hidden neurons. However, some "rules of thumb" can be found in the literature. Following are some examples:

- The number of hidden neurons should be less then twice of the input nodes [9]
- For a three layer network with n input neurons and m output neurons, Masters [36] proposes $\sqrt{n \times m}$ neurons for the hidden layer
- Katz [30] states that the optimal number of hidden layer neurons will generally be found between 0.5 to 3 times of the number of the input neurons.
- Baily and Thompson [7] suggest that for a three layer neural networks, the number of neurons for the hidden layer should be 75% of the number of the input neurons.

Our experiment results on the optimal number of hidden neurons are not consistent with what Masters, Katz and Baily had suggested, but consistent with the first suggestion that the number of hidden neurons should be less than twice of the input nodes. On the other hand, our results show that for different optimal input neuron numbers, the optimal hidden neuron numbers in all cases are the same, which, on some extent can reflect that the optimal hidden neural number may not influenced by the number of input neurons in the networks.

As for each particular stock market, the proposed optimal network topologies for forecasting are different for TRNN and SSPQN models. The prediction performance comparison between these two models is based on their corresponding optimal network topologies in five stock markets determined from testing process. For example, for DS performance comparison in

NASDAQ index prediction, the TRNN model with 16 inputs, 20 hidden neurons and dataset size of 800 are compared with the SSPQN model with 6 inputs, 20 hidden neurons and dataset size of 1000. The performance comparison between the TRNN and SSPQN models based on their corresponding optimal network topologies in five stock markets is analysed in the next Chapter.

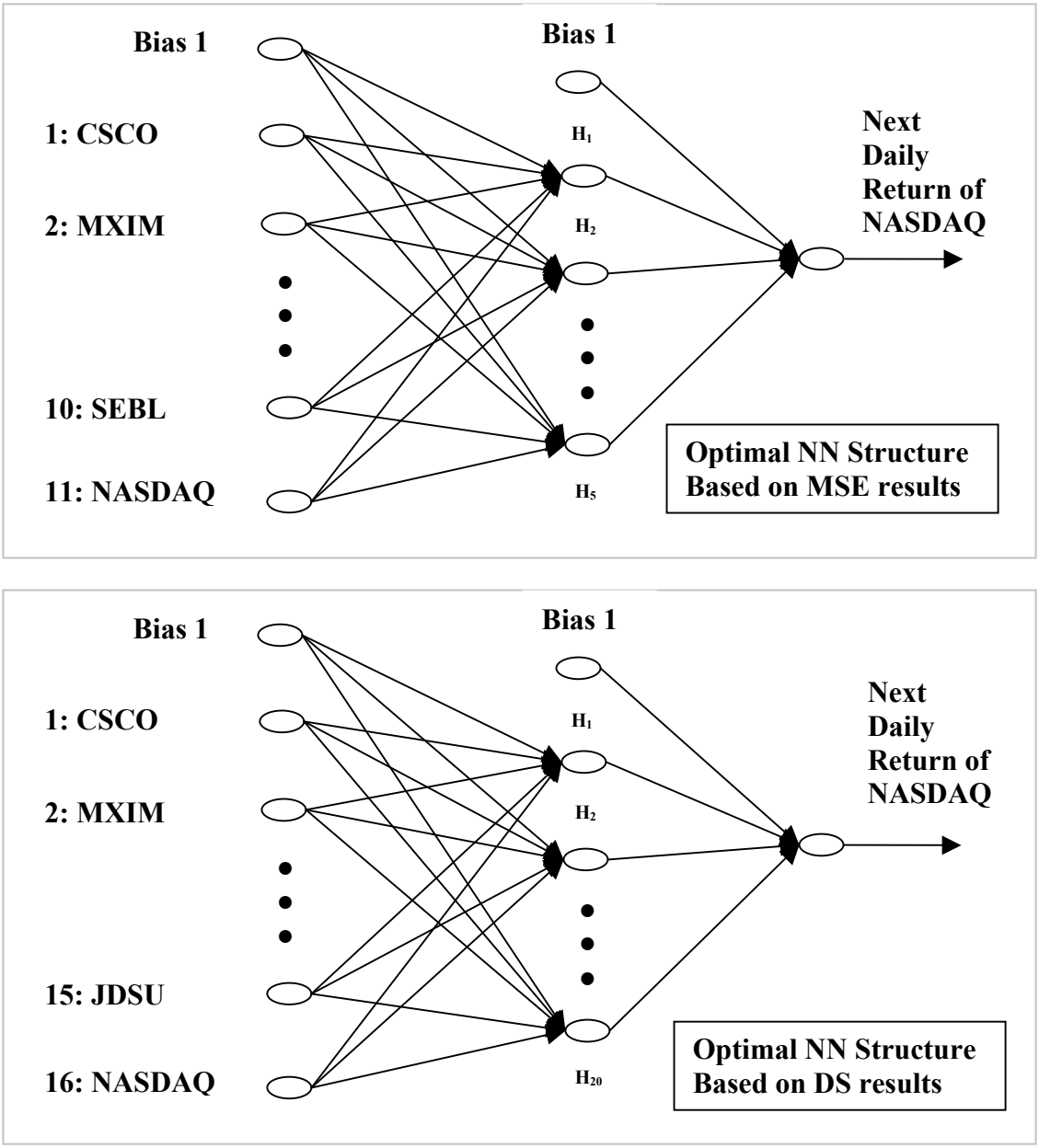


Figure 29 Optimal NN-Structures for One-Day Ahead Forecasting for NASDAQ Index by Trust Region Neural Network Model

Chapter 5

Comparisons and Performance Analysis

In Chapter 5, we conduct stock index increments prediction by both component-based neural network forecasting models trained by trust region algorithms and SSPQN algorithms. Computational results obtained from five stock markets are disclosed. We then analyze which kind of model gives better prediction accuracy in the aspect of one-step sign prediction rate.

Furthermore, additional performance analysis would be conducted on the trust region algorithms based neural network model. Performance comparisons between our purposed forecasting model and models proposed by other researchers in the similar markets would be conducted in order to know whether our purposed model improve the network forecasting accuracy.

5.1. Stock Index Increments Forecasting

After determining the optimal network topologies for both the TRNN and SSPQN models, further experiments are conducted to compare the neural network models' performance in forecasting index increments in five major stock markets. Each experiment is trained 50 times with random selected different sets of starting arc weights. The final set of training arc weights that give the best result in training is then applied in the testing process.

The final estimation of the performance in forecasting is made by means of the one-step sign prediction rate ξ defined on T as follows:

$$\xi = \frac{1}{|T|} \sum_{t \in T} [HS(\Delta C_t \bullet \Delta G_t) + 1 - HS(|\Delta C_t| + |\Delta G_t|)] \qquad (14)$$

where $\Delta C_t = C_t - C_{t-1} = C_{t-1} \times R_t$ is the actual price change at time $t \in T$ and $\Delta G_t = G_t - C_{t-1} = C_{t-1} \times GR_t$ is the guessed price change at the same time step, where GR_t is the guessed return at time t. Note that we assume to know the value of C_{t-1} to evaluate ΔG_t. HS is a modified Heaviside function, $HS(x)=1$ for $x>0$ and 0 otherwise. The argument of the summation in (14) gives one if ΔC_t and ΔG_t are non-zero and with same sign, or if ΔC_t and ΔG_t are both zero. For our model uses both the index and component stock returns as network inputs, thus the sign prediction rate ξ can also be expressed without change in value as follows:

$$\xi = \frac{1}{|T|} \sum_{t \in T} [HS(R_t \bullet GR_t) + 1 - HS(|R_t| + |GR_t|)] \qquad (15)$$

In other words, ξ is the probability of a correct guess on the sign of the price increment estimated on T. In fact, the probability to make a correct guess on the sign of the increment seems independent from the magnitude of the increment ΔC itself.

To check and compare the performance of our proposed network models, the optimal network topology is applied to perform one-day ahead prediction of five different indices (DAX, DJIA, FTSE, HSI, and NASDAQ) daily increments from 14 May 2002 to 30 September 2002. Table 12 illustrates the prediction performance of the two models in one-step sign prediction rate.

