Université d'Ottawa · University of Ottawa

Université d'Ottawa · University of Ottawa

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Zhuowen WANG

AUTEUR DE LA THÈSE - AUTHOR OF THESIS

Master of Computer Science

GRADE - DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT - FACULTY, SCHOOL, DEPARTMENT

TITRE DE LA THÈSE - TITLE OF THE THESIS

Prediction of Stock Market Prices Using Neural Network Techniques

M.C.E. Yagoub

DIRECTEUR DE LA THÈSE - THESIS SUPERVISOR

J. Oommen

CO-DIRECTEUR DE LA THÈSE - THESIS CO-SUPERVISOR

EXAMINATEURS DE LA THÈSE - THESIS EXAMINERS

M. Lanthier

A. Nayak

J.-M. De Koninck, Ph.D.

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

DEAN OF THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

PREDICTION OF STOCK MARKET PRICES USING NEURAL NETWORK TECHNIQUES

By

Zhuowen Wang, B. Computer Science,

A thesis submitted in partial fulfillment of the requirements

for the degree of

Master of Computer Science

University of Ottawa

2004

Your file Votre référence
ISBN: 0-494-01637-X
Our file Notre référence
ISBN: 0-494-01637-X

Supervisor: Mustapha Yagoub

Electronic Engineering Department

University of Ottawa

Ontario, Canada

Co-Supervisor: John Oommen

Computer Science Department

Carleton University

Ontario, Canada

ABSTRACT

Issuing stocks is the key method to raise money for corporations. Today, stocks have become the most important financial instruments. Currently, there are several methods by which one can predict financial markets, but none of them is quite accurate. After introducing some basic concepts and the history of stocks, this work continues to introduce some typical fundamental and technical analysis methods already developed by economists, and then presents a relatively new system to forecast the stock market using revised Back Propagation (BP) algorithms. The system exploits BP neural networks to help find the correlation between stock price and the affecting factors hidden behind the financial market. The topology is a typical three-layer neural network with one input layer, one hidden layer and one output layer. The supervised algorithms are the Feed-forward, Cascade-forward, and Elman BP. They are trained respectively by seven BP techniques: the Gradient Descent BP, the Gradient Descent With Momentum BP, the Gradient Descent With Adaptive Learning Rate BP, the Gradient Descent With Momentum & Adaptive Learning Rate BP, the Levenberg-Marquardt BP, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) Quasi-Newton, and the Resilient Propagation (RPROP) BP. Data used to train and test the neural networks involve the Shanghai Stock Exchange Composite Index, and the Shenzhen Stock Exchange Component Index.

Keywords: Stock Market, Finance, Prediction, Neural Network, BP Algorithm, Technical Analysis, Fundamental Analysis, Feed-forward, Cascade-forward, Elman, Gradient Descent, Quasi-Newton, RPROP, Levenberg-Marquardt BP, Momentum, Adaptive Learning Rate.

ACKNOWLEDGMENTS

Many thanks to Dr. Mustapha Yagoub for giving me inspiration and instruction when I was writing this Thesis. I really appreciate his help, advice and guide.

Many thanks to Dr. John Oommen for giving me the most valuable suggestions and help throughout the Thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

CHAPTER I
INTRODUCTION

1.1 Motivation

Stocks are attracting more and more people. The primary reason that people want to invest in the stock market is to gain profit. The profits come by investing in stocks whose price increases. Thus, what the investors are trying to accomplish, one way or the other, is to predict the future of the market. The present work is a contribution to such prediction of the stock market.

There are many methods used by analysts to predict the market, such as those involving a fundamental analysis and a technical analysis. But results have proved that none of these are quite efficient in forecasting the future trend of the stock price because different people have different opinions when it concerns interpreting the same fundamental and technical analysis reports. People have also tried to use advanced modeling methods like neural network techniques to achieve prediction, but the results have not been very satisfying because their experiments and research are limited. This thesis investigates a few revised algorithms, reports the results of a fairly large number of experiments, and presents the conclusions of novel techniques which can be used to predict the financial market using variants of the back propagation (BP) algorithms.

1.2 Thesis Proposal Overview

In the second chapter, we introduce some basic contents necessary to understand the essence of this thesis, including the history and principles of stocks. Afterwards, in Chapter 3, we introduce some fundamental analysis and technical analysis schemes. In the fundamental analysis part, we introduce two methods, namely the Earnings Per Share and Price/Earning ratio methods. In the technical analysis part, we detail three patterns (Cup and Handle Pattern, Head and Shoulders Pattern, Double Bottom Pattern) and three indicators (Moving Average, Exponential Moving Average and Moving Average Convergence Divergence). In Chapter 4, we point out the defects of the fundamental and technical analysis, and present a novel solution to forecast the stock market; namely the neural network method trained by the BP algorithm. In this chapter, we also introduce, in detail, the concepts of neural networks and the essence of the algorithms and training methods to be used in the next chapter. Finally, in the last chapter, we compare twenty-one combinations introduced in Chapter 4, and determine the best one, which can be used to forecast the stock market. We do 1050 experiments and list them in detail in Chapter 5. We conclude the thesis by forecasting the stock market and analyzing the results.

1.3 Thesis Contributions

This thesis contributes to the scientific research in the following fields:

(1) We have presented a comprehensive study of how the Back Propagation (BP) NN can be used for stock market prediction.

(2) We have studied the three main BP schemes, and seven associated updating

mechanisms. Thus, our results are, hopefully, fairly complete.

(3) Of the 21 possible NNs, our experimental results show that most of them, although well reported in the literature, do not converge for the stock market problem, and even when they converge, their prediction capabilities are poor. The reasons for this are yet unknown, but the conclusion seems to be true.

(4) This thesis has resulted in two potential publications [64] and [65].

CHAPTER II
OVERVIEW OF STOCK HISTORY AND CONCEPTS

As the most important and popular financial derivative, stocks have received the most attention since they came into being. The fluctuation of stock price in the financial market reveals the changes of the value of a corporation. It is a sensitive indicator of the corporation's achievements and performance.

In the twentieth century, maximization of a corporation's value has become its final business purpose. That is, enterprises, in every business front, make every effort to maximize the value of their stocks in the market. People assess a corporation by its stock value. If the stock volume is fixed, the fluctuation of stock price is closely related to the corporation's internal value in the financial market.

There are a lot of factors that affect stock price, such as, interest rate, exchange rate, inflation rate, business performance, and even politics and wars. Stock price fluctuates every minute in the stock exchange, and it is so important to both short period speculators and long time investors.

People invest in the stock market to make profit. The profits come by investing in stocks whose price goes up. Thus, what they are trying to accomplish, one way or the other is to predict the future of the market.

Many years ago, economists began their research on forecasting stock price. Basically, there are several kinds of methods useful to predict the stock price. Before we introduce these forecasting methods, in order to make the reader understand this work better, we would like to present some basic knowledge of stock and finance.

2.1 Introduction to the History of Stocks

Stock is an outcome of the commodity economy, and a corporation's productivity and growth. Its history can be described by three steps.

2.1.1 Sixteenth Century Trading

In the 16th century, as a method to raise money and decentralize risk, stock permeated throughout long voyage trade. The history can be traced as shown below.

In the 15th century, the Italian voyager, Christopher Columbus, discovered South America. After that, Ferdinand Magellan, a Portuguese seafarer, accomplished the voyage around the whole earth for the first time. These endeavours broke a new path between the east and the west. Overseas trade became a shortcut to be rich overnight. But the long oversea voyage cost a fortune. Furthermore, the voyagers often met cyclones occurring in the sea and were attacked by aboriginal inhabitants. The voyagers always ran the risk of loss and being attacked. At that time, there was no single wealthy individual who was rich enough to afford the voyage himself, and willing

to take such a big risk.

In order to raise money for the long voyage and decentralize the risk, the prototype of stocks came into being. People raised money by distributing shares before the voyage, and returned the money and shared the profits after the voyage based on the proportion of investment. U.K. and Holland even issued laws to protect such entities. Several years later, the companies changed their policies to keep the capital raised from the investors to run the business in the long run instead of returning the money when the voyage was finished. This became the prototype of common stocks.

2.1.2 Seventeenth to Nineteenth Century Trading

From the 17th century to the mid-19th century, the industrial revolution broke out in U.K. and France. Large industry overran small-sized "cottage industries", and commodity economy grew fast. The banking and transportation industry required heavy capital to extend their businesses.

In those days, the popular way to raise money was to issue stocks and establish corporations. The Bank of England, the first capitalist national bank, was founded in 1694 by issuing stocks [1]. Because stocks of banks have the properties of higher profit and lower risk than those related to long voyages, stocks became more and more popular in the finance industry. Along with the invention of the steam engine, the industrial revolution spread by means of the steamship and the associated machinery. Several capitalists' investments couldn't satisfy the industrial financial need any longer. Thus, issuing stocks was the only way to raise such a large amount of money. With the capital raised by issuing stocks, the U.K. and USA built thousands of miles of canals and railroads.

After the 1960's, corporations which were established by issuing stocks became dominant in the industry. Entrepreneurs could raise increasingly more money by issuing stocks. For example, by issuing stocks, U.S. Steel was capitalized at $1,400,000,000 in the early 20th century and became the first billion-dollar corporation in American history [2].

2.1.3 Stocks and Regulations

As the stock trade grew, related regulations and laws became more mature. Also, the stock exchange had been developed to meet the need of growing stock trade. The London Stock Exchange was formed by several stockbrokers in the U.K. in 1773 [3]. The world's largest marketplace for securities, the New York Stock Exchange was formally constituted as the New York Stock and Exchange Board in 1817 [4]. The Dow Jones Industry Average was computed in 1897 [5]. Today the Dow Jones Industrial, Transportation and Utilities Averages have been considered the world's leading indexes.

In order to protect and regulate the development of corporations and stocks, corporate laws, securities laws and bankruptcy laws were enacted in many countries. These laws successfully protected the rights and privileges of stockholders.

Stocks emerged in China in the mid-1980s. The Shanghai Stock Exchange (SSE) was founded on November 26, 1990 and began in operation on December 19 of the same year [6]. The Shenzhen Stock Exchange (SZSE) was established on December 1st 1990 and was located in the coastal city of Shenzhen. It is one of the two stock exchanges in mainland China. Both SSE and SZSE are non-profit-making membership institutions directly governed by the China Securities Regulatory Commission (CSRC) [7].

Thanks to these facilities, it is easy to obtain all types of data from the Chinese stock market to accomplish this research. Consequently, the main part of the data used in this thesis is from the SSE and SZSE, and the experiments of this work are based on SSE Composite Index and SZSE Component Index.

2.2 Introduction to the Modern Concepts of Stocks

2.2.1 Modern Concepts of Typical Stocks

Today, as the most important form of securities, stock has been associated with various concepts. *Stock* is the capital that a corporation raises by selling shares that give the shareholders rights to the profits and to other privileges of ownership; it represents an ownership position in a corporation, and a claim on its share pro rata in the corporation's dividends and assets [8] [9] [10]. Most stock also provides voting rights, which give shareholders a proportional vote in certain corporate decisions. In general, there are two types of stocks, preferred stocks and common stocks.

Preferred Stock entitles the holders to a greater claim on dividend and, in the event of bankruptcy liquidation, to the corporation's assets, than the "common stock" holders [10] [11]. Preferred stock holders have partial ownership on the corporation but do not enjoy any voting rights, and therefore, have no power to determine the corporation's business. Preferred stock holders always receive their dividends first. Furthermore, the dividends are fixed and do not fluctuate.

Common stocks are ordinary capital shares of a corporation that have exclusive residual claim on the net assets and net income of the corporation after all prior claims have been paid, they entitle the holders voting rights and equity ownership in a

corporation [10] [12]. Common stock holders enjoy a corporation's dividends and capital appreciation. If the corporation goes bankrupt, the common shareholders claim the corporation's assets after the bondholders, other debt holders and preferred stock holders.

In addition to voting rights, common stock holders are also entitled to preemptive rights, which allow common stock holders to purchase as many new shares of the stock as necessary to keep their proportional ownership in the corporation when the corporation issues new stocks. Stocks traded in the stock exchange are common stocks. A corporation always has a large number of common stocks but only a small quantity of preferred stocks. Some corporations do not even have preferred stocks at all. Therefore, common stocks are much more important and represent the value of a corporation. In this work, we only discuss common stocks that are traded in financial markets.

2.2.2 Some Other Important Concepts Related to Stocks

One should understand some terminologies when s/he studies the financial chart. The price of the first transaction of a day is called the *opening* or *open price*, and the price of the last transaction of a day is called the *close* or *closing price*. The highest price reached in a day is called the *daily high*, the lowest is the *daily low*. The number of the stock shares traded in a day is called the *volume* or *trading volume*. *Amount* refers to the total value traded in the transactions in the day. The closing price indicates the stock's final value in a day that people agree with. It also has impact on the opening price of the next day. Therefore, the closing price is more important than the opening price. When people talk about financial market, they refer to closing price.

2.3 Conclusion

This chapter introduced the history, typical types and some of the most important concepts of stocks. Of the two types of stock, common stock is more popular. Today, when people talk about the stock market, they refer to the common stocks. Common stocks have different kinds of prices, and the closing price is the most important index because it not only is the final price level of a day, but also affects the opening price for the next day. Therefore, this thesis discusses the closing price of common stock.

CHAPTER III

LITERATURE REVIEW OF STOCK PREDICTION

3.1 Fundamental Analysis to Forecast the Stock Market

Fundamental analysis is a security valuation method to assess a corporation's earning, sales, assets, debt, financials and growth potential [13] [14] [15]. Fundamental analysts only consider the factors directly related to the corporations themselves, instead of considering technical analysis indexes or factors affecting the overall financial market, such as a presidential election, wars and oil crisis. There are two main metrics for Fundamental Analyses: Earnings Per Share (EPS) and Price Earning Ratio (PE).

3.1.1 Earnings Per Share (EPS)

Earnings per Share (EPS) is the portion of a corporation's profit assigned to each share outstanding of common stock [13] [14] [16]. EPS indicates how much earning each share of stock can contribute to its holder, that is, the profitability of a corporation.

The formula

$$EPS = \frac{Total\ Net\ Earnings - Dividends\ on\ Preferred\ Stocks}{Number\ of\ Shares\ Outstanding\ of\ Common\ Stocks} \quad (3\text{-}1)$$

calculates the amount of net income that each share of common stock contributes, on average, to the holders. This is the most simple and popular measurement in dictating a share's price. A large number of EPS means that a corporation has a high profitability and indicates a bullish (upward movement) future of the stock, and vice versa. For example, let us assume that last year's EPS for a corporation was $1, and this year's EPS is $2. Under the condition that other affecting factors remain the same, this corporation's stock price this year should be twice of what it was last year. This indicates a soaring trend. Furthermore, EPS statistics information is easily obtained from a corporation's financial statement, which is reported at least once a year.

As an illustration, Figures 3-1 and 3-2 (referenced from BELCO's official web site [17]) show the EPS and share price of BELCO Holding Limited (BHL), which invests in energy and utility related services, located in Hamilton, Bermuda. BHL stocks trade on the main board of the Bermuda Stock Exchange (BSX).

In these two figures, BHL's share price goes roughly in the same trend as its EPS does from 1993 to 2002. In 1999 BHL experienced a low EPS with values between $3.00 and $3.25, and accordingly, its share price dropped into a trough with a value of about $21.

Because EPS roughly indicates the stock price trend and its statistics data are easy to find, most investors take EPS into consideration when they evaluate a stock's value.

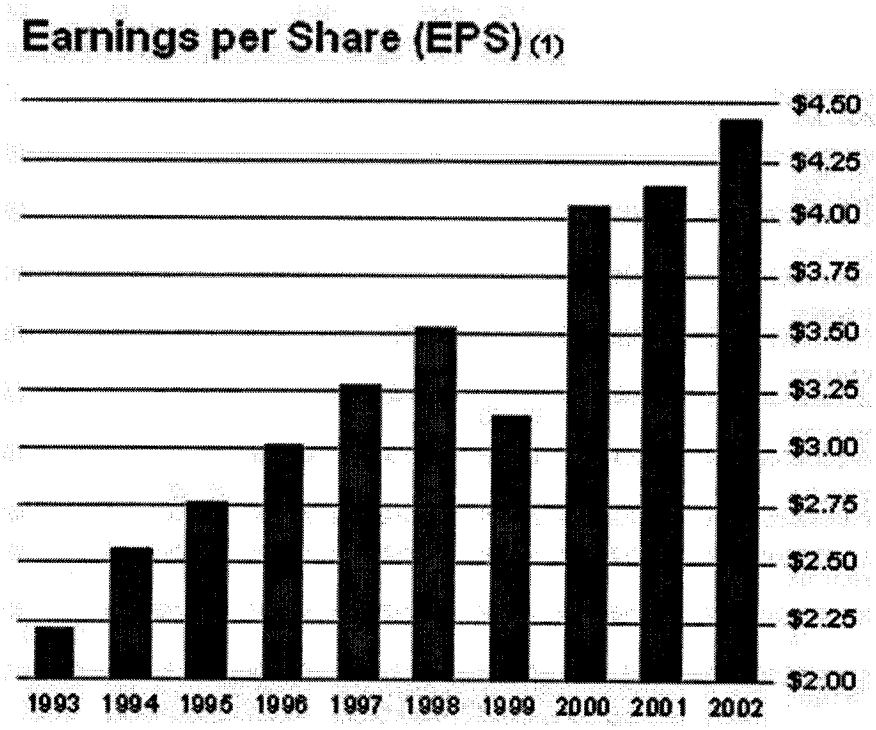

Figure 3.1 Earnings Per Share of BELCO Holding Limited (BHL)

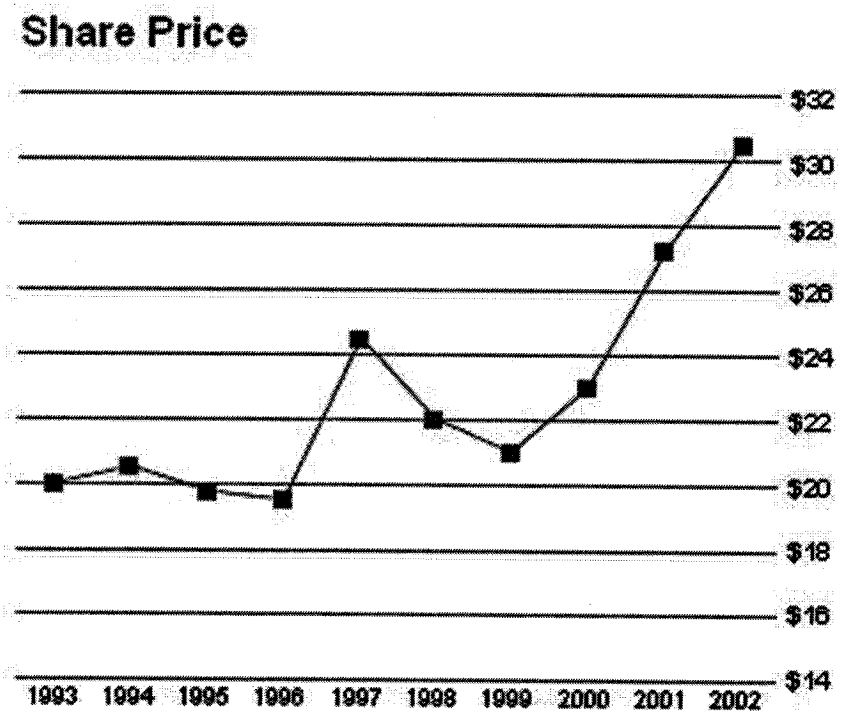

Figure 3.2 Share Price of BELCO Holding Limited (BHL)

3.1.2 Price/Earnings Ratio (P/E)

P/E ratio is a valuation ratio of a corporation's stock price compared to its Earnings Per Share [13] [14] [18]. It is used by many people to determine whether the stock market or a specific stock is expensive or cheap. It is also referred to as the confidence that the investors have in the corporations. As its name implies, the P/E ratio is calculated as:

$$P/E = \frac{Market\ Share\ Price}{Earnings\ Per\ Share\ (EPS)} \tag{3-2}$$

Basically, people calculate the P/E ratio using the EPS from the last four quarters. This is also called the trailing P/E. Occasionally, the P/E ratio is calculated using the EPS figures coming from estimated earnings expected during the next four quarters. This kind of P/E is also known as the projected or leading P/E. The third kind of P/E uses the actual EPS of the past two quarters, and the estimated EPS of the next two quarters [13] [14] [19] [20].

Analysts regard P/E ratio as an indication of how confident the investors are about the corporation's future. For example, if the P/E ratio = 2, that means investors are not very confident in the company. Only if they can get their investment back within about two years, they will consider a buy. They might not expect any profit at any time beyond two-year period. At that time, the corporation maybe goes bankrupt or it is not be worth investing in it any more.

On the contrary, if the P/E ratio is 90, it means a high projected earnings in the future and shows the market has strong confident in the corporation. The investors are still willing to buy even though their investment will not yield returns until 90 years later. In other words, in order to own a share of the corporation, the investors are currently paying the equivalent of 90 years' worth of earnings.

The P/E ratio is a vital indication that investors often consider upon buying or selling stocks.

3.1.3 Conclusion

Fundamental analysis studies the financial statement, such as the balance sheet, statement of cash flow and income sheet. The research is primarily based on the EPS and P/E. The area of research is definitely too narrow, and it thus causes some inevitable defects that will be discussed in the following chapter. Therefore, a fundamental analysis cannot be used solely to predict the stock market. Instead, it must be considered together with some other forecasting tools, such as technical analysis.

3.2 Technical Analysis to Forecast The Stock Market

Technical analysis involves examining statistics generated by the historical price, volume, and market activity. This type of method presumes that data from the financial market are helpful to forecast future price trend. It assumes that market psychology affects stock trading in a way that enables the analysts to forecast. Currently, it is becoming more and more popular, since more people believe that historical statistics are a strong indication of the future. Different technical analyses are usually used together to confirm the patterns, trends and turning points in the market.

Unlike the fundamental analysis, technical analysis does not study the corporation's financial statement or the intrinsic values of the stocks. Instead, it only takes into consideration market data, such as the price and volume. Technical analysts believe that the market moves in a predictable pattern and that market itself can reveal the future trend. They thus transform the same price chart into different technical analysis charts by doing arithmetic operations of adding, subtracting, multiplying and

dividing, and then search for defined patterns and indicators in the chart, attempting to thus predict the trend reversal and price. Some speculators have even succeeded in trading stocks by studying only technical analysis charts even though they do not even know what the companies actually do. There are hundreds of technical analyses that are used as indicators in the market. The most widely used patterns are Cup and Handle, Head and Shoulders, and Double Bottom. The most basic indicators are MA (Moving Average Line), EMA (Exponential Moving Average), and MACD (Moving Average Convergence Divergence). Each of these will be described in the sub-sections that follow.

3.2.1 Cup and Handle Pattern

This pattern bears the name as because it has a shape like a cup with a handle on its right side, as shown in Figures 3.3 and 3.4.

The pattern can span more than one year or as short as several weeks. The bottom of the cup is a U shape. The handle is a bar with a slight downward drift. As the price goes up to test the old high, the stock will incur selling pressure since many people who bought stocks at or near this point are selling their shares to balance their costs. Therefore the stock price will move down a little bit. This downtrend may last several weeks, and then the price often experiences a "break out".

Figure 3.3 is an instance selected from Yahoo! Finance. It is a price chart of Giga-Tronics Incorporation, San Ramon, California, USA [21]. This chart starts at May 2003 and ends at March 2004. From August 2003, we observe that the shape of a 'cup' emerged; it lasted for three months to finish the whole 'cup'. Then the price moved toward a downside to construct the 'handle'. After a two-month period of accumulating strength, the price came up to fight the selling pressure and then "broke out". A perfect 'cup and handle' pattern had thus finished.

This pattern is a classic should-buy indication since it suggests a bullish trend. At the end of the handle, point B, the 'cup and handle' pattern can be confirmed. And the

investors may consider a buy since the risk here is relative low.

Figure 3.4 is another good example that is found on the Yahoo! Finance web site [22]. The price of CompuDyne Corp. Finished the construction of the cup's bottom shape in late November 2003, and then the 'handle' in late December 2003. As the B point flagged in Figure 3.4, there is a low risk if the investors buy at this moment because the 'cup and handle' pattern has been clearly confirmed, and it implied that the price could soon be on the way up.

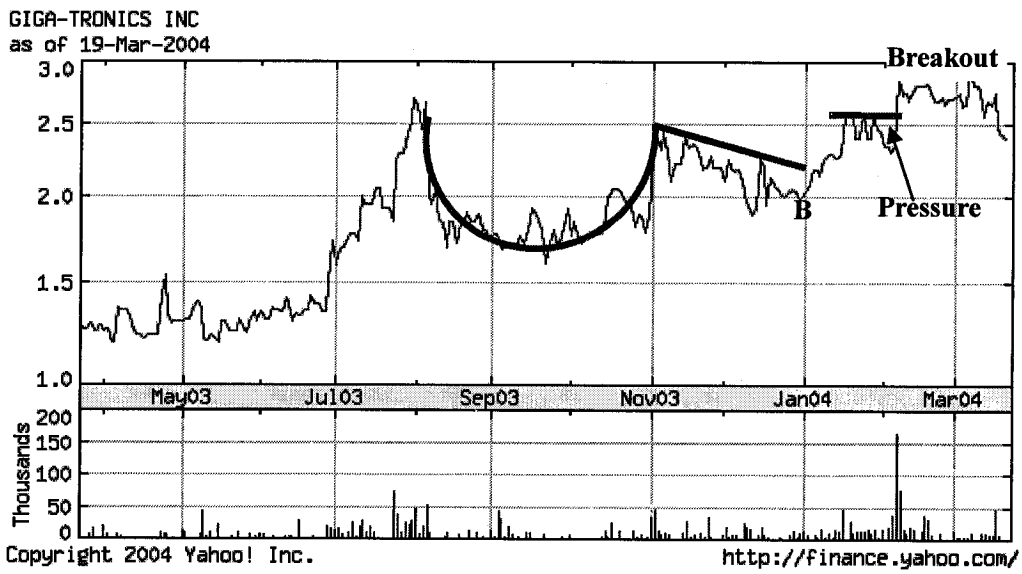

Figure 3.3 Cup and Handle Pattern (Giga-Tronics Inc.)

