Machine learning in stock indices trading and pairs trading

Xiangyu Zong

A Dissertation

Submitted in fulfilment of the requirements of the
Degree of Doctor of Philosophy

Adam Smith Business School
College of Social Sciences
University of Glasgow

May 2021

Abstract

This thesis focuses on two fields of machine learning in quantitative trading. The first field uses machine learning to forecast financial time series (Chapters 2 and 3), and then builds a simple trading strategy based on the forecast results. The second (Chapter 4) applies machine learning to optimize decision-making for pairs trading.

In Chapter 2, a hybrid Support Vector Machine (SVM) model is proposed and applied to the task of forecasting the daily returns of five popular stock indices in the world, including the S&P500, NKY, CAC, FTSE100 and DAX. The trading application covers the 1997 Asian financial crisis and 2007-2008 global financial crisis. The originality of this work is that the Binary Gravity Search Algorithm (BGSA) is utilized, in order to optimize the parameters and inputs of SVM. The results show that the forecasts made by this model are significantly better than the Random Walk (RW), SVM, best predictors and Buy-and-Hold. The average accuracy of BGSA-SVM for five stock indices is 52.6%-53.1%. The performance of the BGSA-SVM model is not affected by the market crisis, which shows the robustness of this model. In general, this study proves that a profitable trading strategy based on BGSA-SVM prediction can be realized in a real stock market.

Chapter 3 focuses on the application of Artificial Neural Networks (ANNs) in forecasting stock indices. It applies the Multi-layer Perceptron (MLP), Convolution Neural Network (CNN) and Long Short-Term Memory (LSTM) neural network to the task of forecasting and trading FTSE100 and INDU indices. The forecasting accuracy and trading performances of MLP, CNN and LSTM are compared under the binary classifications architecture and eight classifications architecture. Then, Chapter 3 combines the forecasts of three ANNs (MLP, CNN and LSTM) by Simple Average, Granger-Ramanathan's Regression Approach (GRR) and the Least Absolute Shrinkage and Selection Operator (LASSO). Finally, this chapter uses different leverage ratios in trading according to the different daily forecasting probability to improve the trading performance. In Chapter 3, the statistical and trading performances are estimated throughout the period 2000-2018. LSTM slightly outperforms MLP and CNN in terms of average accuracy and average annualized returns. The combination methods do not present improved empirical evidence. Trading using different leverage ratios improves the annualized average return, while the volatility increases.

Chapter 4 uses five pairs trading strategies to conduct in-sample training and backtesting on 35 commodities in the major commodity markets from 1980 to 2018. The Distance Method (DIM) and the Co-integration Approach (CA) are used for pairs formation. The Simple Thresholds (ST) strategy, Genetic Algorithm (GA) and Deep Reinforcement Learning (DRL) are employed to determine trading actions. Traditional DIM-ST, CA-ST and CA-DIM-ST are used as benchmark models. The GA is used to optimize the trading thresholds in ST strategy, which is called the CA-GA-ST strategy. Chapter 4 proposes a novel DRL structure for determining trading actions, which replaces the ST decision method. This novel DRL structure is then combined with CA and called the CA-DRL trading strategy. The average annualized returns of the traditional DIM-ST, CA-ST and CA-DIM-ST methods are close to zero. CA-GA-ST uses GA to optimize searches for thresholds. GA selects a smaller range of thresholds, which improves the in-sample performance. However, the average out-of-sample performance only improves slightly, with an average annual return of 1.84% but an increased risk. CA-DRL strategy uses CA to select pairs and then employs DRL to trade the pairs, providing a satisfactory trading performance: the average annualized return reaches 12.49%; the Sharpe Ratio reaches 1.853. Thus, the CA-DRL trading strategy is significantly superior to traditional methods and to CA-GA-ST.

Contents

List of Tables

List of Figures

Acknowledgments

I would like to express my gratitude to those who have helped me in my Ph.D. study.

I have received plenty of support from my family. My parents are very supportive of my study, and have provided me with financial support and spiritual encouragement. My wife has always been with me in the UK and greatly supported me in my everyday life.

I would like to express my deepest gratitude to my supervisors Professor Georgios Sermpinis and Dr. Charalampos Stasinakis. I met Professor Georgios Sermpinis when I was studying for a master's degree at the University of Glasgow and he has guided me in the right research direction. During my studies, he has identified errors and provided me with valuable suggestions. His rigorous guidance has been very helpful for my academic progress. Dr. Charalampos Stasinakis also often discusses my progress with me. He has always listened very patiently to my description of my research, even if it may be problematic sometimes. He has given me advice both on study and life. His encouragement has helped me greatly in times of difficulties. Without the help of Professor Georgios Sermpinis and Dr. Charalampos Stasinakis, I would not have been able to complete my Ph.D. research. Thus, I would like here to thank them once again sincerely.

I acknowledge the fully funded support from the China Scholarship Council and would like to thank all the staff of the education section of the Chinese Embassy in the UK.

Moreover, I would like to thank all the following friends: Pengcheng Song, Zerun Liu, Zhekai Zhang, Yihan Zou, Jinpeng Liu, Yujin Chi and Arman Hassanniakalager, who have given me a great deal of help in my academic studies. Lastly, I am grateful to all the staff and colleagues at Adam Smith Business School who create such a favourable research environment.

Declaration

I declare that, except where explicit reference is made to the contribution of others, that this dissertation is the results of my own work and has not been submitted for any other degree at the University of Glasgow or any other institution.

Printed Name: Xiangyu Zong

Signature:

Abbreviations

ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
ADF	Augmented Dickey–Fuller
AIS	Artificial Immune System
ANN	Artificial Neural Networks
AR	Autoregression
ARIMA	Autoregressive Integrated Moving Average
BFA	Bacterial Foraging Algorithm
BGSA	Binary Gravity Search Algorithm
BP	Back-propagation
BRNN	Bayesian Regularized artificial Neural Network
BSE	Bombay Stock Exchange
CA	Co-integration Approach
CBR	Case Based Reasoning
CFO	Central Force Optimization
CGARCH	Component GARCH
CNN	Convolution Neural Network
CRBM	Continuous Restricted Boltzmann Machine
DBN	Deep Belief Network
DDPG	Deep Deterministic Policy Gradient
DIM	Distance Method
DM	Diebold-Mariano test
DNN	Deep Neural Network
DRL	Deep Reinforcement Learning
DT	Decision Tree
EB-CNN	Event Embeddings input Convolutional Neural Network
EGARCH	Exponential GARCH
EMH	Efficient Market Hypothesis
ESM	Exponential Smoothing Model
EWMA	Exponentially Weighted Moving Average
FNN	Fuzzy Neural Network
GA	Genetic Algorithm

GARCH	Generalized Autoregressive Conditional Heteroscedasticity
GD	Gradient Descent
GRR	Granger-Ramanathan's Regression approach
GSA	Gravity Search Algorithm
GW	Giacomini-White test
HMM	Hidden Markov Model
IBCO	Improved Bacterial Chemotaxis Optimization
ICA	Independent Component Analysis
IGSA	Improved Gravitational Search Algorithm
KNN	K-Nearest Neighbour
KPCA	Kernel Principal Component Analysis
LASSO	Least Absolute Shrinkage and Selection Operator
LDA	Linear Discriminant Analysis
LPP	Locality Preserving Projection
LSTM	Long Short-Term Memory
MA	Moving Average
MLP	Multi-Layer Perceptron
M-P	McCulloch-Pitts model
MRAR	Morningstar Risk-Adjusted Return
NAV	Net Asset Value
OPF	Optimal Power Flow
PCA	Principal Component Analysis
PE	Processing Element
PG	Policy Gradient
PNN	Probabilistic Neural Network
PPO	Proximal Policy Optimization
PSO	Particle Swarm Optimization
RBF	Radial Basis Function
RBM	Restricted Bolzmann Machine
RGA	Real Genetic Algorithm
RL	Reinforcement Learning
RNN	Recurrent Neural Network
RW	Random Walk
SA	Simulated Annealing
SGD	Stochastic Gradient Descent

SSD	Sum of Euclidean Squared Distance
ST	Simple Threshold
SVM	Support Vector Machine
SVR	Support Vector Regression
SZII	Shenzhen Integrated Index
TDNN	Time Delay Neural Network
VaR	Value at Risk
VAR	Vector Autoregressive

Chapter 1 Introduction

1.1 Background and motivation

The performance of quantitative trading is mainly affected by two factors: the forecast of the future and the rules of trading strategies. A high forecasting accuracy rate provides the potential to establish a profitable portfolio for quantitative trading. A reasonable trading strategy further enhances the trading performance and even brings portfolios close to arbitrage. In the above steps, the keys to improving the quantitative trading performance are making a valid forecast and making the optimum trading decisions in trading strategies. For traders, the trading process can also be divided into these two steps, to judge the trend of future asset prices and make trading decisions based on experience and established rules. These jobs require traders to accumulate extensive experience. Machine learning provides a possibility that inexperienced traders can complete these two key steps of trading and gain profits relying on machine learning algorithms.

Machine learning techniques have two significant improvements in recent decades. The first was in the late 1970s when MLP was proven to be able to fit non-linear functions, which theoretically implies that machine learning algorithms can learn any type of functions and have the potential to solve many complex problems. The second was in the last two decades, machine learning algorithms are more advanced and have wide applications with the improvement of computer hardware technology. The application of machine learning algorithms improves productivity. Machine learning techniques provide unparalleled efficiency in carrying out a large number of repetitive tasks, such as image and text recognition. For example, the classification of thousands of images is time-consuming for humans, while machine learning algorithms can quickly classify images. Additionally, machine learning has also made progress in dealing with complex and fuzzy problems in the past decade, such as AlphaGo (which is proficient at playing the game Go). The capability of performing challenging tasks further broadens the applications of machine learning algorithms.

The prediction and decision-making in financial markets are complex and fuzzy tasks, which

are also the focuses of machine learning algorithms. This study uses machine learning algorithms to complete the two crucial steps in quantitative trading: financial time series forecasting (Chapters 2 and 3) and trading decision-making (Chapter 4). Financial time series forecasting is a task with multi-constraints and various objectives, as it needs to take into account high-dimensional financial data, behavioural factors and other exogenous effects. Therefore, linear models often face challenges in processing high-dimensional and non-linear data. It is also difficult for traders to forecast asset prices based on their experience, since they are incapable of making accurate calculations on a large amount of messy data. However, machine learning-based techniques have had promising performances in trading applications (Van-Gestel et al., 2001; Kim, 2003; Sermpinis et al., 2016, etc.). Many studies use machine learning techniques, such as clustering algorithms (Shen et al., 2011; Lai et al., 2009), SVMs (Kim, 2003; Huang, 2005; Dunis et al., 2013, etc.) and ANN (Guresen et al., 2011; Kim & Han, 2000; Fernandez-Rodrıguez et al., 2000; Jasic, 2004) to predict financial time series.

Chapters 2 and 3 focus on two main types of machine learning methods: hybrid SVMs and Deep Neural Networks (DNNs). Although these two methods are widely used in forecasting financial time series, there are gaps in the model improvement that need to be filled. As SVM's performance is significantly affected by parameters and inputs, Chapter 2 improves SVM's performance by optimizing the parameters and inputs with BGSA, and then forecasts the sign of return for stock indices. Regarding DNN, although previous studies apply various types of DNNs to the task of stock indices forecasting, they use a relatively small number of layers in DNNs and do not compare mainstream DNNs under the same inputs, which are the gaps when using DNN in stock indices forecasting. Chapter 3 uses MLP, CNN, LSTM and their combined method to forecast stock indices. The aims of Chapter 3 are to adjust the structure of those Neural Networks (NNs) to fit the financial time series data, achieve higher accuracy, and compare the performances of different NNs to fill the gap of the literature.

In respect of trading strategy optimization, the use of machine learning methods to optimize trading actions is a promising and cutting-edge field. Two reasons for choosing machine learning methods to optimize trading strategies are worth noting. First, the optimization of trading actions is difficult for traditional methods, while machine learning methods are adept in solving this type of problem. The second reason is that the trigger conditions of the trading actions of the traditional trading strategy are relatively inflexible, and thus a smarter 'brain'

is required to make reasonable trading actions. Compared with conventional learning tasks, dynamic action optimization is more challenging due to the lack of supervised information from human experts. Among different kinds of machine learning algorithms, DRL provides an excellent model framework for dynamic action optimization tasks, and has made a good performance in playing games (Guo et al., 2014; Mnih et al., 2015; Levine et al., 2016; Watter et al., 2015). In recent years, DRL has gradually been applied to the field of quantitative trading (Deng et al., 2016; Buehler et al., 2019; Xiong et al., 2018).

Chapter 4 focuses on optimizing the trading strategy for pairs trading by DRL and GA. Pair trading is a relatively mature trading method, which has been used as early as the 1980s. Although the traditional pairs trading methods, such as CA-ST and DIM-ST, have been successfully applied previously, there are still several questions to be addressed. *Can these traditional methods achieve statistical arbitrage in the recent futures market? How can investors tap the potential of pairs trading strategies and get higher returns? There is no literature on using DRL in pairs trading portfolios, can DRL outperform traditional methods for pair trading?* The superiority of machine learning techniques in optimizing trading strategies and these questions motivate Chapter 4.

1.2 Structure and contribution

The main focus of this thesis is to develop machine learning models for forecasting tasks and trading decision-making tasks. Chapters 2 and 3 improve the SVMs and NNs to solve the forecasting task of financial time series, and successfully improve the forecasting accuracy and returns. In Chapter 4, GA and DRL are used to optimize the trading actions of pairs trading. Compared with the traditional method, DRL successfully improves the return under the same risk. Chapter 5 makes a conclusion for the thesis and explains the limitations and future works.

The main contribution of Chapter 2 is that it introduces a novel BGSA-SVM machine learning model that is suitable for forecasting financial time series. The parameters and inputs of SVM are optimized by BGSA for the first time and have achieved a better performance than SVM. Much literature focuses on optimizing SVM (especially with heuristic methods), as previous literature emphasizes that SVM's performance is

significantly affected by parameters. Compared with hybrid SVM models of prior literature, the advantage of BGSA-SVM is that it can forecast the financial time series with high-dimensional inputs while losing less information. This means that BGSA-SVM is more promising to have a higher performance. Another contribution of Chapter 2 is to successfully improve the forecasting accuracy of stock indices, proving that BGSA-SVM is significantly better than benchmark models with Diebold-Mariano (DM) and Giacomini-White (GW) tests. Trading performances show that BGSA-SVM beats all other benchmark models in average returns and the Sharpe Ratio. The high forecasting accuracy and the best trading performance prove the ability of BGSA-SVM for stock index forecasting. The results of BGSA-SVM support the view that SVM's forecasting performance can be significantly improved by optimizing the parameters and inputs of SVM.

The contributions of Chapter 3 are described in three aspects. The first is that it uses MLP, CNN, and LSTM to forecast stock indices in the same inputs pool, and improves the structures of MLP, and CNN for stock indices forecasting. The results indicate that LSTM is slightly better than the other two NNs in terms of average performance. This phenomenon shows that the improvement of neural networks' algorithms only has a small impact on stock indices forecasting, which is due to the limitation of the information contained in the inputs. A significant increase in forecasting accuracy requires the selection of better inputs. Second, Chapter 3 provides evidence that the pre-processing of inputs affects the performances of NNs. It uses different inputs for MLP, finding that more inputs do not lead to a better result as inefficient inputs reduce the model's forecasting accuracy. Third, a leverage rule is designed in Chapter 3 based on the daily forecast probabilities given by NNs, which improves the average annualized return and Sharpe Ratio.

Chapter 4 makes three main contributions. First, it uses a novel CA-DRL for the first time to trade a large number of commodities and successfully improves the trading performance. The empirical results show that traditional methods only generate very small returns, while CA-DRL yields significantly higher annualized returns with similar risks. Second, Chapter 4 solves two crucial problems in the DRL for pairs trading: 1) falling into the local optimum; 2) the huge amount of calculation. The pre-training technique is adopted to solve these two problems and enable DRL to form portfolios with a great number of pairs. In the pre-training process, one selected pair (as the pre-training sample) is trained multiple times to obtain both good in-sample and out-of-sample performances. The connection weights of the DNN (the

brain of DRL) in the pre-trained model are used as the initial connection weights of DNNs for all other DRL models. The pre-training method makes the training repeatable and reduces the training times for other DRL models. Third, GA is applied to optimize the parameters in ST, which slightly improves the annualized returns. The performance of CA-GA-ST indicates that even if the traditional method obtains the optimal in-sample solution, it still cannot significantly improve out-of-sample performance. Chapter 4 has been modified to a paper and submitted to a journal.

Chapter 2 A stock index trading strategy based on Binary Gravity Search Algorithm and Support Vector Machine

2.1 Introduction

In this chapter, a hybrid Support Vector Machine (SVM) model is proposed and applied to the task of forecasting daily returns of five popular stock indices in the world, including S&P500, NKY, CAC, FTSE100 and DAX. The trading application covers the 1997 Asian financial crisis and 2007-2008 global financial crisis. The originality of this work is that the Binary Gravity Search Algorithm (BGSA) is utilized, in order to optimize the parameters and the inputs of SVM. The results show that the prediction of this model is distinctly better than the RW, SVM, best predictors and Buy-and-Hold. The average accuracy of BGSA-SVM for five stock indices is 52.6%-53.1%. The performance of BGSA-SVM model is not affected by the market crisis, which shows the robustness of this model. In general, this chapter proves that a profitable trading strategy based on BGSA-SVM prediction can be realized in a real stock market.

As one kind of the most widely used machine learning algorithms, SVMs are expected to forecast the financial market based on the feature of SVMs, particularly in stock index prediction. For example, the daily return of a stock index is determined by many factors and some of them are vague. Additionally, the relationship between these factors with the daily return is non-linear. When using related historical features and some samples to train SVMs, the SVMs are able to forecast the stock index. However, their forecasting performances are significantly affected by inputs and parameters (Mukherjee et al., 1997; Trafalis & Huseyin, 2000; Kim, 2003). Unsuitable inputs and parameters cause over-fitting, under-fitting or low accuracy in the out-of-sample. Thus, selecting proper inputs and parameters is essential in SVM training. This study employs a heuristic search algorithm called Binary Gravity Search Algorithm to optimize the inputs and parameters of the SVM classifier. Heuristic search algorithms have a great deal of potential when it comes to dealing with high-dimensional optimization problems with multiple parameters. The optimization of the SVM classifier is a task with multiple discontinuous parameters with multi-peak. BGSA yields superior performance in solving this kind of problem (Rashedi & Nezamabadi-pour, 2012). Hence, a

hybrid BGSA-SVM model is adopted here that aims to forecast the sign for the log return of stock indices.

One motivation for this chapter is to provide an effective prediction model that can be utilized as a basis for trading strategies. The expected accuracy of RW is 50%. An effective prediction model can significantly impact the trading performance and bring a sizable increase in returns by obtaining above 50% expected accuracy. The other motivation is to introduce a novel hybrid SVM model that improves the SVM model in financial time series forecasting.

The rest of the chapter is organized as follows. The literature review on SVMs to predict financial time series and heuristics is presented in section 2.2. Section 2.3 describes the dataset employed for this study. The algorithms and the model structure are examined in section 2.4. Then the statistical evaluation, trading performance and the conclusion are respectively presented in sections 2.5, 2.6 and 2.7.

2.2 Literature Review of SVM and heuristic methods in finance

2.2.1 Support Vector Machines

The original SVM was generated by Vapnik in 1979. Since Vapink improved it in 1995, SVMs have been applied in various research fields as the non-linear classifier and non-linear regression, such as pattern recognition, speaker identification, density estimation, benchmark time series prediction, credit rating prediction and bankruptcy prediction. SVMs adapt to solve classification and regression problems with high-dimensional inputs, especially in picture recognition and text classification (Schuldt et al., 2004). The soft margin method and the kernel function are implemented to address the problem of linear inseparability in SVM optimization. SVM can successfully deal with a small number of training sets with higher classification accuracy than the traditional technique (Mantero et al., 2005). The learning process of SVM follows structural risk minimization, which allows it to minimize errors on unseen data without probability distribution assumptions, whereas statistical methods usually need to have prior assumptions of data distribution such as maximum likelihood estimation (Mountrakis et al., 2011).

SVMs have generally been used in time series forecasting. For instance, Mukherjee et al. (1997) apply a Support Vector Regression (SVR) in chaotic time series forecasting as a new regression technique at that time. Comparing the performance of SVR with different approximation techniques, they find that the SVR outperforms neural networks, polynomial and rational approximation, local polynomial techniques and Radial Basis Functions. In addition, they conclude that the SVR is a very promising regression technique. Mukherjee et al. (1997) also discuss the sensitivity of SVR to parameters and inputs dimension. However, they do not offer any specific solutions for determining the dimensions of parameters and inputs. Similarly, Trafalis & Huseyin (2000) apply SVR to predict the stock prices of AOL, IBM and YAHOO. They state that SVR is a robust method for function approximation, compared with Radial Basis Function Networks and Back-propagation algorithm (BP) in financial time series. With regard to the selection of parameters, in contrast to Mukherjee et al. (1997), Trafalis & Huseyin (2000) fix two parameters in SVR and adjust only one to observe the final predictive performance of SVR. The impact of the number of inputs on the forecasting performance is not considered by Trafalis & Huseyin (2000). The kernel functions used in both studies are Radial Basis Function (RBF).

After many studies in financial time series, Cao & Tay (2001; 2002; 2003) conclude that SVMs have better performances than the BP neural networks, with the criteria including directional symmetry, normalized mean square error, mean absolute error and weighted directional symmetry. In addition, they analyze four reasons for this superior performance: the structural risk minimization principle, fewer free parameters, converging to global solutions, less care and experience required than for the validation set in BP networks. They also indicate that the selection of parameters has a significant impact on forecasting accuracy and therefore, they continue to research the selection of the kernel function and parameters of SVMs.

Kim (2003) has a similar view as he adopts SVMs to forecast stock indices and compares them with BP neural networks and Case Based Reasoning (CBR). He also stresses the importance of selecting parameters. Ince & Trafalis (2006) predict exchange rates with a hybrid two-stage forecasting model, concluding that the hybrid SVM model and Autoregressive Integrated Moving Average (ARIMA) or Vector Autoregressive (VAR) outperform the hybrid model of the ANN with these two techniques. However, it is not

enough to judge the pros and cons of SVMs and neural networks just on the accuracy and peak value. The classification hyper-plane of SVMs is continuous and smooth, while it is discontinuous in BP neural networks, which leads to the result that BP neural networks are more sensitive to the interference of noise. SVMs and neural networks are both excellent classification tools. Their performances depend on the way they are used; for instance, in the above research, the inputs used are beneficial for SVMs, which cannot indicate that SVMs are more adaptable than BP neural networks in terms of financial time series forecasting.

The SVM classifier is also used to classify stocks. Fan & Palaniswami (2001) use the SVM to classify the stocks on the Australian Stock Exchange. They seek to select outperforming stocks that beat the market. They seem to get a good result, as the portfolio of selected stocks remarkably outperforms the benchmark during five test years, reaching 207% compared with the result (71%) of the benchmark. However, Fan & Palaniswami use data from 1992-2000 when the stock market kept rising, which throws doubt on their result. The stocks they select are partial to high market risk, which means the portfolio tends to lose more than the benchmark when the whole market goes down.

2.2.2 Data pre-processing methods for SVMs

Data pre-processing is an important component in a supervised learning model. In a training set, many variables are related, or vary greatly in order of magnitude, which is disadvantageous for learning algorithms. Therefore, data pre-processing methods are needed to optimize the data matrix with a higher information repetition rate, such as dimensionality reduction, normalization and so on. Some research improves the accuracy of SVM models by optimizing the training set. For example, Lu et al. (2009) indicate that inherent high noise is a key problem for financial time series forecasting. To reduce the impact of inherent noise, they use Independent Component Analysis (ICA) to optimize the inputs of the SVR model. Their empirical results show that the model that pre-processes data with ICA performs better than the model with a non-filtered training set and an RW model. In addition to ICA, Principal Components Analysis (PCA) is also used to optimize the inputs.

PCA is an exploratory multivariable technique for transferring correlated variables to a smaller number of uncorrelated variables (Jackson, 2005). Principal components are the linear combination of variables with different weights (called eigenvectors). PCA represents

intricate multi-dimensional data with fewer principal components without losing much valuable information. As a dimensionality reduction method, PCA can reduce the inputs of a machine learning model significantly, which makes the whole algorithm faster and more effective while losing less information.

In Neumann's (2002) software risk categorization model, PCA is used to provide a way of normalizing the input data and making it orthogonal, thus eliminating the negative impacts of multicollinearity. The machine learning algorithm used for Neumann's model is ANN, which classifies high-risk software successfully. Similarly, Yetilmezsoy & Demirel (2008) use PCA-ANN to predict the efficiency of Pb (II) ions' removal from an aqueous solution by Antep pistachio shells. Although they are different fields, the algorithms used in these two studies are the same. In the study of Yetilmezsoy & Demirel (2008), PCA is also utilized to pre-process training data. In addition, Choi & Park (2001) apply multivariate regression, ANN and a hybrid method that combines PCA as a pre-processing stage to data from industrial wastewater processes. Actually, the hybrid technology used in their research is PCA-ANN. Their empirical results show that PCA-ANN enhances prediction capability and decreases the overfitting problem of neural networks. Moreover, PCA-ANN has the best information extraction capability in its benchmark model.

The hybrid prediction model of PCA and SVM is also applied in many fields. Cao et al. (2003) use PCA, Kernel Principal Component Analysis (KPCA) and ICA to extract features and then use SVM for training and forecasting. They examine the sunspot data, Santa Fe data set A and five real futures contracts. Their result indicates that SVM with feature extraction, by PCA, KPCA or ICA, performs better than without feature extraction. Among these three hybrid techniques, they also find KPCA-SVM performs best in their time series tests, but do not explain which of these three pre-processing methods is best for different types of data. In the signal recognition field, Subasi & Gursoy (2010) use SVM to predict seizures and apply PCA, ICA and Linear Discriminant Analysis (LDA) to reduce data dimensions. In their research, there is not much difference among the performances of the three data pre-processing methods, while LDA-SVM has the best predictive performance.

In financial time series forecasting, there are many examples of using PCA as a data pre-processing method and using SVMs for prediction (Yu et al., 2014; Sermpinis & Stasinakis, 2017). In stock forecasts, Zahedi & Rounaghi (2015) accurately predict stock prices on the

Tehran Stock Exchange with artificial neural networks and PCA. Additionally, Yu et al. (2014) use SVM with PCA to select stocks on the Shanghai Stock Exchange market. The return of their portfolio is higher than the stock index. However, Yu et al. (2014) do not mention the parameter selection. The over-fitting of Mean-standardization PCA-SVM in their paper is very likely to have occurred.

2.2.3 Heuristics

Heuristic algorithms are optimization methods that can be used to optimize the inputs and parameters of SVMs. Many hybrid SVM algorithms have been used to predict financial prices and have achieved better results than classical SVMs. The use of heuristics in conjunction with SVMs in these hybrid algorithms is widespread.

Heuristics are techniques that aim to control the computational cost at an acceptable level while seeking good (near-optimal) solutions. However, heuristics cannot guarantee feasibility or optimality (Russell & Norvig, 2003). The typical algorithms of heuristic algorithms include Artificial Immune System (AIS) (Farmer et al., 1986), Ant Colony Optimization (ACO), GA (Tang et al., 1996), Bacterial Foraging Algorithm (BFA) (Gazi & Passino, 2004), Particle Swarm Optimization (PSO) (Kennedy, 2011), and Simulated Annealing (SA) (Kirkpatrick et al., 1983).

Heuristic algorithms mimic physical or biological processes to find solutions that come close to the optimum. For example, AIS mimics the biological immune system. GA is inspired by Darwin's theory of evolution. ACO simulates ants searching for food (Dorigo & Caro, 1999). PSO mimics the movement of a flock of birds (Bergh & Engelbrecht, 2006). Central Force Optimization (CFO), established by Formato (2007), is a deterministic heuristic search algorithm based on gravitational kinematics to initialize several random particles, iterate and find the optimal solution. These heuristic algorithms are widely used to solve optimization problems with great computational demands and optimization problems without analytical solutions in various fields.

The Gravity Search Algorithm (GSA) has similar principles to CFO. In GSA, the solutions are also called agents. Due to gravity, they interact with others and move towards those with heavier mass, and the best solution is the one with the heaviest mass. Rashedi et al. (2009)

propose GSA based on gravity and mass interactions. They find GSA is a strong search algorithm that outperforms CFO, PSO and the Real Genetic Algorithm (RGA) in most cases, for both unimodal and multimodal functions. Rashedi et al. (2009) also generalize the GSA to a binary system in 2010 and introduce BGSA, which is more suitable for solving discrete problems.

Recently, GSA and BGSA have been successfully applied to solve optimization problems in many fields. Bahrololoum et al. (2012) use GSA to solve classification problems and find the best positions for the representatives in a UCI machine learning repository. In their tests, GSA's performance is better than that of an artificial bee colony and particle swarm optimization. Duman et al. (2012) use GSA to find the optimal solution for an Optimal Power Flow (OPF) problem in a power system. Their simulation results indicate that "GSA provides an effective and robust high-quality solution for OPF problem." BGSA is proposed to select features to improve the precision of Content-based image retrieval by Rashedi & Nezamabadi-pour (2012). Their tests confirm the efficiency of BGSA in selecting features.

In addition to BGSA, other derivative algorithms of GSA are also applied to solve some specific problems. Li & Zhou (2011) propose an Improved Gravitational Search Algorithm (IGSA) and apply it to the parameter identification of a hydraulic turbine governing system. The IGSA proposed by Li & Zhou (2011) is the combination of the search strategy of PSO and GSA. In their experiments, IGSA is better than GSA, GA and PSO in terms of convergence speed and parameter accuracy. Similarly, Mirjalili, Hashim & Sardroudi (2012) propose a hybrid of PSO and GSA (PSOGSA), which is used as new training techniques for feedforward neural networks in order to examine the efficiency of the algorithm in decreasing the dilemma of getting trapped in local minima and the converging speed of the learning algorithm. Their PSOGSA algorithm improves the searching speed in the final iterations. For the problem of slow searching speed in the final iterations, Shaw, Mukherjee & Ghoshal (2012) also propose an opposition-based GSA to accelerate the performance of GSA. Their research employs opposite numbers to improve the convergence speed of GSA and obtains promising results.

In summary, most heuristic search algorithms use multiple initial points to search in parallel, including BGSA and GSA. In an algorithm based on a parallel search with multiple initial points, each individual takes a series of special operations and shares the information with

other individuals. The operation of each individual is simple, but their collective impact, called swarm intelligence (Tarasewich & McMullen, 2002), generates astonishing results.

2.2.4 Combination of SVMs with Heuristics in finance

Parameters and inputs have a significant impact on the forecasting performance of SVMs, and this causes SVMs to be sensitive to parameters' optimization. Therefore, in many studies, hybrid machine learning methods that combine heuristic algorithms (used as optimization techniques) and SVMs are utilized as prediction methods. In the financial field, these hybrid models are widely proven to have higher predictive accuracy than classical SVMs.

GA-SVM is the most common hybrid Heuristic-SVM model in the financial field. Min, Lee & Han (2006) use GA to optimize both parameters of SVM and a feature subset simultaneously for bankruptcy prediction. Wu et al. (2007) also use GA-SVM to predict bankruptcy. Nevertheless, in the study by Wu et al. (2007), the GA only optimizes two parameters of the SVM. Compared with the work of Min, Lee & Han (2006), the efficiency of GA in Wu et al.'s model is low. This is because, in the case of a small number of optimized parameters (Wu et al. only optimize two parameters), GA requires more calculation times than grid search. Hong (2006) uses the same algorithm to predict exchange rates. He uses the GA to select the parameters of linear and non-linear SVM. These three papers all show that the out-of-sample accuracy of GA-SVM is higher than that of SVM. Additionally, Dunis et al. (2013) use a method similar to Hong's (2006) to predict stock indices (FTSE100 and ASE20). They compare GA-SVM with the moving average, high order neural network, naive Bayesian classifier, etc. and find that GA-SVM outperforms the above benchmark models.

In recent years, in addition to GA-SVM, more and more other heuristic optimization algorithms have been used in conjunction with SVM and applied in financial research. Zhao et al. (2015) use VAR to measure the relationship between oil price and both market factors and non-market factors, in order to obtain a more accurate prediction for crude oil prices. They combine VAR and SVM to build a VAR-SVM model, finding that their VAR-SVM model is superior in accuracy compared with the ANN, Component GARCH (CGARCH) and VAR models. Sermpinis & Stasinakis (2013; 2014; 2015; 2017) have conducted abundant research on financial forecasting with hybrid SVMs. GA-SVM is utilized to predict

inflation, unemployment (Sermpinis et al., 2014) and EUR exchange rates (Sermpinis et al., 2015). Sermpinis, Stasinakis & Hassanniakalager (2017) apply a novel heuristic algorithm called the Kill Herd algorithm to combine with SVR and Locally weighted SVR (LSVR) for forecasting and trading exchange traded funds. In their study, Krill Herd-LSVR shows superior predicting power over GA-LSVR and SVR.

Chen et al. (2013) use the Artificial Bee Colony (ABC) algorithm in conjunction with SVM to predict corporate credit rating problems for the USA during the period 2001-2008. Likewise, Zhiqiang et al. (2013) forecast the Shanghai stock market index and Dow Jones index with Locality Preserving Projection (LPP) and SVM optimized by PSO. They use LPP to reduce the dimension of the training data and PSO to optimize the parameter of SVM. However, the data collected by Zhiqiang et al. (2013) do not include the financial crisis (Shanghai stock market index data is collected from 2000 to 2004; Dow Jones index data is collected from 1996 to 1998), so there can be no proof of the robustness of their model.

The combinations of GSA or BGSA with SVM have been used in some research, such as Sarafrazi & Nezamabadi-pou (2013) and Li et al. (2015). As one of the inventors of GSA and BGSA, Nezamabadi-pour and his collaborators tested the classification capabilities of BGSA-SVM in 2013. They tested the BGSA-SVM with eight sets of well-known machine learning data repository of the University of California UCI Machine Learning Repository (Center for Machine Learning and Intelligent Systems) and compared the BGSA-SVM with GA-SVM and PSO-SVM. Ultimately BGSA-SVM performs better than GA-SVM and PSO-SVM. Sarafrazi & Nezamabadi-pour (2013) conclude that BGSA is more efficient in feature subset selection than benchmark models. Li et al. (2015) propose another improved GSA and combine it with SVM to predict 14 sets of UCI data. Their chaos-embedded GSA-SVM also successfully performs better than GA-SVM and PSO-SVM in their experiments.

However, this BGSA-SVM has not been applied in financial forecasting. Therefore, this study uses the BGSA-SVM algorithm and fills this gap. BGSA is employed to select the parameters and inputs of SVM classifiers. BGSA performs better than some earlier developed heuristic optimization methods such as ant colony optimization and GA in solving the high-dimensional multiple-peak disperse function, while the mapping of the parameters and inputs of SVM to the SVM predictive performance is a high-dimensional multi-peak disperse function. These are the reasons for adopting BGSA in this study.

2.3 Dataset

This study uses BGSA-SVM to predict the sign of log return of 5 stock indices, including S&P500, NKY, FTSE100, CAC40 and DAX. The sign of log return is the output y. If the daily return is positive, then $y = 1$. If the daily return is negative, $y = -1$. The data were collected from 1990-2016, which includes the 1997 Asian financial crisis and financial crisis of 2007–2008. Thus, the performance of this model in extremely poor financial situations is investigated. The summary of log returns is shown in Table 2.1.

Table 2.1: The summary of log returns

	S&P500	FTSE100	NKY	CAC40	DAX
Mean of return	0.000133%	0.00328%	0.00212%	0.00159%	0.0138%
Standard deviation of return	0.012267	0.12061	0.015396	0.014741	0.015174
Skewness of return	-0.183518	-0.149663	-0.391722	-0.028207	-0.053529
Kurtosis of return	11.45306	9.145355	9.158393	7.827611	7.327344
Jarque-Bera of return (p value)	0.0000	0.0000	0.0000	0.0000	0.0000
ADF (p value)	0.0001***	0.0000***	0.0001***	0.0000***	0.0001***

Note: The model aims to predict the sign of the log returns. *** denotes that the hypothesis of ADF test is rejected at the 1% significance level.

Table 2.1 indicates that the five stock indices time series are non-normal and stationary, while the skewness is all negative and the kurtoses are all high. All returns series exhibit small skewness and high kurtosis. The Jarque–Bera statistic confirms that the five return series are non-normal at the 99% confidence level. The Augmented Dickey–Fuller (ADF) reports that the null hypothesis of a unit root is rejected at the 99% statistical level for all stock indices.

The total number of independent variables is 189. I select some of the parameters used by Kim (2003), Trafalis & Ince (2000), Cao & Tay (2003), Dunis et al. (2013) and Sermpinis et al. (2017), which are given in Appendix A.1.1. The final model for training and predicting uses 15 to 25 dimensional data that are processed by PCA and BGSA, rather than 189-dimension vectors.

Due to some missing independent variables in the dataset, I remove the days of missing data to guarantee train set intensity. This model runs 11 tests of each stock index from 1997 except the UK FTSE100 index. For every test, the start and end dates of the training set, test set, and out-of-sample are all shown in Appendix A.1.2. It groups 252 trading days as a trading year for all five indices. Six trading years are regarded as an in-sample, and the

subsequent year (the seventh year) is an out-of-sample. Owing to the different missing data in the five stock indices, the amounts of processed data are different, and the FTSE100 index has the most missing data. Therefore, the FTSE100 index has only nine tests. The start and end dates of each stock index are also different because of the different missing data in the five stock indices. This means that the time of training and forecasting is not completely identical in the five stock indices. This data-processing method does not influence the accuracy of tests in the model, and it reduces the usage of data, simplifying the process of the training model.

Compared with employing the demarcation of the financial crisis to decide different periods of prediction and analyses, using the fixed period here is more practical for two reasons. First, the financial crisis is defined after having happened, and cannot be predicted. Therefore, when using historical data, there is no need to regard the financial crisis as the boundary for splitting into three parts: before, during and after the financial crisis. Before the crisis happens, people do not know when it will start and end. Second, the fixed prediction period in this chapter increases the number of tests and has the advantage of batch-processing the data.

In order to obtain higher forecasting accuracy, it is better to update the in-sample in this model on a daily basis. However, if the in-sample used here updates daily rather than yearly, the calculation amount will increase significantly (252 times the current calculation amount). The calculation time for personal computers is probably over several months, and therefore, this study selects the method involving much less calculation. Nevertheless, in real trading, it is possible to train a new model every day, which can generally improve the accuracy and make the annualized return more stable.

2.4 Theoretical framework and Model building

2.4.1 Support Vector Machines

The SVM classifier is a continuous multiple non-linear classifier. In terms of two groups of samples that can be linearly classified in two-dimensional space, the linear perceptron, which is the original format of SVM, can separate two groups of samples with a straight line.

Extending to high-dimensional space, samples can be separated by a hyper-plane. However, it is common that samples cannot be differentiated by a hyper-plane. With the method of mapping data in low-dimensional space to high-dimensional space, SVMs can make the sample tend to be linearly separable and use a kernel function to achieve this process. Additionally, soft margin SVM classifiers strengthen the generalization performance. SVMs have different hybrids, which can be used to classify various kinds of problems, as well as regressions. This chapter only needs to classify two classes. The SVM model this chapter uses is introduced as follows.

The vector x_i is the input of day i, while y_i is the output. Thus, the training set can be written as $\{(x_1,y_1), (x_2,y_2)\ldots, (x_n,y_n)\}$, where n is the total number of the training sample. The aim of SVM is to find a hyper-plane to classify the two different y ($y = 1$, $y = -1$) in the training sample.