Models	Average Accuracy		Best Accuracy	
	TRNN	**SSPQN**	**TRNN**	**SSPQN**
DAX	60.27%	57.30%	68%	61%
DJIA	61.46%	58.56%	70%	67%
FTSE	65.51%	57.87%	73%	64%
HSI	64.39%	63.35%	74%	70%
NASDAQ	64.86%	62.03%	73%	67%
Average	63.30%	59.82%	71.60%	65.80%

Table 12, Performance Comparison between Two Models

It is easy to notice that, in all the five markets predictions, the Trust Region Neural Networks always outperform the SSPQN neural networks. The average one-step sign prediction rates by TRNN model are higher than 60% in all the five stock markets forecasting. Forecasting accuracy values at or above 60% are statistically significant [50]. Furthermore, the average accuracy in FTSE-100, HSI and NASDAQ even reaches as high as 65.51%, 64.39% and 64.86% respectively. For SSPQN model, only two prediction results are more than 60% accuracy. In the aspect of best testing one-step sign prediction, the accuracy rate of TRNN model even exceed 70% in four markets of DJIA, FTSE, HSI and NASDAQ. For the SSPQN model, only in HSI index prediction, the best accuracy reaches 70%. In fact, the best prediction result obtained by the TRNN model is 74%, both for network training and testing. The proposed Trust Region Neural Networks deliver impressive results for forecasting the financial indices, especially in the aspect of index price increments.

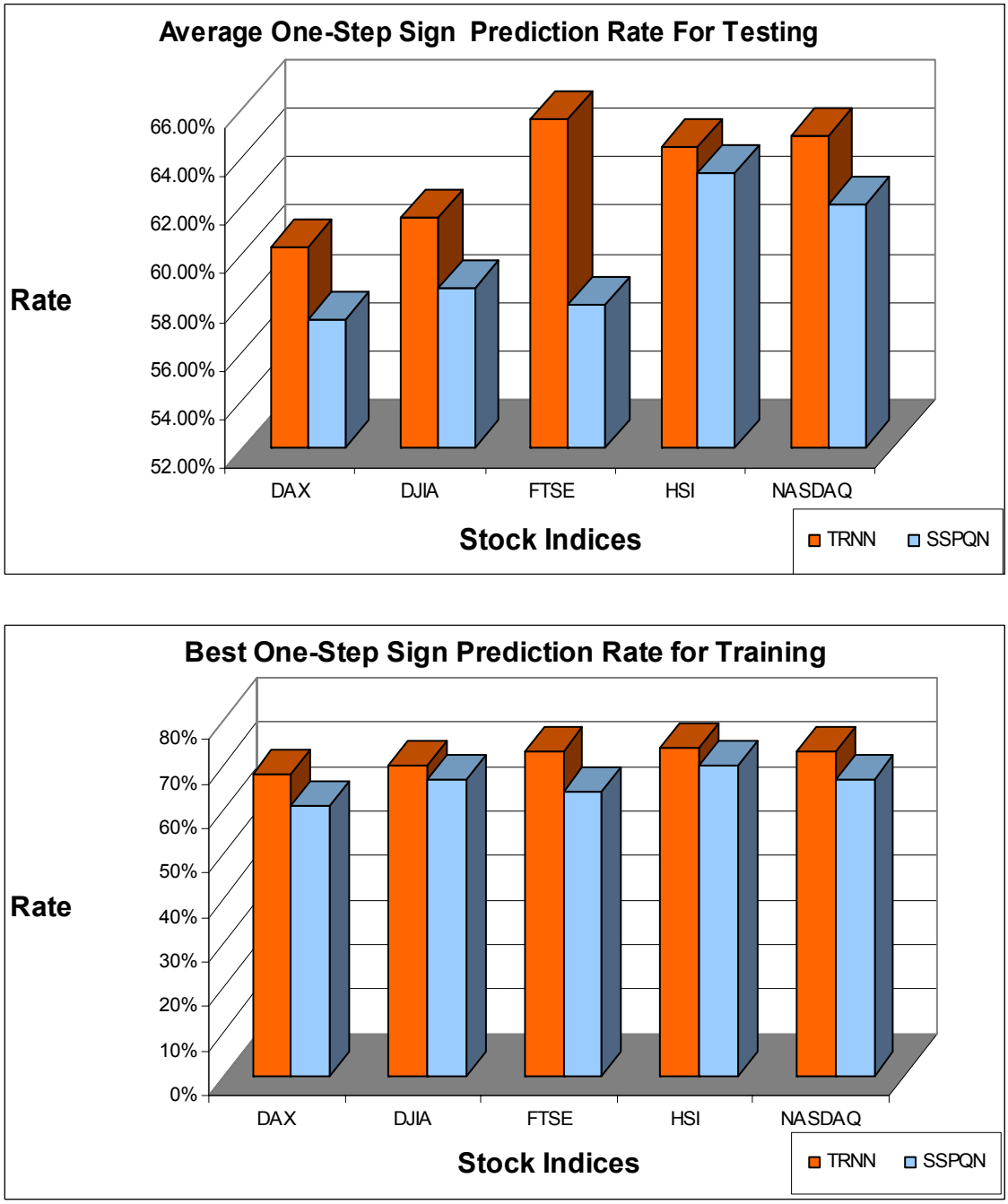


Figure 30 Performance Comparison based on Average and Best Prediction Results between Two Models

Figure 30 compares the average and the best index price increments prediction results obtained by the Trust Region Neural Networks and the SSPQN neural networks in five stock markets. In all the stock markets presented, the Trust Region Neural Network model outperforms the SSPQN

neural network model. On the other hand, the MSE results obtained by the TRNN model also obviously outperform than those of SSPQN model both in training and testing process. These outcomes strongly demonstrate that the neural networks training by Trust Region Dogleg Path Algorithms have better convergence capacity than neural networks training by line search based optimization methods, especially in solving complex and high nonlinear optimization problems in real financial forecasting applications.

There seems to be a scarcity of research works on predictions of the indices of DJIA, NASDAQ, FTSE, DAX and HSI while comparable many such works can be found for S & P 500. Some researchers have evaluated their works based on results of directional symmetry (DS). We quoted some of these works here for comparison purposes. They include: Azoff (1994), Dorsey and Sexton (1998) [15], Landasse et al. (2000) [31] and Phua and Ming (2000) [39]. Table 13 summarizes DS results obtained by various network models. Based on DS results, Table 13 shows that our proposed network model outperforms all the other network models considered here. In fact, the best DS obtained by our model even reach the rate as high as 74%, in the testing process.

The figures in table 13 are not computed on the same data. There seems to be a scarcity of former research works on predictions of the indices of DJIA, NASDAQ, FTSE, DAX and HSI, and in these forecasting researches even few evaluated their works based on the results of directional symmetry (DS). So, we quoted all the few works here for comparison purposes. For each former research work evaluated in DS results, we only quoted the best DS results

they could obtain by their proposed models. Thus we just compared the best possible DS results that each models could obtain from the applied data. Though the DS results obtained by different models are not computed on the same data, the comparison results still can show that the best prediction accuracy by our proposed model outperforms all the best prediction results by other models. On the other hand, though the comparison results computed on different data cannot demonstrate that our trust region based neural network model always outperforms other models proposed by former researchers, the results still could show that our model can obtain the best prediction accuracy evaluated in DS by far.

Similar Research Works	Average DS for Training	Average DS for Testing	The Best DS for Training	The Best DS for Testing
Our Results	68.85%	65.51%	75.12%	74%
Phua, P K H & D H Ming(2000)	68.50%	65.75%	71.11%	70.00%
Landasse et al (2000)	60.30%	57.20%	×	×
Dorsey & Sexton (1998)	58.68%	53.97%	×	×
Azoff (1994)	56.50%	54.50%	58.50%	56.00%

Table 13 Comparison of Stock Index Direction Prediction by Different Researches

5.2. Performance Analysis on TRNN model

To check the convergence performance of our proposed Trust Region Neural Networks in more detail, the optimal network structure is applied again to perform one-day ahead prediction of DJIA daily returns on another time series: from 1/24/95 to 6/15/95. The actual and predicted values of DJIA daily returns

for this period are shown in figure 31. Figure 32 shows the actual and predicted daily prices of DJIA. These figures show that our prediction results by trust region neural network model agree with actual value of DJIA impressively. Furthermore, our predicted results show that the common problem of the laziness of neural networks has been overcome, see figure 32 for instance.