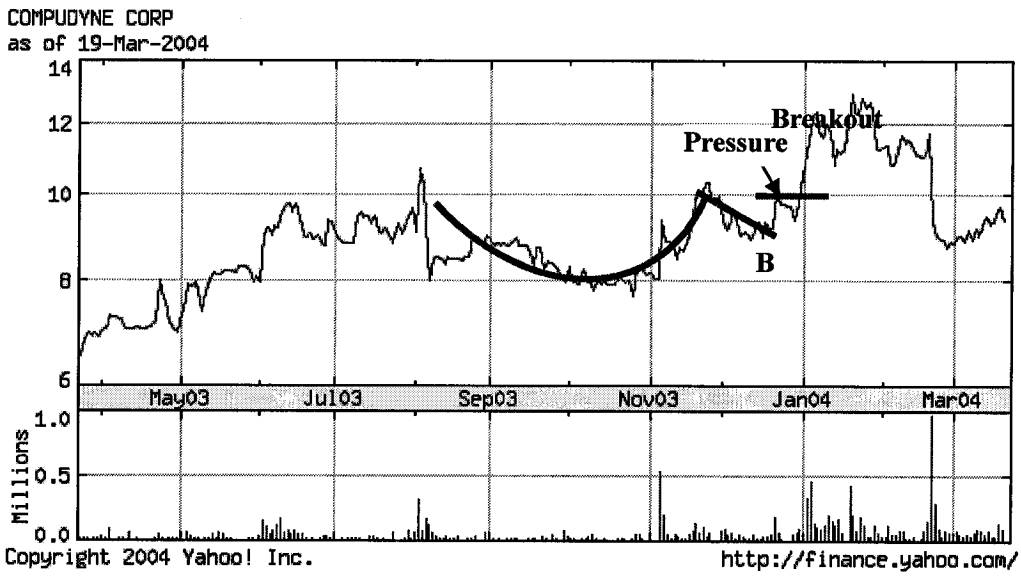

Figure 3.4 Cup and Handle Pattern (CompuDyne Corp.)

3.2.2 Head and Shoulders Pattern

There are two types of 'Head and Shoulders' Patterns: one in an uptrend, the other one in a downtrend. Both of them are regarded as a hint of the reversal.

This pattern in an uptrend has a shape like a head between left and right shoulders. The price goes up to a peak and then down; rises up again above the previous peak, declines again; and it rises the third time but declines before it reaches the second high, as shown in Figure 3.5 [23]. The price dropping before it touches the second peak means that its strength has been exhausted and will plunge soon. The first and the last peaks form the two shoulders; the second peak is the head. This pattern suggests a bearish (downward moving) market.

The formation of the left shoulder began on January 3, 2003 and ended on March 12, 2004. And the price rose up and formed the head in about seventy days, ending on May 20, 2003. Then the price rose again in May 20, 2003. But it did not reach the second peak and declined.

This is a clear reversal trend. It is not a surprise that price of Guangcai Construction Co. Ltd. Stocks dropped down all the way until mid November 2003, lasting five and a half months. It lost 50% value from RMB ¥ 10.00 per share, the peak of the head, to RMB ¥ 5.11, the bottom in November 18, 2003. Although there is a wave-up attempt from July 29 to September 4, 2003, it does not change the overall downward trend at all.

The 'Head and Shoulders' Pattern in a downtrend is found less often than that in an uptrend. Its shape is like an upside-down head and shoulders, which implies a rise as shown in Figure 3.6 [24]. This chart is selected from the Yahoo! China finance channel. The company in this example is a Chinese enterprise, Wuhan Plastic Machinery General Factory, listed on the Shenzhen Stock Exchange.

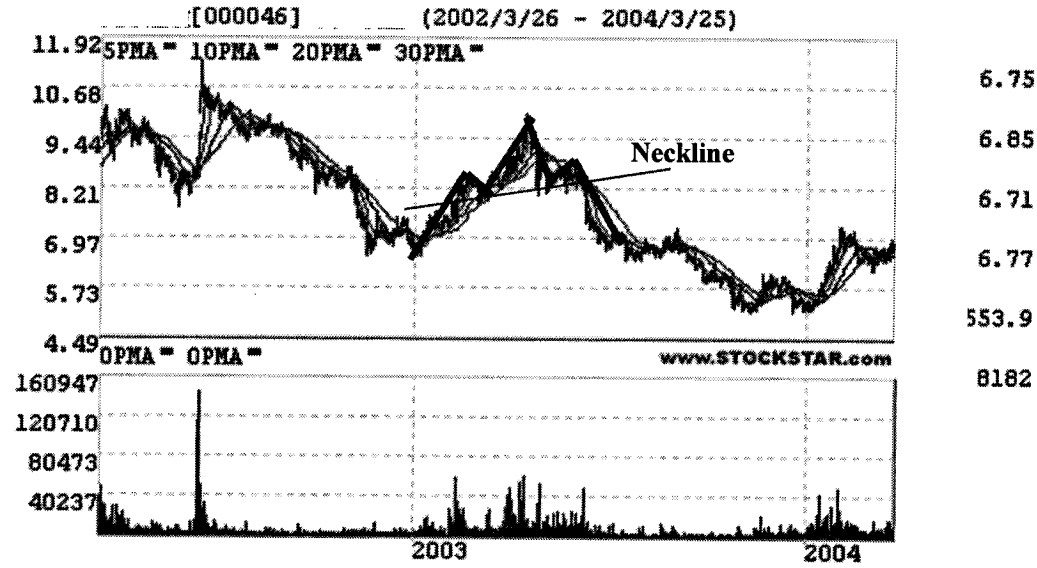

Figure 3.5 Head and Shoulders Pattern in an Uptrend (Guangcai Construction Co. Ltd.)

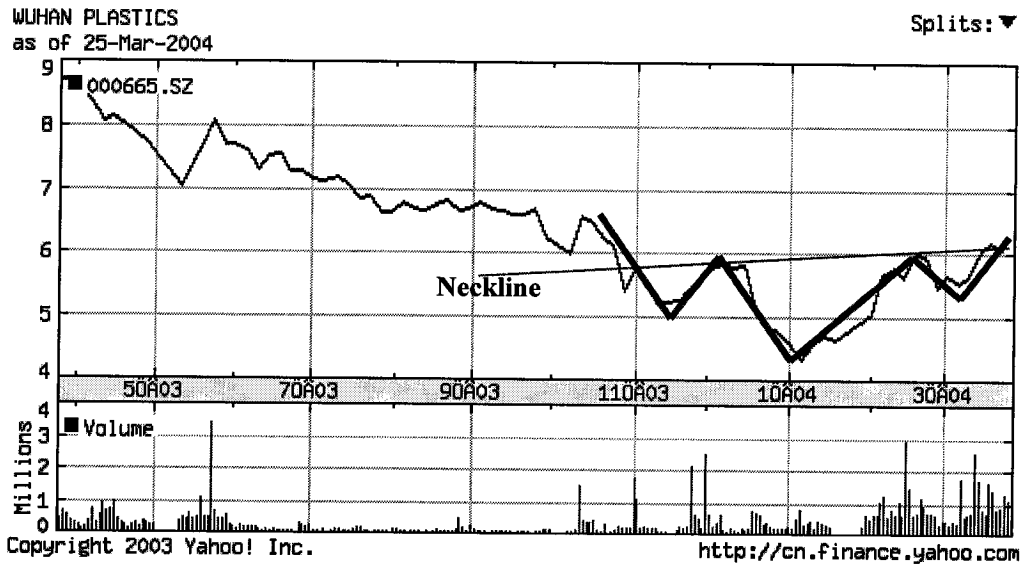

Figure 3.6 Head and Shoulders Pattern in a Downtrend (Wuhan Plastic Machinery General Factory)

3.2.3 Double Bottom Pattern

The 'Double Bottom' Pattern is also known as the 'W Bottom' Pattern since its bottom is like a shape of W, with two similar price levels at the bottom. This pattern may span as short as a few weeks or as long as many months.

As demonstrated in Figure 3.7 [25] (selected from Yahoo! Finance China), the Chinese enterprise, Shenzhen Huafa Electronics Co. Ltd. Experienced a bearish trend in the market from September 2003. Since then its share price plunged from the top point of RMB ¥ 9.85 all the way down to RMB ¥ 5.42, in November 2003. The price rose after November 2003 and dropped again from mid December 2003. When the price dropped to the similar low level, it rose again. The double bottoms formed.

Usually, after the rightmost side of the 'W' shape rises above the middle point of the 'W', the 'Double Bottom' Pattern can be confirmed. The 'Double Bottom' Pattern is pretty easy to find and recognize in the market. Figure 3.8 lists some examples currently in China's the stock market [26].

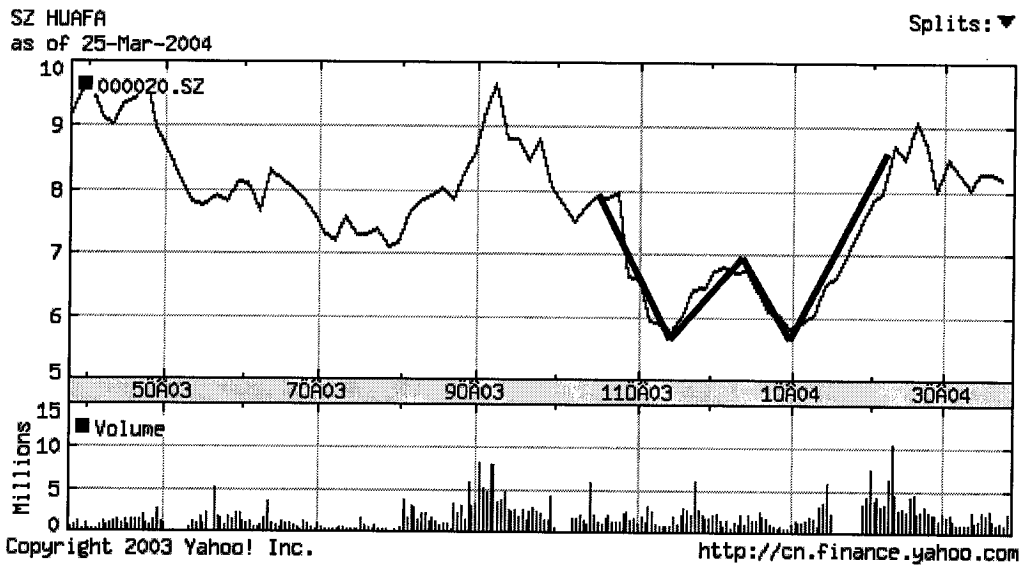

Figure 3.7 'Double Bottom' Pattern
(Shenzhen Huafa Electronics Co. Ltd.)

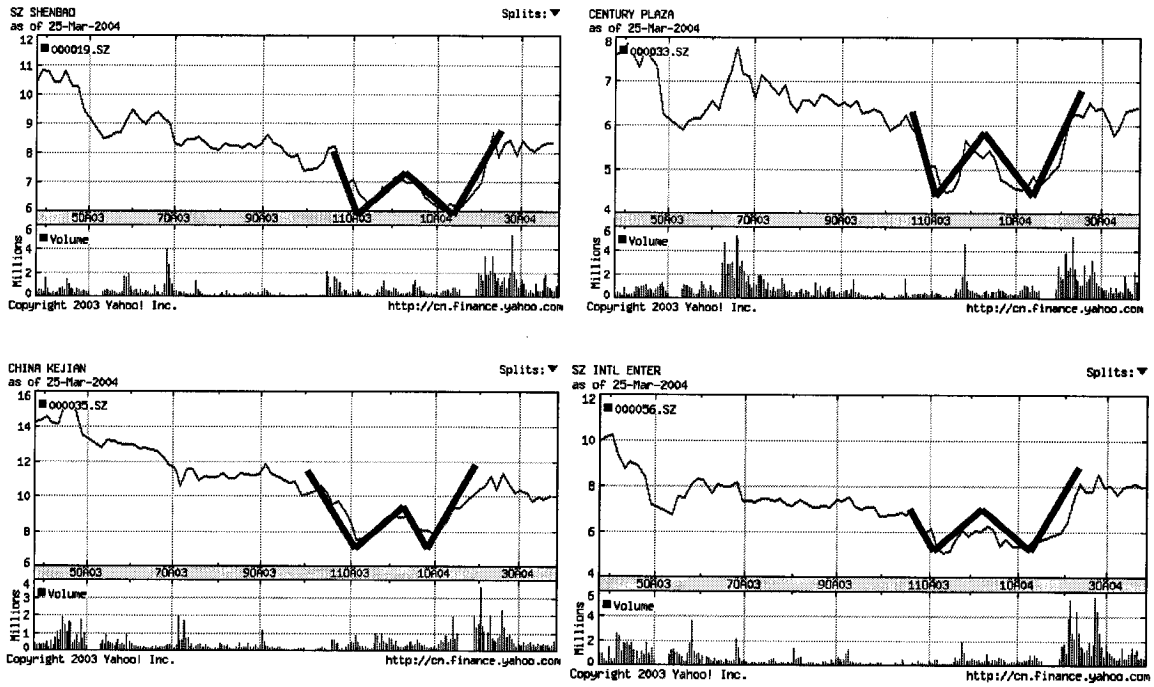

Figure 3.8 Four More Examples of the 'Double Bottom' Pattern (Enterprises Selected From China's The stock market)

Besides the patterns, there is another type of technical analysis indicator, which is widely used by analysts to predict the market.

3.2.4 MA (Moving Average Line)

This indicator is used more widely than any of the patterns discussed above, and is probably even the most-often-used one of all the indicators. It can not only help in finding the direction of the trend, but also help locate an oversold and overbought market. The MA indicator may suggest a buy-and-sell signal when used together with other indicators. It can be regarded as a filter or smoothing method to remove the noise of the market.

The MA indicator is calculated as the following formulation:

$$P_A = \frac{1}{N}\sum_{i=1}^{N} P_i \qquad (3\text{-}3)$$

where P_A is the moving average line of N days, P_i is the price of the i^{th} day, i is the current day, ranging from 1 to N. N is the number of samples used to calculate the indicator. For example, if the investors calculate Friday's value of a five-day moving average line, they add up the daily prices from Monday to Friday and divide the sum by 5.

Usually the number *N* could vary from as small as 3 to even as big as 200, which is considered a long term indicator. The most popular way to use the Moving Average Line is to study the way that the price line crosses MA. There are eight most important patterns that analysts conclude from their experience.

Figure 3.9 is a classic should-buy pattern. The dotted line is the MA, and the solid line is the price. The MA line goes down and recently rises. The price line is consistently below the MA; it recently goes up and rises more quickly than MA does. After it crosses the MA line, one may consider a buy. The position of the red dot signals the suggested time to enter the market with a buying trade.

Similarly, in Figures 3.10, 3.11 and 3.12, the red dots suggest a buy. The main difference is: MA is rising in Figures 3.10 and 3.11, but falling in Figure 3.12.

There are another four patterns that investors often use to locate alerts to selling stocks that they hold. To confirm the pattern, the investors should be careful: the MA in Figure 3.13 first rises and then goes sideways; the MA in Figure 3.14 and 3.15 is falling; the MA in Figure 3.16 is rising, the price line is far above the MA and plunges.

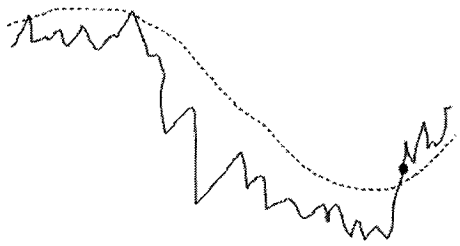

Figure 3.9 Should-Buy Pattern 1

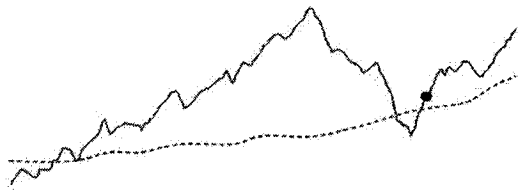

Figure 3.10 Should-Buy Pattern 2

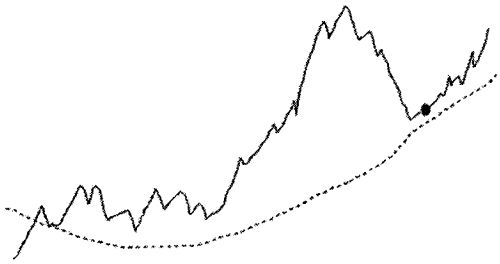

Figure 3.11 Should-Buy Pattern 3

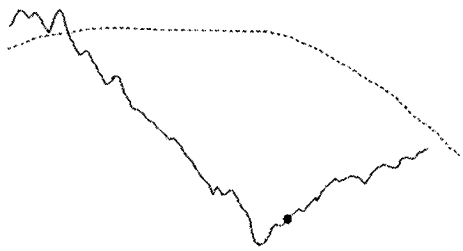

Figure 3.12 Should-Buy Pattern 4

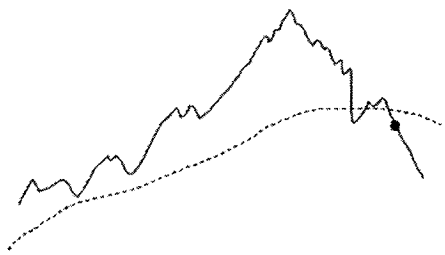

Figure 3.13 Should-Sell Pattern 1

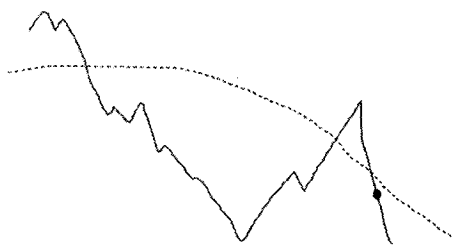

Figure 3.14 Should-Sell Pattern 2

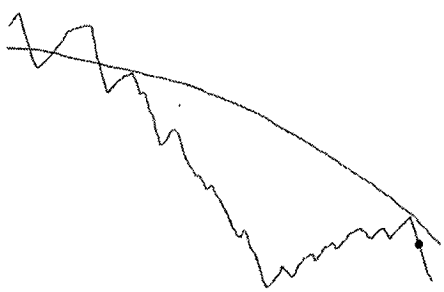

Figure 3.15 Should-Sell Pattern 3

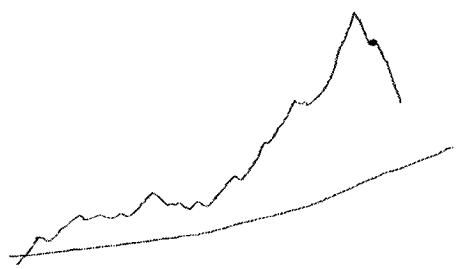

Figure 3.16 Should-Sell Pattern 4

Another popular method that analysts often use is to study Moving Average Cross. The investors may compare different lengths of the time span used in the Moving Average Line, such as a five-day, a ten-day or a thirty-day MA, which respectively indicate a short, a medium and a long period trend. When the long period line (the dotted line in Figure 3.17) changes from going up to moving horizontally or even downward, and it is crossed by the medium (dashed line) and then the short period trend line (solid line) respectively from above, it usually indicates a bearish trend. The time span of the moving average line may vary in a wide rage. Usually it can be as short as 5 days or as long as 55 days. The intuition here is that the short term line is more sensitive than the long-term one. When the price starts to plunge in a rising trend, the short term line reverses first. At that time, the long term line may be still rising. Thus in a bearish market, the short term line always crosses the long term line from above. The same theory may be applied to the rising trend, as shown in Figure 3.18. It is roughly a reverse plot of the bearish trend.

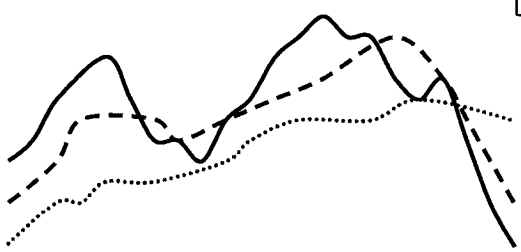

Figure 3.17 MA Cross – Pattern 1: Bearish Trend

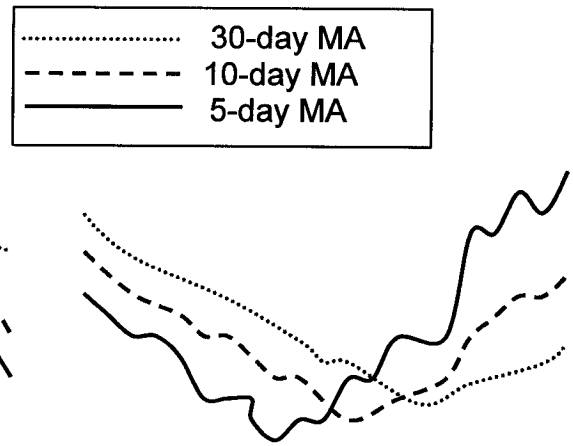

Figure 3.18 Moving Average Cross – Pattern 2: Bullish Trend

3.2.5 EMA (Exponential Moving Average)

In the Simple Moving Average, the oldest price is discarded from the system when the latest price is added. But Exponential Moving Average never discards the old price data. EMA applies a weight to the past data by a transfer function. Thus the EMA assigns less weight to the old data. The older the data are, the less impact they have on the EMA. The EMA is calculated as:

$$EMA = Ft+1 = (1-a)Ft + a \cdot Pt \tag{3-4}$$

where a is exponential percentage, $a \in [0.1, 0.9]$;

$$a = 2/(time\ period + 1) \tag{3-5}$$

$$time\ period = (2/exponential\ percentage) - 1 \tag{3-6}$$

P_t: the closing price of the i^{th} day;

F_t: the *EMA* value of the t^{th} day;

F_{t+1}: the *EMA* value of the $(t+1)^{th}$ day.

The initializing *EMA* value, F_1, is closing price. The table below is an example of *EMA*, where a = 1/3 and *time period* = 5. This is a five-day *EMA* with *exponential percentage* value of 1/3, which means, in this system, the weight of today's price is 1/3, while the weight of the past five-day exponential moving average is 2/3.

Table 3.1 EMA

Day	1	2	3	4	5	6	7	8	9
Price($)	10	11	13	12	13	11	12	15	18
EMA: 0.33333 (or 1/3)			11.33	11.55	12.03	11.69	11.79	12.86	14.57

The initializing EMA in the above table is the simple moving average of the first two days, Day 1 and Day 2. It is calculated as (10+11)/2=10.5, and the EMA is calculated as:

13*1/3+10.5*2/3=11.33;
12*1/3+11.33*2/3=11.55; ... ...

In the calculation above, the weight of the whole past is 2/3, and the weight of today's price itself is 1/3. By doing this, we weight recent values more heavily than the past ones. Figure 3.19 shows that the EMA goes closer to the price than the simple MA does. The corporation is a Chinese Internet company, Sina.com, listing on NASDAQ, whose market value is 2.1 billion (according to data of April 9, 2004). The blue line is its price [27]. It is obvious that 10-day EMA is closer to the real price (blue line) than 10-day MA.

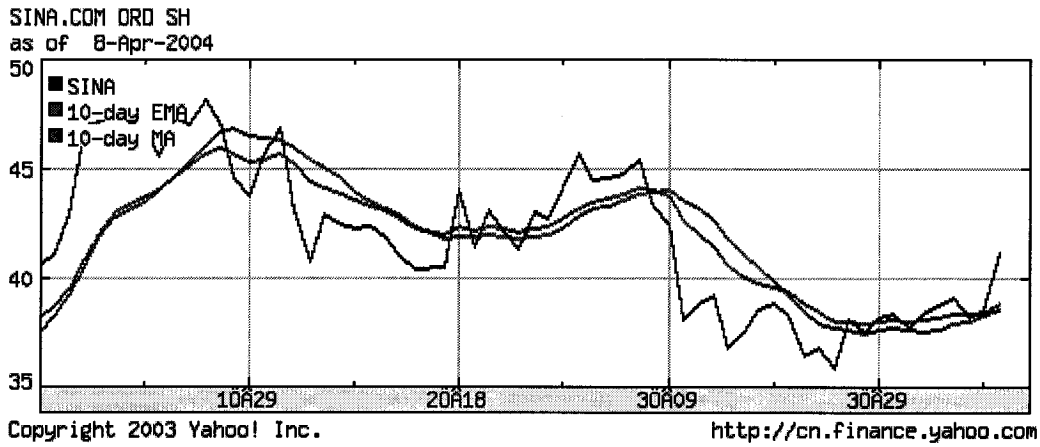

Figure 3.19 Contrast Between EMA and simple MA of Sina Corp.

Trading Signals Generated by EMA

The trading approach for the EMA is quite similar to the simple MA system. But the EMA is more sensitive than the Simple Moving Average and reacts faster to recent price changes. It is considered a selling signal when the EMA is moving downtrend and the price crosses the EMA from above to below, as shown by the dashed circle in Figure 3.20. Similarly, a buy signal is generated when the EMA is going upward and the price crosses it from below to above. Another way is to look for the cross of short and long period EMA. A buy signal is generated when the long period EMA demonstrates an uptrend, and is crossed by the short term EMA from below to above, and vice versa, as shown by the solid circle in Figure 3.20 [28].