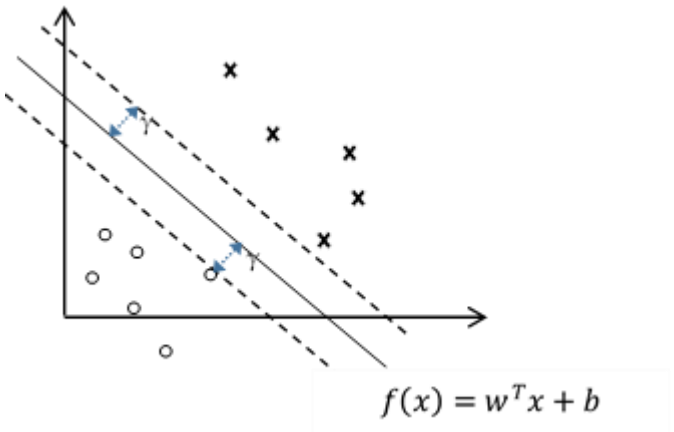
The hyper-plane can be considered as a linear form:

$$f(x) = w^T x + b \tag{2.1}$$

where w is a column vector with the same dimensions as x.

The hyper-plane is not unique. Thus, the optimum classification hyper-plane needs to be defined. A common definition is that the optimum hyper-plane gets the biggest geometry distance Υ between the hyper-plane and two different sets as shown in Figure 2.1.

Figure 2.1: The geometry distance of the training set



Notes: 'x' and 'o' are two different types of points. Υ is the largest geometry distance between $f(x) = w^T x + b$ and the two types of points.

The calculation of the hyper-plane is:

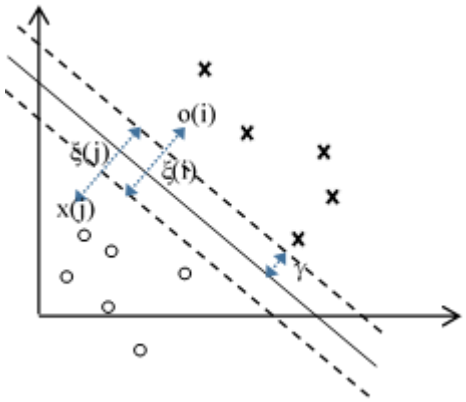
$$Min \quad \|w\|$$
$$Subject\ to \quad y_i(wx_i + b) \geq 1\ for\ all\ i \tag{2.2}$$

when Υ is a constant 1.

Considering the generalization performance, SVM introduces ξ_i which makes misclassification possible. Assuming that we do not allow misclassification to occur, then overfitting in the training set is almost inevitable. The ξ_i for x_i:

$$\xi_i = \max\big(0,1 - y_i(wx_i + b)\big) \tag{2.3}$$

Figure 2.2: Soft variable in classification problems



Notes: $\xi(i)\ and\ \xi(j)$ are distances of misclassifications.

Figure 2.2 gives an example of a soft variable in classification problems. The soft variable is the allowed misclassification point. After the introduction of soft variables, we only need to penalize soft variables in the optimization process. The optimization problem turns to:

$$\min\ \|w\| + c\sum_{i=1}^{I}\xi_i$$
$$subject\ to \quad y_i(wx_i + b) \geq 1 - \zeta_i\ for\ all\ i \tag{2.4}$$

Where c is the punish parameter for SVM.

The SVM above is still linear. The kernel function is introduced to make the SVM non-linear next. The whole problem is seeking the w accurately. The w can be expressed as a linear combination of the training set:

$$w = a_1y_1x_1 + a_2y_2x_2 + \cdots + a_ny_nx_n \tag{2.5}$$

Where a_i is a natural number, which is called Lagrange Multipliers. Most Lagrange

Multipliers are 0, for only minority samples determine the classification hyper-plane. Thus, the function $f(x) = \langle w, x \rangle + b$ can be written as:

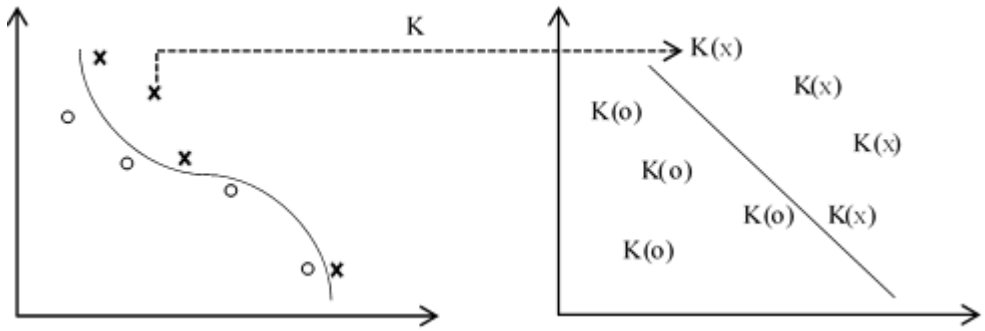
$$f(x) = \sum_{i=1}^{n} a_i y_i \langle x_i, x \rangle + b \tag{2.6}$$

Two linearly inseparable sets in low-dimensional space can tend to be linearly separable in high-dimensional space by function mapping. There are those kinds of function $K(w, x)$, which accept the inputs from low-dimensional space with the output of inner product $\langle w', x' \rangle$ in a high-dimensional space, where w' and x' are mapped from a low dimension. Thus, the training function in a high dimension is:

$$g(x') = \sum_{i=1}^{n} a_i y_i \langle x_i', x' \rangle + b \tag{2.7}$$

In summary, though this problem is linearly inseparable, we can consider it as a linearly separable problem as Figure 2.3. However, the selected kernel function needs to be used to calculate the dot product. In this way, the calculated a and kernel function can be used to calculate the hyper-plane.

Figure 2.3: Mapping a linearly inseparable problem to a linearly separable problem by a kernel function



Notes: K is the kernel function, which transfers the low-dimensional space to high-dimensional space. It makes the sets linearly separable in high-dimensional space.

Here, the kernel function is RBF. The reason is that RBFs require only one parameter σ and provide good forecasting results in similar SVM applications (Yeh et al., 2011; Kao et al., 2013). Thus, the SVM used here has two parameters, c and σ. What remains to be done is to optimize the two parameters and the inputs.

2.4.2 Gravity Search Algorithm

Heuristic algorithms have been widely used in optimization, and one of these is GSA which is proposed by Rashedi et al. in 2009. Its main purpose is to solve the problem of optimizing the curse of dimensionality and discontinuous functions. Compared with the gradient descent algorithm, GSA not only concerns the position of each agent, but also adds the mass of different agents, which enables mass interaction between agents and finds the global solution more efficiently, while the movements of agents in the gradient descent algorithm only considers the position of each agent. Moreover, the gradient descent algorithm can only be used to solve continuous optimization problems, while GSA is capable of optimizing discontinuous problems. Additionally, compared with another similar agent-based search algorithm PSO, the difference between GSA and PSO is that agents of GSA receive the information of their own positions and all of the others, while agents in PSO receive the information of their own best positions and the best global solution position by far. With regard to algorithms, the convergent speed of PSO is faster than GSA, but PSO is more likely to fall into the trap of local solutions.

The structure of GSA is shown in Figure 2.4. First, considering a parameter space with N agents (masses) and n dimensions, the position and speed of the agent i is defined by:

$$X_i = (x_i^1, x_i^2, \dots x_i^n) \tag{2.8}$$

$$V_i = (v_i^1, v_i^2, \dots v_i^n) \tag{2.9}$$

where X_i presents the position and V_i presents the speed of the agent i.

The mass of the agent i is defined by:

$$m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \tag{2.10}$$

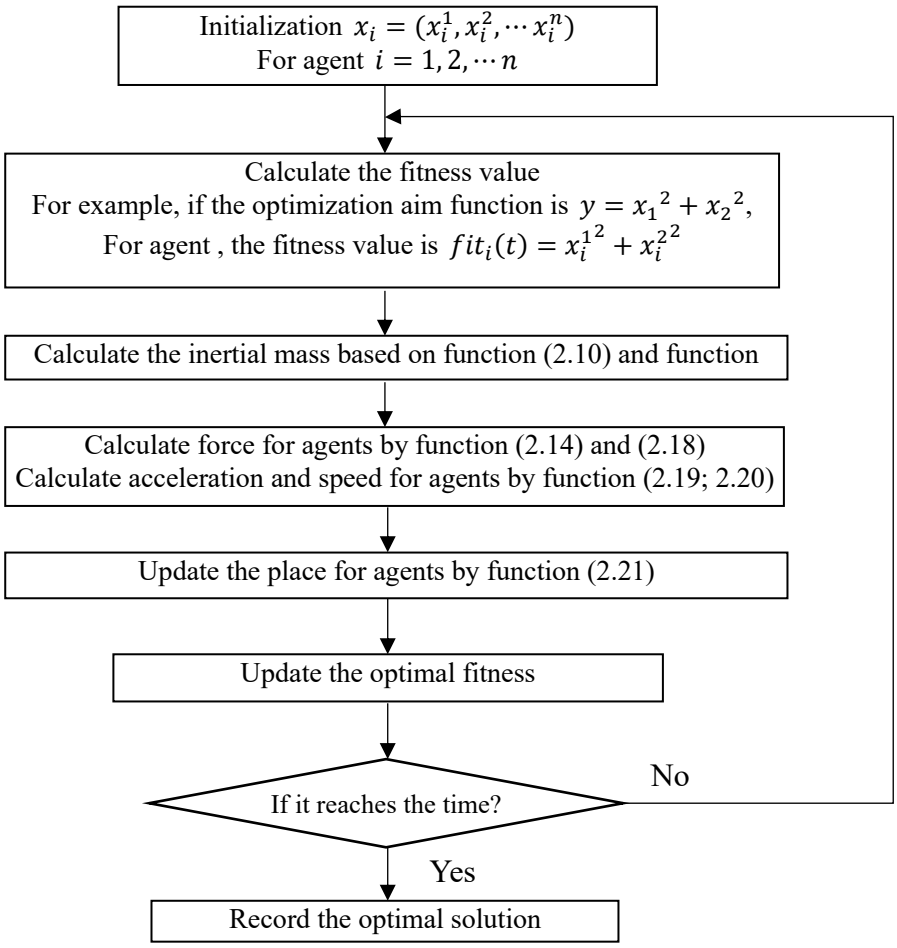
$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \tag{2.11}$$

where $fit_i(t)$ and $M_i(t)$ respectively present the fitness value and mass of the agent i at time t. $best(t)$ and $worst(t)$ are respectively the best fitness value and the worst fitness value at time t, which are defined as follows.

$$best(t) = \min_{j \in \{1,\dots,n\}} fit_j(t) \tag{2.12}$$

$$worst(t) = \max_{j \in \{1,\dots,N\}} fit_j(t) \tag{2.13}$$

Figure 2.4: The structure of GSA



Notes: This figure displays the workflow of GSA. GSA uses the gravity determined by the agents' current position to decide the position and gravity of the next iteration, and finds the optimal solution during the movement of the agents.

This algorithm originates in the simulation of universal gravitation. However, it is not limited to the precise universal gravitation function in physics. At dimension d, the force that acts on mass j from i is defined as follows:

$$F_{ij}^d(t) = G(t)\frac{M_{pi}(t) \times M_{aj}(t)}{R_{ij}(t) + \varepsilon}\left(X_j^d(t) - X_i^d(t)\right) \tag{2.14}$$

where G_t presents the gravitational constant at time t. In fact, the value of G_t can be a general constant instead of a gravitational constant. G is a function of G_0 (initial value) and t.

$$G_t = G(G_0, t) \tag{2.15}$$

Here, G_t is as follows:

$$G_t = G_0 * e^{-\frac{20*t}{T}} \tag{2.16}$$

T is maximum t.

$R_{ij}(t)$ is the Euclidian distance between two agents and ε is a small constant that prevents the denominator from becoming zero.

$$R_{ij}(t) = \left\| X_i(t), X_j(t) \right\|_2 \tag{2.17}$$

The total force that acts on agent i in the dimension d is as follows:

$$F_i^d(t) = \sum_{j=\text{kbest}.j\neq i}^{N} rand_j F_{ij}^d(t) \tag{2.18}$$

where $kbest$ is the set of first K agents with the best fitness value. The element number K of the $kbest$ decreases linearly with time (iterations), which begins with the initial value k_0=N and linearly decreases with time to k_T =1. All agents apply the force initially and decrease gradually, and at the end, only one agent applies force to the other agents. In this way, getting trapped in a local optimum can be effectively avoided.

Then the acceleration is calculated. According to Newton's second law of motion, in the dimension d, the acceleration of the agent i at time t is defined as follows:

$$a_i^d(t) = \frac{F_i^d(t)}{M_{ii}(t)} \tag{2.19}$$

Furthermore, the position and velocity need to be updated as follows:

$$v_i^d(t+1) = rand_i \times v_i^d(t) + a_i^d(t) \tag{2.20}$$

$$x_i^d(t+1) = x_j^d(t) + v_i^d(t+1) \tag{2.21}$$

where $rand_i$ is a uniform random variable in the interval [0,1]. This random number is used to give a randomized characteristic to the search.

Then this study compares the optimal agent for this iteration and the historically optimal agent, selects the agent with better fitness between them as the new historically optimal agent and records its location and fitness. At this point, an iteration is completed, the new locations of all agents are used to iterate until the preset iteration time is reached.

2.4.3 Proposed Methodology: BGSA with SVM

This chapter applies the BGSA to optimize the parameters and inputs of SVM. BGSA is

modified from GSA. In GSA, if the velocity of an agent is large, it means the current position of the agent is not proper and a great movement is required to make it leave its position. In contrast, a small absolute value of the velocity indicates that the current position of the agent is close to the optimum position and the movement of the agent should be small to reach the optimum position. Furthermore, if the agent finds the optimal solution, its velocity should be zero.

Based on the concepts of the GSA, BGSA's concepts should consider that: "A large absolute value of velocity must provide a high probability of changing the position of the mass respect to its previous position (from 1 to 0 or vice versa). A small absolute value of the velocity must provide a small probability of changing the position. In other words, a zero value of the velocity represents that the mass position is good and must not be changed" (Rashedi et al., 2010, p.732).

In BGSA, the rule of movement of GSA should be changed. $S(v_i^d)$ is defined to transfer v_i^d to a probability function.

$$S\left(v_i^d(t)\right) = \left|\tanh\left(v_i^d(t)\right)\right| \tag{2.22}$$

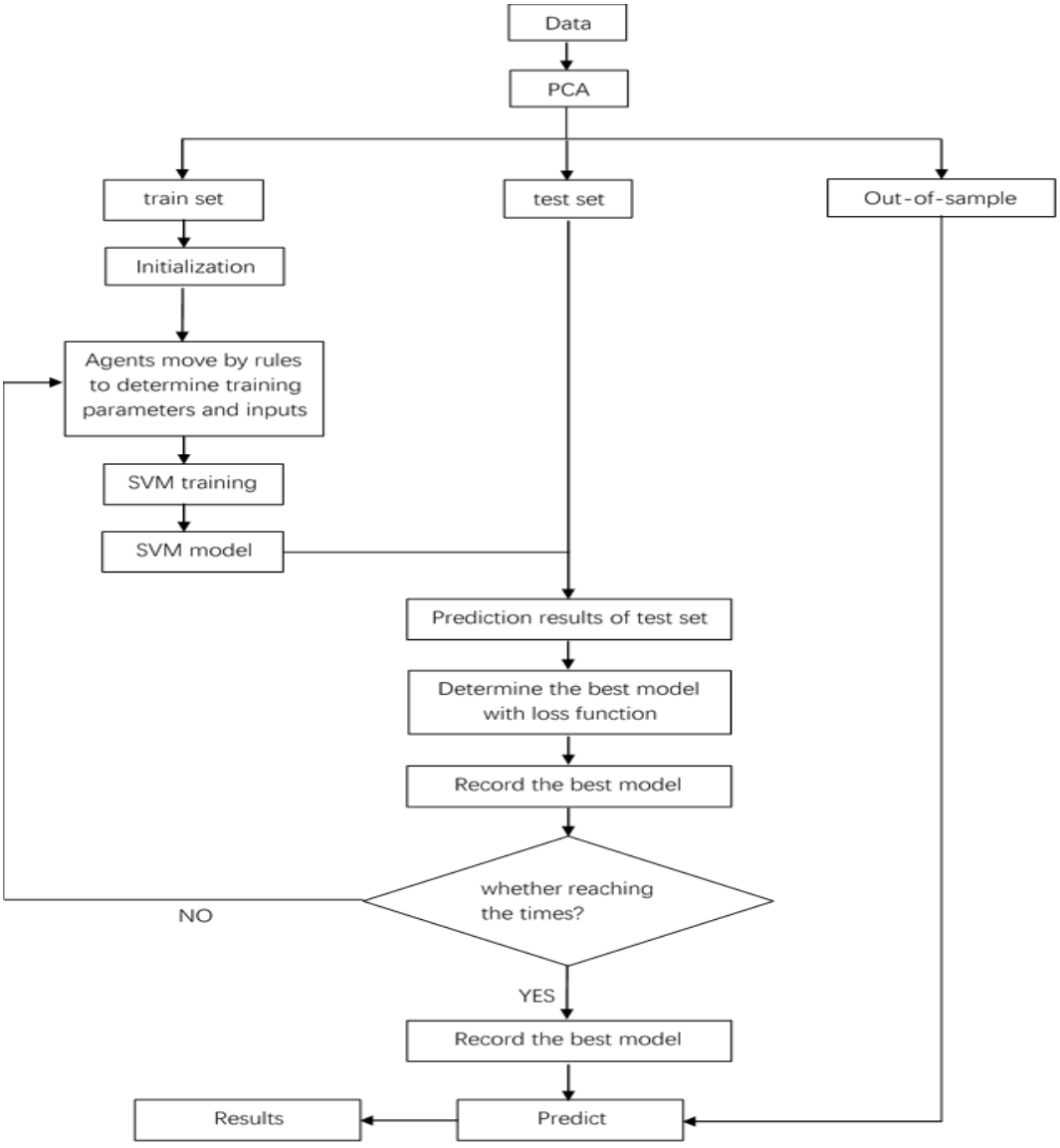
Then, the agent movement function is modified as:

$$if\ rand < S(v_i^d(t+1) \quad then\ x_i^d(t+1) = complement(x_i^d(t))$$
$$else\ x_i^d(t+1) = x_i^d(t) \tag{2.23}$$

Based on Rashedi's paper (2010), the velocity limit here is lower than 6 ($v_i^d < 6$).

The structure of the whole model for forecasting is shown in Figure 2.5. This study pre-processes initial data by PCA in the model. Then data is divided into three sets: the training set, the test set and the out-of-sample set. The training set is used to train some SVM classifiers with random inputs and parameters. After that, the test set is used to test those SVM classifiers. According to the performances of those classifiers and the iteration rules of BGSA, the inputs and parameters of those SVM classifiers are updated. The iteration continues until it reaches a certain maximum time. Finally, the SVM classifier with the best performance in the training set and test set during the iteration is utilized to predict the out-of-sample set and obtain the final results.

Figure 2.5: Model structure of BGSA-SVM



Notes: This figure describes the workflow of BGSA-SVM. The study divides the data into three sets: the training set, the test set and the out-of-sample set. The training set is employed to train the SVM model, the test set is used to modify inputs and parameters, and the out-of-sample set is used to obtain results.

The contribution of this supervised machine learning model is that it provides an effective way to train high-dimensional financial data in a limited training set. In addition, this is the first time that BGSA-SVM has been used in the field of financial forecasting.

2.4.4 Benchmark models

In this chapter, BGSA-SVM is compared with several traditional models, which include SVM, the Random Walk, the Buy-and-Hold, and the first and the second best inputs. Using SVM as a benchmark can show whether the conjunction of BGSA is effective. This study defines the Buy-and-Hold as the daily returns of stock indices over a specific period. The reason for establishing the Buy-and-Hold and the RW is to prove the overall validity of the prediction model. When the expected return of BGSA-SVM exceeds the Buy-and-Hold without increasing risks, we can state that it is valid to develop the trading strategy according to the prediction model.

This chapter also uses inputs predictors with the top two performances in the in-sample set as benchmarks to test whether BGSA-SVM always outperforms the inputs used by itself. $best_1$ is the input with the best trading performance in the training sample and $best_2$ is the second best input of all inputs. These two best-performing inputs are different for every test. $best_1$ and $best_2$ are shown in Table 2.2 and Table 2.3.

Table 2.2: The selected inputs for $best_1$ in every test

	FTSE	SPX	NKY	CAC	DAX
F1	ARMA(8,10)	K(1)	ARMA(6,10)	ARMA(1,3)	ARMA(10,6)
F2	ARMA(10,7)	Return(17)	ARMA(6,10)	ARMA(6,8)	ARMA(3,6)
F3	ARMA(10,7)	ARMA(9,8)	ARMA(6,10)	ARMA(6,8)	ARMA(10,6)
F4	ARMA(3,3)	ARMA(9,7)	ARMA(6,10)	ARMA(6,8)	ARMA(10,6)
F5	ARMA(3,3)	ARMA(9,7)	Return(5)	ARMA(6,8)	ARMA(10,6)
F6	ARMA(9,9)	ARMA(9,7)	Return(5)	ARMA(6,8)	ARMA(10,6)
F7	ARMA(9,9)	ARMA(9,7)	Return(5)	ARMA(9,7)	AR(6)
F8	ARMA(9,9)	ARMA(9,7)	Return(8)	ARMA(10,8)	ARMA(5,10)
F9	ARMA(4,3)	ARMA(4,3)	Return(8)	ARMA(9,9)	ARMA(5,10)
F10		ARMA(8,10)	Return(8)	ARMA(9,9)	ARMA(10,9)
F11		ARMA(3,2)	Return(8)	ARMA(9,9)	ARMA(9,10)

Note: Details of selected inputs are in Appendix A.1.1 Inputs pool.

Table 2.3: The selected inputs for $best_2$ in every test

	FTSE	SPX	NKY	CAC	DAX
F1	ARMA(9,9)	K(2)	ARMA(9,8)	ARMA(6,8)	ARMA(4,10)
F2	ARMA(8,8)	K(1)	ARMA(8,6)	ARMA(7,9)	ARMA(10,6)
F3	ARMA(8,7)	M(1)	ARMA(8,6)	ARMA(7,7)	ARMA(5,8)
F4	ARMA(10,7)	ARMA(9,8)	ARMA(7,7)	ARMA(7,5)	AR(6)
F5	ARMA(9,9)	ARMA(9,8)	ARMA(7,7)	ARMA(5,8)	AR(6)
F6	ARMA(3,3)	ARMA(9,8)	Return(16)	ARMA(8,7)	ARMA(5,10)
F7	ARMA(1,6)	ARMA(9,8)	Return(16)	ARMA(9,9)	ARMA(10,6)
F8	ARMA(10,3)	ARMA(7,9)	Return(16)	ARMA(10,6)	ARMA(10,6)
F9	ARMA(9,10)	ARMA(6,7)	Return(16)	ARMA(10,8)	ARMA(10,9)
F10		ARMA(9,9)	ARMA(7,8)	ARMA(10,8)	ARMA(9,10)
F11		ARMA(9,9)	ARMA(7,8)	ARMA(10,6)	ARMA(10,9)

Note: Details of selected inputs are in Appendix A.1.1 Inputs pool.

2.5 Statistical evaluation

This section provides the out-of-sample performances of the BGSA-SVM model and the benchmark models. The prediction accuracy of the BGSA-SVM model and the comparison with the benchmark models are described in Section 2.5.1. Section 2.5.2 evaluates the BGSA-SVM model and the benchmark models with the Diebold-Mariano and Giacomini-White tests.

2.5.1 Statistical accuracy

The statistical accuracy of the prediction of BGSA-SVM is presented in Table 2.4. A direct way to show the effectiveness of prediction is to examine whether the out-of-sample accuracy is higher than 50%. Table 2.4 shows that the probability of the out-of-sample accuracy being lower than 50% is small (marked in bold in Table 2.4), occurring 7 times in all 53 forecasting times. The probability that the RW prediction accuracy is less than 50% is close to 50%, which means that the expectation in 53 experiments is 26.5 times. This also proves that the prediction accuracy of the BGSA-SVM model is significantly higher than that of the RW. The detailed performances of the benchmark models are presented in Appendix A.2.

Table 2.4: Accuracy of BGSA-SVM

	Sample	FTSE100	S&P500	NKY	CAC40	DAX
F1	Training sample	0.7173	0.7153	0.6726	0.6935	0.6190
	Test sample	0.6508	0.6171	0.6151	0.6329	0.6190
	Out-of-sample	0.5278	0.5516	0.5595	**0.4881**	**0.4643**
F2	Training sample	0.7331	0.6052	0.6538	0.6448	0.6111
	Test sample	0.6329	0.6190	0.6190	0.6052	0.6052
	Out-of-sample	0.5317	0.5516	0.5476	0.5397	0.5278
F3	Training sample	0.6171	0.6409	0.5952	0.6587	0.7063
	Test sample	0.6171	0.6528	0.6111	0.6052	0.6091
	Out-of-sample	0.5357	0.5437	0.5357	0.5040	**0.4881**
F4	Training sample	0.6260	0.7123	0.6260	0.6607	0.7252
	Test sample	0.6171	0.6429	0.6270	0.6270	0.6310
	Out-of-sample	0.5357	**0.4881**	0.5079	0.5079	0.5992
F5	Training sample	0.6339	0.5992	0.6270	0.6300	0.6875
	Test sample	0.6230	0.6270	0.5933	0.6091	0.6389
	Out-of-sample	0.5198	0.5238	0.5000	0.5476	0.5040
F6	Training sample	0.6290	0.5883	0.6835	0.6012	0.7004
	Test sample	0.6131	0.5913	0.6091	0.5913	0.6190
	Out-of-sample	**0.4762**	0.5278	0.5379	**0.4762**	**0.4921**
F7	Training sample	0.6696	0.6161	0.6528	0.7897	0.6300
	Test sample	0.6012	0.6171	0.6111	0.5933	0.6349
	Out-of-sample	0.5119	0.5119	0.5317	0.5714	0.5714
F8	Training sample	0.6796	0.6409	0.6706	0.7083	0.7500
	Test sample	0.6111	0.6111	0.6290	0.6409	0.6290
	Out-of-sample	0.5119	0.5278	0.5238	0.5317	0.5357
F9	Training sample	0.6012	0.6419	0.6329	0.6111	0.6389
	Test sample	0.6389	0.6369	0.6409	0.6270	0.6389
	Out-of-sample	0.5159	0.5595	0.5317	0.5833	0.5198
F10	Training sample		0.6280	0.6875	0.6052	0.6667
	Test sample		0.6429	0.6369	0.6171	0.6131
	Out-of-sample		0.5159	0.5357	0.5278	0.5079
F11	Training sample		0.7361	0.8095	0.6796	0.6806
	Test sample		0.6528	0.6190	0.5873	0.6091
	Out-of-sample		0.5437	0.5079	0.5159	0.5238
Average	Training sample	0.6563	0.6477	0.6647	0.6621	0.6742
	Test sample	0.6228	0.6283	0.6192	0.6124	0.6225
	Out-of-sample	0.5282	0.5314	0.5290	0.5267	0.5281

Note: The accuracy of out-of-sample lower than 50% is marked in bold.

In addition, the average out-of-sample accuracy for all indices is higher than 50%. The average out-of-sample accuracy for FTSE, S&P500, NKY, CAC and DAX increases by 2.6%-3.1% compared with the expected accuracy (50%) of random classification. In terms of the average accuracy of forecasting the five stock indices, the performance of BGSA-SVM is stable.

The sensitivity of BGSA-SVM in extreme situations is not very significant. As shown in Table 2.4, around 2008 (F6-F7), FTSE, CAC and DAX each have one failed forecast, but

S&P500 and NKY are normal. During the 1998 Asian financial crisis, there were no failed predictions with an accuracy lower than 50% in NKY.

Table 2.4 also shows the performances of the training sample and the test sample. The average accuracy of the training sample is between 64% to 67.5%, whereas that of the test sample is 61% to 63%. The accuracy difference (around 10 percentage points) between the in-sample set and the out-of-sample set shows that there is still a slight over-fitting. BGSA-SVM with more input information is able to increase the accuracy of the in-sample set and the out-of-sample set and reduce the over-fitting.

In summary, BGSA-SVM increases the out-of-sample accuracy by two to three percentage points compared with the expectation of random classification. BGSA-SVM has the potential to increase accuracy further with more fundamental information.

2.5.2 Diebold-Mariano test and Giacomini-White test

This section uses the Diebold-Mariano test to compare the accuracy of BGSA-SVM with each benchmark model. A rejection of the null hypothesis suggests that the first forecast (the BGSA-SVM) is more accurate.

Table 2.5 shows the results of the DM test for BGSA-SVM with all benchmark models. In the DM test, the probabilities of BGSA-SVM being significantly better than any benchmark models are 20.4%, 26.8%, and 30.6%, respectively, at the 1%, 5% and 10% significance levels. According to the results, the DM test cannot directly prove that BGSA-SVM is significantly better than other benchmark models. This may be because BGSA-SVM has no obvious advantage over the benchmark models in some forecasting tests. As mentioned above, BGSA-SVM only improves the average accuracy by two to three percentage points compared with the RW. In addition, it can be observed in Table 2.5 that in some tests (e.g. F1, CAC; F5, FTSE), BGSA-SVM shows the comprehensively superior performance over all benchmark models. This suggests that BGSA-SVM performs very well in certain trading years.

Table 2.5: DM test for BGSA-SVM with benchmark models

	Index	RW	SVM	$best_1$	$best_2$	Buy-and-Hold
F1	FTSE	0.6959	0.8141	0.6901	0.3679	0.0280**
	S&P	0.0005***	0.9998	0.0005***	0.9998	0.0005***
	NKY	0.3844	1.0000	0.1339	0.2466	0.0546*
	CAC	0.0000***	0.0000***	0.0000***	0.0000***	0.0000***
	DAX	0.0000***	0.9795	0.0000***	0.0000***	0.0000***
F2	FTSE	0.6395	0.0707*	0.7370	0.9018	0.7488
	S&P	0.0040***	1.0000	0.0001***	0.0001***	0.0001***
	NKY	0.9875	1.0000	0.9994	0.9987	0.8932
	CAC	0.9121	1.0000	0.8472	0.8762	0.1567
	DAX	0.1721	0.9998	0.3056	0.3769	0.1174
F3	FTSE	0.9999	1.0000	0.9998	1.0000	1.0000
	S&P	0.0648*	0.6447	0.6645	1.0000	0.0468**
	NKY	0.9936	1.0000	0.9039	0.9084	0.7853
	CAC	0.9637	1.0000	0.1688	0.2900	0.3473
	DAX	0.4609	0.1258	0.8725	0.3082	0.6433
F4	FTSE	0.2878	0.6728	0.4767	0.0510*	0.1774
	S&P	0.9356	0.3419	0.7022	0.5575	0.9849
	NKY	0.8733	0.0000***	0.9497	0.9402	0.9998
	CAC	0.9984	1.0000	0.9977	0.9970	0.9998
	DAX	0.0004***	0.2878	0.0432**	0.0164**	0.0026***
F5	FTSE	0.0000***	0.0000***	0.0000***	0.0000***	0.0000***
	S&P	0.9997	1.0000	0.9999	0.9997	1.0000
	NKY	0.7059	0.2279	0.4523	0.1868	0.0520*
	CAC	0.0024***	1.0000	0.0481**	0.0133**	0.4028
	DAX	0.6051	0.8848	0.8280	0.4497	0.7469
F6	FTSE	0.023**	0.0001***	0.0022***	0.0241**	0.0000***
	S&P	0.0023***	0.0002***	0.0056***	0.013**	0.0036***
	NKY	0.0469**	0.4490	0.1864	0.0773*	0.0242**
	CAC	0.9995	0.9847	0.9971	0.9969	0.9992
	DAX	0.0000***	0.0000***	0.0000***	0.0000***	0.0000***
F7	FTSE	0.2010	0.8058	0.0964*	0.3620	0.1461
	S&P	0.7465	0.0000***	0.8071	0.8230	0.7499
	NKY	0.2512	0.0064***	0.7459	0.0512*	0.4087
	CAC	0.9972	0.7477	0.6303	0.8324	0.5096
	DAX	0.4401	0.0916*	0.3907	0.0003***	0.0004***
F8	FTSE	0.8401	0.4014	0.9864	0.9107	0.4680
	S&P	0.1359	0.1804	0.0442**	0.0106**	0.0046***
	NKY	0.6053	0.1427	0.0843*	0.2403	0.0620*
	CAC	0.9992	0.0000***	0.9968	0.9796	0.9833
	DAX	0.6466	0.9998	0.8235	0.9635	0.6367
F9	FTSE	0.9978	0.8852	0.9994	0.9971	0.9911
	S&P	0.0002***	1.0000	0.0014***	0.0078***	0.0005***
	NKY	0.1849	0.4316	0.0615*	0.0438**	0.0893*
	CAC	0.0005***	1.0000	0.6185	0.8333	0.0828*
	DAX	0.5691	0.8664	0.5153	0.5943	0.9321
F10	S&P	0.0108**	0.8726	0.2998	0.4802	0.0112**
	NKY	0.6154	0.4098	0.7393	0.0719*	0.6144
	CAC	0.9677	0.4300	0.6662	0.9007	0.6770
	DAX	0.4250	0.9976	0.5776	0.9183	0.3618
F11	S&P	0.0011***	0.1242	0.0000***	0.0000***	0.7683
	NKY	0.8693	0.4445	0.9305	0.3785	0.8597
	CAC	0.7112	1.0000	0.0417**	0.5365	0.3385
	DAX	0.0000***	1.0000	0.0000***	0.0000***	0.0000***

Note: ***, ** and * denote that the DM null hypothesis is rejected at the 1%, 5% and 10% significance levels respectively. $best_1$ and $best_2$ refer to the best individual predictors in terms of statistical and trading performances respectively in the in-sample period.

Table 2.6: GW test for BGSA-SVM with benchmark models

	Index	Best model	RW	SVM	$best_1$	$best_2$	Buy-and-Hold
F1	FTSE	BGSA-SVM	0.5018-	0.9086-	0.3943-	0.2101-	0.0102-**
	S&P	BGSA-SVM	0.0000-***	0.0000-***	0.0000-***	0.0000-***	0.0000-***
	NKY	BGSA-SVM	0.2783-	0.0000-***	0.1340-	0.5157-	0.0262-**
	CAC	$best_2$	0.0000-***	0.0000-***	0.0000+***	0.0000+***	0.0000-***
	DAX	RW	0.0000+***	0.0034-***	0.0000+***	0.0000+***	0.0000-***
F2	FTSE	RW	0.6585+	0.0189-**	0.5562+	0.3025+	0.8910-
	S&P	BGSA-SVM	0.0012-***	0.0000-***	0.0000-***	0.0000-***	0.0000-***
	NKY	BGSA-SVM	0.0007-***	0.0000-***	0.0000-***	0.0000-***	0.0065-***
	CAC	BGSA-SVM	0.0033-***	0.0000-***	0.0884-*	0.0188-**	0.7892-
	DAX	$best_2$	0.1614-	0.0000-***	0.5083+	0.5933+	0.0326-**
F3	FTSE	BGSA-SVM	0.0000-***	0.0000-***	0.0000-***	0.0000-***	0.0000-***
	S&P	$best_1$	0.4383-	0.1817-	0.1685+	0.0000-***	0.1881-
	NKY	RW	0.0014+***	0.0000-***	0.0248-**	0.0465-**	0.0396-**
	CAC	$best_1$	0.0733-*	0.0000-***	0.2535+	0.5456+	0.5534-
	DAX	NR	0.7999+	0.0223-**	0.3673+	0.4061+	0.2924+
F4	FTSE	NR	0.6341-	0.0572-*	0.9744-	0.0076-***	0.9500+
	S&P	NR	0.0242+**	0.7278+	0.2057+	0.1944+	0.0036+***
	NKY	NR	0.0323-**	0.0000-***	0.0306-**	0.0129-**	0.0000+***
	CAC	SVM&NR	0.0000+***	0.0000+***	0.0000+***	0.0005+***	0.0000+***
	DAX	BGSA-SVM	0.0000-***	0.3729-	0.0020-***	0.0001-***	0.0004-***
F5	FTSE	NR	0.0000-***	0.0000-***	0.0000-***	0.0000-***	0.0000+***
	S&P	SVM	0.0000-***	0.0000+***	0.0000-***	0.0000-***	0.0000-***
	NKY	SVM	0.7227-	0.3934+	0.7988-	0.0830-*	0.2190-
	CAC	NR	0.0002-***	0.0000-***	0.0014+***	0.0000-***	0.0994+*
	DAX	$best_1$	0.9975-	0.6939-	0.7723+	0.3101+	0.3722+
F6	FTSE	$best_2$	0.0000+***	0.0000+***	0.0000+***	0.0000+***	0.0000-***
	S&P	$best_2$	0.0005-***	0.0000+***	0.0005+***	0.0014+***	0.0033-***
	NKY	BGSA-SVM	0.0272-**	0.9635-	0.3501-	0.0774-*	0.0704-*
	CAC	SVM	0.0000-***	0.0003+***	0.0001+***	0.0003+***	0.0000+***
	DAX	NR	0.0000-***	0.0000-***	0.0000-***	0.0000-***	0.0000+***
F7	FTSE	RW	0.7363+	0.1032-	0.4025-	0.1695+	0.3840-
	S&P	$best_1$	0.9596+	0.0000+***	0.6456+	0.6179+	0.7435+
	NKY	$best_2$	0.9519+	0.0086+***	0.3698-	0.1963+	0.6206-
	CAC	RW	0.0022+***	0.5809-	0.0346-**	0.0035-***	0.1698-
	DAX	BGSA-SVM	0.0152-**	0.0000-***	0.0001-***	0.0000-***	0.0000-***
F8	FTSE	BGSA-SVM	0.0778-*	0.9851-	0.0029-***	0.0431-**	0.0781-*
	S&P	$best_2$	0.0317-**	0.0742-*	0.0099-***	0.0020+***	0.0033-***
	NKY	$best_1$	0.4511-	0.2309-	0.4209+	0.6256+	0.3416-
	CAC	BGSA-SVM	0.0018-***	0.0000-***	0.0076-***	0.1042-	0.0302-**
	DAX	BGSA-SVM	0.1798-	0.0000-***	0.2476-	0.0136-**	0.5491-
F9	FTSE	BGSA-SVM	0.0000-***	0.0091-***	0.0000-***	0.0001-***	0.0000-***
	S&P	RW	0.0000+***	0.0000-***	0.0000+***	0.0002+***	0.0000-***
	NKY	$best_1$	0.3973-	0.7979+	0.1155+	0.1091+	0.1777+
	CAC	BGSA-SVM	0.0000-***	0.0000-***	0.2045-	0.4169-	0.0334-**
	DAX	NR	0.1732-	0.0494-**	0.6076-	0.7430-	0.0179+**
F10							
	S&P	SVM	0.0169-**	0.0747+*	0.2333-	0.7442-	0.1216-
	NKY	NR	0.8586+	0.8876+	0.5028+	0.0202-**	0.6886+
	CAC	SVM	0.0391+**	0.7829+	0.3433+	0.0804+*	0.4273-
	DAX	NR	0.2739-	0.0000+***	0.9449+	0.1166+	0.7955+
F11							
	S&P	NR	0.0000-***	0.0771-*	0.0000-***	0.0000-***	0.0015+***
	NKY	NR	0.2612+	0.8859+	0.2428+	0.0000-***	0.0950+*
	CAC	BGSA-SVM	0.1656-	0.0000-***	0.0405-**	0.9924-	0.5249-
	DAX	BGSA-SVM	0.0000-***	0.0000-***	0.0000-***	0.0000-***	0.0000-***

Note: ***, ** and * denote that the GW test is at the 1%, 5% and 10% significance levels respectively. $best_1$ and $best_2$ refer to the best individual predictors in terms of statistical and trading performances respectively in the in-sample period. The best model list shows the best model in one test. Sign "-" means BGSA-SVM performs better than the benchmark model, "+" means benchmark model beats BGSA-SVM.

The unconditional Giacomini-White test is used for the out-of-sample predictive ability testing and forecast selection. According to Giacomini and White (2006), the null hypothesis of the GW test is equivalent in forecasting accuracy between two forecasting models. The sign of the test statistic specifies the superior model according to its forecasting performance. A positive realization of the GW test statistic indicates that the second model is more accurate than the first one, whereas a negative realization indicates the opposite. The GW test is calculated based on the Mean Squared Error loss function.