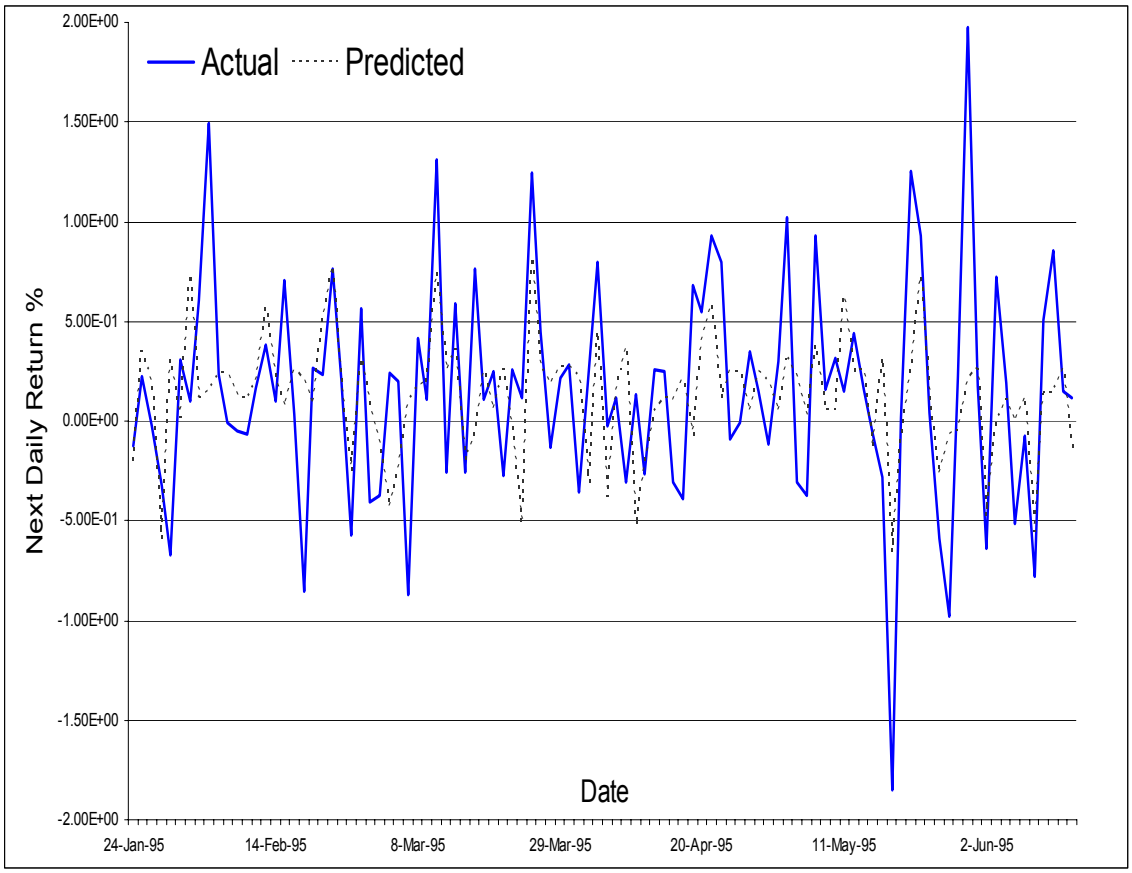


Figure 31, Actual and Predicted Daily Returns of DJIA

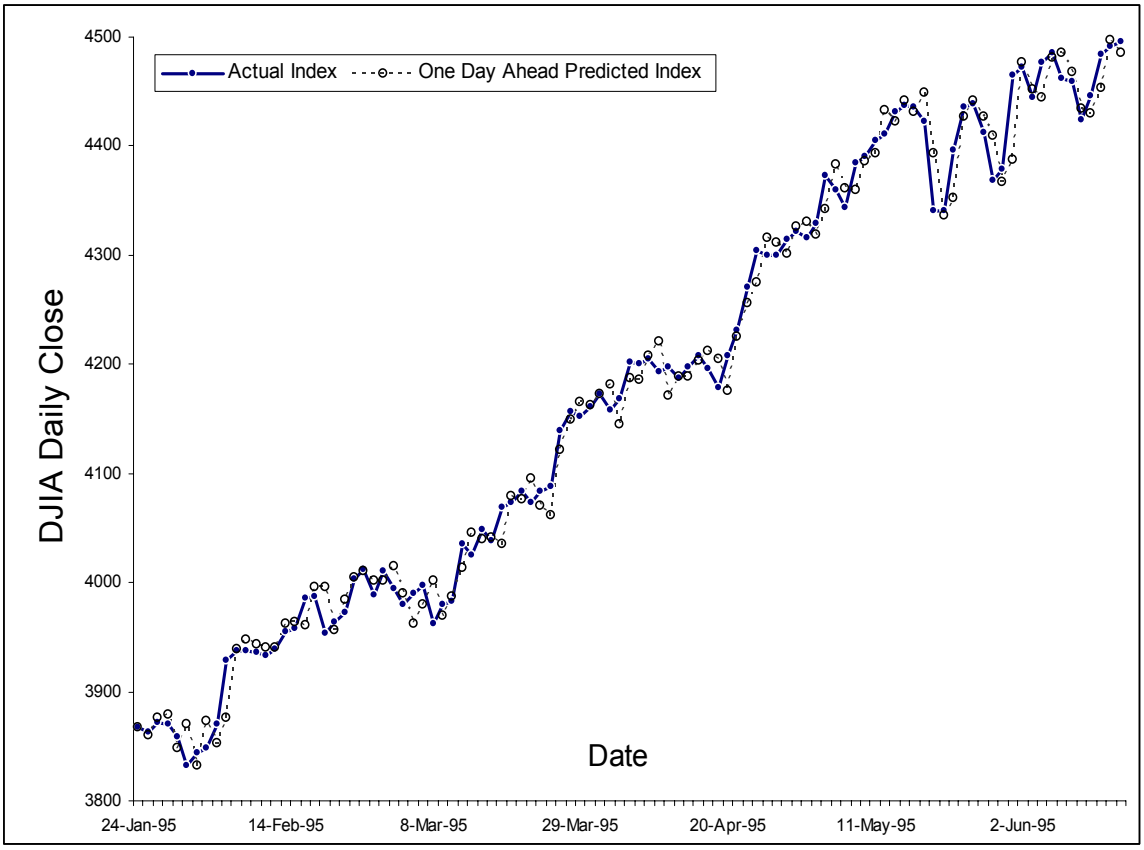


Figure 32, Actual and Predicted value of DJIA daily close prices

Chapter 6

Conclusions and Discussions

This thesis presents a comprehensive study of applying artificial neural networks in predicting stock index increments. The data of five major stock exchange indices, DAX, DJIA, FTSE-100, HSI and NASDAQ, are applied to test our network models. Unlike other financial forecasting models, our model directly uses the component stocks of the index as inputs for the prediction. For the neural network training, a trust region dogleg path algorithm is applied. For comparison purposes, other neural network training algorithms are also considered; in particular, optimization techniques with line searches are applied for solving the same class of problems. Optimal neural network topologies are determined for each model by experiments. Computational results from five different stock markets show that the trust region based neural network model obtained better results compared with the results obtained by other neural network models. In particular, we show that our model is able to forecast the sign of the index increments with an average success rate above 60% in all the five stock markets. Furthermore, our prediction results for FTSE-100, HSI and NASDAQ are exceeding an average accuracy of 64%.

Besides the issue on learning algorithms, a major challenge faced by neural network researchers is that there are no formal theories for determining the optimal network model. Neural network modelling is a complex process that is

currently more art than science. Thus for an artificial neural network applied to a specific problem, experiments must be conducted to determine the performance differences between alternative models. In our research, three major factors that have influence on the network performance are considered in the process of neural network modelling. Both the individual and interactive sensitivity analysises are conducted in order to study how these major factors as well as their interrelationships influence the neural network performance, especially in financial forecasting. Based on the analysis results, the following general guidelines on network modelling in financial forecasting are suggested:

(1). Sample size in the training set affects the network prediction accuracy. Our result shows that it's not the larger sample size in training set the better the network performance. The optimal sample size for training set should be determined by experiments. On the other hand, the sensitivity of the network prediction accuracy on sample size depends on the noise within the data. The noisier the data the more sensitive the network performance is on the sample size. In this thesis, we exams the effects of different in-sample time periods and sample size on the network performance. We considered five different sample sizes of 600, 800, 1000, 1500 and 2000 in the experiments and the results show that it's not the larger the training sample size the better the prediction results. In fact, our experiments show that the optimal sample sizes for prediction are the latest 600 or 800 sample data in most cases. After determining the optimal training samples for each model, we make the final prediction. Though some data used in this thesis is as old as dated

in 1994, they are just used to determine the optimal training sample and to exam the effects of training sample on network performance. In fact, none of these very old data are used in the real financial forecasting for the period of May 2002 to September 2002. The sample data in the most recently two or three years are finally used for the financial forecasting for the 100 days in 2002. Our experiments also show that the recent data is better than old data in making financial forecasting.

