Financial Time Series Forecasting
Using Neural Networks

Submitted in Total Fulfilment of the Requirements of
the Degree of Master by Research

Weiduo Zhang

Faculty of Science, Engineering and Technology
Swinburne University of Technology
Hawthorn, VIC 3122, Australia

2019
This page is intentionally left blank
Abstract

The modelling and prediction of stock prices and exchange rates are very attractive research areas due to their importance in managing and analysing financial systems and markets. In practice, the well-developed financial models can help governments to develop the better policies and make smarter decisions to stabilise and stimulate economic developments. However, due to the complex and nonlinear properties, the modelling and prediction of financial systems are often extremely difficult and challenging. Although many modelling and prediction methods have been developed, these methods are not very efficient in applications due to their insufficient representation of complex system dynamics. In recent years, the researchers have been using artificial neural networks (ANNs) to model foreign currency exchange rates and stock prices. This is because neural networks have the capability of powerful universal approximation and parallel processing. It has been shown that the well-trained ANN models are capable of capturing the complex dynamic pattern features embedded in the financial data and exhibit strong robustness against the noises and disturbances in financial data.

In this thesis, traditional time series methodologies are briefly reviewed, and their characteristics for financial data modelling and prediction are discussed. Then, the basic structures of single hidden layer feedforward neural networks (SLFNs) and a few corresponding training methods are studied. In particular, the state-of-the-art back-propagation (BP) and the extreme learning machine (ELM) training algorithms are described in detail. In view of the existing neural predictive models that are lack of the robustness against noises and uncertainties, In this research work, the development of robust neural models for eliminating the effects of disturbances and uncertainties are studies. The main contributions of this thesis are listed as follows.
Firstly, an SLFN model with the tapped-delay-lines at the input layer is developed to model and forecast a class of dynamical exchange rate systems. The purpose of using the tapped-delay-lines at the input layer is to produce the temporal input signals to ensure that the dynamical relationship between the system inputs and outputs can be sufficiently modelled by using the SLFN model. The technical indicators, used as the model inputs, describe the market characteristics, such as short-term and long-term trends and market momentums. A momentum-based back-propagation (BPM) training algorithm is then used to train the SLFN model in the sense that both the local minimum problem can be avoided and robustness against the disturbances can be achieved through training. The simulations are conducted to show the modelling and forecasting performances in comparison with the existing linear time series models.

Secondly, the SLFN model trained with the regularized extreme learning machine (R-ELM) is proposed for predicting the closing price of the Standard & Poor's 500 (S&P 500) stock index. It is seen that ELM-based neural models exhibit fast learning speed and excellent generalization capability, which are desirable for financial forecasting applications. However, the performance of the neural models trained with ELM is often affected by both internal and external disturbances. In this work, we will use the R-ELM to train the output weights in the sense that the robustness of the neural model against the disturbances can be improved and also the forecasting accuracy of the model can be enhanced. The experiment results show that the neural model trained with R-ELM exhibits the strong robustness and excellent prediction performance compared to the SLFN model trained with the conventional ELM.
Declaration

This is to certify that:

1. This thesis contains no material which has been accepted for the award to the candidate of any other degree or diploma, except where due reference is made in the text of the examinable outcome.

2. To the best of the candidate’s knowledge, this thesis contains no material previously published or written by another person except where due reference is made in the text of the examinable outcome.

3. The work is based on joint research and publications; the relative contributions of the respective authors are disclosed.

Weiduo Zhang, 2019
This page is intentionally left blank
Acknowledgements

Above all, I would like to express my sincere gratitude to my principal supervisor, Professor Zhihong Man, and my co-supervisor Dr Jinchuan Zheng. They both offered me great helps and academic suggestions during my study. In particular, I am deeply grateful to Professor Zhihong Man, for his most valuable guidance, profound knowledge, insightful advice, and great support. I have been inspired by and have benefitted from his high academic attainments. Without his continuous encouragement, I would have never achieved this thesis.

I would also like to thank Associate Professor Zhenghua Zhou. I really appreciate his willingness to review the manuscripts of my thesis and make valuable comments and suggestions on each page. His academic guidance has been very helpful for my thesis writing.

I would like to thank my colleagues in Professor Man’s research group: Mengqiu Tao, Zhenyi Shen, Pengcheng Wang, Yew-Wee Wong, Lingkai Xing, Hong Du, Linxian Zhi, Hai Wang and Wenjie Ye, not only for all their useful helps and suggestions but also for being there to listen when I needed an ear. It was a pleasure working with them all. Thanks are also due to many friends in my life, including Luying Ding, Lidong Cui, Chengda Lu, Yiliu Sun, Chaoyi Wu, Chuanji Tian, Tuo Ji, Zhe Sun and Jing Xun, for their friendships, support and help.

I would like to express my deep appreciation to my father Huanman Zhang and mother Yi Jin. My father has given me strong support and help during my study. Furthermore, during my childhood, my father taught me lessons about life, which are not only helpful for my study, but also helpful during my life.
This page is intentionally left blank
## Contents

Chapter 1  Introduction .................................................................................... 1  
1.1 Background ............................................................................................. 1  
1.2 Motivations and contributions ................................................................. 4  
1.3 Overview of the thesis ............................................................................. 5  

Chapter 2  Literature Review ........................................................................ 7  
2.1 Introduction ............................................................................................. 7  
2.2 Time series models for financial forecasting ........................................... 9  
2.2.1 ARIMA models ..................................................................................... 9  
2.2.2 GARCH models .................................................................................. 11  
2.3 Basics of ANNs ....................................................................................... 13  
2.3.1 A single neuron ................................................................................... 14  
2.3.2 An SLFN structure ............................................................................. 15  
2.3.3 Learning algorithms for ANNs ........................................................... 16  
2.4 ANNs for financial forecasting ................................................................. 24  
2.4.1 ANNs for exchange rate modelling and forecasting ......................... 25  
2.4.2 ANNs for stock price forecasting ....................................................... 26  
2.5 Conclusion ............................................................................................... 28  

Chapter 3  An SLFN Model Trained with BPM for Exchange Rate Forecasting ........................................................................ 29  
3.1 Introduction ............................................................................................. 29  
3.2 BPM ......................................................................................................... 31  
3.3 An SLFN model for exchange rate forecasting ......................................... 36  
3.4 Simulations and experimental results ....................................................... 44  
3.4.1 Data collection and pre-processing ..................................................... 44
3.4.2 Experiment results ................................................................. 46
3.5 Conclusion .................................................................................. 52

Chapter 4 An SLFN Model with R-ELM Algorithm for Stock Price Forecasting ................................................................. 53
  4.1 Introduction ................................................................................ 53
  4.2 R-ELM ..................................................................................... 55
  4.3 An SLFN model for forecasting stock closing price ................. 59
  4.4 Simulations and experimental results ........................................ 64
    4.4.1 Data description ................................................................. 64
    4.4.2 Experiment results ............................................................ 66
  4.5 Conclusion ............................................................................... 76

Chapter 5 Conclusion and Future Work ............................................ 77
  5.1 Summary and conclusions ...................................................... 77
  5.2 Future work ........................................................................... 78

Bibliography .......................................................................................... 79
List of Figures

Figure 2.1   A single neuron [46] ................................................................. 14
Figure 2.2   The structure of an SLFN model .............................................. 16
Figure 2.3   An SLFN trained with the BP algorithm ................................. 17
Figure 2.4   An SLFN trained with the ELM algorithm ............................... 21
Figure 3.1   The structure of the SLFN model ............................................. 32
Figure 3.2   A RNN for exchange rate forecasting ...................................... 38
Figure 3.3   An SLFN model for exchange rate forecasting ......................... 39
Figure 3.4   Diagram of the modified SLFN trained with the BPM ............. 43
Figure 4.1   An SLFN model ..................................................................... 56
Figure 4.2   A modified SLFN model ........................................................ 61
Figure 4.3   Testing RMSEs of different combinations of $N, \gamma$ ............. 67
Figure 4.4   Testing RMSEs of different regularization parameters ............ 68
Figure 4.5   Training performances of the SLFN model trained with the R-ELM ................................................................. 70
Figure 4.6   Testing performances of the SLFN model trained with the R-ELM ................................................................. 70
Figure 4.7   RMSEs of the SFLN model trained with the ELM and the R-ELM with different numbers of hidden node .......................... 72
Figure 4.8   MAEs of the SFLN model trained with the ELM and the R-ELM with different numbers of hidden node ....................... 72
Figure 4.9   Forecasting performance of the R-ELM-based SLFN model ........ 75
Figure 4.10  Forecasting performance of BPM-SLFN model ...................... 75
This page is intentionally left blank
### List of Tables

| Table 3-1 | Performances of the SLFN model with different input lengths | 47 |
| Table 3-2 | Performances of the SLFN model with different hidden node numbers | 49 |
| Table 3-3 | Performances of the SLFN model with different learning rates | 50 |
| Table 3-3 | Performances of the SLFN model with or without momentum term | 51 |
| Table 3-4 | Performance comparison of the proposed model with other models | 51 |
| Table 4-1 | The 39 financial factors related to the S&P 500 index | 65 |
| Table 4-2 | Testing RMSEs of the SLFN model trained with the R-ELM | 69 |
| Table 4-3 | Performances of the SLFN model with the R-ELM and the ELM | 71 |
| Table 4-4 | Comparison with a few existing models | 74 |
This page is intentionally left blank
Chapter 1

Introduction

1.1 Background

It is well known that financial modelling, analysis and forecasting based on optimization and statistics can improve the success of business operations and also help governments to make healthy economic development plan and avoid financial crises [1]. Moreover, the business investors can use financial forecasting techniques to improve the decision makings, prevent potential losses from market risks and therefore make profits [2]. However, in practice, the forecasting of stock prices and foreign exchange rates is usually a difficult research area in the financial domain with many uncertain factors, including political factors, social factors, financial factors and economic factors, which will heavily affect the movements of financial indicators.

Because the statistical properties of financial data are complex, the performances of most statistical models cannot perform very well in many cases. For example, the commonly used linear time series models are difficult to capture characteristics of financial data with both complex dynamics and statistics. Therefore, how the intelligent algorithms with modern methodologies, such as neural networks and artificial intelligence, can be developed to explore both dynamic and statistical features of financial data is still challenge job.

Financial data are usually non-stationary, dynamic and chaotic [3]. However, due to rapid information change in financial market, more uncertainties are involved.
Hence, it is essential to develop appropriate methodologies for the description of complex change trends in financial market.

In the field of financial prediction and modelling, stock price forecasting can help investors to analyse potential opportunities to make profits [4]. Since movements of stock prices is affected by many factors such as political changes, general economic environments, and expectations of investors, the researchers have developed many robust algorithms, based on time series linear regression, random walk, decision tree and ANNs, to eliminate the effects of uncertainties on the modelling and achieve good prediction performances in many cases. However, more researches should be done for developing the algorithms to further analyse the effects of uncertainties on the growth of the stock data and the dynamic behaviours of the stock data against their complex statistics [5].

Since 1970s, the autoregressive integrated moving average (ARIMA) model has been widely used for financial system modelling and prediction. The popularity of the ARIMA model is due to its statistical properties as well as the well-known Box-Jenkins methodology, as discussed in [6], in the model development process. In addition, many other exponential smoothing models have been developed based on ARIMA models [7]. It has been noted that, although ARIMA models are flexible for representing a few different types of time series, such as pure autoregressive (AR), pure moving average (MA) and combined AR and MA (ARMA) series, the major limitation of these models is that they are formulated in linear forms, that is, a linear correlation structure is assumed and there are no complex nonlinear patterns that can be captured by the ARIMA model as well as the related linear models mentioned in the above. This is because the approximation of using linear models to complex nonlinear financial systems is not always satisfactory in practice.

Considering the complexity of financial systems and data, recently, the researchers in financial area have been using ANNs, as nonlinear financial time series models, for financial system modelling and forecasting. A detailed review about ANNs with applications to financial systems is presented in [8]. It has been widely recognized that the neural networks are capable of performing nonlinear
modelling with proper number of layers and nodes. Because of the nonlinear structures of ANNs, there is no particular model structure is needed. Instead, the neural model can be adaptively formulated based on our understanding of the features of the financial data to be processed. Such a data-driven approach is suitable for many practical situations, where the empirical data sets have complex dynamics and statistical characteristics. It has been noted that the most commonly used the ANN models are the multi-layer perceptron (MLP) [9]-[12], radial basis functions (RBFs) [13], [14] and recurrent neural networks (RNNs) [15]-[17]. The recent studies have shown that the ANN models are flexible and are capable of representing any complex financial systems with uncertainties [1].

The advantage of the ANN models as predictors is their learning capability. According to [18], an ANN can approximate any complex nonlinear relationship if the network has enough number of hidden nodes. Comparing with many existing financial time series prediction models, we have seen that the well-trained the ANN models can improve the quality of decision-making and avoid loss and risk [19]. In recent years, the ANNs have also been extensively used in industrial systems and scientific researches to perform diverse and sophisticated modelling and prediction cases, such as signal processing [20], pattern classification [21] and industrial system fault diagnosis [22].

In modern financial systems, the ANNs are often trained to capture the underlying dynamics of the currency exchange rates [15], [23], [24]. In early 1990s, some banks used the neural network models to process financial predictions as seen in [25]. Nowadays, a few large investment banks in the world, including Goldman Sachs and Morgan Stanley, have had their groups using neural networks for processing their business investments [26]. In recent years, many ANNs’ applications in financial forecasting have been reported as seen in [26] in detail. No doubt, the ANN-based models outperform mathematical and statistical models in most financial applications. The detailed discussions about the ANN models for exchange rates forecasting can be seen in [27]-[29], the ANN modelling for stock prices forecasting are reported in [30]-[32] and sales forecasting are in [33], [34].
1.2 Motivations and contributions

Prediction of financial time series, such as stock price and exchange rate, is a complicated task, because the underlying relationship of financial factors and the market trend is highly nonlinear, dynamic, complex and chaos. Furthermore, the movements of financial market is affected by many factors, such as financial market factors, social factors, political factors and environmental factors [35]. These factors include: 1) financial market factors, such as interest rates, exchange rates, monetary growth rates, commodity prices, and general economic conditions; 2) social factors, such as investors’ expectations and investment group of investors; 3) environmental factors, such as changes in company policies, income statements, and dividend yields; and 4) political variables, such as the occurrence and the release of important political events. Similar to image data, the financial data normally contains large information redundancy, the accuracy of forecasting for a financial index is always affected by the model has the capability to remove redundant information and noisy or not. Furthermore, as the financial market can be viewed as a dynamic system, the trend of financial market is also affected by the historical financial prices and factors. Therefore, the forecasting function of a specific market, such as stock or exchange rate, is contained by the external factors and internal prices. In classical time series methods, the forecasting function is a linear function comprises the historical observations and the white noise [6]. The simply linear structure and rough representation of influence factors leads to the forecasting performance is far away from the requirement.

The ANNs are successfully and widely employed in the area of image processing [36] and computer vision [37] with excellent performance. Because of the nonlinearity and strong redundant of image data, appropriate constructing the network structure and efficient optimizing the parameters for the neural network are needed to perform the accurate classification and recognition of image. Based on the researches of ANNs in different engineer and scientific areas, ANNs have the capability of performing universal approximation and capturing the underlying nonlinear dynamic relationship from the representative data. Furthermore, according
to the data-driven characteristic of the ANN, there is no extra needs of assumption between the input and output of the constructed model. In real-world applications, the structure of neural network and the tuning of free-parameters is developed by experience. Although, many researches of ANNs in financial forecasting give many attractive proposal on the modelling process and achieve good forecasting performance, there are still many valuable problems.