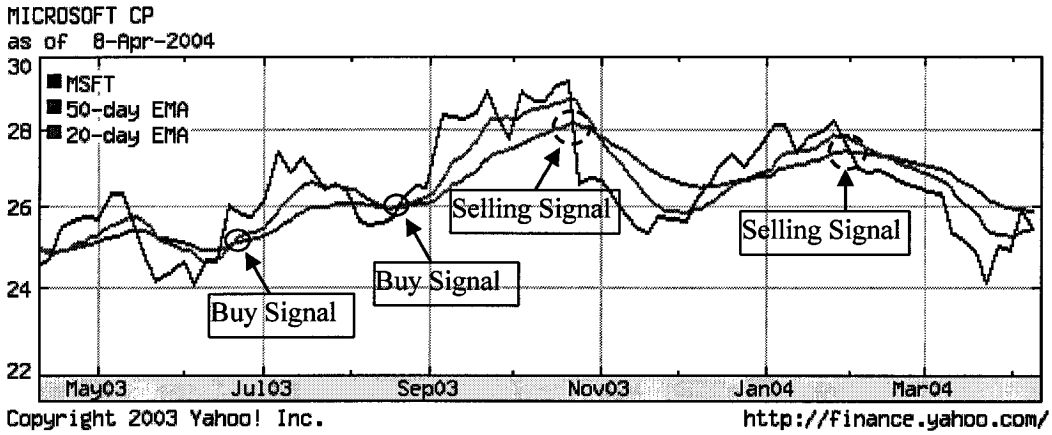

Figure 3.20 Long and Short Term EMA of Microsoft

3.2.6 MACD (Moving Average Convergence Divergence)

The Moving Average Convergence/Divergence strategy was invented by Gerald Appel in 1960's, and further developed and improved by others later.

The MACD is an indicator that reveals the relationship between long and short period exponential moving averages of prices. The long period EMA is usually a 26-day EMA. The short period EMA is usually a 12-day EMA. The MACD line is plotted by subtracting the 26-day EMA from the 12-day EMA. The difference may be a positive or a negative number. Another line plotted in the MACD analysis is called the signal line, which is the 9-day EMA of the difference between the 12-day and 26-day EMA. The formula is:

$$MACD = 12\ day\ EMA - 26\ day\ EMA \tag{3-7}$$

$$Signal\ Line = 9\ day\ EMA\ of\ MACD \tag{3-8}$$

The MACD is a centered oscillator that fluctuates above and below zero as the short term EMA rises above or falls below the long term EMA.

MACD Trading Signal

The MACD line stays far above zero when the short term EMA rises far above the long term EMA, and vice versa. The MACD stays at zero when the two EMAs cross each other. If the MACD fluctuates between 1.0 and -1.0, the market is ranging and the signals are not reliable. If the MACD stays far above or below zero, the signals are strong. It is a good signal to buy when the MACD crosses the signal line from below, and conversely a signal that suggests a good selling option occurs when the MACD crosses the signal line from above.

When the MACD falls far below zero, it suggests an oversell market and indicates that the stock price will rise soon. The converse is also true: when the MACD rises far above zero, it suggests an overbought market and indicates that the stock price will fall soon.

3.2.7 Overall Remarks

From an overall perspective, we see that what technical analysis studies, is the price chart. The technical analysts try to categorize the chart into different patterns and look for the pre-defined patterns in the charts. But typically, the pattern is not efficient or accurate enough. The technical analysts place more trust on the patterns than they should by merely relying on such an analysis. Therefore, we believe that there is a need for further research to provide other methods to forecast the stock market.

3.3 Conclusion

The fundamental analysis for stock markets only looks at the corporation's financial statement. This feature suggests that it should complement other tools. On the other hand, the technical analysis only studies the charts and looks for a few pre-defined patterns. Although the latter is better than a fundamental analysis, it is still not a good forecasting tool. Because of the poor results obtained from a fundamental and technical analysis, people have started to look for a new strategy to analyze the stock market, and these involve neural network forecasting tools, which we shall currently study.

CHAPTER IV
NEURAL NETWORK TO FORECAST STOCK PRICE

The development of computer technology is the crowning success of the 20th century, and the field of neural networks is one of the most amazing breakthroughs in this "hi-tech" field. The field of neural networks (NNs) is a new paradigm for computing, that involves numerous parallel processing elements, analogous to the biological brain. NNs have been widely used to achieve different forecasting tasks. With the aid of NNs, people have been able to forecast weather, predict sales data for enterprises, and also forecast stock prices for investors.

4.1 Why Use Neural Networks to Forecast Stock Market?

Neither the fundamental nor technical analyses can really reveal the rules hidden behind a stock price. The EPS and P/E ratio are the most important and powerful fundamental analysis methods. Unfortunately, even they cannot provide a precise scheme for predicting prices that satisfy investors. The fundamental analysis focuses on

the corporation's financial statement. The information it studies is typically what transpired a long time ago, and sometimes even a full year ago. Therefore, it cannot be considered to be a real time analysis in any sense of the word.

The obsoleteness of the data that a fundamental analysis studies, renders such an analysis inadequate. Furthermore, the fundamental analysis only studies the impacts that the financial statement makes on the stock price, even though there are other issues that affect the price – issues that the financial statement cannot disclose. Thus, such an analysis is inadequate to forecast the stock market. There are two kinds of elements that affect the stock price: enterprise-related and non-enterprise-related information. Apparently the latter cannot be revealed by a fundamental analysis. For example, people usually have pessimistic expectation on the future of the stock market when the country is involved in a war. Thus, they attempt to clear what they have. In such a case, the non-enterprise-related information affects the stock market but cannot be inferred by a mere fundamental analysis.

Human elements also constitute important affecting factors that should not be ignored. Any price is a result by one person agreeing to sell and another agreeing to buy. The decision is based on their personal expectations of the future price. Different people have different opinions on the same stock, and different issues have varied impacts on different individuals. The thought process of human beings' cannot be easily quantified or predicted by such an analysis. For example, a person's family, divorce, health, accident, education and career, can all have an impact on his confidence, expectations and decisions.

The involvement of human emotion in the picture challenges the fundamental analysis. If the investors are perfectly logical and make investment decisions without being affected by personal emotions, the fundamental analysis, which is based on

earnings, will determine the stock price. Now that all the investors have the same logical and reasonable expectations on the stocks, the prices will not change unless all the relevant financial reports and news items are released.

4.1.1 EPS is not reliable

It has been argued that the EPS indicator has some serious flaws, which render it a poor forecasting tool for the stock price. The defects in seven aspects can be summarized as below [29]:

(1) EPS is a kind of accounting (reported) earnings "metric". Different accounting methods result in different EPS reports. Alternative accounting methods may be exploited to make the reported profits change, but in fact, they do not affect the corporations' real value and profitability at all.

(2) Risks (both business risk and financial risk) are not accounted for in a corporation's annual reports. A high EPS may be result from a high risk speculation. Indeed, ill-fortune from a previous year does not imply ill-fortune this year.

(3) Investment requirements (changes in, for example, the working capital) are typically not considered in the reported earnings.

(4) The Dividend distribution policy is typically ignored. The dividend policy makes the annual dividend fluctuate, thereby making the reported earnings appear to be more or less in the corporation's financial statement, but which does not, in fact, have impact on the real value of the EPS.

(5) The time value of money is not taken into consideration. A dollar last year is not

equal to a dollar this year. The corporation's profit does not emerge, all at the same time, but over a period of time. The income comes day by day, little by little. One dollar earned in the early of the year is more valuable than one dollar earned at the end of year. But an EPS calculation just simply adds them up, and divides it by the shares. Thus EPS does not reveal the real value of a corporation, and thus, it cannot be used to forecast stock price in a good way.

(6) In the late 20th century, intangible factors played roles in business, which transformed the industry economy into a services-and-knowledge-oriented economy. These intangible factors cannot be measured by currency.

(7) The EPS is reported, at the most, once every financial quarter. It is thus inadequate to lead to clues about daily stock prices.

4.1.2 Inadequacy of the P/E ratio

The P/E ratio is not as useful as people expect. The three types of P/E ratios mentioned earlier are not much different. Whereas the first type of P/E ratio uses real historical data, the second and the third are partially or totally based on estimates that are not always accurate.

Like EPS, the P/E ratio cannot be used solely. It is only a supplementary measure to obtain hints of a stock's future trend in the financial market.

Consider the following simple question: If the P/E ratio is higher, does that imply that the stock price trend will be higher? The answer is NO. Instead, a high P/E ratio may be excessive and could indicate an unreal circumstance. Instead, the high P/E may be also suggesting a high risk, implying that the price is abnormally high with

regard to the earnings.

In some circumstances, a high P/E ratio may even indicate an eve of plunging. In late 1999 and early 2000, the most famous leading Internet corporation, Yahoo, had its share price soar as high as 400 US dollars. As a result, its P/E ratio was much higher than the normal level, and even as high as 1000! It was definitely a surprise and shock to the analysts of Wall Street.

At that moment, the P/E ratios of other Internet corporations listed on NASDAQ were also all extremely high. The investors' attitudes to high tech companies could not be expressed in terms of confidence, but rather described to be fanatic and fetishistic. Later on, the Internet bubble burst and the stock prices of all Internet companies plunged dramatically. Yahoo's share price also fell all the way down.

In some conditions, the P/E ratio is more likely related to recent news and business performance rather than last year's EPS value.

The P/E is a simple indicator but very hard to interpret. In fact, in some circumstances, the P/E ratio is almost meaningless; it actually does not tell a lot by itself. It is useful only when it is compared with the corporation's own historical P/E ratio, with other companies in the same industry or with the overall market. Investors should probe into the facts and reasons hidden behind the ratio. If an investor believes that the P/E ratio is informative at all times, we believe that he is placing more value on the P/E ratio than what it really guarantees.

The following two figures (Figure 4.1 and 4.2) are selected from BHL's official web site [17]. They are the examples that demonstrate that the share price does not match with the P/E ratio.

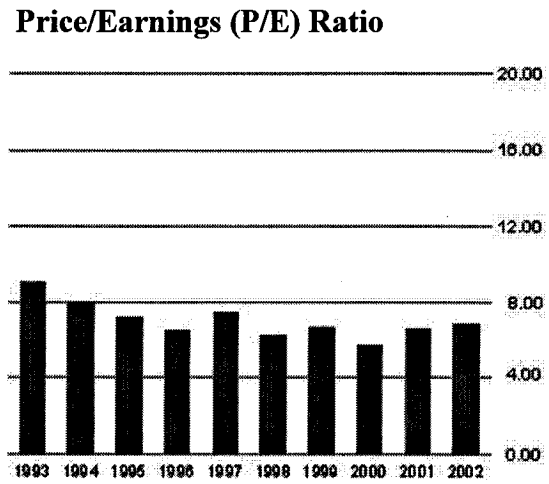

Figure 4.1 P/E Ratio of BELCO Holding Limited (BHL)

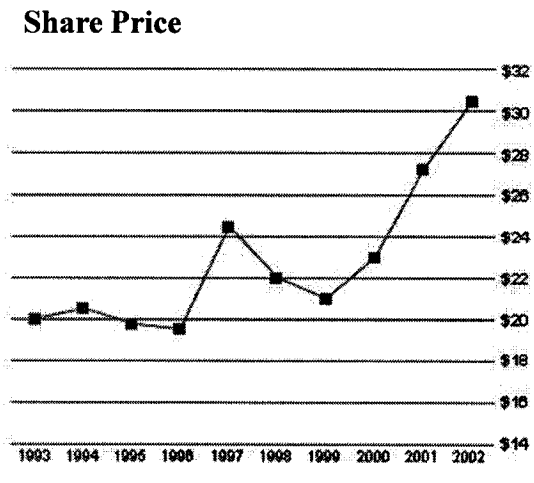

Figure 4.2 Share Price of BELCO Holding Limited (BHL)

4.1.3 Technical Analysis is not Efficient

An important assumption of a technical analysis is that all past prices and volumes stand for the perception of the complete market of what is known and knowable about the stocks. Therefore, historical price and volume movements are predictable in the future. They believe this trend will not end until something new occurs to change the trend, and the prices and market activities are predictable until the change happens.

A Moving Average indicator may tell the investors the general trend that the stock market will go toward. But economists have observed that the MA still has notable drawbacks, since it only works properly in a trending market. If the market is directionless and moving in a "whipsaw" path, the MA may not necessarily work in a good way, and even be meaningless in some extreme circumstances. The financial

market is so complex and random that a clear trend can seldom be found.

There are hundreds of methods which utilize a technical analysis developed by economists. It is hard, redundant, and usually not feasible to consider every technical analysis tool when trading in the stock market. Another "fatal defect" that renders technical analysis inadequate as a promising forecasting tool, is that different indicators may give out contrary results. Besides, the hint from a technical analysis could be delayed, and the actual reversal could happen before the associated alarm. Thus, in some way, a technical analysis could be subjective and intuitive. To different people, the same crossovers and indexes could have different meanings and suggestions. Based on the same level of the same index, one person may decide to buy, while another may decide to sell. The decision depends on the person's confidence and experience. Most technical analyses lead to some kind of transmutation of a price line, like the MA, EMA and MACD. Thus, in one sense, they are alternative ways to plot the price line and display it in different ways. Some traders say that a technical analysis contains nothing more than the price itself. Also, the information that both the fundamental and technical analysis provide is usually "after the event". In other words, this information could be delayed, and it could be that the real reversal happens before the signal.

4.1.4 Remarks

From the above, we conclude the following.

Security analysts have to study both fundamental and technical data. They must read all the information about the company and watch the stock movement in the market. They must use their own knowledge and experience together with the

research they have done to forecast the stock movement and trend, that both a fundamental and technical analysis do not reveal. Here the human element is the key factor, and cannot be measured or erased. It is hard to say the consequence of a decision is not occasional, even if the forecast is correct. For instance, in one case, in an investment experiment that was devised by Richard Wiseman, a psychologist from the University of Hertfordshire [30], a professional analyst was scarcely able to outperform the decisions of a five-year-old girl.

The experiment was organized by the British Association for the Advancement of Science as part of the National Science Week, an initiative backed by the United Kingdom Government and that aimed to celebrate science and its importance in our lives. In the experiment, a five-year-old girl, Tia Roberts along with Christeen Skinner, a financial astrologer, and Mark Goodson, an independent analyst "invested" 5,000 Sterling Pounds in a fantasy portfolio. One year later, the investment results were announced to compare different ways of choosing the best shares. Christeen Skinner studied the movement of the planets to choose her portfolio. Mark Goodson used his years of investment expertise and computer analyses of fundamental and technical indicators to pick his. And Tia just chose her shares at random. One year later - and at the start of National Science Week 2002 - the results were reported as follows: Tia Roberts: up 5.8 per cent. Christeen Skinner: down 6.2 per cent. Mark Goodson: down 46 per cent. Tia from London not only outperformed her fellow experiment participants, but even defied a 16 per cent drop in the FTSE 100 (the Financial Times index of the 100 top-performing companies on the London Stock Exchange) with her randomly selected choice of shares. [30]

The nature of the traditional analyses contains the risk that they would not be precise or reliable, even to merely forecast trend. Therefore, people have to look for some brand new methods to help traders buy and sell stocks. Some have moved their

focus to neural networks, the combination of the most rapidly progressing technology, computer science, and the most complex part of lives, the brain.

4.2 Artificial Neural Network (ANN) Forecasting

The development of computer science has demonstrated the fastest progress in this century. It has been only 15 years from the release of the 80386 DX on October 17, 1985 to the release of AMD Athlon 1GHz on March 6, 2000 [31]. The supercomputer, Earth Simulator, made by NEC in 2002, computed over 35 teraflop/second operations using 5104 processors. It was used for research in Earth related sciences, especially suited for simulating complex linked systems, such as in forecasting the weather [32]. The rapid progress of supercomputers and parallel computing technology has become the foundation of neural networks, and has boosted its development.

Neural networks work in a manner analogous to the biological brain, which is an extremely complex parallel system.

The brain is not a uniform mass. It has features, some of which are a centimeter large, some a millimeter and others much smaller. It consists of a large quantity of nerve cells, also known as neurons. A neuron is a well-defined tiny region of the brain, sending (transmitting) and receiving electro-chemical signals to and from the brain and nervous system. There are about 100 billion neurons in the brain, separated by walls. The walls have a chemical composition different from the interior of the cell. There are many types of neurons. They vary in size from 4 microns (.004 mm) to 100 microns (.1 mm) in diameter. Neurons have very irregular forms. They are far from spherical as shown in Figure 4.3 [34].

The neuron consists of a *cell body* (or *soma*) with branching *dendrites* (signal receivers) and a projection called an *axon*, which conducts the nerve signal. At the other end of the axon, the *axon terminals* transmit the electro-chemical signal across the gap between the axon terminal and the receiving cells. The *axon*, a long extension of a nerve cell, takes information away from the cell body. Dendrites bring information to the cell body. *Myelin* coats and insulates the axon, increasing transmission speed along the axon. Myelin consists of 70-80% lipids (fat) and 20-30% protein. The *cell body* (*soma*) contains the neuron's nucleus (with DNA and typical nuclear organelles). Dendrites branch from the cell body and receive messages. A typical neuron has about 1,000 to 10,000 synapses (that is, it communicates with 1,000-10,000 other neurons, muscle cells, glands, etc.). [35] [36]

Each neuron has physical connections with dozens of thousands of others, and thus a very complicated intercommunicating network is built up. The connections are not only on or off, to transmit through electrical signals among neurons, but also have different strength allowing strong or weak signals to pass through. Many functions of the brain, especially learning ability, result from the varying connection strengths. The simultaneous cooperation of a large number of simple neurons with different strength connections represents all the complex brain activities. Figure 4.4 demonstrates how signals transmit among neurons. The real biological activity is, of course, much more complex. [35] [36]

An Artificial Neural Network (ANN) simulates the human brain in the way that the neurons process information. Neurons of the biological brain are prototyped by nodes, the basic components of ANN. Every node is a processing entity that can be programmed to release a signal when its inputs reach a threshold. The nodes mathematically represent the modeling of the functionality of the brain and the programmed connections imitate the dendrites linking neurons in the biological system.

Different weights are programmed to be assigned to the connections between inputs and neurons, and thus some connections allow a large signal to pass through while others only allow a weak signal, which is an analogy to the biological phenomenon of synapses varying in strength. In the real brain, the synapses can be excitatory to add to the received signal from dendrites or inhibitory to reduce the received signal. To imitate this, the connections are also programmed to be positive or negative.

The ANN is a multiprocessor computer system whose architecture is modeled after the brain. All the computers in the network are connected to constitute a very complex intercommunication network, whose structure is different from traditional computers. Each unit in this special network is an analogy to the real neuron which is triggered to send a new signal upon receiving a strong enough signal from other neurons. The weight, which is an analogy to the connection strength between the neurons, can be adjusted to perform different tasks.

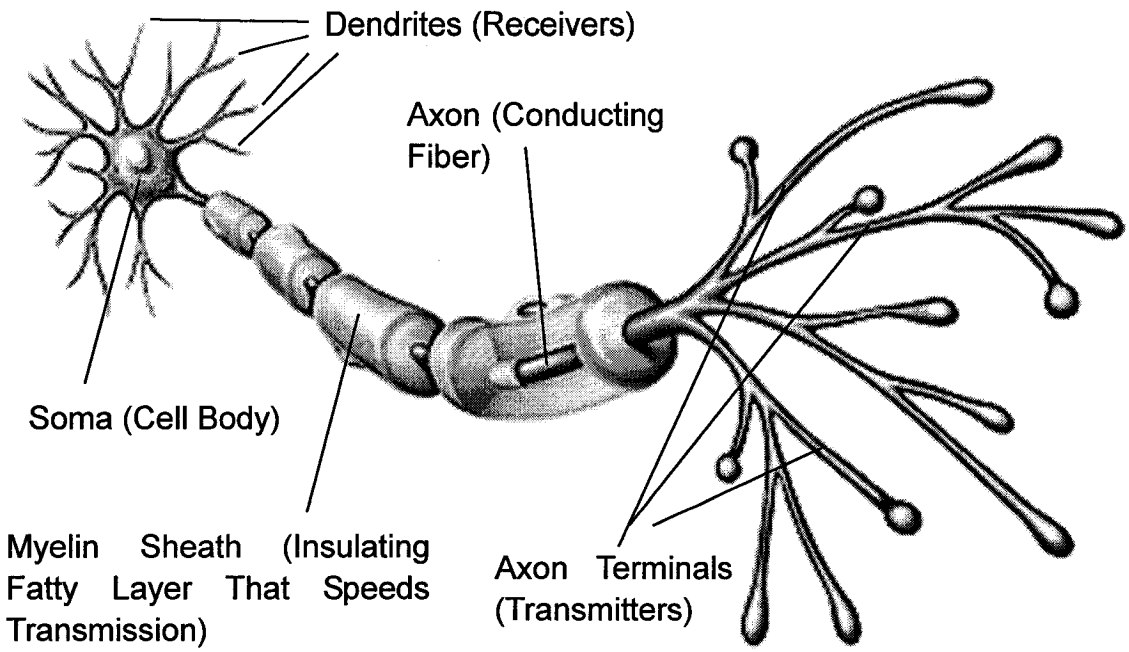

Figure 4.3 Neuron Structure

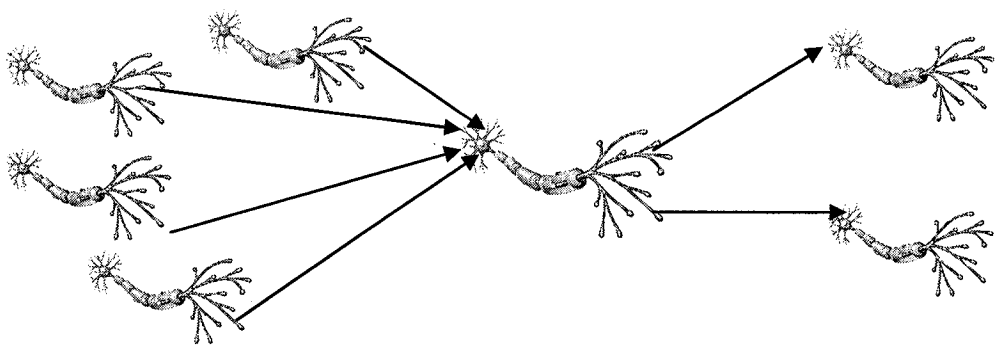

Figure 4.4 Spike Transmission Among Neurons

4.3 Neural Based Modeling: Problem Statement

Let $\boldsymbol{x}$ be an N_x-vector containing input parameters of a given device. Let $\boldsymbol{y}$ be an N_y-vector containing the responses of the device under consideration.

The relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$ may be multidimensional and nonlinear, and represented by

$$y = f(x) \tag{4-1}$$

Such a relationship can be modeled by a neural network, by training it through a set of input-output sample pairs given by,

$$\{(x_p, d_p),\ p=1,2,.....,N_p\} \tag{4-2}$$

where $\boldsymbol{x_p}$ and $\boldsymbol{d_p}$ are N_x- and N_y-dimensional vectors representing the inputs and the desired outputs of the neural network respectively. Let the neural network model of this relationship be represented by,

$$y = y(x, w) \tag{4-3}$$

where $\boldsymbol{w}$ is the vector (or set of) parameters of the neural network model, which is also called the weight vector in the literature. The definition of $\boldsymbol{w}$, and how $\boldsymbol{y}$ is computed through $\boldsymbol{x}$ and $\boldsymbol{w}$, determine the structure of the neural network.

The training problem of a neural network is to minimize the difference between the neural model outputs and desired outputs as:

$$E(w)=\frac{1}{2}\sum_{p=1}^{N_p}\sum_{k=1}^{N_y}\left(y_{pk}(x_p,w)-d_{pk}\right)^2 \qquad (4\text{-}4)$$

where d_{pk} is the k^{th} element of vector $\boldsymbol{d}_p$, $y_{pk}(\boldsymbol{x}_p,\boldsymbol{w})$ is the k^{th} output of the neural network when the input presented to the network is $\boldsymbol{x}_p$. Once the relationship is grasped by a neural network model, it can be used for predicting the output values given only the values of the input variables [37], [38].

4.4 Review of Neural Network Structures

The most commonly used neural network structure is the multilayer perceptrons (MLP). Other basic structures include the radial basis function (RBF) network, wavelet neural network, and self organizing maps (SOM). Other advanced structures as knowledge-based neutral network (KBNN) and recurrent neural networks have been also developed by researchers [37], [38].

4.4.1 Multilayer Perceptrons

Typically, the MLP network consists of an input layer, one or more hidden layers and an output layer, as shown in Figure 4.5 [39].

Suppose the total number of hidden layers is *L*. The input layer is considered as

hidden layer 0. Let the number of neurons in the hidden layer l be N_l, l = 1, 2, ..., L. Let $w_{ij}{}^{l}$ represent the weight of the link between j^{th} neuron of l-1^{th} hidden layer and i^{th} neuron of l^{th} hidden layer, and θ_i^l be the bias parameter of i^{th} neuron of l^{th} hidden layer. In fact, a weight parameter called bias could be added to improve the modeling. The effect of adding the bias is that the weighted sum is equal to the bias when all the previous hidden layer neuron responses are zero. Let x_i represent the i^{th} input parameter to the MLP. Let $\bar{y}_i{}^{l}$ be the output of the i^{th} neuron of the l^{th} hidden layer, which can be computed according to the standard MLP formulae as [37],

$$\bar{y}_i{}^{l} = \sigma\left(\sum_{j=1}^{N_{l-1}} w_{ij}{}^{l}\ \bar{y}_j{}^{l-1} + \theta_i{}^{l}\right), \quad i = 1,\ldots N_l, \quad l = 1,\ldots L \tag{4-5}$$

$$\bar{y}_i{}^{0} = x_i, \quad i = 1,\ldots,N_x, \quad N_x = N_0 \tag{4-6}$$

where σ is usually a monotone function. Let q_{ki} represent the weight of the link between i^{th} neuron of L^{th} hidden layer and k^{th} neuron of output layer, and φ_k be the bias parameter of k^{th} output neuron.