(2). Although numerous heuristics have been suggested for determining the number of nodes in the hidden layer, they do not apply across all the reported studies. Our result shows that in financial forecasting, the less the number of nodes in the hidden layer the better the network prediction accuracy. The main reason for this result is that the financial time series data are highly noisy and highly nonlinear, thus smaller networks should be used to increase generalization ability and avoid overfitting to the noise.

(3). Under the component-based index forecasting method, the number of component stocks that should be used as inputs for the networks also affects the network prediction accuracy. There is no clear guidance on the selection of the inputs number under the component-based forecasting scheme. Determining the number of inputs nodes is still part of the 'art' of neural networks.

(4). Our research shows that there exist interrelationships between these major factors in network modelling and such interrelationships also affect the network performance. For such impact could be obvious in some cases, we must consider both the direct and indirect impacts of each particular factor on the network prediction accuracy in the process of network modelling. Our results show that both of the network architecture factors (input and hidden nodes) have obvious influence on the impact of sample size on network performance and hidden nodes number also has above average influence on the impact of input nodes on network performance.

(5). Under different evaluation criteria for the network performance, the effects of each factor as well as the interrelationships between these major factors on the network performance are usually different. Thus it's no strange to find that the optimal network topologies for the same particular time series data under different evaluation criteria are usually different. Under MSE criteria, the network architecture factors normally have more impact on the network performance than the sample size does and the interrelationships between all these factors are usually low on average, while all these relationships will reverse under the DS criteria.

Particularly, as the experimental results in this thesis are valid only for prediction of stock returns for the 100 days in 2002, if one is to use the methods presented in the thesis to predict stock returns in 2003, our

recommendations on determining the network training algorithms, training samples, network topology and number of iterations would be as following:

- Though the results are only valid for prediction of stock returns for the 100 days in 2002, computational results obtained from five different stock markets demonstrate that the trust region based algorithms always outperform the line search based SSPQN algorithms in the aspect of prediction accuracy on the one-step sign prediction rate. And our experiments conducted on five different financial time series data show that the average one-step sign prediction rates by trust region algorithm based network model are higher than 60% in all markets, which is statistically significant. The main difference between the predictions in 2002 or in 2003 is the different time series data being used for the prediction. As we have shown that for different financial time series data in different stock markets, the trust region algorithms based model always outperform the other model, we would strongly recommend the trust region based network training algorithms presented in the thesis if someone is to use the component-based neural network model for the stock index increments forecasting in 2003.
- Sample size in the training set affects the network prediction accuracy. Our experiment results for 2002 prediction have shown that it's not the larger sample size in training set the better the network performance. The optimal sample size for training set should be determined by experiments. On the other hand, the sensitivity of the network prediction accuracy on sample size depends on the noise within the data. The

noisier the data the more sensitive the network performance is on the sample size. Particularly, our experiments show that for DAX index forecasting the optimal sample size is always 600 for both models under two criteria of MSE and DS. For DJIA index forecasting, the optimal sample size is always 800. In the case of FTSE forecasting, the optimal sample size is different when different models are used: 800 for TRNN model and 600 for SSPQN model. While in the cases of HSI and NASDAQ, the difference between optimal sample sizes for different criteria or different models becomes very obviously when comparing with the former three cases: the optimal sample size is 1500 for HSI forecasting when TRNN model is used and is 600 when SSPQN model is used; the optimal sample size is 600 for NASDAQ index forecasting under MSE criterion while be 1000 under DS criterion. As we have shown in Chapter one that the data in HSI and NASDAQ indices are obviously more noisy than the data in DAX, DJIA and FTSE indices. That's why the optimal samples sizes for HSI and NASDAQ indices forecasting show obvious diversity for different models or different criteria. For 2003 stock index forecasting, we would recommend 600 data samples for DAX index forecasting and 800 data samples for DJIA index forecasting. For FTSE forecasting, we would recommend 600 or 800 data samples depending on different models. But for HSI and NASDAQ indices forecasting, we strongly recommend practitioners to determine the optimal sample size by experiments. For noisy data the optimal sample size is affected by many factors thus the optimal sample size would be inconsistent for different cases.

- Although numerous heuristics in former researches have been suggested for determining the number of nodes in the hidden layer, they do not apply across all the reported studies. Our results in 2002 prediction shows that in financial forecasting, the less the hidden neurons the better the prediction accuracy in MSE. The main reason for this result is that the financial time serious data are highly noisy and highly nonlinear, thus smaller networks should be used to increase generalization ability and avoid overfitting to the noise. If someone would predict the stock returns in 2003, we will recommend them not to use too many nodes in the hidden layer, for too many hidden nodes will produce a network that memorizes the input data and lacks the ability to generalize. By our experiments, around 5 hidden nodes will lead to the best results in MSE. While under DS criterion, the optimal number of hidden nodes is unconstant for different stock markets. The possible explanation for such difference for different criteria maybe that MSE is the objective function for the network training algorithm while DS is not the objective function and just reflects the percentage of correctly predicted directions with respect to the stock index. Our experiments show that larger architectures are normally required for complex response surfaces, thus optimal hidden nodes under DS criterion don't always follow the rule of " the less the hidden nodes the better the prediction accuracy in DS". If researchers were to predict the stock returns under DS criterion in 2003, we would recommend them to determine the optimal hidden nodes by experiments.

- Under our proposed component-based index forecasting method, the number of component stocks that should be used as inputs for the networks also affects the network prediction accuracy. As the optimal inputs are sample data based under both criteria of MSE and DS, there is no clear guidance on the selection of the inputs under the component-based forecasting scheme. Determining the number of input nodes is basically problem-dependent and requires an experimental trial-and-error approach.
- Our experiments show that it's not the more iterations the better the prediction results. Though training algorithms guarantee that total error in the training set will continue to decrease as the number of iteration increases, training with repeated applications of the same data set may result in the phenomenon of overtraining. Overtraining occurs when the neural network attempts to exactly fit the limited set of points and loses its ability to interpolate between those points (Hecht-Nielsen 1990). In practice, our experiments show that the MSE (or DS) results for testing will reverse its trends to decreasing to increasing (or increasing to decreasing) at some particular iteration number in the training process. In theory, as training processes, there is always an intermediate stage at which the algorithm reaches a good balance between accurately fitting the training set examples and still providing a reasonable good interpolation capability. The problem created from overtraining is determining when sufficient iterations have been accomplished to achieve the desired prediction accuracy. The "best" predictive performance should be obtained with the set of weights that produces

the minimum value for the error function in the testing set of data. Iterations beyond that point will not improve predictive performance. Thus the training process of our network should be terminated when the MSE (or DS) results for testing reverse its trends. The internal architecture (nodes in hidden and input layers) of network will determine the number of connection weights of the neural network model, thus finally determine the degrees of freedom of the network (the variables and terms of the model). On the other hand, the sample data size will determine the points to be fitted by the neural network model. Thus, all these variables of training samples and network topology will determine the optimal iteration number for network training. In our prediction experiments in 2002, we obtained the optimal iteration number for some particular stock market and training algorithm by averaging all the possible results from the combination of the three variables of training samples (600, 800, 1000, 1500, 2000), inputs (5, 10, 15) and hidden nodes (5, 10, 15, 20). The average results of optimal iteration number for network training under two criteria are listed in the Table 3 of the thesis. All the computations and comparisons between the two models in the thesis are all based on the averaged optimal iteration numbers listed in Table 3. Though researchers could take the average results in table 3 as reference if they are to predict the stock indices in 2003, we would recommend them to re-determine the optimal iteration number under the new training sample data by the method we proposed in this thesis. In my opinion, the difference sometimes may be large, because

the variable of training sample is a key element for determining the optimal iteration number for training, which could be reflected in Table 3.

Our research conducted detailed sensitivity analysis on several design factors that significantly impact the accuracy of neural network forecasts and the proposed optimal network topology is determined by such analysis. Furthermore, a trust region dogleg path algorithm is applied to train the proposed neural network model and this TRNN model has been shown to give an impressive result in financial forecasting. Though the forecasting accuracy values by TRNN model have been statistically significant, more researches could be conducted in the following areas in order to improve the network performance even further:

(1). The component-based input selection method in this paper is mainly based on the correlation coefficient between the **returns** of the index and the component stock prices. The $\underline{m}$ component stocks that highest correlated with the corresponding index in addition with the index itself are selected as the inputs ($\underline{m}+1$) for the network forecasting. Several other input selection schemes should also be considered: (a). Based on the correlation coefficient between the **returns** of the index and the component stocks, only the $\underline{m}$ component stocks that highest correlated with the corresponding index are selected as the inputs ($\underline{m}$); (b). Based on the correlation coefficient between the **prices** of the index and the component stocks, the $\underline{m}$ component stocks that highest correlated with

the corresponding index in addition with the index itself are selected as the inputs ($\underline{m}+1$); (c). Based on the correlation coefficient between the **prices** of the index and the component stocks, only the $\underline{m}$ component stocks that highest correlated with the corresponding index are selected as the inputs ($\underline{m}$). Experiments must be conducted to determine the network performance differences between alternative inputs selection schemes.