The main contributions of this thesis are two-folds: Firstly, a BPM-based SLFN neural predictive model is developed for forecasting daily exchange rate price. Tapped-delay-lines applied at the input layer enables strong dynamical modelling capability of the proposed neural predictive model for capturing complex dynamical relationships between the input-output financial data pairs. What is more, an improved momentum-based BP neural training algorithm is developed to train the SLFN model for predicting exchange rate, in that the BP training algorithm with the momentum term show effectiveness in reducing the effects of noises of the training data on the training process and ensure that the weights of SLFN quickly converge to the optimal solution in the weight space.

Secondly, an SLFN model trained with the R-ELM is developed for forecasting the daily closing price of a specific stock index. In particular, the input weights and the hidden biases are randomly selected in a small range so that most of the hidden neurons will work mainly at linear region of the activation functions. Regularization technique is applied in this model to enhance the neural model’s robustness against both internal and external disturbances. In the modelling processing, the additionally financial, economic and technical indicators are used as the model’s inputs to ensure that the underlying characteristics, such as short- and long- term trends and market momentum, is more clearly represented to the SLFN model.

### 1.3 Overview of the thesis

Chapter 2 presents a literature review of traditional financial forecasting methods, two typical stochastic time series models, namely, ARIMA and Generalised
Autoregressive Conditional Heteroscedasticity (GARCH) are discussed in detail. Furthermore, the fundamental elements and structure of ANNs for financial modelling and forecasting are discussed and the corresponding training algorithms including BP and ELM are studied. The applications of ANNs for forecasting exchange rate and stock price are investigated.

**Chapter 3** explores an SLFN model with tapped-delay-lines at the input layer trained with the BPM algorithm for modelling and forecasting exchange rate. Technical indicators are used as the input factors so that we can represent the features of exchange rate market more efficiently. Furthermore, the SLFN with tapped-delay-lines, added to the input layer, not only learn the complex market’s characteristics, but also capture the dynamic relationship between historical price and technical indicators. It has been noted that financial data are often highly nonlinearly related. Thus, the cost function is highly complex with many local minima. The BPM algorithm is used to update the weights of the SLFN model, the searching can avoid local minima and reach the global minimum in the parameter space in most cases. A number of experiments are carried out to confirm the effectiveness of the modified SLFN model as well as the BPM training algorithms.

**Chapter 4** investigates an SLFN model trained with R-ELM for modelling and forecasting the daily closing price of the S&P 500 index. In this model, the randomly assigned input weights to ensure the hidden feature vectors can response swiftly to the changes in the input patterns and the regularisation-based batch learning type of last squared method ensures fast learning speed and improved robustness of neural model. The experiment results show the efficiency and effectiveness of the R-ELM-based SLFN model for stock price forecasting. Furthermore, some comparisons of different number of hidden nodes and regularization term also show the proposed model with different hyperparameters has different efficiency and effectiveness. Therefore, the hyperparameters of the proposed model should be discreetly chosen.

**Chapter 5** summarises the contributions of this thesis and presents some interesting topics in financial forecasting and modelling for future work.
Chapter 2
Literature Review

2.1 Introduction

Since 1920s, financial modelling and forecasting have been attracting a great deal of attention from researchers in mathematics, economics, finance and other related fields [7]. The main purpose of financial modelling and forecasting research is to develop methodologies for finding out the patterns and dynamics of both the current and historical data and then forecasting the future variation of exchange rates and stock prices and so on.

The early methods for financial modelling and forecasting are fundamental analysis and technical analysis [30]. Fundamental analysis is mainly based on the dynamical information of demand and supply in economic and financial systems, to find out the main factors that dominate the business operations and then use the developed models to do the predictions for price evolutions and stock valuations et al [35]. The change of supply and demand lead to the change of the price. Therefore, from the view point of supply and demand, the fundamental analysis is able to make full use of the relationships among supply, demand, price and other factors, build the system models, and then perform forecasting of the trends of economic and financial events.

Technical analysis in principle is to predict the direction of prices based on models and trading rules with the price and volume information. Unlike the fundamental analysis, technical analysis does not consider external economic information as well as any other external news and events [38]. It is assumed that all
of the factors affecting the market movements are immediately included in the price. In practice, technical analysis uses many indicators, including relative strength index, directional movement indicators, on-balance volumes, moving averages, momentum and rate of change and charting, to analyse market movements [39]. However, both fundamental and technical analysis require substantial specialist experience, and thus, the prediction performances using both methods are not satisfactory in many complex financial and economic situations.

Since 1970s, statistical modelling techniques have been an important part in many financial and economic areas. These statistical time series models include linear regression, ARIMA model, GARCH model and their variations. In statistical modelling, the time series data is often assumed to be generated from a linear process with noises and disturbances. In the process of building a statistical model, three iterative steps are often required, that is, the model structure selection, parameter estimation and model evaluation. This three-step model building process is typically repeated, until a satisfied model is obtained. Then, the model can be used to capture the statistical properties of financial data and forecast the future movements of the corresponding financial data stream. Since financial markets are complex, noisy, nonlinear and non-parametric dynamic, it is very hard for using statistical methods to achieve accurate forecasting. And thus, more advanced adaptive and flexible models are required to overcome the disadvantages of the existing statistical models for capturing more information of the process and further improving forecasting accuracy.

ANNs and the corresponding training methodologies, as a branch of machine learning, was first adopted for modelling and forecasting financial time series in the early 1990s. Unlike the conventional statistical modelling methods, an ANN is capable of modelling the complex non-linear relationship between the input and output of a system, and, through learning from samples, the ANN model parameters (weights) can be optimally determined, without the requirement of the system details described by the input and output data. A well-trained ANN model is then capable of representing the complex dynamics of a financial or economical system and performing good prediction and forecasting for financial or economical system or
event. Because the ANN models for financial systems is the main focus of this research, the basics of the ANN will be briefly discussed in the following sections.

The rest of this chapter is organized as follows: Section 2.2 introduces the conventional time series models, including ARIMA and GARCH models. Section 2.3 presents the basics of ANN, including a single neuron, an SLFN model and two learning algorithms. Section 2.4 discusses the applications of ANNs to financial time series forecasting for both exchange rate and stock price systems. Lastly, conclusion is drawn in Section 2.5.

2.2 Time series models for financial forecasting

In order to analyse the behaviours of financial systems and do the prediction and forecasting, researchers need to first investigate the dynamic and statistical characteristics of financial data and then develop the corresponding statistical models, which can well represent the consider financial system. However, the prediction performance of statistical model largely depends on model structure as well as model parameter selection. In this section, we will present a brief overview of two important statistical models, that is, ARIMA and GARCH models.

2.2.1 ARIMA models

The ARIMA model was first proposed by Box and Jenkins in 1976 [40] and its mathematical expression can be represented as:

\[ \phi_p(B)(1 - B)^d y_t = \theta_0 + \theta_q(B)a_t \]  \hspace{1cm} (2.1)

with

\[ \phi_p(B) = (1 - \phi_1 B - \cdots - \phi_p B^p) \]  \hspace{1cm} (2.2)
\[ \theta_q(B) = (1 - \theta_1 B - \cdots - \theta_q B^q) \]  \hspace{1cm} (2.3)
\[ a_t \sim i.i.d(0, \sigma^2) \]  \hspace{1cm} (2.4)
where $B$ is a backward linear operator, $\phi_p$ is the $p^{th}$ order polynomial of $B$, and $\theta_q$ are is the $q^{th}$ order polynomial of $B$, $(1 - B)^d$ is the $d^{th}$ order polynomial of $B$, $y_t$ is the model response at time $t$, $\theta_0$ is a constant and $\alpha_t$ is the white noise.

The above model is characterized by three parameters $p$, $d$ and $q$, where $p$ is the order of the polynomial $\phi_p(B)$, $d$ is the order of the polynomial $(1 - B)^d$, and $q$ is the order of the polynomial $\theta_q(B)$, respectively.

In ARIMA ($p$, $d$, $q$) model, the future value of a time series is assumed to be a linear function of both the past observations and random errors. It is seen that the ARIMA model in the above includes the auto-regressive (AR) part, the integrated (I) part and the moving average (MA) part. The AR part is weighted historical variable of target time series, the I part is the difference linear operator of the original time series, the MA part is function of current and historical white noises.

Based on the earlier work in [6], [41], [42], a practical approach was developed to build ARIMA models. In last 20 years, Box and Jenkins modelling methodology has made the time series forecasting much easier. The Box-Jenkins model includes three iterative steps: model structure selection, parameter estimation and model evaluation. The idea of model structure selection is that, if the data is generated from an ARIMA process, autocorrelation properties should be considered. Due to the empirical autocorrelation patterns compared with the theoretical ones, we may adopt one or a few potential models for analysing and processing time series data. In Box and Jenkins’ work [6], the autocorrelation and the partial autocorrelation functions of the sample data are used to determine the order of the ARIMA model.

In the model structure selection step, the terms in the model are selected in the sense that: (i) The data should be transformed to have stationary property in order for insuring that its statistical characteristics such as the mean and the autocorrelation structure are constant over time; (ii) When the observed time series data presents both trend and heteroscedasticity, the model should be able to stabilize the variance.

After the model is constructed, the estimation of the model parameters can be done by using last square method or other regularization optimization methods.
Generally, the parameters are estimated in the sense that the total measurement errors are minimized.

In the model evaluation step, what we need to do is to check if the model assumptions are satisfied. If the model does not match, we may need to consider a new tentative model, following above three-steps to design the model again until the satisfied model is obtained.

Although ARIMA model is designed for non-stationary process, its linear and stationary natures have limited its applications for modelling complex time series data in many cases. In addition, ARIMA model needs the past information of the system outputs, the residuals of model’s output against actual observation values cannot sufficiently represent the real difference between actual observations and model outputs. This is because, in the model, only a set of white noises are used as inputs. It will be seen in the late sections that the integration of ANN and ARIMA can be used as hybrid model to improve the performance of financial time series modelling and forecasting [43].

### 2.2.2 GARCH models

Statistical research studies suggest that financial data exhibit time-variant characteristics with conditional variances, which are known as the ‘volatility clustering’ phenomenon. To analyse this phenomenon, a type of time series models known as GARCH model was introduced by Bollerslev in 1986 [44]. The mathematical expression of GARCH \((p, q)\) model is given below:

\[
y_t = x_t^T b + \epsilon_t \tag{2.5}
\]

with

\[
\epsilon_t | \Psi_t \sim N(0, \sigma_t^2) \tag{2.6}
\]

\[
\sigma_t^2 = \omega + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{i=1}^{p} \beta_i \sigma_{t-i}^2 \tag{2.7}
\]

where \(y_t\) is the dependent variable or target output, \(x_t\) is the column input vector, \(b\) is the column vector of parameters, \(\epsilon_t\) is the residual or innovation term at time \(t\), \(\Psi_t\)
is the information set of all information through time point $t$, $N(0, \sigma_t^2)$ is the Gaussian distribution of residual with time variant variance $\sigma_t^2$, $\omega$ is a constant, $\alpha_i$ and $\beta_i$ are the parameters for each lagged innovation and variance, respectively.

It is noted that GARCH model in the above is obtained by adding GARCH term $\sum_{i=1}^{p} \beta_i \sigma_{t-i}^2$ to the ARCH model. From equation (2.7), the number of lags in the ARCH terms (i.e. $\sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}^2$) can be significantly reduced by introducing the GARCH terms (i.e. $\sum_{i=1}^{p} \beta_i \sigma_{t-i}^2$), because the GARCH term includes previous information about all ARCH terms though with diminishing weights. According to (2.7), the current variance is dependent on previous lagged innovations, and therefore, the GARCH model is more efficient and flexible in sense of both the model structure and computational complexity.

In the applications, the GARCH model is considered as a mechanism focusing on identifying the effect of previous innovations and volatilities of previous periods on current volatility. It reveals the statistical and dynamic knowledge of the risks as well as the variance over time. It is also seen that innovations in this model are simply treated as the residuals between the expectation of the model and the actual observations. However, the assumption that positive and negative shocks have the same effects on volatility due to the fact that the squares of the previous shocks are not practical. In practice, we have known that the price of a financial asset may have different responses to positive and negative shocks.

In summary, in the above modelling processes, a financial system is treated as a linear system that is related to both current and historical prices and noises. However, most financial systems behave with highly nonlinearities in practice. In addition, some economic and political information may be used as the system input components, and the financial price and/or financial movements can be considered as the system outputs. Therefore, the process of modelling a financial system is to train the model parameters in the sense that the nonlinear dynamics embedded in the financial data can be learnt well. In the following sections, we will briefly discuss the ANN based learning models for financial systems.
2.3 Basics of ANNs

In modern engineering and financial areas, ANNs have been proved to be powerful parallel computational tools, which can learn dynamics from historical data and forecast the unseen data. Recently, ANNs have been more attractive in financial time series modelling and forecasting. It is well known that ANNs have the strong capability of modelling the systems with nonlinear, dynamic and nonstationary characteristics [45]. Based on the studies in financial modelling and forecasting, a financial system can be treated as a highly non-linear, time-varying and dynamic system. Therefore, in the absence of a complete understanding of the behaviour of the system, it will be of interest to use an ANN-based approach to characterize complex financial behaviour [18]. A well-trained ANN model has following useful properties and capabilities for financial modelling and forecasting:

**Nonlinearity:** An ANN which is made up of a group of interconnected nonlinear neurons performs a nonlinear mapping from input to output. The nonlinear architecture enables ANN to learn complex nonlinear dynamics that are not easy to be learnt with conventional modelling methods.

**Adaptivity:** ANN has variate of model structures and free parameters to adapt different types of data. In a real-world financial application, financial forecasting has many different types of problems. It is possible to use one class of ANN model to solve different types of financial issues.

**Parallel processing:** An ANN is composed of highly parallel connections between different layers of nodes that function as localized processing units. In financial forecasting and modelling, a large number of financial data is able to use in an ANN model, which makes possible to analyse different kinds of data.

In this section, the architecture of a single neuron, an SLFN architecture and learning algorithm of the ANN are summarized.
2.3.1 A single neuron

Neurons are elementary components of ANNs [18]. In a simple layer of an ANN model, basically, the role of neurons in an ANN is to perform the parallel signal processing in an efficient way. Fig. 2.1 shows the architecture of the \( k \)th neuron which is a basic element of the ANN models present in this later chapters. Mathematically, a neuron can be expressed as follows:

\[
y_k = g(v_k) = g\left(\sum_{i=1}^{n} w_{ik} x_i + b_k\right)
\]

where \( x_1, x_2, \ldots, x_n \) are the input signals, \( w_{1k}, w_{2k}, \ldots, w_{nk} \) are the input weights, \( b_k \) denotes the bias, \( g(\cdot) \) represents the activation function which is also known as squashing function, \( y_k \) and \( v_k \) are the output and input of \( k \)th neuron, respectively.

![Figure 2.1 A single neuron [46]](image)

where \( w_{1k}, w_{2k}, \ldots, w_{nk} \) are the input weights, \( b_k \) is a bias.

In Fig. 2.1, the activation function \( g(\cdot) \) is a linear or nonlinear operator, transforming the sum of the weighted input signals to the output [47]. A simple linear activation function can be described as:

\[
y_k = av_k
\]

The most popular sigmoidal nonlinear activation function is of the form:
\[ y_k = \frac{1}{1 + e^{-av_k}} \] (2.10)

In (2.10), \( a \) is the slope parameter, through tuning the parameter \( a \), we obtain different slopes of sigmoid function. In particular, as the slope parameter approaches infinity, the sigmoid function becomes a threshold function. The sigmoid function in (2.10) is differentiable, whereas the threshold function is not.