The outputs of MLP can be computed as,

$$y_k = \sum_{i=1}^{N_L} q_{ki}\ \bar{y}_i{}^{L} + \varphi_k, \quad k = 1,\ldots,N_y \tag{4-7}$$

For function approximation, the output neurons are usually linear neurons. The most commonly used σ is the logistic sigmoid function given by,

$$\sigma(t) = \frac{1}{(1+e^{-t})} \tag{4-8}$$

$$\sigma(t) \longrightarrow \begin{cases} 1 & \text{as} \quad t \to +\infty \\ 0 & \text{as} \quad t \to -\infty \end{cases} \tag{4-9}$$

Other possible candidates for σ are the arctangent function given by,

$$\sigma(t)=\left(\frac{2}{\pi}\right)*\arctan(t), \tag{4-10}$$

and the hyperbolic tangent function given by,

$$\sigma(t)=\frac{(e^{t}-e^{-t})}{(e^{t}+e^{-t})} \tag{4-11}$$

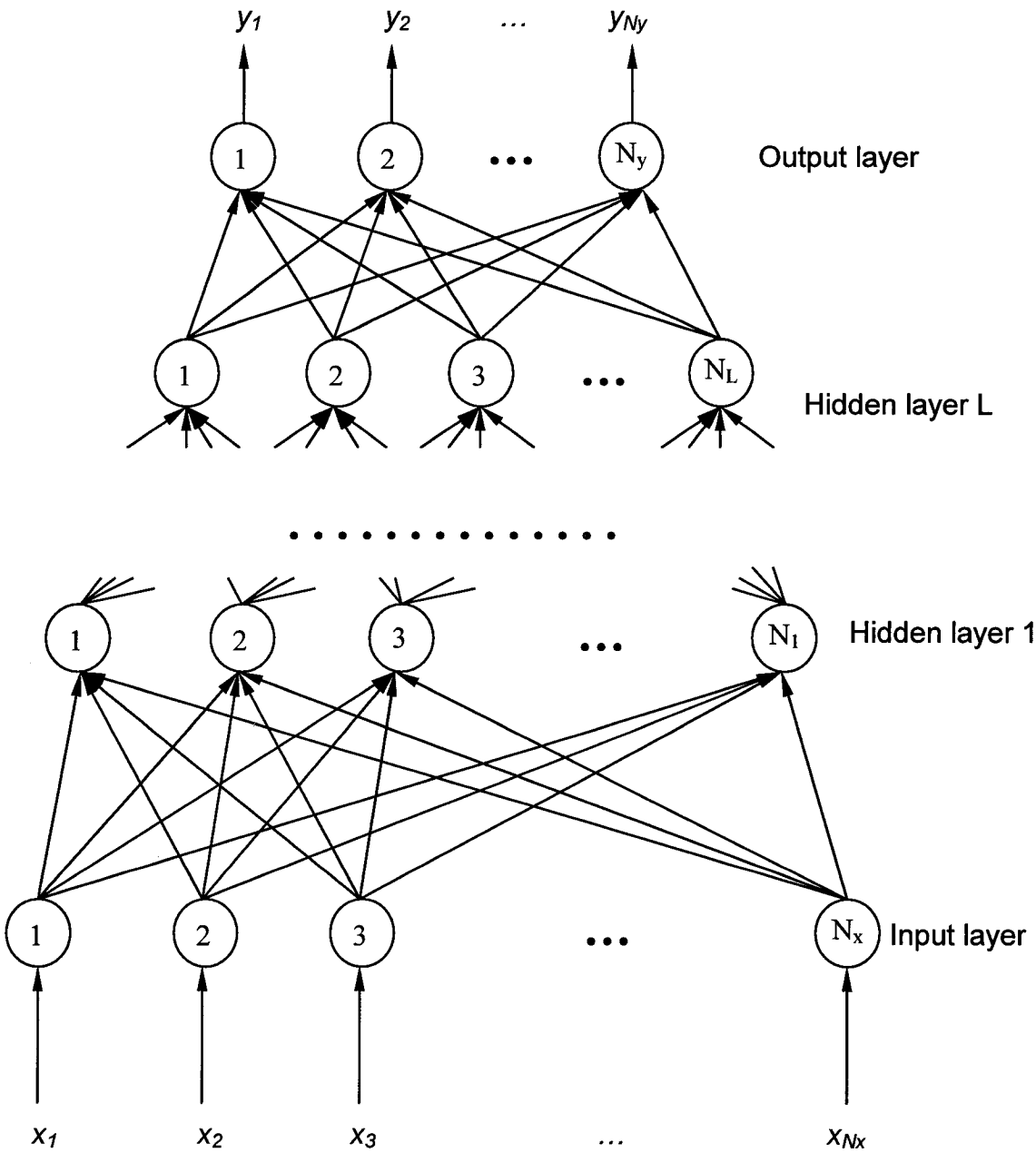

Figure 4.5 Multilayer Perceptron (MLP) Structure (Typically, the network consists of an input layer, one or more hidden layers and an output layer.)

All these functions are bounded, continuous, monotonic and continuously differentiable. Training for the parameters w includes computing the pertinent values of w as:

$$\begin{aligned} w=[\ & w_{ij}^{l} \quad j=1,\ldots,N_{l-1},\ i=1,\ldots,N_l,\ l=1,\ldots,L; \\ & \theta_i^{l} \quad i=1,\ldots,N_l,\ l=1,\ldots,L; \\ & q_{ki} \quad i=1,\ldots,N_L,\ k=1,\ldots,N_y; \\ & \varphi_k \quad k=1,\ldots,N_y\ ] \end{aligned} \tag{4-12}$$

It is well known that a two-layered MLP (no hidden layers), is not capable of approximating generic nonlinear continuous functions [37]-[40]. The universal approximation theorem [41], [42], [43], states that a 3-layer perceptron, with one hidden sigmoidal layer, is capable of modeling any function to any desired degree of accuracy, provided sufficiently many hidden neurons are available.

Feedforward neural networks which have only one hidden layer, and which use radial basis activation functions in the hidden layer, are called Radial Basis Function (RBF) networks. Radial basis functions are derived from the regularization theory in the approximation of multivariate functions [44]. Park and Sandberg showed that RBF networks also have universal approximation ability [45]. The output neurons of RBF networks are also linear neurons. The overall input-output transfer function of RBF networks is defined as,

$$y_j = \sum_{i=1}^{N_1} v_{ji}\sigma(\| x-\theta_i \|), \qquad j=1,\ldots,N_y \tag{4-13}$$

where θ_i is the center of the radial basis function of the i^{th} hidden neuron, v_{ji} is the

weight of the link from the i^{th} hidden neuron to the j^{th} output neuron. Some of the commonly used radial basis activation functions are [46],

$$\sigma(t) = \exp\left[-\left(\frac{t}{\lambda}\right)^2\right] \tag{4-14}$$

$$\sigma(t) = (c^2 + t^2)^{\beta}, \quad 0 < \beta < 1 \tag{4-15}$$

where (4-14) is the Gaussian function, (4-15) is the multiquadratic function and, λ, c and β are the function parameters. Training parameters $\boldsymbol{w}$ includes $\theta_i, v_{ji}, i=1,.....,N_1, j=1,.....,N_y$ and λ or c and β. Although MLP and RBF are both feedforward neural networks, their activation functions are quite different. Moreover, MLP networks construct global approximations to nonlinear input-output mapping. Consequently, they are capable of generalizing in those regions of the input space where little or no training data is available, which is an essential point for our work. On the contrary, RBF networks use exponentially decaying localized nonlinearities (e.g., Gaussian functions) to construct local approximations to nonlinear input-output mapping [38], [47].

4.4.2 Neural Network Structures with Prior Knowledge

For MLP and RBF, the entire information about the given problem comes from training data. Consequently, large amounts of training data are usually needed to ensure model accuracy. In financial applications, obtaining a relatively large set of training data is quite difficult. Therefore, using prior knowledge during the neural model development process could be the best choice to avoid such limitations.

In Prior Knowledge Input (PKI) structures [48], the source model outputs are used as inputs for the neural network model in addition to the original problem inputs. As such, the input/output mapping that must be learnt by the NN is the one involved between the output response of the source model and that of the target model. In the extreme case, where the target outputs are the same as the source model outputs, the learning problem is reduced to a one-to-one mapping.

Unfortunately, this approach uses symbolic knowledge in the form of rules to establish the structure and the weights of the neural network, making it quite suitable to pattern recognition area but not adequate for financial fields where the important problem knowledge is more functional than symbolic/structural.

4.4.3 Knowledge-Based NNs

In [49], a Knowledge-Based Neural Network (KBNN) was introduced. In this method the knowledge in the form of empirical functions or analytical approximations is embedded into the structure of the NN. This structure was inspired from the fact that practical empirical functions are usually valid only in a certain region of the parameter space. To build a neural model for the entire space, several empirical formulae, and the mechanism to switch between them are needed. The KBNN structure is a non-fully connected structure as shown in Figure 4.6.

There are 6 layers in the structure, namely the input layer $\boldsymbol{X}$, the knowledge layer $\boldsymbol{Z}$, the boundary layer $\boldsymbol{B}$, the region layer $\boldsymbol{R}$, the normalized region layer $\boldsymbol{R'}$ and the output layer $\boldsymbol{Y}$ [37], [47].

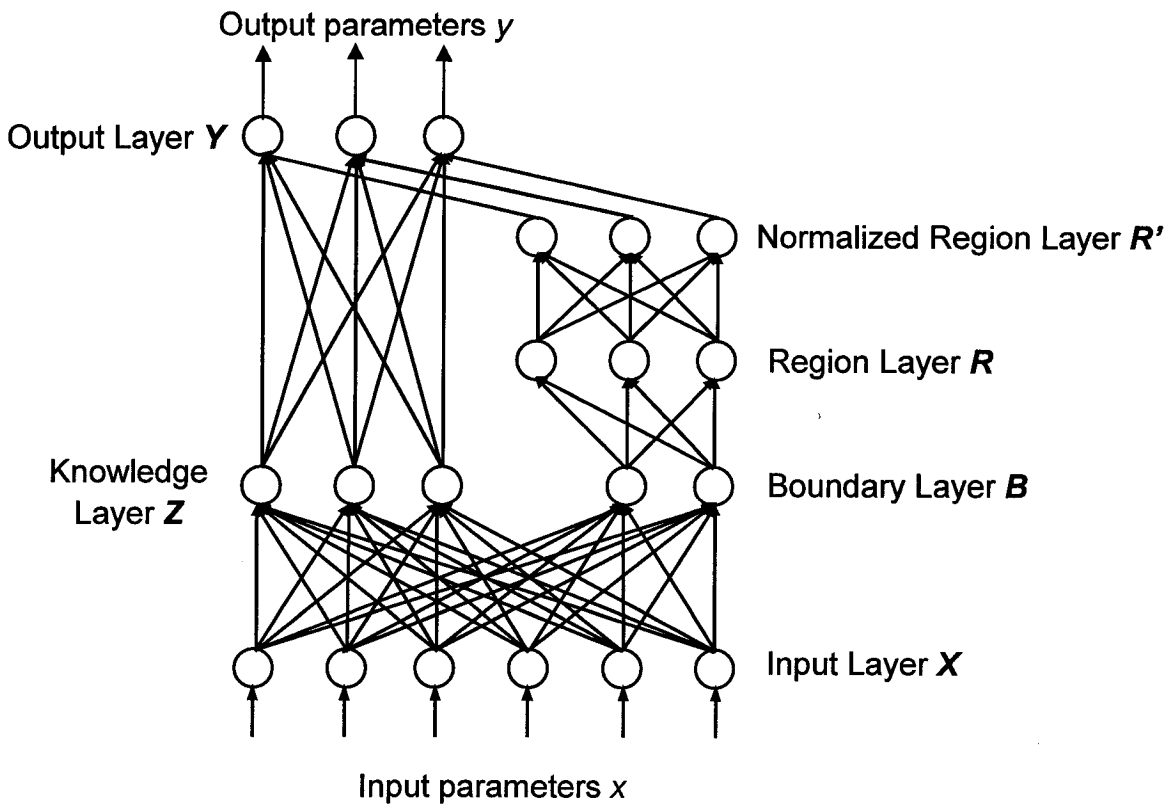

Figure 4.6 Illustration of the structure of Knowledge Based Neural Networks (KBNN) (The KBNN model typically includes six layers.)

The input layer ***X*** accepts parameters ***x*** from outside the model. The knowledge layer ***Z*** contains the knowledge function, $\psi()$. The output of the i^{th} knowledge neuron is given by,

$$z_i = \psi_i(x, w_i), \quad i = 1,2,\ldots,N_z \tag{4-16}$$

where ***x*** is a vector including the NNs inputs, and $\boldsymbol{w_i}$ is a vector of parameters in the knowledge formula. The boundary layer ***B*** can incorporate knowledge in the form of

problem dependent boundary functions ***B***(). In the absence of boundary knowledge, ***B*** incorporates knowledge just as linear boundaries. Output of the i^{th} neuron is calculated by,

$$b_i = B_i(x, v_i), \quad i = 1,2,\ldots,N_b \tag{4-17}$$

where $\boldsymbol{v_i}$ is a vector of parameters in $\boldsymbol{B_i}$ defining an open or closed boundary in the input space ***x***. The region layer ***R*** contains neurons to construct regions from boundary neurons,

$$r_i = \prod_{j=1}^{N_b} \sigma(\alpha_{ij} b_j + \theta_{ij}), \quad i = 1,2,\ldots,N_r \tag{4-18}$$

where α_{ij} and θ_{ij} are the scaling and bias parameters, respectively. The normalized region layer ***R'*** contains rational function-based neurons to normalize the outputs of ***R***

$$r'_i = \frac{r_i}{\sum_{j=1}^{N_r} r_j}, \quad i = 1,2,\ldots,N_{r'}, \quad N_{r'} = N_r \tag{4-19}$$

The output layer $\boldsymbol{Y}$ contains second order neurons combining knowledge neurons and normalized region neurons,

$$y_j = \sum_{i=1}^{N_z} \beta_{ji} z_i \left(\sum_{k=1}^{N_{r'}} \rho_{jik} r'_k \right) + \beta_{j0}, \quad j = 1,2,\ldots,N_y \tag{4-20}$$

where β_{ij} reflects the contribution of the i^{th} knowledge neuron to output neuron y_j, and β_{jo} is the bias parameter.

ρ_{jik} is 1 indicating that region r'_k is the effective region of the i^{th} knowledge neuron contributing to the j^{th} output. A total of $N_{r'}$ regions are shared by all the output neurons. Training parameters w includes,

$$\begin{aligned} w=[&w_i \ i=1,.....,N_z; \\ &v_i \ i=1,.....,N_b; \\ &\alpha_{ij},\theta_{ij} \ i=1,.....,N_r, \ j=1,.....,N_b; \\ &\beta_{ji} \ j=1,.....,N_y, \ i=0,.....,N_z; \\ &\rho_{jik} \ j=1,.....,N_y, \ i=1,.....,N_z, \ k=1,.....,N_{r'}] \end{aligned} \tag{4-21}$$

The knowledge function is usually in the form of empirical or semi-analytical functions. Unfortunately, such formulae are not available in our case. Therefore, the use of KBNN in stock price forecasting is still open.

4.4.4 Other Neural Network Structures

The idea of combining wavelet theory with neural networks resulted in a new type of NNs called wavelet networks [37]. Though wavelet theory has offered efficient algorithms for various purposes, their implementation is usually limited to wavelets of small dimension. It is known that NNs are powerful tools for handling problems of large dimension. Combining wavelets and NNs can hopefully remedy the weakness of each other, resulting in networks with efficient constructive methods that are capable of handling problems of moderately large dimension [37], [47]. But since this is not the primary focus of this thesis, we shall not

elaborate on these structures any further.

4.5 Why Use BP Algorithm To Forecast Stock Market?

The BP algorithm is the most often chosen NN algorithm wherever the following conditions hold: [52]

(1) It is not possible to discover a formula relating the phenomena and the underlying variables causing them;

(2) There are a lot of data examples available that reveal the relationship between the phenomena and the underlying variables;

(3) The user is required to "generalize" beyond the training data.

Another four reasons justifying the use of the BP algorithm for forecasting stock price are:

(1) The BP algorithm has the proven ability to approximate any function;

(2) The stock market has some repeated properties and rules hidden behind of the data, which reflect mathematical relationship with the market;

(3) The BP has excellent fault tolerance properties;

(4) The BP algorithm is easy to implement.

To predict stock price trend, many people study the data in the market, and only the data in the market. Is it reasonable? Is this data adequate to predict the future? We

believe that it is. Although many kinds of information have impact on the market, all of them are reflected in stock market. We believe that every piece of information known by people and affecting people's activity is reflected in the market, as the saying goes: "the market itself is efficient enough". One can wonder: "What if some information is not known by investors?" We argue that in this case, it does not affect people's activity to invest, and thus, has no impact on the market. To predict the future, studying the market itself is enough. Our algorithm is based on scrutinizing the information available on the market, and use the BP algorithm to find the rules hidden behind the pricing mechanism.

4.6 The BP Algorithm Revisited

The research team led by Rumelhart and McCelland proposed the detailed Back Propagation (BP) scheme in the book, *Parallel Distributed Processing*, in 1986. In this book, they analyzed the error of the BP multi-layer NN with a non-linear continuous transfer function. The purpose of BP algorithm is to adjust the network weights so that the network produces the desired output in response to every input in a predetermined set of training patterns. [53]

The BP training process begins with a set of random weights assigned to the connections of the input vector, and the layer directly receiving the input. The input vector is fed forward through the BP network. At the end of every epoch, the output vector is calculated using the weights and compared with the desired output vector. The BP network error is defined as the sum of the Euclidean distance in the weight space for the difference between the true output and the desired output vectors. The purpose of the BP network is to minimize the Euclidean distance by adjusting the set of

weights at every epoch. When the network error falls within the pre-defined error bound, the BP training process stops.

The typical BP network basically has three layers: input, output and one hidden layer. Each layer consists of a number of processing nodes, which are analogous to biological neurons. There are programmed connections between each pair of layers and weights assigned to the connections. Usually the transfer function is a sigmoid function. Every neuron receives the weighted sum as its inputs from its previous layer, and generates an output based on the weighted sum, which, in turn, passes through to the next layer.

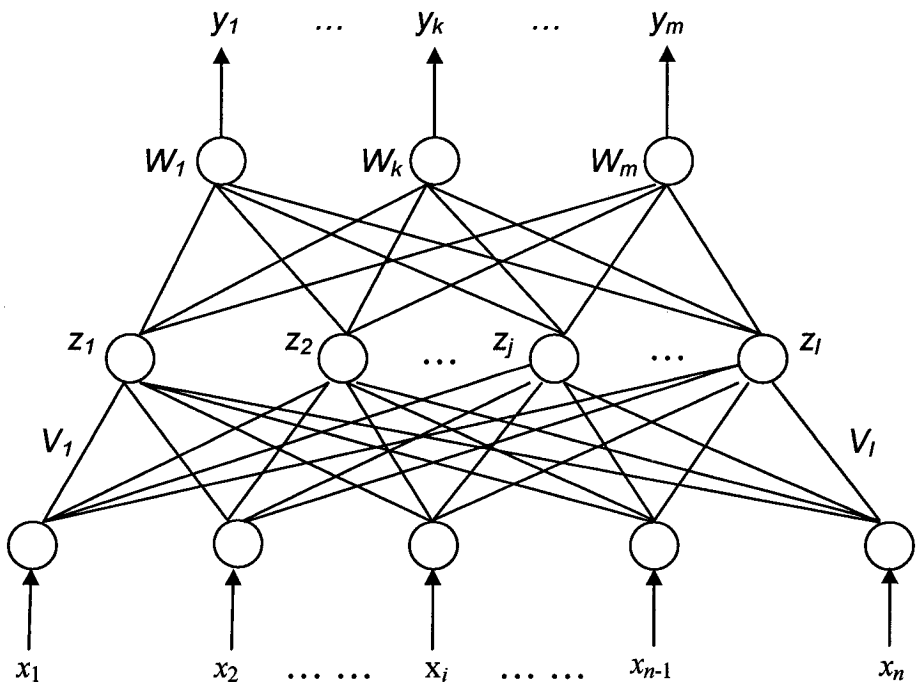

Figure 4.7 The Three-Layer BP.

In the three-layered feed-forward network, as shown in Figure 4.7, the *n*-input vector is $X = [x_1, \ldots, x_i, \ldots, x_n]^T$. An element $x_0 = -1$ can be added as the bias of the

hidden layer. The l-output vector of the hidden layer is $Z = [z_1, \ldots, z_j, \ldots, z_l]^T$. An element $z_0 = -1$ can be added as the bias of the output layer.

The m-output vector of the output layer is $Y = [y_1, \ldots, y_k, \ldots, y_m]^T$. The desired output vector is $D = [d_1, \ldots, d_k, \ldots, d_m]^T$. The weight matrix connecting the input layer and the hidden layer is $V = [V_1, \ldots, V_i, \ldots, V_l]^T$, where v_j represents the weight vector of the j^{th} neuron in the hidden layer. The weight matrix connecting the hidden layer and the output layer is $W = [W_1, \ldots, W_k, \ldots, W_m]^T$, where W_k the weight vector of the k^{th} neuron in the output layer. [54]

Let

$$y_k = f(net_k) \tag{4-23}$$

where $k = 1, 2, \cdots, m$, and

$$net_k = \sum_{j=0}^{l} w_{jk} z_j \tag{4-24}$$

Where, in turn, for $j = 1, 2, \cdots, l$

$z_j = f(net_j)$, and (4-25)

$$net_j = \sum_{i=0}^{n} v_{ij} x_i \tag{4-26}$$

In formulae (4-23) and (4-25), the transfer function $f(x)$ is the sigmoidal function featured by:

$$f'(x) = f(x)[1 - f(x)] \tag{4-27}$$

The formulae from (4-23) to (4-27) constitute the mathematical model of the three-layer feed-forward network.

The error ε exists when the actual observed output does not match the desired output. It is defined as:

$$\varepsilon = \frac{1}{2}(D - Y)^2 = \frac{1}{2}\sum_{k=1}^{m}(d_k - y_k)^2 \tag{4-28}$$

If we apply function (4-28) to the hidden layer, then:

$$\varepsilon = \frac{1}{2}\sum_{k=1}^{m}[(d_k - f(net_k)]^2 = \frac{1}{2}\sum_{k=1}^{m}[(d_k - f(\sum_{j=0}^{l} w_{jk} z_j)]^2 \tag{4-29}$$

If we apply this to the input layer, we obtain:

$$\varepsilon = \frac{1}{2}\sum_{k=1}^{m}\{d_k - f[\sum_{j=0}^{l} w_{jk} f(net_j)]\}^2 = \frac{1}{2}\sum_{k=1}^{m}\{d_k - f[\sum_{j=0}^{l} w_{jk} f(\sum_{i=0}^{n} v_{ij} x_i)]\}^2 \tag{4-30}$$

According to (4-30), the error ε is the function of w_{jk} and v_{ij}, and so the error ε can be decreased by adjusting the weights. To decrease the error ε, the weight adjustment should follow the function:

$$\Delta w_{jk} = -\eta \frac{\partial \varepsilon}{\partial w_{jk}} \tag{4-31}$$

where $j = 0,1,2,\cdots,l$; $k = 1,2,\cdots,m$.

$$\Delta v_{ij} = -\eta \frac{\partial \varepsilon}{\partial v_{ij}} \tag{4-32}$$

where $i = 0,1,2,\cdots,n$; $j = 1,2,\cdots,l$. The negative sign represents a gradient descent, and η is learning rate, where $\eta \in (0,1)$. In every epoch, the weights are adjusted according to (4-31) and (4-32), until the error ε becomes acceptably small. The pseudocode of BP algorithm is:

for $l = 1$ to *L* (where *L* is the number of layers), do

{initialize $W^{(l)}$};

define *maximum epochs*;

let *ME*= *maximum epochs*;

define ε as the allowed error;

E=ε+1;

while ($ME > 0$) do

{while ($E > \varepsilon$) do

{*E*=0;

for every pair of (x_i, y_i) in the set, do

{calculate the desired output O_i using x_i;

calculate E_i;

$E = E + E_i$;

adjust weights $W^{(L)}$;

$l = L - 1$;

while ($l \neq 0$) do

{adjust weights $W^{(l)}$;

$l = l - 1$;

}

}

}

$ME = ME - 1$;

}

$E = E/2$;

4.7 How To Select the Appropriate Neural Network?

Selection of the most adequate NN for a specific problem is not a straightforward task, since it depends on several different criteria such as the prior knowledge that the user has, the nonlinearity of the problem, the range of the input parameter space, the number of training data, etc. However, based on the literature review and the available data information that we have, we believe that the MLP structure is the most suitable one to generate the neural model. [37]

4.8 Number of Hidden Layers

The universal approximation theorem asserts that a three-layer MLP network,

with bias, a single sigmoid function hidden layer, and a linear function output layer, can approximate any nonlinear function. But the theorem does not say how many neurons in the hidden layer have to be considered to best match a given problem, and which algorithm is the most suitable for our work. Most of MLP networks have three layers, especially for the problem of function approximation and regression area. More layers will make the network much more complex, and consequently make the convergence much slower. Although more layers might increase the accuracy and decrease the error, we have noticed that we could achieve the same results by adding additional neurons to the network rather than adding layers.

Therefore, in order to increase the network performance, adjustment of the number of neurons is always the first choice. As a general observation, a three-layer MLP network with 20 neurons in the hidden layer might outperform a four-layer MLP with 10 neurons in each hidden layer, though they both have 20 neurons totally. The former is not only easier to adjust and monitor (with regard to the performance), but also generally speaking, converges much faster. In order to build a reliable and high performance model, this thesis would add neurons in each layer to achieve more precise results rather than add layers. Thus, this thesis utilizes three-layer NN structures.

4.9 Number of Neurons in the Hidden Layer

It is observed that highly nonlinear problems need more neurons than smooth problems. But unfortunately, there are no results available that propose a closed-form formula to choose the right number of neurons for a given problem. Too few neurons

give the network an insufficient ability to learn, and so it requires more epochs to train. The worst thing is that a network with an insufficient number of neurons has difficulty to meet the performance criteria, implying that it cannot approximate accurately. But too many neurons can also cause problems. The network will need more computational time for each epoch, and also overfits the data. The only way to make decision on this, is to use one's personal experience, and by repeated experimentation. A general method is to choose the number of neurons that satisfy the problem's criteria, and then to add a few more neurons to optimize the convergence speed. Based on our experience, it seems reasonable that the number of the neurons should be between 10 and 50.

4.10 Architecture Design of MLP

After deciding on the number of neurons and layers, in this thesis, we intend to proceed to also consider the detailed topology, the way the network has to be trained, and the information being transferred.