Table 2.6 shows the results of the GW test for BGSA-SVM with all benchmark models, and it also indicates the model with the best performance in every test. BGSA-SVM can improve forecasting accuracy, whereas it cannot guarantee to beat the benchmark models every time. In 53 predictions, BGSA-SVM is the best of the six models 17 times. If BGSA-SVM is invalid, then the expectation that it would be the best model should be less than 8.83 (53/6) times. However, in this experiment, it appears up to 17 times, which indicates it is valid. If we assume that the return distributions of all benchmark models and the BGSA-SVM model are normal distributions, BGSA-SVM is the best of the six models under the 99% confidence level according to the chi-square test (Appendix A.3.2).

2.6 Trading performance

The trading method is as follows: when the prediction result is positive, buy a certain amount when the stock market opens and sell the same amount at the close; if the prediction result is negative, then sell a certain amount when the stock market opens and buy the same amount at the close. According to this trading method, the out-of-sample annualized returns of BGSA-SVM and the benchmark models are given in Table 2.7.

Table 2.7 shows that the average annualized returns of BGSA-SVM are higher than five benchmark models (including the RW, SVM, best_1, best_2 and Buy-and-Hold) in all five stock indices. Moreover, the average returns of BGSA-SVM are all positive, at 11.12%, 11.44%, 10.98%, 12.96% and 10.86% respectively of FTSE, S&P, NKY, CAC and DAX. The average annualized returns of BGSA-SVM are stable for different stock indices. This shows that BGSA-SVM forecasting is not affected by the performance of the stock index (Buy-and-Hold).

Table 2.7: Annualized returns of BGSA-SVM and benchmark models in out-of-sample (%)

	Index	BGSA-SVM	RW	SVM	$best_1$	$best_2$	Buy-and-Hold
F1	FTSE	**15.98**	-26.74	13.05	15.68	14.55	-13.12
	S&P	**35.41**	-15.08	16.67	16.67	16.67	16.67
	NKY	**32.64**	-24.36	-13.81	27.87	28.91	-47.78
	CAC	-8.00	-19.21	-19.41	9.28	**33.48**	-25.51
	DAX	-14.67	**28.82**	-27.35	19.73	17.18	-27.35
F2	FTSE	19.07	**36.39**	15.61	33.80	19.69	-15.86
	S&P	**30.09**	-21.36	-19.26	-42.35	-19.26	-19.26
	NKY	**15.08**	11.50	2.29	-5.81	-21.24	-34.24
	CAC	**34.20**	2.58	-57.82	11.52	6.91	-57.82
	DAX	-3.30	-11.15	-31.68	-1.33	**3.81**	-49.74
F3	FTSE	**24.42**	0.86	-9.66	17.89	5.92	20.96
	S&P	9.92	-7.69	-30.44	**22.28**	4.19	-42.01
	NKY	3.25	**7.80**	-1.18	-27.71	-49.47	-1.18
	CAC	10.35	-15.76	-10.2	**19.06**	13.63	-10.25
	DAX	2.06	15.64	-19.89	11.56	8.80	**25.50**
F4	FTSE	10.49	-0.74	6.81	-9.94	-10.57	**11.07**
	S&P	-6.28	6.55	1.81	18.30	23.59	**29.94**
	NKY	10.92	10.68	-27.81	-1.30	-18.16	**27.81**
	CAC	-8.39	7.35	**16.22**	12.80	9.52	**16.22**
	DAX	**24.10**	-18.44	19.17	17.88	-3.18	7.23
F5	FTSE	10.72	9.84	-12.64	-10.05	3.31	**14.03**
	S&P	15.62	3.17	**25.81**	4.47	-2.84	10.99
	NKY	17.88	17.87	**20.67**	4.86	-13.94	-12.92
	CAC	16.39	5.57	6.60	20.07	9.20	**20.19**
	DAX	-11.99	-3.12	-4.62	**21.76**	9.70	20.65
F6	FTSE	-16.18	16.85	10.59	12.51	**22.43**	-23.64
	S&P	5.19	4.97	8.99	15.41	**15.77**	3.03
	NKY	**25.61**	9.60	24.81	3.03	14.42	-16.31
	CAC	-0.82	-8.81	**21.26**	5.29	0.59	13.75
	DAX	12.43	-10.20	10.79	-9.79	-10.99	**15.05**
F7	FTSE	22.45	**25.50**	19.34	16.10	24.32	-8.97
	S&P	-13.44	-2.97	-8.92	**23.36**	15.11	8.92
	NKY	-4.17	9.25	22.94	-9.06	**49.17**	-48.35
	CAC	24.23	**51.01**	14.38	9.78	3.61	-17.63
	DAX	**31.00**	2.90	-22.37	-5.38	25.25	0.15
F8	FTSE	**3.96**	-11.72	0.89	-2.37	-22.54	-8.31
	S&P	14.71	-5.23	-3.36	14.50	**21.68**	-5.59
	NKY	10.84	-9.35	6.55	**27.94**	11.80	-9.93
	CAC	**35.46**	-5.22	23.78	18.55	14.72	-33.14
	DAX	**41.76**	-1.86	7.25	6.89	3.27	-46.73
F9	FTSE	**9.13**	3.25	0.44	-11.03	-19.09	-2.08
	S&P	16.06	**33.50**	-7.39	20.87	23.52	-7.39
	NKY	-0.70	-31.03	4.01	**16.20**	-0.45	0.41
	CAC	**41.61**	28.41	10.58	14.64	17.32	6.79
	DAX	29.33	17.03	20.21	3.99	-6.41	**34.84**
F10	S&P	8.47	-0.61	**16.12**	6.25	3.60	7.35
	NKY	9.50	24.13	16.74	20.30	-3.35	**25.50**
	CAC	-16.87	23.17	**37.64**	-2.59	-2.64	-22.09
	DAX	1.68	-12.60	6.49	-1.09	2.27	**7.19**
F11	S&P	10.05	-23.98	5.38	-6.84	1.05	**22.04**
	NKY	-0.1228	35.52	22.39	23.79	-4.94	**35.90**
	CAC	**14.42**	5.24	13.08	-4.90	-3.34	8.26
	DAX	**7.02**	-32.45	-15.17	-2.19	-1.30	-15.17
Average	FTSE	**11.12**	7.06	7.16	6.95	4.22	-2.88
	S&P	**11.44**	-2.61	1.04	8.45	9.37	2.24
	NKY	**10.98**	5.60	8.87	7.28	-0.66	-7.37
	CAC	**12.96**	6.76	5.12	10.32	9.36	-9.20
	DAX	**10.86**	-2.31	-3.38	5.64	4.40	-2.58

Note: The highest annualized returns in each forecasting exercise per index are marked in bold. The units in the table are %.

BGSA-SVM has 13 negative returns among total 53 predictions, which is significantly lower than the number in the benchmark models (RW 22 times, SVM 19 times, $best_1$ 17 times, $best_2$ 18 times and Buy-and-Hold 27 times over all 53 times). Table 2.7 also indicates that the average performances and stability of both $best_1$ and $best_2$ are better than that of SVM.

This is because the input used in this study is large, which makes the SVM very prone to overfitting.

This study adopts the 0.5% annual transaction cost. In practice, the transaction cost depends on trading times and amount. According to the rules applied here (given that there are 252 trading days every year), we need to buy 252 times and sell 252 times annually. However, in the real situation, trading is not that frequent, as when the prediction of stock tomorrow is the same as today's stock sign, there is no need to sell (or buy) at the close, which can reduce many transaction costs. In order to have a straightforward calculation, this study still uses the transaction cost of 0.5% every year.

Table 2.8: Annualized returns after deducting 0.5% transaction cost of BGSA-SVM Model

	FTSE100	S&P500	NKY	CAC40	DAX
F1	15.48	34.91	32.14	-8.50	-15.17
	(1.03)	(1.30)	(1.51)	(1.47)	(1.38)
F2	18.57	29.59	14.58	33.70	-3.80
	(1.44)	(1.33)	(1.42)	(1.61)	(2.30)
F3	23.92	9.42	2.75	9.85	1.56
	(1.15)	(1.55)	(1.27)	(1.89)	(2.08)
F4	9.99	-6.78	10.42	-8.89	23.60
	(0.57)	(1.22)	(0.83)	(0.88)	(0.98)
F5	10.22	15.12	17.38	15.89	-12.49
	(0.62)	(0.71)	(1.14)	(0.69)	(0.75)
F6	-16.68	4.69	25.11	-1.32	11.93
	(1.08)	(0.64)	(1.50)	(0.86)	(0.88)
F7	21.95	-13.94	-4.67	23.73	30.50
	(2.07)	(0.63)	(2.32)	(1.40)	(1.25)
F8	3.96	14.21	10.34	35.46	41.26
	(0.89)	(1.01)	(1.37)	(2.33)	(2.36)
F9	8.63	15.56	-1.20	41.11	28.83
	(0.78)	(2.41)	(1.16)	(1.33)	(1.54)
F10		7.97	8.90	-17.37	1.18
		(1.18)	(1.11)	(1.59)	(1.15)
F11		9.55	-0.62	14.42	6.52
		(0.88)	(1.49)	(1.11)	(1.74)

Note: The table reports the annualized return with the 0.5% transaction cost. The daily volatilities of the year are in parenthesis. The units in the table are all %.

Table 2.8 shows the annualized return after deducting transaction costs (0.5%), and the standard deviations of the daily return per year. It indicates that the annualized return of BGSA-SVM is not stable. Particularly with DAX, the annualized return ranges from -15.17% to 41.26%. For this problem, the stability of the return can be improved by using two methods. The first is to increase the update frequency of the BGSA-SVM model. This study uses only one trained model for predictions in a trading year, due to the limitation in computing power. If computing power were sufficient, the forecasting model could be

updated daily. The return is more stable when the model is updated more frequently. In other words, if the model is trained on a daily basis and the prediction model is updated daily, the annualized return of around 11% can be stably obtained. Second, the stability of the return can be improved by enhancing the trading strategy.

The annualized return of the BGSA-SVM model is not affected by the volatility of stock indices. Appendix A.3.3 shows that there is no significant relationship between annualized return of BGSA-SVM and daily standard deviation.

Table 2.9: Sharpe Ratios of BGSA-SVM and benchmark models

	FTSE100	S&P500	NKY	CAC40	DAX
BGSA-SVM	**0.71**	**0.70**	**0.95**	0.57	**0.59**
RW	0.17	-0.31	0.26	0.23	-0.17
SVM	0.19	-0.10	0.40	0.12	-0.32
$best_1$	0.27	0.35	0.39	**1.03**	0.50
$best_2$	0.08	0.57	-0.05	0.72	0.39
NR	-0.41	0.01	-0.29	-0.48	-0.11

Note: Bold is the highest Sharpe Ratio in the models for each index.

The Sharpe Ratio is also used to measure the trading performance. The equation of the Sharpe Ratio is given in Appendix A.3.1. The Sharpe Ratio and its mean of each index are presented in Table 2.9. The risk-free ratio employed here is the average Libor overnight during the forecasting period. For example, for FTSE100, the risk-free ratio is the average of the GBP Libor overnight during the period 1997-2016. The Sharpe Ratios of BGSA-SVM for the five stock indices are all positive, and are significantly better than the Sharpe Ratios of the benchmark models. Only the Sharpe Ratios of $best_1$ and $best_2$ are higher than that of BGSA-SVM in the CAC40, which is because the standard deviations of annualized returns of $best_1$ and $best_2$ are small in CAC40. This is not the case with the other indices for $best_1$ and $best_2$.

Looking closely at Table 2.4, Table 2.7 and Table 2.8, that financial crises do not have any effect on the predictive accuracy of the BGSA-SVM model. When Buy-and-Hold is negative, the return of BGSA-SVM is not affected by it. For example, during the periods of F1 and F2, NKY, CAC40 and DAX all have significant drops in prices, and the annual decreases are more than 20%, even close to 50%. However, the performances of the predictions with the BGSA-SVM model are all good, and even when the model loses money, its predicting losses are also lower than the losses of Buy-and-Hold. The distribution of losses with BGSA-

SVM model prediction has no correlation with financial crisis. This all indicates that BGSA-SVM prediction and the trading strategy are proved to be effective.

2.7 Conclusions

In this chapter, a hybrid BGSA-SVM model is introduced to forecast the daily returns of five popular stock indices in the world. BGSA is a heuristic optimization technique designed to address the optimization problem of discrete functions. BGSA optimizes SVM parameters and inputs based on the gravitation principle and the interaction information from agents of BGSA. Then the optimized parameters and inputs are used to train the SVM classifier. Finally, this SVM classifier predicts the sign of the daily log return in the next trading year. In addition, the data pre-processing method of this study is PCA. The dimension of inputs can be reduced to a certain range by PCA, which means the searching dimension of BGSA is lower. Thus, the problem that the small quantity of training samples cannot support the high-dimensional search is solved.

In summary, the contribution of this study is to introduce a novel machine learning model that is suitable for predicting financial time series. Compared with previous hybrid SVM models, the advantage of BGSA-SVM is that it can forecast financial time series with high-dimensional inputs while losing less information. This means that BGSA-SVM is more likely to produce higher performance. In addition, this is the first application of the BGSA-SVM machine learning model in the financial field.

Based on the technical data during the period 1990 to 2016, the model makes 11 forecasts and trading on four stock indices (S&P500, NKY, CAC and DAX), and 9 forecasts and trading on FTSE. In terms of the empirical results, the predictive results of stock indices with the BGSA-SVM model are better than the benchmark models, which include RW, best predictors, SVM and Buy-and-Hold. Concerning the trading performance, the expected returns of BGSA are higher than zero for all five stock indices, although some annualized returns are lower than zero in some tests. This proves that the five stock markets are not strictly efficient markets. In other words, under the Efficient Markets Hypothesis (EMH) framework, the efficient stock market does not exist in the experiments, not even weak-form market efficiency. Besides, during the financial crisis, the trading performance of BGSA-

SVM does not significantly decline, which shows the robustness of BGSA-SVM in extreme situations.

However, kernel functions are not optimized as they are not considered as parameters, which is a disadvantage of the used approach. There are not many commonly used kernel functions; therefore, it is better to investigate them one by one to select the best kernel for the model. This shortcoming will be researched further. Moreover, the annualized returns are not stable and BGSA-SVM is not always the best predicting model compared with all the benchmark models in 53 test times. If more information is used in prediction, BGSA-SVM promises to perform better. In addition, this chapter emphasizes forecasting more than trading strategy. The instability of the annualized return can be solved by trading strategies such as hedging or other methods.

Chapter 3 Forecasting and trading INDU index and FTSE100 index by MLP, CNN, LSTM and NNs combination

3.1 Introduction

ANNs were used as early as the late 1980s in the field of stock forecasting. Due to the complexity, dynamism and chaoticness of stocks, stock forecasting has proven to be a difficult task. Nonetheless, a large amount of research has been using various types of ANNs to forecast stocks for decades. Prior literature proves that ANNs have the ability to extract information from inputs and provide effective forecasts for stock prices. However, two questions that (1) which ANN has the best performance for stock forecasting among mainstream ANNs structures and (2) how ANNs can further improve the accuracy of stock forecasting remain to be investigated, which motivate this chapter.

This chapter forecasts the INDU index and FTSE100 index using different architectures of ANNs, which include MLP, CNN and LSTM. The structures of MLP, CNN and LSTM for the financial time series data are designed in this chapter, aiming to get better forecasting accuracy. The statistical and trading performances of three neural networks are compared based on the same inputs pool. This chapter examines the binary classification categories and eight classification categories. Then, the predicting results of NNs are combined by traditional Simple Average, GRR and LASSO, respectively. In the trading part, this chapter uses leverage based on the daily forecasting probability to improve the trading performance.

The empirical results show that MLP, CNN and LSTM all beat the Buy-and-Hold. The average accuracy of MLP, CNN and LSTM is 52.32%, 53.06% and 53.63% respectively for the FTSE100 and 52.21%, 52.32% and 53.32% for the INDU. The average annualized returns of MLP, CNN and LSTM are 8.82%, 11.32% and 12.49% respectively for the FTSE100 and 8.57%, 9.37% and 10.25% for the INDU. Although LSTM outperforms MLP and CNN in average annualized returns, LSTM is not significantly better than MLP and CNN regarding the times that LSTM becomes the best performing model in all tests. The

performance of the binary classification categories is better than that of the eight classification categories. This is because that the eight classification categories are more prone to overfitting. In addition, the combination methods do not improve the trading performance, while different leverages based on different probabilities of the forecasting results significantly improve the trading performance.

The structure of the chapter is as follows. The literature review about the use of NNs to predict stock indices is in section 3.2. The dataset and software are introduced in section 3.3 and after that, the employed architectures of NNs and the combination methods are described in section 3.4. Then, the comparison and explanation of the statistical and trading performances for models are respectively presented in sections 3.5 and 3.6. The conclusion and further developments are summarized in section 3.7.

3.2 Literature review

The overview of ANN is introduced firstly in section 3.2.1, which shows the history and development of ANN. Then, the applications of ANN in stock forecasts are presented in greater detail in later sections. Section 3.2.2 reviews the research in the stock forecasting field by MLP. Section 3.2.3 is a summary of the literature on hybrid ANNs. The main hybrid models include combinations of new data pre-processing methods and training methods. New neural network architectures, which include Recurrent Neural Network (RNN), CNN and LSTM, are reviewed in section 3.2.4. New neural network architectures that are designed to deal with stock forecasts usually have better forecasting performances than classical ANN in the stock forecasting field.

3.2.1 The overview of ANN

Research on ANN started in the 1950s when some researchers successfully built a single perceptron by combining the viewpoints of physiology and psychology. The representative researchers in this period include Marvin Minsky, Frank Rosenblatt & Bernard Widrow. In 1969, Minsky & Papert mathematically proved that the perceptron cannot solve many simple problems, which include the XOR problem (Minsky & Papert, 2017). Later, researchers found that the perceptron cannot solve linear inseparability problems but MLP can.

Neural network architectures have been built gradually, and the most representative work is the Back-propagation algorithm. BP was introduced by Rumelhart, Hinton & Williams (1986) to solve the learning problem in MLP. After BP was proposed, research on ANN was developed rapidly, especially deep learning. Deep learning refers to a neural network with multiple hidden layers, which allows architectures that consist of multiple hidden layers to learn data with multiple levels of abstraction (LeCun, Bengio & Hinton, 2015). The representative architectures include CNN, RNN and LSTM (the improved version of RNN). The main contributions of deep CNN are the breakthroughs in processing images, video and audio, whereas RNN and LSTM have strengths in time series data such as text and speech. Deep nets have also shone a light on the financial forecasting field, especially in securities forecasting, stock indices and bankruptcy prediction (Trippi & Turban, 1992; Kimoto et al., 1990; Tam & Kiang, 1991 & 1993).

3.2.2 The application of MLPs in the stock forecasting field

As early as the end of the 1980s, MLPs tried to forecast stock returns. White (1988) proves that the linear method will never predict IBM's common stock daily returns under simple EMH. Then he uses a three-layer feedforward network with five inputs and five hidden units to forecast the daily return of IBM stock. However, White fails to forecast the out-sample returns with his neural network. The results and contributions of White's tests show that non-linear regularities can exist even under simple EMH, but MLPs cannot easily find this non-linear regularity and they are easily trapped into overfitting with as many as 1000 observations.

Research in subsequent years has progressed significantly when it comes to the forecasting of stocks. Kimoto et al. (1990) successfully use a three-layer full connected MLP to forecast the sign of the daily return of Tokyo Stock Exchange Price Indices. Compared with White (1988), the most important improvement made by Kimoto et al. (1990) is that they use more types of inputs. For example, they use the returns for later days, turnover, interest rate, foreign exchange rate, Dow-Jones index price and so on, while White (1988) only uses returns and volatility. The Buy-and-Hold strategy based on prediction shows a stable and better performance than the index's performance. Kimoto et al. (1990) have an optimistic impact on later research. The work of Baba & Kozaki (1992) is similar to that of Kimoto et al. (1990). They use MLP which combines the modified BP method with the random

optimization method to forecast stocks returns in the Japanese market.

In the same period, Yoon & Swales (1991) use a four-layer MLP to forecast the annualized return of companies in the Fortune 500 and Business Week's 'Top 1000' according to their fundamental information. Yoon & Swales (1991) compare the performance of the four-layer MLP with multiple discriminant analysis methods. The performance of their MLP is significantly better than that of multiple discriminant analysis methods. One year later, Swales & Yoon (1992) publish a similar paper to classify well-performing firms and poorly performing firms. Swales & Yoon use more types of inputs in order to add more information to their model. In addition, Swales & Yoon compare the two-layer MLP, three-layer MLP and four-layer MLP. The deeper network performs better. Similarly, Kryzanowski, Galler & Wright (1993) successfully select stocks under the Quebec Stock Savings Plan with a neural network.

Yoon, Guimaraes & Swales (1994) make a hybrid Expert System that combines MLP and Rule-Based approach mainly for investment decision-making. They use ANN to generate a knowledge base for a Rule-Based approach. As with the fuzzy neural stock selection system of Yoon, Guimaraes & Swales (1994), Wong et al. (1992) also conduct similar research that contains a rule base of 32 company rules.

After the mid-nineties, MLP was used in various stock markets around the world. Kai & Wenhua (1997) use GA to train MLP to forecast the Shanghai securities index. Quah & Srinivasan (1999) use MLP to select stocks in the Singapore market. MLPs are also employed to the forecasting tasks in the Taiwan market (Wang & Leu, 1996), the Tokyo stock market (Mizuno et al.,1998) and the Madrid stock market (Fernandez-Rodriguez & Gonzalez-Martel, 2000). Olson & Mossman (2003) use MLP to forecast the stock returns in the Canadian market and compare MLP with the Ordinary Least Squares regression and logistic regression, finding that MLP beats the linear methods. Olson & Mossman (2003) indicate that MLP classifiers with four to eight output categories have better results than either binary classification models or nets with 16 classification categories.

Lam (2004) uses MLP to predict the financial performance of S&P companies with predictors which include 16 financial statement variables and 11 macroeconomic variables. Cao, Leggio & Schniederians (2005) build a three-layer MLP that beats Fama and French's

model in the Chinese stock market. O'Connor & Madden (2006) successfully use external indicators, such as commodity prices and currency exchange rates, to predict the Dow Jones Industrial Average index. Zhu et al. (2008) use a three-layer MLP to study whether the trading volume has an effect on stock index increments under different horizons. They conclude that trading volume has an impact on the stock index, especially under medium and long-term horizons.

In the last ten years, researchers have usually compared the performance of the MLP model with some hybrid neural network models and other new neural network architecture. However, the MLP model does not show significant disadvantages in some research. The results of Guresen, Gulgun & Daim (2011) show that the classical MLP outperforms the dynamic artificial neural network and hybrid neural networks, but only by a slight margin. Guresen, Gulgun & Daim (2011) use Generalized Autoregressive Conditional Heteroscedasticity (GARCH) inputs for MLP and dynamic artificial neural network but get worse results. They explain that the inputs of GARCH have a noise effect on NNs due to the inconsistencies.

In comparison with the ARIMA model, MLP always outperforms the ARIMA model. Mostafa (2010) uses the MLP on the Kuwait stock market, finding that MLP defeats linear regression and ARIMA. Adebiyi, Adewumi & Ayo (2014) compare the forecasting performance of MLP and ARIMA with published stock data obtained from the New York Stock Exchange. Their empirical results show the superiority of MLP over ARIMA. Adebiyi, Adewumi & Ayo (2014) also mention that hybrid MLPs can improve forecasting accuracy.

Moghaddam, Moghaddam & Esfandyari (2016) attempt to improve the layers and neurons in the hidden layer of the MLP model to forecast the NASDAQ exchange rate. The hidden layers they use in the network are 20-40-20 neurons. Their main contribution is to test the effects of the two different activation functions on forecasting results. Their research proves that the MLP model still has forecasting potential, and its performance can be improved by optimizing the hidden layers and activation functions.

3.2.3 The application of hybrid neural networks in the stock forecasting field

Research from the early 1990s has found that the performance of the ANN is affected by input parameters. Therefore, from the late 1990s, various papers have proposed input optimization methods combined with ANN to build models. Hybrid neural networks have also been used to forecast stocks. With regard to stock forecasting, the research of Kim & Han (2000) is representative. They use GA combined with ANN to forecast the daily Korean stock price index. In their hybrid model, GA is used to select the inputs of the neural network and optimize the connection weights in the nets. Their results show that input selection makes a significant contribution to the forecasting performance; GA and BP's optimizations of connection weights have similar impacts on final forecasting performance. Kim & Lee (2004) employ GA-ANN to forecast the Korean stock price index, which once again proves that feature selection improves the performance of ANN. With respect to the optimization problem of connection weights, Zhang & Wu (2009) put forward a heuristic method named Improved Bacterial Chemotaxis Optimization (IBCO) to integrate into the BP-ANN. They use IBCO-BP-ANN to predict the S&P500 index and compare the performance of IBCO-BP-ANN with that of BP-ANN. Their original model has less computational complexity, better prediction accuracy and requires less training time.

Kuo, Chen & Hwang (2001) consider adding the effect of qualitative factors (e.g. political effect) into stock prediction. They use a GA-based Fuzzy Neural Network (FNN) to acquire a qualitative index. Then, they integrate the qualitative index into the technical indices and train them with an ANN. Their hybrid model beats classical ANN. The work of Abraham, Nath & Mahanti (2001) is similar to that of Kuo, Chen & Hwang (2001). Abraham, Nath & Mahanti (2001) use PCA to pre-process the inputs (technical indices) for ANN. They then feed the prediction value and the qualitative variables into an FNN to make a decision. Their model also defeats classical ANN in the Nasdaq stock market. Similar research is conducted by Leigh, Purvis & Ragusa (2002), who empirically examine the New York stock exchange composite index using a sophisticated decision support system combined with a GA-ANN. Armano, Marchesi & Murru (2005) introduce a novel genetic-neural architecture for stock indices forecasting. In their study, the result is a group interaction of some experts, where every expert has an architecture that integrates GA and ANN. Their hybrid model

outperforms Buy-and-Hold in COMIT and the S&P500 index.

Additionally, ANN can be used as a data pre-processing method in combination with other predictive models to forecast stocks. Hassan, Nath & Kirley (2007) propose a hybrid model to forecast stocks returns. The structure of their model is the first to use ANN to convert the daily stock technical indices into independent sets of values that become the inputs to the Hidden Markov Model (HMM). The GA is used to optimize the parameters of the HMM. Their hybrid model outperforms the ARIMA.

Roh (2007) combines ANN with the Exponentially Weighted Moving Average (EWMA), GARCH and Exponential GARCH (EGARCH) respectively to predict the volatility of KOSPI 200. However, their hybrid model is not successful as only the performance of ANN-EWMA is superior to that of classical ANN, while the performances of ANN-GARCH and ANN-EGARCH are worse. Based on the research of Roh (2007), Wang et al. (2011) conduct experiments on the dynamic artificial neural network. They compare the performances of classical ANN, hybrid ANN, DNN and hybrid DNN when predicting the NASDAQ index. Their results are similar to those of Roh (2007), supporting the view classical ANN performs the best. The performance of either ANN or DNN combined with GARCH and EGARCH is worse. They argue that more research should be focused on the selection of inputs.

Qian & Rasheed (2007) claim that the accuracy of their hybrid ANN prediction for the DJIA index reaches 65%. However, they use the Hurst exponent to select the best predictability period (i.e. 1969-1973) for their prediction instead of predicting the index in all periods. In their research, they also mention that the movement of the DJIA index after 1980 is more random. Despite this, their work has contributed to improving the accuracy of stock forecasts. They respectively use ANN, K-Nearest Neighbour (KNN) and Decision Tree (DT) to predict DJIA index return. Then they propose a model that has ensembles of classifiers to integrate the results of ANN, KNN and DT for the final forecasting. Their ensembles model outperforms individual classifiers.

There are also some studies that consider the results of the ANN and other predictive models together when forecasting stocks. For instance, Wang et al. (2012) use the Exponential Smoothing Model (ESM), ARIMA, and the classical ANN to forecast the daily close price of the Shenzhen Integrated Index (SZII) and the daily open price of DJIA index. Then the

results are processed by a weighted average, where the weight is determined by GA. Their hybrid model outperforms ESM, ARIMA, ANN, the equal weight hybrid model and RW.

3.2.4 CNN, RNN and LSTM in the stock forecasting field

The development of neural network algorithms means that more neural networks of different structures have been proposed and applied in the field of stock forecasting. Among these algorithms, CNN, RNN and the LSTM (a variant RNN) are the most representative. RNN performs well in forecasting time series, and thus many scholars go a step further and expect RNN and RNN variants to perform well in stock forecasting. CNN is adept in handling forecasts with a large amount of data and capturing the characteristics of adjacent data, and it thus has significant advantages in high-frequency prediction (Tsantekidis et al., 2017). Saad, Prokhorov & Wunsch (1998) are the first to use RNN in the stock prediction field. They predict short-term stock trends based on historical daily closing prices by using three different networks: Time Delay Neural Network (TDNN), Probabilistic Neural Network (PNN) and RNN. Their results indicate that all three networks are feasible, and RNN shows the best performance in accuracy but with implementation complexity.

RNN has also been combined with other methods to forecast stock prices. Hsieh, Hsiao & Yeh (2011) use RNN to forecast several international stock indices, including DJIA, FTSE100, Nikkei225 and Taiwan Capitalization Weighted Stock Index. They optimize RNN by using the ABC algorithm. In respect of inputs pre-processing, they apply the Haar wavelet to decompose the stock price time series and employ Stepwise Regression-Correlation Selection to choose other fundamental and technical features. Thanks to their exquisite models, all the tested indices make profits. Yoshihara et al. (2014) integrate RNN, the Restricted Bolzmann Machine and Deep Belief Network (DBN) to build a six-layer hybrid NN to forecast the sign of the following day's return for ten stocks on the Nikkei stock market. They aim to evaluate the long-term effects of news and events. Thus, the inputs data of their model are news articles represented as word vectors by the bag-of-words representation. They suggest that hybrid RNN has potential to capture events with long-term effects in the stock market.

As for Agarwal & Sastry (2015), they use the autoregressive moving average model, exponential smoothing model and RNN to forecast the daily close price for 25 stocks on the

Bombay Stock Exchange (BSE). The performance of RNN is better than the two linear models. In order to increase predictive accuracy, they combine the prediction results of three models by a weighted average, where the weight is determined by GA. Their model structure is similar to that of Wang et al. (2012) (mentioned in section 3.2.3).

CNN is applied later in stock forecasting than RNN. Ding et al. (2015) propose a deep learning method for event-driven stock market forecasting based on CNN. Ding et al. (2015) extract events from news text and represent them as dense vectors. CNN and ANN are then used to model both the short-term and long-term impacts of events on the stock piece. Their results show that the performance of CNN is better than ANN because it captures long-term influence better. Compared with state-of-the-art baseline methods for the S&P500 index, their Event Embeddings input Convolutional Neural Network (EB-CNN) model achieves an improvement of nearly six percentage points. CNN shows a solid performance not only in events-based forecasting, but also in high-frequency forecasting. Tsantekidis et al. (2017) propose a deep learning methodology based on CNN to predict the price movement of stocks by tick-data. They compare the forecasting accuracy of CNN, SVM and ANN, and the results show that CNN is significantly better than the other two benchmarks. The predictive accuracy of CNN for the sign of return reaches 59.44% in their study.

Studies on forecasting stocks using LSTM have appeared in recent years, showing that the performance of LSTM is generally better than that of RNN and MLP. Nelson, Pereira & de Oliveira (2017) use LSTM to predict the Brazilian stock exchange return in 15 minutes into the future. LSTM as a variant RNN performs better than RNN in long-term memory, which is more suitable for stock prediction. The prediction task of Nelson, Pereira & de Oliveira is a binary classification problem and the average accuracy of prediction is 55.9%. Compared with the work for forecasting the daily return, shorter-term forecasting tends to obtain higher accuracy. Moreover, they do not use feature selection methods to reduce the dimension of the inputs, as LSTM has the ability to deal with high dimensional data.

Unlike Nelson, Pereira & de Oliveira (2017), Pang et al. (2018) pay attention to the inputs pre-processing for LSTM. They introduce an embedded layer in the LSTM architecture to convert high-dimensional data into low-dimensional data and call the model ELSTM. Another contribution made by them is that they change the weights optimization method by pre-training with an automatic encoder with a Continuous Restricted Boltzmann Machine

(CRBM). This process can avoid the final result falling into local optimization. Their results show that both methods for achieving improvement efficiently enhance the LSTM. The predictive accuracy of daily return for the Shanghai stock exchange index reaches 57.2% and 56.9% with ELSTM and CRBM respectively, and the predictive accuracy for individual stocks is 52.4% (with ELSTM) and 52.5% (with CRBM). They provide evidence for the potential for improvement that the LSTM approach has, showing that inputs pre-processing significantly improves the performance of LSTM.

In addition to CNN and RNN, other deep nets are also applied to stock forecasting. Chen, Leung & Daouk (2003) compare the performances of PNN, RW and Generalized Methods of Moments in the Taiwan stock exchange index. PNN beats the benchmark models in their paper. Enke & Thawornwong (2005) examine level-estimation nets (three-layer ANN, PNN and generalized regression neural network) and classification (ANN, PNN) for their capability to offer an effective forecast of the S&P500 index. However, the trading strategies guided by these networks do not generate significantly higher profits than the Buy-and-Hold strategy. Their networks' performances are not significantly different, but they are all significantly better than linear regression.

Additionally, Enke & Thawornwong (2005) employ the cross-validation technique to improve the generalization ability of the nets. They also point out that a shortcoming of the ANN training of feed-forward NNs is that they are not very stable, since the training process may depend on the choice of a random start. To solve this problem, Asadi et al. (2012) use GA to optimize the initial weights. They also use Levenberg-Marquardt to replace BP to optimize ANN weights. Moreover, they also carry out the input selection. Ticknor (2013) proposes the Bayesian Regularized artificial Neural Network (BRNN) to forecast the stock price. BRNN assigns the networks a nature that automatically penalizes complex models, which reduces the potential for overfitting and overtraining. Another advantage of BRNN is its ability to adapt to different types of data, and thus BRNN outperforms classical ANN without data pre-processing. In summary, new neural network architectures in recent years perform better in stock forecasting than classical ANN models. Adjusting the architecture of the neural network based on the characteristics of the inputs data (data used to forecast stock) has the potential to improve predictive accuracy.

3.3 Dataset and tools

This chapter uses MLP, CNN and LSTM to predict the log return of the INDU index and FTSE100. The outputs of the forecasting models are the categories of the daily log return for stock indices. One task of the models is to predict the sign of the daily log return, which is a binary classification categories problem. The other task is an eight classification categories problem, where the daily log returns are divided into eight categories based on their values. The classification rules of binary and eight classification categories are shown in Table 3.1 and Table 3.2, respectively.

Table 3.1: Classification rules for binary classification categories

Binary classification categories	
If $daily\ log\ return < 0$	Output $y = [1,0]$
Else	Output $y = [0,1]$

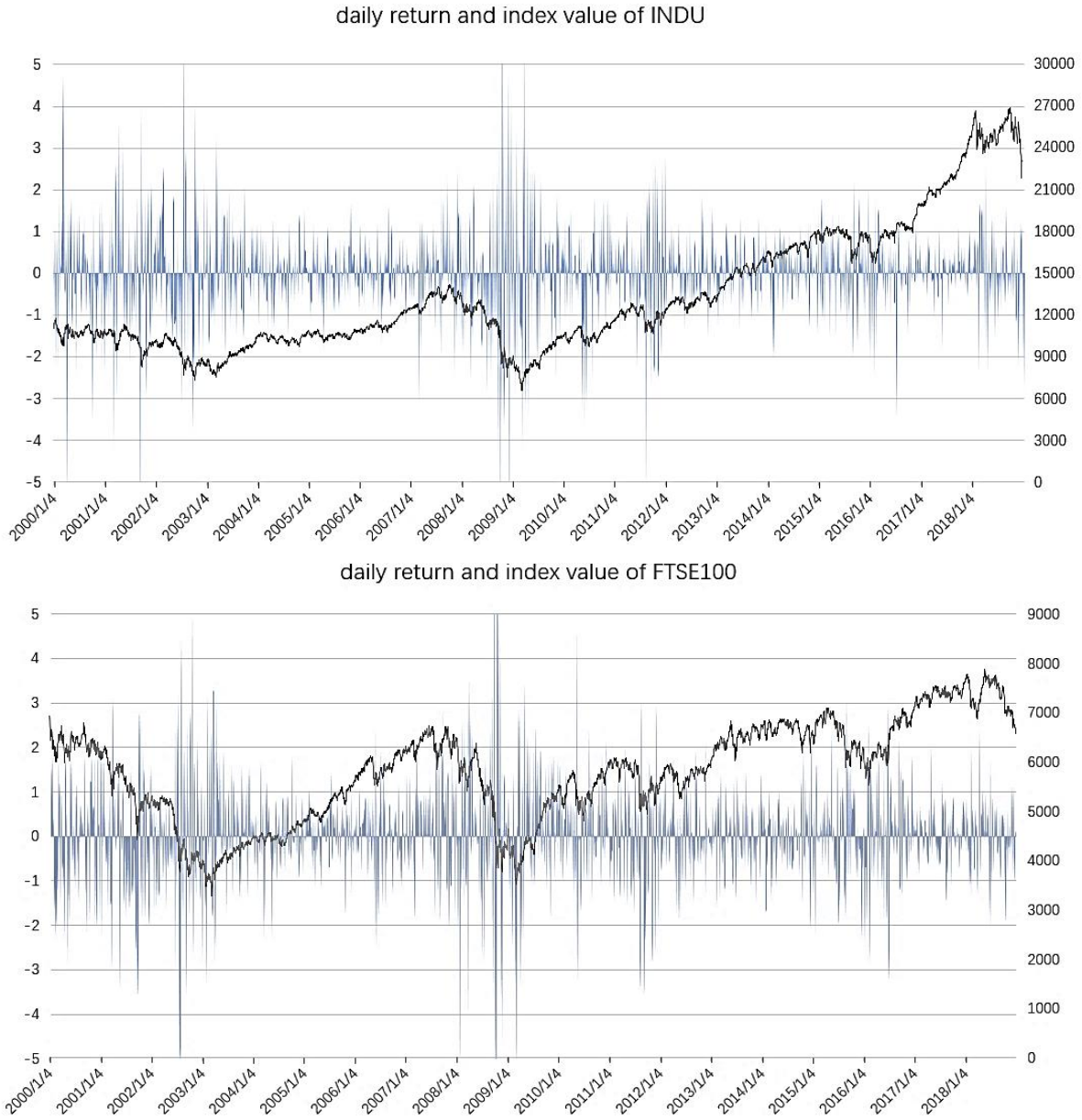
Note: This table describes the classification rule for binary classification categories. If the daily log return is negative, the output is assigned as a vector [1,0], otherwise [0,1].

Table 3.2: Classification rules for eight classification categories

Eight classification categories	
If $daily\ log\ return < -2\%$	Output $y = [1,0,0,0,0,0,0,0]$
If $-2\% \leq daily\ log\ return < -1\%$	Output $y = [0,1,0,0,0,0,0,0]$
If $-1\% \leq daily\ log\ return < -0.5\%$	Output $y = [0,0,1,0,0,0,0,0]$
If $-0.5\% \leq daily\ log\ return < 0\%$	Output $y = [0,0,0,1,0,0,0,0]$
If $0\% \leq daily\ log\ return < 0.5\%$	Output $y = [0,0,0,0,1,0,0,0]$
If $0.5\% \leq daily\ log\ return < 1\%$	Output $y = [0,0,0,0,0,1,0,0]$
If $1\% \leq daily\ log\ return < 2\%$	Output $y = [0,0,0,0,0,0,1,0]$
If $2\% \leq daily\ log\ return$	Output $y = [0,0,0,0,0,0,0,1]$

Note: The output is an eight-dimensional vector as the task is eight classification categories. There are eight output vectors, corresponding to eight daily log return intervals.