### 2.3.2 An SLFN structure

An SLFN is shown in Fig. 2.2, where \( w_{ij} \), for \( i = 1, 2, \ldots, n \) and \( j = 1, 2, \ldots, N \) are the connected weights between the \( i \)th neuron of the input layer and the \( j \)th neuron of the hidden layer, \( \beta_{jq} \), for \( j = 1, 2, \ldots, N \) and \( q = 1, 2, \ldots, m \) are the output weights connecting between the \( j \)th neuron of the hidden layer and the \( q \)th neuron of the output layer. \( b_j \) is the bias of the \( j \)th hidden neuron, and \( g(\cdot) \) is the nonlinear activation function in the hidden layer, the activation function in the output layer is identity function. The corresponding output of the \( j \)th node in the hidden layer can be described as:

\[ h_j = g(v_j) \quad \text{for } j = 1, 2, \ldots, N \] (2.11)

with the induced input signal for \( j \)th hidden neuron as:

\[ v_j = \sum_{i=1}^{n} w_{ij}x_i + b_j \] (2.12)

Then, the \( q \)th output of the SLFN model can be represented as

\[ y_q = \sum_{j=1}^{N} \beta_{jq}h_j \quad \text{for } q = 1, 2, \ldots, m \] (2.13)
It is worth mentioning that, SLFNs have been successfully used to model nonlinear systems due to their strong generalization capability of dealing with uncertainties and the complex input-output relationships. Essentially, many existing effective training algorithms are the crucial components for SLFN learning. In practice, BP-based algorithm and the ELM are the two prominent learning methodologies for SLFNs’ training, which will be discussed in more details in the following sections.

2.3.3 Learning algorithms for ANNs

(1) BP algorithm

The most popular learning algorithm for the SLFN is the BP. Since the time the BP was developed by Rumelhart et al. [48] in the middle 1980s, it has been the cornerstone for neural computing. The learning process of ANNs with BP algorithm has major two phases: the forward phase and backward phase. In the forward step, the weights of ANNs are initialized for a certain distribution and the input is propagated through the ANN, layer-by-layer. In the backward step, the output signal
is used to compare with the target, then the error signal is back-propagated layer-by-layer to adjust the weights of the ANN by using the optimization method.

Mathematically, the structure of a simple SLFN-based neural model is shown in Fig. 2.3, where \( w_{ij} \), for \( i = 1, 2, ..., n \) and \( j = 1, 2, ..., N \) are the input weights connecting the \( i \)th neuron of the input layer to the \( j \)th neuron of the hidden layer, \( \beta_{jq} \), for \( j = 1, 2, ..., N \) and \( q = 1, 2, ..., m \) are the output weights connecting the \( j \)th neuron of the hidden layer to the \( q \)th neuron of the output layer, \( b_j \) is the bias of the \( j \)th hidden neuron. The corresponding output of the \( j \)th node in the hidden layer can be described as:

\[
h_j = g(v_j), \quad \text{for } j = 1, 2, ..., N
\]  

(2.14)

with the induced input signal for \( j \)th hidden neuron as:

\[
v_j = \sum_{i=1}^{n} w_{ij} x_i + b_j
\]  

(2.15)

\( g(\cdot) \) the nonlinear activation function in the hidden layer; the bias \( b_j \) an external hidden bias term of \( j \)th neuron.
Then, the $q$th output of the SLFN model can be represented as

$$y_q = \sum_{j=1}^{N} \beta_{jq} h_j, \quad \text{for } k = 1, 2, \ldots, m$$  \hspace{1cm} (2.16)

Given a training dataset $\{(x^p, d^p), x^p \in \mathbb{R}^n, d^p \in \mathbb{R}^m\}_{p=1}^M$, where the superscript $p$ denotes the $p$th training sample, the cost function is defined as:

$$E = \frac{1}{2} \sum_{p=1}^{M} \|d^p - y^p\|_2^2 = \frac{1}{2} \sum_{p=1}^{M} \sum_{q=1}^{m} (d^p_q - y^p_q)^2$$  \hspace{1cm} (2.17)

with

$$y^p_q = \sum_{j=1}^{N} \beta_{jq} g \left( \sum_{i=1}^{n} w_{ij} x^p_i + b_j \right)$$  \hspace{1cm} (2.18)

The objective of BP can be formulated in terms of an unconstrained optimisation problem:

Minimise $E = \frac{1}{2} \sum_{p=1}^{M} \sum_{q=1}^{m} (d^p_q - y^p_q)^2$  \hspace{1cm} (2.19)

Accordingly, the chain rule can then be used for computing the correction weights in the backward pass layer by layer.

Firstly, randomly choosing initiation values of weights and biases $\beta_{jq}(0), w_{ij}(0), b_j(0)$, for $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, N$, and $q = 1, 2, \ldots, m$. The output weights can then be updated with the following gradient descent rule as:

$$\beta_{jq}(k + 1) = \beta_{jq}(k) - \eta \frac{\partial E}{\partial \beta_{jq}}$$  \hspace{1cm} (2.20)

with

$$\frac{\partial E}{\partial \beta_{jq}} = \frac{\partial E}{\partial e_q} \frac{\partial e_q}{\partial y_q} \frac{\partial y_q}{\partial \beta_{jq}}$$  \hspace{1cm} (2.21)

$$\frac{\partial E}{\partial \beta_{jq}} = \sum_{p=1}^{M} \sum_{q=1}^{m} (d^p_q - y^p_q) h_j$$  \hspace{1cm} (2.22)

After that, the input weights and the biases can be computed as:
\begin{align}
    w_{ij}(k + 1) &= w_{ij}(k) - \eta \frac{\partial E}{\partial w_{ij}} \tag{2.23} \\
    b_j(k + 1) &= b_j(k) - \eta \frac{\partial E}{\partial b_j} \tag{2.24}
\end{align}

with

\begin{align}
    \frac{\partial E}{\partial w_{ij}} &= \frac{\partial E}{\partial e_j} \frac{\partial e_j}{\partial y_j} \frac{\partial y_j}{\partial h_j} \frac{\partial h_j}{\partial v_j} \frac{\partial v_j}{\partial w_{ij}} \tag{2.25} \\
    \frac{\partial E}{\partial b_j} &= \frac{\partial E}{\partial e_j} \frac{\partial e_j}{\partial y_j} \frac{\partial y_j}{\partial h_j} \frac{\partial h_j}{\partial v_j} \frac{\partial v_j}{\partial b_j} \tag{2.27}
\end{align}

\begin{equation}
    \frac{\partial E}{\partial w_{ij}} = \sum_{p=1}^{M} \sum_{q=1}^{m} \sum_{j=1}^{N} (d^p_q - y^p_q) \beta_{jk} g'_j(v_j) x_i \tag{2.26}
\end{equation}

\begin{equation}
    \frac{\partial E}{\partial b_j} = \sum_{p=1}^{M} \sum_{q=1}^{m} \sum_{j=1}^{N} (d^p_q - y^p_q) \beta_{jq} g'_j(v_j) \tag{2.28}
\end{equation}

where \( g'_j(v_j) \) is the derivative of the activation function with respect to \( v_j \).

In summary, based on the derivations in the above, the BP for training the SLFN model in Fig. 2.2 can be summarized as follows:

Step 1: Select a set of representative training data pairs to optimize the SLFN model.

Step 2: Initialize the weights and biases \((w_{ij}, \beta_{jq}, b_j)\), then calculate the cost function, set the learning rate, maximum iterations and target error.

Step 3: Calculate the output of the SLFN model.

Step 4: Calculate the corresponding error for weights connecting hidden layer and output layer.

Step 5: Update the weights connecting hidden layer and output layer by using \[ \beta_{jq}(k + 1) = \beta_{jq}(k) - \eta \frac{\partial E}{\partial \beta_{jq}}. \]

Step 6: Calculate the corresponding error for weights connecting hidden layer and input layer.
Step 7: Update the weights connecting hidden layer and output layer by using $$w_{ij}(k+1) = w_{ij}(k) - \eta \frac{\partial E}{\partial w_{ij}}$$ and the hidden layer biases by using $$b_j(k+1) = b_j(k) - \eta \frac{\partial E}{\partial b_j}$$.

Step 8: Compare the error at iteration $$k$$ with the target error, if the error is less than target error, jump to step 9, otherwise go back to step 4, or if reach the maximum iterations, also jump to step 9.

Step 9: Test the model on testing data and use the model for forecasting the target system.

It has been shown that the ANN models trained with the BP have the capability of approximating arbitrary non-linear mapping between inputs and outputs [49]-[53]. Although the BP is usually one of the most important weight updating algorithms for neural networks, it still has some major drawbacks [54]. The BP algorithm uses the gradient of the cost function to descent to a minimum, the convergence of the algorithm to the global minimum mainly depends on the location of the initial position in the parameter space. If the initialized weights are far from the global minimum, the optimization process is possibly trapped at a local minimum. If the local minimum is far from the global minimum, the training result may not be satisfied. Therefore, in many practical applications, the trial and error method for initial weights’ selection is required to ensure a good performance [18]. This trial and error process makes BP algorithm spend a longer training time.

For financial modelling and prediction, BP-based neural models are considered impractical as the timeliness and effectiveness of the neural models are important criteria in practical applications. Therefore, it is necessary to find alternative neural computing methods for development of effective financial predictive neural models.

(2) ELM

The ELM was proposed by Huang et. al. [55]. The ELM algorithm is a highly effective learning algorithm firstly designed for SLFNs. The uniqueness of the ELM lies in the random selection of input weights that do not require any tuning, and the
analytical solution of the output weights using the pseudo-inverse. The training procedures of the ELM differ significantly from the conventional iterative methods used to train neural networks such as BP method. Many researchers [56]-[59] have reported that the ELM is capable of learning thousands of times faster than the BP methods and tends to perform better in terms of classification accuracy and generalization on unseen samples.

The architecture of an SLFN as shown in Fig. 2.4 consists of a output layer with $m$ output nodes, a hidden layer with $N$ hidden nodes and a input layer with $n$ input nodes, where $w_{ij}$, for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, N$ are the input weights, $b_i$, for $i = 1, 2, \ldots, N$ are the input biases, $g(\cdot)$ is the activation function of hidden nodes, and $\beta_{ij}$, for $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, m$ are the output weights.

In Fig. 2.4, the input vector $\mathbf{x}$ and the output vector $\mathbf{y}$ are expressed as:

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$$  \hspace{1cm} (2.29)
The output of the $i$th hidden node in SLFN can then be computed as:

$$h_i = g(x^T w_i + b_i), \quad \text{for } i = 1, 2, \ldots, N \quad (2.31)$$

with the input weight vector $w_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^T$, for $i = 1, 2, \ldots, N$, $g(\cdot)$ is the activation function. The output vector $y$ of the SLFN is then expressed as:

$$y = h \beta$$

with

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_N^T \end{bmatrix} \quad (2.33)$$

$$\beta_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^T, \quad \text{for } i = 1, 2, \ldots, N \quad (2.34)$$

$$h = [g(x^T w_1 + b_1), g(x^T w_2 + b_2), \ldots, g(x^T w_N + b_N)] \quad (2.35)$$

In order to train the SLFN model using the ELM, $M$ distinct sample data pairs $\{(X, D), X \in \mathbb{R}^n \times \mathbb{R}^m, D \in \mathbb{R}^m \times \mathbb{R}^M\}_{p=1}^M$ are used, where $X$ is the given input matrix and $D$ is the corresponding desired matrix. In the ELM training process, the input weights $w_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^T$, for $i = 1, 2, \ldots, N$, and biases $b_i$, for $i = 1, 2, \ldots, N$, are randomly selected from a certain range. Then, the hidden layer output matrix $H$ can be expressed as:

$$H = \begin{bmatrix} g(x_1^T w_1 + b_1) & \cdots & g(x_1^T w_N + b_N) \\ \vdots & \ddots & \vdots \\ g(x_M^T w_1 + b_1) & \cdots & g(x_M^T w_N + b_N) \end{bmatrix} \quad (2.36)$$

$$x_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T, \quad \text{for } i = 1, 2, \ldots, M \quad (2.37)$$

Thus, the output matrix of the neural model can be expressed as:

$$Y = H \beta$$

with

\[
Y = \begin{bmatrix}
    y_1^T \\
    \vdots \\
    y_M^T
\end{bmatrix}
\]  
\tag{2.39}

\[
y_i = [y_{i1}, y_{i2}, \ldots, y_{im}]^T, \quad \text{for } i = 1, 2, \ldots, M
\]  
\tag{2.40}

\[
\beta = \begin{bmatrix}
    \beta_1^T \\
    \vdots \\
    \beta_N^T
\end{bmatrix}
\]  
\tag{2.41}

\[
\beta_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{mi}]^T, \quad \text{for } i = 1, 2, \ldots, N
\]  
\tag{2.42}

Let

\[
Y = D
\]  
\tag{2.43}

with

\[
D = \begin{bmatrix}
    d_1^T \\
    \vdots \\
    d_M^T
\end{bmatrix}
\]  
\tag{2.44}

\[
d_i = [d_{i1}, d_{i2}, \ldots, d_{im}]^T, \quad \text{for } i = 1, 2, \ldots, M
\]  
\tag{2.45}

Hence, the optimal output weight matrix \( \beta \) can be obtained through the batch-learning linear least squares method with a Moore-Penrose pseudo-inverse \([55]\) as:

\[
\hat{\beta} = H^\dagger D
\]  
\tag{2.46}

where \( H^\dagger \) denotes the Moore-Penrose generalized inverse of hidden output matrix \( H \).

In summary, the three steps of the SLFN trained with the ELM algorithm are as follows:

Step 1: Given a training data set, randomly select input weights and hidden biases.

Step 2: Calculate the hidden layer output matrix \( H \).

Step 3: Estimate the output weights \( \hat{\beta} \) by using the Moore-Penrose pseudo-inverse.
According to statistical optimization theory [60], the pseudo-inverse is a form of empirical risk minimization (ERM) strategy, where the focus is on minimizing the training error for approximating to the training sample set. However, achieving the minimum training error does not directly translate to good performance in the testing phase, where generalization ability is essential. In fact, over-fitting is often observed in the applications of the SLFN model trained with the ELM algorithm [55]. A good forecasting model should optimally balance the structural risk and empirical risk minimizations. In order to improve the generalization ability of the ELM, several modified ELMs [55], [57], [61] combining regularization techniques in the determination of the output weights have been proposed.

### 2.4 ANNs for financial forecasting

The trained ANNs have been useful and attractive nonlinear models in the domain of financial modelling and forecasting, because of their universal approximation capability. In the past three decades, a large number of ANNs have been proposed to model and simulate many real-world financial events [8], involving financial data processing [4], trade signal recognition [23], text mining of financial information [62], investment risk assessment [25], cash flow modelling [63] and financial time series forecasting [64] and so on. According to the existing researches on exchange rate forecasting and stock price forecasting [26], [65]-[71], ANN models have achieved better performances than many other classical models, such as time series models [7], linear regression models [28] and statistical models [72]. Also, many studies have found out that developing an appropriate ANN structure and using an effective learning algorithm can further improve the ANN models’ modelling and forecasting performance [19]. In the following, a few different ANN structures and learning algorithms used in both exchange rate and stock prices modelling and forecasting are briefly discussed.
2.4.1 ANNs for exchange rate modelling and forecasting

Due to the high potential of ANNs to model complex nonlinear systems, a few ANN models for exchange rate forecasting have been developed. In [72], a feedforward neural model is developed and the performance is compared to a few existing conventional models, including GARCH, ARIMA and their variations. Because the parameters of the neural model are optimally determined by using the cross-validation method, the overfitting problem in a noisy environment has been avoided. In particular, under the condition that only a small training set is available, the performance of the neural model is still better than the one of the classical models. Furthermore, the authors use a few statistical indicators and their autocorrelation information as the input variables of the neural model, the forecasting performance of the neural model can be further improved. The experiments in [72], using the data of the US dollar against five major currencies, have shown the good modelling and prediction performances over the ones of the classical models.