(2). Artificial neural network training usually requires two main sets of data: the training set which must be representative of the entire domain and the test set which is used to evaluate the prediction accuracy of the model. There are many alternative ways of dividing the whole time series data into the two parts. For example, 90% whole data could be used as the training set and the remaining 10% were used as the test set. Further researches should be conducted to see whether alternative sample data dividing method could improve the network prediction accuracy. How do we choose an appropriate sample dividing method especially for ANN financial forecasting is an interesting issue for further research.

(3). Besides one-step-ahead forecasting, multi-step-ahead forecasting should also be considered for further research in order to see whether neural networks could also produce impressive better results than

traditional statistical methods, as well as whether the TRNN model still could significantly outperform than other neural network models.

Bibliography

[1] A. Abhyankar, L.S. Copeland and W. Wang, “Uncovering Nonlinear Structure in Real Time Stock Market Indexes: The S&P 500, the DAX, The Nikkei 225, and the FTSE -100”, Journal of Business & Economic Statistics, Vol. 15, no. 1, pp.1-14, 1997.

[2] A.Lapedes and R. Farber, " Nonlinear signal processing using neural networks", Proceedings of the IEEE Conference on Neural Information Processing System - Natural and Synthetic, 1987.

[3] A.N. Refenes, A. Zapranis and G. Francies, "Stock performance modelling using neural networks: a comparative study with regression models", Neural Network, Vol. 5, pp. 961-970, 1994.

[4] Armstrong W, Dwelly A, Liang J, Lin D, Reynolds S.,"Learning and generalization in adaptive logic networks", Artificial Neural Networks, Proceedings of the 1991 International Conference on Artificial Neural Networks, 1173-1176, 1991.

[5] B. Freisleben, "Stock market prediction with back propagation networks" Proceedings of the 5th International Conference on Industrial and Engineering Application of Artificial Intelligence and Expert System, pp 451-460, June 1992.

[6] B.H.Solnik, "Note on the Validity of the random walk for European stock prices", Journal of Finance, December 1973.

[7] Baily D. and Thompson D. M., "Developing neural network application", AI Expert, pp. 33-41, Sept. 1990.

[8] Ballo M. G.,"Enhanced training algorithm and integrated training / architecture selection for multilayer perception networks" IEEE Transactions on Neural Networks, 3, 864-875, 1992.

[9] Berry M. J. A. and Linoff, G., Data Mining Techniques, New York: John and Wiley and Sons, 1997.

[10] C. G. Broyden, “Convergence of A Class of Double Rank Minimization Algorithms”, Journal of the Institute of Mathematics and Its Applications, vol. 6, pp. 76-90, 1970.

[11] Caldwell,R.,"Performance Metric for Neural Network-based Trading System Development", NeuroVest Journal, Vol. 3Num 2, pp. 13-23.

[12] Charitou A, Charalambous C., "The prediction of earnings using financial statement information: empirical evidence with Logit models and artificial neural networks.", International Journal of Intelligent Systems in Accounting, Finance and Management, 5: 199-215, 1996.

[13] D. F. Shanno, “Conditioning of Quasi -Newton Methods for Function Minimization”, Mathematics and Computation, vol. 24, pp. 647-656, 1970.

[14] D. Goldfarb, “A Family of Variable Metric Methods Derived by Variational Means”, Mathematics and Computation, vol. 24, pp. 23-26, 1970.

[15] Dorsey R. and Sexton R., "The Use of Parsimonious Neural Networks for Forecasting Financial Time Series", Journal of Computational Intelligence in Finance, vol.6, pp. 24-31, January/February 1998.

[16] Fahlman SE, Lebiere C.,"The Cascade-Correlation Learning Architecture", Technical Report: CMU-CS-90-100, Carnegie-Mellon University, 1990.

[17] Filippo Castiglione, "Forecasting Price Increments Using An Artificial Neural Network", Advances in Complex Systems, Vol. 4, No. 1, pp. 45-56, 2001.

[18] G.A.Shultz, R.B.Schnabel and R.H.Byrd, "A family of trust-region-based algorithms for unconstrained minimization with strong global convergence properties", SIAM Journal on Numerical Analysis 22 (1985) 47-67.

[19] G.E. PBox and G.M. Jenkins, Time Series analysis: forecasting and control, Holdenday, San Francisco (1976).

[20] Hertz J, Krogh A, Palmer RG. 1991. Introduction to the Theory of Neural Computation. Addison-Wesley: Reading, MA.

[21] J. O. Katz, “Developing Neural Network Forecasters for Trading”, Technical Analysis of Stock and Commodities, pp. 58-90, April 1992.

[22] J.P.Bulteau and J.Ph.Vial, "A restricted trust region algorithm for unconstrained optimization", Journal of Optimization Theory and Applications 47 (1985), 413-434.

[23] J.T. Yao and H.L. Poh, "Equity forecasting: a case study on the KLSE index", Neural Networks in Financial Engineering, Proceedings of the 3rd International Conference on Neural Networks in the Capital Markets, Oct 1995. A.P N. Refenes, Y. Abu-Mostafa, J.

[24] J.T. Yao, C.L. Tan and H.L. Poh,"Neural Networks for Technical Analysis: A Study on KLCI", International Journal of Theoretical and Applied Finance, Vol. 2, No. 2, pp. 221-241, 1999.

[25] James R. Coakley and Carol E. Brown, " Artificial Neural Networks in Accounting and Finance: Modeling Issues" International Journal of Intelligent Systems in Accounting, Finance & Management, Vol. 9, pp. 119-144, 2000.

[26] Jhee WC, Lee JK. 1993. "Performance of neural networks in managerial forecasting", Intelligent Systems in Accounting, Finance and Management, 2: No. 1, 55-71.

[27] Jianzhong Zhang and Chengxian Xu, "A Class of Indefinite Dogleg Path Methods for Unconstrained Minimization", SIAM Journal of Optimization, Vol. 9, No. 3, pp 646-667, 1999.

[28] Jianzhong Zhang and Chengxian Xu, "Trust region dogleg path algorithms for unconstrained minimization", Annals of Operations Research, vol. 87, pp. 407-418, 1999.

[29] Jingtao Yao and Chew Lim Tan, “Time Dependent Directional Profit Model for Financial Time Series Forecasting”, Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, 2000, vol. 5, pp. 291-296, 2000.

[30] Katz J. O., "Development neural network forecasters for trading", Technical Analysis of Stocks and Commodities, pp. 58-90, April 1992.

[31] Lendasse A., De Bodt E., Wertz V. and Verleysen M., "Non-linear financial time series forecasting-Application to the Bel 20 stock market

index", European Journal of Economics and Social Systems, vol. 14, No.1, pp.81-91, 2000.

[32] Leonard J. A. and M. A. Kramer, "Improvement of the back propagation algorithm for training neural networks", Computers Chem. Engineering. 14,337-341,1990.

[33] M. C. Biggs, “A Note on Minimization Algorithms Which Make Use of Non-Quadratic Properties of the Objective Function”, Journal of the Institute of Mathematics and its Applications, vol. 7, pp. 337-338, 1973.

[34] M. Jorgensen, Experience with the accuracy of software maintenance task effort prediction models, IEEE Trans. Software Engineering, 21(8), Page 674-681, 1995.

[35] Marques L, Hill T, Worthley R, Remus W. 1991, "Neural network models as an alternative to regression", Proceedings of the 24th Annual Hawaii International Conference on Systems Sciences, Vol. IV. 129-146.

[36] Masters, T., "Practical Neural Networks in C++", Academic Press, New York, 1993.

[37] Micheal Y.Hu, Peter Zhang, Christine X. Jiang and B. Eddy Patuwo, “A Cross –Validation Analysis of Neural Network Out-of-Sample Performance in Exchange Rate Forecasting”, Decision Sciences, Vol. 30, no. 1, pp.197-216, 1999.

[38] P. K. H. Phua, “Multi-Directional Parallel Algorithms for Unconstrained Optimization”, Optimization, vol. 38, pp. 107-125, 1996.