In this thesis, we have implemented three BP supervised algorithms: the feed-forward, the cascade-forward and the Elman BP algorithms. A literature review will give us the ability to investigate some of the most widely-used training algorithms in the field of NNs, and this has permitted us to come up with seven different revised methods to train a neural model: the Levenberg-Marquardt BP, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton BP, the Resilient Propagation (RPROP) BP, the gradient descent BP, the gradient descent with momentum BP, the gradient descent with adaptive learning rate BP, and the gradient

descent with momentum and adaptive learning rate BP. This thus leads us to 21 different combinations. This thesis will compare the 21 BP algorithms one by one, and recommended one scheme as the best which can be used to forecast the stock market.

4.10.1 Introductions to the Three Supervised BP Algorithms

4.10.1.1 Introduction to Feed-Forward BP Algorithm

The feed-forward network is a supervised network. It is commonly used for prediction, pattern recognition, and nonlinear function fitting. It has one-way connections from the input to the output layers. The connections between units do not form cycles. Typically, every unit is only connected to units in the next layer, as shown in Figure 4.8. Every unit sums its inputs and adds the bias to form a total input x_j, and applies a function f_j to x_j to yield output y_j. The links have weights w_{ij} which multiply the signals being transferred along them [55]. The weights of the first layer come from the input, and the weights of every subsequent layer come from the previous layer. The output is the last layer. Every layer of the feed-forward network has biases. Consequently, the feed-forward network represent by Figure 4.8 can be described as the function:

$$y_k = f_k\left(\beta_k + \sum_{j\to k} w_{jk} f_j\left(\beta_j + \sum_{i\to j} w_{ij} x_i\right)\right) \qquad (4\text{-}33)$$

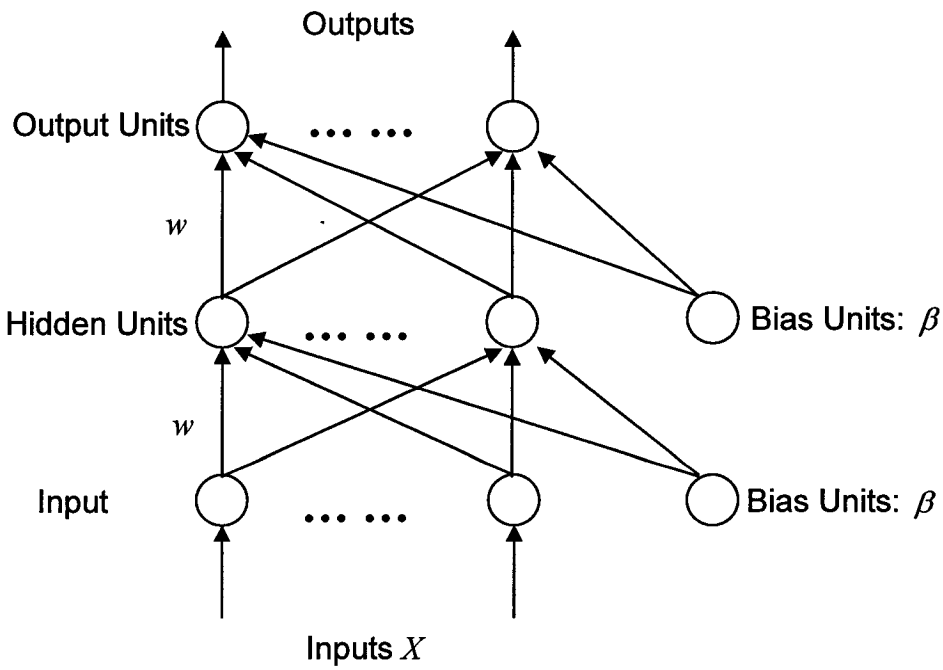

Figure 4.8 Feed-forward Network Structure

Usually, the bias can be considered as an input with value β, weighted by 1. Thus the bias can be omitted in the structure diagram since it can be regarded as a normal unit whose weight value is unity.

4.10.1.2 Introduction to Cascade-Forward BP Algorithm

Cascade Backpropagation (CBP) uses the same connection strength adjustment strategy and learning algorithm as BP. The major difference between CBP and BP is [56]:

Unlike BP, which chooses a fixed number of hidden units before one starts the training phase, CBP begins with a network with no hidden layer. It adds one hidden

unit at a time.

When CBP starts, the inputs are connected directly to the outputs. CBP stops training when the error of the network has converged. If the error is small enough, the network stops. Otherwise the network freezes the current connections between the input layer and the output layer. Then it adds one more hidden unit and fully connects it to the input layer and to the output layer. The network is trained again until the error has converged. Then CBP freezes the connections going to and coming from the new unit, and add a second hidden unit. This unit is also fully connected to the input layer and the output layer, and all previously added hidden units (in this case, only the first hidden unit). Thereafter it is trained again. This routine is repeated until a new unit cannot decrease the error. The new unit can be retried, with different randomly-initialized connections and for a pre-determined number of attempts, before the CBP training stops.

Every time CBP is trained, it selects its weights so as to maximize C, the magnitude of the covariance of the new unit's response, R, with the residual error [53] measured as:

$$C = \sum_{o} \left| \sum_{p} (R_p - \bar{R})(E_{p,o} - \overline{E_o}) \right|, \tag{4-34}$$

where o refers to the output units, and p to the training patterns. $\bar{R}$ and $\overline{E_o}$ are the averages of R and Eo over all the patterns. C can be maximized iteratively by a gradient ascent, strategy, where the derivative of C with respect to the i^{th} weight is:

$$\frac{\partial C}{\partial w_i} = \sum_{p,o} \sigma_o (E_{p,o} - \overline{E_o}) f'_p I_{i,p} \tag{4-35}$$

In the above, σ_o is the sign of the correlation between the candidate's value and the output o, f' is the derivative of the candidate unit's activation function with respect to the sum of its inputs for patterns p, and $I_{i,p}$ is the input the unit receives from unit i for pattern p. When C stops improving, the new unit is added to the network and its input weights are frozen. [53]

The advantage of CBP over BP is that CBP can automatically configure the number of hidden neurons instead of choosing the number by trial or experience. Thus CBP can take full advantage of the network's storage capacity.

In this respect, CBP is less 'greedy' than BP, as it has no upper bound on the number of hidden units. The representations in the hidden layer of the CBP network may still be distributed and compact, and CBP's learning may be comparably slow (although training within CBP is always done on only one hidden unit at a time), whereas BP always trains all the connections in the network.

4.10.1.3 Introduction to Elman BP Algorithm

As shown in Figure 4.9, the Elman network has a multi-layer structure, which has not only an input, output and hidden layer, but also an extra layer called the context layer or state layer, which receives feedback from the normal hidden layer. The outputs of the neurons in the context layer are fed forward to the hidden layer.

The context units act as one-step time delays and memorize the previous activations of the hidden units. The feedforward connections (represented by black solid line in Figure 4.9) are modified after each epoch. The recurrent connections (represented by blue dashed line) are fixed. At a given time t, the inputs are the previous activation of the hidden units (at time t-1) and the current

inputs (at time t). These inputs are propagated forward to generate the output. The BP algorithm is then employed to train the network. When the training stops, the activation of the hidden units at time t are sent back through the recurrent links to the context units and stored there for the next training step (at time t+1). [57]

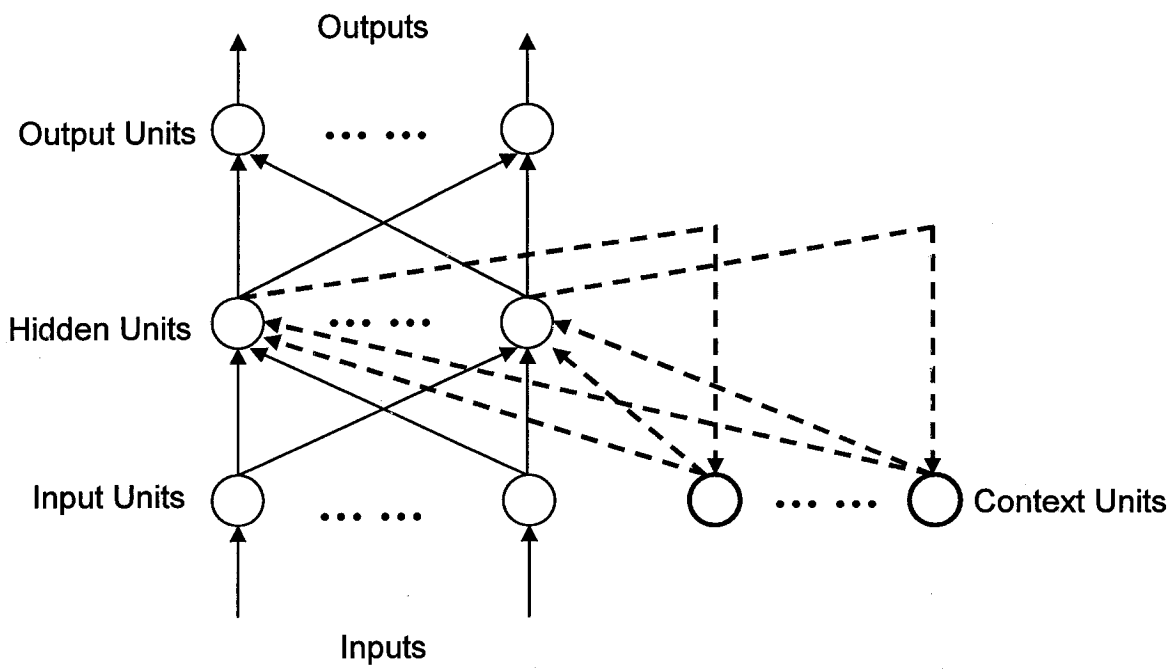

Figure 4.9 The Elman Network Structure

4.10.2 Introductions to the Seven Revised BP Training Methods

4.10.2.1 Gradient Descent BP

In the Gradient Descent BP method, the weight vector moves directly down the local gradient as:

$$\Delta w = -\eta \frac{\partial \varepsilon(w)}{\partial w} \tag{4-36}$$

where w is the weight and η is the learning rate that controls how much w changes in each iteration, $\frac{\partial \varepsilon(w)}{\partial w}$ is the derivative of error ε with respect to w. This method may converge very slowly when the gradient, and thus the step size, approaches zero at the minimum. [53]. The pseudocode for this is straightforward and thus omitted.

4.10.2.2 Gradient Descent With Momentum BP

Gradient Descent works well when the problem is not complex and thus the error surface is simple, as shown in Figure 4.10. It starts at some point defined by the initial weights, and moves downhill to the global minimum.

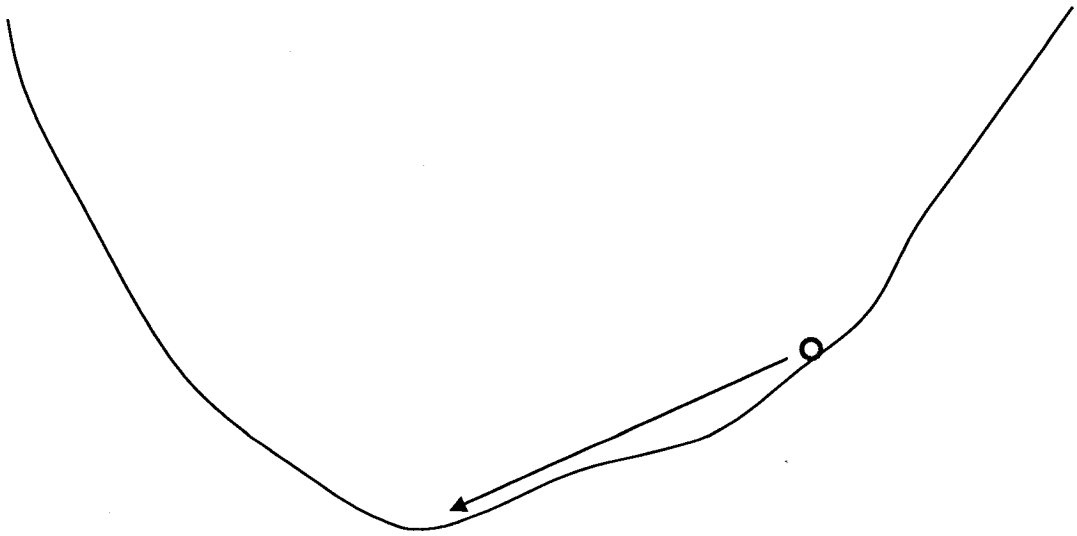

Figure 4.10 Error Surface of Simple Problem

But Gradient Descent does not work well when the problem is complex and if the error surface possesses several local minima. It then has the potential of getting in the local minimum and cannot escape, as shown in Figures 4.11 and Figure 4.12.

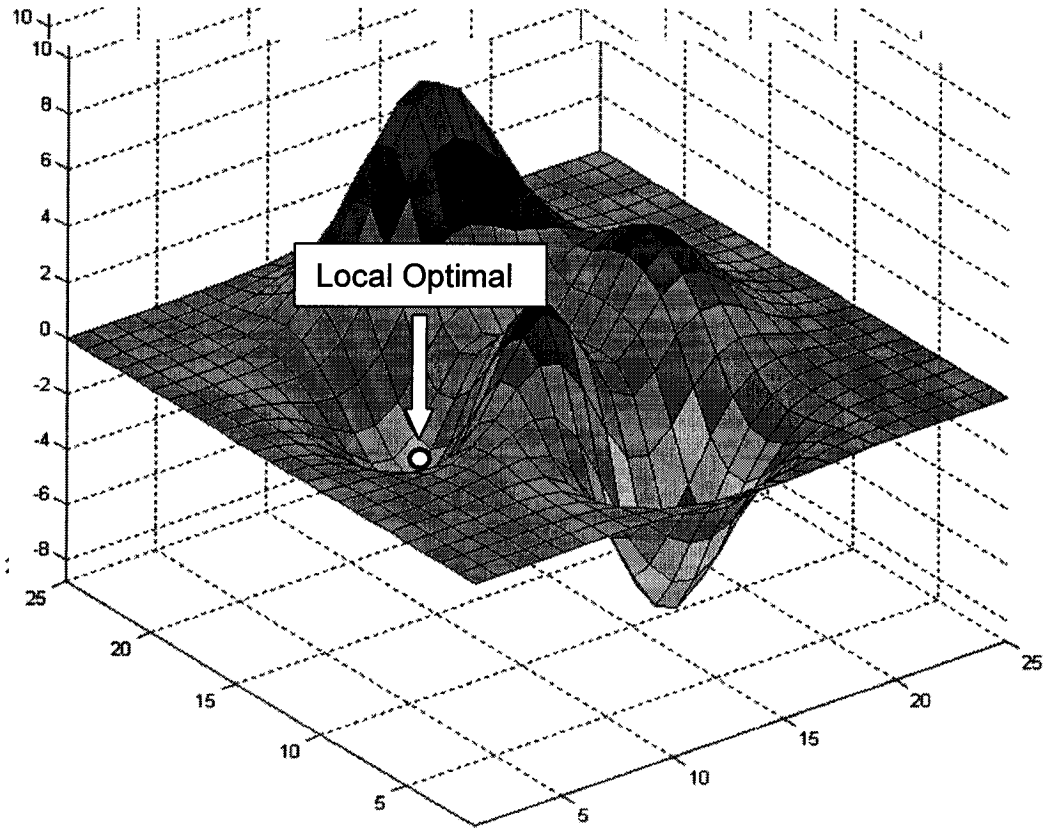

Figure 4.11 Gradient Descent is Trapped in a Local Optimal

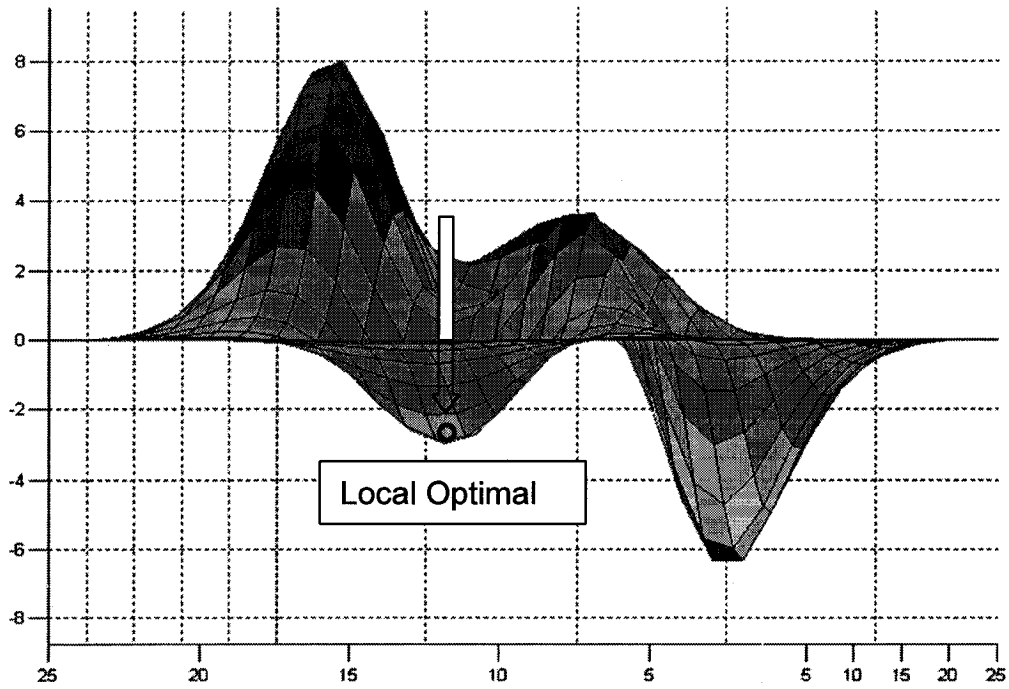

Figure 4.12 Side View of Local Optimal

Consider Figures 4.11 and 4.12, where a ball is descending down a valley. Once trapped in a local minimum, the ball has to climb higher before it can reach the global minimum. Gradient Descent always goes downhill and thus can never reach the global minimum. It ignores the previous gradient descent and only takes the current gradient descent into consideration when it updates weights. To accelerate the convergence and help escape from the local minimum, a momentum constant is added, which is a global parameter determined by trial and error. Thus the update formula for the weights is:

$$\Delta w_{now} = \eta \delta y + m \Delta w_{old} \tag{4-37}$$

where η is the learning rate, m is momentum constant ranging from 0 to 1, and y is the inputs of the given layer.

By adding a momentum constant, the previous weights are considered in the updating. If the gradient keeps pointing to the same direction, the momentum will increase the step size. If the gradient keeps changing direction, the momentum will smooth the fluctuation. Thus it will help accelerate the convergence to the global minimum and help to escape the local minimum.

4.10.2.3 Gradient Descent With Adaptive Learning Rate BP

This method employs an adaptive learning rate to accelerate the convergence. Each weight w has its own local learning rate η, which is changed adaptively following the behavior of the local gradient error function $\partial\varepsilon/\partial w$. The learning rate η is increased when the signs of term $\partial\varepsilon/\partial w$ are the same in at least two successive iterations, and decreased when the signs are opposite. The learning rate η is updated by [58]:

$$\eta_{next} = \begin{cases} \eta_{now}\left(\dfrac{\eta^{\max}}{\eta_{now}}\right)^{\gamma}, \text{ when } s_{now} = s_{old} \\ \\ \eta_{now}\left(\dfrac{\eta^{\min}}{\eta_{now}}\right)^{\gamma}, \text{ when } s_{now} \neq s_{old} \end{cases} \tag{4-38}$$

where $s_{now} = sign(\partial\varepsilon_{now}/\partial w) = +1$ if $\partial\varepsilon_{now}/\partial w > 0$,

$s_{now} = sign(\partial\varepsilon_{now}/\partial w) = -1$ if $\partial\varepsilon_{now}/\partial w < 0$

where s_{now} is the sign of the gradient $\partial\varepsilon_{now}/\partial w$, and η_{now} is the current learning rate value.

$\eta^{\max}$ and $\eta^{\min}$ are the current maximum and minimum learning rate values, and γ is the learning rate adaptation coefficient ranging from 0 to 1.

4.10.2.4 Gradient Descent with Momentum and Adaptive Learning Rate BP

This method is a combination of the previous two, which employs the momentum and adaptive learning rate at the same time.

4.10.2.5 Broyden-Fletcher-Goldfarb-Shanno (BFGS) Quasi-Newton BP

BFGS Quasi-Newton training method is an update version of Newton's method. Newton method is a fast training algorithm, represent by:

$$w_{next} = w_{now} - H_{now}^{-1} g_{now} \tag{4-39}$$

where w is the weight, g is the gradient, H is the Hessian matrix that is composed of the second derivatives of the performance function with respect to the weight and bias variables. Newton's training technique is fast, but requires extensive computing because it has to compute the matrix of second derivatives. Thus, the Quasi-Newton method is employed to avoid excessive computation. This method updates the Hessian matrix by computing a function with respect to g instead of computing the second derivatives. In the quasi-Newton training method, the

weights are updated as:

$$w_{next} = w_{now} - \eta H_{now}^{-1} g_{now} \tag{4-40}$$

$$H_{now}^{-1} = H_{old}^{-1} + \Delta H_{now}^{-1} \tag{4-41}$$

where the step size η controls how much w changes in each iteration. H^{-1} is not calculated, but estimated by:

$$\Delta H_{now}^{-1} = \frac{\Delta w \Delta w^T}{\Delta w^T \Delta g} - \frac{H_{old}^{-1} \Delta g \Delta g^T H_{old}^{-1}}{\Delta g^T H_{old}^{-1} \Delta g} \tag{4-42}$$

$$\Delta H_{now}^{-1} = (1 + \frac{\Delta g^T H_{old}^{-1} \Delta g}{\Delta w^T \Delta g}) \frac{\Delta w \Delta w^T}{\Delta w^T \Delta g} - \frac{\Delta w \Delta g^T H_{old}^{-1} + H_{old}^{-1} \Delta g \Delta w^T}{\Delta w^T \Delta g} \tag{4-43}$$

where

$\Delta w = w_{now} - w_{old}$, and (4-44)

$$\Delta g = g_{now} - g_{old} \tag{4-45}$$

In every iteration, the inverse Hessian approximation is reset to the identity matrix, thus avoiding the need for a large number of units of space to store the approximation of the inverse Hessian matrix. The BFGS Quasi-Newton is a simplified version of Quasi-Newton method. Though it needs the computation of the estimation of inverse Hessian matrix, it is still faster than Newton's method and the conjugate gradient methods. [37] [59]

4.10.2.6 Levenberg-Marquardt BP

Levenberg-Marquardt is a popular method useful for solving the nonlinear least squares problems. [37]

Let ***e*** be a vector containing the individual error terms, and J be the Jacobian [60] matrix containing the derivatives of the error ***e*** with respect to ***w***. The Levenberg-Marquardt (LM) method can be applied when the Jacobian matrix J_{now} is rank deficient. In the LM method, the weight update is given by

$$w_{next} = w_{now} - \left(J_{now}^T J_{now} + \mu I\right)^{-1} J_{now}^T e_{now} \tag{4-46}$$

where μ is a non-negative number, and I is an identity matrix.

4.10.2.7 RPROP BP

RPROP stands for "Resilient Propagation" which is a local adaptive learning method.

The main difference between it and most other heuristic backpropagation variations is that the learning rate adjustments and weight changes depend only on the signs of the gradient terms, not their magnitudes. It is argued that the gradient magnitude depends on scaling of the error function and can change greatly from one step to the next. On the complicated nonlinear error surface, the magnitude is basically unpredictable and there is no reason why the step size, in general, should be proportional to the magnitude. In fact, it can be argued that the step size should be inversely proportional in order to take large steps where the gradient is small, and to take small, careful, steps where the gradient is large. [37]

[62]

The advantage of RPROP BP is the ability that it can eliminate the harmful influence of the size of the partial derivative on the weight step. In RPROP BP, only the sign of the derivative is considered to indicate the direction of the weight update. The size of the weight change is exclusively determined by a weight-specific Δ_{ij}, *update-value* [60] [61] [62] as:

$$\Delta w_{ij}^{(now)} = \begin{cases} -\Delta_{ij}^{(now)}, & \text{if } \frac{\partial \varepsilon_{now}}{\partial w_{ij}} > 0 \\ +\Delta_{ij}^{(now)}, & \text{if } \frac{\partial \varepsilon_{now}}{\partial w_{ij}} < 0 \\ 0, & \text{if } \frac{\partial \varepsilon_{now}}{\partial w_{ij}} = 0 \end{cases} \tag{4-47}$$

where $\frac{\partial \varepsilon_{now}}{\partial w_{ij}}$ is the summed gradient information over all patterns of the pattern set. In such a case, RPROP BP determines the new update-values $\Delta_{ij}^{(now)}$ based on a sign-dependent adaptation process:

$$\Delta_{ij}^{(now)} = \begin{cases} \eta^{+} \cdot \Delta_{ij}^{(old)}, & \text{if } \frac{\partial \varepsilon_{old}}{\partial w_{ij}} \cdot \frac{\partial \varepsilon_{now}}{\partial w_{ij}} > 0 \\ \eta^{-} \cdot \Delta_{ij}^{(old)}, & \text{if } \frac{\partial \varepsilon_{old}}{\partial w_{ij}} \cdot \frac{\partial \varepsilon_{now}}{\partial w_{ij}} < 0 \end{cases} \tag{4-48}$$

where η^+ is the increase factor, and η^- is the decreased factor ($0<\eta^-<1<\eta^+$).

The adaptation works as follows: The update-value $\Delta_{ij}^{(now)}$ is decreased by the factor η^- every time when the partial derivative of the weight w_{ij} changes its sign, which means that the last update was too large and that the algorithm had jumped over the local minimum. If the sign of the derivative remains the same, the update-value is slightly increased to accelerate convergence. If the sign changes, the adaptation will not be employed in the next learning step. The following steps can be used to clarify for RPROP BP. In the description, ***max*** and ***min*** operators are defined to return the maximum and minimum number of the two arguments, and the ***sign*** operator returns +1 if the argument is positive, -1 if the argument is negative, and 0 otherwise. [60] [61]

Initially $\forall i, j$,

$\Delta_{i,j}^{(now)} = \Delta_o$ and

$$\frac{\partial \varepsilon_{old}}{\partial w_{i,j}} = 0$$

Subsequently, while the algorithm has not converged, the gradient $\frac{\partial \varepsilon_{now}}{\partial w}$ is computed.