In [73], the authors compared feedforward neural model and RBF network model with the traditional time series models, including ARIMA, ARIMA-GARCH models, to forecast Brazilian exchange rate (Brazilian real/US dollar) in a period. The experiments show that the ANN models have better performance in 15 minutes, 60 minutes, 120 minutes, daily and weekly prices of Brazilian exchange rate. And the neural models still have space to further improve the prediction performance through choosing appropriate input horizon and optimization methods.

The research in [74] has indicated that a well-trained MLP model is an effective for predicting exchange rate. In this work, the MLP model is optimally trained with a representative training set and then used to forecast the exchange rate of Turkish lira against the US dollar with technical indicators as additional input data. The experimental results have shown that the MLP model is able to extract more information about exchange rate data compared with a few conventional models. In the experiments and comparisons. Also, the MLP model has showed a stronger robustness against the disturbances in exchange rate data, compared to the classical models.
In [75], an MLP model was proposed to forecast the foreign exchange rate. In this study, the target output signals of the MLP model is the differences between two consecutive day’s exchange rates. In such a way, a financial regression problem is viewed as a classification problem and thus, many classification methods and theories are able to be used for MLP models’ training. In the experiments, technical indicators including the standard deviation, simple moving averages, exponential moving averages, and relative strength index from different horizons, are used as the input signals, after training, the performances of ANN models can be greatly improved.

In order to enhance the generalisation and modelling capability of ANN models, the functional link neural network and the cascaded functional link neural network are developed to predict currency exchange rate among US dollar and Indian rupees, Japanese yen, and British pound [76]. It is seen that these models exhibit higher prediction accuracy and lower computational cost in comparison to the existing models. In order to further improve the adaptiveness and efficiency of the neural models, a self-organised MLP inspired by the evolutionary algorithm is proposed in [78], to predict USD/GBP exchange rate, JPY/USD exchange rate, and USD/EUR exchange rate and so on.

2.4.2 ANNs for stock price forecasting

Considering the advantages of ANN’s universal approximation capability, many researchers have been employing ANNs to forecast stock prices. Martinez et al. [77] proposed an ANN model to forecast highest and lowest price with a recent trading days’ data. This model is able to assist investors to make a better decision and gain a generous profit. Through comparison between ANN model and four benchmark models, It is seen that the ANN model can help the investors to gain very good profit. However, the training processing of the ANN model in [77] is longer than the ones of the benchmark models.

In [24], an MLP model was developed to forecast one-day ahead closing price of a stock from Indian stock market. In this work, the authors aimed to design the optimal architecture of the MLP and also use the optimization technique to design
weights in each layer of MLP. Such an optimal MLP model has greatly improved the performance of forecasting the stock prices in Indian stock market.

In [78], an MLP model was used to predict a stock price of a Brazilian company, where the resilient BP algorithm was applied to optimize ANN model parameters. It has been shown that, by using the resilient BP algorithm, each weight has an individual evolving update value and the weight-step is only determined by its update-value and the sign of the gradient. The simulation results show that the training of the ANN model with resilient BP is faster and behaves with a strong robustness against the external disturbances.

Vanstone et al. [79] designed an MLP model for stock price forecasting in Australian markets. The characteristic of this model is that some indicators from the companies’ fundamental data are used as the model inputs. The experimental results show that, after the training, the MLP model is employed accurately evaluate the profitability of a trading system.

In order to explore the effects of ANN architectures on the prediction accuracy, [80] studied the MLP model trained with BP, the RBF model with genetic algorithm and general regression neural model with gradient decent method, respectively. The convergence properties, the robustness against the data fluctuation and disturbances of these neural models are summarized, which are very helpful for handling the predictions of the stock market with different disturbances and uncertainties.

Liu and Wang [81] proposed an enhanced Legendre ANN model to predict price fluctuation in stock markets. In this paper, authors assumed that investors make their investitive decisions based on the historical data and some important technical indicators that can influence the movement of the current price. The main feature of this model is that a parameter generated by a random walk is used to weight each historical data. The experiments with the SBI, SAI, DJI and IXIC stock prices information showed that the enhanced Legendre ANN model can well predict price fluctuations of stock markets.
Ticknor [82] proposed a novel ANN model to forecast daily stock prices of Microsoft Corp and Goldman Sachs Groups Inc. In order to improve neural models’ the robustness and generalisation capability, the Bayesian regularized method was used for training this ANN model. It is seen that the algorithm can automatically adjust its weights in a Bayesian manner so that effect of the disturbances on the prediction accuracy can be reduced and hence the generalisation capability of the model can be improved.

2.5 Conclusion

In this chapter, two classical time series models, namely, the ARIMA and the GARCH, have been first discussed. The basics about ANN models and their training algorithms for financial modelling and prediction have then been studied. It is recognized that, due to the complex and nonlinearity of financial markets, influenced by many technical, social and political factors, ANN will play a very important role in next few decades for financial modeling and forecasting due to its powerful capability of universal approximation. It is believed that the ANN based artificial intelligence techniques will be the future research direction for developing advanced financial models for predicting complex exchange rate and stock price systems.
Chapter 3
An SLFN Model Trained with BPM for Exchange Rate Forecasting

3.1 Introduction

The forecasting of exchange rate is one of the most challenging tasks in financial and economic areas. This is because the exchange rate and many factors, such as financial market factors, social factors, political factors and environmental factors, have completed nonlinear dynamic relationships [83], [84]. And thus, from the viewpoint of system modelling and prediction, the prediction of exchange rate is always difficult. In the traditional studies of financial modelling, the linearly structured econometric models are commonly used. In many simple cases, where the financial data sequences present stationary characteristics in some time slots, the forecasting results based on the linear models are acceptable. However, as the data are highly fluctuated with complex statistics and dynamics, the predictions from the linear models are always far from the reality. This is because of their simple parametric model structures and the lack of the capability from the noisy and sophisticated capturing nonlinear features of financial data.

With the recent advancements in high-speed computing and data mining, the high-dimensional financial data have been widely used to study and gain more deep insight into the basic characteristics and operational mechanisms of financial market. Moreover, the recent developments in quantitative finance and machine learning have attracted the financial researchers to develop nonlinear models such as MLPs and RNNs to analyse the financial systems and events. Many results have shown that
these neural models are able to achieve much better performance, compared to the conventional linear methods [28], [69], [84], [85].

It has been noted that, the modelling and forecasting of exchange rate are data driven in nature. As the one of most suitable methodologies for learning the complex dynamics and statistics, the ANNs are able to approximate the latent mechanism of the financial systems by training with the input-and-output datasets. The prominent property of a neural network-based model is its universal learning capability from the data with complex dynamics and statistics. Usually, a well-trained ANN model is capable of describing the complex non-linear relationships between multi-inputs and multi-outputs of a financial system.

The ANN has been mathematically proven to be the universal function approximators [18]. In practice, both ANNs and SLFNs can be trained with gradient descent method based BP, which recursively adjusts the weights of networks. By obtaining the optimal solution, the neural networks are able to capture the system dynamical features from the training data pairs. Many successful applications can be found in system modelling [86], decision-making [87] and pattern classification [88]. The BP algorithm is normally designed with gradient descent training approach, which searches the global minimum of the cost function in the parameter space. However, when the gradient-based algorithms are employed to recursively search the global minimum point, the local minima problem may occur. This is because the gradient descent-based searching cannot distinguish the difference between the global minimum and a local minimum. To solve this problem, the BPM algorithm was developed in 1980s [18]. It has been seen that, with the help of the momentum term in the modified gradient descent algorithm, the searching can go out of local minima and reach the global minimum in the parameter space in most cases.

Consider the fact that, in most cases, the money exchange process shows complex dynamics. For example, the current closing price is not only related to both current social, economic, political and other input factors, but also related to the historical or past closing price factors as well as the past input factors. Thus, the neural network with only feedforward structures may not be sufficient to
approximate the complex dynamics of exchange rate. In this chapter, a RNN is first constructed to describe the complex dynamics relationship between the current closing price and the past closing prices as well as the past input factors. Then an SLFN with the augmented inputs is applied to approximate the recurrent neural model. In such a way, both the training and analysis of the neural model can be simplified.

The rest of the chapter is organised as follows: In Section 3.2, the gradient descent update law with momentum is studied. In Section 3.3, an SLFN model with augmented inputs, used to approximate the relationship of the exchange rate with financial and technical factors, are developed. Section 3.4 presents the simulations of the SLFN model for forecasting of exchange rate, which confirm the good performance of the SLFN model. Section 3.5 concludes this chapter.

3.2 BPM

In past three decades, BP method and its variations have been the most popular training algorithms, which is used to iteratively train ANNs with data pairs for the prediction and analysis of system dynamic behaviours [18]. It has shown that BP algorithm is particularly suitable for the training of ANNs as well as SLFNs. Through training with the sample data pairs, SLFNs are capable of learning the dynamics of complex systems very well for different tasks, including, nonlinear system modelling [20], decision-making [87], and pattern classification [88].
To describe the details of the BP algorithm, an SLFN model is shown in Fig. 2.3. The hidden layer has $N$ hidden nodes, and $w_{ij}$, for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, N$ is the input weights connecting the $i$th neuron of the input layer to the $j$th neuron of the hidden layer, $\beta_{jq}$, for $j = 1, 2, \ldots, N$ and $q = 1, 2, \ldots, m$, is the output weights connecting the $j$th neuron of the hidden layer to the $k$th neuron of the output layer, $b_j$ is the bias of the $j$th hidden neuron. The corresponding output of the $j$th node in the hidden layer can be described as:

$$h_j = g(v_j), \quad \text{for } j = 1, 2, \ldots, N \quad (3.1)$$

with the induced input signal for $j$th hidden neuron as:

$$v_j = \sum_{i=1}^{n} w_{ij} x_i + b_j \quad (3.2)$$

where $g(\cdot)$ is the nonlinear activation function in the hidden layer; the bias $b_j$ is an external hidden bias term of $j$th neuron.

Then, the $q$th output of the SLFN model can be represented as
\[ y_q = \sum_{j=1}^{N} \beta_{jq} h_j, \quad \text{for } q = 1, 2, ..., m \]  

(3.3)

Given a training dataset \( \{(x^p, d^p), x^p \in \mathcal{R}^n, d^p \in \mathcal{R}^m \}^{M}_{p=1} \), the sum of squared errors, that is, the cost function in the weight space can be described as:

\[
E = \frac{1}{2} \sum_{p=1}^{M} \|d^p - y^p\|^2_2 = \frac{1}{2} \sum_{p=1}^{M} \sum_{q=1}^{m} (d^p_q - y^p_q)^2
\]

(3.4)

with

\[ y^p_q = \sum_{j=1}^{N} \beta_{jq} g \left( \sum_{i=1}^{n} w_{ij} x_i + b_j \right) \]

(3.5)

The objective of the BP algorithm is to minimize the following unconstrained optimisation problem:

\[
\text{Minimise } E = \frac{1}{2} \sum_{p=1}^{M} \sum_{q=1}^{m} (d^p_q - y^p_q)^2
\]

(3.6)

The chain rule can then be used for updating the weights in the backward from the top layer to the bottom layer.

Mathematically, all of the weights and biases in the Fig. 3.1 are randomly initialized. The output weights is then updated as follows:

\[ \beta_{jk}(k + 1) = \beta_{jk}(k) - \eta \frac{\partial E}{\partial \beta_{jk}} \]

(3.7)

where \( \beta_{jk}(k) \) is the updated output weight at time \( k \), \( \eta \) is learning rate, and \( \frac{\partial E}{\partial \beta_{jk}} \) is the partial derivative of the cost function with respect to the output weights \( \beta_{jk} \), calculated by

\[
\frac{\partial E}{\partial \beta_{jq}} = \frac{\partial E}{\partial e_q} \frac{\partial e_q}{\partial y_q} \frac{\partial y_q}{\partial \beta_{jq}}
\]

(3.8)

or

\[
\frac{\partial E}{\partial \beta_{jq}} = \sum_{p=1}^{M} \sum_{q=1}^{m} (d^p_q - y^p_q) h_j
\]

(3.9)

Similarly, the update rule for the input weights is of the form:
\[ w_{ij}(k + 1) = w_{ij}(k) - \eta \frac{\partial E}{\partial w_{ij}} \] (3.10)

with

\[ \frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial e_j} \frac{\partial e_j}{\partial y_j} \frac{\partial y_j}{\partial h_j} \frac{\partial h_j}{\partial v_j} \frac{\partial v_j}{\partial w_{ij}} \] (3.11)

or

\[ \frac{\partial E}{\partial w_{ij}} = \sum_{p=1}^{M} \sum_{q=1}^{m} \sum_{j=1}^{N} (d_q^p - y_q^p) \beta_{jq} g'_j(v_j) x_i \] (3.12)

where \( g'_j(v_j) \) is the derivative of the activation function with respect to \( v_j \).

The update rule for hidden nodes biases is expressed as:

\[ b_j(k + 1) = b_j(k) - \eta \frac{\partial E}{\partial b_j} \] (3.13)

with

\[ \frac{\partial E}{\partial b_j} = \frac{\partial E}{\partial e_j} \frac{\partial e_j}{\partial y_j} \frac{\partial y_j}{\partial h_j} \frac{\partial h_j}{\partial v_j} \frac{\partial v_j}{\partial b_j} \] (3.14)

or

\[ \frac{\partial E}{\partial b_j} = \sum_{p=1}^{M} \sum_{q=1}^{m} \sum_{j=1}^{N} (d_q^p - y_q^p) \beta_{jq} g'_j(v_j) \] (3.15)

where \( g'_j(v_j) \) is the derivative of the activation function with respect to \( v_j \).

**Remark 3.1** The learning rate \( \eta \) is an important parameter in BP algorithm. Generally, a small learning rate will lead to a slow convergence of the parameter searching in the weight space, a large learning rate, however, will lead the parameter vibration. In addition, the initialization of all weights must be conducted properly, because a poor random initialization might lead to poor allocation of initial weights far from the optimal point.