[39] P.K.H. Phua and D. Ming, “Parallel Nonlinear Optimization Techniques for Training Neural Networks”, Accepted for publication in the IEEE Transactions on Neural Networks.

[40] Peter Tino, C. Schittenkopf and G. Dorffner, "Financial Volatility Trading Using Recurrent Neural Networks", IEEE Transactions on Neural Networks, Vol. 12, No.4, 2001, July.

[41] R. Fletcher, “A New Approach to Variable Metric Algorithms”, Computer Journal, vol. 13, pp. 317-322, 1970.

[42] R.H.Byrd, R.B.Schnable and G.A.Shultz, "Approximate solution of the trust region problem by minimization over two-dimensional subspaces", Mathematical Programming 40 (1988) 247-263.

[43] Randall B. Caldwell, “Three Methods of Neural Network Sensitivity Analysis for Input Variable Reduction: A Case Study in Forecasting the S&P 500 Index”, NeuroVest Journal, pp.17-22, 1996.

[44] Renate Sitte and Joaquin Sitte, "Analysis of the Predictive Ability of Time Delay Neural Networks Applied to the S&P 500 Time Series”, IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Review, Vol. 30, No. 4, 2000, November.

[45] Resta, M., “Towards an Artificial Technical Analysis of Financial Markets”, Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks 2000, Vol. 5, pp. 117 -122, 2000.

[46] Rumelhart DE, Hinton GE, Williams RJ. 1986. Learning representations by back-propagation errors. Nature 323: 533-536.

[47] S. S. Oren and D. G. Luenberger, “ Self-scaling Variable Metric (SSVM) Algorithms, Part I: Criteria and Sufficient Conditions For Scaling a Class of Algorithms”, Management Science, vol. 20, pp. 845-862, 1974.

[48] S.Taylor, "Modeling financial time series", John Wiley & Sons 1986.

[49] Steven H. Kim and Se Hak Chun, “Graded forecasting using an array of bipolar predictions: application of probabilistic neural networks to a stock market index”, International Journal of Forecasting, vol. 14, pp. 323-337, 1998.

[50] Steven Walczak, "An Empirical Analysis of Data Requirements for Financial Forecasting with Neural Networks", Journal of Management Information Systems, Vol. 17, No. 4, pp. 203-222, 2001, spring.

[51] Subramanian V, Ming SH, Hu MY. 1993 " An experimental evaluation of neural networks for classification", Computers and Operations Research 20: No. 7, 769-782.

[52] Tim Hill et al, Neural Network Models for Time Series Forecasts, Management Science, Vol. 42, Issue 7, Page 1082-1092.

[53] Wong FS.,"A 3D neural network for business forecasting", proceedings of the 24th Annual Hawaii International Conference on System Sciences, Vol. IV, 113-123, 1991.

[54] Yong Liu and Xin Yao, “Evolving neural networks for Hang Seng stock index forecast”, Proceedings of the 2001 Congress on Evolutionary Computation, vol.1, pp.256-260, 2001.

[55] Yves Bentz, A.P. N. Refenes and A. Neil Burgess, " Neural Network in Financial Engineering: A Study in Methodology" IEEE Transactions on Neural Network, Vol. 8, No. 6, 1997, November.

APPENDIX A

Distributions of Journals Publishing NNs Applications in Finance

JOURNAL	NUMBER OF ARTICLES
Journal of Management Science	11
European Journal of Operational Research	10
Decision Support System	9
IEEE Transactions on Neural Networks	6
Computers and Operations Research	5
Journal of Forecasting	5
Journal of AI Expert	4
Journal of Decision Science	4
Intelligent Systems for Finance and Business	3
Journal of Business and Economics Statistics	3
Journal of Expert System	3
Journal of Financial Analyst	3
Journal of Future Markets	3
Journal of Management Information System	3
Operational Research Society	3
Computers and Industrial Engineering	2
IEEE Expert (Intelligent Systems & Their Application)	2
IEEE Transactions on Systems, Man and Cybernetics	2
Journal of Applied Business Research	2
Journal of Business forecasting	2
Journal of Finance	2
Journal of Futures	2
Journal of Operational Research Society	2
Journal of Risk and Insurance	2
A Review of Financial Studies	1
Advances in Complex Systems	1
America Business Review	1
Application of Artificial Intelligence	1
Applied Financial Economics	1
IEEE Computational Science and Engineering	1
IEEE Transactions on Evolutionary Computation	1
Journal of Accounting, Auditing and Finance	1
Journal of America Statistics Association	1
Journal of Applied Economics	1
Journal of Banking and Finance	1
Journal of Financial and Quantitative Analysis	1
Journal of Financial Management	1
Journal of Financial Markets	1
Journal of Fixed Income	1

Journal of Information and Management	**1**
Journal of Interface	**1**
Journal of Managerial Finance	**1**
Journal of PC AI	**1**
Journal of Portfolio Management	**1**
Journal of Real Estate Appraiser	**1**
Journal of Real Estate Research	**1**
Journal of Transport Economics and Policy	**1**
Marketing Research	**1**
Neural Computer and Applications	**1**
Property Tax Journal	**1**
Quarterly J. of Business and Economics	**1**
Studies in Economics and Finance	**1**
The Executive's Journal	**1**
Theoretical and Applied Finance	**1**
TOTAL NUMBER	**123**

APPENDIX B

Distributions of Conference Proceedings on NNs Applications in Finance

NAME OF THE INTERNATIONAL CONFERENCE ON NEURAL NETWORKS	NUMBER OF ARTICLES	YEAR HELD
Neural Networks, 1999. IJCNN '99. International Joint Conference on	12	1999
Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on	12	1994
Neural Networks, 1995. Proceedings., IEEE International Conference on	9	1995
Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on	8	2000
Neural Networks, 2001. Proceedings. IJCNN '01. International Joint Conference on	7	2001
Neural Networks, 1991. 1991 IEEE International Joint Conference on	7	1991
Neural Networks, 1997., International Conference on	5	1997
Neural Networks, 1990., 1990 IJCNN International Joint Conference on	5	1990
Computational Intelligence for Financial Engineering, 1996., Proceedings of the IEEE/IAFE 1996 Conference on	4	1996
Neural Networks, 1996., IEEE International Conference on	4	1996
Artificial Neural Networks and Expert Systems, 1993. Proceedings., First New Zealand International Two-Stream Conference on	4	1993
Neural Networks, 1993. IJCNN '93-Nagoya. Proceedings of 1993 International Joint Conference on	4	1993
Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on	4	1991
Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on	3	1998
Neural Networks, 1988., IEEE International Conference on	3	1988
Computational Intelligence and Multimedia Applications, 1999. ICCIMA'99. Proceedings. Third International Conference on	2	1999
Intelligent Processing Systems, 1997. ICISP'97. 1997 IEEE International Conference on	2	1997
Computational Intelligence for Financial Engineering, 1995., Proceedings of the IEEE/IAFE 1995 Conference on	2	1995
Information Systems: Decision Support and Knowledge-Based Systems, 1994., Proceedings of the Twenty-Seventh	2	1994

Hawaii International Conference on		
System Sciences, 1991., Proceedings of the Twenty-Fourth Hawaii International Conference on	**2**	**1991**
Evolutionary Computation, 2001. Proceedings of the 2001 Congress on	**1**	**2001**
IFSA World Congress and 20th NAFIPS International Conference 2001.	**1**	**2001**
Adaptive Systems for Signal Processing, Communications and Control Symposium 2000. AS-SPCC. The IEEE 2000.	**1**	**2000**
Computational Intelligence for Financial Engineering, 2000. Proceedings of the IEEE/IAFE/INFORM 2000 Conference on	**1**	**2000**
Neural Networks, 2000. Proceedings. Sixth Brazilian Symposium on	**1**	**2000**
Systems, Man and Cybernetics, 2000 IEEE International Conference on	**1**	**2000**
Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on	**1**	**1999**
NAFIPS'97., 1997 Annual Meeting of the North America	**1**	**1997**
Neural Networks for Industrial Applications, IEEE Colloquium on, 1997.	**1**	**1997**
Neural Networks, 1997. Proceedings., IVth Brazilian Symposium on	**1**	**1997**
Fuzzy Systems Symposium, 1996. Soft Computing in Intelligent Systems and Information Processing., Proceedings of the 1996 Asian	**1**	**1996**
Neural-Fuzzy Systems, 1996. AT'96., International Symposium on	**1**	**1996**
System Sciences, 1996., Proceedings of the Twenty-Ninth Hawaii International Conference on	**1**	**1996**
Systems, Man and Cybernetics, 1996., IEEE International Conference on	**1**	**1996**
Artificial Neural Networks and Expert Systems, 1995. Proceedings., Second New Zealand International Two-Stream Conference on	**1**	**1995**
Military Communications Conference, 1995. MILCOM'95, Conference Record, IEEE	**1**	**1995**
Artificial Intelligence for Applications, 1994., Proceedings of the Tenth Conference on	**1**	**1994**
Artificial Neural Networks, 1993., Third International Conference on	**1**	**1993**
Neural Networks, 1993., IEEE International Conference on	**1**	**1993**
Artificial Intelligence on Wall Street, 1991. Proceedings, First International Conference on	**1**	**1991**
TOTAL NUMBER OF THE ARTICLES		**121**