Figure 3.1: Stock daily performance of INDU and FTSE100 from 2000 to 2018



Note: The blue bar represents the daily return corresponding to the left vertical axis (%). The black line represents the index value corresponding to the right vertical axis.

The data run from January 2000 to January 2019, including the financial crisis of 2007-2008, which can show the performances of the models in extremely poor financial situations. The performances for the INDU and FTSE100 from 2000 to 2018 are displayed in Figure 3.1.

The summary of daily log returns is shown in Table 3.3. Both stock indices time series are non-normal and stationary, while the skewness is all negative and the kurtoses are all high. The Jarque–Bera statistic confirms that the FTSE100 and INDU return series are non-normal at the 99% confidence level. The Augmented Dickey-Fuller (ADF) reports that the null hypothesis of a unit root is rejected at the 99% statistical level for the two stock indices.

Table 3.3: Summary of daily log return

	FTSE100	INDU
Mean of return	-0.0003%	0.0140%
Standard deviation	1.1571%	1.1168%
Skewness	-0.1621%	-0.1108%
Kurtosis of return	9.6445	11.6362
Jarque-Bera of return (p value)	0.0000	0.0000
ADF (p value)	0.0000***	0.0001***

Note: The model aims to predict the sign of the log returns. *** denotes that the hypothesis of the ADF test is rejected at the 1% significance level.

The total number of independent variables is 100 or 80 for the networks. In comparison to the other NNs models, the MLP with AR & MA model has 20-dimensional Autoregression (AR) and Moving Average (MA) inputs in addition. The features mentioned here refer to the work of Ciner, Gurdgiev & Lucey (2013), Sheta, Ahmed & Faris (2015) and Partalidou et al. (2016), which are shown in Appendix B.1.1. All the nets employed here are tested 14 times for each stock index from 2000 onwards. For every test, the start and end dates are shown in Appendix B.1.2. Each trading year is set up to have 252 trading days. All nets specify five consecutive trading years as the in-sample set and the next trading year as the out-of-sample set.

Data preparation is entirely conducted in Python 3.5, relying on the packages Numpy (Van Der Walt, Colbert & Varoquaux, 2011) and Pandas (McKinney, 2010). The deep learning networks used are developed with Keras on top of Google TensorFlow. The combination part is processed by MATLAB 2017.

3.4 Methodology

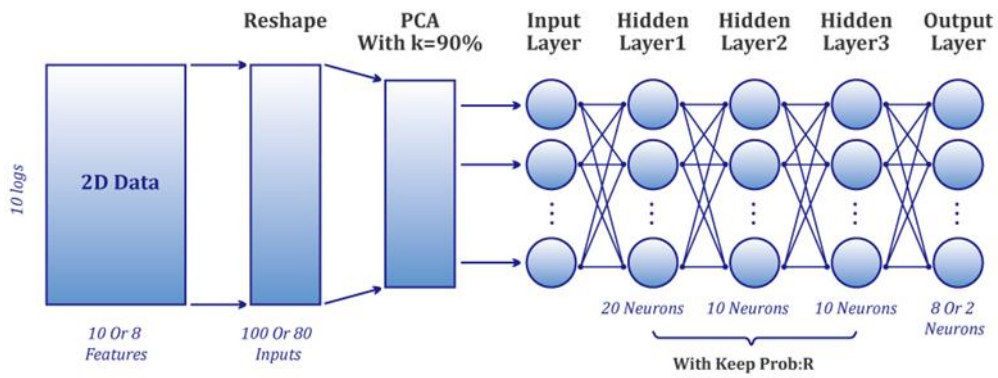
3.4.1 The Multi-layer Perceptron Model for forecasting stock indices

MLP is a kind of traditional neural network architecture, which is also one of the most widely used neural network. MLP has the capability to approximate arbitrary functions (Principe, Euliano & Lefebvre, 2000). This suggests that MLP has potential when it comes to problems of non-linear dynamics and function mapping. Section 3.4.1.1 introduces the structure of the MLP designed by this study, which includes the data pre-processing, layers and numbers of neurons, the methods of optimization and the dropout technique. Then, section 3.4.1.2 and 3.4.1.3 describe the structures of the single neuron and the BP algorithm in detail.

3.4.1.1 The structure of the MLP in this study

The architecture of the MLP forecasting model employed in this chapter is shown in Figure 3.2. The initial data is 2D data with 10 or 8 features and 10 lags. This chapter sets two different inputs pools for MLP. The difference between two inputs stems from whether or not autoregression and moving average have been added.

Figure 3.2: The architecture of the MLP forecasting model in this study



Note: 2D data is a matrix, which means the size of data is 2 dimensional. The 2D data in this paper has 10 features (or 8 features) and their 10 lags. The 2D data matrix is reshaped to a vector with 100 (or 80) inputs. k is the keep value of variance, which means the PCA keeps 90% linear information.

Since the value range for different features is different, the features are normalized individually. In this step, all inputs are converted to a range from 0 to 1. Then the 2D data is reshaped to 1D data (a vector). After that, I run PCA with k value equal to 90%. The neuron number of the inputs layer is decided by the inputs number after the PCA. The advantage of the MLP used here is that it is simpler and faster than the CNN and LSTM structures. The disadvantage is that MLP easily falls into the local optimum when dealing with high-dimensional data. Thus, the data is pre-processed using PCA, which also causes some information loss.

Compared with the research of Kimoto et al. (1990), Baba & Kozaki (1992) and Yoon & Swales (1991), this study examines fully connected nets that have more neurons in each hidden layer. The reason for this choice is based on the tests. The forecasting results are easily trapped into the same class (all negative or all positive) when the nets are simple (with fewer hidden layers and fewer neurons) and thus I use more complex nets with more neurons to solve this problem.

MLPs are usually trained with the BP algorithm. This study also uses the BP algorithm for network training. In addition, this research employs the adaptive moment estimation optimizer (AdamOptimizer), Gradient Descent (GD) and Stochastic Gradient Descent (SGD) to optimize the weights in nets. Based on the test results, this study finds that GD and SGD are easily trapped into local optimum compared with the AdamOptimizer. Thus, the final models all use the AdamOptimizer, which is an improved optimization method based on GD. The learning step in every iteration has a certain range for the AdamOptimizer. In this way, the value of weights can be more stable than GD and SGD. The AdamOptimizer provided by TensorFlow can automatically control the learning speed according to the first-order moment estimation and second-order moment estimation for every weight in the optimization.

Another technique adopted here is Dropout, which was proposed by Hinton et al. (2014) to solve the over-fitting problem in deep learning. Dropout with Keep probability 60% is employed because the MLP I use is complicated and Dropout reduces over-fitting by randomly removing a certain probability of neurons during the training of the network. If Dropout with Keep probability 60% is used for a layer, when training, each iteration only randomly activates 60% of the neurons in that layer.

3.4.1.2 The structure of the Processing Element (PE) in the MLP

The MLP structure and the overall method used in this chapter have now been introduced, and the mathematical basis of MLP are as follows. A neuron is the most basic unit (component) that makes up NNs. Therefore, in order to introduce MLP or other NNs, this section first introduces artificial neurons. Each neuron can accept a set of input signals from other neurons in the system. Each input corresponds to a weight, and the weighted sum of all inputs determines the activation state of the neuron. As shown in Figure 3.3, the n inputs are represented by $x_1, x_2, \dots, x_n$ respectively, and the corresponding weights are $w_1, w_2, \dots, w_n$ respectively. All inputs and weights are shown as an inputs vector X and a weights vector W:

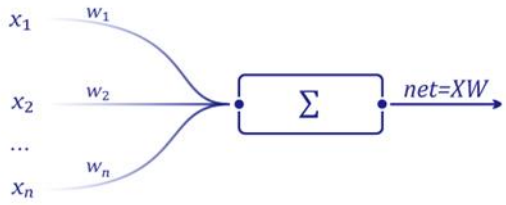
$$\mathrm{X} = (x_1, x_2, \dots, x_n) \tag{3.1}$$

$$W = (w_1, w_2, \dots, w_n)^T \tag{3.2}$$

An artificial neuron without an activation function is expressed as:

$$\mathrm{net} = \mathrm{XW} \tag{3.3}$$

Figure 3.3: The basic model of an artificial neuron



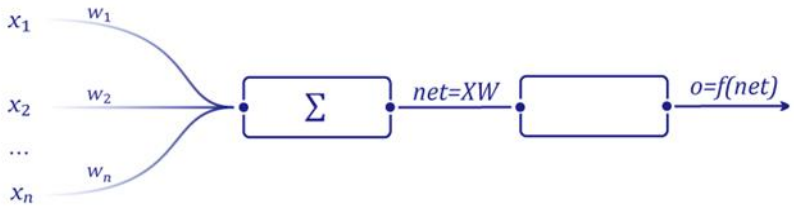
Note: An artificial neuron multiplies two vectors: $\mathrm{X} = (x_1, x_2, \ldots, x_n)$ and $W = (w_1, w_2, \ldots, w_n)^T$.

A neuron should produce an output when it obtains an inputs vector. Each neuron has a threshold. When the net value obtained by the neuron exceeds the threshold, the neuron is in an activated state; otherwise, it is in a suppressed state. This step is expressed as an activation function:

$$\mathrm{o} = f(\mathrm{net}) \tag{4}$$

The McCulloch-Pitts (M-P) model is composed of the basic model of artificial neurons and the activation function (Funahashi, 1989), which can also be called the Processing Element, as shown in Figure 3.4.

Figure 3.4: M-P model



Note: M-P model adds an activation function to an artificial neuron. An appropriate activation function can significantly improve the performances of NNs.

3.4.1.3 BP algorithm

This section explains how the parameters of neurons are trained in the BP algorithm in multi-forward training. The training steps include:

（1） Take a sample (X_i, Y_i) in the sample set

（2） Calculate the actual output of the network O

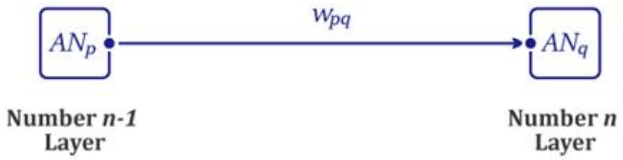
（3） Obtain calculation errors $D = d(Y_i - O)$; where $d(\cdot)$ is the function that calculates errors

（4） Adjust W according to D

（5） Repeat the above process until the times of training are reached or the error does not exceed the certain range.

In the above steps, the key issue is to solve the problem of adjusting the weight of layers. The BP algorithm is the most mainstream solution. It uses the error of the output layer to estimate the error of one layer before this output layer, and then employs the backward process to use estimated error to continue to estimate the error of the layer before. Repeating in this way, the error estimates for all other layers can be obtained. Although the accuracy of this error estimation continues to decrease with backward propagation, it provides an effective approach to training multi-layer networks.

Figure 3.5: The weights adjustment methods of the output layer



Note: The weights of the output layer are directly adjusted based on loss.

This study uses the notation in Figure 3.5 to explain the adjustment of the weights of the output layer. The calculation of hidden layers will be discussed later. In Figure 3.5, AN_q is the q^{th} neuron of the output layer, AN_p is the p^{th} neuron of the layer before the output layer, and w_{pq} is the weight of the two neurons.

For the i^{th} training:

$$w_{pq}^{i} = w_{pq}^{i-1} + \Delta w_{pq}^{i} \tag{3.5}$$

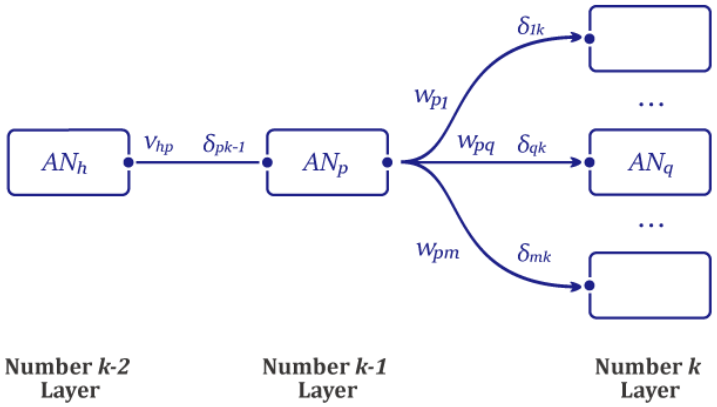
Taking the GD optimization method as an example, Δw_{pq}^{i} can be calculated by:

$$\Delta w_{pq}^{i} = \alpha \delta_q^i o_p^i \tag{3.6}$$

$$\delta_q^i = f_n'(net_q^i)\mathrm{d}(y_q^i - o_q^i) \tag{3.7}$$

Where δ_q^i is the error of AN_q, which is determined by the output of AN_q, the real value of AN_q and the output of AN_p. $f_n(\cdot)$ is the activation function of the n^{th} layer, and $\mathrm{d}(\cdot)$ is the loss function.

Figure 3.6: The weights adjustment method of hidden layers



Note: The loss of hidden layers needs to be estimated. Then the weights are adjusted based on the loss of hidden layers.

Figure 3.6 displays the weights adjustment method of hidden layers. It is assumed that when AN_h is a hidden layer neuron, δ_{pk-1} cannot be directly calculated but Δv_{hp} is determined by δ_{pk-1}, thus we need to give δ_{pk-1} an estimate. Figure 3.6 indicates that the value of δ_{pk-1} is related to δ_{1k}, δ_{2k},…, δ_{mk}, and thus δ_{1k}, δ_{2k},…, δ_{mk} can be used to estimate δ_{pk-1}. Meanwhile δ_{pk-1} is associated with w_{p1}, w_{p2},…, w_{pm} and δ_{1k}, δ_{2k},…, δ_{mk}. We can deduce that the output error of AN_p is related to $w_{p1}^i\delta_{1k}^i + w_{p2}^i\delta_{2k}^i + \ldots + w_{pm}^i\delta_{mk}^i$. In this way, we can approximate the difference between the ideal output value of AN_p and real output. According to:

$$\delta_{pk-1}^i = f_{k-1}'(net_q^i)\mathrm{d}(w_{p1}^i\delta_{1k}^i + w_{p2}^i\delta_{2k}^i + \ldots + w_{pm}^i\delta_{mk}^i) \tag{3.8}$$

We can obtain:

$$\Delta v_{hp}^i = \alpha\delta_{pk-1}^i o_{hk-2}^i \tag{3.9}$$

$$v_{hp}^i = v_{hp}^{i-1} + \Delta v_{hp}^i \tag{3.10}$$

Where o_{hk-2}^i represents the output of the h^{th} neuron on the $\mathrm{k}-2^{th}$ layer.

In the above way, the weights in the BP neural network are adjusted. The weights adjustment methods of CNN and LSTM in this chapter are also the BP method.

3.4.2 Convolution Neural Network for forecasting stock indices

The original form of CNN was proposed by Fukushima (1980). The hidden layer of the neocognitron model he proposed is composed of simple-layer and complex-layer, which partially realizes the functions of the convolution layer and pooling layer in CNN. LeCun et al. (1989) use SGD for learning in the networks they build. They were the first to use the word 'convolution' when introducing the network structure, which is where CNN gets its name.

The common hidden layers of CNNs include the convolutional layer, the pooling layer and the fully connected layer. The fully connected layer was introduced above and the pooling layer is not used in this study, so I will only introduce the convolutional layer in this section. Its function is to extract features from the data using convolutional kernels (patches). When the convolutional kernel is working, the input features are regularly scanned. The input features are multiplied by the matrix elements in the receptive field, and we can sum the deviation:

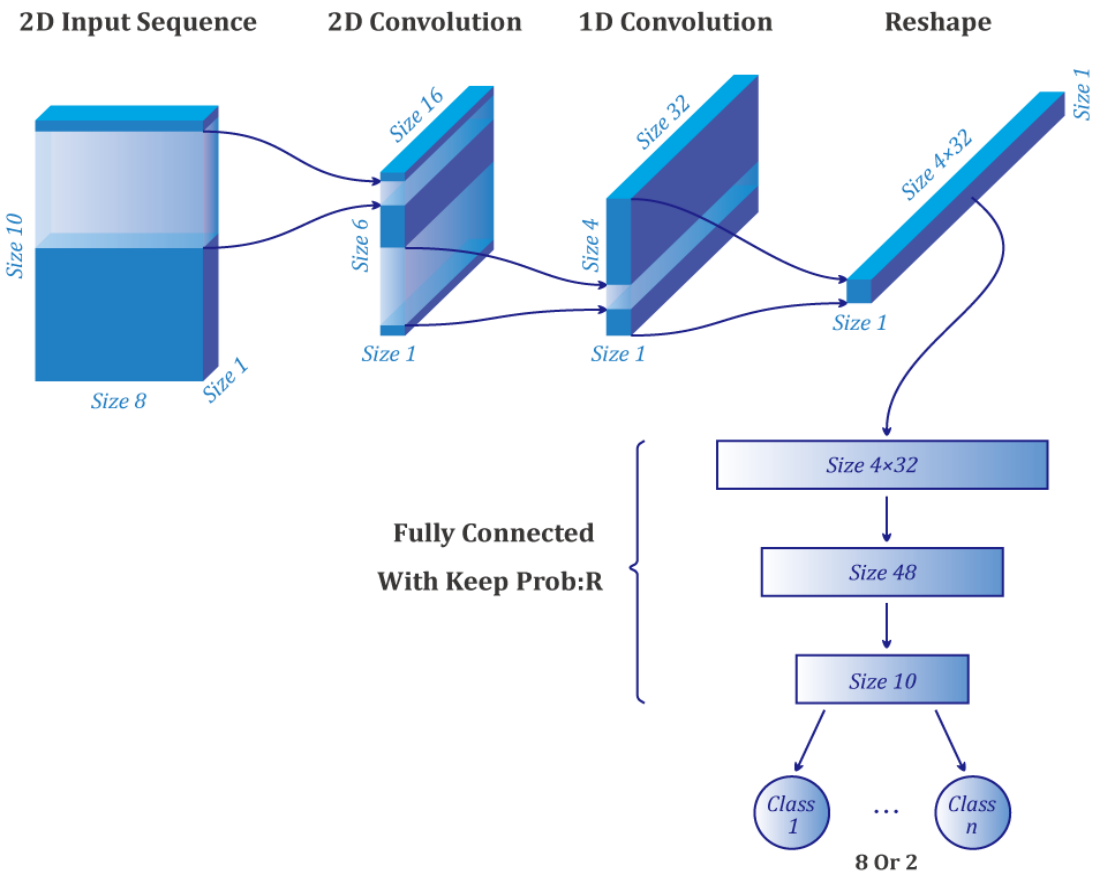
$$Z^{l+1}(\mathrm{i},\mathrm{j}) = [Z^l \otimes w^l](\mathrm{i},\mathrm{j}) + \mathrm{b} \tag{3.11}$$

$$\text{where } (\mathrm{i},\mathrm{j}) \in \{0,1,\dots,L_{l+1}\}, \qquad L_{l+1} = \frac{L_l + 2p - f}{s_0} + 1$$

where b is the deviation amount, Z^l represents the output of the l^{th} layer and the input of the $l+1^{th}$ layer. Similarly, Z^{l+1} represents the output of the $l+1^{th}$ layer and the input of the $l+2^{th}$ layer. L_{l+1} is the size of Z_{l+1}. $Z^{l+1}(\mathrm{i},\mathrm{j})$ is the data at the location (i,j) in receptive field, which is a vector. K is the number of filters. f, s_0 and p are the parameters of a convolutional layer, corresponding to the size of patches, stride and the numbers of padding.

As shown in Figure 3.7, the hidden layers of CNN constructed in this study consist of two convolution layers and the three-layer fully connected network. The initial inputs used in this study are 2D data with 8 features and 10 lags (see Appendix B.1.1 for details).

Figure 3.7: The architecture of the CNN forecasting model in this study



Note: This figure shows the architecture of the CNN model. The blocks show the shapes of the data. For example, the first block represents a matrix size (10,8).

Figure 3.7 displays the structure of the CNN model constructed in this research:

(1) The first hidden layer is a 2D convolutional layer with 16 filters of size (5,8), the stride is 1, with Rectified Linear Unit (ReLU) activation function (ReLU is a type of activation function. Mathematically, it is defined as $y = max(0, x)$).

(2) The second hidden layer is a 1D convolutional layer with 32 filters of size (1,4), the stride is 1, with ReLU activation function.

(3) Then reshape outputs into 1D form, enter the reshaped outputs into the three-layer fully connected networks with ReLU activation function, where the fully connected networks use the Dropout technique with probability 60%.

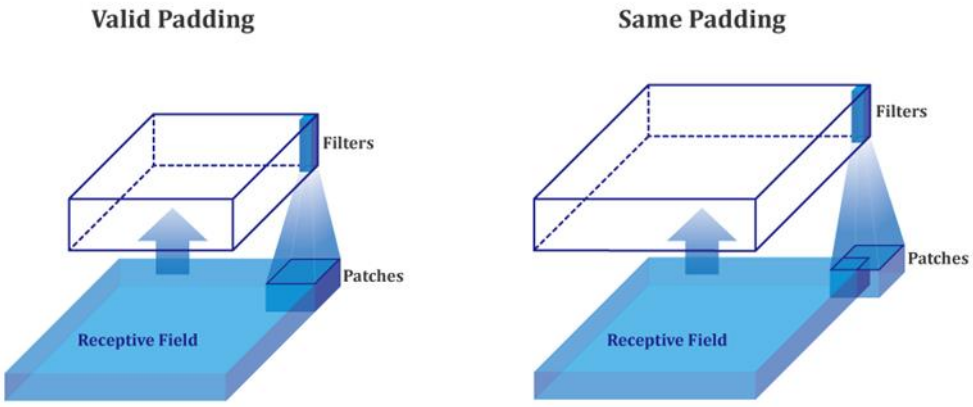
(4) Finally, I output the forecasting probability of each class with a SoftMax activation function.

The ReLU activation function and SoftMax activation function are common activation

functions in NNs training. SoftMax activation function is the gradient logarithm normalization of the finite term discrete probability distribution. In this study, the SoftMax activation function in the output layer gives the probability for daily return categories. For example, for a two categories classification, the output of day t is $[p, 1-p]$, for an eight categories classification, the output of day t is $[p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8]$, where the sum is one.

Unlike the general form of CNN, this study does not use a pooling layer after the convolutional layer, which is for two reasons: first, the max pooling method commonly used in the pooling layer is not suitable for processing financial data. Max pooling is more suitable for processing pixel data for image reduction. Second, the padding method of the patch in this study is 'Valid' rather than 'Same'. 'Valid' makes the length and width of the data smaller after passing through the convolutional layer, and the total amount of data (length*width*height) is also smaller. Therefore, the pooling layer used to reduce the amount of data can be omitted. The explanation of 'Valid' and 'Same' is given in Figure 3.8.

Figure 3.8: The convolutional lingo



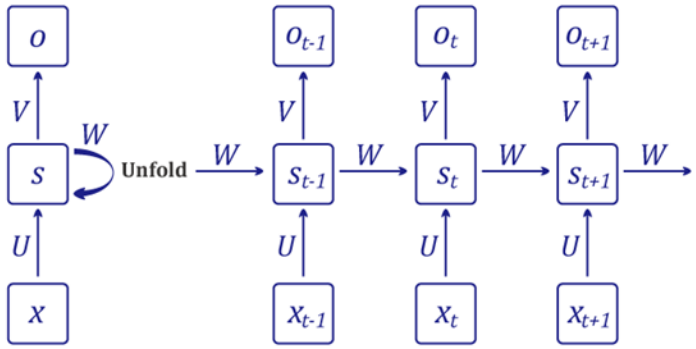
Note: In this study, Valid Padding helps to reshape the data. In this way, the pooling layer can be omitted, thereby reducing the loss of information.

3.4.3 Long Short-Term Memory for forecasting stock indices

Long Short-Term Memory is one of the many variations of RNN, which was first proposed by Jordan in 1986 based on the Hopfield network (Elamn, 1990). Under the parallel distributed processing theory, every neuron in hidden layers of Jordan's network connects with a 'state unit' which is used for inputs lags. In 1990, Elamn proposed the first fully

connected RNN. In 1991, Hochreiter discovered the long-term dependencies problem of RNN, that is, when learning the sequence, gradient vanishing and gradient explosion will appear. This means that RNN is unable to learn long-term nonlinear relationships. In order to solve this long-term dependencies problem, many improved algorithms have been developed, among which the most widely used is LSTM.

Figure 3.9: Unfolding the architecture of a recurrent neural network



Note: RNN uses the same V (weights of the output layer), U (weights of the hidden layer) and W (transition weights of the hidden state) for input and output at different t.

The RNN and LSTM models are introduced as follows. Figure 3.9 shows that an RNN model is unfolded into a full network. x_t is the input vector at time *t*. s_t is the hidden state at time t; it is calculated based on the input vector and the previous hidden state. s_t is calculated as:

$$s_t = f(Ux_t + Ws_{t-1}) \tag{3.12}$$

where $f(\cdot)$ is the activation function. The initial hidden state s_0 used to calculate the first hidden state s_i is typically initialized to zero. *U* and *W* are the weights of the hidden layer and transition weights of the hidden state respectively. o_t is the output at time t, which can be formulated as follows:

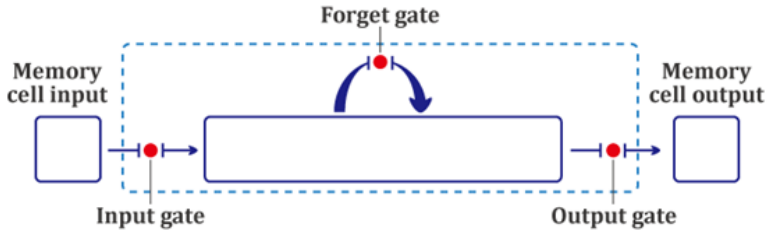
$$o_t = f(Vs_t) \tag{3.13}$$

where *V* are the weights of the output layer.

As explained above, RNN has difficulty learning long-term dependencies because of the vanishing gradient and gradient explosion problems. LSTM is an effective solution for combatting vanishing gradient and gradient explosion by memory cells. The memory unit consists of four units: an input gate, an output gate, a forget gate, and a self-recurrent neuron, as shown in Figure 3.10. The interactions between adjacent memory cells and the memory cell itself are controlled by the gates. The input gate controls whether the input signal can

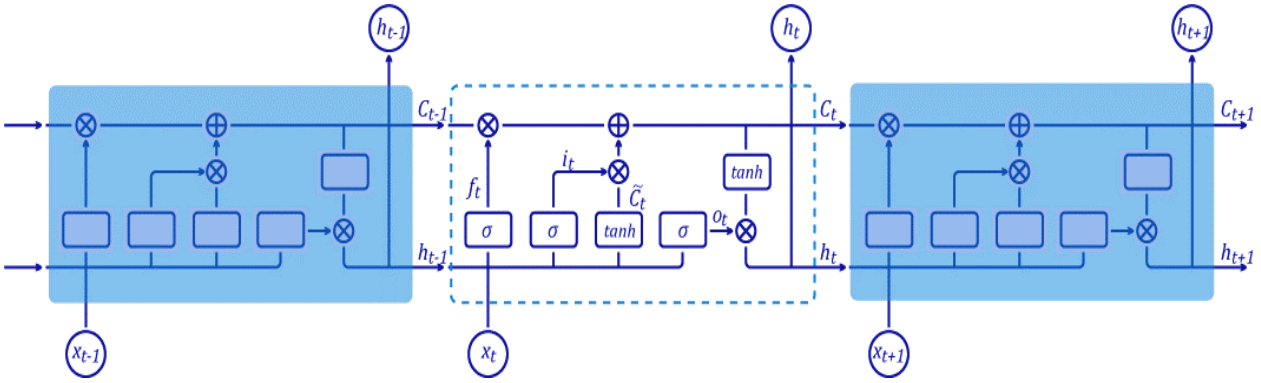
change the state of the memory cell. Additionally, the output gate governs the state of the memory cell, determining whether it is able to change another memory cell's state. Moreover, the forget gate decides to remember or forget its previous state.

Figure 3.10: The architecture of the LSTM memory cell



Note: The dashed box shows a memory unit in LSTM. The three red dots are the input gate, forget gate and output gate. The rectangle with solid lines in the middle is a self-recurrent neuron.

Figure 3.11: The repeating module in the LSTM



Note: The dashed box shows the execution details of a unit of LSTM. ⊗ is the tensor product. ⊕ means the matrix addition.

Figure 3.11 shows an LSTM model being unrolled into a full network, which describes how the value of each gate is updated. In Figure 3.11, x_t is the input vector at time t. i_t is the values of the input gate and $\tilde{C}_t$ is the candidate sate of the memory cell at time t:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{3.14}$$

$$\tilde{C}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{3.15}$$

f_t is the value of the forget gate and C_t is the state of the memory cell at time t, which can be calculated by:

$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \tag{3.16}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{3.17}$$

o_t is the value of the output gate at time t. h_t is the value of the memory cell at time t, which

can be formulated as:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o C_t + b_o) \tag{3.18}$$

$$h_t = o_t * \tanh(C_t) \tag{3.19}$$

Where W_i, W_f, W_c, W_o, U_i, U_f, U_c, U_o and V_o are weight matrices. b_i, b_f, b_c and b_o are bias vectors. The LSTM network used in this study is a basic LSTM network with 10 neurons in the hidden layer connected to an output layer with the SoftMax activation function.

3.4.4 Forecasting combination techniques

In this section, three techniques are used to combine MLP, CNN and LSTM forecasts. These combination techniques are used by Sermpinis et al. (2012) in their paper that forecasts the EUR/USD exchange rate. The aim of these techniques is to follow the best individual forecasting or significantly improve the combination of forecasting. This section combines the predicted outputs of NNs with the combination techniques used by Sermpinis et al. (2012). The combination techniques combine the prediction probability of three NNs for binary classification. All out-of-sample predictions are used for the parameters' estimation for the combination techniques.

3.4.4.1 Simple Average

The first combination technique used in this section is a simple average, which can be considered as a benchmark forecast combination model. Given the NNs' forecasting vectors f_{MLP}^t, f_{CNN}^t and f_{LSTM}^t at time t, the forecasting vectors at time t are calculated as:

$$f_{SA}^t = \frac{f_{MLP}^t + f_{CNN}^t + f_{LSTM}^t}{3} \tag{3.20}$$

Where the f_{SA}^t, f_{MLP}^t, f_{CNN}^t, f_{LSTM}^t are two-column matrices (the elements are probabilities, the sum of every row is one), which can also be considered as a one-dimensional vector.

3.4.4.2 Granger and Ramanathan Regression Approach (GRR)

Bates and Granger (1969) indicate that combining a set of forecasts performs better than individual forecasts. According to this idea, Granger and Ramanathan (1984) propose three

regression models as follows:

[GRR-1]:

$$f_{c1} = a_0 + \sum_{i=1}^{n} a_i f_i + \varepsilon_1 \tag{3.21}$$

[GRR-2]:

$$f_{c2} = \sum_{i=1}^{n} a_i f_i + \varepsilon_2 \tag{3.22}$$

[GRR-3]:

$$f_{c3} = \sum_{i=1}^{n} a_i f_i + \varepsilon_3, \text{ where} \sum_{i=1}^{n} a_i = 1 \tag{3.23}$$

Where $f_i \; for \; i = 1, \dots, n$ are the individual one-step-ahead predictions. f_{c1}, f_{c2}, f_{c3} are the combination predictions. a_0 is the constant term of the regression. a_i is the coefficient of the regression for each model. $\varepsilon_1, \varepsilon_2, \varepsilon_3$ are the error terms.

The GRR-1 model is applied in the research of Sermpinis et al. (2012). This study also selects GRR-1. The forecasting results of NNs in this study are probabilities. Thus, the estimation function needs some adjustment:

$$(f_{c1} - 0.5) = a_0 + \sum_{i=1}^{n} a_i (f_i - 0.5) + \varepsilon_1 \tag{3.24}$$

The GRR model at time t is specified as follows:

$$(f_{c1}^t - 0.5) = -0.0029 + 1.8650(f_{MLP}^t - 0.5) + 4.0715(f_{CNN}^t - 0.5) + 4.2992(f_{LSTM}^t - 0.5) \tag{3.25}$$

3.4.4.3 Least Absolute Shrinkage and Selection Operator (LASSO)

LASSO is proposed by Tibshirani (1996) based on Ridge Regression. Compared to linear regression, LASSO is suitable for solving the problem of over-fitting or multi-collinearity between variables. Compared to Ridge Regression, LASSO is more adaptable to in-samples of fewer variables with medium/large effects. Given a set of samples with the vectors of independent and dependent variables:

$$\begin{bmatrix} X_1^T \\ \vdots \\ X_N^T \end{bmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1N} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{NN} \end{pmatrix}, Y = [y_1, \dots y_N]^T \quad (3.26)$$

The LASSO coefficients are estimated as follows:

$$\hat{\beta}_{lasso} = \arg min_{\beta} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{d} \beta_i x_{ij} \right)^2 \right\} \text{subject to} \sum_{j=1}^{d} |\beta_i| \leq k, k > 0. \quad (3.27)$$

Where k is the 'tuning parameter', which controls the amount of shrinkage applied to the coefficients. In this case, one is required for the constraint k since the forecasting results of NNs in this study are probabilities. Thus, the range of the combination should be zero to one.

$$|\beta_{MLP}| + |\beta_{CNN}| + |\beta_{LSTM}| \leq 1 \quad (3.28)$$

Subject to the constraint model takes the form:

$$f_l^t = 0.1119 f_{MLP}^t + 0.3923 f_{CNN}^t + 0.4715 f_{LSTM}^t + \varepsilon^t \quad (3.29)$$

3.5 Statistical performance

The annualized accuracy of ANNs (MLP, CNN and LSTM) and combination methods are presented in this section. The accuracy is calculated every trading year from 2004 to 2018. This study also uses eight classification architecture NN model to test accuracy, which is shown in Appendix B.2.1.

3.5.1 Statistical accuracy of MLP, CNN and LSTM

The statistical accuracy of the binary classification of MLP, MLP with AR & MA, CNN and LSTM is presented in Table 3.4 (for the FTSE100) and Table 3.5 (for the INDU). A direct way to show the effectiveness of prediction is to see if the accuracy of the out-of-sample is higher than 50%. Tests of the out-of-sample lower than 50% are in bold. Table 3.4 and Table 3.5 indicate that the likelihood of obtaining accuracy lower than 50% in the out-of-sample is small, since it occurs 3 times in 56 predictions for FTSE forecasting and 6 times for the INDU. The accuracy of the prediction of RW being lower than 50% should be 49.8% with the expectation close to 28 times. This also proves that the predictive accuracy of the neural

networks in this study is significantly higher than RW.

Table 3.4: Accuracy for binary classification forecasts for the out-of-sample for the FTSE100

Period	MLP	MLP_with_AR&MA	CNN	LSTM
F1	0.5516	0.5238	0.5278	0.5397
F2	0.5198	**0.4960**	0.5040	0.5635
F3	0.5278	0.5119	0.5357	0.5198
F4	0.5119	**0.4841**	0.5516	0.5516
F5	0.5079	0.5278	0.5397	0.5317
F6	0.5119	0.5040	0.5198	0.5079
F7	0.5357	0.5159	0.5476	0.5357
F8	0.5119	0.5317	0.5159	0.5079
F9	0.5357	**0.4802**	0.5357	0.5635
F10	0.5278	0.5159	0.5516	0.5278
F11	0.5079	0.5198	0.5119	0.5238
F12	0.5238	0.5198	0.504	0.5556
F13	0.5159	0.5198	0.5317	0.5675
F14	0.5357	0.5238	0.5516	0.5119
Average accuracy	0.5232	0.5125	0.5306	0.5363

Note: The tests of the out-of-sample lower than 50% are in bold.

Table 3.5: Accuracy for binary classification forecasts for the out-of-sample for the INDU

Period	MLP	MLP_with_AR&MA	CNN	LSTM
F1	**0.4921**	0.5119	0.5159	0.5119
F2	0.5000	0.5119	0.5079	0.5159
F3	0.5476	0.5159	0.5238	0.5595
F4	0.5079	0.5159	0.5079	0.5476
F5	0.5278	0.5556	**0.4881**	0.5119
F6	0.5476	**0.4921**	0.5675	0.5556
F7	0.5437	**0.4921**	0.5278	0.5079
F8	**0.4921**	0.5278	0.5357	0.5397
F9	0.5437	0.5119	0.5437	0.5556
F10	0.5159	0.5040	0.5040	0.5714
F11	0.5079	0.5040	**0.4881**	0.5357
F12	0.5159	0.5357	0.5317	0.5159
F13	0.5278	0.5079	0.5595	0.5317
F14	0.5397	0.5238	0.5238	0.504
Average accuracy	0.5221	0.5150	0.5232	0.5332

Note: The tests of the out-of-sample lower than 50% are in bold.

Additionally, the average accuracy of all neural networks for both the FTSE100 and INDU is higher than 50%. LSTM especially shows the best performance in average accuracy, at 3.63 percentage points (for FTSE100) and 3.32 percentage points (INDU) higher than the expectation accuracy (50%) of random classification. Moreover, the forecasting accuracy of NNs is not significantly affected by the performance of stock markets in the tests. The financial crisis in 2007 and 2008 corresponds to F4 and F5 respectively. However, only one prediction failure occurs in the experiments of these two periods.

The inputs of MLP with AR & MA have 20-dimensional AR & MA more than the inputs of MLP. However, the forecasting performance of MLP with AR & MA has not increased but decreased, which may result from improper use of inputs. The AR & MA information is extracted from the return lags. Thus, it is unnecessary to add AR & MA with return lags in the inputs pool. Another reason is that the information vanishes after pre-processing the forecasting results of AR & MA. This study uses the normalization method to pre-process the forecasting results of AR & MA, and then inputs the pre-processed results into MLP. Since the magnitude of the forecasting results of AR & MA is different from that of other inputs (such as return), normalization is applied separately for different features. This step will lose the interrelated information among the features. This problem exists not only for AR & MA as inputs, but also for other financial features. Compared to picture recognition, text recognition and speech recognition, the features of financial classification are more challenging to pre-process. For example, the ranges of the pixels in picture recognition are all between 0-255, and thus all features can be normalized uniformly, but financial features are usually not in the same range, as their range in some stock indices is from a thousand to ten thousand, and some dummy financial features are 0 or 1. This study argues that the heterogeneity of inputs affects the accuracy of financial forecasting.

Compared to previous similar studies (Fischer & Krauss, 2018; Pang et al., 2018; Nelson, Pereira & de Oliveira, 2017; Tsantekidis et al., 2017), the accuracy of NNs used here does not reach their accuracy level, which may be due to three reasons. The first is because the UK and US stock markets are highly efficient markets, and thus the forecasting effectiveness will be worse. Pang et al. (2018) use LSTM to predict the Chinese stock exchange index with an accuracy rate of 57.2%, while the similar method used by Fischer & Krauss (2018) for the US stock index only has a prediction accuracy rate 54.3%.

The second reason is that in order to compare the performances among NNs, the same inputs are employed for MLP, CNN and LSTM, which leads to insufficient input lags for LSTM. LSTM specializes in learning data with long lags, but this study only uses 10 lags, which does not reflect the advantages of LSTM (while Nelson, Pereira & de Oliveira (2017) use 100 lags). The reason why this study does not use long lags is that they will increase the inputs dimension and cause the MLP to be easily trapped into overfitting.

The third reason is that this study is predicting daily returns. Predicting the return over a

shorter period can result in higher accuracy. Evidence comes from Nelson, Pereira & de Oliveira (2017) and Tsantekidis et al. (2017) who use tick-data to predict stock indices' performance in the next few minutes and achieve relatively high accuracy (55.9% and 59.44% respectively)

3.5.2 Statistical accuracy of NNs combination

The statistical accuracy of three combination techniques (Simple Average, GRR and LASSO) is presented in Table 3.6 (for the FTSE100) and Table 3.7 (for the INDU). Regarding the average accuracy, the three combination techniques do not perform better than the neural networks alone. For the FTSE100, the average accuracy of the three methods is very close. The average accuracy of 14 experiments is 53.17%, 53.20% and 53.26%, corresponding to Simple Average, GRR and LASSO. For the INDU, the performance is slightly worse, and the average accuracy is 51.93%, 52.78% and 52.69%. After observing the daily prediction results, this study finds that the forecasting results of GRR and LASSO are the same most of the time.