As mentioned in Section 3.1, the update law in (3.7), (3.10) and (3.13) could possibly stop at local minima and also be sensitive to the external disturbances. In
order to overcome the disadvantages of the update law, the BPM is employed in this work to train the SLFN model. The BPM update rule of output weights can be expressed as follows:

\[ \beta_{jq}(k + 1) = \beta_{jq}(k) - \Delta \beta_{jq}(k) \]  

(3.16)

with

\[ \Delta \beta_{jq}(k) = \eta \frac{\partial E}{\partial \beta_{jq}} + \alpha \Delta \beta_{jq}(k - 1) \]  

(3.17)

\[ \frac{\partial E}{\partial \beta_{jq}} = \sum_{p=1}^{M} \sum_{q=1}^{m} (d^p_q - y^p_q) h_j \]  

(3.18)

The BPM update rule of input weights can also be expressed as:

\[ w_{ij}(k + 1) = w_{ij}(k) - \Delta w_{ij}(k) \]  

(3.19)

with

\[ \Delta w_{ij}(k) = \eta \frac{\partial E}{\partial w_{ij}} + \alpha \Delta w_{ij}(k - 1) \]  

(3.20)

\[ \frac{\partial E}{\partial w_{ij}} = \sum_{p=1}^{M} \sum_{q=1}^{m} \sum_{j=1}^{N} (d^p_q - y^p_q) \beta_{jq} g_j' x_i \]  

(3.21)

In addition, the BPM update rule of hidden nodes bias can be expressed as:

\[ b_j(k + 1) = b_j(k) - \left( \eta \Delta b_j(k) + \alpha \Delta b_j(k - 1) \right) \]  

(3.22)

with

\[ \Delta b_j(k) = \eta \frac{\partial E}{\partial b_j} + \alpha \Delta b_j(k - 1) \]  

(3.23)

\[ \frac{\partial E}{\partial b_j} = \sum_{p=1}^{M} \sum_{q=1}^{m} \sum_{j=1}^{N} (d^p_q - y^p_q) \beta_{jk} g_j' \]  

(3.24)

where \( w_{ij}(k + 1) \) is the updated input weights at time \( k + 1 \), \( b_j(k + 1) \) is the updated input weights at time \( k + 1 \), \( \beta_{jq}(k + 1) \) is the updated output weights at time \( k + 1 \), \( \alpha \) is the momentum parameter. Momentum gradient descent is found to
be useful for going over sallow local minima and increase the learning speed at monotonic slope on the error surface. Also, another useful property of this method is its stable search direction by bringing the weights search system close to its critical damping and hence reducing oscillating search direction in a ravine region of the error surface which will cause network’s unstable performance.

**Remark 3.2** It is well-known that financial time series data are often highly nonlinearly related. Thus, the cost function or energy function of the error is also highly complex with many local minima. The introduction of the momentum term to (3.18), (3.20) and (3.23) is indeed useful for the optimal weight searching to stride all of over sallow local minima in the weight space. Another advantage of the inclusion of the momentum term is that the convergence behaviour of the weights of the neural model in the weight space can be stabilised due to damping nature of the momentum-based learning steps. In financial applications, this feature is particularly useful to reduce the effects of turbulent error surface caused by noises and outliers in the input financial data on the learning trajectory of the model’s weights.

### 3.3 An SLFN model for exchange rate forecasting

According to [89], an exchange rate as the system output is related to many input factors, including political, economic and financial factors. In general, the relationship between one-day ahead closing price of exchange rate and input factors can be expressed as follows:

\[
y(p + 1) = F(y(p), x(p))
\]  
(3.25)

where \(y(p)\) is the closing price vector of a specified exchange rate, \(x(p)\) is the related system input vector of the exchange rate, \(F(\cdot)\) is a nonlinear function describing the nonlinear relationship between the current closing price and both the historical closing price and input factors.

The nonlinear function \(F(\cdot)\) in (3.25) can be approximated by a well-trained SLFN model. Considering the fact that the current closing price is related to both the
historical closing prices and input factors, we may construct the neural model as shown in Fig. 3.2, where s tapped-delay-lines are added to the network input layer to provide the historical input data to the network, and together with a tapped-delay-line structured feedback from the SLFN model output to the network inputs. The historical output information is thus added to the network inputs. Such a network is constructed in the sense that the neural model can sufficiently express the input-and-output dynamics of a considered exchange rate system.

Remark 3.3 It is noted that the lengths of the tapped-delay-lines at the input layer of the network in Fig. 3.2 should be chosen according to the input factors’ effects on the closing price. For instance, the tomorrow’s stock index may be highly related to the indices of the history of the last seven days, while the crude oil price has a strong relationship to its last month’s values and so on.

According to approximation theory and the simulations, it is noted that, when the recurrent network in Fig. 3.2 is trained with gradient descent method, the error between the network output \( y(p) \) and the desired output \( d(p) \) is getting smaller and smaller. However, the convergence analysis of the RNN in Fig. 3.2 is quite difficult. For solving this problem, we use the feedforward neural network in Fig. 3.3 to approximate the recurrent network in Fig. 3.2.

It is seen from Fig. 3.3 that the feedback signal \( y(p) \) is replaced by the reference output signal \( d(p) \). This is because \( d(p) - y(p) = e(p) \) and, as the error signal \( e(p) \) is small enough, \( d(p) \cong y(p) \), that is, \( y(p) \) can be approximated by \( d(p) \).
Figure 3.2 A RNN for exchange rate forecasting
In Fig. 3.3, the input vector is denoted as:

\[
\mathbf{x}(p) = [x_1(p), ..., x_1(p - n_1 + 1), ..., x_s(p), ..., x_s(p - n_s + 1), d_1(p), ..., d_t(p - n_0 + 1)]^T
\]  

(3.26)

where \(x_i(p)\) is the \(i\)th factors at \(p\)th day, \(d(p)\) is the historical closing price at \(p\)th day, \(n = n_0 + n_1 + \cdots + n_s\) is the nodes of input \(x(p)\), \(n_0, n_1, ..., n_s\) are the length of unit-delays of each factor, \(s\) is the number of total factors. The output \(h_j(p)\) of \(j\)th hidden node is expressed as:

\[
h_j(p) = g(v_j(p)), \quad \text{for } j = 1, 2, ..., N
\]  

(3.27)
\[ v_j(p) = \sum_{i=1}^{n} w_{ij} x(p) + b_j \]  

(3.28)

where \( v_j(p) \) is the input and \( b_j \) is the bias for the \( j \)th hidden node, respectively. \( w_{ij} \) is the weight connect \( i \)th input node to \( j \)th hidden node. \( g(\cdot) \) is the activation function.

Thus, the network’s output \( y(p+1) \) can be expressed as:

\[ y(p+1) = \sum_{j=1}^{N} \beta_j h_j(p) \]  

(3.29)

where \( \beta_j \) is the weight connected the \( j \)th hidden node to the output node, \( N \) is the number of hidden nodes.

Therefore the nonlinear function in (3.25) can be approximated by the SLFN model in Fig. 3.3. In order to train the proposed model, the \( M \) distinct exchange rate data samples \( \{(x(p), d(p+1)), x(p) \in \mathbb{R}^n, d(p+1) \in \mathbb{R}^M\} \) are selected, the elements of \( x(p) \) are expressed in (3.26) and the \( d(p+1) \) is the target output.

Based on the BPM algorithm presented in Section 3.2, the cost function for the training data is defined as:

\[ E = \frac{1}{2} \sum_{p=0}^{M-1} e^2(p+1) = \frac{1}{2} \sum_{p=0}^{M-1} (d(p+1) - y(p+1))^2 \]  

(3.30)

where \( e(p+1) \) represents the error term between the network output for the \( p+1 \)th input samples and correspondent target, the \( d(p+1) \) is the target value, \( y(p+1) \) is defined in (3.29).

Therefore, with the aim of minimising the cost function \( E \) in (3.30), the following expression can be obtained by solving the unconstrained optimisation problem as:

\[ \text{minimise } E = \frac{1}{2} \sum_{p=1}^{M} e^2(p) \]  

(3.31)

according to the training process of the BPM algorithm, the update rule of the output weights can be calculated as (3.32)-(3.35):
\[
\beta_j(k + 1) = \beta_j(k) - \Delta \beta_j(k) \tag{3.32}
\]

with

\[
\Delta \beta_j(k) = \eta \frac{\partial E}{\partial \beta_{jk}} + \alpha \Delta \beta_j(k - 1) \tag{3.33}
\]

\[
\frac{\partial E}{\partial \beta_j} = \frac{\partial E}{\partial e_j} \frac{\partial e_j}{\partial y_j} \frac{\partial y_j}{\partial \beta_j} \tag{3.34}
\]

\[
\frac{\partial E}{\partial \beta_j} = \sum_{p=1}^{M} (d(p) - y(p)) h_j \tag{3.35}
\]

The update formulation for the input weights is shown as (3.36)-(3.39):

\[
w_{ij}(k + 1) = w_{ij}(k) - \Delta w_{ij}(k) \tag{3.36}
\]

with

\[
\Delta w_{ij}(k) = \eta \frac{\partial E}{\partial w_{ij}} + \alpha \Delta w_{ij}(k - 1) \tag{3.37}
\]

\[
\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial e_j} \frac{\partial e_j}{\partial y_j} \frac{\partial y_j}{\partial h_j} \frac{\partial h_j}{\partial v_j} \frac{\partial v_j}{\partial w_{ij}} \tag{3.38}
\]

\[
\frac{\partial E}{\partial w_{ij}} = \sum_{p=1}^{M} \sum_{j=1}^{N} (d(p) - y(p)) \beta_j g_j'(v_j(p)) x_i \tag{3.39}
\]

where \(g_j'(v_j(p))\) is the derivative of the activation function with respect to \(v_j(p)\).

The update formulation for input bias is expressed as (3.40)-(3.43):

\[
b_j(k + 1) = b_j(t) - \Delta b_j(k) \tag{3.40}
\]

with

\[
\Delta b_j(k) = \eta \frac{\partial E}{\partial b_j} + \alpha \Delta b_j(k - 1) \tag{3.41}
\]

\[
\frac{\partial E}{\partial b_j} = \frac{\partial E}{\partial e_j} \frac{\partial e_j}{\partial y_j} \frac{\partial y_j}{\partial h_j} \frac{\partial h_j}{\partial v_j} \frac{\partial v_j}{\partial b_j} \tag{3.42}
\]
\[ \frac{\partial E}{\partial b_j} = \sum_{p=1}^{M} \sum_{j=1}^{N} (d(p) - y(p)) \beta_j g'_j(v_j(p)) \]  \hspace{1cm} (3.43)

where \( g'_j(v_j(p)) \) is the derivative of the activation function with respect to \( v_j(p) \).

In summary, based on the derivations in the above, the training algorithm for the proposed SLFN model in Fig. 3.3 can be illustrated in Fig. 3.4.
Select a set of representative training data pairs

Initialize the weights and biases: \((w, \beta, b)\), set target error and maximum iterations

Calculate the output of SLFN model with (3.29)

Calculate the corresponding error for output weights

Update the output weights with (3.32)-(3.35)

Calculate the corresponding error for input weights and hidden biases

Update the input weights with (3.36)-(3.39) and the hidden biases with (3.40)-(3.43)

Check if the error is smaller than the target error?

Check if the maximum iteration arrive

End

Figure 3.4 Diagram of the modified SLFN trained with the BPM
3.4 Simulations and experimental results

In this section, the performance of the proposed BPM-based SLFN model is validated by forecasting the daily closing price of the Australian dollar against United States dollar (AUD/USD) exchange rate.

3.4.1 Data collection and pre-processing

The financial data set used in this work are collected from the Yahoo! Finance website [62]. The AUD/USD exchange rate prices data are collected from 08/02/2011 to 12/31/2015, which contains 1278 trading days. The original data include the daily closing price (target), the daily opening price, the daily highest price and the daily lowest price. In the forecasting process, in order to predict the next day’s closing price, the historical observations of these exchange rate prices are used as the input signals. In many researches [90]-[93], they only used the historical observations of the prices as the input signals, which is unable to comprehensively represent the volatility and dynamics of the exchange rate markets. Consider that technical indicators in quantitative investment is widely used for the investors to analyse and simulate the exchange rate markets [39]. Five popular technical indicators are added into the inputs for improving the forecasting performance of our model, the formulations of these technical indicators are given in the following:

a) Simple moving average (SMA):

\[ SMA_n = \frac{1}{n} \sum_{i=0}^{n-1} C_{t-i} \] \hspace{1cm} (3.44)

where \( C_t \) is a closing price on the \( t \)th day, and \( n \) is the input window length.

b) Exponential moving average (EMA):

\[ EMA_n = \frac{1}{n} \sum_{i=0}^{n-1} \omega_i C_{t-i} \] \hspace{1cm} (3.45)

where \( \sum_{i=0}^{n-1} \omega_i = 1 \), and \( n \) is the input window length, \( \omega_i \) is the weight for past price \( C_{t-i} \).
c) Relative strength index (RSI):

\[
RSI_n = 100 - \frac{100}{1 + \frac{EMA_n(DM^+)}{EMA_n(DM^-)}}
\]  

(3.46)

where \( EMA_n(DM^+) = \max(C_t - C_{t-1}, 0) \) is the positive directional movement, and \( EMA_n(DM^-) = \min(C_t - C_{t-1}, 0) \) is the negative directional movement.

d) Triple exponential moving average (TEMA):

\[
TEMA_n = 3 \times EMA_n - 3 \times EMA_n(EMA_n) + EMA_n(EMA_n(EMA_n))
\]  

(3.47)

where \( EMA_n \) is computed by (3.45), and \( n \) is the input window length.

e) Moving average convergence divergence (MACD):

\[
MACD = EMA_{12} - EMA_{26}
\]  

(3.48)

where \( EMA_{12} \) and \( EMA_{26} \) are calculated by using (3.45), the lengths of the input window are 12 and 26, respectively.

In addition, the following two statistical metrics are utilized to evaluate the accuracy of the prediction errors. The formulations of these statistical criteria are expressed as follows:

1) Rooted Mean Squared Error (RMSE)

\[
RMSE = \sqrt{\frac{1}{M} \sum_{i=1}^{M} (p_i - t_i)^2}
\]  

(3.49)

2) Mean Absolute Error (MAE)

\[
MAE = \frac{1}{M} \sum_{i=1}^{M} |p_i - t_i|
\]  

(3.50)

In (3.49) and (3.50), \( p_i \) represents the predicted value and \( t_i \) represents the target value.
To avoid the influence of different data scales, we normalize feature values linearly in a range of [0 1]. Otherwise the factors with large numeric range in value will dominate those with small numeric range. The used normalization function is as follows:

\[ T_t = \frac{x_t - \min(x_t)}{\max(x_t) - \min(x_t)} \]  \hspace{1cm} (3.51)

where \( T_t \) is taken as the normalized data at time \( t \) for original data \( x_t \), \( \min(x_t) \) and \( \max(x_t) \) represent the minimum value and maximum value over the whole series, respectively. All 9 factors are normalized before experiments in terms of (3.51).

### 3.4.2 Experiment results

1. **Experiment settings**

In order to evaluate the forecasting performance of the BPM-based SLFN model for predicting the one-day ahead closing price of the AUD/USD exchange rate, several experiments and comparisons are carried out in this section. The proportional splits for the data set is as follows: the 80% of the data set is used for training and the remaining 20% is used for testing. Total 1022 data samples from the time period between 08/02/2011 to 07/01/2015 are used as training data and the remaining 256 data samples from the time horizon between 08/01/2015 to 12/31/2015 are testing data. The modified SLFN model shown in Fig 3.3 is used as the forecasting model in this chapter which has one node at the output layer, and 9 tapped-delay-line memories at the input layer. The optimal length of each tapped-delay-line memory at the input layer is determined by trial and error method in the following sub-section. The activation functions of the hidden nodes and the output nodes are selected as sigmoid function and linear function, respectively. Total 9 technical and price factors which is presented in Section 3.4.1 are regarded as the input signals and the one-day ahead closing price of the AUD/USD exchange rate is the target. The rooted mean squared error (RMSE) and the mean absolute error (MAE) are used as the measurements to verify the forecasting performance of the models in this section. All experiments are implemented in Python 3.6 [94] in a Lenovo laptop computer with
Intel i7 processor, 2.20 GHz CPU and 8 GB of random-access memory. A popular Python machine learning package called Keras [95] is used to build SLFN model.