APPENDIX C

A Brief Introduction of Trust Region Dogleg Path (TRDP) Algorithm

In this appendix, we give a brief introduction to the Trust Region Dogleg Path (TRDP) algorithm proposed by [27, 28] for solving unconstrained minimization problems. Consider the solution of the problem

$$\min\left\{q_k(\delta) \overset{def}{=} f_k + g^{(k)^T}\delta + \frac{1}{2}\delta^T B_k \delta \,\middle|\, \|\delta\|_2 \le \Delta_k\right\} \tag{I.1}$$

in trust region methods for minimizing a smooth function $f(x)$, $x \in R^n$, where $f_k = f(x^{(k)})$, $g^{(k)} = \nabla f(x^{(k)})$, $\delta_k = x - x^{(k)}$, B_k is either $\nabla^2 f(x^{(k)})$ or its approximation and Δ_k is the trust region radius. The solution of problem (I.1) generally satisfies the system

$$(B_k + \mu I)\delta(\mu) = -g^{(k)}, \left\|\delta^{(k)}\right\|_2 = \Delta^{(k)},$$

where $\mu \ge 0$ such that $B_k + \mu I$ is at least positive semi-definite, except that if B_k is positive definite and $\left\|B_k^{-1} g^{(k)}\right\|_2 \le \Delta_k$, the solution is $\delta^{(k)} = -B_k^{-1} g^{(k)}$. However, there is no definite method to determine such a μ. Most algorithms find an approximate solution of (I.1). Shultz et al. [18, 42] proposed an approximate solution of (I.1) by performing a two-dimensional quadratic minimization:

$$\min\left\{q_k(\delta) \,\middle|\, \delta \in \varsigma, \|\delta\|_2 \le \Delta_k\right\} \tag{I.2}$$

where ς is a two dimensional subspace.

Let Δ_k vary, the solution points of (I.1) form a curvilinear path in the 2-dimensional space, called the optimal path which minimizes $q_k(k)$ within the trust region. Most practical two-dimensional curvilinear paths work well when B_k is positive definite, but they are unable to deal with the non-positive definite case. To improve this situation, TRDP algorithm proposes some indefinite single dogleg paths for the solution of (I.1). These paths are obtained by considering negative curvature directions for indefinite B_k . Bunch-Parlett factorization of a symmetric matrix is employed to factorize B_k

$$PB_kP^T = LDL^T \tag{I.3}$$

where *P* is a permutation matrix, *L* a unit lower triangular matrix and *D* a block diagonal matrix with 1×1 and 2×2 diagonal blocks. If B_k is positive definite, *D* is diagonal. Without loss of generality, it is assumed in the sequel that $P = I$. It is known from [22] that the elements of *L* are bounded with bounds independent of the matrix B_k , i.e. there exist positive constants c_1, c_2, c_3 and c_4 such that $c_1 \le \|L\|_2 \le c_2, c_3 \le \|L^{-1}\|_2 \le c_4$. Positive definiteness of B_k is implied from that of *D*, whose eigenvalues are easy to calculate and Newton directions are generated.

In the case, B_k is indefinite, then the most negative eigenvalues μ_1 and d_1 of B_k and *D* satisfy relations

$$d_1 \|L\|_2^2 \le \mu_1 \le d_1 / \|L^{-1}\|_2^2 \tag{I.4}$$

Let $d_1 \le d_2 \le \cdots \le d_n$ be eigenvalues of *D* and $u^1, u^2, ..., u^n$ be the corresponding orthonormal eigenvectors. Let $\aleph^- = \{i \,|\, d_i \le 0\}$. The direction

$$d \overset{def}{=} -\mathrm{sgn}(g^{(k)^T} L^{-T} \upsilon) L^{-T} \upsilon, \upsilon \in \varsigma = \{\upsilon \,|\, \upsilon = \sum_{i \in \aleph^-} \mathrm{l}_i u^i, \forall \mathrm{l}_i \in R\} \quad \text{(I.5)}$$

is a direction of negative curvature of B_k, since $d^T B_k d = \upsilon^T D \upsilon = \sum_{i \in \aleph^-} d_i (\mathrm{l}_i)^2 < 0.$

The model algorithm presented in this section locates a minimizer of a smooth function $f(x)$. At each iteration, the gradient $g^{(k)}$ and the matrix B_k are evaluated. A dogleg path, denoted by $\Gamma^{(k)}$, is formulated from Bunch-Parlett factorization of B_k and the problem

$$\min\{q_k(\delta) = f_k + g^{(k)^T}\delta + \frac{1}{2}\delta^T B_k \delta \,|\, \delta \in \Gamma^{(k)}, \|\delta\|_2 \le \Delta_k\} \qquad \text{(I.6)}$$

is solved to get $\delta^{(k)}$. Then either $x^{(k)} + \delta^{(k)}$ is accepted as a new point or Δ_k is reduced, depending upon a comparison between the actual reduction $ared(\delta^{(k)})$ of the objective function and the predicted reduction $pred(\delta^{(k)})$ by the quadratic model

$$ared(\delta^{(k)}) \overset{def}{=} f_k - f(x^{(k)} + \delta^{(k)}), pred(\delta^{(k)}) \overset{def}{=} -g^{(k)^T}\delta^{(k)} - \frac{1}{2}\delta^{(k)^T} B_k \delta^{(k)}$$

if the reduction in the objective function is satisfactory, we start a new iteration at $x^{(k+1)} = x^{(k)} + \delta^{(k)}$ with the updated trust region radius; otherwise, the iteration continues at point $x^{(k)}$ with a reduced Δ_k. The model of these algorithms is as follows.

Trust Region Dogleg Path (TRDP) Algorithms

1. Given $x^{(0)} \in R^n, \Delta_{mas}$ and $\Delta_0 (< \Delta_{mas})$. Set $0 < \eta_1 < \eta_2 < 1, 0 < \gamma_1 < 1 < \gamma_2$, and $k = 0$

2. Evaluate $f_k = f(x^{(k)})$ and $g^{(k)} = g(x^{(k)})$

3. *Convergence test*. If not determined, generate B_k and form a dogleg path $\Gamma^{(k)}$.

4. Determine $\delta^{(k)} = \arg\min\{q_k(\delta) \big| \delta \in \Gamma^{(k)}, \|\delta\|_2 \le \Delta_k\}$

5. Calculate $\theta_k = ared(\delta^{(k)}) / pred(\delta^{(k)})$. If $\theta_k < \eta_1$, then $\Delta_k = \gamma_1 \Delta_k$. Go to step 4.

6. $x^{(k+1)} = x^{(k)} + \delta^{(k)}$ and $\Delta_{k+1} = \begin{cases} \min\{\gamma_2 \Delta_k, \Delta_{mas}\}, if\ \theta_k \ge \eta_2 and \|\delta_k\|_2 = \Delta_k \\ \Delta_k, otherwise \end{cases}$ set $k \leftarrow k+1$ and go to step 2.

In the following, $[x^{(k)}, y, w] or [x^{(k)}, y, w)$ denote a single dogleg path starting from $x^{(k)}$ and turning direction at *y*. The former is a finite single dogleg path with end point *w*, while the latter is an infinite single dogleg path where the second piece is a ray starting at point *y* along the direction *w*. the single dogleg path $\Gamma^{(k)}$ in step 3 is formulated in the following ways:

(1). If B_k is positive definite, $\Gamma^{(k)}$ is Powell's single dogleg path:

$$\Gamma_{ps}^{(k)} = [x^{(k)}, x_{cp}^{(k)}, x_{np}^{(k)}], x_{cp}^{(k)} = x^{(k)} + \delta_{cp}^{(k)} = x^{(k)} - \beta_k g^{(k)}, x_{np}^{(k)} = x^{(k)} + \delta_{np}^{(k)} = x^{(k)} - B_k^{-1} g^{(k)},$$

where $\beta_k = g^{(k)^T} g^{(k)} / g^{(k)^T} B_k g^{(k)}$.