Table 3.6: Annual accuracy of the out-of-sample combination for the FTSE100

	Simple Average	GRR	LASSO
F1	0.5278	0.5357	0.5357
F2	0.5198	0.5437	0.5397
F3	0.5476	0.5278	0.5278
F4	0.5238	0.5397	0.5397
F5	0.5317	0.5317	0.5317
F6	0.5198	0.5278	0.5278
F7	0.5357	0.5119	0.5119
F8	0.5556	**0.4960**	0.5079
F9	0.5278	0.5516	0.5516
F10	0.5516	0.5238	0.5238
F11	0.5079	0.5357	0.5357
F12	0.5000	0.5437	0.5476
F13	0.5595	0.5516	0.5476
F14	0.5357	0.5278	0.5278
Average accuracy	0.5317	0.5320	0.5326

Note: The tests of the out-of-sample lower than 50% are in bold.

Table 3.7: Annual accuracy of the out-of-sample combination for the INDU

	Simple Average	GRR	LASSO
F1	0.5000	0.5198	0.5198
F2	**0.4921**	0.5159	0.5159
F3	**0.4960**	0.5357	0.5278
F4	0.5238	0.5317	0.5357
F5	0.5119	0.5317	0.5317
F6	0.5556	0.5595	0.5595
F7	0.5159	0.5159	0.5159
F8	0.5000	0.5397	0.5357
F9	0.5238	0.5516	0.5516
F10	0.5437	0.5278	0.5238
F11	0.5079	0.5119	0.5119
F12	0.5317	0.5159	0.5159
F13	0.5278	0.5278	0.5278
F14	0.5397	0.5040	0.5040
Average accuracy	0.5193	0.5278	0.5269

Note: The tests of the out-of-sample lower than 50% are in bold.

The combination techniques do not improve the predictive power of NNs, which is different from the results of Sermpinis et al. (2012). They use NNs to do regression fitting of the EUR/USD exchange and the forecasting result is the exchange rate. In their paper, using combination techniques can reduce volatility, trying to approximate the true value. However, this study forecasts the sign of stock indices and the forecasting result is the probability within the range 0 to 1. In the case of binary classification categories, if the value of the first item of the vector is higher than 50%, then the price of the stock is predicted to rise; if the probability is lower than 50%, the price is predicted to decrease. When these probabilities are combined, the combination may turn into a judgment problem as the naive strategy used in this study does not buy more stocks when the higher probability of price increase occurs. Thus, this shortcoming of the trading strategy weakens the performance of combination techniques. This study suggests that adjusting the leverage of buying (or selling) based on the forecasting probabilities given by the NNs will improve the performance.

3.6 Trading performance

The trading strategy in this study is to buy one unit or stay 'long' of the index when the forecast return is positive and sell one unit or stay 'short' of the index when the forecast return is negative. In section 3.6.1, trading performances are calculated annually for MLP, CNN, LSTM, and combination methods. Section 3.6.2 introduces a method to offer different leverages for daily trading based on the daily forecasting probabilities. The annual returns

of eight-classification NNs and NNs combination are in Appendix B.2 and Appendix B.3, respectively. Transaction costs have not been taken into account in this chapter.

3.6.1 Trading performances of NNs and combination methods

The annualized trading performances of MLP, CNN, LSTM, and combination methods are presented in this section. Table 3.8 (for the FTSE100) and Table 3.9 (for the INDU) show the out-of-sample results of MLP, CNN, LSTM and Buy-and-Hold with binary classification. All NNs beat Buy-and-Hold on average annualized returns, but all NNs are not guaranteed to outperform Buy-and-Hold consistently every year. This proves that forecasting the stock index by NNs is not stable. The instability of the forecasting results provides a new direction for future study. This study argues that increasing the updating frequency and the forecasting frequency of the model can reduce the instability of returns. The model can be updated daily during the actual transaction process (while in this study it is updated once a year). Additionally, a higher forecasting frequency can be adopted, such as forecasting and trading once per hour.

Table 3.8: Trading performances of MLP, CNN and LSTM with binary classification categories for the FTSE100 (%)

	Buy-and-Hold	MLP	MLP with AR & MA	CNN	LSTM
F1	14.98	**18.79**	8.47	5.30	6.92
F2	10.74	13.15	-8.39	6.98	**16.01**
F3	0.88	0.19	10.56	**31.59**	11.26
F4	-48.84	-5.29	-45.59	26.30	**27.95**
F5	30.09	12.50	31.70	**40.63**	12.74
F6	7.56	**13.05**	3.16	-4.49	-3.37
F7	-3.23	24.37	1.45	-16.61	**24.96**
F8	6.09	9.93	**17.19**	11.86	-15.95
F9	**13.14**	1.12	-7.05	3.29	10.46
F10	3.11	4.18	1.95	11.45	**15.75**
F11	-13.37	11.32	10.10	-0.46	**11.97**
F12	14.56	12.59	14.50	30.33	**24.93**
F13	7.08	4.14	8.49	-3.00	**20.01**
F14	3.45	3.39	2.67	**15.24**	11.30
Average	3.30	8.82	3.52	11.32	**12.49**

Note: The best performance in every forecast period is in bold. The units in the table are all %.

Table 3.9: Trading performances of MLP, CNN and LSTM with binary classification categories for the INDU (%)

	Buy-and-Hold	MLP	MLP with AR & MA	CNN	LSTM
F1	1.64	-1.02	**6.22**	-1.36	2.54
F2	**14.22**	-5.25	8.89	-4.49	-10.31
F3	6.25	**13.87**	-7.46	6.91	-1.65
F4	-56.21	-13.24	3.87	**36.52**	33.55
F5	30.32	**38.09**	21.59	20.01	35.42
F6	8.43	7.02	**11.78**	2.78	4.48
F7	4.55	-1.13	2.93	6.39	**14.97**
F8	14.65	4.97	1.18	2.55	**19.48**
F9	14.82	**17.98**	-6.13	11.48	2.95
F10	9.01	**11.55**	11.35	8.39	9.60
F11	-4.95	**18.29**	-2.91	-10.50	14.00
F12	13.16	13.16	18.72	**30.45**	10.07
F13	16.75	-7.06	1.74	**21.02**	3.10
F14	13.21	**22.71**	21.85	0.98	5.33
Average	6.13	8.57	6.69	9.37	**10.25**

Note: The best performance in every forecast period is in bold. The units in the table are all %.

In Table 3.8 and Table 3.9, the best performing model in each forecast period is in bold. Considering the FTSE100 and INDU together (in total 28 forecasts), Buy-and-Hold wins 2 times, MLP wins 8 times, MLP with AR & MA wins 3 times, CNN wins 6 times and LSTM wins 9 times. It seems that no NN is significantly better than the other models at a 90% confidence level according to the chi-square test (see Appendix B.4).

In respect of the average annualized returns, higher average forecasting accuracy contributes to higher average returns. CNN and LSTM perform significantly better than MLP, and LSTM has the highest average annualized return. During the tests, in the fully connected part of MLP and CNN, if the number of neurons is too small, the absolute value of some final forecasting return is the same as the absolute value of Buy-and-Hold return. This means the forecast results are all in the same category (i.e. returns are all positive or all negative). This should be considered as an invalid forecast that falls into local optimum. Thus, this chapter uses more neurons per layer compared to the literature of Kimoto et al. (1990), Baba & Kozaki (1992) and Yoon & Swales (1991).

Table 3.10: The summary of out-of-sample trading performances for the FTSE100

	Buy-and-Hold	Neural networks			Forecast combination		
		MLP	CNN	LSTM	Simple average	GRR	LASSO
Annualized return	3.30%	8.82%	11.32%	12.49%	11.29%	11.92%	11.96%
Annualized volatility	17.36%	7.65%	15.50%	11.11%	10.02%	9.86%	9.83%
Sharpe Ratio	0.19	1.15	0.73	1.12	1.13	1.21	1.22
Information Ratio		0.72	0.52	0.83	0.80	0.87	0.88

Note: MLP, CNN and LSTM belong to Neural networks; Simple average, GRR and LASSO belong to Forecast combination. Transaction costs have not been taken into account.

Table 3.11: The summary of out-of-sample trading performances for the INDU

	Buy-and-Hold	Neural networks			Forecast combination		
		MLP	CNN	LSTM	Simple average	GRR	LASSO
Annualized return	6.13%	8.57%	9.37%	10.25%	8.02%	9.96%	9.74%
Annualized volatility	19.00%	13.16%	12.85%	12.20%	12.32%	11.92%	11.90%
Sharpe Ratio	0.32	0.65	0.73	0.84	0.65	0.84	0.82
Information Ratio		0.19	0.25	0.34	0.15	0.32	0.30

Note: MLP, CNN and LSTM belong to Neural networks; Simple average, GRR and LASSO belong to forecast combination. Transaction costs have not been taken into account.

The summary of trading performances of individual NNs and combination techniques, based on binary classification categories, is presented in Table 3.10 (for the FTSE100) and Table 3.11 (for the INDU). The calculation methods of the Sharpe Ratio and the Information Ratio are explained in Appendix A.3 and Appendix B.5. In terms of the Sharpe Ratio, Buy-and-Hold is significantly worse than all other forecast models, which proves that all NNs and combination models used in this chapter are more effective than Buy-and-Hold. For the average results in 14 trading years, Buy-and-Hold also has the lowest average annualized return and the highest annualized volatility. This proves that in the long term, transactions based on NNs forecasts not only increase the average rate of return but also reduce the risk. With regard to the Sharpe Ratio and the Information Ratio, LSTM performs better than MLP, CNN and Buy-and-Hold, while the performances of three combination techniques are similar. NNs and combination techniques perform better for the FTSE100 than for the INDU in average return, the Sharpe Ratio and the Information Ratio.

3.6.2 Trading performance using leverage based on the daily forecast probability

The MLP, CNN and LSTM models provide probabilities of signs of daily returns. Based on the probabilistic results, this study assigns different leverage ratios for different probabilities. The higher leverage is set for days with higher forecasting probabilities. In terms of forecasting probabilities that are close to 50%, the corresponding leverage is low or even zero. This method can reduce failed trading days and reduce transaction times.

The threshold value of leverage is determined by the probability distribution of all out-of-sample results. Figure 3.12 shows the distribution of results for MLP, CNN and LSTM. The horizontal axis indicates the interval of the forecast results, where every one percentage point has a bar. The vertical axis shows the height of bars, which describes the number of occurrences. The MLP forecast results are clustered at intervals 48%-49% and 49%-50% (the probability that the next day's return is positive), which are both close to 1500 times in a total of 7056 forecasting times. The distributions of the results for CNN and LSTM are closer to normal distribution compared with the results for MLP.

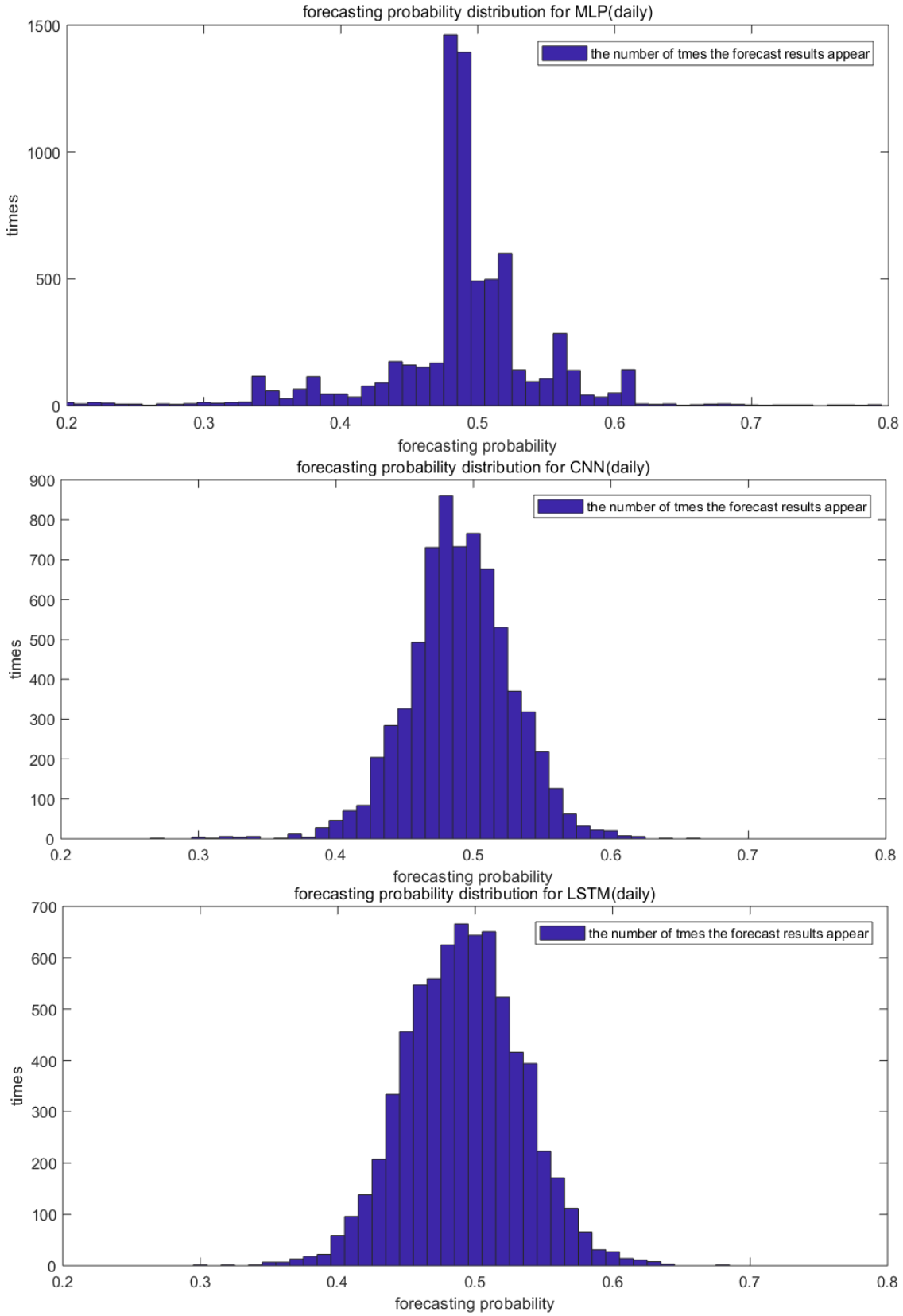
Table 3.12: Summary of daily forecasting probability

	MLP	CNN	LSTM
Mean	0.4863	0.4903	0.4915
Standard deviation	0.0669	0.0381	0.0421
Skewness	-0.9045	-0.2363	0.0082
Kurtosis	8.9999	4.5086	3.3652
Kolmogorov-Smirnov test	0.0000***	0.0000***	0.4783

Note: *** denotes that the hypothesis of the Kolmogorov-Smirnov Test is rejected at the 1% significance level.

Table 3.12 indicates that according to the Kolmogorov-Smirnov Test, the forecasting results for MLP and CNN do not obey normal distribution, and the results for LSTM are normally distributed. The Kolmogorov-Smirnov Test for CNN yields results that are not identical to my intuitive perception, and thus this section analyzes the data further.

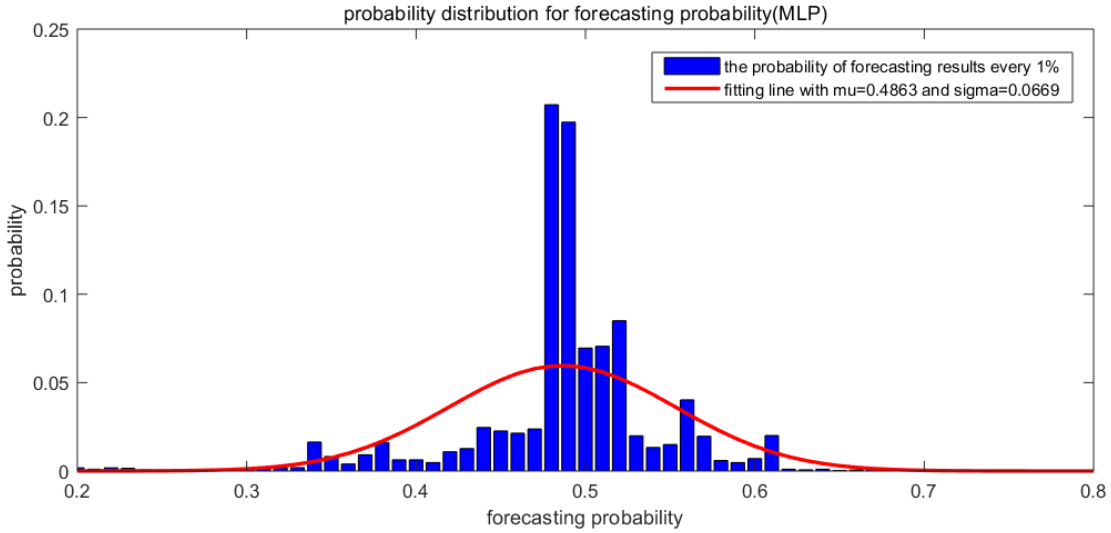
Figure 3.12: Forecasting probability distribution for MLP, CNN and LSTM (daily)



Note: This figure displays the forecasting probability distributions for MLP, CNN and LSTM (from top to bottom in the figure). The width of each bar is 1%. The height of bars represents the number of times the forecasting result falls within the probability interval.

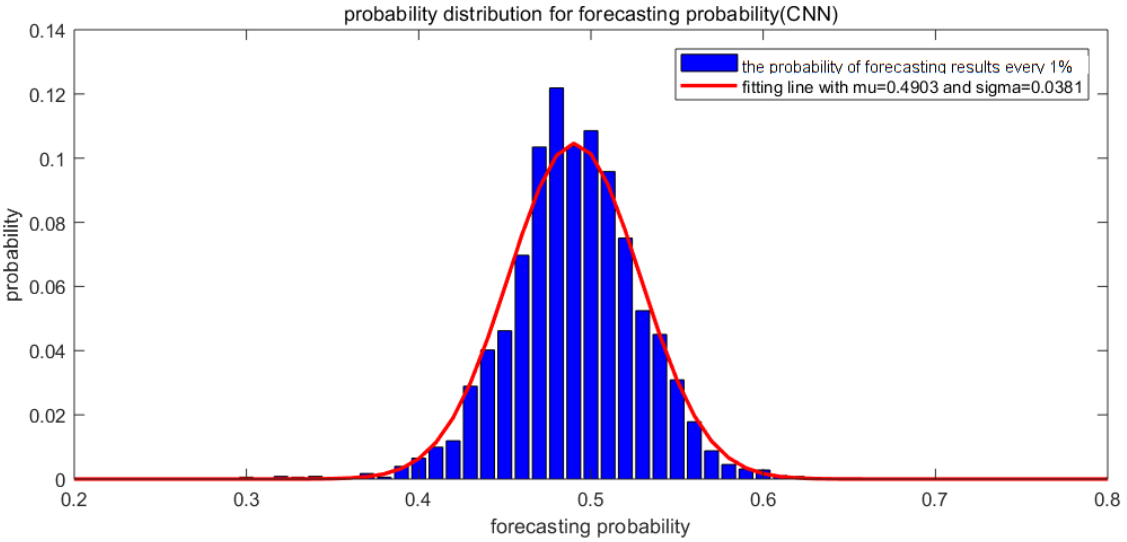
Figures 3.13, 3.14 and 3.15 use the normal distribution function curve to fit the forecast results of NNs. The fitting shows that the forecasting results for CNN and LSTM fit the normal distribution function. In Appendix B.6, the normplot function in MATLAB is used to further analyze the distribution of NNs forecasting results, finding that CNN and LSTM can be regarded as normal distributions. Thus, this study considers that both CNN and LSTM are normally distributed. In addition, the MLP forecasting results are not leveraged because they are very unstable, and most of the values fall within the interval close to 50% with a relatively large standard deviation. This will result in few transactions in some years.

Figure 3.13: Fitting the forecasting results with Normal distribution function (MLP)



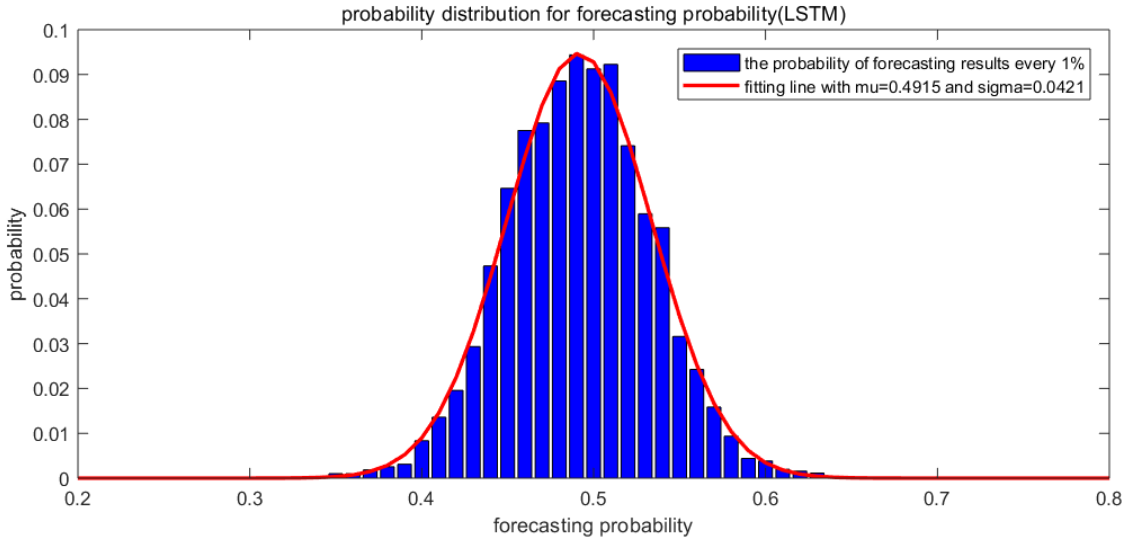
Note: The fitting line is a normal distribution function. Its mean (μ) and standard deviation (σ) are reported in the figure. The bars are the distribution of forecasting results. Whether forecasting results are close to the normal distribution can be observed from the fit of line and bars.

Figure 3.14: Fitting the forecasting results with Normal distribution function (CNN)



Note: The fitting line is a normal distribution function. Its mean (μ) and standard deviation (σ) are reported in the figure. The bars are the distribution of forecasting results. Whether forecasting results are close to the normal distribution can be observed from the fit of line and bars.

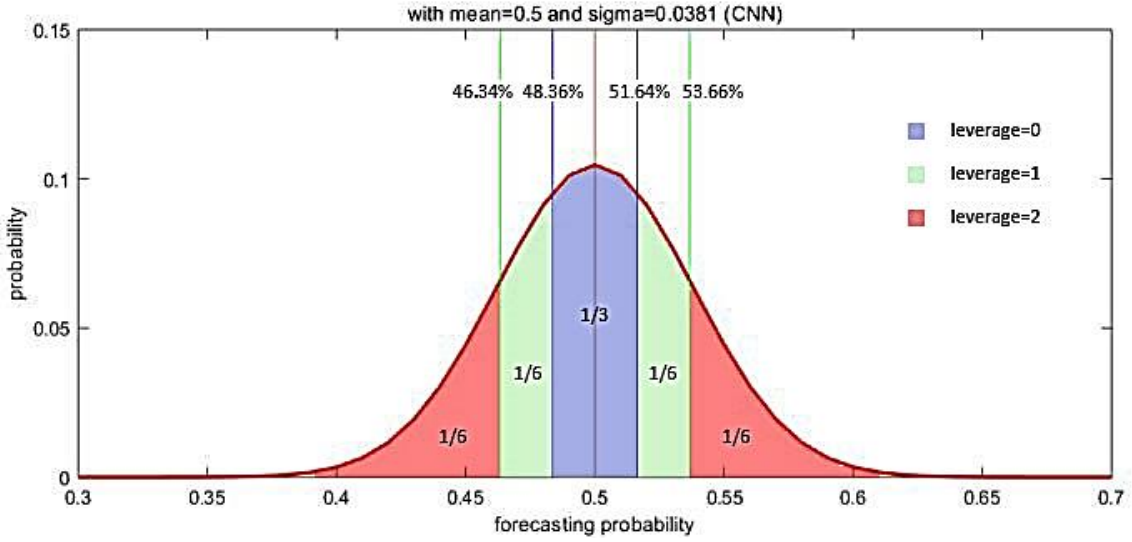
Figure 3.15: Fitting the forecasting results with Normal distribution function (LSTM)



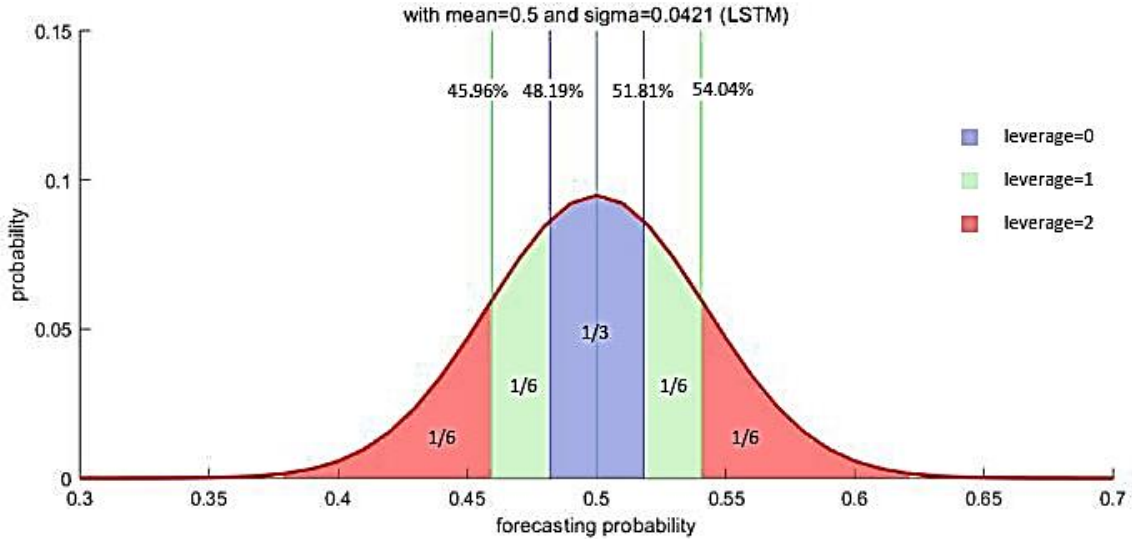
Note: The fitting line is a normal distribution function. Its mean (μ) and standard deviation (σ) are reported in the figure. The bars are the distribution of forecasting results. Whether forecasting results are close to the normal distribution can be observed from the fit of line and bars.

The threshold of the leverage in this study is determined by the standard deviation of the distribution. The mean is not 50%, which is mainly caused by the left fat-tail of the sample, which is explained in detail in Appendix B.6. In order to maintain the fairness of the positive and negative forecasts, this study uses 50% as the mean and regards the standard deviation of the normal distribution as the leverage criteria. As shown in Figure 3.16, the 1/3 proportion close to the 50% (the blue area in Figure 3.16) is assigned leverage 0; the green part is endowed with leverage 1; leverage 2 is allocated to the red part, which means a higher forecasting success rate.

Figure 3.16: Leverage rules for CNN and LSTM



(1) Leverage rules for CNN



(2) Leverage rules for LSTM

Note: This figure consists of leverage rules for CNN and LSTM. According to the forecasting probability, the results are divided into three parts: blue, green and red areas. Each part is assigned a leverage ratio.

Based on the leverage allocation mentioned above, this study calculates the annualized returns of CNN and LSTM. In Tables 3.13 and 3.14, the average annualized returns of CNN and LSTM increase after using leverage both in the FTSE100 and INDU. Leverage trading improves the trading performance by an average of 1-2 percentage points per year. However, leverage trading does not beat non-leverage trading in every trading year.

Table 3.13: Trading performances of CNN and LSTM with leverage for the FTSE100 (%)

	Buy-and-Hold	CNN	LSTM	CNN with leverage	LSTM with leverage
F1	**14.98**	5.30	6.92	7.54	10.27
F2	10.74	6.98	16.01	**26.73**	17.36
F3	0.88	**31.59**	11.26	14.47	24.31
F4	-48.84	26.30	**27.95**	24.51	18.68
F5	30.09	**40.63**	12.74	15.58	23.56
F6	**7.56**	-4.49	-3.37	-15.32	-5.28
F7	-3.23	-16.61	**24.96**	-24.87	11.00
F8	6.09	**11.86**	-15.95	3.16	-10.14
F9	13.14	3.29	10.46	**33.53**	7.93
F10	3.11	11.45	15.75	**28.02**	5.24
F11	-13.37	-0.46	11.97	5.23	**14.85**
F12	14.56	30.33	24.93	6.40	**30.77**
F13	7.08	-3.00	20.01	22.88	**33.16**
F14	3.45	15.24	11.30	**32.48**	25.94
Average	3.30	11.32	12.49	12.88	**14.83**

Note: The best performance in every forecast period is in bold.

Table 3.14: Trading performances of CNN and LSTM with leverage for the INDU (%)

	Buy-and-Hold	CNN	LSTM	CNN with leverage	LSTM with leverage
F1	1.64	-1.36	2.54	**10.82**	10.18
F2	**14.22**	-4.49	-10.31	-11.43	-15.56
F3	6.25	6.91	-1.65	**14.34**	-14.24
F4	-56.21	**36.52**	33.55	35.97	35.63
F5	30.32	20.01	**35.42**	23.61	16.35
F6	8.43	2.78	4.48	3.92	**21.05**
F7	4.55	6.39	14.97	20.99	**27.23**
F8	14.65	2.55	**19.48**	-4.62	2.34
F9	14.82	11.48	2.95	**20.57**	14.22
F10	9.01	8.39	9.60	6.40	**11.42**
F11	-4.95	-10.50	14.00	-15.81	**25.15**
F12	13.16	**30.45**	10.07	28.88	1.14
F13	16.75	**21.02**	3.10	12.23	16.01
F14	13.21	0.98	5.33	13.15	**19.87**
Average	6.13	9.37	10.25	11.36	**12.20**

Note: The best performance in every forecast period is in bold. The units in the table are %.

In Tables 3.15 and 3.16, the volatility of annualized returns after using leverage has increased, which is due to the decline in the number of transactions per year. Although the volatility has increased, the annualized average return has increased more, resulting in an increase in both the Sharpe Ratio and the Information Ratio. For the Sharpe Ratio of the FTSE100, leverage trading helps the CNN and LSTM increase from 0.73 to 0.77 and from 1.12 to 1.21 respectively. For the Sharpe Ratio of the INDU, leverage trading helps the CNN and LSTM increase from 0.73 to 0.80 and from 0.84 to 0.86 respectively. The results demonstrate that assigning different leverage ratios based on different forecasting probabilities can improve the trading performance.

Table 3.15: The summary of out-of-sample trading performances with leverage for the FTSE100

	Buy-and-Hold	Neural networks			Neural networks with leverage	
		MLP	CNN	LSTM	CNN	LSTM
Annualized return	3.30%	8.82%	11.32%	12.49%	12.88%	14.83%
Annualized volatility	17.36%	7.65%	15.50%	11.11%	16.73%	12.26%
Sharpe Ratio	0.19	1.15	0.73	1.12	0.77	1.21
Information Ratio		0.72	0.52	0.83	0.57	0.94

Note: Trading performances of CNN and LSTM with leverage ratios are presented. Transaction costs are not taken into account.

Table 3.16: The summary of out-of-sample trading performances with leverage for the INDU

	Buy-and-Hold	Neural networks			Neural networks with leverage	
		MLP	CNN	LSTM	CNN	LSTM
Annualized return	6.13%	8.57%	9.37%	10.25%	11.36%	12.20%
Annualized volatility	19.00%	13.16%	12.85%	12.20%	14.26%	14.17%
Sharpe Ratio	0.32	0.65	0.73	0.84	0.80	0.86
Information Ratio		0.19	0.25	0.34	0.37	0.43

Note: Trading performances of CNN and LSTM with leverage ratios are presented. Transaction costs are not taken into account.

3.7 Conclusions

In this chapter, MLP, CNN and LSTM are applied to forecast the daily return of the FTSE100 and INDU, and three combination methods are used to combine NNs forecasts. First, this chapter uses MLP, CNN and LSTM with the same inputs pool and compares the statistical accuracy and trading performances of the NNs with a naive trading strategy. Second, the simple average, GRR and LASSO are employed to combine the forecasting results of three NNs. Finally, this chapter applies leverage based on the daily forecast probabilities for trading. The models are trained with data of the first five years, and data of the sixth year is used for the out-of-sample test. The time span of the out-of-sample test is 2004-2018, and the start date of each test is one trading year apart (252 trading days). This chapter conducts 14 tests each with the FTSE100 and INDU.

In respect of the forecasting results, trading performances and average accuracy, MLP, CNN and LSTM are proved to provide effective forecasts. Of these models, the average performance of CNN is better than that of MLP, and the LSTM slightly outperforms the CNN. After analyzing the annualized trading performance, this research finds that due to insufficient trials, it is not possible to determine which of the NNs is significantly better than the other two. This study also tests the impact of different inputs on MLP's performance and

finds that it declines with more duplicate information. This proves that the performances of NNs are significantly affected by the quality of inputs, and thus the pre-processing of inputs is important in the NNs model when forecasting stock indices.

Additionally, the combination techniques used in this study do not help NNs improve performances because NNs are used as classifiers. The shortcoming of the naïve trading strategy used here weakens the performance of combination techniques. Leverage trading based on the daily forecast probability helps the average annualized return increase, but the volatility also increases at the same time. In general, the Sharpe Ratio and Information Ratio increase both for the INDU and FTSE100. Thus, the results demonstrate that leverage trading based on forecasting probabilities can improve the trading performance.

This chapter contributes to the field of financial time series forecasts in three aspects. First, it improves the structures of MLP, CNN and LSTM for stock indices forecasting. Compared to MLP used in prior literature, this chapter adopts a more complex neural network by using five hidden layers with more neurons in each layer. Increasing the complexity of neural networks is prone to overfitting, and thus this study employs the Dropout technique to avoid overfitting. This chapter also designs the structure of CNN based on stock indices data. Valid Padding is used to reduce the data size instead of using the pooling layer. The Dropout technique is used for the fully connected layer in CNN. The empirical results confirm the superiority of MLP, CNN and LSTM to Buy-and-Hold, while LSTM slightly outperforms CNN and MLP. Second, it provides evidence that the pre-processing of inputs has an impact on the performances of NNs. This chapter uses different inputs for MLP, finding that more inputs do not lead to better performance. Third, leverage trading is applied based on the daily forecast probabilities given by NNs, which improves the trading performance.

Three improvements can be made based on the results. First, inputs with more features and more lags can be used to further improve the performances of LSTM and CNN. Second, the pre-processing method of the inputs needs to be improved. Third, leverage can be further optimized. This study separates the forecast results into three parts and assigns different leverages, but if continuous leverage is used, the Sharpe Ratio will be higher. Further research can be conducted to solve the above problems.

Chapter 4 Deep Reinforcement Learning and Genetic Algorithm for a Pairs Trading Task on commodities

4.1 Introduction

Pairs trading is a quantitative method of speculation which originated on Wall Street. A great deal of literature uses this method to backtest various assets, especially in stock trading. The aim of pairs trading is to find two assets whose prices fluctuate together. When the spread between them widens, investors short the winner and buy the loser. If their prices converge, investors earn excess returns. Studying whether this simple strategy based on past price dynamics and contrarian investing can arbitrage in the commodity market in recent years, Gatev, Goetzmann & Rouwenhorst (1999) claim that if the market is efficient, positive risk-adjusted returns from pairs trading are not possible. However, the traditional pairs trading models that gain no positive excess returns do not prove that the market is efficient. Pairs trading with a more reasonable pairs-selection method and flexible trading actions instead of traditional methods can achieve positive returns in the market.

This chapter proposes a novel pairs trading strategy that combines the CA and DRL, which is denoted as CA-DRL. CA-DRL uses CA to form pairs. Then the DRL structure that is designed for dealing with a large amount of pairs trading data is used to learn the in-sample environment. After that, the trained DRL model makes the trading decisions in the out-of-sample. During the testing period of 1980-2018 in commodity markets, the average annualized return in the out-of-sample of CA-DRL reaches 12.49%, which beats the traditional methods and CA-GA-ST method. The traditional methods, the benchmark models in this chapter, use the DIM and CA to form pairs and the ST strategy to trade. These benchmark models include DIM-ST, CA-ST and CA-DIM-ST. Additionally, the CA-GA-ST method also proposed here uses CA to form pairs and then employs the GA to optimize the parameters of ST strategy.

In the tests, the in-sample excess profits of these traditional methods are still significant, but their out-of-sample excess profits are close to zero, which means these traditional strategies are ineffective in commodity markets during the sample period. Moreover, this chapter uses

GA to optimize the parameters in the ST method. The in-sample performance is significantly improved, but the out-of-sample improvements are not significant, with an average out-of-sample annualized return of 1.84%. Regarding the risks involved in the methods this chapter employs, the Sharpe Ratio, Sortino Ratio, Value at Risk (VaR), Morningstar Risk-Adjusted Return (MRAR) and the Maximum Drawdown of the portfolios of CA-ST, CA-GA-ST and CA-DRL strategies are analyzed. CA-DRL performs better than other models in all estimations, while the performance of the Maximum Drawdown of CA-DRL is similar to that of CA-ST and CA-GA-ST.

One of the aims of this chapter is to verify the effectiveness of traditional pairs trading methods in commodity markets in recent years. The second aim is to propose new methods of improving traditional pairs trading methods from two perspectives: one is to optimize the parameters in traditional methods, and the other is to use the machine learning method to make trading decisions. The third aim is to solve the problem of excessive computation when the machine learning method handles large batches of pairs.

This chapter makes three main contributions. First, the use of novel DRL for decision-making of trading actions is successful in terms of both returns and risks. Second, this chapter solves two problems with DRL when it is used for a large number of samples; that is, it avoids falling into the local optimum and reduces the calculation amount. Solving these two problems enables DRL to deal with a large number of pairs to form portfolios. Third, this chapter employs GA to optimize the parameters in ST and slightly improves the performance.

Chapter 4 is organized as follows. The literature review about pairs trading and DRL is in section 4.2. The dataset used in this chapter is introduced in section 4.3 and the traditional benchmark strategy, CA-GA-ST method and CA-DRL model are examined in section 4.4. Then, the comparison and explanation of statistical and trading performances for all strategies are presented in section 4.5. Section 4.6 evaluates the robustness and risks of the strategies. The conclusion and further developments are presented in section 4.7.

4.2 Literature review

4.2.1 Pairs trading

Pairs trading is a neutral trading strategy, which is identified as a statistical arbitrage and convergence trading strategy (Kanamura, Rachev & Fabozzi, 2009). Pairs trading was introduced by Gerry Bamberger and then developed by Morgan Stanley in the 1980s. Since then, scholars have proposed different methods for optimizing pairs trading. The main methods include the Distance Method (Gatev, Goetzmann & Rouwenhorst, 1999; Perlin, 2009; Broussard & Vaihekoski, 2012; Jacobs & Weber, 2015), the Co-integration Approach (Vidyamurthy, 2004), Machine Learning (Huck, 2009; 2010) and Stochastic Methods (Jurek & Yang, 2007; Liu & Timmermann, 2013). The summary of the literature about authors, methods, data and main results is presented in Appendix C.1.

4.2.1.1 The Distance Method in pairs trading

The traditional DIM has been widely tested in pairs trading literature. Gatev, Goetzmann & Rouwenhorst (2006) propose it and use it to examine the risk and return of pairs trading over the period 1962–2002. They use stocks from the Center for Research in Security Prices (CRSP) daily files. They construct a cumulative total return index for each stock, and then stocks are matched by finding the minimum sum of squared deviations between two normalized price series. Their pairs trading strategy of the fully invested portfolio of the top five pairs yields a mean monthly excess return of 1.31%, before transaction costs, and 1.44% per month for a portfolio of the top 20 pairs. The portfolios yield annualized excess returns of about 11% for top pairs, which are robust to conservative transaction-cost estimates. They show that the profits of pairs trading differ from the profits by simple mean reversion in the previous literature.