(2) Delay-unit length analysis

According to the discussion in 3.3, the function of tapped-delay lines memory is to capture the underlying dynamic relationship between historical observations and the targets. Therefore, choosing the length of delay-unit is the first step in exchange rate forecasting. In this subsection, an BPM-based SLFN model with 300 nonlinear hidden nodes and a single linear output node is used to forecast the exchange rate of AUD/USD, whereas all nine technical and price indicators are used as the input signals. The activation function of the hidden nodes and the output nodes are sigmoid function \( g(x) = \frac{1}{1+e^{-x}} \) and linear function, respectively. The learning rate is set as 0.01, the momentum term is 0.9 and the maximum epochs is set as 15000. According to studies [7], [30], [90] for the applications of financial time series forecasting and modelling, the length of each tapped-delay memory for exchange rate forecasting is same and between 5 to 30. The RMSE and the MAE of training and the RMSE and the MAE of testing are shown in Table 3-1.

<table>
<thead>
<tr>
<th>ANN model</th>
<th>Training performance</th>
<th>Testing performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RMSE</td>
<td>MAE</td>
</tr>
<tr>
<td>5*9-300-1</td>
<td>0.0071</td>
<td>0.0057</td>
</tr>
<tr>
<td>15*9-300-1</td>
<td>0.0097</td>
<td>0.0079</td>
</tr>
<tr>
<td>30*9-300-1</td>
<td>0.0109</td>
<td>0.0087</td>
</tr>
</tbody>
</table>

In Table 3-1, the expression 5*9-300-1, in the first column, means that we use 9 financial factors for previous five days as the inputs, the number of hidden nodes is 300 and the output number is 1. It is seen from Table 3-1 that the overall training performance in term of the RMSE is similar to that of the MAE. It is obviously that the training time increases when the units of tapped-delay-line memory is increasing. The forecasting performance is slightly improving as the number of days decreased. The reason is that long term information has limit effect on the previous price.
According to the efficient markets hypothesis [96], the long-term information of the market is absorbed by the price of the exchange rate. Therefore, based on the experimental and the theoretical results, the five days ahead leads to the SLFN model has the optimal forecasting performance and is used in the rest of the experiments.

(3) Comparison of the performances with different number of hidden nodes

The hidden nodes play a crucial role in the SLFN modelling capabilities, where the hidden nodes make the SLFN model has the generalization capability and able to perform complicated nonlinear mappings. The least number of hidden nodes makes the SLFN model has better generalization capability and avoids the overfitting problem. To the best of our knowledge, there is no a general rule for determining the optimal number of hidden nodes. The number of hidden nodes is set from a small number to a reasonable big number. For each chosen the number of hidden nodes, the performance is validated on testing data. After testing the performance on every number of hidden nodes, the number of hidden nodes produce the smallest testing RMSE on the testing data is marked as the optimal output. The other free-parameters are same as previous sub-section. Table 3-2 shows the training and testing performances of the SLFN models with the numbers of hidden nodes from 100 to 900.

It is seen from Table 3.2 that when the number of hidden nodes is not more than 600, the values of RMSE and MAE are reduces as the number of hidden nodes is increased from 100 to 600. However, when the number of hidden nodes is more than 600, the performances of the ANN models are unsatisfying. Due to the highly noisy environment of financial market, with too many hidden nodes, the SLFN model learns many unrelated information from historical data, therefore the testing performance is far away from satisfying. Table 3-2 indicates that when the number of hidden nodes is set more than 600, the overfitting problem exists. From the experiments, the SLFN with 600 hidden nodes has the best performance and is used in the rest of the experiments. The adjustment of the hidden nodes numbers has restricted effect on the performance of exchange rate forecasting.
Table 3-2 Performances of the SLFN model with different hidden node numbers

<table>
<thead>
<tr>
<th>No. of hidden nodes</th>
<th>Training performance</th>
<th>Testing performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RMSE</td>
<td>MAE</td>
</tr>
<tr>
<td>100</td>
<td>0.0107</td>
<td>0.0092</td>
</tr>
<tr>
<td>200</td>
<td>0.0085</td>
<td>0.0070</td>
</tr>
<tr>
<td>300</td>
<td>0.0071</td>
<td>0.0057</td>
</tr>
<tr>
<td>400</td>
<td>0.0067</td>
<td>0.0052</td>
</tr>
<tr>
<td>500</td>
<td>0.0063</td>
<td>0.0048</td>
</tr>
<tr>
<td>600</td>
<td>0.0066</td>
<td>0.0051</td>
</tr>
<tr>
<td>700</td>
<td>0.0061</td>
<td>0.0046</td>
</tr>
<tr>
<td>800</td>
<td>0.0062</td>
<td>0.0047</td>
</tr>
<tr>
<td>900</td>
<td>0.0061</td>
<td>0.0046</td>
</tr>
</tbody>
</table>

(4) Comparison of forecasting performance with different learning rates

Learning rate is an important term of BPM algorithm which affects the convergence of the cost function. The smaller we make the learning rate parameter, the smaller the changes to the weights in the network will be from one iteration to the next, and the smoother will be the trajectory in the weight space. If the learning rate is too large in order to speed up the rate of learning, the resulting large changes in the weights assume such a form that the network may become unstable. It determines that the cost function is able to converge to the global minimum or not. Therefore, the appropriate learning rate has pivotal contribution to the desired results. If the learning rate is too small, the SLFN model need more epochs to achieve the minimum, so when the leaning rate is 0.001, 0.0001 and 0.0001, the maximum epoch is set as 25000, 35000 and 45000, respectively. Therefore, the training time increases and in the results are shown in Table 3-3.

As we can see in Table 3-3, the optimal learning rate is 0.001 with the smallest RMSE and MAE on both training and testing data. Due to the increasing of the maximum epoch, the training time increases as well. However, the model with the optimal learning rate 0.001 only uses 151 seconds with 25000 epochs. The training time is acceptable in the real financial time series forecasting applications and it can be shorter with the improvement of the computer capability. The SLFN model with
0.00001 learning rate has very unsatisfying performance both on testing and training sets. The reason may be that the too small learning rate results in the cost function is not convergence to the minimum. We choose 0.001 as the optimal parameter and use it for the rest of our experiments.

Table 3-3 Performances of the SLFN model with different learning rates

<table>
<thead>
<tr>
<th>Learning Rate</th>
<th>Training performance</th>
<th>Testing performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RMSE</td>
<td>MAE</td>
</tr>
<tr>
<td>0.01</td>
<td>0.0066</td>
<td>0.0051</td>
</tr>
<tr>
<td>0.001</td>
<td>0.0058</td>
<td>0.0044</td>
</tr>
<tr>
<td>0.0001</td>
<td>0.0071</td>
<td>0.0052</td>
</tr>
<tr>
<td>0.00001</td>
<td>0.0213</td>
<td>0.0169</td>
</tr>
</tbody>
</table>

(5) The effectiveness of the momentum term

In order to verify the effectiveness of the momentum term, a comparison between the BP and BPM algorithms is carried out in this sub-section. Theoretically, the momentum term can help to jump out the local minima and reach the global minimum [18]. Furthermore, this technique contributes to improve the convergence speed. In this sub-section, the BPM algorithm is compared with the BP algorithm in order to verify the effect of the momentum term. The learning rate is set as 0.001 which is determined by the experiments in the previous. And the number of hidden nodes is selected as 700 depending on the experiments of hidden nodes number. The training RMSE, MAE and time and the testing RMSE, MAE of the SLFN model trained with or without the momentum are shown in Table 3-3.

It can be seen from Table 3-3, the SLFN model trained with BPM has smaller RMSE and MAE of training and testing than the ones of the model trained with BP. The SLFN with BP may be trapped in local minimum, so RMSE and MAE are far away from the satisfying value. RMSE and MAE of the model trained with BPM is smaller than the model with BP, because the momentum term helps to go out of the local minima and possible converge to the global minimum. Furthermore, the training time of the SLFN model with BPM is shorter than that of the SLFN model.
trained with BP. Accordingly, the momentum contributes to the performance of the model as well as the training speed.

Table 3-4 Performances of the SLFN model with or without momentum term

<table>
<thead>
<tr>
<th>Training algorithm</th>
<th>Training performance</th>
<th>Testing performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RMSE</td>
<td>MAE</td>
</tr>
<tr>
<td>BPM</td>
<td>0.0058</td>
<td>0.0044</td>
</tr>
<tr>
<td>BP</td>
<td>0.0099</td>
<td>0.0075</td>
</tr>
</tbody>
</table>

(6) Models comparisons

Some classical models, such as ARIMA, GARCH, linear regression (LR) and AR-GARCH (AR-GARCH) [30], are applied to compare the forecasting performance with the proposed model. LR is a linear model that only uses the financial factors as inputs, which poorly represents the long-term dynamics of exchange rate. The 9 financial factors described in section 3.4.1 are used as the inputs of the LR model. The parameters $p, d, q$ of the ARIMA model are set as 1, 1, 2 by a lot of statistical tests. The forecasting results of the five models for one-day ahead AUD/USD exchange rate price are shown in Table 3-4.

Table 3-5 Performance comparison of the proposed model with other models

<table>
<thead>
<tr>
<th>Models</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Testing RMSE</td>
</tr>
<tr>
<td>the SLFN model</td>
<td>0.0135</td>
</tr>
<tr>
<td>ARIMA</td>
<td>0.0339</td>
</tr>
<tr>
<td>GARCH</td>
<td>0.0326</td>
</tr>
<tr>
<td>LR</td>
<td>0.0296</td>
</tr>
<tr>
<td>AR-GARCH</td>
<td>0.0309</td>
</tr>
</tbody>
</table>

It can be seen from Table 3-4 the proposed model achieves the best RMSE and MAE which are much lower than the ones of other models. The reason is that the neural model can learn the complex nonlinear dynamics of data very well. However, because the hidden layer of the neural model is set as 700, the training time of the
proposed model is 151 seconds which is much longer than the ones of other models. In the real-world applications, this length of training time is acceptable.

3.5 Conclusion

In this chapter, an SLFN model trained with BP momentum algorithm has been developed for modelling and forecasting the AUD/USD foreign exchange rate price. It has been seen that the developed SLFN model is the approximation of a recurrent ANN model, which can sufficiently capture dynamics of the financial data pairs. Due to the highly complicated relationship between financial factors and exchange rate price, the cost function has many local minima in weight space. By adding the momentum term to the BP weight update rule, the searching in the weight space may jump out some shallow local minima and reaches the global minimum point. Especially, when the model is used for forecasting, the model performance is better than the ones of all of existing linear models. Simulation results have shown that the SLFN model with BPM algorithm in this chapter has a great potential for practical applications.
Chapter 4
An SLFN Model with R-ELM Algorithm for Stock Price Forecasting

4.1 Introduction

Modelling and forecasting of stock market are crucial for financial investors to formulate the investment strategy and reduce the investment risk. As reviewed in Section 2.2, traditional statistical time series models are able to effectively model simple dynamic stock markets and systems. However, due to the complexity and turbulent data environment of the stock market system and the simple dynamic structure of the statistical time series models [7], [43], the statistical models such as the ARIMA [40] and the GARCH [44] models are incapable of modelling the complex dynamics of the stock market. Therefore, in order to achieve better modelling results for the complex dynamic behaviour of stock market and enhance the prediction accuracy of stock price, it is crucial to develop more sophisticated algorithms to help investors to make better decisions.

Recently, neural network based machine learning and artificial intelligent techniques have been explored for analysing and forecasting financial markets [14], [90], such as foreign exchange rate markets [28], [97], commodity markets [98], gold market [99], [100], futures markets [101], spot markets [72], [102], interbank lending market [103], [104] and stock markets [3], [17], [19], [26]. Among these methods, ANNs are widely used for modelling the complex financial system through learning from the given input-output data pairs due to their universal approximation capability.
In the previous chapter, a modified SLFN trained with the BPM algorithm has been developed to predict exchange rate and the simulation results have shown the superiority of learning and forecasting ability of the modified SLFN model. As discussed in the previous chapter, due to the recursive nature of the BP-based weights optimization procedure, neural models are subjected to inherent issues such as, the local minimum problem and the slow convergence problem. In light of this, the ELM was developed [55], [57], [60] to train SLFNs. The unique of the ELM is that the input weights and hidden layer biases are randomly assigned and then the output weights can be determined by Moore-Penrose generalized inverse of hidden layer outputs. The learning speed of the ELM is often extremely fast and it exhibits good generalization performance.

However, it is noted that ELM-based SLFN models can be very sensitive to noises in the inputs as randomly generated input weight matrix is very large sometime and hence influence of input noises will have significant impact on the movements of the hidden feature vectors. As a result, structural and empirical risks of the model can be high as output weights also become very sensitive to input noises [105]. In [106], a modified ELM called R-ELM is proposed where the cost function consists of both the sum of error squares and the sum of squared output weights (i.e. the regularization term). According to regularization theory [18], by introducing the regularization term into the cost function, the optimized output weights are penalized in such way that a smoother hyper-plane is generated through reducing effect of sudden changes and outliers in the hidden feature space on the formation of the hyper-plane. Due to the complicated nonlinear and dynamic of stock market, we apply the R-ELM for our proposed neural model in this chapter.

In this chapter, an SLFN with tapped-delay-lines at input layer trained by using R-ELM is developed to perform the one-day-ahead closing price of the S&P 500 index. In particular, considering the stock price is not only related to the current financial factors but also related to the history of financial factors and stock price, the tapped-delay-lines are added to the input layer to represent the inputs with temporal pattern vectors, ensuring the dynamic features of stock market can be sufficiently captured. The R-ELM was used to improve the robustness of the neural model [106],
where the input weights and biases are randomly selected from a certain value range and the output weights are optimized using a modified regularised batch learning type of least square method.

The rest of this chapter is organized as follows: Section 4.2 presents the details of the R-ELM algorithm. In Section 4.3, the stock price forecasting using SLFNs with the R-ELM is described. In Section 4.4, the simulations and comparisons are carried out with the S&P 500 data to evaluate the effectiveness of the R-ELM algorithm and the forecasting performance of the R-ELM-based neural model. Section 4.5 gives the conclusion.

4.2 R-ELM

Mathematically, a simple SLFN trained with the R-ELM algorithm will be described in this section. The architecture of the SLFN as shown in Fig. 4.1 consists of a output layer with \( m \) output nodes, a hidden layer with \( N \) hidden nodes and a input layer with \( n \) input nodes, where \( \beta_{ij} \), for \( i = 1, 2, \ldots, N \) and \( j = 1, 2, \ldots, m \), are the output weights, \( w_{ij} \), for \( i = 1, 2, \ldots, n \) and \( j = 1, 2, \ldots, N \), are the input weights, \( b_i \), for \( i = 1, 2, \ldots, N \), are the input biases. \( g(\cdot) \) is the activation function of the hidden nodes.
In order to train the SLFN model with the R-ELM algorithm, $M$ distinct sample data pairs $(x_i, d_i)$, for $i = 1, 2, ..., M$, are used. The $i$th input vector is $x_i = [x_{i1}, ..., x_{in}]^T$ and the $i$th target vector $d_i = [d_{i1}, d_{i2}, ..., d_{im}]^T$. According to the ELM theory, the input weights and hidden biases are randomly selected from certain value intervals to ensure that the hidden neurons are all working within the linear region of the activation functions. Sigmoid function $g(\cdot)$ is used as the activation function for the hidden neurons of the model. Then, the hidden layer output matrix $H$ can be expressed as:

$$H = \begin{bmatrix}
g(x_1^T w_1 + b_1) & \cdots & g(x_1^T w_N + b_N) \\
\vdots & \ddots & \vdots \\
g(x_M^T w_1 + b_1) & \cdots & g(x_M^T w_N + b_N)
\end{bmatrix}$$  \hspace{1cm} (4.1)$$

with $b_i$, for $i = 1, 2, ..., N$, and $w_i = [w_{i1}, w_{i2}, ..., w_{im}]^T$, for $i = 1, 2, ..., N$.