The weights and biases are now computed using the following three cases.

Case 1: If $\left(\frac{\partial \varepsilon^{(old)}}{\partial w_{i,j}} \cdot \frac{\partial \varepsilon^{(now)}}{\partial w_{i,j}} \right) > 0$

$$\Delta_{i,j}^{(now)} = \boldsymbol{min}\left(\Delta_{i,j}^{(old)} \cdot k^{+}, \Delta_{\max}\right)$$

$$\Delta w_{i,j}^{(now)} = -\boldsymbol{sign}\left(\frac{\partial \varepsilon^{(now)}}{\partial w_{i,j}} \cdot \Delta_{i,j}^{(now)}\right)$$

$$w_{i,j}^{(next)} = w_{i,j}^{(now)} + \Delta_{i,j}^{(now)}$$

$$\frac{\partial \varepsilon^{(old)}}{\partial w_{i,j}} = \frac{\partial \varepsilon^{(now)}}{\partial w_{i,j}}$$

Case 2: If $\left(\frac{\partial \varepsilon^{(old)}}{\partial w_{i,j}} \cdot \frac{\partial \varepsilon^{(now)}}{\partial w_{i,j}} < 0\right)$

$$\Delta_{i,j}^{(now)} = \boldsymbol{max}\left(\Delta_{i,j}^{(old)} \cdot k^{-}, \Delta_{\min}\right)$$

$$\frac{\partial \varepsilon^{(old)}}{\partial w_{i,j}} = 0$$

Case 3: if $\left(\frac{\partial \varepsilon^{(old)}}{\partial w_{i,j}} \cdot \frac{\partial \varepsilon^{(now)}}{\partial w_{i,j}} = 0\right)$

$$\Delta w_{i,j}^{(now)} = -\boldsymbol{sign}\left(\frac{\partial \varepsilon^{(now)}}{\partial w_{i,j}} \cdot \Delta_{i,j}^{(now)}\right)$$

$$w_{i,j}^{(next)} = w_{i,j}^{(now)} + \Delta_{i,j}^{(now)}$$

$$\frac{\partial \varepsilon^{(old)}}{\partial w_{i,j}} = \frac{\partial \varepsilon^{(now)}}{\partial w_{i,j}}$$

4.11 Conclusion

This chapter introduced the MLP and BP algorithms, and discussed the criterion for the selection of the number of layers, and the number of neurons. We then studied three different network topologies and seven training methods applicable for each. In the next chapter of This thesis, we will compare the performance of the twenty-one combinations and select the best one to predict the stock market.

CHAPTER V

BP ALGORITHMS TO PREDICT STOCK PRICE

5.1 Programming Language

All the programs which implemented the NN strategies were developed using Matlab. Matlab is a language useful for technical computing developed by Cleve Moler in 1970s to help students work with matrices. The programming package used to get the results for this thesis is the Neural Network Toolbox 4.0.2 in Matlab 6.5 Release 13. This toolbox contains a complete set of functions suitable from the design to the implementation and simulation of neural networks [63].

5.2 Experimental Data

Usually, it is almost impossible to procure the data of the stock market from any stock exchange, since this data is both commercial and secret. Although stock traders can read the price of any stock every day, they cannot access any original file that

stores the historical data in any particular format in the database. We are very grateful that we were given access to two Chinese stock exchanges [6] [7]. Each stock data is stored in a file that records the historical data including the open price, the daily high, the daily low, the closing price, etc. The file is named after its symbol, which is a six-digit number. The data file of the Shanghai Stock Exchange Composite Index (SSE composite index) is 000001.day, and that of the Shenzhen Stock Exchange Component Index (SZSE Component Index) is 399001.day. Both of them are encrypted files. After decryption, all the data, except the closing prices were deleted to perform the experiments required in this thesis. In order to reduce the impact of noise caused by occasional factors to the minimum level and to smooth the curve, we calculated the average of every three closing prices. Thus every dot in the figures represents an average of three closing prices.

Other kinds of data, such as open price, daily high, daily low and volume, were discarded because they were already considered when the closing price was taken into consideration. All kinds of data and information (including open price, daily high, daily low and volume) that affect the closing price are thus reflected in its fluctuation, and thus we believe that using all kinds of data as the input set would be redundant. Furthermore, the multi-dimensional input vector will make the network much more complex and render the convergence much more difficult.

5.3 Network Structure

In this study, we implemented the most popular network suggested earlier, the three-layer network. The number of neurons in the input and output layers is 1. The input is a vector, which contains a lot of values of the sequential number of the date for the prices. The output is also a vector, which has the same number of values of prices in sequence. In the hidden layer, we will attempt to use 10, 20, 30, 40 and 50 neurons.

5.4 Methodology

In this thesis, we implement the results of three supervised algorithms: Feedforward, Cascade-forward and Elman. This thesis also implements seven training methods: Gradient Descent, Gradient Descent with Momentum, Gradient Descent with Adaptive Learning Rate, Gradient Descent with Momentum and Adaptive Learning Rate, Broyden-Fletcher-Goldfarb-Shanno (BFGS) Quasi-Newton, Levenberg-Marquardt and RPROP (Resilient Propagation) BP. In this chapter, we will compare the twenty-one combinations of the three supervised algorithms and seven training methods as shown in Table 5.1.

Table 5.1 Experiment Outline

NN Type	Training Method	Notation
Feed-forward trained by	Gradient Descent BP	<1.1>
	Gradient Descent With Momentum BP	<1.2>
	Gradient Descent With Adaptive Learning Rate BP	<1.3>
	Gradient Descent W/Momentum and Adaptive Learning Rate BP	<1.4>
	BFGS Quasi-Newton BP	<1.5>
	Levenberg-Marquardt (LM) BP	<1.6>
	RPROP BP	<1.7>
Elman trained by	Gradient Descent BP	<2.1>
	Gradient Descent With Momentum BP	<2.2>
	Gradient Descent With Adaptive Learning Rate BP	<2.3>
	Gradient Descent W/Momentum and Adaptive Learning Rate BP	<2.4>
	BFGS Quasi-Newton BP	<2.5>
	Levenberg-Marquardt (LM) BP	<2.6>
	RPROP BP	<2.7>
Cascade-forward trained by	Gradient Descent BP	<3.1>
	Gradient Descent With Momentum BP	<3.2>
	Gradient Descent With Adaptive Learning Rate BP	<3.3>
	Gradient Descent W/Momentum and Adaptive Learning Rate BP	<3.4>
	BFGS Quasi-Newton BP	<3.5>
	Levenberg-Marquardt (LM) BP	<3.6>
	RPROP BP	<3.7>

5.5 Experiment Environment

The experiments conducted for the thesis were done on a PC with the following specifications summarized as following:

Operating System: Microsoft Windows XP

CPU: Pentium III 1G

Memory: 512M

5.6 Experiment Outline

We tried every combination listed above, {<1.1>, ... <1.7>, <2.1>, ... <2.7>, <3.1>, ... <3.7>} and then reported the detailed results to compare the various schemes. In the experiments, we used the mean squared error (MSE) to measure the performance of the NN, where the MSE is given by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(PV_i - TV_i\right)^2 \qquad (5\text{-}1)$$

where PV_i is the i^{th} predicted value and TV_i is the i^{th} target value out of n sample cases. In each combination, we tried to incorporate 10, 20, 30, 40 and 50 neurons respectively for the hidden layer. The maximum number of epochs was set at 5,000. The network stopped training when the maximum number of epochs exceeded 5,000, which was normally enough to reach a good user-defined error of less than 1%.

5.7 Experiment Details

The following tables report the experimental results in detail. We combined every one of the three algorithms with seven training methods to yield the 21 different combinations. Also, for each combination, we selected 10, 20, 30, 40 and 50 neurons in the hidden layer. For every trial of the number of the hidden neurons, we then performed five experiments to check the overall performance. Therefore, each algorithm was tested 175 times. Thus, the three algorithms trained by the seven

methods yielded 525 experimental results. The following three tables list the error performance of the 525 experimental results, measured in terms of the MSE.

The training data used was the SSE Composite Index of seven months starting from December 1990.

5.7.1 Experiment Results of Feed-forward BP

The method <*i*, *j*> is as per the last column of Table 5.1. The symbol 'MGR' in the following tables implies Minimum Gradient Reached, which means the performance gradient attained below the defined minimum gradient, which was $1.0e^{-10}$. When the minimum gradient was reached, we stopped the experiment because the search did not improve or go downhill. The maximum number of epochs defined in this thesis is 5,000 and the performance goal was to obtain an MSE of 0.001. If any experiment met the performance goal within 5,000 epochs, we recorded the number of the epoch, where the program stopped. If the experiment did not meet the performance goal within 5,000 epochs, we also recorded the MSE of 5000^{th} epoch. Thus, any performance goal better than 0.001 was listed (in parentheses) together with the number of the epoch where the performance goal was met. Any MSE larger than 0.001 was listed by itself and the number of the epoch was omitted since, in each case, it was 5,000.

Method	Expmnt	10 neurons	20 neurons	30 neurons	40 neurons	50 neurons
<1.1>	1-5	MGR	MGR	MGR	MGR	MGR
	Best	NA	NA	NA	NA	NA
	Average	NA	NA	NA	NA	NA
<1.2>	1-5	MGR	MGR	MGR	MGR	MGR
	Best	NA	NA	NA	NA	NA
	Average	NA	NA	NA	NA	NA
<1.3>	1-5	MGR	MGR	MGR	MGR	Not Converge
	Best	NA	NA	NA	NA	NA
	Average	NA	NA	NA	NA	NA
<1.4>	1	80.0718	80.0581	80.0674	80.0581	80.0581
	2	80.0583	80.0581	80.6212	80.0581	80.0581
	3	80.0606	80.0581	80.0581	80.0581	80.0581
	4	80.2019	80.1483	80.0581	80.0599	80.3060
	5	82.3994	80.0587	80.2241	80.0594	80.0581
	Best	80.0583	80.0581	80.0581	80.0581	80.0581
	Average	80.5584	80.0763	80.2058	80.0587	80.1077
<1.5>	1-5	MGR	MGR	MGR	MGR	MGR
	Best	NA	NA	NA	NA	NA
	Average	NA	NA	NA	NA	NA

Table 5.2 Feed-forward Neural Network Performance (MSE)
(To Be continued)

Method	Expmnt	10 neurons	20 neurons	30 neurons	40 neurons	50 neurons
<1.6>	1	1.0727	9.9870e-4 (1719)	1.2239e-3	9.9894e-4 (281)	1.1543e-005 (9)
	2	0.1361	3.3035e-3	9.9719e-4 (477)	9.9997e-4 (579)	1.7058e-4 (8)
	3	36.7964	9.8954e-4 (667)	9.7715e-4 (2632)	9.9993e-4 (2630)	1.7480e-4 (10)
	4	2.6253	9.8443e-4 (3394)	9.9808e-4 (2363)	9.9971e-4 (1692)	7.1761e-7 (6)
	5	2.0900	1.2359e-3	0,03975	9.9731e-4 (3728)	1.5340e-4 (9)
	Best	0.1361	perfmnce goal met at epoch 667	perfmnce goal met at epoch 477	perfmnce goal met at epoch 281	perfmnce goal met at epoch 6
	Avg	8.5441	NA	NA	perfmnce goal met at average epoch 1782	perfmnce goal met at average epoch 8.4
<1.7>	1	MGR	8.8052	8.5650	7.0424	8.0905
	2	71.1155	28.0189	7.1253	8.2423	27.8994
	3	MGR	71.1156	0.7469	71.1238	0.8757
	4	MGR	6.8302	8.2366	71.1160	8.2387
	5	8.5261	71.1158	7.1483	8.8017	7.1297
	Best	NA	6.8302	0.7469	7.0424	0.8757
	Avg	NA	37.1771	6.3644	33.2652	10.4468

Table 5.2 Feed-forward Neural Network Performance (MSE)

Table 5.2 shows the results of 175 experiments of the Feed-forward algorithm trained by seven training methods. Some of the experiments stopped in the middle, and never finished because the gradient descent was too small in each update. In the experiments of Feed-forward trained by Gradient Descent With Momentum and Adaptive Learning Rate BP, the number of neurons chosen for the hidden layer did not matter, the MSEs were the same, around 80. This indicated that this combination was not a good choice for the problem studied in this thesis.

The best result in Table 5.2 was obtained for the Feed-forward algorithm trained

by LM BP. By using only 10 neurons in the middle layer, the network attained a very good performance with a MSE less than 10. At this point, the *Approximation_Error* was very small, usually less than 0.02, where

$$Approximation\_Error = \frac{1}{n}\sum_{1}^{n}\left|\frac{approximation\ value - real\ value}{real\ value}\right| \qquad (5\text{-}2)$$

which is recorded as a percentage. When the network had 10 neurons in the hidden layer, one of resulting *MSEs* was even as small as 0.136115, leading to a *Approximation_Error* of 0.0019. When the number of hidden neurons was increased to 20, the network was able to reach the performance goal (*MSE*=0.001) as early as in the 667th epoch, where *Approximation_Error* was 7.7938e-005. When the number of neurons in the hidden layer was increased to 50, the network has the best performance. It reaches the performance goal within 10 epochs in every trial. The combination of the Feed-forward and LM BP yielded outstanding results for our problem.

5.7.2 Experiment Results of Elman BP

The results for the Elman BP algorithm are given in Table 5.3 below.

Method	Expmnt	10 neurons	20 neurons	30 neurons	40 neurons	50 neurons
<2.1>	1	80.0581	MGR	MGR	71.1321	80.0577
	2	80.0581	MGR	MGR	80.0571	80.0577
	3	MGR	MGR	80.0581	MGR	80.0581
	4	80.0581	MGR	MGR	80.0581	MGR
	5	80.0581	80.0577	MGR	MGR	80.0581
	Best	80.0581	80.0577	80.0581	80.0571	80.0577
	Average	NA	NA	NA	NA	NA
<2.2>	1-3	MGR	MGR	MGR	MGR	MGR
	4	MGR	MGR	MGR	MGR	80.0582
	5	MGR	MGR	MGR	MGR	MGR
	Best	NA	NA	NA	NA	80.0582
	Average	NA	NA	NA	NA	NA
<2.3>	1	MGR	MGR	MGR	MGR	121.061
	2	MGR	MGR	MGR	MGR	82.5037
	3	MGR	MGR	MGR	MGR	80.7061
	4	MGR	MGR	MGR	MGR	103.8680
	5	MGR	MGR	80.3297	MGR	132.4130
	Best	NA	NA	80.3297	NA	80.7061
	Average	NA	NA	NA	NA	104.11036
<2.4>	1	80.0718	80.0583	MGR	80.0682	80.0582
	2	80.3337	80.0581	MGR	81.0480	80.0581
	3	80.0582	80.0581	80.0581	80.0581	80.0581
	4	84.9556	80.1483	MGR	80.0583	81.5973
	5	80.0581	MGR	80.2241	80.0601	80.0581
	Best	80.0581	80.0581	80.0583	80.0581	80.0581
	Average	81.0955	NA	80.0581	80.2585	80.3660
<2.5>	1-5	MGR	MGR	MGR	MGR	MGR
	Best	NA	NA	NA	NA	NA
	Average	NA	NA	NA	NA	NA

Table 5.3 Elman Neural Network Performance (MSE)
(To Be Continued)

Trained By	Expmnt	10 neurons	20 neurons	30 neurons	40 neurons	50 neurons
<2.6>	1	MGR	0.0066	42.2857 (Exc)	80.0581 (Exc)	80.0581 (Exc)
	2	MGR	71.1155	71.3991 (Exc)	(Exc)	(Exc)
	3	MGR	71.1155	(Exc)	(Exc)	(Exc)
	4	MGR	0.0016	(Exc)	(Exc)	(Exc)
	5	MGR	MGR	(Exc)	(Exc)	(Exc)
	Best	NA	0.0016	NA	NA	NA
	Average	NA	NA	NA	NA	NA
<2.7>	1	MGR	8.6257	8.7115	8.7164	7.4834
	2	MGR	7.1947	71.1156	7.5092	7.5306
	3	MGR	71.1156	7.4831	7.5340	7.5622
	4	MGR	7.4533	7.6234	7.4491	7.4722
	5	MGR	MGR	7.4069	7.4807	7.5544
	Best	NA	7.1947	7.4069	7.4491	7.4722
	Average	NA	NA	20.4681	7.7379	7.5205

Table 5.3 Elman Neural Network Performance (MSE)

The symbol 'Exc' implies that the time was excessive for each epoch. In our experiments, the term 'Exc' is used if the network needs more than one hour for each epoch.

From the table we see that the Elman algorithm is unsuitable for the problem studied in this thesis.

Independent of the training method used and the number of neurons used, the network never yielded an acceptable result. The Elman algorithm performed badly in delivering a good approximation for a non-linear problem. The worst combination is the Elman trained by LM BP. Not only were the results bad, but also the network needed a lot of time for each epoch. When the number of the hidden neurons reached 50, it needed one hour for each of the 50 epochs, and thus several days to finish 5,000 epochs. It was extremely slow. Elman has an extra context layer, which functions as one-step time delay. At any time *t*, Elman has inputs which include inputs of time *t* from

input layer and inputs of time t-1 from context layer. The inputs from the input layer are always in sequence. But when the inputs from the context layer join, the sequence is destroyed. That might be the reason why Elman algorithm performed badly.

5.7.3 Experiment Results of Cascade-forward BP

The results of the Cascade-forward BP algorithm are given below in Table 5.4. In this Table, the symbol 'NC' implies that the scheme did not converge and 'MGR' implies that the minimum gradient was reached.

Method	Expmnt	10 neurons	20 neurons	30 neurons	40 neurons	50 neurons
<3.1>	1-5	NC	NC	NC	NC	NC
	Best	NA	NA	NA	NA	NA
	Average	NA	NA	NA	NA	NA
<3.2>	1	MGR	15.0161	14.3928	18.5208	24.7539
	2	46.6597	46.6597	14.3294	15.4876	18.512
	3	46.6597	18.5103	24.7539	18.5115	14.3136
	4	18.5117	46.6597	24.7538	24.7536	24.7539
	5	46.6597	18.5117	46.6597	18.5107	46.6597
	Best	18.5117	15.0161	14.3294	15.4876	14.3136
	Average	NA	29.0715	24.9779	19.1568	25.7986
<3.3>	1	15.0687	7.8456	2.7850	2.7937	2.8346
	2	62.8488	NC	NC	6.3416	0.9860
	3	12.2176	NC	6.5511	3.8239	1.7454
	4	13.5903	NC	2.3736	2.3249	4.1263
	5	NC	NC	3.5357	2.3380	2.4167
	Best	12.2176	7.8456	2.3736	2.3249	0.9860
	Average	NA	NA	NA	3.5244	2.4218
<3.4>	1	14.3871	5.3770	4.7587	2.3971	1.3891
	2	14.6348	9.7632	6.5018	1.0013	1.0178
	3	14.4610	7.1646	1.0629	1.0013	0.2536
	4	11.3124	7.7431	6.5491	1.5149	1.3534
	5	7.9536	6.0075	3.1573	0.6776	1.0748
	Best	7.9536	5.3770	1.0629	0.6776	0.2536
	Average	12.5498	7.2111	4.4059	1.3184	1.0177
<3.5>	1	0.0336	0.1129	0.0546	0.0794	0.0140
	2	MGR	0.4774	0.1163	0.0654	MGR
	3	MGR	0.0518	MGR	MGR	MGR
	4	MGR	0.0882	0.1277	0.0961	MGR
	5	3.0180	MGR	MGR	MGR	MGR
	Best	0.0336	0.0518	0.0546	0.0654	0.0140
	Average	NA	NA	NA	NA	NA

Table 5.4 Cascade-forward Neural Network Performance (MSE)
(To Be Continued)

Method	Expt	10 neurons	20 neurons	30 neurons	40 neurons	50 neurons
<3.6>	1	13.5281	4.0638	0.1954	0.0398	0.0568
	2	3.0252	0.7681	0.0409	0.0169	9.99133e-4 (1042)
	3	0.2842	0.9512	2.0321	0.1208	0.0593
	4	0.0829	0.0330	0.0418	0.0196	9.87685e-4 (369)
	5	0.0829	13.5289	0.0410	0.1486	0.0279
	Best	0.0829	0.0330	0.0409	0.0169	9.99133e-4 (1042)
	Avg	3.4007	3.8690	0.4703	0.0691	NA
<3.7>	1	7.5595	7.5623	5.0196	0.0781	0.1064
	2	10.0899	7.5956	7.6192	0.0433	0.0419
	3	MGR	7.3882	8.4402	7.2266	5.3115
	4	MGR	1.4042	5.1879	7.5827	0.8403
	5	MGR	2.9167	6.1924	0.4153	0.1131
	Best	7.5595	1.4042	5.0196	0.0433	0.0419
	Avg	NA	5.3734	6.4919	3.0692	1.2826

Table 5.4 Cascade-forward Neural Network Performance (MSE)
(End)

The experimental results in the table also show that Cascade-forward does not yield a satisfactory result for the problem studied in this thesis, even though it sometimes meets the performance goal when trained by the LM BP scheme.

We are not able to determine why only one Feed-forward NN trained by the LM BP algorithm works well for the stock market. There seems to be no apparent reason, but that is exactly what we have observed. It may be because the model presented by this NN fits the Shanghai index most appropriately.

Until now, the experiments we have done use three-day average prices to remove the effects of noise. We now consider if it is better to use such an average than all data. To study this, we do the experiments again using all data points, and without averaging. The following are the results obtained.

Trained By	Expmnt	10 neurons	20 neurons	30 neurons	40 neurons	50 neurons
Gradient Descent BP	1	80.2415	MGR	MGR	MGR	MGR
	2-5	MGR	MGR	MGR	MGR	MGR
	Best	80.2415	NA	NA	NA	NA
	Average	NA	NA	NA	NA	NA
Gradient Descent With Momentum BP	1-5	MGR	MGR	MGR	MGR	MGR
	Best	NA	NA	NA	NA	NA
	Average	NA	NA	NA	NA	NA
Gradient Descent With Adaptive Learning Rate BP	1-5	MGR	MGR	MGR	MGR	MGR
	Best	NA	NA	NA	NA	NA
	Average	NA	NA	NA	NA	NA
Gradient Descent With Momentum & Adaptive Learning Rate BP	1-5	MGR	80.2417	80.2417	80.2417	80.2417
	Best	NA	80.2417	80.2417	80.2417	80.2417
	Average	NA	80.2417	80.2417	80.2417	80.2417
BFGS Quasi-Newton BP	1-5	MGR	MGR	MGR	MGR	MGR
	Best	NA	NA	NA	NA	NA
	Average	NA	NA	NA	NA	NA
Levenberg-Marquardt BP	1	0.227172	Not Converge	0.00962588	0.0156835	0.0129606
	2	2.44674	0.121386	0.0523246	0.00894971	0.00906919
	3	0.922198	0.0892799	0.0267692	0.0254026	0.0142929
	4	1.29218	0.0755347	0.0157757	0.0147932	Not Converge
	5	Not Converge	0.157125	Not Converge	0.0151983	Not Converge
	Best	0.227172	0.0755347	0.00962588	0.00894971	0.00906919
	Average	NA	NA	NA	0.016005462	NA
RPROP BP	1	70.7166	MGR	29.6677	65.218	70.6751
	2	70.7239	29.7223	29.6106	70.7057	28.6083
	3	MGR	70.676	70.7864	29.7453	0.992781
	4	MGR	70.6784	23.5673	70.6759	1.79584
	5	MGR	70.6795	70.6783	70.6786	8.71851
	Best	70.7166	29.7223	23.5673	29.7453	0.992781
	Average	NA	Na	43.64198	61.4047	22.1581062

Table 5.5 Performance (MSE) of Feed-forward Neural Network Trained by All Data

Trained By	Expmnt	10 neurons	20 neurons	30 neurons	40 neurons	50 neurons
Gradient Descent BP	1	80.2417	MGR	MGR	MGR	MGR
	2-5	MGR	MGR	MGR	MGR	MGR
	Best	80.2417	NA	NA	NA	NA
	Average	NA	NA	NA	NA	NA
Gradient Descent With Momentum BP	1-5	MGR	MGR	MGR	MGR	MGR
	Best	NA	NA	NA	NA	NA
	Average	NA	NA	NA	NA	NA
Gradient Descent With Adaptive Learning Rate BP	1	MGR	MGR	MGR	MGR	89.7119
	2	MGR	MGR	MGR	MGR	80.4636
	3	MGR	MGR	MGR	MGR	83.3878
	4	MGR	MGR	MGR	MGR	110.605
	5	MGR	MGR	MGR	MGR	138.621
	Best	NA	NA	NA	NA	80.4636
	Average	NA	NA	NA	NA	100.55786
Gradient Descent With Momentum & Adaptive Learning Rate BP	1	80.2417	MGR	80.3296	80.2827	80.2417
	2	80.2417	MGR	80.2417	80.2417	80.2417
	3	80.2417	MGR	80.2556	80.2417	80.2417
	4-5	80.2417	MGR	80.2417	80.2417	80.2417
	Best	80.2417	NA	80.2417	80.2417	80.2417
	Average	80.2417	NA	80.26206	80.2499	80.2417
BFGS Quasi-Newton BP	1-5	MGR	MGR	MGR	MGR	MGR
	Best	NA	NA	NA	NA	NA
	Average	NA	NA	NA	NA	NA
Levenberg-Marquardt BP	1	28.2513	MGR	MGR	MGR	MGR
	2	28.1882	MGR	MGR	MGR	MGR
	3-5	MGR	MGR	MGR	MGR	MGR
	Best	28.1882	NA	NA	NA	NA
	Average	NA	NA	NA	NA	NA
RPROP BP	1	MGR	29.8319	29.809	29.1485	29.0562
	2	MGR	29.8398	28.5562	29.0742	29.0254
	3	MGR	70.6743	28.1104	28.3665	28.0115
	4	MGR	11.0258	27.9335	27.9875	28.3665
	5	MGR	29.8338	29.0635	28.9599	28.9885
	Best	NA	11.0258	27.9335	27.9875	28.0115
	Average	NA	34.24112	28.69452	28.70732	28.68962