Based on this work, Broussard & Vaihekoski (2012) test the DIM, using data from the Finnish stock market over the period 1987–2008. The annualized return of their strategy is as high as 12.5%. Additionally, they find that it is necessary to use lower thresholds to raise returns, even after taking trading costs into account (low thresholds will result in a high frequency of trading), suggesting that a more optimal trade initiation threshold may be available. They state that their pairs trading strategy generates positive alpha in their sample

period. They conclude that pairs trading is a safe strategy, insignificantly affected by market risk.

Regarding stock markets in developing countries, Perlin (2009) uses the daily, weekly and monthly prices of the Brazilian financial market to examine the performance of the pairs trading strategy, finding that it provides profits and is a market-neutral strategy. He adjusts the parameters in the method, such as the level of the thresholds in ST and the frequency of trading, observing that these parameters have a great impact on earnings. Daily frequency has the best results compared to weekly and monthly frequency. However, their final result uses the in-sample information, which causes the rate of return to be excessively high and hence invalid.

Jacobs & Weber (2015) use the DIM to carry out a large-scale test of pairs trading on 34 international stock markets, finding that pairs trading strategy is persistently profitable, whereas the returns are not stable over time. In particular, they analyze data for NYSE and AMEX stocks from 1960 to 2008 and indicate that the strategy's time-varying profitability results mainly from investors' under- or over-reaction to news. The over-reactions lead to biased estimates of prices, which provide opportunities for pairs trading strategies to be profitable.

4.2.1.2 The Co-integration Approach in pairs trading

The Co-integration Approach is a commonly used method of pairs formation. The forecasts of individual asset prices are generally acknowledged to be challenging, while the value of the portfolio formed by certain assets is easier to be forecasted. Assuming that there is a co-integration relationship between two assets, when the co-integration irregularity appears, the prices of the two assets are expected to return to the co-integration relationship. Thus, a pairs trading strategy can be built based on forecasting in contrary to irregularity (Alexander, 2001).

The typical work on co-integration-based pairs trading is by Vidyamurthy (2004), who presents a theoretical framework using co-integration for pairs trading. The stock pairs are selected based on statistical or fundamental similarity measures and there is a possibility that they are co-integrated. Vidyamurthy uses the Engle-Granger two-step approach to test for

co-integration. Vidyamurthy's trading rule is the ST method, which is the same as that of Gatev, Goetzmann & Rouwenhorst (1999). In addition, he provides the optimization method of thresholds. The optimization target (the profit for each threshold level) is the absolute value of the threshold level multiplied by the number of occurrences. However, this is not equivalent to total profit. Owing to this problem, these results are criticized by Krauss (2017). Although Vidyamurthy's calculation method greatly simplifies the calculation amount of total profit, that method is meaningless. This study resolves this issue using GA.

Li (2014) uses CA to analyze 38 dual-listed companies in China A-share and Hong Kong H-share, that is, two stocks of the same company in different markets forming a pair. The simple thresholds, as used by Gatev, Goetzmann & Rouwenhorst (2006), are then adopted to make trading decisions. Their result is very successful, with an average annualized return of 17.6%. However, since the two stock prices are close, the trading frequency is high. Owing to the high trading frequency and market commissions, the annualized return falls to 10.8% once transaction costs have been taken into account. The CA-ST strategy has also been applied to the Brazilian stock market (Caldeira & Moura, 2013), and the empirical results are good over the sample period of 2005-2012, with an average annualized return of 16.38%, Sharpe Ratio of 1.34 and low correlation with the market.

Some literature provides evidence that CA outperforms other methods in pairs formation. Bogomolov (2011) explores DIM-ST, CA-ST and the Stochastic spread method on the Australian share market over the period 1996-2010, finding that these three pairs trading strategies are profitable before transaction costs, and CA-ST reaches the highest average monthly return of 1.05%. However, profits are considerably diminished once transaction costs and liquidity issues have been taken into account.

Additionally, Huck & Afawubo (2015) use the data of S&P 500 component stocks over 2000-2011 to verify the DIM-ST method of Gatev, Goetzmann & Rouwenhorst (2006) and the CA-ST method of Vidyamurthy (2004), finding that the pairs selected by CA perform better. Rad, Low & Faff (2016) investigate DIM, CA, and copula methods for pairs trading on the US equity market over the period 1962-2014. These methods exhibit monthly excess return of 91, 85 and 43 bps before transaction costs, respectively. The copula method retains its frequency of trading opportunities from 2009, whereas the frequency of trading is decreased considerably for DIM and CA. They conclude that CA is a superior method of

trading in turbulent market conditions.

The improvement of pairs formation based on CA can generate a better trading performance. Lin et al. (2006) improve CA by introducing a co-integration coefficient weighting rule to the pair formation process. They also optimize the trading thresholds according to the conditions necessary to guarantee a minimum profit for each trade. They select two Australian Stock Exchange quoted bank shares from January 2001 to August 2002 as a test set, where the data of the year 2001 forms the in-sample and data of the second half-year is used as the out-of-sample. They find that, given a reasonable minimum profit level, the number of transactions or the total profit does not decrease exceedingly, compared to the strategy without the minimum profit constraint. However, they only use one pair during 2001-2002 to run the test, and this small dataset is not sufficiently convincing. They do not take special circumstances into account, and thus the risks of their strategy cannot be evaluated.

4.2.1.3 The machine learning method and pairs trading

There is limited literature on pairs trading frameworks that incorporate machine learning methods. Huck (2009) introduces a method based on the forecasting of Elman networks and the ranking of ELECTRE III for pairs selection. He uses Elman networks to forecast the returns of all stocks, and then ranks them according to the rules of ELECTRE III, buys high-ranked stocks and sells low-ranked stocks to form pairs. He conducts experiments on S&P 100 constituents from 1992 to 2006. The results are very successful, with the excess return surpassing 0.6% per week. Huck (2010) improves the previous model and employs it in S&P 100 constituents again. Huang et al. (2015) use GA to optimize the pairs formed by 10 stocks on the Taiwan stock market from 2003 to 2012. The optimization targets are the weights of stocks and trading thresholds. Their model beats the Buy-and-Hold method. However, their model requires a large amount of computation and is only suitable for a small number of assets for pairs trading.

Elliott, Van Der Hoek & Malcolm (2005) propose a mean-reverting Gaussian Markov chain model that simulates the spread movement of the price of two stocks. They compare the simulation results with subsequent observations of the spread, to make trading decisions. They only prove that their model can potentially be applied in the financial markets which

are observed to be out of equilibrium. The weakness of their results is that they do not test real market data.

Avellaneda & Lee (2010) use PCA to create a seemingly mean-reverting time series, which is similar to Elliott et al. (2005). Their trading signals are produced in two methods, including using PCA or regressing returns of stocks on sector Exchange Traded Funds (ETFs). The returns are modelled as mean-reverting processes in both methods. After deducing transaction costs, the average annualized Sharpe Ratio of PCA-based strategies is 1.44 and the average annualized Sharpe Ratio of ETF-based strategies is 1.1 throughout the period 1997-2007. They find that the Sharpe Ratio of both methods decreases over time. They then propose a new trading signal that uses trading volume to improve the performance of ETF-based signals. Their results show that the Sharpe Ratio of ETF-based strategies with trading volume information reaches 1.51 during the period 2003-2007.

4.2.1.4 Other methods

In addition to the DIM, CA and machine learning methods, the stochastic method and copula approach are also used in some studies. Do, Faff & Hammza (2006) analyze three pairs trading methods: DIM, CA and the stochastic spread method. They then introduce a general approach to build the pairs trading model in view of asset pricing theory, which they call the stochastic residual spread model. The selected pairs show clear mean reversion behaviour in their relative pricing, which proves that there is arbitrage in the pairs they choose. In their later research, Do & Faff (2010) verify their method using data from the US stock market over the period 1962-2009, finding that pairs trading profitability decreases over time. The average monthly return reduces from 0.86% in 1962-1988 to 0.37% in 1989-2002 and then 0.24% in 2003-2009. They also find risks have grown over time. In addition, they propose alternative algorithms to improve the performance of pairs trading by adding two new messages to the rules for pairs selection: industry homogeneity and historical frequency of reversal in the price spread. In this way, their pairs trading is still profitable, albeit at a relatively low level.

Mudchanatongsuk, Primbs & Wong (2008) propose a stochastic control approach for pairs trading. They assume log-relationship between a pair of stock prices as an Ornstein-Uhlenbeck process, and use this to formulate a portfolio. The Ornstein-Uhlenbeck model is

used to simulate the movement of a pair of stock prices as a Gaussian Markov chain model. Based on the Ornstein-Uhlenbeck process, Mudchanatongsuk, Primbs & Wong obtain the optimal solution to this control problem in the closed form via the corresponding Hamilton-Jacobi-Bellman equation. Similar research is carried out by Ekström, Lindberg & Tysk (2011), Herlemont (2003) and Bogomolov (2013).

The copula approach is a method similar to CA. Copulas are suitable for modelling joint distributions and dependence between assets to determine whether there is a continuous and stable relationship between two assets, and this method has also been employed in a few publications on pairs trading. Liew & Wu (2013) for example use the copula method and compare it with the CA for pairs trading. They find that the copula approach offers more trading opportunities with greater confidence relative to CA and DIM. However, they only provide the empirical results of one pair, and that pair has a co-integration relationship both in the in-sample and the out-of-sample, which makes their results statistically unreliable. Additionally, Krauss & Stübinger (2017) propose a copula-based pairs trading framework to investigate the S&P 100 constituents over the period 1990-2014. They sort the selected pairs according to mean-reversion and momentum, select top 5, top 10 and top 20 for out-of-sample trading, and compare them with the naive S&P 100 Buy-and-Hold strategy. The results show that top 5 has the best out-of-sample performance with an average out-of-sample return of 7.98% per year for the top 5 mean-reversion pairs and 7.22% per year for the top 5 momentum pairs.

4.2.2 Deep Reinforcement Learning

The combination of the advances in deep learning for learning feature representations and Reinforcement Learning (RL) is a significant development (Krizhevsky et al., 2012; Hinton et al., 2012), tracing back to the much earlier work of Tesauro (1995) and Bertsekas & Tsitsiklis (1995). The impressive application of DRL is to execute a sequence of actions in different environments. Guo et al. (2014) use DRL to train agents to play Atari games based on raw pixels and to acquire advanced manipulation skills with raw sensory inputs. Similar research in recent years has been done by Mnih et al. (2015), Levine et al. (2016) and Watter et al. (2015). The performance of DRL in 3D locomotion and manipulation tasks is significant (Schulman et al., 2015; Lillicrap et al., 2015). The trading process can be considered as a game and market information can be considered as the environment, and

hence the success of DRL in the games field indicates that there will be a promising application of DRL in trades.

Recently, a small amount of literature has employed DRL to finance. Deng et al. (2016) introduce contemporary deep learning into a typical DRL framework for financial signal processing and online trading. Their results on both the stock-index and commodity futures contracts demonstrate the effectiveness of the learning system in simultaneous market condition summarization and optimal action learning. However, their models only handle one share. Additionally, Buehler et al. (2019) present a framework for hedging a portfolio of derivatives in the presence of market frictions such as transaction costs, liquidity constraints or risk limits using DRL. They illustrate their approach by an experiment on the S&P500 index and by showing the effect on hedging under transaction costs in a synthetic market driven by the Heston model, where they outperform the standard 'complete-market' solution.

The research group at Cornell University has published a large number of papers in financial trading using machine learning methods, which include the DRL (Xiong et al., 2018). Specifically, they use DRL to optimize the portfolio that consists of 30 stocks from the DJIA. Xiong et al. apply the Deep Deterministic Policy Gradient (DDPG) algorithm (an algorithm in DRL) to determine the weights of the stocks in the portfolio with the aim of a higher return. In the out-of-sample, their strategy decided by DRL beats the DJIA index and traditional min-variance portfolio allocation strategy in cumulative return and Sharpe Ratio. Similarly, Liang et al. (2018) try to use DDPG, Policy Gradient (PG) and Proximal Policy Optimization (PPO) to make the decisions (buy or sell) for stocks on the China stock market. Their results show that PG outperforms DDPG and PPO. Liang et al. then introduce a new training process for PG known as the Adversarial Training method and show that it can improve training efficiency, the average daily return and Sharpe Ratio in the out-of-sample.

Few studies use DRL in decision-making for trading, especially in pairs trading. The major gaps are in using machine learning in pairs trading and optimizing the parameters of traditional pairs trading strategies. This chapter aims to improve the performances of pairs trading strategies by adopting two approaches. The first is to use GA to optimize the parameters of traditional pairs trading strategies. The second is to apply DRL to make decisions in pairs trading. In addition, the model framework of DRL introduced here can be extended to other strategy-based trading methods.

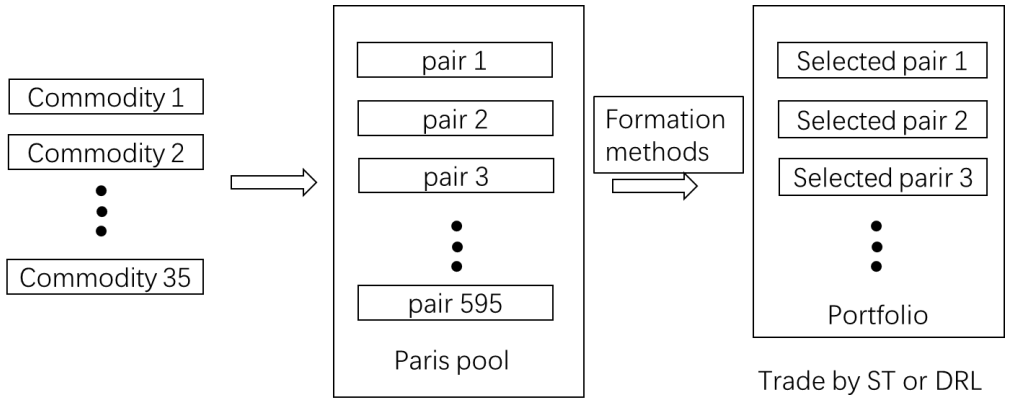
4.3 Dataset

This chapter uses 35 commodities in 9 commodity markets. The list of commodities and their corresponding markets are presented in Appendix C.2. The implementation of pairs trading has two stages. First, a proper method needs to select the pairs from a pairs pool (total 35*34/2=595 pairs per period), and the portfolio consists of the selected pairs (Figure 4.1). This stage is called pairs formation. Second, trading decisions need to be made in those selected pairs. Trading rules can be given directly by ST or DRL. Then, the trading rules are applied to the out-of-sample. The out-of-sample period and the in-sample period in this study are different from the study done by Gatev, Goetzmann & Rouwenhorst (2006), since this research forms pairs over a 24-month period (in-sample period), and trade them in the next 12-month period (out-of-sample period). The data used here are taken from Bloomberg from 1980 to 2018.

Table 4.1 presents the statistical summary of all commodities used in this study. Some agriculture products (cotton, coffee, lumber, lean hogs, orange juice and sugar) are stationary during 1980-2018. The daily returns of all the commodities do not correspond to the normal distribution, the kurtosis of commodities being much higher than that of the normal distribution.

Data preparation is conducted in Python 3.5, relying on the packages Numpy (Van Der Walt, Colbert, & Varoquaux, 2011) and Pandas (McKinney, 2010). The DRL used in this chapter is developed with Keras on the top of Google TensorFlow. GA is calculated by the Torch. The VaR of the portfolio is calculated by MATLAB with Financial Toolbox.

Figure 4.1: Portfolio formation for every year



Note: The number of pairs in the portfolio is different for every year, which depends on the formation methods.

Table 4.1: Data summary of commodities

Commodities	Stationary	ADF (p value)	Std.Dev	Skewness	Kurtosis	Jarque-Bera of return (p value)
Aluminum alloy	0	0.5059	0.0121	-0.1416	12.8233	0.000
Aluminum	0	0.1708	0.0136	-0.2745	7.4679	0.000
Soybean oil	0	0.1852	0.0146	0.0010	6.1648	0.000
Corn	0	0.1138	0.0162	-1.2297	25.6908	0.000
Cocoa	0	0.0452	0.0189	0.0397	5.9167	0.000
Crude oil	0	0.4372	0.0232	-0.6973	17.8454	0.000
Copper	0	0.5716	0.0158	-0.0887	7.6912	0.000
Cotton	1	0.0013	0.0183	-8.1219	377.6215	0.000
Feeder cattle	0	0.6696	0.0093	-0.2109	12.2533	0.000
Gold	0	0.9309	0.0116	-0.0593	11.6327	0.000
Copper	0	0.5908	0.0164	-0.2905	7.7036	0.000
Heating oil	0	0.4428	0.0224	-1.2627	22.2659	0.000
Coffee	1	0.0041	0.0223	0.0431	11.0409	0.000
Wheat	0	0.0501	0.0158	-0.5389	12.8510	0.000
Lumber	1	0.0021	0.0206	0.5070	11.3126	0.000
Live cattle	0	0.3370	0.0111	-1.5245	17.4444	0.000
Lead	0	0.3821	0.0197	-0.1627	9.6522	0.000
Lean hogs	1	0.0025	0.0217	-0.2911	37.2748	0.000
Wheat spring	0	0.0589	0.0153	-0.1750	40.3114	0.000
Natural gas	0	0.0229	0.0334	0.1281	11.5553	0.000
Nickel	0	0.2140	0.0240	-0.4916	20.1650	0.000
Orange juice	1	0.0035	0.0193	0.4931	13.6691	0.000
Palladium	0	0.9992	0.0191	-0.2050	8.9515	0.000
Pork bellies	0	0.0546	0.0225	1.4001	52.2220	0.000
Platinum	0	0.4657	0.0140	-1.0669	23.3215	0.000
Rough rice	0	0.1527	0.0161	0.0814	26.5361	0.000
Canola	0	0.0752	0.0123	-1.0377	19.6641	0.000
Soybean	0	0.1448	0.0148	-0.7901	10.3020	0.000
Sugar	1	0.0316	0.0251	0.1220	13.1169	0.000
Brent oil	0	0.5087	0.0218	-1.0155	24.8093	0.000
Silver	0	0.0144	0.0196	-0.7170	16.0766	0.000
Soybean meal	0	0.0609	0.0170	-1.0099	14.5188	0.000
Tin	0	0.5235	0.0158	-0.0024	10.8515	0.000
Wheat	0	0.0240	0.0181	-0.9215	25.4862	0.000
Zinc	0	0.3934	0.0200	-0.9849	26.5490	0.000

Note: The ADF test is level. If the p value is lower than 1%, the price of the commodity is considered to be stationary. The Std.Dev, Skewness, Kurtosis and Jarque-Bera tests are for the log return of commodities.

4.4 Methodology

4.4.1 Pairs formation methods

4.4.1.1 Distance Method

Regarding DIM, as it is applied to the n assets under consideration, the Sum of Euclidean Squared Distance (SSD) for the price time series of $n(n-1)/2$ possible combinations of pairs is calculated. A certain number of top pairs with minimum SSD histories are considered

in a subsequent out-of-sample trading period. p_i^t and p_j^t denote realizations of the normalized price processes $P_i = (p_i^t)_{t \in T}$ and $P_j = (p_j^t)_{t \in T}$ of the assets i and j of a pair. The sum of Euclidean squared distance (SSD_{P_i,P_j}) is calculated as:

$$SSD_{P_i,P_j} = \sum_{t=1}^{T} \left(p_i^t - p_j^t\right)^2 \tag{4.1}$$

This study calculates the SSD_{P_i,P_j} of every pair, and rank them by SSD_{P_i,P_j} value. The pairs with smaller SSD_{P_i,P_j} are at the top. After this step, the trading opportunities can be found by using the DIM, according to the ranking. For example, the top 10 pairs can be selected to trade in the out-of-sample.

The advantages of the DIM are that it is easy to implement, robust to data snooping, and results in statistically significant risk-adjusted excess returns. However, the choice of Euclidean squared distance as a selection metric is analytically suboptimal. Let us assume that a rational pairs trader has the objective of maximizing excess returns per pair. As such, a pairs trader aims for spreads exhibiting frequent and strong divergences and subsequent convergences to equilibrium. In other words, the profit-maximizing rational investor seeks out pairs with high spread variance and strong mean-reversion properties. However, the 'ideal pair' (best choice) of SSD has a spread of zero and thus produces zero profits. Thus, the DIM's selection metric tends to form pairs with low spread variance and limited profit potential (Krauss, 2017).

4.4.1.2 Co-integration Approach

Compared with the DIM, using the CA to select opportunities makes more sense. CA can select more 'ideal pairs', that with high spread variance and strong mean-reversion. Using co-integration as a theoretical basis, the spread is generated based on the actual error term ε_{ij}^t of the long-term relationship:

$$\varepsilon_{ij}^t = -a_i^t + \gamma a_j^t + C \tag{4.2}$$

Where a_i^t and a_j^t denote the price processes of assets i and j. The co-integration coefficient γ is a non-zero real number, so that the spread ε_{ij}^t as a linear combination of a_i^t and a_j^t is stationary. C is a constant. Next, all pairs are tested with the Engle-Granger approach.

(i) $E(\varepsilon_{ij}^t)$ is independent with time t

(ii) $VAR(\varepsilon_{ij}^t)$ is bigger than zero and independent with t

(iii) $Cov(\varepsilon_{ij}^t, \varepsilon_{ij}^s)$ is correlated with $t - s$

In this way, the opportunities for pairs trading are found by co-integration analysis.

4.4.2 Decision-making methods

4.4.2.1 Simple trading thresholds method

In Gatev, Goetzmann & Rouwenhorst (2006), trades are opened when the spread diverges by more than two historical standard deviations and closed upon mean-reversion, at the end of the trading period, or upon delisting. This study calls this method the Simple Thresholds method, which is generalized and used in many pairs trading strategies (Perlin, 2009; Vidyamurthy, 2004). Empirical spread variance s_{P_i,P_j}^2 can be expressed as

$$s_{P_i,P_j}^2 = \frac{1}{T}\sum_{t=1}^{T}\left(p_i^t - p_j^t\right)^2 - \left(\sum_{t=1}^{T}\left(p_i^t - p_j^t\right)^2\right)^2 \tag{4.3}$$

Where the estimation of coefficient and constant is:

$$\varepsilon_{ij}^t = -a_i^t + \gamma a_j^t + C \tag{4.4}$$

$$\hat{a}_i^t = \gamma \hat{a}_j^t + C \tag{4.5}$$

$$p_i^t = a_i^t \tag{4.6}$$

$$p_j^t = \gamma + a_j^t C \tag{4.7}$$

The standard spread of a_i^t and a_j^t is:

$$sp_{i,j}^t = \frac{p_i^t - p_j^t}{s_{P_i,P_j}^2} \tag{4.8}$$

s_{P_i,P_j}^2 is also the historical standard deviation (standard deviation in the in-sample). The decision of trading actions is made based on $sp_{i,j}^t$.

Table 4.2 shows how trading decisions are made. For example, as shown in Figure 4.2, the horizontal axis is time, and the vertical axis is the spread of two assets with the unit of standard deviation. The trade is open when the spread is higher than 2 standard deviations or lower than -2 standard deviations. The trade is closed when the spread returns to 0. Gatev, Goetzmann & Rouwenhorst (2006) only use the DIM combined with the ST to trade.

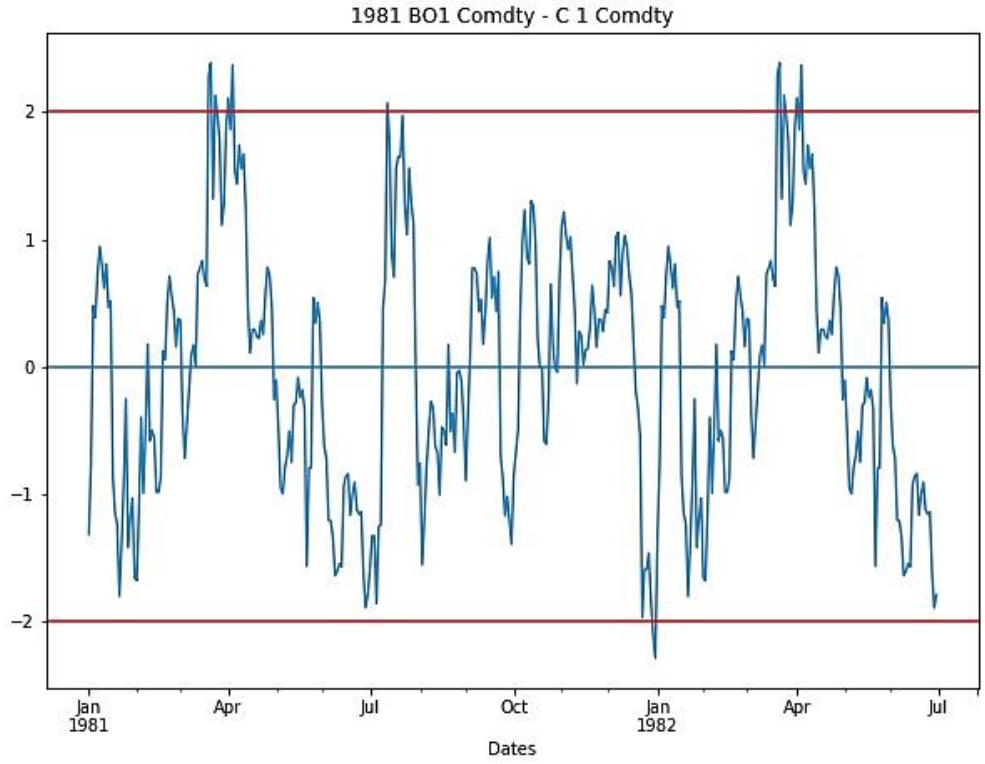
In this study, the ST is combined both with CA and DIM.

Table 4.2: The trading decisions

The distance of two assets	Long position of the distance of two assets	Square position	Short position of the distance of two assets
Higher than 2 historical standard deviation	None	Action (1)	Action (2)
Pass 0	Action (3)	Action (2)	Action (1)
Lower than -2 historical standard deviation	Action (2)	Action (3)	None

Note: The initial position starts with a square position. 'None' means this situation cannot happen. Action (1) is the action to buy a_i with a value of $V/2$, and sell a_j with a value of $V/2$. Action (2) means that there is no transaction. Action (3) is to sell a_i with a value of $V/2$ and buy a_j with a value of $V/2$.

Figure 4.2: A sample of individual pairs to explain the ST



Note: This figure shows the linear combination of two commodities given by the CA. The vertical axis is the standardized deviation.

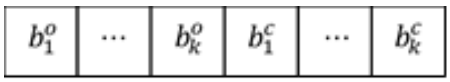
The ST method has four weaknesses. First, it assumes that the distribution of the distance of two assets follows the normal distribution, but this may not be the case. Second, the threshold value (± 2 standard deviations) of decision-making is unreasonable, and a good threshold value may improve the trading performance significantly. Third, the ST method ignores the influence of time-series, by only using the current information (the distance between two assets). Fourth, it is not a continuous method and will lose some profits compared with

continuous methods at the same level of risk.

4.4.2.2 Genetic Algorithm combined with CA and ST for pairs trading

This study uses the CA to pre-select m pairs trading opportunities from n assets with $n(n-1)/2$ possible combinations. The threshold of ST strategy is $\pm 2 *$ historical standard deviation. The choice of this threshold may not be optimal. The GA can search for the optimal threshold in the in-sample. Thus, in this study, the GA is used to optimize the open threshold and close threshold based on the in-sample performance. The GA is introduced as follows. The chromosome in the GA is shown in Figure 4.3.

Figure 4.3: Chromosome encoding

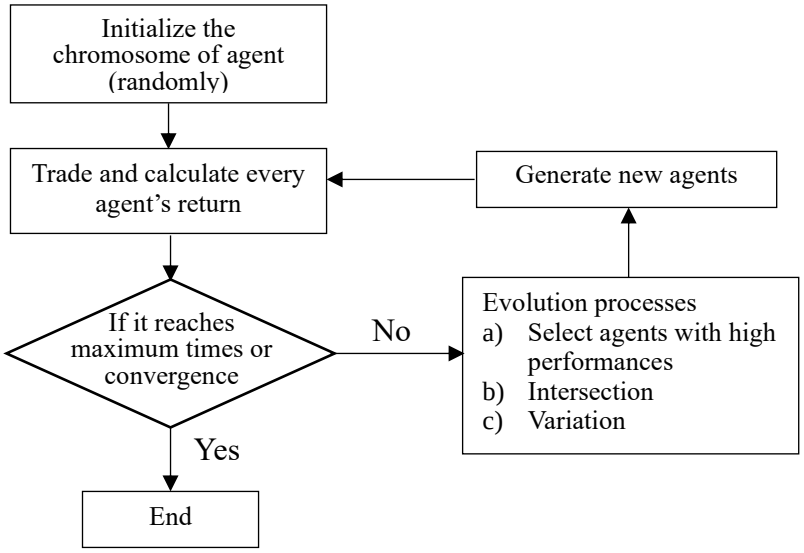


Note: The chromosome is a loci vector with elements of the encoding for open trade threshold and close trade threshold. Each element takes the value 1 or 0.

The binary coding scheme is used to represent a chromosome in the GA. In Figure 4.2, loci b_1^o-b_k^o and b_1^c-b_k^c represent the encoding for open threshold and close threshold (the trades open when the spread is higher than the open threshold $*$ historical standard deviation and close when the spread is lower than close threshold $*$ historical standard deviation). k decides the optimization accuracy, and here I use $k = 10$. b is 0 or 1. Then the agents are trained, as shown in Figure 4.4.

Figure 4.4 shows the training steps for the GA. First, I initialize q agents. Every agent has a random chromosome as presented in Figure 4.3. The chromosome determines the open and close thresholds. Then, I use the thresholds given by agents to trade the pairs and calculate the performance (this research uses the return). In this step, this study uses the initial capital C_0. If a trade opens, the investment of one pair at time t equals C_t/m. After that, following the evolutional processes (select agents with high performances, intersection and variation), a new group of agents can be obtained, repeating the process until it reaches the maximum times or convergence. Finally, I get the best performing agent to decide the threshold of ST and use it in the out-of-sample.

Figure 4.4: The training steps of the GA

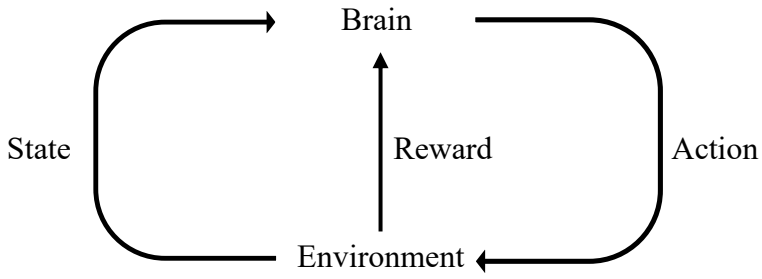


Note: The chromosome of an agent is a loci vector. The vector determines the open and close thresholds in the search range. The search range here is [1,3] for open threshold and [0,1] for close threshold, every search range is equally divided into 2^k sub-ranges for searching.

4.4.2.3 Methodology of DRL in pairs trading

Reinforcement Learning is one of the methodologies of machine learning. It is used to describe and solve the problem that an agent learns a strategy to maximize returns or achieve a specific goal in the process of interacting with the environment (Sutton & Barto, 2018). RL differs from traditional supervised machine learning in the sense that it considers not only the short-term consequences of actions/decisions, but also long-term outcomes (Sutton et al., 2000). RL is different from supervised learning as it only has a reward rather than a teacher or labels. This reward is generated based on the decision of the agent in the RL.

Figure 4.5: The key features of Reinforcement Learning



Note: This figure shows the structure of RL. RL trains the brain through iteration.

RL is characterized by Figure 4.5. The key elements of RL are environment, reward, action,

state and brain, with which a reinforcement learning model can be built. RL aims to obtain an optimal policy for a specific problem. Thus, the reward gained under this policy is the largest. The so-called policy is a series of actions, and these actions are sequential data.

DRL combines the fitting ability of deep learning with the decision-making ability of RL. It is an artificial intelligence technique that is closer to human thinking. Deep learning has a strong fitting capability but lacks decision-making ability, while RL has decision-making ability and cannot deal with the fitting problem. Therefore, the combination of these two methods can obtain complementary advantages, providing a solution to the fitting and decision problem of complex systems.

This section now briefly introduces the RL model and how it can be used in pairs trading in this study. The CA is used for pre-select pairs. m pairs trading opportunities are pre-selected. The price spread of assets $i, j \in \{1 \dots m\}$ at time t is s_{ij}^t.

$$s_{ij}^t = -a_i^t + \gamma a_j^t + C \tag{4.9}$$

ac_{ij}^t represents the trading action of pair ij at time t. s_{ij}^t could be positive or negative, so the action has three choices:

$$ac_{ij}^t = \begin{cases} Buy, & when\ output\ of\ DNN = [1,0,0] \\ Hold, & when\ output\ of\ DNN = [0,1,0] \\ Sell, & when\ output\ of\ DNN = [0,0,1] \end{cases} \tag{4.10}$$

where the output of DNN is the estimated rewards of three actions in this study. If the output in function (4.10) is $[1,0,0]$, ac_{ij}^t is the action 'buy'. ac_{ij}^t also represents the actions 'hold' and 'sell', if the outputs in function (4.10) are $[0,1,0]$ and $[0,0,1]$ repectively.
st_{ij}^t is the state of pair ij at time t.

$$st_{ij}^t = \begin{cases} opened^+ \\ closed \\ opened^- \end{cases} \tag{4.11}$$

$$E_{ij}^t = \left[s_{ij}^t, \dots s_{ij}^{t-x}, st_{ij}^t\right] \tag{4.12}$$

The environment E_{ij}^t is x lags of s_{ij}^t and the state st_{ij}^t, and the output is ac_{ij}^t at time t. An agent is assumed to trade the pair consisting of assets i, j. At each timestamp t, the agent takes action ac_{ij}^t, the state of the environment will change to E_{ij}^{t+1} from E_{ij}^t, and the agent will receive a reward R_{ij}^{t+1}. Possible actions in this agent are 'buy', 'sell' and 'hold' as presents in function (4.10). The reward in this study is the sum of a daily return rt_{ij}^{t+1} at time $t+1$ and the maximum weighted reward in E_{ij}^t estimated by the agent (see function

(4.13)). The ultimate goal is to correctly estimate the rewards of ac_{ij}^{t} under E_{ij}^{t}.

$$R_{ij}^{t+1}\left(E_{ij}^{t}, ac_{ij}^{t}\right) = rt_{ij}^{t+1} + \delta max_{ac_{ij}^{t+1}} R_{ij}^{t+2}\left(E_{ij}^{t+1}, ac_{ij}^{t+1}\right) \quad (4.13)$$

where δ is a parameter to decide the importance of a future reward. $\delta \in [0,1)$. If δ is too close to 1, the DNN may not converge. This study chooses δ of 0.95.

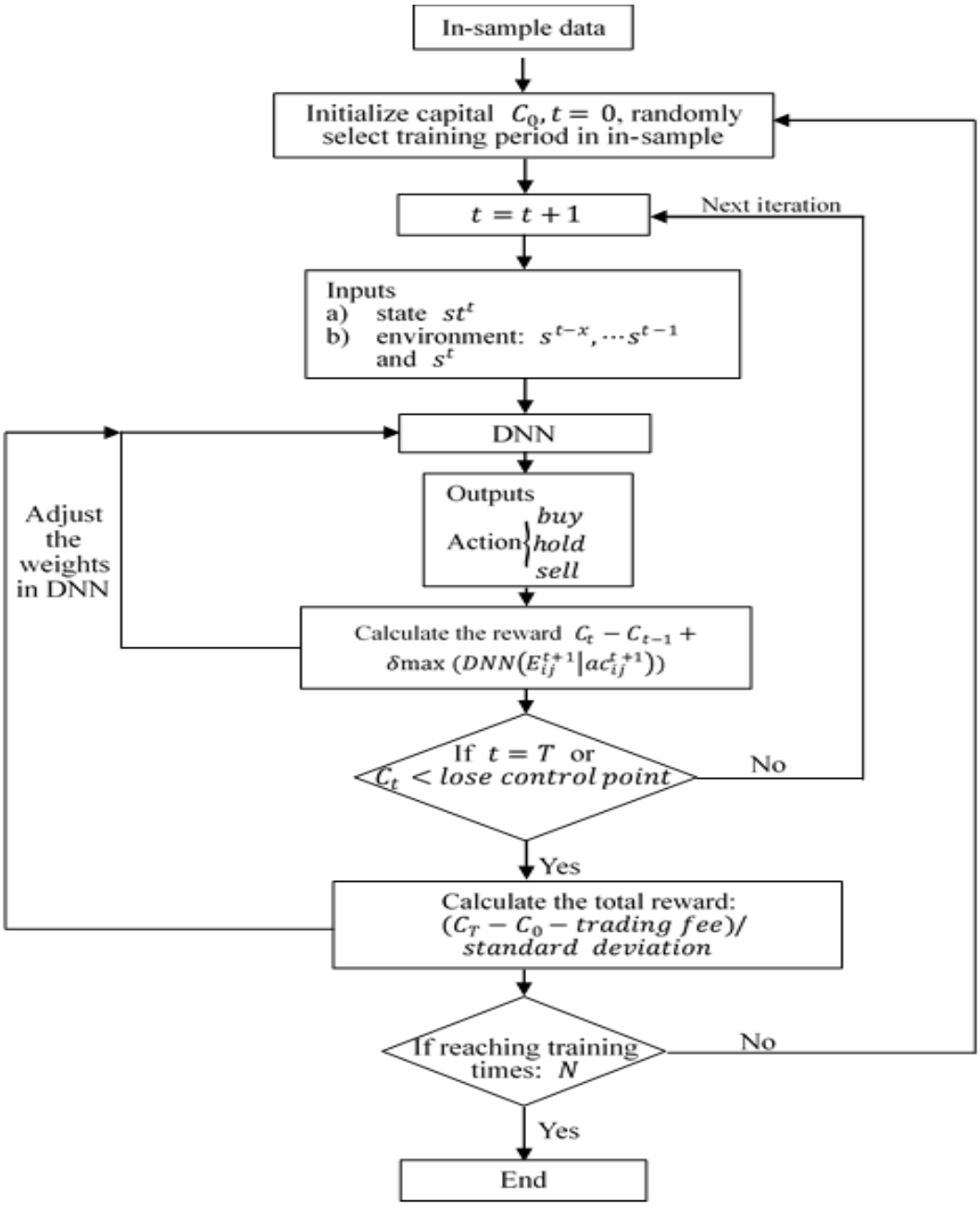
The agent of RL is a Q table, which shows the optimum solution for every E_{ij}^{t} for every ac_{ij}^{t}. However, in trading, the states of the environment are too many to explain in a table. Thus, DNN is used as the brain of the agent in RL (so-called DRL). Using one pair as an example, the DRL model proposed by this study is trained as Figure 4.6.

This chapter chooses the training data (continuous 504 days) from in-sample data in sequence. Then this study initializes the capital C_0, $t = 0$. The inputs of DNN are the state st_{ij}^{t} and the environment x lags of s_{ij}. The outputs decide the action ac_{ij}. Then the temporary reward is calculated and the DNN weights are adjusted. Repeating the iteration until $t = T$. Then the DNN model after training is saved and used as the brain. After that, the capital, time and trading data are initialized, and the whole model is trained again based on the DNN in the last training until the maximum training times N is reached. The DNN used in this study and the weights adjustment method in the DNN are set out in Appendix C.3.

This research uses a pre-training technique to solve two problems during the test. First, every pair has its own model in trading. The calculation amount will be extremely high if the models for all pairs are fully trained. Thus, this study needs a pre-trained model and train the models for pairs based on it. In this way, not only can we get a model which better fits the specific pairs, but we can also reduce the 90% calculation amount. Second, if the study does not carry out pre-training, the model can easily fall into the local optimum. If the model is trapped in the local optimum, this will cause all the decisions of an agent to be the same. For example, the DNN may make 'buy' decisions in all environments, as the results are positive in some in-sample data. In this situation, the training is not effective. In addition, if we start the training with the random weights in the DNN, there will be a high chance of getting trapped into the local optimum, which will cause non-effective training and non-effective calculation. Thus, it is necessary to adopt a pre-training technique for the task of trading a large number of pairs.