Thus, the output matrix of the SLFN model can be calculated as:

$$Y = H\beta$$  \hspace{1cm} (4.2)$$

with
\[
Y = \begin{bmatrix} 
y_1^T \\
\vdots \\
y_M^T
\end{bmatrix}
\]  
(4.3)

\[
y_i = [y_{1i}, y_{2i}, \ldots, y_{Mi}]^T, \quad \text{for } i = 1, 2, \ldots, m
\]  
(4.4)

and

\[
\beta = \begin{bmatrix} 
\beta_1^T \\
\vdots \\
\beta_N^T
\end{bmatrix}
\]  
(4.5)

\[
\beta_i = [\beta_{1i}, \beta_{2i}, \ldots, \beta_{mi}]^T, \quad \text{for } i = 1, 2, \ldots, N
\]  
(4.6)

In order to improve the robustness of the neural model against both internal and external disturbances, the R-ELM algorithm (i.e. batch-learning type of least square method) is used for computation of the optimal output weight matrix of the neural model. In particular, the Lagrange multipliers are utilized as a constraint condition in the output weights optimization procedure. The optimization problem described above can be expressed as [107]:

Minimize \[
\frac{\gamma}{2} \| \epsilon \|^2 + \frac{1}{2} \| \beta \|^2
\]  
Subject to \[\epsilon = D - Y = D - H\beta\]  
(4.7)

(4.8)

where \(\gamma\) is a constant balancing parameter for adjusting the balance of the empirical risk and the structural risk, \(D\) is the target matrix, and \(Y\) is the output matrix of the SLFN model. This problem can be solved by using the method of Lagrange multipliers:

\[
L = \frac{\gamma}{2} \sum_{i=1}^{N} \sum_{j=1}^{m} \epsilon_{ij}^2 + \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{N} \beta_{ij}^2 - \sum_{k=1}^{N} \sum_{p=1}^{m} \lambda_{kp} (h_k^T \beta_p - d_{kp} - \epsilon_{kp})
\]  
(4.9)

where \(\epsilon_{ij}\) is the \(ij\)th element of the error matrix \(\epsilon\), \(\beta_{ij}\) is the \(ij\)th element of the output weight matrix \(\beta\), \(d_{ij}\) is the \(ij\)th element of the target output matrix \(D\), \(h_k\) is the \(k\)th column of the hidden layer output matrix \(H\), \(\beta_p\) is the \(p\)th column of the output weight matrix \(\beta\), \(\lambda_{kp}\) is the \(kp\)th Lagrange multiplier, \(\gamma\) is real positive regularization.
parameter and $||\beta||^2$ is the regularization term of the output layer. Differentiating $L$ in (4.9) with respect to $\beta_{ij}$, the formula is shown as:

$$\frac{\partial L}{\partial \beta_{ij}} = \beta_{ij} - \sum_{k=1}^{N} \lambda_k h_k^T \beta_j$$  \hspace{1cm} (4.10)

where

$$h_k^T \beta_j = [h_{k1} \cdots h_{kN}] \begin{bmatrix} \beta_{1j} \\ \vdots \\ \beta_{Nj} \end{bmatrix} = \sum_{k=1}^{N} h_{kp} \beta_{pj}$$  \hspace{1cm} (4.11)

Therefore, (4.10) can be described as:

$$\frac{\partial L}{\partial \beta_{ij}} = \beta_{ij} - \sum_{k=1}^{N} \lambda_k h_k = \beta_{ij} - (\lambda_{1j} h_{1j} + \lambda_{2j} h_{2j} + \cdots + \lambda_{Nj} h_{Nj})$$  \hspace{1cm} (4.12)

Let $\frac{\partial L}{\partial \beta_{ij}} = 0$, we have:

$$\beta_{ij} = \lambda_{1j} h_{1j} + \lambda_{2j} h_{2j} + \cdots + \lambda_{Nj} h_{Nj}$$  \hspace{1cm} (4.13)

Therefore, the output weight matrix $\beta$ can be expressed as:

$$\beta = \begin{bmatrix} \beta_{11} & \cdots & \beta_{1m} \\ \vdots & \ddots & \vdots \\ \beta_{N1} & \cdots & \beta_{Nm} \end{bmatrix} = \begin{bmatrix} h_{11} & \cdots & h_{1m} \\ \vdots & \ddots & \vdots \\ h_{M1} & \cdots & h_{MN} \end{bmatrix} \begin{bmatrix} \lambda_{11} & \cdots & \lambda_{1m} \\ \vdots & \ddots & \vdots \\ \lambda_{N1} & \cdots & \lambda_{Nm} \end{bmatrix}$$  \hspace{1cm} (4.14)

Furthermore, (4.14) also can be re-expressed as:

$$\beta = H\lambda$$  \hspace{1cm} (4.15)

Differentiating $L$ in (4.9) with respect to $\varepsilon_{ij}$, the formula is shown as:

$$\frac{\partial L}{\partial \varepsilon_{ij}} = \gamma \varepsilon_{ij} + \lambda_{ij}$$  \hspace{1cm} (4.16)

Let $\frac{\partial L}{\partial \varepsilon_{ij}} = 0$, we have:

$$\lambda_{ij} = -\gamma \varepsilon_{ij}$$  \hspace{1cm} (4.17)
Furthermore, the vector form of the (4.17) can be expressed as:

\[ \lambda = -\gamma \varepsilon \]  

(4.18)

Considering the constraint in (4.8), (4.18) can be expressed as:

\[ \lambda = -\gamma (H\beta - D) \]  

(4.19)

Using (4.19) in (4.8), the optimal output weight matrix \( \beta \) can be computed as:

\[ \beta = \left( \frac{1}{\gamma} I + H^T H \right)^{-1} H^T D \]  

(4.20)

where \( I \) is a \( N \times N \) unity matrix, and \( \gamma \) is the regularization parameter.

**Remark 4.1** The regularization term \( \frac{1}{2} \| \beta \|_2^2 \) in (4.7) is set to reduce both empirical risks and structural risks and increase the robustness of the SLFN model. \( \gamma \) in (4.20) is the regularization parameter which controls the sensitivity of the output weights to the hidden layer outputs. In particular, when a small value of \( \gamma \) is used, the values of the optimised output weights become reluctant to large changes in the hidden outputs which might be caused by both internal and external disturbances (i.e. input noises and randomly generated input weights). On the other hand, a large \( \gamma \) means the constraint is weak, and hence the output weight is sensitive to any outliers in the hidden feature space. When \( \gamma \to \infty \), the R-ELM is equivalent to the ELM.

### 4.3 An SLFN model for forecasting stock closing price

In this work, an SLFN model with tapped-delay-lines at the input layer is developed to predict the daily closing price of the S&P 500 index. This index is a representative index for stock market of the United States (US), as it is the capitalization-weighted index of the largest 500 listed companies in the US stock market. In the literature of economics [62], the S&P 500 index is not only a financial indicator for individual investment, but also an important economic indicator for the global economy. In many financial studies [101], it is found that the movements of the S&P 500 are
affected by the gold price and the crude oil price, because the crude oil companies always rank in the top of these listed companies. As the development of global economy, the volatility of S&P 500 is affected by other countries’ stock market’s movements such as China and Japan. Furthermore, the exchange rate also affects the movements of the S&P 500. In conclusion, total 39 financial factors grouped as 5 classes are treated as inputs in this study. The details of these factors are shown in Section 4.4. Therefore, a modified SLFN trained with R-ELM is explored to learn the underlying dynamic and nonlinear relationship between these financial data and the closing price of the S&P 500 index.

During to the financial experiments and theories, the current closing price is not only affected by many internal and external factors, but also influenced by history of price and factors. The structure of the modified SLFN model is shown in Fig. 4.2, one-day ahead daily closing price of the S&P 500 is regarded as the output of the neural network model, $N$ is the number of the hidden nodes, the input layer has $s + 1$ tapped-delay-lines with total $n$ input nodes, $w_{ij}$, for $i = 1, 2, ..., n$ and $j = 1, 2, ..., N$ are the input weights, $\beta_j$, for $j = 1, 2, ..., N$ are the output weights, $b_i$, for $i = 1, 2, ..., N$ are the bias of the hidden layer.
In order to train the model in Fig. 4.2, the historical data set before day $p + 1$ are used as training data pairs, including $M$ input vectors:

$$X = [x(p), x(p-1), ..., x(p-M+1)]$$ \hspace{1cm} (4.21)

with
\[ x(p) = [x_1(p), ..., x_1(p - n_1 + 1), ..., \\
\quad x_s(p), ..., x_s(p - n_s + 1), d_i(p), ..., d_i(p - n_0 + 1)]^T \] (4.22)

where \( x_i(p) \) is the \( i \)th factors at \( p \)th day, \( d(p) \) is the historical closing price at \( p \)th day, \( n = n_0 + n_1 + \cdots + n_s \) is the number of input \( x(p) \), \( n_0, n_1, ..., n_s \) are the length of unit-delays of each factor, \( s \) is the number of total factors. And the corresponding target output vector

\[ d = [d(p + 1), d(p), ..., d(p - M)]^T \] (4.23)

By feeding the \( M \) input vectors into the model, the corresponding output vector

\[ y = [y(p + 1), y(p), ..., y(p - M)]^T \] (4.24)

is calculated as:

\[ y = H\beta \] (4.25)

with

\[ H = \begin{bmatrix}
g(x(p)^T w_1 + b_1) & \cdots & g(x(p)^T w_N + b_N) \\
\vdots & \ddots & \vdots \\
g(x(p - M + 1)^T w_1 + b_1) & \cdots & g(x(p - M + 1)^T w_N + b_N)
\end{bmatrix} \] (4.26)

\[ \beta = [\beta_1, \beta_2, ..., \beta_N]^T \] (4.27)

\[ w_i = [w_{1i}, ..., w_{ni}]^T, \quad \text{for} \ i = 1, 2, ..., N \] (4.28)

where \( g(\cdot) \) is the activation function, \( b_i \), for \( i = 1, 2, ..., N \) is the bias.

**Remark 4.2** According to the R-ELM algorithm, the input weights and bias are randomly generated in a range. In this chapter, the sigmoid function is used as the hidden layer activation function due to its boundedness and rich dynamics. In order to improve the model’s hidden neurons activity level, the input weights and biases need to be selected from a small value range. As a result, the output of hidden nodes is highly responsive to subtle changes in the input data. Such design of the hidden neurons’ activeness is mainly due to the nature of stock markets, in that the
movements of stock prices are sensitive to the changes of other financial factors or events.

According to the R-ELM algorithm derived in Section 4.2 (4.7)-(4.20), the output weight matrix $\beta$ can be estimated as:

$$
\beta = (\frac{I}{\lambda} + H^T H)^{-1} H^T D
$$

(4.29)

In summary, the training process of the ANN model can describe as following:

Step 1: Collect a set of representative historical data sets, in which some financial factors and technical factors are the input signals, and the corresponding one-day ahead close prices of the S&P 500 are used as the desired outputs.

Step 2: Randomly select the input weights $W$ and hidden bias $b$.

Step 3: Calculate the output of hidden layer matrix $H$ by (4.26).

Step 4: Estimate the generalized output weight vector $\beta$ from (4.29).

Step 5: Predict the future daily closing price by using the estimated output weight vector $\beta$.

Remark 4.3 By using batch learning type of R-ELM method as described in (4.21)-(4.29), the following advantages can be achieved: (i) the local minima in recursive learning algorithms is avoided; (ii) the global optimization in the parameter space can be arrived. (iii) the learning speed is extremely fast. In financial modeling and prediction applications, timely and robust predictive models are sought-after due to the fast changing and noisy nature of the market environment. Under such circumstances, R-ELM-based models are considered a suitable approach for this type of problem.
4.4 Simulations and experimental results

In this section, a description of data set is firstly presented. Then, the selections of parameters for the proposed model are tested and compared. In order to investigate the performance of the generalization term, the SLFN model trained with R-ELM compare with the model trained with ELM. Lastly, the SLFN model trained with the R-ELM is compared with the conventional models.

4.4.1 Data description

The financial data set used in the experiments are collected from the Yahoo! Finance website [62] and Investing website [108]. Different financial factors and stock prices data are collected from the time period between 01/01/2004 to 06/31/2018, which includes 3415 trading days. In Chapter 3, 5 technical indicators are added to the input data set to improve the representation of the original data. In this chapter, because the stock market is more complicated and active than the exchange rate market, more extra financial and economic factors are used as the input signals for the SLFN model. As we can see from Table 4-1, 39 types of financial and economic data related to the S&P 500 index are shown. These data set have six classes: 1) the daily prices and the volume of the S&P 500 index, 2) the technical indicators of the S&P 500 index, which are mathematical techniques to analyse and forecast the movements of the stock index in quantitative investment, 3) the exchange rate of US dollar against other countries’ currencies, 4) relevant major stock index all over the world, 5) the price of seven companies in the US, 6) important financial and economic indicators, such as gold price and crude oil price. The data is normalized according to the normalization formulation in Chapter 3.
Table 4-1 The 39 financial factors related to the S&P 500 index

<table>
<thead>
<tr>
<th>Classes</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original data</td>
<td>OPN</td>
<td>Opening prices of the S&amp;P 500 index</td>
</tr>
<tr>
<td></td>
<td>HIGH</td>
<td>High prices of the S&amp;P 500 index</td>
</tr>
<tr>
<td></td>
<td>LOW</td>
<td>Low prices of the S&amp;P 500 index</td>
</tr>
<tr>
<td></td>
<td>CLOSE</td>
<td>Closing prices of the S&amp;P 500 index</td>
</tr>
<tr>
<td></td>
<td>VOLUME</td>
<td>Volume of the S&amp;P 500 index</td>
</tr>
<tr>
<td>Technical indicators</td>
<td>EMA</td>
<td>Exponential moving average</td>
</tr>
<tr>
<td></td>
<td>SMA</td>
<td>Simple moving average</td>
</tr>
<tr>
<td></td>
<td>TEMA</td>
<td>Triple exponential moving average</td>
</tr>
<tr>
<td></td>
<td>PPO</td>
<td>The percentage price oscillator</td>
</tr>
<tr>
<td></td>
<td>MACD</td>
<td>Moving average convergence divergence</td>
</tr>
<tr>
<td></td>
<td>RSI</td>
<td>Relative strength index</td>
</tr>
<tr>
<td></td>
<td>OBV</td>
<td>On-balance volume</td>
</tr>
<tr>
<td></td>
<td>STOS</td>
<td>Stochastic oscillator</td>
</tr>
<tr>
<td>Exchange rate of USD with some major currencies</td>
<td>USD/CAD</td>
<td>Exchange rate between US dollar and Canadian dollar</td>
</tr>
<tr>
<td></td>
<td>USD/CNY</td>
<td>Exchange rate between US dollar and Chinese Yuan</td>
</tr>
<tr>
<td></td>
<td>USD/EUR</td>
<td>Exchange rate between US dollar and European Dollar</td>
</tr>
<tr>
<td></td>
<td>USD/GBP</td>
<td>Exchange rate between US dollar and British Pound</td>
</tr>
<tr>
<td></td>
<td>USD/JPY</td>
<td>Exchange rate between US dollar and Japanese Yen</td>
</tr>
<tr>
<td>Financial and economic indices</td>
<td>WFC</td>
<td>Wells Fargo stock price</td>
</tr>
<tr>
<td></td>
<td>DAAA</td>
<td>The Moody’s yield on seasoned corporate bonds</td>
</tr>
<tr>
<td></td>
<td>DBAA</td>
<td>The Moody’s yield on month corporate bonds</td>
</tr>
<tr>
<td></td>
<td>DGS</td>
<td>Gold price</td>
</tr>
<tr>
<td></td>
<td>DTB6</td>
<td>Market yield on US Treasury securities at 6 months</td>
</tr>
<tr>
<td></td>
<td>DTB3</td>
<td>Market yield on US Treasury securities at 3 months</td>
</tr>
<tr>
<td></td>
<td>DTB1</td>
<td>Market yield on US Treasury securities at 1 month</td>
</tr>
<tr>
<td></td>
<td>RWTCO</td>
<td>Relative change in the price of the crude oil</td>
</tr>
<tr>
<td>The closing price of another important stock markets</td>
<td>GDAIX</td>
<td>DAX index return</td>
</tr>
<tr>
<td></td>
<td>GE</td>
<td>General Electric stock price</td>
</tr>
<tr>
<td></td>
<td>IXIC</td>
<td>NASDAQ composite price</td>
</tr>
<tr>
<td></td>
<td>DJI</td>
<td>Dow Jones Industrial Average price</td>
</tr>
<tr>
<td></td>
<td>HIS</td>
<td>Hang Seng index price</td>
</tr>
<tr>
<td></td>
<td>SEE</td>
<td>Shang Hai stock exchange price</td>
</tr>
<tr>
<td>The closing price of seven</td>
<td>G00G</td>
<td>Google Inc price</td>
</tr>
<tr>
<td></td>
<td>AAPL</td>
<td>Apple Inc stock price</td>
</tr>
</tbody>
</table>
### 4.4.2 Experiment results