Table 5.6 Performance (MSE) of Elman Neural Network Trained by All Data

Trained By	Expmnt	10 neurons	20 neurons	30 neurons	40 neurons	50 neurons
Gradient Descent BP	1-5	Not Converge	Not Converge	Not Converge	Not Converge	Not Converge
	Best	NA	NA	NA	NA	NA
	Average	NA	NA	NA	NA	NA
Gradient Descent With Momentum BP	1-5	Not Converge	Not Converge	Not Converge	Not Converge	Not Converge
	Best	NA	NA	NA	NA	NA
	Average	NA	NA	NA	NA	NA
Gradient Descent With Adaptive Learning Rate BP	1	51.552	32.6559	Not Converge	2.93638	8.84848
	2	49.5446	31.5292	19.5972	4.18051	9.37739
	3	63.4093	31.5795	20.2986	3.06144	9.78211
	4	57.7889	32.6428	20.144	2.90221	8.93225
	5	376.74	31.8479	19.4787	3.47654	12.9074
	Best	49.5446	31.5292	19.4787	2.90221	8.84848
	Average	119.40548	32.05106	NA	3.311416	9.969526
Gradient Descent With Momentum & Adaptive Learning Rate BP	1	48.9013	45.7247	32.103	30.3863	13.84
	2	48.4034	45.7479	30.5644	30.9805	14.6189
	3	46.3409	52.9423	31.3344	34.4808	15.1817
	4	80.1414	45.7962	33.6194	36.5738	14.845
	5	48.7483	45.8331	31.0961	32.8819	15.4983
	Best	46.3409	45.7247	30.5644	30.3863	13.84
	Average	54.50706	47.20884	31.74346	33.06066	14.79678
BFGS Quasi-Newton BP	1	MGR	Not Converge	0.58415	45.6261	45.6261
	2	Not Converge	0.14052	0.527411	47.6985	1.18517
	3	MGR	0.26583	Not Converge	53.3659	25.3654
	4	MGR	0.1569	0.41223	40.2356	45.6261
	5	MGR	0.14589	0.32698	55.2489	3.5698
	Best	NA	0.14052	0.32698	40.2356	1.18517
	Average	NA	NA	NA	48.435	24.274514

Table 5.7 Performance (MSE) of Cascade-forward Neural Network Trained by All Data (To Be Continued)

Trained By	Expmnt	10 neurons	20 neurons	30 neurons	40 neurons	50 neurons
Levenberg-Marquardt BP	1	11.0095	0.12563	0.457133	25.3659	0.12258
	2	0.25963	1.0537	0.77783	5.22845	3.25489
	3	Not Converge	0.14158	0.96582	1.66895	1.9083
	4	3.88118	3.5386	0.4598	14.5879	0.9854
	5	5.6985	Not Converge	0.456822	3.7892	8.5463
	Best	0.25963	0.12563	0.456822	1.66895	0.12258
	Average	NA	NA	0.623481	10.12808	2.963494
RPROP BP	1	MGR	45.7256	7.14256	45.6256	3.7072
	2	45.7125	6.0025	6.97868	45.6256	0.16287
	3	MGR	37.4563	45.6266	45.6264	8.6968
	4	45.9562	4.5238	9.8878	0.1125	5.3677
	5	MGR	MGR	MGR	6.7855	1.8456
	Best	45.7125	4.5238	6.97868	0.1125	0.16287
	Average	NA	NA	NA	28.75512	3.956034

Table 5.7 Performance (MSE) of Cascade-forward Neural Network Trained by All Data

The results show the MSEs are much greater than those obtained using the previous averaged values. If we use the original data without averaging, the noise is so significant that the network cannot learn well.

5.7.4 Conclusions

In this chapter we first used three-day averages to train the network. We reported the results of using 10, 20, 30, 40 and 50 neurons for every combination of the families

of NNs. The results cover three algorithms, seven training methods, and five choices of number of neurons. We ran each experiment five times, and thus we reported the results of 25 experiment results for each combination and 175 results for each of the three algorithms, the Feed-forward, the Elman and the Cascade-forward BP. Thus we totally reported the results of 525 experiments. Some of the experiments gave an excellent result in just several seconds. Some of them could give a good result only after several days of running the program. Then we tried another 525 experiments using the original data without averaging out the noise. The results show the network converges better when we use the average prices.

There are several criteria to measure the performance of the network: the error, the number of epoch and time spent in each epoch. In other words, a good network should lead to a small error. When the networks have the same error, the one that can reach the performance goal in fewer epochs is better. And finally, it is desirable that the scheme should need less time for each epoch.

The worst combination of the algorithms tried was the Cascade-forward algorithm trained by the Gradient Descent BP, as shown in Table 5.4. Independent of the number of neurons in the hidden layer, the network error increases, indicating that the search is going further and further from the global optimal.

The slowest combination is the Elman algorithm trained by the LM BP scheme. It needs more time in each epoch. Unfortunately, the results are not accurate. When the network uses 50 neurons in the hidden layer, it used about one hour to run 50 epochs, and thus took several days to finish 5,000 epochs. It means that, in each epoch, the network need much more computing. When the network is running, the indicator in Performance of Microsoft Windows Administrative Tools shows the CPU usage has reached the peak, 100%. To our great disappointment, the results were not good at all,

as shown in Table 5.3.

The best result obtained was for the Feed-forward algorithm trained by the LM BP, which not only converged each time, but also reach the performance goal, 0.001, within ten epochs. Furthermore, it needed less time for each epoch, indicating that the network does not need a lot of computing. The performance of the network was amazing.

The results of the 1050 experiments show that the best solution for our problem is the Feed-forward algorithm trained by LM BP. In the following forecasting task, we shall utilize this combination.

5.8 Stock Market Forecasting

In this section we show how we can achieve forecasting of the stock market when the Feed-forward NN is trained by the LM BP. In each case we used 50 neurons in the hidden layer.

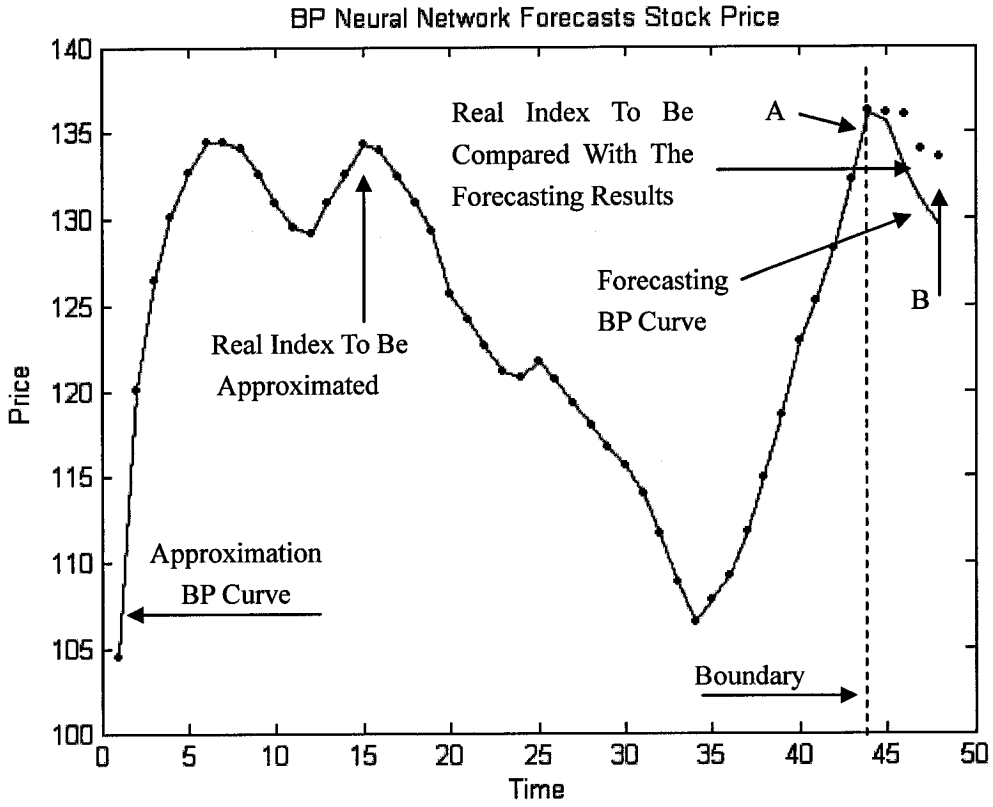

Figure 5.1 Forecasting Experiment 1 With SSE Composite Index

The experiment shown in Figure 5.1 uses data from 144 days of SSE Composite Index to train the network and predicts the next 12 days. To remove noise, we used the average of three days indices as each data point. Thus every black dot in figure 5.1 stands for the average of three-day indices. The black dotted line is a boundary which divides the approximation BP and forecasting curve. Every black dot in the figure is a date point. The black dots before the boundary are used for approximation, and those after the boundary are used to compare with the forecasting curve. The blue line before the boundary (including that day) is the approximated curve. The blue curve after the boundary is the forecasting result.

The network used to predict the stock market was the Feed-forward algorithm trained by LM BP. The performance goal was pre-defined to have the *MSE* of 0.001. The maximum number of epochs was set to be 5,000. The network stopped training when either of the two terminating conditions was satisfied.

At epoch 47, the performance goal was met. At that time, the performance indicator, the *MSE* had the value 0.000971188. In this experiment, the *MSE of the forecast* was 8.5044 indicating how close the prediction curve stays from the real dots, where

$$MSE\ of\ forecast = \frac{1}{n}\sum_{i=1}^{n}\left(real\ data - forecasting\ result\right)^2 \qquad (5\text{-}3)$$

We also defined two other variables:

$$Forecast\_Error = \frac{1}{n}\sum_{1}^{n}\left|\frac{forecast\ value - real\ value}{real\ value}\right| \qquad (5\text{-}4)$$

$$Decision\_Index = the\ last\ forecasting\ index - the\ last\ training\ index \qquad (5\text{-}5)$$

The *Forecast-Error* and *Approximation-Error* are recorded as percentages. If *Decision_Index* > 0, the market was expected to rise, and a buy or keep was preferred. If *Decision_Index* < 0, the forecasting result was lower than current index and a sale was preferred. In Figure 5.1, *Decision_Index* = *point B – point A* = -3.8696. The system recommended that the decision should be to "sell". Indeed, according to the real indices shown in Figure 5.1, this decision is correct because the index goes down after the boundary. Thus the performance of one testing Experiment is shown in the following table:

MSE of approximation	0.000971188
Approximation_Error (%)	0.0026189
MSE of forecast	8.5044
Forecast_Error (%)	0.64
Decision_Index	-3.8696
Decision	Sell

Table 5.8 Performance of Experiment 1

Figure 5.2 shows another experiment that also yielded a good result.

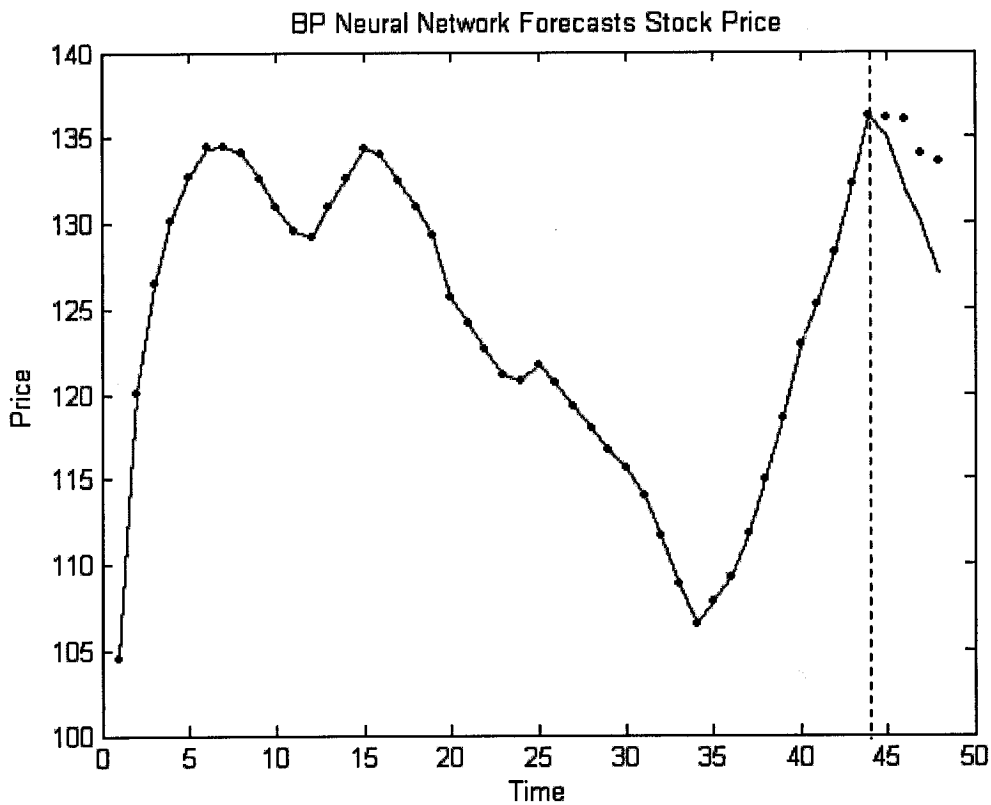

Figure 5.2 Another Forecasting Example using the SSE Composite Index

We ran the program 100 times to check the accuracy. Table 5.9 is the results of the 100 experiments.

Expt	*MSE of approximation*	*Approximation_Error (%)*	*MSE of forecast*	*Forecast_ Error (%)*	*Decision_ Index*	Decision
1	0.000971188	0.0026189	8.5044	0.64	-3.8696	sell
2	0.00050272	0.007037	18.6456	0.96	-6.4694	sell
3	0.000996462	0.0051759	4.0282	0.42	2.5598	buy
4	0.000967944	0.0074639	284.356	3.74	-23.712	sell
5	0.000998954	0.0078092	582.8281	5.24	-36.213	sell
6	0.000206141	0.0040914	332.1411	3.99	-26.6116	sell
7	0.000549716	0.0065828	369.6714	4.2	-27.8844	sell
8	0.000987737	0.010498	766.4074	6.02	-41.252	sell
9	0.00135831	0.0097487	13.0519	0.71	5.6983	buy
10	0.000320987	0.0044832	284.2529	3.73	-24.2752	sell
11	0.000192319	0.0027	202.7392	3.2847	-26.2736	sell
12	8.23265E-06	4.2292E-04	292.9452	3.8769	-31.6062	sell
13	0.000918057	0.0076	230.6848	3.4483	-28.5211	sell
14	9.00273E-06	5.0993E-04	218.5959	3.4308	-26.7871	sell
15	0.000315039	0.0032	272.7328	3.7387	-30.7585	sell
16	0.000771805	0.0065	264.668	3.7174	-29.569	sell
17	0.000302518	0.006	249.6954	3.614	-29.1027	sell
18	0.000143626	0.0024	262.3175	3.7059	-29.574	sell
19	0.000371102	0.0046	280.4923	3.9036	-29.7932	sell
20	0.000253652	0.0057	276.5219	3.8261	-30.1898	sell
21	1.10095E-06	2.2182E-04	321.9738	4.0657	-32.932	sell
22	0.000591382	0.0067	119.0871	2.5466	-20.5699	sell
23	0.000286184	0.004	345.3893	4.1469	-35.2246	sell
24	0.000581904	0.003	238.1812	3.5896	-27.3666	sell
25	0.00142808	0.0108	2.0086	0.3129	-0.4799	sell
26	2.84541E-06	1.0614E-04	240.9431	3.5557	-28.449	sell
27	0.000192319	0.0027	202.7392	3.2847	-26.2736	sell
28	8.23265E-06	4.2292E-04	292.9452	3.8769	-31.6062	sell
29	0.000997737	0.0079	57.4436	1.8555	-10.6145	sell
30	0.000918057	0.0076	230.6848	3.4483	-28.5211	sell
31	9.00273E-06	5.0993E-04	218.5959	3.4308	-26.7871	sell
32	0.000315039	0.0032	272.7328	3.7387	-30.7585	sell
33	0.000771805	0.0065	264.668	3.7174	-29.569	sell
34	0.000302518	0.006	249.6954	3.614	-29.1027	sell
35	0.000143626	0.0024	262.3175	3.7059	-29.574	sell

Table 5.9 Statistics of 100 Forecasting Experiments in which the correct decision was "to sell" in every case (To be continued)

Expt	*MSE of approximation*	*Approximation_Error (%)*	*MSE of forecast*	*Forecast_ Error (%)*	*Decision_Index*	Decision
36	0.000371102	0.0046	280.4923	3.9036	-29.7932	sell
37	0.000253652	0.0057	276.5219	3.8261	-30.1898	sell
38	1.10095E-06	2.2182E-04	321.9738	4.0657	-32.932	sell
39	0.000591382	0.0067	119.0871	2.5466	-20.5699	sell
40	0.000286184	0.004	345.3893	4.1469	-35.2246	sell
41	0.000581904	0.003	238.1812	3.5896	-27.3666	sell
42	0.00142808	0.0108	2.0086	0.3129	-0.4799	sell
43	2.84541E-06	1.0614E-04	240.9431	3.5557	-28.449	sell
44	1.9405E-06	2.4583E-04	238.6696	3.5355	-28.3318	sell
45	0.000999634	0.0078	363.0318	4.3334	-35.3964	sell
46	0.000676141	0.0059	235.7682	3.5643	-27.5497	sell
47	0.000973282	0.0131	293.7956	4.014	-30.802	sell
48	0.000999711	0.0078	384.0745	4.4502	-36.3955	sell
49	0.00099959	0.0079	357.7458	4.3109	-35.0651	sell
50	0.000765153	0.0067	253.8852	3.6592	-28.8873	sell
51	0.000511816	0.0066	239.2041	3.5135	-28.8723	sell
52	0.000987144	0.0069	220.8179	3.3712	-28.0561	sell
53	0.00110084	0.0101	11.6352	0.8704	-7.2047	sell
54	1.85713E-05	0.0012	269.7271	3.7255	-30.3369	sell
55	0.00151232	0.0111	0.8125	0.2071	-1.4324	sell
56	0.000995975	0.0079	416.5617	4.6123	-37.9732	sell
57	0.000994725	0.0029	3.0318	0.3915	-0.0182	sell
58	0.000999419	0.0089	402.357	4.5582	-37.1197	sell
59	0.000997325	0.0111	264.9547	3.6799	-30.8571	sell
60	0.000933036	0.0087	13.9923	0.9566	-6.2769	sell
61	0.00136043	0.01	1.0095	0.2323	-1.2767	sell
62	0.000187438	0.0039	283.6536	3.8534	-30.2664	sell
63	1.52049E-06	2.2752E-04	296.0413	3.9265	-31.1249	sell
64	0.000996149	0.007	239.4319	3.4887	-29.2186	sell
65	0.000999688	0.0098	430.1725	4.7023	-38.3772	sell
66	0.00075072	0.0078	384.4608	4.5609	-35.2359	sell
67	0.000989914	0.0084	13.498	0.9344	-7.6308	sell
68	0.00138051	0.0102	2.1607E+03	10.9332	-76.2352	sell
69	8.48655E-05	0.0024	259.3162	3.68	-29.4852	sell
70	0.000668794	0.0078	238.0095	3.5438	-28.3538	sell

Table 5.9 Statistics of 100 Forecasting Experiments in which the correct decision was "to sell" in every case (To be continued)

Expt	*MSE of approximation*	*Approximation_Error (%)*	*MSE of forecast*	*Forecast_ Error (%)*	*Decision _Index*	Decision
71	0.000988986	0.012	244.6275	3.5816	-28.531	sell
72	0.00135408	0.01	3.6697	0.4897	-5.2765	sell
73	5.12069E-05	0.0034	249.592	3.5644	-29.827	sell
74	0.00143741	0.0108	22.8985	1.203	-9.2313	sell
75	0.000999838	0.0099	436.3451	4.7374	-38.61	sell
76	0.000998511	0.008	372.9096	4.3889	-35.87	sell
77	0.000995189	0.0074	207.1241	3.2818	-27.058	sell
78	0.000282872	0.0039	207.5257	3.3041	-26.779	sell
79	0.000902801	0.0117	627.3602	5.6615	-45.827	sell
80	0.000359898	0.005	255.3077	3.6564	-28.945	sell
81	0.000192319	0.0027	202.7392	3.2847	-26.274	sell
82	8.23265E-06	4.2292E-04	292.9452	3.8769	-31.606	sell
83	0.000997737	0.0079	57.4436	1.8555	-10.615	sell
84	0.000918057	0.0076	230.6848	3.4483	-28.521	sell
85	9.00273E-06	5.0993E-04	218.5959	3.4308	-26.787	sell
86	0.000315039	0.0032	272.7328	3.7387	-30.759	sell
87	0.000771805	0.0065	264.668	3.7174	-29.569	sell
88	0.000302518	0.006	249.6954	3.614	-29.103	sell
89	0.000143626	0.0024	262.3175	3.7059	-29.574	sell
90	0.000371102	0.0046	280.4923	3.9036	-29.793	sell
91	0.000253652	0.0057	276.5219	3.8261	-30.19	sell
92	1.10095E-06	2.2182E-04	321.9738	4.0657	-32.932	sell
93	0.000591382	0.0067	119.0871	2.5466	-20.57	sell
94	0.000286184	0.004	345.3893	4.1469	-35.225	sell
95	0.000581904	0.003	238.1812	3.5896	-27.367	sell
96	0.00142808	0.0108	2.0086	0.3129	-0.4799	sell
97	2.84541E-06	1.0614E-04	240.9431	3.5557	-28.449	sell
98	1.9405E-06	2.4583E-04	238.6696	3.5355	-28.332	sell
99	0.000999634	0.0078	363.0318	4.3334	-35.396	sell
100	0.000676141	0.0059	235.7682	3.5643	-27.55	sell
Avg.	0.000603213	0.005631106	259.768905	3.365636	-26.199	NA

Table 5.9 Statistics of 100 Forecasting Experiments in which the correct decision was "to sell" in every case.

Table 5.9 shows that the correct decision was recommended 98% of the time in these experiments.

The points where the trend reverses are considered the critical points. To study these further, we also moved the time window to test the non-critical points using the Feed-forward network trained by LM BP algorithm.

We started from the 30th point and moved the time window. We used the data of u points (which represents closing prices of $3u$ days) to train the network, and then attempted to forecast the prices for the next 12 days. This window u was increased from 30 to 47 points. A few of the graphs are shown below, and the results are thereafter tabulated. Figure 5.3 uses 30 points, which represents 90 closing prices, to train the network, so the time window of Figure 5.3 is 90 days. The next 4 points, which represents 12 closing prices, are forecasted. And then we increase the time window by adding a point each time.

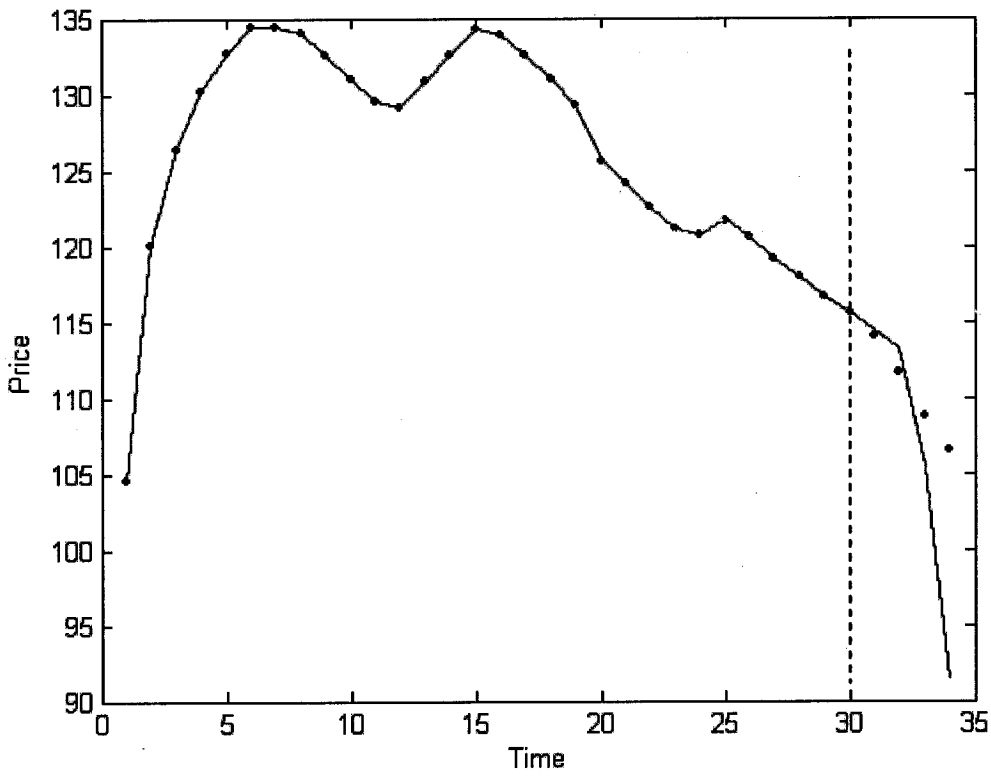

Figure 5.3 Forecasting results of prices from day 93 to 102

The dotted vertical line shows where the training was stopped.