The pre-training technique used in this chapter is described as follows. First, this study trains several models by a chosen pair. The model that performs well both in the in-sample and out-of-sample is selected as the pre-trained model for all other pairs. The parameters of the pre-trained model are used as the initial DNN connection weights of other pairs. The models for all other pairs are trained for 5 episodes (5*504 days) based on the pre-trained model.

Figure 4.6: Instruction of DRL for pairs trading



Note: This flowchart shows the iteration processes of DRL used in this study. The brain of the DRL is the DNN. Actions include buy, hold and sell. The environment is the historical data of two commodities.

4.5 Statistical performance

4.5.1 Benchmark performance

Benchmark models include CA-ST, DIM-ST and CA-DIM-ST. In the CA-ST strategy, this chapter uses the Engle-Granger test to test whether the co-integration relationship exists in two commodities. The total number of selected pairs by CA for every period is presented in Appendix C.4. After selection, the pairs with co-integration relationships are traded with the ST strategy. Then, the portfolio comprises those selected pairs with equal weights.

In the DIM-ST strategy, the SSD_{P_i,P_j} is calculated by function (4.1) for every possible pair. Then the top 20 pairs with the minimum SSD_{P_i,P_j} value are chosen to trade with the ST strategy. The CA-DIM-ST strategy combines the selection results of CA and DIM, and uses the top 10 pairs with the minimum SSD_{P_i,P_j} value after the selection of CA. If the number of pairs is lower than 10, the final pairs of CA-DIM are the same as CA. The transaction cost is not considered in the models because of the low frequency of trading actions in those strategies. Trading performances of portfolios are presented in Table 4.3.

Table 4.3 shows the annualized return of three benchmark models in the in-sample and out-of-sample. The annualized returns of three benchmark models are all positive in the in-sample, showing that CA, DIM and CA-DIM are effective for data selection in the in-sample. The average annualized returns of CA-ST and CA-DIM-ST are 10.79% and 10.59%, which are higher than that of DIM-ST (6.89%). However, the average returns of out-of-samples are all very close to zero (0.12%, 0.37% and 0.43%), and there are many negative annualized returns in the out-of-sample over the period 1980-2018. In summary, the benchmark models (CA-ST, DIM-ST and CA-DIM-ST) all fail in commodities pairs trading.

Table 4.3: Annualized returns of the portfolios for benchmark models (%)

	CA-ST		DIM-ST		CA-DIM-ST	
Year	In-sample	Out-of-sample	In-sample	Out-of-sample	In-sample	Out-of-sample
1980-1982	12.16	-1.64	6.19	0.06	12.16	-1.64
1981-1983	10.92	-4.01	6.40	-5.44	7.74	-5.69
1982-1984	7.49	-0.15	6.71	0.10	6.73	-1.08
1983-1985	11.93	0.75	5.10	-1.91	11.93	0.75
1984-1986	11.74	5.84	7.90	9.85	12.75	13.86
1985-1987	12.89	3.34	9.75	2.62	12.89	3.34
1986-1988	14.06	-1.49	11.10	-4.45	15.04	-2.55
1987-1989	9.34	5.73	8.15	0.39	10.35	2.03
1988-1990	8.63	-1.65	6.07	2.15	9.68	-0.97
1989-1991	9.03	0.79	6.18	-0.01	9.18	1.32
1990-1992	11.49	-0.65	6.30	2.91	13.56	3.30
1991-1993	8.31	2.13	5.45	0.99	8.13	1.02
1992-1994	5.92	-2.62	5.05	1.99	7.24	-1.62
1993-1995	9.82	3.84	7.36	0.87	10.77	4.04
1994-1996	8.12	-2.70	5.47	-2.75	5.97	-0.77
1995-1997	8.03	-6.41	8.06	-3.32	10.01	-4.59
1996-1998	7.74	-0.01	5.14	0.48	7.74	-0.01
1997-1999	12.30	-1.54	9.08	-1.17	11.59	0.07
1998-2000	12.52	1.38	7.53	0.34	9.52	-0.05
1999-2001	8.30	-3.08	5.76	0.34	11.67	0.25
2000-2002	9.76	2.92	5.98	-1.13	11.75	5.12
2001-2003	14.62	-3.24	6.62	-0.16	10.49	0.46
2002-2004	10.38	-0.51	6.41	2.70	7.76	1.06
2003-2005	14.88	-1.01	7.90	1.20	12.31	-0.84
2004-2006	12.13	0.58	8.30	-1.47	11.41	2.26
2005-2007	12.65	-0.72	7.09	-0.15	9.01	-1.32
2006-2008	14.13	4.11	6.93	2.16	10.98	2.68
2007-2009	13.14	4.06	6.57	2.85	14.75	-1.18
2008-2010	15.20	-0.47	8.07	0.50	14.05	0.36
2009-2011	13.77	-0.16	7.96	0.77	14.18	-3.22
2010-2012	14.61	2.02	6.01	0.51	10.81	0.24
2011-2013	7.08	-0.16	7.45	0.42	9.21	-0.13
2012-2014	6.46	-1.49	4.86	-1.88	6.98	-2.84
2013-2015	10.19	-0.79	5.72	-1.00	9.36	-1.40
2014-2016	12.46	-0.01	7.44	0.62	9.73	-0.47
2015-2017	8.95	-1.73	6.69	2.66	14.47	0.18
2016-2018	8.23	3.08	6.30	1.01	10.02	3.85
Average	10.79	0.12	6.89	0.37	10.59	0.43

Note: Every three-year data is a test sample (e.g., 1980-1982), which includes the in-sample for the first two years (e.g., 1980-1981) and the out-of-sample for the third year (e.g., 1982). The portfolios are formed by all selected pairs in the in-sample with equal weights. The units in the table are %.

Table 4.4 presents the pairs with non-cointegration relationship traded with ST, where CA-ST is used for comparison. The average annualized return of non-cointegration pairs (6.74%) is not much lower than that of CA-ST (10.79%). Moreover, the out-of-sample average performances are almost the same for non-selected (non-cointegration) pairs and CA-ST. This proves that for the pairs selected by CA in the in-sample, if the ST strategy is still implemented in the out-of-sample, it will not be profitable.

Table 4.4: Annualized returns of the non-cointegration pairs traded with ST (%)

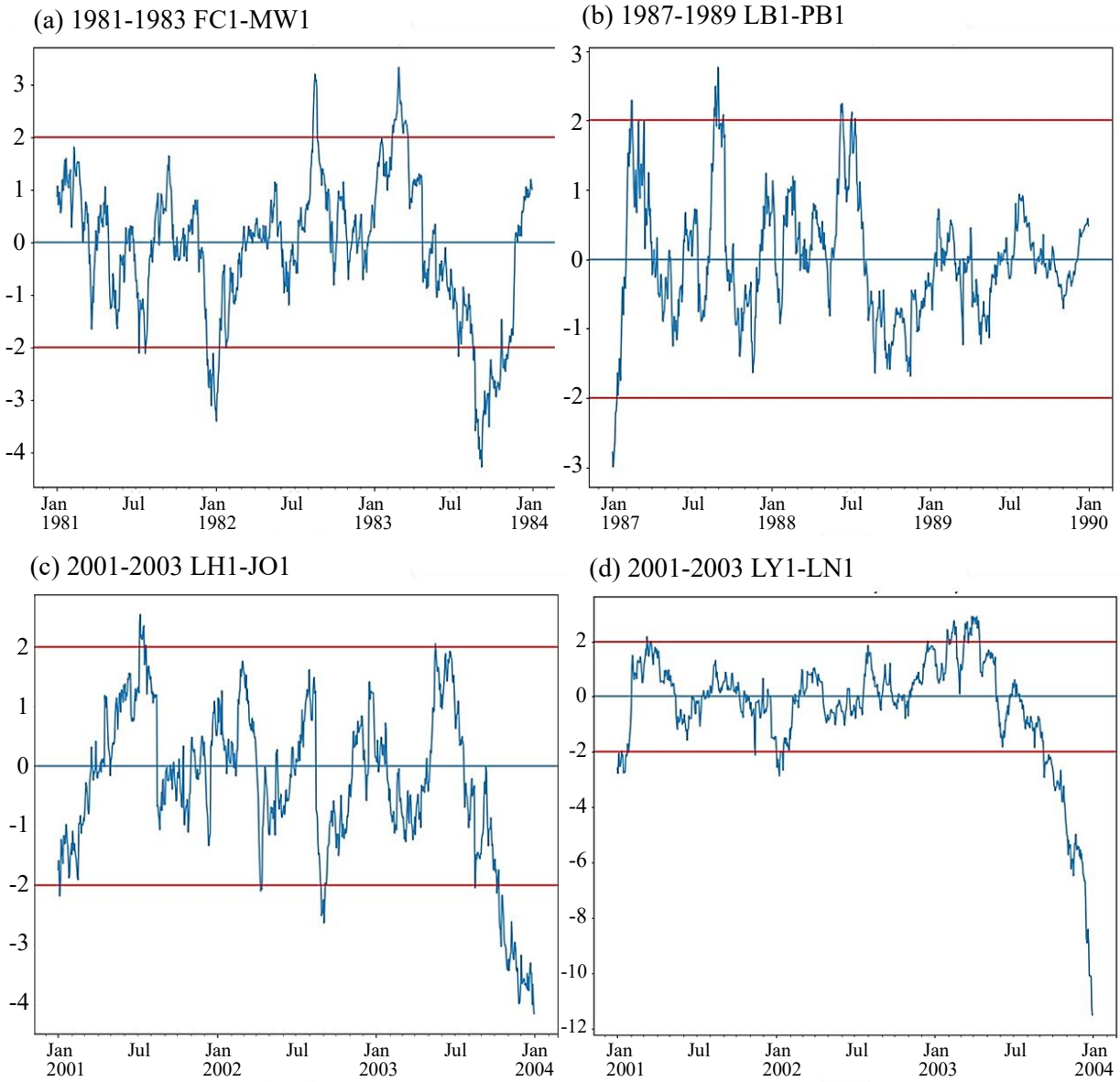
	ST for pairs with non-cointegration		CA-ST	
Year	In-sample	Out-of-sample	In-sample	Out-of-sample
1980-1982	6.30	1.71	12.16	-1.64
1981-1983	7.22	-2.87	10.92	-4.01
1982-1984	5.25	0.09	7.49	-0.15
1983-1985	5.83	-0.94	11.93	0.75
1984-1986	5.59	4.15	11.74	5.84
1985-1987	8.66	0.39	12.89	3.34
1986-1988	8.98	-4.80	14.06	-1.49
1987-1989	7.32	-0.26	9.34	5.73
1988-1990	4.90	0.30	8.63	-1.65
1989-1991	6.65	0.99	9.03	0.79
1990-1992	6.77	1.44	11.49	-0.65
1991-1993	6.70	2.28	8.31	2.13
1992-1994	5.71	0.39	5.92	-2.62
1993-1995	6.79	1.23	9.82	3.84
1994-1996	6.23	-0.56	8.12	-2.70
1995-1997	6.67	-1.49	8.03	-6.41
1996-1998	6.41	0.83	7.74	-0.01
1997-1999	5.89	-1.07	12.30	-1.54
1998-2000	5.70	2.03	12.52	1.38
1999-2001	8.11	0.20	8.30	-3.08
2000-2002	7.39	-0.92	9.76	2.92
2001-2003	6.89	0.06	14.62	-3.24
2002-2004	7.55	-0.83	10.38	-0.51
2003-2005	7.27	0.24	14.88	-1.01
2004-2006	5.92	-1.10	12.13	0.58
2005-2007	7.23	-1.05	12.65	-0.72
2006-2008	6.53	2.63	14.13	4.11
2007-2009	4.83	0.74	13.14	4.06
2008-2010	9.99	2.82	15.20	-0.47
2009-2011	9.92	1.36	13.77	-0.16
2010-2012	8.32	0.49	14.61	2.02
2011-2013	6.69	-0.62	7.08	-0.16
2012-2014	4.81	-0.66	6.46	-1.49
2013-2015	3.70	-1.03	10.19	-0.79
2014-2016	7.73	0.75	12.46	-0.01
2015-2017	6.55	-0.53	8.95	-1.73
2016-2018	6.47	2.10	8.23	3.08
Average	6.74	0.23	10.79	0.12

Note: The performances of all in-samples are better than out-of-samples. That is because correlation coefficients and standard errors are calculated by in-sample data and then use those parameters in the out-of-sample to calculate the spread and make the trading decisions. Every three-year data is a test sample (e.g., 1980-1982), which includes the in-sample for the first two years (e.g., 1980-1981) and the out-of-sample for the third year (e.g., 1982). The units in the table are %.

The low performances of CA-ST, DIM-ST and CA-DIM-ST can be explained in the performances of individual pairs. Figure 4.7 presents four trading samples of CA-ST. In those samples, trading opportunities are rare. For example, the pair '1981-1983FC1 Commodity - MW1 Commodity' has five trading opportunities in three years (two-year in-sample and one-year out-of-sample). The pair '1987-1989 LB1 Commodity – PB1 Commodity' only has three trading opportunities, which are all in the in-sample. The low number of trading opportunities causes a low expectation of returns both in the in-sample and out-of-sample. In addition, the low trading frequency results in low efficiency of capital. Another important cause of the low return in the out-of-sample is that the co-integration relationship may not exist in the out-of-sample of some pairs, which creates a huge loss in

some pairs, as the stop-loss targets are not set in the models. For example, the pair '2001-2003 LH1 Commodity-JO1 Commodity' and the pair '2001-2003 LY1 Commodity-LN1 Commodity' show the situation of loss. Those very high losses cause the average return of the out-of-sample to be low. However, the stop-loss targets are also difficult to set, as I do not know where the turning point is. In summary, CA-ST, DIM-ST and CA-DIM-ST with two standard errors do not work in commodities. The trading thresholds need to be adjusted.

Figure 4.7: Sample of pairs of the price spread



Note: The titles of charts are named by sample period and two commodities' names in the pair. Chart (a) is the pair of feeder cattle and wheat spring; Chart (b) shows the pair of lumber and pork bellies; Chart (c) displays the pair of lean hogs and orange juice; Chart (d) is the pair of aluminium alloy and nickel.

4.5.2 The performance of CA-GA-ST and CA-DRL

This study uses the same data to test the CA-GA-ST trading model. The results are shown in Table 4.5. The GA is used to optimize thresholds, which has a considerable impact on the annualized return of the in-sample. The average annualized return of the in-sample increased from 10.79 of CA-ST to 21.57%, while the improvement of the out-of-sample is small, and the average annualized return of CA-GA-ST is only 1.84%. In the test over 37 years, there are 11 losses for CA-GA-ST, and 23 for CA-ST.

Table 4.5: Annualized returns of CA-GA-ST pairs trading model (%)

	CA-GA-ST		CA-ST	
Year	In-sample	Out-of-sample	In-sample	Out-of-sample
1980-1982	21.94	11.06	12.16	**-1.64**
1981-1983	15.40	2.21	10.92	**-4.01**
1982-1984	8.09	**-3.19**	7.49	**-0.15**
1983-1985	24.40	7.65	11.93	0.75
1984-1986	27.94	7.55	11.74	5.84
1985-1987	23.02	2.69	12.89	3.34
1986-1988	23.91	**-3.93**	14.06	**-1.49**
1987-1989	14.53	3.55	9.34	5.73
1988-1990	19.54	**-0.16**	8.63	**-1.65**
1989-1991	19.86	6.35	9.03	0.79
1990-1992	25.63	4.65	11.49	**-0.65**
1991-1993	18.83	**-8.71**	8.31	2.13
1992-1994	14.54	**-2.67**	5.92	**-2.62**
1993-1995	21.82	2.57	9.82	3.84
1994-1996	15.93	4.36	8.12	**-2.70**
1995-1997	14.42	**-2.32**	8.03	**-6.41**
1996-1998	13.96	2.18	7.74	**-0.01**
1997-1999	25.98	5.86	12.30	**-1.54**
1998-2000	28.77	**-2.11**	12.52	1.38
1999-2001	14.39	2.68	8.30	**-3.08**
2000-2002	19.04	0.84	9.76	2.92
2001-2003	26.87	3.04	14.62	**-3.24**
2002-2004	22.39	2.54	10.38	**-0.51**
2003-2005	23.44	0.29	14.88	**-1.01**
2004-2006	24.70	4.15	12.13	0.58
2005-2007	28.56	**-8.29**	12.65	**-0.72**
2006-2008	29.52	**-5.82**	14.13	4.11
2007-2009	21.85	7.27	13.14	4.06
2008-2010	39.29	1.12	15.20	**-0.47**
2009-2011	32.69	**-0.74**	13.77	**-0.16**
2010-2012	30.52	**-1.58**	14.61	2.02
2011-2013	15.42	4.42	7.08	**-0.16**
2012-2014	11.07	0.22	6.46	**-1.49**
2013-2015	20.97	7.80	10.19	**-0.79**
2014-2016	24.42	5.18	12.46	**-0.01**
2015-2017	17.25	4.75	8.95	**-1.73**
2016-2018	17.07	2.48	8.23	3.08
Average	21.57	1.84	10.79	0.12

Note: The negative annualized returns are marked in bold. Every three-year data is a test sample (e.g., 1980-1982), which includes the in-sample for the first two years (e.g., 1980-1981) and the out-of-sample for the third year (e.g., 1982). The units in the table are %.

Although CA-GA-ST performs better than CA-ST, the performance of CA-GA-ST is in general unsatisfactory. The trading thresholds obtained using the GA search are already close to the optimal solution in the in-sample, compared to CA-ST without optimizing thresholds, and the out-of-sample average annualized return only increases slightly by using the CA-GA-ST strategy. It is reasonable to draw the conclusion that no matter how the trading thresholds are adjusted based on the volatility of price spread in the in-sample, the out-of-sample trading performance will not be improved significantly, but only in-sample performance is improved. This shows that the Simple Thresholds trading strategy does not have any potential, and a more flexible trading strategy is needed to replace it.

Regarding DRL, the in-sample and out-of-sample data for CA-DRL are the same as for the benchmark models. The annualized return of the CA-DRL pairs trading model is presented in Table 4.6. The CA-DRL pairs trading model shows high annualized returns both in the in-sample and out-of-sample, on average 36.00% and 12.49%. The trading performance of the CA-DRL in the out-of-sample even outperforms the in-sample performance of the CA-ST. In addition, only two negative trading periods occur in the out-of-sample, in 01/01/1995-31/12/1995 and 01/01/1998-31/12/1998.

Table 4.6: Annualized returns of CA-DRL pairs trading model (%)

	DRL			DRL	
Year	In-sample	Out-of-sample	Year	In-sample	Out-of-sample
1980-1982	31.20	17.75	1999-2001	33.06	9.55
1981-1983	40.53	8.93	2000-2002	37.40	9.71
1982-1984	20.88	11.54	2001-2003	43.67	16.35
1983-1985	21.72	10.63	2002-2004	36.61	17.20
1984-1986	39.96	21.93	2003-2005	45.01	18.29
1985-1987	48.86	25.02	2004-2006	41.30	14.17
1986-1988	34.02	15.02	2005-2007	42.75	9.66
1987-1989	38.26	7.52	2006-2008	34.70	9.34
1988-1990	33.22	8.36	2007-2009	35.39	12.35
1989-1991	24.96	7.75	2008-2010	56.68	16.36
1990-1992	35.63	12.99	2009-2011	40.06	20.57
1991-1993	40.75	5.43	2010-2012	47.45	18.11
1992-1994	26.26	8.48	2011-2013	32.99	10.40
1993-1995	40.20	**-11.85**	2012-2014	16.89	16.82
1994-1996	23.63	12.47	2013-2015	33.01	13.11
1995-1997	28.51	6.88	2014-2016	38.37	13.54
1996-1998	28.55	**-0.46**	2015-2017	35.29	15.04
1997-1999	49.84	21.90	2016-2018	32.44	20.93
1998-2000	42.01	10.22	Average	36.00	12.49

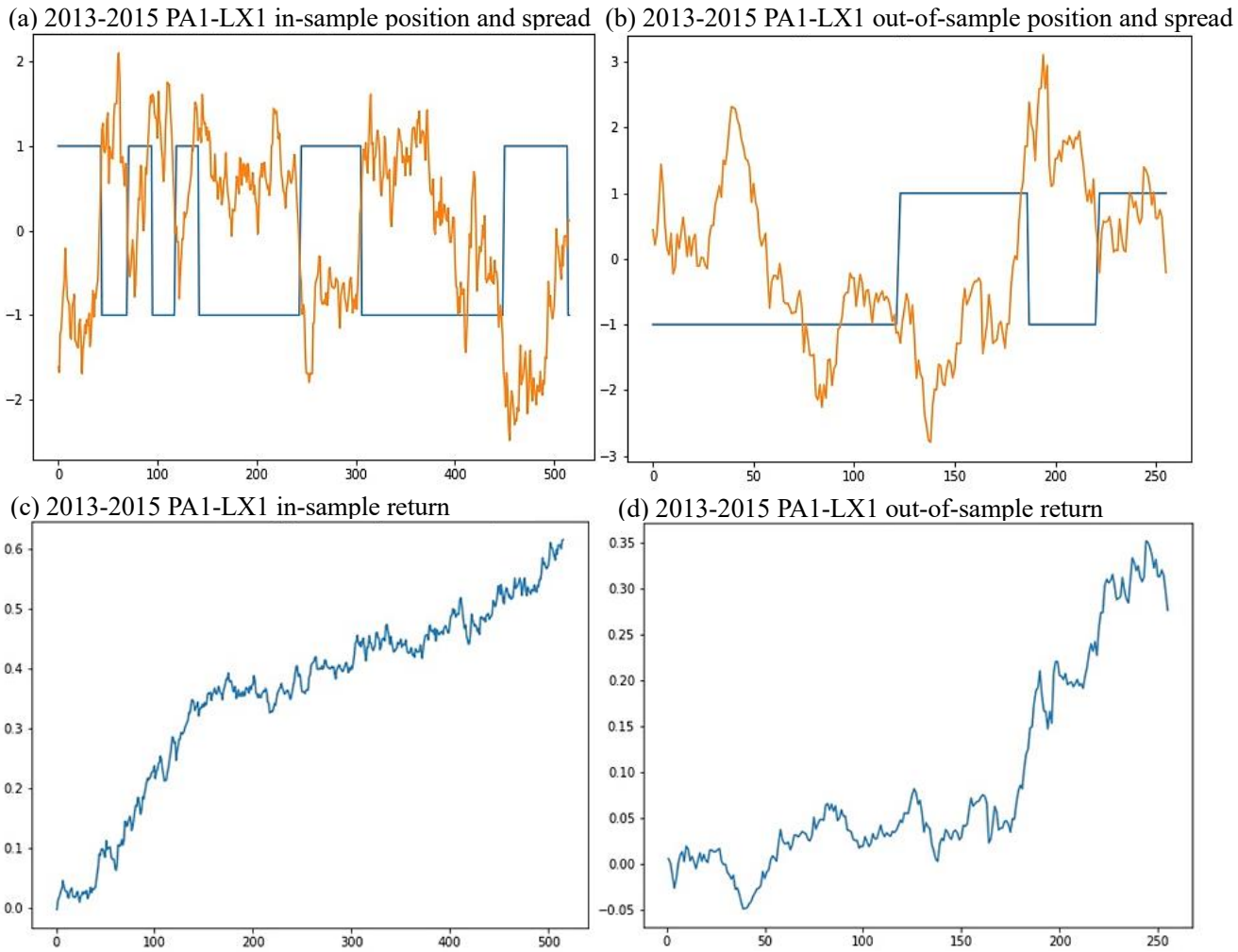
Note: The negative annualized returns are marked in bold. Every three-year data is a test sample, which includes the in-sample for the first two years and the out-of-sample for the third year. The units in the table are %.

The annualized returns in the in-sample are all much higher than in the out-of-sample, since this study trains model for every pair. Every individual pair is trained individually, which

leads to a higher in-sample performance. In addition, having different models for different pairs also improves the performance in the out-of-sample.

In every trading period, this study trains a DRL model for every pair. However, there are two reasons why the DRL may fail in the in-sample. First, I set maximum training times, which may cause the model still not to converge when it reaches the maximum training times. Second, the training may get trapped into local optimum. There are few failures in the in-sample. Thus, this study chooses the pairs to trade in the out-of-sample only if the return in the in-sample is positive, and this process increases the average return. The results that are not processed by this step are presented in Appendix C.5.

Figure 4.8: A sample of a pair traded by DRL

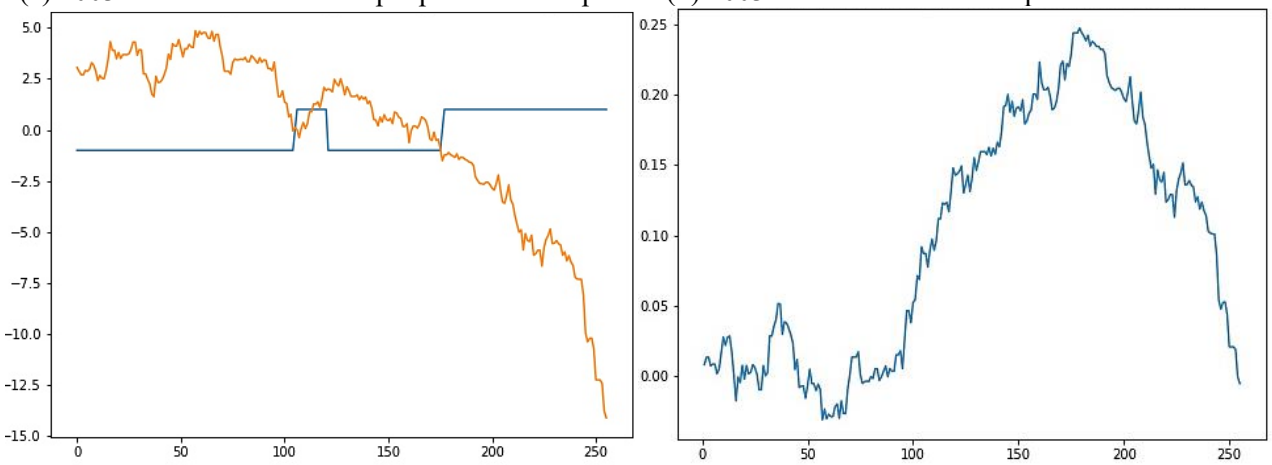


Note: This figure shows a pair (palladium and zinc) traded by DRL. The top two charts display the positions (blue line) and spread (yellow line) of the pair in the in-sample (the top-left chart) and out-of-sample (the top-right chart). The bottom two charts show the return of the commodity pair in the in-sample (the bottom-left chart) and the out-of-sample (the bottom-right chart). The data for 2013-2015 is a test sample, which includes the in-sample for the first two years (2013 & 2014) and the out-of-sample for the third year (2015).

Figure 4.8 shows an example of a commodity pair traded by DRL. In the top two charts, the blue line represents the positions: $position = 1$ means a long position in 'commodity1' and a short position in 'commodity2', $position = 0$ is the close position, and $position = -1$ represents a short position in 'commodity1' and a long position in 'commodity2'. The yellow line represents the spread of two commodities after being divided by the standard error of their price spread in the in-sample. Figure 4.8 shows that the trading performance of DRL is good both in the in-sample (see the top-left chart) and out-of-sample (see the top-right chart). The return reaches 60% in the two-year in-sample and 25% in the one-year out-of-sample reaches. The top two charts show the position of DRL. Compared with the ST method (in Figure 4.7), DRL is rarely in the close position, which makes the capital efficiency of DRL higher than that of ST. In addition, the DRL trading model does not change its position frequently, which means a very low transaction cost.

Figure 4.9: The failed sample of a pair traded by DRL

(a) 2003 LY1-LN1 out-of-sample position and spread (b) 2003 LY1-LN1 out-of-sample return



Note: This is a failed sample (the pair of aluminium alloy and nickel in 2003) for DRL. Although the highest return during the out-of-sample test reaches 25%, the final return is close to zero. The left chart displays the positions (blue line) and spread (yellow line) of the pair in out-of-sample. The right chart shows the return of the commodity pair in the out-of-sample.

Figure 4.9 shows that the DRL trading strategy does not avoid loss in the case of a sudden increase in price spread. However, due to the positive return before the loss, the overall return of the out-of-sample is close to 0. The large loss in the extreme situation of DRL indicates that DRL in this study does not have the ability to generate the stop-loss targets by its DNN. However, in a normal situation, DRL has higher returns compared with ST, which leads to the positive return of the portfolio.

In summary, CA-DRL performs significantly better than the benchmark models from the perspective of average returns, both in the in-sample and out-of-sample. Although the in-sample performance of the CA-DRL trading strategy is significantly better than that of the out-of-sample, the out-of-sample still has a high annualized return.

4.6 Risk evaluation

4.6.1 The return on actual employed capital

The benchmark models, CA-GA-ST and CA-DRL all have the same committed capital ($\sum C_0^i$) but different actual employed capital. The average annualized return on committed capital takes the sum of the returns over all pairs during the trading period, and divides it by the number of pairs in the portfolio, which were used in Section 4.5. This measure of this return is conservative, as there are many close positions for pairs when capital is not employed. A hedge fund would be more flexible in its use of funds. Calculating return relative to the actual employed capital is a more realistic measure of the performance. Capital efficiency is introduced to evaluate the ratio of how much capital is employed on average. Capital efficiency is calculated by actual employed capital divided by committed capital. Lower capital efficiency means that the portfolio is always in a close position and uses less capital.

Table 4.7 shows that CA-ST's capital efficiency is low, which means that the CA-ST strategy has more close positions. This is because the absolute value of the trading thresholds is large in the CA-ST strategy. When the trading thresholds are reduced, for example, CA-GA-ST's capital efficiency is higher. The capital efficiency of CA-DRL is always higher than 99%, which means the CA-DRL strategy is more aggressive and always in a full position.

Table 4.7: Capital efficiency of CA-ST, CA-GA-ST and CA-DRL (%)

	CA-ST		CA-GA-ST		CA-DRL	
Year	In-sample	Out-of-sample	In-sample	Out-of-sample	In-sample	Out-of-sample
1980-1982	22.25	29.65	85.76	81.90	99.25	99.46
1981-1983	23.48	28.28	82.61	85.44	99.27	99.66
1982-1984	23.11	18.01	88.41	87.83	99.44	99.22
1983-1985	23.99	32.06	87.72	89.53	99.61	99.48
1984-1986	18.03	27.32	86.94	85.40	99.32	99.65
1985-1987	21.71	14.35	85.12	84.72	99.33	99.39
1986-1988	23.07	41.73	86.76	90.11	99.39	99.45
1987-1989	20.66	25.82	85.59	85.84	99.24	99.72
1988-1990	21.99	34.36	84.96	87.42	99.37	99.20
1989-1991	23.28	22.12	85.83	87.08	99.46	99.51
1990-1992	18.68	26.53	83.82	84.57	99.34	99.64
1991-1993	14.14	19.84	85.14	83.32	99.36	99.53
1992-1994	22.09	24.70	83.38	84.08	99.30	99.58
1993-1995	22.56	22.78	85.37	86.66	99.24	99.76
1994-1996	20.08	28.74	82.23	84.24	99.20	99.27
1995-1997	19.34	19.18	83.99	82.84	99.38	99.49
1996-1998	18.56	23.40	83.59	82.46	99.45	99.55
1997-1999	19.19	25.94	84.09	85.33	99.33	99.63
1998-2000	19.99	25.03	84.75	84.87	99.34	99.35
1999-2001	18.93	21.75	84.96	85.96	99.33	99.45
2000-2002	19.52	26.84	84.99	85.65	99.36	99.74
2001-2003	21.18	28.99	85.58	85.76	99.44	99.22
2002-2004	20.78	23.85	84.51	85.58	99.26	98.61
2003-2005	18.69	10.25	85.25	84.96	99.51	99.66
2004-2006	19.55	25.34	85.31	86.49	99.39	99.26
2005-2007	16.74	11.29	82.17	79.71	99.23	99.61
2006-2008	18.05	26.28	83.88	83.89	99.46	99.18
2007-2009	20.76	16.21	84.05	83.16	99.26	99.53
2008-2010	19.85	28.72	85.14	87.04	99.15	99.55
2009-2011	18.16	20.23	82.83	84.01	99.22	98.89
2010-2012	20.62	20.01	84.05	82.96	99.33	99.66
2011-2013	20.01	25.87	84.00	84.52	99.27	99.25
2012-2014	25.54	36.90	86.25	87.94	99.38	99.37
2013-2015	21.95	17.46	85.19	83.91	99.24	99.34
2014-2016	20.94	26.24	85.15	85.70	99.32	99.37
2015-2017	22.32	26.36	84.42	84.25	99.39	99.46
2016-2018	21.79	25.82	84.86	86.35	99.26	99.18
Average	20.58	24.55	84.83	85.17	99.34	99.43

Note: Capital efficiency is calculated by actual employed capital divided by committed capital. The higher capital efficiency means that a strategy uses more capital during the whole trading period. The units in the table are %.

Table 4.8 shows the return on actual employed capital, which is calculated by the return on committed capital divided by the corresponding capital efficiency. The use of loose thresholds such as CA-ST, produces the best performance in the in-sample, but the worst performance in the out-of-sample. CA-GA-ST also has low out-of-sample returns with an average out-of-sample return of 2.16%. This proves that the ST method, no matter how the thresholds are adjusted, can only obtain low returns in the 1980-2018 commodities markets. CA-DRL's aggressive trading strategy and high capital efficiency make the return on actual employed capital and the return on committed capital almost the same.

Table 4.8: Return on actual employed capital (%)

	CA-ST		CA-GA-ST		CA-DRL	
Year	In-sample	Out-of-sample	In-sample	Out-of-sample	In-sample	Out-of-sample
1980-1982	54.65	-5.53	25.58	13.50	31.44	17.85
1981-1983	46.51	-14.18	18.64	2.59	40.83	8.96
1982-1984	32.41	-0.83	9.15	-3.63	21.00	11.63
1983-1985	49.73	2.34	27.82	8.54	21.81	10.69
1984-1986	65.11	21.38	32.14	8.84	40.23	22.01
1985-1987	59.37	23.28	27.04	3.18	49.19	25.17
1986-1988	60.94	-3.57	27.56	-4.36	34.23	15.10
1987-1989	45.21	22.19	16.98	4.14	38.55	7.54
1988-1990	39.25	-4.80	23.00	-0.18	33.43	8.43
1989-1991	38.79	3.57	23.14	7.29	25.10	7.79
1990-1992	61.51	-2.45	30.58	5.50	35.87	13.04
1991-1993	58.77	10.74	22.12	-10.45	41.01	5.46
1992-1994	26.80	-10.61	17.44	-3.18	26.45	8.52
1993-1995	43.53	16.86	25.56	2.97	40.51	-11.88
1994-1996	40.44	-9.39	19.37	5.18	23.82	12.56
1995-1997	41.52	-33.42	17.17	-2.80	28.69	6.92
1996-1998	41.70	-0.04	16.70	2.64	28.71	-0.46
1997-1999	64.10	-5.94	30.90	6.87	50.18	21.98
1998-2000	62.63	5.51	33.95	-2.49	42.29	10.29
1999-2001	43.85	-14.16	16.94	3.12	33.28	9.60
2000-2002	50.00	10.88	22.40	0.98	37.64	9.74
2001-2003	69.03	-11.18	31.40	3.54	43.92	16.48
2002-2004	49.95	-2.14	26.49	2.97	36.88	17.44
2003-2005	79.61	-9.85	27.50	0.34	45.23	18.35
2004-2006	62.05	2.29	28.95	4.80	41.55	14.28
2005-2007	75.57	-6.38	34.76	-10.40	43.08	9.70
2006-2008	78.28	15.64	35.19	-6.94	34.89	9.42
2007-2009	63.29	25.05	26.00	8.74	35.65	12.41
2008-2010	76.57	-1.64	46.15	1.29	57.17	16.43
2009-2011	75.83	-0.79	39.47	-0.88	40.37	20.80
2010-2012	70.85	10.09	36.31	-1.90	47.77	18.17
2011-2013	35.38	-0.62	18.36	5.23	33.23	10.48
2012-2014	25.29	-4.04	12.83	0.25	17.00	16.93
2013-2015	46.42	-4.52	24.62	9.30	33.26	13.20
2014-2016	59.50	-0.04	28.68	6.04	38.63	13.63
2015-2017	40.10	-6.56	20.43	5.64	35.51	15.12
2016-2018	37.77	11.93	20.12	2.87	32.68	21.10
Average	52.43	0.49	25.43	2.16	36.24	12.56

Note: The return on actual employed capital is the return on committed capital divided by capital efficiency. The units in the table are %.

4.6.2 The Risk-Adjusted Return

4.6.2.1 The Sharpe Ratio and Value at Risk of portfolio

The Sharpe Ratio and Value at Risk of the portfolio with CA-ST, CA-GA-ST and CA-DRL strategies are presented in Table 4.9. The annualized out-of-sample return of CA-ST is very low (0.12%), which results in a CA-ST strategy with almost no trading value, even though the volatility of CA-ST is low. The annualized return of CA-GA-ST is also much lower than that of CA-DRL, while its volatility is lower than that of CA-DRL but higher than that of CA-ST. Though the annualized volatility of CA-DRL is the highest, its outstanding annualized return means that CA-DRL still has a satisfactory Sharpe Ratio of 1.853. The calculation of annual Value at Risk is based on the Portvrisk function in the MATLAB Financial toolbox. CA-DRL has the best VaR performance, with a VaR of 0 at a 95% confidence level and a VaR of -3.19% at a 99% confidence level. This is because losses rarely occur in the out-of-sample, showing the robustness of CA-DRL for pairs trading. Though the VaR performance of CA-ST and CA-GA-ST is not bad, owing to their low rate of return, it is not a wise choice to invest using CA-ST and CA-GA-ST strategies. The probability of CA-ST having returns below zero is greater than 50%, which again indicates that CA-ST is ineffective in community markets. CA-GA-ST has a 29.73% probability that the annualized returns will be lower than zero. As a portfolio, this performance is not satisfactory. However, the CA-DRL portfolio only has a 5.41% probability of returns lower than zero. The minimum historical annualized return of CA-DRL is lower than the CA-ST and CA-GA-ST.

Table 4.9: Summary of annually out-of-sample trading performance of the portfolio

	CA-ST	CA-GA-ST	CA-DRL
Annualized return	0.12%	1.84%	12.49%
Volatility	2.72%	4.47%	6.74%
Sharpe Ratio	0.044	0.412	1.853
Annual Value at Risk			
1%	-6.20%	-8.56%	-3.19%
5%	-4.35%	-5.52%	0
10%	-3.36%	-3.89%	0
15%	-2.70%	-2.80%	0
20%	-2.16%	-1.93%	0
Probability below 0	62.16%	29.73%	5.41%
Min. historical observation	-6.41%	-8.71%	-11.85%

Note: The Sharpe Ratio is based on the return on committed capital. Risk-free is assumed to equal zero.

4.6.2.2 The Sortino Ratio

Keating & Shadwick (2002) define the Sortino Ratio to measure the risk-adjusted return. Different from the Sharpe Ratio, the Sortino Ratio does not assume a normal distribution of returns. The Sortino Ratio is defined as:

$$S = \frac{\mu - \tau}{\sqrt{\int_{-\infty}^{\tau}(\tau - R)^2 dF(R)}} \tag{4.14}$$

Where:

- $F(\cdot)$ = the cumulative density function for total returns on an investment
- $\tau =$ threshold return
- $\mu =$ the expected periodic return = $\int_{-\infty}^{\infty} RdF(R)$

The evaluation of the return in the Sortino Ratio and the Sharpe Ratio is the same. The Sortino Ratio evaluates the risk by only considering the returns lower than τ, while the Sharpe Ratio considers both positive and negative returns. Table 4.10 shows that under similar risk evaluations (0.103, 0.150 and 0.119), CA-DRL provides the best expected return. Thus, CA-DRL shows the best Sortino Ratio (1.053) compared with CA-GA-ST and CA-ST.