In order to verify the effectiveness of the R-ELM-based SLFN model, the daily closing price of the S&P 500 index is the target output and total 39 multi-type financial and economic factors are used as inputs. So, the input layer includes 39 tapped-delay-line memories, where each tapped-delay-line memory contains a kind of financial or economic factor. According to some studies for stock price forecasting [8], [30], [90], the length of each tapped-delay-line memory is set as 5, then the total number of nodes at the input layer is 195. The data set used in this section is segmented into two parts: the first 80% of the data set is used for training and the remaining 20% is the testing data. According to this segmentation method, total 2732 data samples from time horizon between 19/08/2004 to 24/06/2015 are used as training data and the remaining 683 data samples from the time period between 25/06/2015 to 20/03/2018 are regarded as testing data. All experiments are implemented in MATLAB R2018a [94] in a computer with Intel i7 processor.

In order to measure the predicted performance of the modified SLFN model with the R-ELM algorithm, several experiments are carried out on the one-day ahead S&P 500 closing price forecasting. In previous discussion in section 4.3, the randomly selected range of input weights and hidden biases should be small to effectively map features into feature space. Therefore, the input weights are randomly selected between $-0.001$ to $0.001$ and the hidden layer bias are randomly selected between $-0.01$ to $0.01$. The sigmoid function $g(x) = \frac{1}{1+e^{-x}}$ is applied as the activation function of the hidden layer nodes and the linear function is the activation function of the output layer nodes. In this experiments, rooted mean squared error (RMSE) and mean absolute error (MAE) which are mentioned in previous chapter are used to assess the forecasting performance.
(1) Experimental analysis of parameters

According to previous research in the R-ELM algorithm [106], [108], [109], we use different combinations of parameter $\gamma$ and the number of hidden nodes $N$. The number of hidden nodes $N$ are gradually increased by an interval of 50 in $[50, 2500]$ and $\gamma = [10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}, 10^{5}]$ are recursively grouped to train and test.

![Figure 4.3 Testing RMSEs of different combinations of $(N, \gamma)$](image)

Fig. 4.3 illustrates the testing RMSE of different combinations of $(N, \gamma)$. It is seen that, when $\gamma$ is between 0.1 to 1000, the adjustment of $N$ cannot affect the testing performance. In the R-ELM algorithm, $\gamma$ is used to trade-off between structural and empirical risks. The difference of $\gamma = 10^{4}, 10^{5}, 10^{6}$ is not obvious and some parts of these three values of $\gamma$ entangle to each other in the Fig. 4.3. In order to further investigate the effect of $(N, \gamma)$ on forecasting performance, we only keep the RMSE curves corresponding to $\gamma = 10^{4}, 10^{5}, 10^{6}$ shown in Fig. 4.4.
Fig. 4.4 shows the RMSE performances of the neural model when $\gamma = 10^4, 10^5, 10^6$, respectively. It is seen that, when the number of hidden nodes is less than 250, the model with $\gamma = 10^5$ performed well. However, when the number of hidden nodes is greater than 1250, the model with $\gamma = 10^4$ performed well. The reason about the RMSE performances shown in Fig. 4.4 are still under further investigation. The detailed performances of the model with $\gamma = 10^4, 10^5$ are shown in Table 4-2.
Table 4-2 summarizes the RMSEs and the MAEs of the SLFN model with the values of $\gamma$ and the number of hidden nodes are changed. It is seen that, when $\gamma = 10^4$ and $N = 2450$, the model has the best MAE value (17.69). When $\gamma = 10^5$ and $N = 550$, the model has the best RMSE value (18.01). Moreover, the model with $\gamma = 10^5$ and $N=550$ has lower computational cost, because this model use fewer hidden nodes. In this experiment, the model with $\gamma = 10^5$ and $N = 550$ is regarded as a better choice.

<table>
<thead>
<tr>
<th>$\gamma$</th>
<th>$N$</th>
<th>RMSE</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^4$</td>
<td>2250</td>
<td>25.92</td>
<td>17.91</td>
</tr>
<tr>
<td>$10^4$</td>
<td>2300</td>
<td>25.62</td>
<td>17.74</td>
</tr>
<tr>
<td>$10^4$</td>
<td>2350</td>
<td>25.99</td>
<td>18.01</td>
</tr>
<tr>
<td>$10^4$</td>
<td>2400</td>
<td>25.76</td>
<td>17.72</td>
</tr>
<tr>
<td>$10^4$</td>
<td>2450</td>
<td>25.59</td>
<td><strong>17.69</strong></td>
</tr>
<tr>
<td>$10^4$</td>
<td>2500</td>
<td>25.93</td>
<td>17.86</td>
</tr>
<tr>
<td>$10^5$</td>
<td>500</td>
<td>26.10</td>
<td>18.37</td>
</tr>
<tr>
<td>$10^8$</td>
<td>550</td>
<td><strong>25.53</strong></td>
<td>18.01</td>
</tr>
<tr>
<td>$10^5$</td>
<td>600</td>
<td>25.55</td>
<td>18.10</td>
</tr>
<tr>
<td>$10^5$</td>
<td>650</td>
<td>25.71</td>
<td>18.29</td>
</tr>
<tr>
<td>$10^8$</td>
<td>700</td>
<td>25.99</td>
<td>18.55</td>
</tr>
<tr>
<td>$10^5$</td>
<td>750</td>
<td>26.11</td>
<td>18.75</td>
</tr>
</tbody>
</table>
Fig. 4.5 and Fig. 4.6 show the performances of the R-ELM-based SLFN model with $\gamma = 10^5$ and $N=550$ in the stages of training and testing, respectively. In Fig. 4.6, the red curve is the output of the model and blue curve is the actual closing price of the S&P 500 index. In the period of first 200 days, the forecasting values are close to targets. From the 201th to the 600th days, the stock market is stable and steadily...
increasing. During this period, the forecasting performance is good. In the last 80 days, the performance of model has not been as good as that in the previous periods. One possible reason is that the innovations of stock market occur, so the features learned from historical data are not suitable for current market.

(2) Comparison with the ELM algorithm

In this sub-section, a comparison between the SLFN model trained with the R-ELM and the ELM is given. The ELM developed in [55], [60] is originally developed to optimize the output weights of the SLFN model. In the setup processing of the ELM, the input weights are randomly selected between −0.001 to 0.001 and the hidden layer bias are randomly selected between −0.01 to 0.01. The number of hidden nodes is gradually increased by an interval of 50 in [50 2500]. The activation function of the hidden layer is chosen as sigmoid function.

<table>
<thead>
<tr>
<th>models</th>
<th>$N$</th>
<th>RMSE</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-ELM</td>
<td>550</td>
<td>25.53</td>
<td>18.01</td>
</tr>
<tr>
<td>ELM</td>
<td>100</td>
<td>35.86</td>
<td>28.34</td>
</tr>
</tbody>
</table>

Table 4-3 Performances of the SLFN model with the R-ELM and the ELM
In Table 4-3, the best performance of the model trained with the R-ELM and the ELM algorithms are presented. RMSE and MAE of the model trained with R-ELM
are 25.53 and 18.01, respectively. The values of the R-ELM algorithm are much lower than that of the ELM achieving 35.86 and 28.34, respectively. With the use of regularization term, the SLFN model trained with R-ELM uses 550 hidden nodes which is more than the ELM’s 100. In the theories of the ELM, the hidden nodes number is an important element to improve the learning capability. In many highly noisy environments, a large number of hidden nodes may result in a bad performance. The results of this experiment show that the regularization term can solve this problem perfectly.

The comparison of the ELM and the R-ELM algorithms with different hidden nodes number is shown in Fig. 4.7 and Fig. 4.8. In the processing of the R-ELM, with the regularization term, the RMSE performance is remarkable and stably. Due to noises and disturbances in financial data, when the hidden node number is very large, the SLFN model is affected and perform unsatisfying. With the regularization term, the disturbances and noises can be removed, so the model trained with R-ELM not only has better performance but also high level of robustness.

(3) The comparisons with the classical models

In this section, the proposed R-ELM-based SLFN model is compared with some classical models such as BPM-SLFN (reference to Chapter 3), LR, ARIMA and GARCH. The forecasting target of all these models is the one-day ahead S&P 500 stock closing price. For BPM-SLFN, R-ELM-based SLFN and LR models, the 39 financial factors shown in Table 4-1 are used as the inputs and the delay-units of these inputs are selected as 5. The inputs for ARIMA and GARCH models are the delays of closing price. The length of delay-units is determined from the statistical tests in literature [62]. The RMSE and MAE are shown in Table 4-4 and the forecasting results of BPM-SLFN and R-ELM-based SLFN are illustrated in Fig 4.9 and Fig. 4.10.
Table 4-4 Comparison with a few existing models

<table>
<thead>
<tr>
<th>Models</th>
<th>Testing RMSE</th>
<th>Testing MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-ELM-SLFN</td>
<td>25.53</td>
<td>18.01</td>
</tr>
<tr>
<td>BPM-SLFN</td>
<td>28.56</td>
<td>20.31</td>
</tr>
<tr>
<td>LR</td>
<td>36.16</td>
<td>27.71</td>
</tr>
<tr>
<td>ARIMA</td>
<td>44.12</td>
<td>35.12</td>
</tr>
<tr>
<td>GARCH</td>
<td>41.02</td>
<td>32.12</td>
</tr>
</tbody>
</table>

As shown in Table 4-4, the SLFN model trained with R-ELM exhibits remarkable performances in both the RMSE and the MAE comparing to other models. The testing RMSE and MAE of R-ELM-based SLFN model are 25.53 and 18.01, respectively, which are much smaller than other models. With good hyper-parameter tuning, the SLFN model trained with R-ELM is able to achieve accurate results. The BPM-SLFN model achieves the second best testing performance. These two models use SLFN architecture which is able to capture the dynamic and nonlinear characteristics in stock price data pairs. The LR, ARIMA and GARCH models are traditional linear statistical models, which is unable to capture the nonlinear relationship in financial data pair. Therefore, the performance of these three models is worse than the ones of ANN models. The daily performance comparison on testing horizon is illustrated in Fig. 4.9 and Fig. 4.10.
The predictions by using the R-ELM-SLFN model and the BPM-SLFN model are illustrated in Fig. 4.9 and Fig. 4.10, respectively, where the blue curve is the actual testing data and the red curve is the output of the BPM-SLFN and R-ELM-SLFN. In the Fig. 4.9, it shows that the forecasting results are very close to the
desired daily closing price of the S&P 500. There are small gap between the forecasting results and the actual values after day 600.

4.5 Conclusion

In this chapter, a modified SLFN model trained with the R-ELM has been developed for modelling the daily closing price of the S&P 500 index. The main features of the proposed model are summarized as follows: (i) The tapped-delay-lines are added to the input layer to ensure that the model can effectively capture the dynamics in financial data; (ii) The R-ELM algorithm is used to train the modified SLFN model, to ensure that the model can behave with the strong robustness against noises and disturbances in financial data. The simulation and experimental results have proven that the modified SLFN trained with the R-ELM algorithm exhibits the excellent prediction performance compared with the ELM-based neural model for forecasting the S&P 500 index.
Chapter 5

Conclusion and Future Work

5.1 Summary and conclusions

In this thesis, the ANN models for the predictions of exchange rate and stock price have been investigated. The focus of this thesis is to choose the proper structure of the neural models to ensure that, after training, the models can sufficiently represent the dynamics of the financial input and output data. In addition, the strategy for choosing the appropriate input and output variables of the ANN models has been discussed in detail, together with the learning methods for optimizing both the model structure and the weights. Compared with a few existing traditional time series models, the ANN models developed in this work have exhibited excellent performances for improving the forecasting accuracy and the robustness against noises in financial data. The main contributions of this thesis are summarized as follows.

In Chapter 3, an SLFN model with tapped-delay-lines at input layer of the SLFN has been proposed to model the dynamic and nonlinear relationship between the exchange rate of AUD/USD and the market input variables (technical indicators). With the special design of the input layer, the current technical indicators and prices with time temporal features are used as the input signals, which can greatly help the neural model to extract the financial data features for forecasting purpose. The SLFN model trained with BPM can then avoid the shallow local minima and arrive at global minimum in most of cases. Both theoretical analysis and experimental results have shown the good prediction performance of the SLFN model trained with the BPM.
In Chapter 4, an R-ELM-based SLFN has been developed to model and forecast the daily closing price of the S&P 500 index. It has been seen that the neural model is capable of capturing the dynamics of the stock market from the model output. Through training with R-ELM, the SLFN model behaves with strong robustness against the noised financial data environment. In particular, 39 finance-related factors have been used as the inputs of the neural model for accurately modelling and predicting the daily closing price of the S&P 500 index. The simulation and experimental results have demonstrated the excellent forecasting performance of the stock market. In comparison with the traditional ELM-based algorithm, it is believed that the developed neural model can have stronger robustness against the disturbances and better prediction performance in practice.

5.2 Future work

Based on the research of this thesis, the following further work can be done:

Firstly, in Chapter 3, the tapped-delay-lines are used to improve the capability of ANN model to capture the dynamic features in financial data. It has been noted that this neural model structure is mainly suitable for the financial data prediction with short term memories. However, most financial data have long-term memories. Thus, it is suggested that a novel ANN model that is able to learn the long term memory data features may be developed to forecast financial data with the long-term memories [17], [110].

Secondly, it is noted that a $l_2$ norm term is used in the R-ELM algorithm, for improving the robustness of the SLFN model against the disturbances through training with R-ELM. However, it has been shown in many researches that the $l_1$ norm term has some attractive characters, such as sparsity. It is suggested that by adding a $l_1$ norm term to the cost function, the optimal solution may help the model automatically choose the number of hidden nodes and eliminate some useless features of financial data.
Bibliography


[77] L. C. Martinez, D. N. da Hora, J. R. d. M. Palotti, W. Meira, and G. L. Pappa, "From an artificial neural network to a stock market day-trading system: A case