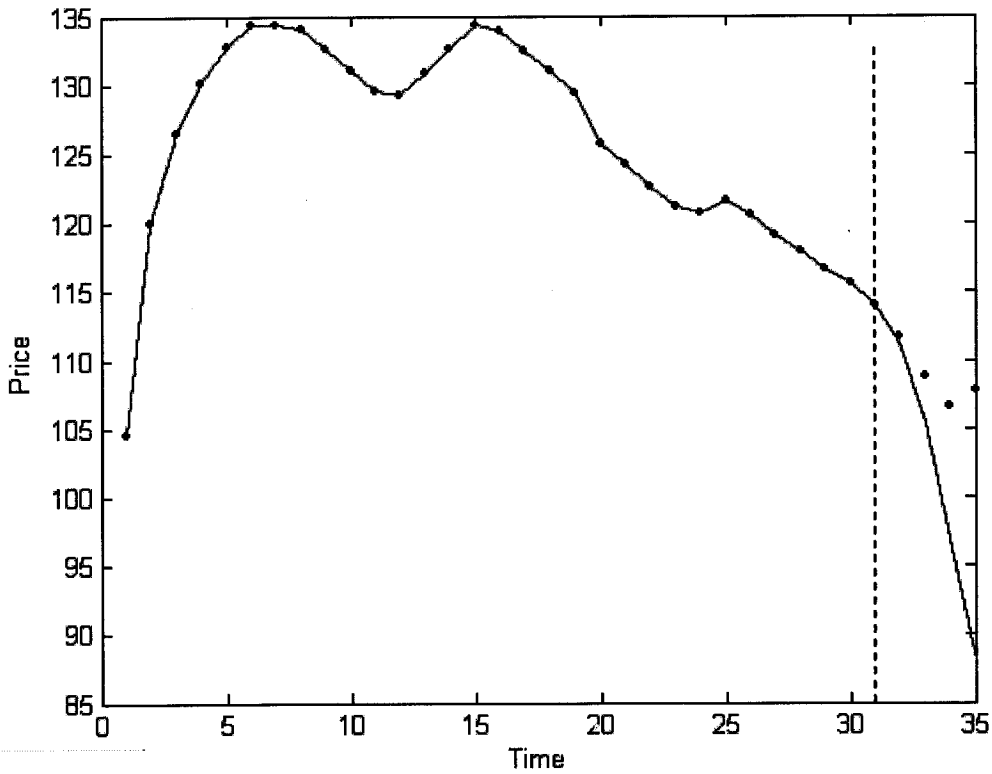

Figure 5.4 Forecasting results of prices from day 96 to 105

The dotted vertical line shows where the training was stopped.

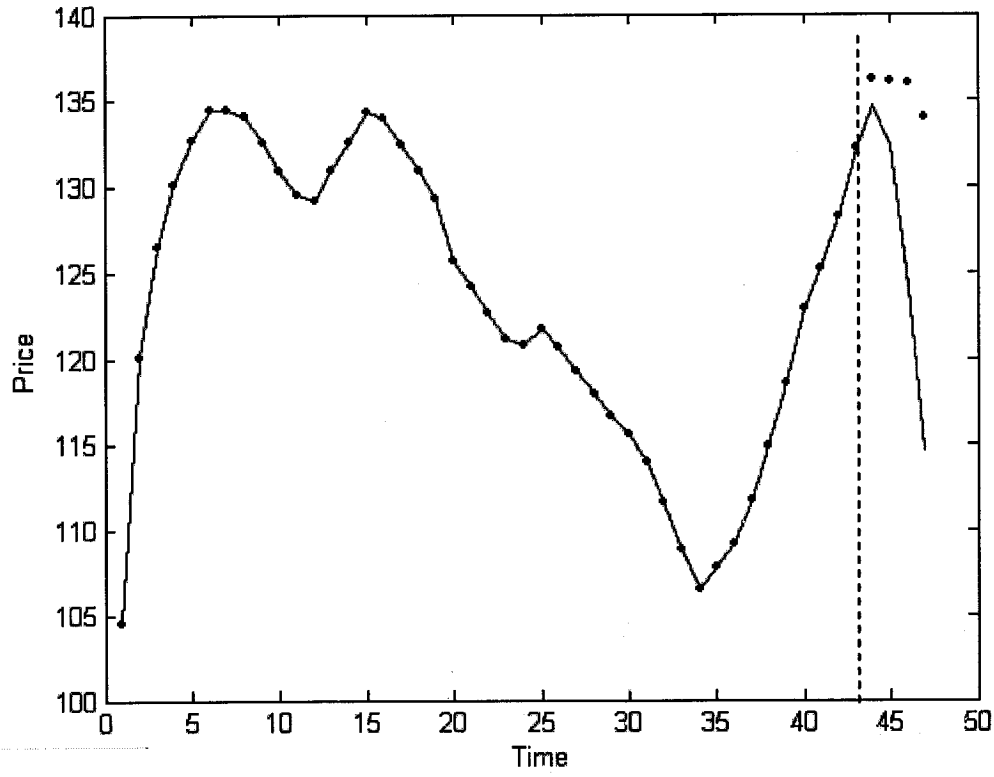

Figure 5.5 Forecasting results of prices from 132nd to 141st days

5.9 Conclusion

In this section, we moved time window and used Feed-forward network trained by LM to forecast stock market. From Table 5.10, we see that between days 93 and 141, we forecast the decisions 14 times. Ten of the forecasts were correct, leading to

an accuracy of 71.43%.

Time Window (Points)	Forecasting Day	*MSE* of approximation	*Approximation_ Error(%)*	*MSE* of forecast	*Forecast_ Error(%)*	*Decision_ Index*	*Decision*	Right or Wrong
30	93-102	0.000648607	6.7445E-03	60.6533	1.59	-15.1184	Sell	Right
31	96-105	1.26222e-006	2.4255e-004	123.278	2.57	-19.7475	Sell	Right
32	99-108	4.06664e-005	1.7778e-003	52.968	1.68	-13.7042	Sell	Right
33	102-111	4.14171e-005	1.1988e-003	652.5934	7.34	32.5365	Buy	Right
34	105-114	0.000137921	3.6943e-003	239.6668	3.49	-27.5193	Sell	Wrong
35	108-117	1.37042e-005	1.4865e-003	45.6477	1.66	10.7631	Buy	Right
36	111-120	2.82997e-005	1.2386e-003	194.8289	3.13	24.169	Buy	Right
37	114-123	0.000756624	7.1726e-003	572.9131	5.26	-17.358	Sell	Wrong
38	117-126	0.000618457	6.5856e-003	105.388	2.43	15.8111	Buy	Right
39	120-129	0.000600514	8.5564e-003	60.4543	1.67	13.0297	Buy	Right
40	123-132	0.000369886	4.4451e-003	91.1834	1.95	16.0308	Buy	Right
41	126-135	0.000318688	6.1495e-003	48.9658	1.58	8.861	Buy	Right
42	129-138	1.41682e-006	4.0638e-004	77.1116	1.95	-13.2787	Sell	Wrong
43	132-141	0.000992435	1.0469e-002	9.1216	0.71	-4.2045	Sell	Wrong

Talbe 5.10 Moving Window Forecast Results

We also demonstrated that the Feed-forward algorithm trained by LM BP yielded an excellent result. It not only approximated the true curve very closely, but also gave an outstanding forecasting result. The possibility of the right decision made by the network was 98%. The network learned each piece of information hidden behind of the data in the market adequately. We thus conclude that, in some circumstances, the rules behind of the data can be learned and exploited.

CHAPTER VI

CONCLUSIONS AND PERSPECTIVES

In this thesis, we presented an approach that utilized NN techniques to predict the financial market. We developed an efficient three-layer neural network with revised BP algorithms to forecast the stock market. We compared several supervised algorithms (Feedforward, Cascade-forward and Elman) involving seven training methods (Gradient Descent, Gradient Descent With Momentum, Gradient Descent With Adaptive Learning Rate, Gradient Descent With Momentum and Adaptive Learning Rate, Broyden-Fletcher-Goldfarb-Shanno (BFGS) Quasi-Newton, Levenberg-Marquardt and RPROP (Resilient Propagation) BP) to figure out the algorithm that can generate the best prediction result.

We used codes from Matlab to save time. However, in order to run each of the seven algorithms, we had to develop our own codes and input/output interfaces to evaluate each neural structure.

The data of the stock market used for this purpose was taken from two Chinese stock exchanges, namely the Shanghai Stock Exchange Composite Index (SSE composite index) and the Shenzhen Stock Exchange Component Index (SZSE

Component Index).

In order to reduce the impacts of noise (caused by occasional factors) to the minimum and to smooth the curve, we calculated the average of every three closing prices and used these as the data points.

Other kinds of data, such as the opening price, the daily high, the daily low and volume, were discarded because they had already been considered in the closing price. Thus, all kinds of data and information (including the open price, the daily high, the daily low and volume) that affect the closing price were reflected in the fluctuation of the closing price, rendering the rest of the data redundant.

In conclusion, the Feed-forward algorithm trained by the LM BP yields an excellent result. It not only approximates the curve very closely, but also gives an outstanding forecasting result. The correct forecast was made 98% of the time.

Our work shows that the rules of fluctuation hidden behind of the stock prices are possible to be learned and disclosed, at least in some circumstances. There are billions of affecting factors that have impacts on the stock prices. The stock prices fluctuate every second. Although we believe there are some relationship between the factors and prices, we cannot find out any closed-form formula. But the neural network successfully learned the rules in our experiments and gave out satisfying results.

This work was achieved for one set of data, so the results as well as the conclusions presented in Chapter V are strongly dependent on the input/output data set. We are aware that another set of data could reach to different conclusions and recommendations regarding the market forecast. Our work might be considered a kind of confirmation that some rules behind of the market can be learned and that the financial market is possible to be forecasted.

Future Work

The attempt by scientists to forecast the stock market never stops. But the work is never simple. There are still some defects of the NN prediction.

The data used to train the network does not contain every piece of required information, and thus is not efficient enough. If some affecting factors occur in the future which did not exist in the past, there is no way any system can forecast, because the knowledge did not exist in the data of the past. For example, the twin-towers in New York were attacked on Sep. 11, 2001, causing the stock market to crash "suddenly". This knowledge did not exist in the data before Sep. 11, 2001, and thus could not be learned. There is no way a NN could forecast the trend after Sep.11, 2001 even if it was trained by the data of the past ten years.

Different pieces of data have different features, and thus different error characteristics. The combination of the Feed-forward and LM BP is not always the best. The best combination changes with the input data. The selection of the best combination is not easy because it involves so many experiments. Furthermore, as the input data gets larger, the error surface becomes much more complex. It is hard to find the global minimum in such a complex error surface.

It is not true that the quality of the results increases with the amount of training data. How close is the relation between the data of ten years ago and the data over the next few hours? Therefore, the size of the data should be decided before they are input to the network. Unfortunately, there is currently no effective way (or formula) to decide the size of the time window. It depends on one's personal experience, and the knowledge of the economy and stock market. Different people have different points of

view. In this respect, the personal subjective issues cannot be eliminated.

Some of the factors affecting the market in the past could disappear in the future. Even if they still exist, they may not have the same strong impact on the market. Instead, new affecting factors could emerge with time. Thus, a network trained by the past data only tells us something about the recent future. It has too little knowledge about the distant future. Thus it is likely that the neural networks could only be exploited by speculators who are not real investors. The neural network leads the user to a hint of the future when it concerns several days, but a real investor is always thinking about the future as it concerns several years. A smart investor never speculates, just like a smart person never expects to become rich by gambling.

The neural network might help a speculator obtain some profit, but the profit will not be much since it is always a result of short-term speculation. This profit is often counteracted by the existence of commission. The share price of Microsoft, the world's leading software company, has soared as high as two hundred times of what it was in the year when Microsoft began to be listed in NASDAQ, namely in 1986. The real investors who held the shares for more than ten years would have gained millions of dollars if they invested 10,000 dollars in 1986. Such a profit cannot be made by a speculator.

A good choice of today might be a crazy choice in the long run. Let us suppose that the speculators clear off their shares, and avoid the loss in the next week. But if they look back one year later, they will realize they did make a bad choice after all. The speculators cleared their shares off in the very beginning, as indicated by the red circle in Figure 6.1. But the index of any day in the next year is much higher than what is indicated by the red circle.

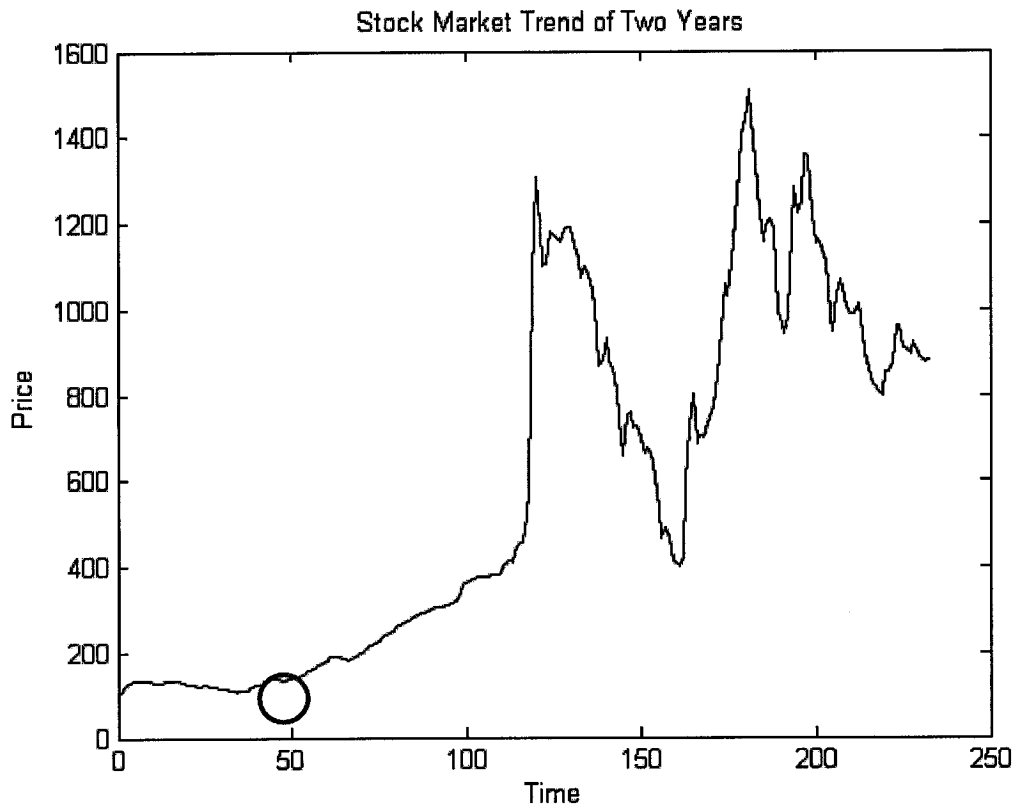

Figure 6.1 Stock Market Trend of Two Years

All these points can be used as arguments to show the defects of neural network forecasting methods. The future work should focus on these defects, and efforts should be made to overcome, or at least reduce the effects of these issues.

Short-time prediction is a kind of speculation. A speculator is not a real investor. Our purpose is to design a model that can help the real investors. Therefore, our future work will focus on forecasting the distant future, instead of immediate future.

In the future, we intend to work on how to choose the best combination of algorithm and training method, so as to avoid a shortsighted decision. This thesis

chose the best combination by experiments and trials. We will continue our work to find out a more rigorous approach to select the best one independently of the input/output set of data presented to the neural structure during its training.

References

[1] *Bank of England.* Encyclopædia Britannica. Retrieved May 4, 2004, from Encyclopædia Britannica Premium Service. <http://www.britannica.com/eb/article?eu=33223>

[2] *USX Corporation.* Encyclopædia Britannica. Retrieved May 4, 2004, from Encyclopædia Britannica Premium Service. <http://www.britannica.com/eb/article?eu=76496>

[3] *London Stock Exchange.* Encyclopædia Britannica. Retrieved May 4, 2004, from Encyclopædia Britannica Premium Service. <http://www.britannica.com/eb/article?eu=50003>

[4] *New York Stock Exchange.* Encyclopædia Britannica. Retrieved May 4, 2004, from Encyclopædia Britannica Premium Service. <http://www.britannica.com/eb/article?eu=56928>

[5] *Dow Jones Average.* Encyclopædia Britannica. Retrieved May 4, 2004, from Encyclopædia Britannica Premium Service. <http://www.britannica.com/eb/article?eu=31595>

[6] Overview page of Shanghai Stock Exchange (SSE) official web site. http://www.sse.com.cn/sseportal/en_us/ps/home.shtml

[7] Overview page of Shenzhen Stock Exchange (SZSE) official web site. http://www.szse.cn/main/en/catalog_1378.aspx

[8] J.D. Schwager and S.C. Turner, *A Study Guide for Fundamental Analysis (Schwager on Futures)*, John Wiley & Sons (April 1996)

[9] G.A. Fontanills and T. Gentile, *The Stock Market Course*, John Wiley & Sons; 1st edition (February 2001)

[10] Lexico Publishing Group, LLC <http://dictionary.reference.com/search?q=stock

[11] J.B. Little and P.A. Samuelson, *Bonds, Preferred Stocks and the Money Market (Basic Investors Library)*, Chelsea House Pub (April 1988)

[12] E.L. Smith, *Common Stocks as Long Term Investments*, Kessinger Publishing Company; (June 2003)

[13] WebFinance Inc., financial glossary online <http://www.investorwords.com>.

[14] Equade Internet Ltd, investing education site <http://www.investopedia.com>.

[15] M.C. Thomsett, *Mastering Fundamental Analysis*, Dearborn Trade Publishing (August 1998)

[16] J.F. Childs, *Earnings per share and management decisions*, Prentice-Hall (1971)

[17] BELCO Holding Limited (BHL), holding company for Bermuda Electric Light Company Limited, Bermuda Gas and Utility Company Limited, BELCO Energy Services Company, and BELCO Properties Limited <http://www.belcoholdings.bm>.

[18] M.L. Leibowitz, S. Kogelman, *Franchise Value and the Price/Earnings Ratio (The Research Foundation of Aimr and Blackwell Series in Finance)*, Institute of Chartered Financial Analysts Research Foundation, J.W. Peavy. Publisher: Association for Investment Management & Research;

(September 2000)

[19] J.D. Schwager, S.C. Turner, *Futures, Textbook and Study Guide : Fundamental Analysis*, John Wiley & Sons; Package edition (May 1997)

[20] J.D. Schwager, *A Complete Guide to the Futures Markets : Fundamental Analysis, Technical Analysis, Trading, Spreads, and Options*, John Wiley & Sons (June 1984)

[21] Giga-Tronics Inc.
<http://finance.yahoo.com/q?s=GIGA&d=t>.

[22] The CompuDyne Corporation
<http://finance.yahoo.com/q?s=CDCY&d=t>.

[23] Finance channel of www.netease.com.
<http://quote.stock.163.com/stock/show_custom.php?code=000046>.

[24] Yahoo! finance channel.
<http://cn.finance.yahoo.com/q?s=000665.SZ&d=c&k=c1&a=v&p=s&t=6m&l=off&z=m&q=l>.

[25] Yahoo! finance channel.
<http://cn.finance.yahoo.com/q?s=000020.SZ&d=c&k=c1&a=v&p=s&t=1y&l=off&z=m&q=l>.

[26] Yahoo! finance channel:
<http://cn.finance.yahoo.com/q?m=z&s=000019&d=c&k=c1&t=1y&a=v&p=s&l=off&z=m&q=l>
<http://cn.finance.yahoo.com/q?m=z&s=000033&d=c&k=c1&t=1y&a=v&p=s&l=off&z=m&q=l>
<http://cn.finance.yahoo.com/q?m=z&s=000035&d=c&k=c1&t=1y&a=v&

p=s&l=off&z=m&q=l>
<http://cn.finance.yahoo.com/q?m=z&s=000056&d=c&k=c1&t=1y&a=v&p=s&l=off&z=m&q=l>

[27] Yahoo! finance channel. <http://cn.finance.yahoo.com/q?s=SINA&d=c&k=c2&a=v&p=m10,e10&t=3m&l=off&z=m&q=l>.

[28] Yahoo! finance channel <http://cn.finance.yahoo.com/q?s=MSFT&d=c&k=c2&a=v&p=e50,e20&t=3m&l=off&z=m&q=l>.

[29] Business Directory aimed at the information needs of Top and Senior Executives <http://www.valuebasedmanagement.net>.

[30] *5-y-o Tia beats the stock market game*, Jamaica Gleaner article, Sept. 22, 2002 <http://www.jamaica-gleaner.com/gleaner/20020922/business/business3.html>.

[31] S. White, *A Brief History of Computing - Complete Timeline*, <http://www.ox.compsoc.net/~swhite/history/timeline.html>

[32] *Japanese 'Computenik' Earth Simulator shatters US supercomputer hegemony*, Primeur Magazine, Tokyo, April 20, 2002

[33] J. Nolte, J.B. Angevine, *The Human Brain: in Photographs and Diagrams*, Mosby; 2nd edition (June 2000)

[34] *Neuron Structure in 3D*, drawn for Medica vol. 4. © Copyright Duncan Baird Publishers, 1998. <http://www.scientific-art.com/portfolio%20medicine%20pages/neuronst.

htm>.

[35] S.J. Dearmond, M.M. Fusco, M.M. Dewey, *Structure of the Human Brain: A Photographic Atlas*, Oxford Press; 3rd edition (May 1989)

[36] A. Peters, S.L. Palay, H.D. Webster, H.D. Wabster, *Fine Structure of the Nervous System: Neurons and Their Supporting Cells*, Oxford Press; 3nd edition (January 1991)

[37] Q.J. Zhang, and Gupta K.C., *Neural Networks for RF and Microwave Design*, Artech House, Norwood, MA, (2000).

[38] S. Haykin, *Neural Networks: A Comprehensive Foundation*, IEEE Press, New York (1994).

[39] B. Chattaraj, *Computer Aided Electromagnetic Design Based On Neural Models*, Master Thesis, Carleton university, Ottawa, ON, Canada, 2002.

[40] F. Scarselli and A.C. Tsoi, "Universal approximation using feedforward neural networks: a survey of some existing methods, and some new results," *Neural Networks*, vol. 11, pp. 15-37, 1998.

[41] B. Widrow and M.A. Lehr, "30 years of adaptive neural networks: Perceptron, Madaline, and Backpropagation," *IEEE Proc.*, vol. 78, pp. 1415-1442, 1990.

[42] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control Signals Systems*, vol. 2, pp. 303-314, 1989.

[43] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359-366, 1989.

[44] M.J.D. Powell, "Radial basis functions for multivariate interpolation: a review," in *Algorithms for Approximation*, J.C. Mason and M.G. Cox, Eds. Oxford, UK: Oxford University Press, 1987, pp. 143-167.

[45] J. Park and I.W. Sandberg, "Universal approximation using Radial-Basis-Function networks," *Neural Computation*, vol. 3, pp. 246-257, 1991.

[46] W. Kaminski and P. Strumillo, "Kernel orthonormalization in radial basis function neural networks," *IEEE Trans. Neural Networks*, vol. 8, pp. 1177-1183, 1997.

[47] V.K. Devabhaktuni, *Neural Network Based Microwave CAD: Towards Automatic Model Generation*, Ph.D. Thesis, Carleton university, Ottawa, ON, Canada, 2002.

[48] P.M. Watson, K.C. Gupta, and R.L. Mahajan, "Development of knowledge based artificial neural network models for microwave components," *IEEE Intl. Microwave Symp. Digest*, pp. 9-12, Baltimore, MD, 1998.

[49] F. Wang and Q.J. Zhang, "Knowledge based neural models for microwave design," *IEEE Trans. Microwave Theory Tech.*, vol. 45, pp. 2333-2343, 1997.

[50] S.E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," *Advances in Neural Information Processing,* vol. 2, D.S. Truretzky, Eds. San Mateo, CA: Morgan kauffman, 1990, pp. 524-532.

[51] J.N. Hwang, S.S. You, S.R. Lay, and I.C. Jou, "The Cascade-Correlation learning: a projection pursuit learning perspective," *IEEE Trans. Neural Networks*, vol. 7, pp. 278-289, 1996.

[52] P. McCollum, *An Introduction to Back-Propagation Neural Networks*, a newsletter of Seattle Robotics Society. http://www.seattlerobotics.org/encoder/nov98/neural.html

[53] R.D. Reed and R.J. Marks. II . *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. Publisher: The MIT Press (1999)

[54] H. Liqun, Artificial Neural Network Theorem, Design and Implementation. Publisher: Chemical Industry Press (Jan. 2002)

[55] B.D. Ripley, University of Oxford, *Pattern Recognition and Neural Networks*. Publisher: Cambridge University Press (1996)

[56] D.T. Pham and X. Liu, *Neural Networks for Identification, Prediction and Control*. Publisher: Springer (1995)

[57] A. Van den Bosch, T. Weijters, J. Van den Herik, *Scaling Effects with Greedy and Lazy Machine-Learning Algorithms*, Department of Computer Science, University of Maastricht

[58] M. Giudici, F. Queirolo and M. Valle, *Evaluation of Gradient Descent Learning Algorithms With Adaptive And Local Learning Rate For Recognizing Hand-Written Numerals*. European Symposium on Artificial Neural Networks, Bruges, Belgium, April 2002

[59] P.E. Gill, W. Murray and M.H. Wright, *Practical* Optimization. Publisher: Academic Press (January 28, 1982)

[60] http://mathworld.wolfram.com/Jacobian.html, a comprehensive and interactive mathematics encyclopedia intended for students, educators, math enthusiasts, and researchers.

[61] M. Riedmiller, *Rprop-Description And Implementation Details* (Jan. 1994)

[62] M. Riedmiller and H. Braun, *A Direct Adaptive Method For Faster Backpropagation Learning: The RPROP Algorithm.*

[63] W. Xin, Z. Lu, W. Danli, X. Xiaoying. *Matlab Neural Network Software Design,* by Publisher: Science Press (September 2002).

[64] Z. Wang, M.C.E. Yagoub. *Computer Manipulation & Stock Price Trend Forecast* (ISBN: 7-5316-2509-1/G·1925), Heilongjiang Education Press, China, in press.

[65] Z. Wang, M.C.E. Yagoub, J. Oommen. *Prediction of Stock Market Prices Using Neural Network Techniques*, 6th WSEAS International Conference on Applied Mathematics, Corfu Island, Greece, to be submitted.