(2). If B_k is indefinite, we give four choices for $\Gamma^{(k)}$. Let B_k be factorized into the form (I.3) and a negative curvature direction *d* be generated from (I.5) with $\upsilon \in £$.

(a) In the case $g^{(k)^T} B_k g^{(k)} > 0$, if $g^{(k)^T} L^{-T} \upsilon \geq 0$ or

$$g^{(k)^T} L^{-T} \upsilon < 0 \text{ and } \frac{g^{(k)^T} g^{(k)}}{g^{(k)^T} B_k g^{(k)}} < \left| \frac{g^{(k)^T} d}{d^T B_k d} \right|, \tag{I.7}$$

the path is chosen to be $\Gamma_{s1}^{(k)} = [x^{(k)}, x_{cp}^{(k)}, d).$

(b) In the case $g^{(k)^T} B_k g^{(k)} > 0$, $g^{(k)^T} L^{-T} \upsilon < 0$ but the second part of (I.7) dose not hold, or in the case $g^{(k)^T} B_k g^{(k)} \leq 0$, we set $\delta_B^{(k)} = -(B_k + \mu I)^{-1} g^{(k)}, x_B^{(k)} = x^{(k)} + \delta_B^{(k)}$, where

$$\mu \in (\left|\mu_1(B_k)\right| + \omega', \theta_1 \max\{\left|\mu_1(B_k)\right|, \mu_n(B_k)\}), \tag{I.8}$$

$\omega' > 0 and \theta_1 > 1$ is a constant such that it makes the right end of the interval greater than the left end. Notice that for such a choice of μ, $B_k + \mu I$ is positive definite and $\left\| B_k + \mu I \right\|_2 \leq (1 + \theta_1) \left\| B_k \right\|_2$ (I.9) when $g^{(k)^T} B_k g^{(k)} > 0$, if $\left\| \delta_B^{(k)} \right\|_2 \geq \Delta_k and \left\| \delta_B^{(k)} \right\|_2^2 > \delta_B^{(k)^T} \delta_{cp}^{(k)} > \left\| \delta_{cp}^{(k)} \right\|_2^2$ (I.10) the path is $\Gamma_{s2}^{(k)} = [x^{(k)}, x_{cp}^{(k)}, x_B^{(k)}].$; otherwise, the path is $\Gamma_{s3}^{(k)} = [x^{(k)}, x_B^{(k)}, \hat{d}),$ $\hat{d} = \text{sgn}(d^T \delta_B^{(k)}) d.$ When $g^{(k)^T} B_k g^{(k)} \leq 0$,if

$$\frac{g^{(k)^T} B_k g^{(k)}}{g^{(k)^T} g^{(k)}} < \frac{d^T B d}{d^T d}, \tag{I.11}$$

(c) the path is $\Gamma_{s4}^{(k)} = [x^{(k)}, x_B^{(k)}, -g^{(k)})$; otherwise, $\Gamma^{(k)}$ is the path $\Gamma_{s3}^{(k)}$.

APPENDIX D

A Brief Introduction of SSPQN Algorithm

In this appendix, we give a brief introduction to the self-scaling parallel Quasi-Newton (SSPQN) algorithm proposed by [7] for solving unconstrained nonlinear optimization problems. Consider minimizing the following objective function:

$$f(w) = \frac{1}{PK}\sum_{p=1}^{P}\sum_{k=1}^{K}[Y_{pk} - Z_{pk}(w)]^2 \qquad \text{(I.1)}$$

Where $w = w^{(1)} \cup w^{(2)}$ represents the weights of the neural network, $Z_{pk}(w)$ are the output values of the networks, and $\{(X_{pi}, Y_{pi}) : i = 0,1,...,m;\ p = 1,2,...,P\}$ is the set of given input/output vectors for training the neural network.

In solving the above minimization problem, Quasi-Newton methods proceed to generate a sequence of solution points:

$$W_{k+1} = W_k + \alpha_k d_k \qquad \text{(I.2)}$$

Where d_k is the search direction used for iteration *k* and α_k is the step-size of the iteration *k*. The Search direction is computed by:

$$d_k = -H_k g_k \tag{I.3}$$

Where H_k is the current approximation to the inverse Hessian matrix, and $g_k = \nabla f(w_k)$ is the current gradient vector. The matrix H_k is obtained through a recursive process of updating the previous inverse Hessian matrix approximation, and H_0 is generally chosen to be the identity matrix I. In fact, the updating matrix H_{k+1} is chosen to satisfy the so-called Quasi-Newton equation:

$$H_{k+1} y_k = S_k \tag{I.4}$$

Where $y_k = g_{k+1} - g_k$ and $s_k = w_{k+1} - w_k$. To improve the performance of QN methods, we propose to use the following three parameter family of updates (see [14]):

$$H_{k+1}(\phi_k, \theta_k, \lambda_k) = [H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \phi_k (y_k^T H_k y_k) v_k v_k^T]\theta_k + \frac{s_k s_k^T}{\lambda_k s_k^T y_k} \tag{I.5}$$

Where

$$v_k = \frac{s_k}{s_k^T y_k} - \frac{H_k y_k}{y_k^T H_k y_k} \tag{I.6}$$

Here, ϕ_k is the parameter proposed by [15], θ_k is a scaling parameter proposed by [16], and λ_k is the parameter proposed by [17] to improve the performance of BFGS updates proposed independently by Broyden in [15], Fletcher in [19], Goldfarb in [20] and Shanno in [21]. Note that the update formula given in Equation I.5 combines the features and merits of the above three classes of updates. In practice, we note that a particular class of QN methods may be 'good' in solving certain types of optimization problems efficiently, however, their efficiencies may degenerate when they are applied to solve other categories of problems (see [23], for instance).

Based on the above observations, the ideal situation would be that relative merits of different QN methods are adopted into the design and development of new algorithms. This lead us the introduction of the following self-scaling parallel Quasi-Newton (SSPQN) methods.

Self-Scaling Parallel Quasi-Newton (SSPQN) Algorithms

1. Initialization

   Set initial values:

   w_0 = the initial random value of the weights;

   H_0 = the initial Approximation Inverse Hessian, say I;

   ε = the accuracy requirement; 10^{-5} for instance;

   k = the iterations; *k*=0 initially;

2. Compute the function and gradient values

Let $f_k = f(w_k)$ and $g_k = \nabla f(w_k)$

3. Compute the parallel search directions

Let *J* be the number of processors available for computing the search directions simultaneously. Compute in parallel,

$$d_{kj} = -H_k(\phi_{kj}, \theta_{kj}, \lambda_{kj}) g_k \qquad \text{(I.7)}$$

4. Perform the parallel line searches

Along each search direction d_{kj} , perform inexact line searches to determine the step-size in parallel to satisfy the following Wolfe conditions:

$$f(w_k + \alpha_{jk} d_{kj}) \leq f(w_k) + \rho_1 \alpha_{jk} g_{kj}{}^T d_{kj} \qquad \text{(I.8)}$$

and

$$g(w_k + \alpha_{jk} d_{kj})^T d_{kj} \geq \rho_2 g_k^T d_{kj}$$

(I.9)

Where $0 < \rho_1 < 0.5$ and $\rho_1 < \rho_2 < 1$ are some positive small quantities. Points satisfying conditions of equation (I.8) and (I.9) are called 'successful points'.

5. Choose the minimum point

If Successful points are found from more than one search directions, let d_k^* denoted the direction that attained the minimum function value, and α_k^* is the step size. That is

$$f(w_k + \alpha_k^* d_k^*) = \min_{1 \le j \le m} f(w_k + \alpha_{kj} d_{kj}) \qquad \text{(I.10)}$$

6. Generate the new iteration points

Let $w_{k+1} = w_k + \alpha_k^* d_k^*$, $f_{k+1} = f(w_{k+1})$ and $g_{k+1} = \nabla f(w_{k+1})$

7. Test for Convergence

Apply the following convergence criterion:

$$\|g_{k+1}\| \le \varepsilon \bullet \max\{1, \|w_{k+1}\|\} \qquad \text{(I.11)}$$

If condition (1.11) is satisfied, then stop; otherwise proceed to step 8.

8. Compute the approximate inverse Hessian

Compute H_{k+1} according to equation I.5.

9. Repeat the process

Set *k*=*k*+1 and repeat the process from step 3