Table 4.10: The Sortino Ratios of annually out-of-sample trading performance (of the portfolio)

	CA-ST	CA-GA-ST	CA-DRL
μ	0.12%	1.84%	12.49%
τ	0	0	0
Risk evaluation	0.103	0.150	0.119
Sortino Ratio	0.019	0.123	1.053

Note: The risk-free rate is assumed to equal zero, and thus τ is 0.

4.6.2.3 Morningstar Risk-Adjusted Return

Morningstar lnc. is a provider of independent investment research in the US which uses the MRAR to rate funds. The MRAR is calculated as follows (Kräussl & Sandelowsky, 2007).

$$TR = \left\{\frac{P_e}{P_b}\prod_{i}^{n}\left(1 + \frac{D_i}{P_i}\right)\right\} - 1 \tag{4.15}$$

Where TR is the total return for one month. P_e is the Net Asset Value (NAV) at the end of the month, P_b is the NAV at the beginning of the month. D_i is the distribution per share at

time i, P_i is the reinvestment NAV per share at time i. n is the number of distributions during this month. Distributions include dividends, distributed capital gains and return of capital. However, the portfolios in this study do not have dividends. D_i is the return of the capital that is not employed. The cumulative value for one unit capital of MRAR monthly is defined as:

$$V_u = \prod_{t=1}^{T}(1 + TR_t) \tag{4.16}$$

Where V_u is the value before adjusting the loads and redemption fees.

$$V = (1 - F)(1 - R)V_u - D(1 - F)\frac{\min(P_0, P_t)}{P_0} \tag{4.17}$$

Where F is the front load and D is the deferred load. R is the redemption fee, P_0 is NAV per share at the start and P_t is the NAV at the end. In this study, the portfolio does not have any fees of loads. Thus $V = V_u$.

MRAR is defined as follows:

$$MRAR(\gamma) = \left[\frac{1}{T}\sum_{t=1}^{T}\left(1 + r_{G_t}\right)^{-\gamma}\right]^{-\frac{12}{\gamma}} - 1 \tag{4.18}$$

Where r_{G_t} is the excess return compared with the risk-free ratio.

$$r_{G_t} = \frac{1 + TR_t}{1 + R_f} - 1 \tag{4.19}$$

Where R_f is the risk-free ratio. γ is the parameter defined to describe the degree of risk aversion. This research adopts $\gamma = 2$, which is taken to correspond to a typical investor (Kräussl & Sandelowsky, 2007).

Table 4.11 presents the cumulative value and MRAR(2) of strategies. Considering the deposit ratio and risk-free ratio, the average adjusted return of CA-ST is negative, and the average adjusted return of CA-GA-ST is very close to zero. CA-DRL shows the best performance with an annualized average adjusted return of 9.98%.

Table 4.11: MRAR(2) of CA-ST, CA-GA-ST and CA-DRL

	CA-ST		CA-GA-ST		CA-DRL	
Year	Cumulative value	MRAR(2)	Cumulative value	MRAR(2)	Cumulative value	MRAR(2)
1980-1982	0.9945	-0.0258	1.0317	0.0088	1.1939	0.1667
1981-1983	0.9757	-0.0447	0.9564	-0.0663	1.0566	0.0307
1982-1984	1.0045	-0.0157	1.0466	0.0249	1.0841	0.0613
1983-1985	1.0226	0.0019	0.8519	-0.1714	1.1061	0.0773
1984-1986	1.0813	0.0565	1.0982	0.0658	1.2474	0.2161
1985-1987	1.0373	0.0162	1.0122	-0.0099	1.2646	0.2348
1986-1988	1.0008	-0.0211	0.9913	-0.0343	1.1386	0.1103
1987-1989	1.0739	0.0522	1.1376	0.1134	1.0248	0.0032
1988-1990	1.0007	-0.0207	1.0063	-0.0187	1.0982	0.0726
1989-1991	1.0079	-0.0123	1.0167	-0.0042	1.0715	0.0488
1990-1992	1.0012	-0.0191	1.0361	0.0143	1.1313	0.1060
1991-1993	1.0278	0.0073	1.0843	0.0602	1.0530	0.0268
1992-1994	0.9791	-0.0410	1.0273	0.0036	1.0853	0.0598
1993-1995	1.0471	0.0255	1.0093	-0.0180	0.8888	-0.1323
1994-1996	0.9810	-0.0386	0.9566	-0.0626	1.1214	0.0982
1995-1997	0.9401	-0.0795	1.0430	0.0160	1.0660	0.0405
1996-1998	1.0189	-0.0017	0.9763	-0.0442	1.0268	0.0050
1997-1999	0.9956	-0.0246	1.0653	0.0417	1.2083	0.1834
1998-2000	1.0224	0.0013	1.0741	0.0499	1.0882	0.0631
1999-2001	0.9777	-0.0421	1.0698	0.0464	1.0754	0.0534
2000-2002	1.0415	0.0202	1.0801	0.0573	1.0950	0.0713
2001-2003	0.9614	-0.0592	0.9950	-0.0274	1.1511	0.1241
2002-2004	1.0010	-0.0190	1.0048	-0.0157	1.1692	0.1443
2003-2005	0.9879	-0.0318	1.1058	0.0755	1.1917	0.1616
2004-2006	1.0137	-0.0071	0.9990	-0.0223	1.1408	0.1165
2005-2007	0.9955	-0.0243	1.0631	0.0395	1.0935	0.0664
2006-2008	1.0520	0.0300	1.0321	0.0067	1.0936	0.0665
2007-2009	1.0705	0.0444	1.2172	0.1782	1.1456	0.1197
2008-2010	1.0027	-0.0178	1.0522	0.0293	1.1698	0.1448
2009-2011	1.0045	-0.0157	1.0544	0.0302	1.2210	0.1936
2010-2012	1.0253	0.0048	1.0386	0.0158	1.2139	0.1829
2011-2013	1.0071	-0.0130	1.0068	-0.0135	1.1082	0.0855
2012-2014	0.9859	-0.0344	0.9763	-0.0435	1.1720	0.1479
2013-2015	1.0009	-0.0192	1.0453	0.0233	1.1386	0.1148
2014-2016	1.0058	-0.0145	1.0043	-0.0165	1.1478	0.1237
2015-2017	0.9878	-0.0318	0.9377	-0.0813	1.1448	0.1203
2016-2018	1.0323	0.0113	1.0545	0.0310	1.2072	0.1825
Average	1.0099	-0.0109	1.0313	0.0076	1.1253	0.0998

Note: 1% deposit ratio and 2% risk-free ratio are adopted in this calculation.

4.6.3 Maximum Drawdown

4.6.3.1 Individual pair Maximum Drawdown

Table 4.12 shows the Maximum Drawdown value of the worst-performing pair among all the pairs selected in the portfolio each year. The Maximum Drawdown in this study is calculated every two years in the in-sample and every year in the out-of-sample.

Table 4.12: Individual pair Maximum Drawdown of CA-ST, CA-GA-ST and CA-DRL (%)

	CA-ST		CA-GA-ST		CA-DRL	
Year	In-sample	Out-of-sample	In-sample	Out-of-sample	In-sample	Out-of-sample
1980-1982	-13.73	-8.18	-23.61	-24.09	-45.85	-24.09
1981-1983	-39.28	-30.74	-49.32	-32.08	-49.62	-32.08
1982-1984	-24.20	-29.62	-26.04	-29.62	-35.21	-35.50
1983-1985	-6.74	-18.29	-8.34	-23.25	-16.05	-18.29
1984-1986	-12.93	-28.17	-32.81	-34.38	-40.74	-24.85
1985-1987	-22.42	-17.77	-34.58	-22.71	-27.91	-15.83
1986-1988	-31.71	-35.05	-36.59	-40.18	-60.88	-49.24
1987-1989	-37.36	-34.70	-67.90	-34.70	-59.35	-41.92
1988-1990	-23.37	-27.34	-43.14	-32.71	-35.25	-27.87
1989-1991	-15.18	-27.68	-22.68	-30.29	-26.16	-24.07
1990-1992	-14.60	-25.74	-14.60	-28.56	-27.58	-21.96
1991-1993	-20.58	-42.76	-54.64	-49.51	-43.59	-53.12
1992-1994	-20.68	-29.52	-31.85	-40.59	-30.87	-29.05
1993-1995	-15.90	-25.82	-29.03	-25.82	-41.76	-38.00
1994-1996	-16.13	-37.70	-46.24	-49.44	-47.70	-39.32
1995-1997	-24.26	-37.45	-51.83	-36.01	-67.81	-40.40
1996-1998	-18.51	-25.26	-31.93	-27.99	-23.79	-27.16
1997-1999	-28.48	-35.68	-51.30	-35.68	-60.15	-38.33
1998-2000	-21.03	-50.36	-44.52	-53.67	-52.76	-44.27
1999-2001	-34.38	-39.36	-51.54	-53.24	-60.25	-54.05
2000-2002	-16.69	-32.16	-50.36	-47.73	-49.68	-35.02
2001-2003	-22.57	-27.33	-35.20	-31.88	-43.62	-31.38
2002-2004	-17.06	-39.59	-43.20	-39.59	-55.26	-27.21
2003-2005	-21.33	-42.80	-29.74	-42.97	-40.99	-22.33
2004-2006	-13.92	-53.11	-58.87	-53.11	-50.70	-37.37
2005-2007	-12.04	-48.85	-63.39	-48.85	-32.04	-26.11
2006-2008	-15.71	-37.28	-65.64	-37.82	-59.55	-39.34
2007-2009	-14.01	-45.70	-30.66	-45.70	-31.70	-45.70
2008-2010	-38.37	-31.17	-28.41	-31.63	-41.94	-24.67
2009-2011	-10.14	-29.37	-20.29	-30.89	-27.54	-17.88
2010-2012	-18.19	-29.83	-18.92	-33.24	-30.30	-20.37
2011-2013	-17.38	-21.77	-34.22	-27.39	-31.51	-22.86
2012-2014	-16.59	-26.65	-32.13	-30.06	-27.30	-27.37
2013-2015	-14.77	-33.85	-16.64	-33.85	-19.35	-33.85
2014-2016	-12.84	-25.09	-21.79	-29.87	-22.40	-28.62
2015-2017	-28.47	-27.60	-50.04	-31.44	-43.36	-30.72
2016-2018	-21.94	-23.80	-39.10	-26.33	-41.11	-31.69
Average	-20.36	-31.98	-37.60	-35.86	-40.58	-31.94
Minimum	-39.28	-53.11	-67.90	-53.64	-60.88	-54.05

Note: The calculated Maximum Drawdown here for the in-sample is the Maximum Drawdown for two years, while for the out-of-sample, the Maximum Drawdown is for one year. The units in the table are %.

Table 4.12 describes that out of 1329 pairs, the out-of-sample Maximum Drawdowns of the worst-performing individual pairs of the three strategies in 1980-2018 are all around 50% (see 'Minimum' in Table 4.12). That is, for instance, at a leverage of 1.8, no pair's assets will become zero during the 1980-2018 trading period. Table 4.12 also shows the worst performance of Maximum Drawdown among all pairs each year. This is an extreme situation, indicating that the prices of the two commodities of a pair deviate a great deal. For most pairs, their Maximum Drawdowns are much smaller.

4.6.3.2 Portfolio Maximum Drawdown

It is conservative to use leverage based on Maximum Drawdown of individual pairs. While for a portfolio, when the value of an asset is less than zero, other assets may have values greater than zero, as long as the total assets of the portfolio are not less than zero, it will not go bankrupt. Therefore, the leverage that the portfolio can withstand is determined by the minimum Maximum Drawdown of the overall portfolio.

Table 4.13: Portfolio Maximum Drawdown (%)

	CA-ST		CA-GA-ST		CA-DRL	
Year	In-sample	Out-of-sample	In-sample	Out-of-sample	In-sample	Out-of-sample
1980-1982	-1.05	-1.51	-4.31	-3.87	-4.99	-3.81
1981-1983	-2.09	-4.65	-3.15	-7.75	-2.32	-6.55
1982-1984	-3.70	-7.35	-3.99	-6.13	-6.03	-3.15
1983-1985	-2.18	-5.07	-4.46	-6.49	-11.05	-5.62
1984-1986	-1.74	-4.81	-3.54	-7.27	-3.10	-2.79
1985-1987	-1.99	-4.91	-6.79	-7.41	-4.23	-2.16
1986-1988	-1.75	-9.15	-2.92	-11.04	-3.54	-6.05
1987-1989	-1.79	-4.44	-2.54	-4.19	-3.26	-3.94
1988-1990	-1.16	-2.92	-2.51	-3.87	-2.04	-2.86
1989-1991	-1.12	-3.41	-1.87	-2.17	-3.15	-2.55
1990-1992	-0.72	-3.68	-1.54	-3.84	-1.37	-2.82
1991-1993	-6.37	-7.71	-6.41	-11.65	-4.67	-6.72
1992-1994	-1.25	-8.23	-1.97	-9.83	-3.14	-3.88
1993-1995	-1.86	-3.99	-4.44	-4.58	-3.67	-13.74
1994-1996	-0.66	-2.44	-1.24	-2.53	-1.51	-1.61
1995-1997	-1.36	-6.58	-2.83	-7.02	-4.78	-3.96
1996-1998	-1.48	-3.17	-3.04	-2.57	-2.74	-3.12
1997-1999	-1.14	-2.33	-1.70	-2.82	-3.10	-1.37
1998-2000	-0.72	-2.96	-2.89	-3.25	-1.09	-4.11
1999-2001	-0.45	-1.43	-1.16	-1.52	-2.07	-1.71
2000-2002	-0.64	-3.76	-1.27	-3.34	-1.17	-2.82
2001-2003	-2.65	-3.87	-3.77	-7.51	-7.23	-5.15
2002-2004	-0.52	-3.20	-1.11	-2.36	-1.21	-1.21
2003-2005	-3.25	-8.94	-6.76	-8.23	-5.59	-5.07
2004-2006	-2.05	-3.85	-2.11	-3.00	-2.35	-3.76
2005-2007	-1.17	-13.71	-1.45	-14.10	-1.69	-7.50
2006-2008	-3.21	-7.41	-6.43	-7.85	-8.58	-6.48
2007-2009	-1.01	-5.05	-2.95	-3.81	-3.66	-5.05
2008-2010	-1.13	-3.29	-2.15	-4.77	-3.56	-3.14
2009-2011	-0.79	-4.49	-1.67	-4.15	-1.57	-2.72
2010-2012	-1.10	-9.16	-1.44	-11.00	-1.39	-3.94
2011-2013	-1.10	-1.95	-1.34	-1.37	-1.15	-2.15
2012-2014	-1.13	-2.65	-1.60	-3.02	-2.28	-0.86
2013-2015	-0.58	-1.96	-1.74	-1.95	-2.00	-1.36
2014-2016	-0.55	-2.27	-1.21	-2.56	-1.29	-3.03
2015-2017	-1.32	-3.40	-1.78	-4.42	-2.23	-2.49
2016-2018	-1.15	-2.54	-1.70	-2.93	-2.83	-1.42
Average	-1.57	-4.66	-2.80	-5.30	-3.29	-3.80
Minimum	-6.37	-13.71	-6.79	-14.10	-11.05	-13.74

Note: The calculated Maximum Drawdown here for the in-sample is the Maximum Drawdown for two years, while for the out-of-sample, the Maximum Drawdown is for one year. The units in the table are %.

As shown in Table 4.13, the average Maximum Drawdown of CA-DRL (-3.80%) is better than that of CA-GA-ST (-5.30%) and CA-ST (-4.66%) in the out-of-sample. The minimum value of Maximum Drawdown of CA-DRL, CA-GA-ST and CA-ST in the out-of-sample

during 1908-2018 are all greater than -20%, which means that investors can withstand a leverage ratio of 5 in the pairing trading in the case of avoiding assets below zero. This makes the expected return of the pairing trading strategy very promising, especially for CA-DRL, where the average annual expected return reaches 62.45% using a leverage ratio of 5.

4.7 Conclusion

This chapter uses five pairs trading strategies to conduct in-sample training and backtesting on 35 types of commodity markets in the world's major commodity markets from 1980 to 2018. The pairs are formed by the DIM and CA respectively. ST, GA and DRL are used to execute trading actions. DIM-ST, CA-ST and CA-DIM-ST are used as benchmark models. The pairs pool consists of all pairs that are formed by every two commodities. This study uses the DIM and CA methods to select the pairs from the pairs pool with data in the first two years as the in-sample, and trade according to ST strategy. Then the data of the following year is used as the out-of-sample and traded according to ST strategy. The improved trading strategies in this research include CA-GA-ST and CA-DRL strategies. CA-GA-ST strategy optimizes the thresholds of ST according to the in-sample performance. CA-DRL abandons the ST trading strategy. In CA-DRL, the price spread change of the two commodities in the pair is considered as the environment, agents of DRL play the pairs trading game based on the environment of the in-sample data. The agents are trained to execute the optimum actions based on the pairs trading rules, environment (historical information), and their state. Finally, the trained agents are traded with out-of-sample data.

The main contributions are described as follows. First, this chapter introduces a novel CA-DRL structure for the decision-making of trading actions, which obtains high returns and low risks. The CA-DRL model designed by this study tests all the possible pairs in the pairs pool, and obtains portfolios that have better performances than traditional methods. It is the first time to use CA-DRL for trading portfolios. Second, this study proposes the pre-training technique to solve two problems with DRL: falling into the local optimum and its large calculation amount. These two problems cause that DRL cannot deal with a large amount of data and only can be used for a small number of pairs. This study solves the problems and extends the use of DRL in portfolios. Third, this research introduces CA-GA-ST by using GA to optimize the parameters in ST, which slightly increases the annualized return

compared with CA-ST. The performance of CA-GA-ST is not significantly improved, which proves that the potential of ST strategy is low. As no matter how the thresholds are optimized, the trading performance does not change significantly with ST strategy.

Regarding the trading performance, the traditional DIM-ST, CA-ST and CA-DIM-ST strategies are ineffective. Though they have low risks and low capital efficiency, they have almost no excess returns. CA-GA-ST uses GA to optimize the trading thresholds of ST. It chooses narrower thresholds, which increases the in-sample performance; however, the increase in returns of the out-of-sample is not significant and the risk is higher. According to the performance of GA, this study argues that adjusting the thresholds of ST cannot significantly improve the out-of-sample performance. Additionally, this research uses DRL, which can make more flexible trading actions to replace the ST strategy. DRL is trained with the pairs selected by CA and then trades in the in-sample and out-of-sample. The result beats the traditional methods and CA-GA-ST. The average annualized return of CA-DRL is 12.49%; the Sharpe Ratio is 1.853; the Sortino Ratio is 1.053; and negative returns of the portfolio are rare, which leads to a good performance of VaR. Additionally, the MRAR(2) of CA-DRL shows the best performance with an annualized average adjusted return of 9.98% while considering the 1% deposit ratio and 2% risk-free ratio. Moreover, this study computes the Maximum Drawdowns of each pair and portfolios for CA-ST, CA-GA-ST and CA-DRL. The Maximum Drawdown in the out-of-sample of CA-DRL is -13.74%, which allows 5:1 leverage and a higher expected return (62.45%).

Chapter 5 Conclusion

5.1 Summary

This thesis introduces several models for quantitative trading from three machine learning architectures (SVM, ANNs and DRL) and applies them in three empirical chapters. Chapter 2 introduces a novel BGSA-SVM (Figure 2.5) to forecast the sign of log return for five stock indices from 1990-2016. In this model, PCA is used to reduce the dimension of the input pool. The BGSA is then employed to optimize the inputs and parameters of the SVM. The final step is to forecast the sign of the out-of-sample return by the optimized SVM. Moreover, Chapter 2 compares the forecasting accuracy and trading performances of RW, SVM, Buy-and-Hold and best predictors with that of BGSA-SVM, finding that the BGSA-SVM beats all the benchmark models.

Chapter 3 applies MLP, CNN and LSTM to the task of forecasting and trading the FTSE100 and INDU indices from 2000 to 2018. The number of neurons and hidden layers of three NNs are designed to fit the financial time series data. The average accuracy of three NNs are all higher than 50%, and the average returns of three NNs are all larger than that of Buy-and-Hold. Additionally, LSTM slightly outperforms MLP and CNN both in accuracy and average annualized returns. Chapter 3 also explores the utility of three combination methods (Simple Average, GRR and LASSO) in combining MLP, CNN and LSTM. However, the combination techniques and individual NN yield similar trading performances. A designed leverage rule in Chapter 3 (i.e., a higher forecast probability corresponds to a higher leverage ratio) increases the average annualized returns and Sharpe Ratio for CNN and LSTM.

Chapter 4 focuses on the application of machine learning to the task of optimizing the trading strategy. A novel CA-DRL is introduced to pairs trade 35 commodities from 1980-2018. CA is used to select the pairs from the pairs pool, and DRL is employed to decide the actions in the pairs trading strategy. CA-DRL beats the traditional CA-ST, DIM-ST and CA-DIM-ST method in the trading performance. Chapter 4 also introduces GA to optimize the trading thresholds of the CA-ST method. The results show that the traditional CA-ST, DIM-ST and CA-DIM-ST are close to martingale, with average annualized returns of 0.12%, 0.37% and 0.43% respectively. CA-GA-ST improves the in-sample performance significantly, slightly improves the out-of-sample performance (a 1.84% average annualized return) and increases

the risk. The limitation of CA-GA-ST indicates that the ST trading strategy does not have the potential to improve the trading performance. In addition, CA-DRL shows great performance in pairs trading with a 12.49% average annualized return and a Sharpe Ratio of 1.853. CA-DRL obtains the best performance in the annualized return, and has a similar Maximum Drawdown to CA-ST and CA-GA-ST.

To summarize the results, the machine learning methods applied to quantitative trading are practical and productive, and the models proposed by this thesis can improve the forecasting accuracy of financial time series and decide the trading actions. The machine learning algorithms make it possible for inexperienced investors to gain profits, and thus they are increasingly applied to the tasks of forecasting and trading. This study compares the machine learning models and traditional models from the perspective of investors. This thesis not only proposes new machine learning models for quantitative trading, but also helps bring academic literature in finance closer to trading applications.

5.2 Limitations

This thesis proposes new machine learning models to forecast financial time series and make trading decisions. The proposed models need to learn and simulate historical data. Machine learning models that extract effective information from historical data provide significant improvements in investment decision-making. However, limitations in machine learning models exist and three are worthy of discussion.

The first limitation is that machine learning models cannot forecast what never happened. They only learn existing patterns from historical data (in-sample). If there are some unpredictable conditions that cannot be extracted from the in-sample, the pattern in the out-of-sample will be different from the in-sample. This makes the model fail to forecast the out-of-sample and even leads to serious losses. Finding solutions for extreme situations can result in a more stable performance of machine learning models.

The second limitation is that the number of samples is limited for financial time series. Machine learning models are very effective in image recognition, speech recognition and text recognition. One important reason is that these application fields have sufficient samples

for learning. Nevertheless, the amount of financial time series data is limited, and the pattern of each market is different. This leads to a small amount of data being available for training in financial time series. Insufficient data leads to that input dimensionality not being very high, which means that a great deal of information cannot be used and the forecasting accuracy of the model decreases. Although this thesis uses some dimensionality reduction methods, such as PCA, it still loses some non-linear relationship information. Machine learning models specially designed for small samples of financial time series are needed to potentially improve the application of machine learning in finance.

The third limitation is that the calculation speed is of concern. For machine learning models, a large number of optimization tasks exist in training, and they face the problem of 'dimensional explosion'. Although in this thesis, some techniques are used to minimize the amount of computing such as a kind of heuristic optimization algorithm called BGSA (Chapter 2) and the pre-training method (Chapter 4), it still takes several days to complete each model with a personal computer. Due to the limitation of computing power, the rolling period of models in this thesis updates yearly instead of daily. If higher computing power were available, the test could be updated on a daily basis. This would increase the stability of the return in the model and improve the trading performance. With the development of computers, the limitation of computing power will become less problematic in the future.

5.3 Future work

The focus of this thesis is to apply machine learning algorithms to quantitative trading, and propose new models for financial time series data. This study has improved and optimized machine learning algorithms, such as using BGSA combined with SVM in Chapter 2, optimizing of the network structure in Chapter 3, and combining CA and DRL in Chapter 4. These optimized models are applied to the tasks of financial forecasting and trading, in order to examine the effectiveness of the models. These models are designed for financial time series and can be applied to the trading of other assets. Additionally, all these models can be further developed in the future. There are three aspects to potential future work based on the algorithms in this thesis.

The first aspect for improvement is the parameter optimization for initial parameters and the

connection weights in neural networks. This thesis optimizes the inputs and parameters of SVM by BGSA. The empirical results support the view that initial parameters have a significant effect on the performance of machine learning models. In the future, more advanced optimization method and more initial parameters can be adopted. Regarding the optimization of connection weights, this thesis uses AdamOptimizer and SGD optimization methods. The algorithms that can reduce the computing amount need to be developed in the future. A smaller calculation amount increases the frequency of the forecasting, and thus offers a more stable trading performance.

The analysis of inputs is the second aspect, as finding the main factors that affect the price of financial assets is crucial to forecasting accuracy. This thesis uses the inputs suggested by relevant literature. Dimensionality reduction methods and models that are able to deal with high dimensional inputs are employed. The forecasting accuracy of MLP, CNN and LSTM is very similar in Chapter 3. This is because the information contained in the inputs is limited and restricts the performance of the NNs. It is necessary to introduce more useful information to improve forecasting accuracy. The relationship of inputs and the forecasting target needs to be further considered with regard to financial knowledge, and then useful inputs need to be selected to form the inputs pool; this will significantly improve the forecasting performance.

The third aspect is the optimization of trading strategies. Chapters 2 and 3 use naive trading strategies, while Chapter 4 adopts the DRL to make trading decisions and uses leverage based on the forecasting probability. If stochastic mathematics is used to generate a continuous method of leverage selection, the return will significantly rise. Therefore, the combination of stochastic mathematical methods and machine learning methods for quantitative trading will be a promising future research direction.

Appendices

Appendix A (Chapter 2)

Appendix A.1 Inputs and dataset

The inputs pool is presented in Appendix A.1.1, and the start and end dates of every prediction are shown in Appendix A.1.2.

A.1.1 Inputs pool

The inputs of BGSA-SVM include %K, Momentum, Williams' %R, Disparity5, OSCP, CCI, Return, Moving average, Volume, AR and ARMA. Their descriptions are provided in Table A.1.

Table A.1: Input pool for Chapter 2

Name	Description	Formula	Total individual forecasts
%K(q)	Stochastic %K. It compares where a security's price closed relative to its price range over a given time period.	$\frac{C_{t-q} - LL_{t-4-q}}{HH_{t-4-q} - LL_{t-4-q}} * 100$ Where : LL_t and HH_t mean lowest low and highest high in the last t days respectively. $q = 1, \dots, 10$	10
Momentum(n)	It measures the amount that a security's price has changed over a given time span.	$C_{t-1} - C_{t-n-1}$ Where: $n = 1, \dots, 10$	10
Williams' %R(a)	Larry William's %R. It is a momentum indicator that measures overbought/oversold levels.	$\frac{H_{t-5a} - C_{t-1}}{H_{t-5a} - L_{t-5a}}$ Where $a = 1 \dots 5$	5
Disparity5(l)	5n-day disparity. It means the distance of the current price and the moving average of 5 days.	$\frac{C_{t-1}}{MA_{5l}(t-1)} * 100$ Where: $l = 1 \dots 5$	5
OSCP	Price oscillator. It displays the difference between two moving averages of a security's price.	$\frac{MA_5(t-m) - MA_{10}(t-m)}{MA_5(t-m)}$	1

Name	Description	Formula	Total individual forecasts
CCI (d)	Commodity channel index. It measures the variation of a security's price from its statistical mean.	$\frac{(M_{t-d} - SM_{t-d})}{(0.015D_{t-d})}$ Where $M_t = (H_t + L_t + C_t)/3$ $SM_t = \frac{\sum_{i=1}^{n} M_{t-i+1}}{n}$ $D_t = \frac{\sum_{i=1}^{n} \lvert M_{t-i+1} - SM_t \rvert}{n}$ $d = 1 \dots 5$	5
Return(k)	Last days' log return	$\log\left(\frac{C_{t-k}}{C_{t-k-1}}\right)$ where: $k = 1 \dots 20$	20
Moving average (p)	Moving average	$MA_{5p}(t-1)$ Where: $p = 1 \dots 10$	10
Volume (i)	Last days' trading volume	V_{t-i} Where: $i = 1 \dots 10$	10
Volatility(h)	Price volatility	$FV_{10h}(t-1)$ Where: $h = 1 \dots 3$	3
$AR(m)$	Close Price predicted by $AR(m)$ model	$\beta_0 + \sum_{i=1}^{m} \beta_i C_{t-i-1}$ Where: $m = 1, \dots, 10$ β_0, β_i are the regression coefficients.	10
$ARMA(m', n')$	Close Price predicted by $ARMA(m', n')$ model	$\varphi_0 + \sum_{j=1}^{m'} \varphi_j C_{t-j-1} + \varepsilon_0 + \sum_{k=1}^{n'} \gamma_k \varepsilon_{t-k-1}$ Where: $m', n' = 1, \dots, 10$ φ_0, φ_j are the regression coefficients. $\varepsilon_0, \varepsilon_{t-k-1}$ are the residual terms. γ_k is the weight of the residual terms.	100
Sum dimension			189

Note: C_t is the close price at time t. L_t is the lowest price at time t. H_t is the highest price at time t. $MA_n(t)$ is the moving average of n days at time t. V_t is the trading volume at time t. $FV_n(t)$ is the volatility in last n days at time t.

A.1.2 Start and end dates of predictions

Table A.2 indicates the start and end dates of the training sample, test sample and out-of-sample of every test.

Appendix A.2 The accuracy and trading performances of BGSA-SVM and Benchmark models

This section shows all the empirical results for BGSA-SVM and benchmark models. Table A.3, Table A.4, Table A.5, Table A.6 and Table A.7 display the accuracy of the training sample, test sample and out-of-sample of S&P500, FTSE100, NKY, DAX and CAC40 respectively.

Table A.2: The start and end dates of the training sample, test sample and out-of-sample set

		FTSE100	S&P500	NKY	CAC	DAX
F1	Training sample	1993/1/21-1998/4/14	1990/3/14-1997/4/2	1993/1/22-1997/12/30	1993/3/10-1998/1/29	1993/3/15-1998/4/15
	Test sample	1998/4/15-2000/7/24	1997/4/3-1999/5/28	1998/1/5-2000/7/14	1998/1/30-2000/3/14	1998/4/16-2000/7/25
	Out-of-sample	2000/7/25-2001/10/1	1999/6/1-2000/6/12	2000/7/17-2002/4/25	2000/3/15-2001/5/10	2000/7/26-2001/8/29
F2	Training sample	1994/4/25-1999/5/27	1991/5/29-1998/4/16	1994/3/30-1999/4/21	1994/12/8-1999/3/4	1994/9/30-1999/6/18
	Test sample	1999/5/28-2001/10/1	1998/4/17-2000/6/12	1999/4/22-2002/4/25	1999/3/5-2001/5/10	1999/6/21-2001/8/29
	Out-of-sample	2001/10/2-2002/12/20	2000/6/13-2001/6/26	2002/4/26-2003/8/4	2001/5/11-2002/8/30	2001/8/30-2002/10/23
F3	Training sample	1995/10/27-2000/7/24	1993/11/24-1999/5/28	1995/7/6-2000/7/14	1996/1/24-2000/3/14	1995/10/17-2000/7/25
	Test sample	2000/7/25 2002/12/20	1999/6/1-2001/6/26	2000/7/17-2003/8/4	2000/3/15-2002/8/30	2000/7/26-2002/10/23
	Out-of-sample	2002/12/23-2004/3/1	2001/6/27-2002/10/11	2003/8/5-2004/10/26	2002/9/2-2003/10/7	2002/10/24-2003/12/4
F4	Training sample	1996/12/24-2001/10/1	1996/2/16-2000/6/12	1996/10/7-2002/4/25	1997/1/28-2001/5/10	1996/10/31-2001/8/29
	Test sample	2001/10/2-2004/3/1	2000/6/13-2002/10/11	2002/4/26-2004/10/26	2001/5/11-2003/10/7	2001/8/30-2003/12/4
	Out-of-sample	2004/3/2-2005/6/9	2002/10/14-2003/10/29	2004/10/27-2005/11/7	2003/10/8-2004/11/10	2003/12/5-2004/12/17
F5	Training sample	1998/4/15-2002/12/20	1997/4/3-2001/6/26	1998/1/5-2003/8/4	1998/1/30-2002/8/30	1998/4/16-2002/10/23
	Test sample	2002/12/23-2005/6/9	2001/6/27-2003/10/29	2003/8/5-2005/11/7	2002/9/2-2004/11/10	2002/10/24-2004/12/17
	Out-of-sample	2005/6/10-2006/11/8	2003/10/30-2004/11/12	2005/11/8-2007/3/16	2004/11/11-2005/12/28	2004/12/20-2006/1/10
F6	Training sample	1999/5/28-2004/3/1	1998/4/17-2002/10/11	1999/4/22-2004/10/26	1999/3/5-2003/10/7	1999/6/21-2003/12/4
	Test sample	2004/3/2-2006/11/8	2002/10/14-2004/11/12	2004/10/27-2007/3/16	2003/10/8-2005/12/28	2003/12/5-2006/1/10
	Out-of-sample	2006/11/9-2008/4/21	2004/11/15-2005/12/27	2007/3/19-2008/4/24	2005/12/29-2007/5/14	2006/1/11-2007/2/16
F7	Training sample	2000/7/25-2005/6/9	1999/6/1-2003/10/29	2000/7/17-2005/11/7	2000/3/15-2004/11/10	2000/7/26-2004/12/17
	Test sample	2005/6/10-2008/4/21	2003/10/30-2005/12/27	2005/11/8-2008/4/24	2004/11/11-2007/5/14	2004/12/20-2007/2/16
	Out-of-sample	2008/4/22-2009/12/1	2005/12/28-2007/1/18	2008/4/25-2009/9/14	2007/5/15-2008/5/22	2007/2/19-2008/2/18
F8	Training sample	2001/10/2-2006/11/8	2000/6/13-2004/11/12	2002/4/26-2007/3/16	2001/5/11-2005/12/28	2001/8/30-2006/1/10
	Test sample	2006/11/9-2009/12/1	2004/11/15-2007/1/18	2007/3/19-2009/9/14	2005/12/29-2008/5/22	2006/1/11-2008/2/18
	Out-of-sample	2009/12/2-2013/2/19	2007/1/19-2008/3/10	2009/9/15-2010/10/29	2008/5/23-2009/9/17	2008/2/19-2009/2/16
F9	Training sample	2002/12/23-2008/4/21	2001/6/27-2005/12/27	2003/8/5-2008/4/24	2002/9/2-2007/5/14	2002/10/24-2007/2/16
	Test sample	2008/4/22-2013/2/19	2005/12/28-2008/3/10	2008/4/25-2010/10/29	2007/5/15-2009/9/17	2007/2/19-2009/2/16
	Out-of-sample	2013/2/20-2016/4/20	2008/3/11-2009/6/12	2010/11/1-2011/12/26	2009/9/18-2010/10/26	2009/2/17-2010/4/13
F10	Training sample		2002/10/14-2007/1/18	2004/10/27-2009/9/14	2003/10/8-2008/5/22	2003/12/5-2008/2/18
	Test sample		2007/1/19-2009/6/12	2009/9/15-2011/12/26	2008/5/23-2010/10/26	2008/2/19-2010/4/13
	Out-of-sample		2009/6/15-2010/7/12	2011/12/27-2013/3/6	2010/10/27-2012/3/27	2010/4/14-2011/4/18
F11	Training sample		2003/10/30-2008/3/10	2005/11/8-2010/10/29	2004/11/11-2009/9/17	2004/12/20-2009/2/16
	Test sample		2008/3/11-2010/7/12	2010/11/1-2013/3/6	2009/9/18-2012/3/27	2009/2/17-2011/4/18
	Out-of-sample		2010/7/13-2011/7/11	2013/3/7-2014/6/26	2012/3/28-2013/9/23	2011/4/19-2012/7/23

Note: Due to missing data and holidays, the start and end dates of the five indices (FTSE, S&P500, NKY, CAC and DAX) are different.

Table A.3: Accuracy for the S&P500

		F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
BGSA-SVM	Training sample	0.7153	0.6052	0.6409	0.7123	0.5992	0.5883	0.6161	0.6409	0.6419	0.6280	0.7361
	Test sample	0.6171	0.6190	0.6528	0.6429	0.6270	0.5913	0.6171	0.6111	0.6369	0.6429	0.6528
	Out-of-sample	0.5516	0.5516	0.5437	0.4881	0.5238	0.5278	0.5119	0.5278	0.5595	0.5159	0.5437
RW	Training sample	0.4911	0.5149	0.5099	0.5159	0.5218	0.5129	0.4931	0.5188	0.5169	0.4931	0.4980
	Test sample	0.5000	0.4940	0.5000	0.5040	0.4603	0.5238	0.5496	0.5000	0.5179	0.5218	0.5417
	Out-of-sample	0.4802	0.4841	0.5119	0.5357	0.5278	0.5516	0.4960	0.4683	0.4960	0.4881	0.4286
SVM	Training sample	1.0000	1.0000	1.0000	0.9990	0.9881	0.9206	1.0000	0.8075	1.0000	1.0000	1.0000
	Test sample	0.5437	0.5119	0.5020	0.4722	0.5040	0.5337	0.4444	0.5575	0.5357	0.5317	0.5417
	Out-of-sample	0.5079	0.4683	0.4563	0.5437	0.5516	0.5079	0.4524	0.5317	0.5357	0.5278	0.5000
$best_1$	Training sample	0.5466	0.4871	0.5298	0.5268	0.5238	0.5179	0.5169	0.5109	0.5208	0.5169	0.5466
	Test sample	0.5298	0.4841	0.5317	0.4980	0.5258	0.5238	0.5159	0.5099	0.5179	0.5298	0.5119
	Out-of-sample	0.5198	0.4921	0.5317	0.5357	0.5119	0.5198	0.5000	0.5357	0.5794	0.5119	0.4960
$best_2$	Training sample	0.5317	0.5030	0.5585	0.5367	0.5446	0.5377	0.5397	0.5169	0.5030	0.5248	0.5159
	Test sample	0.5437	0.5060	0.4881	0.5238	0.5476	0.5456	0.5397	0.5060	0.5337	0.5337	0.5258
	Out-of-sample	0.5079	0.4921	0.4246	0.5635	0.5278	0.5516	0.5357	0.5754	0.5714	0.4921	0.4881
Buy-and-Hold	Training sample	0.5317	0.5437	0.5585	0.5308	0.5159	0.4792	0.4831	0.4940	0.5169	0.5476	0.5456
	Test sample	0.5437	0.5119	0.4881	0.4464	0.4782	0.5417	0.5556	0.5536	0.5357	0.5298	0.5536
	Out-of-sample	0.5079	0.4683	0.4246	0.5317	0.5516	0.5595	0.5476	0.5238	0.5357	0.5714	0.5754

Note: This table shows the accuracy of all the models for S&P500 in Chapter 2. The SVM has significant overfitting, as the accuracy is almost 100% in the training sample for all periods and around 50% in the out-of-sample.

Table A.4: Accuracy for the FTSE100

		F1	F2	F3	F4	F5	F6	F7	F8	F9
BGSA-SVM	Training sample	0.7173	0.7331	0.6171	0.6260	0.6339	0.6290	0.6696	0.6796	0.6012
	Test sample	0.6508	0.6329	0.6171	0.6171	0.6230	0.6131	0.6012	0.6111	0.6389
	Out-of-sample	0.5278	0.5317	0.5079	0.5357	0.5198	0.4325	0.5119	0.5000	0.5159
RW	Training sample	0.4732	0.5129	0.5407	0.5198	0.4772	0.4931	0.4752	0.5060	0.5069
	Test sample	0.5298	0.4940	0.4782	0.4861	0.4782	0.5119	0.5317	0.4861	0.4980
	Out-of-sample	0.4762	0.5079	0.4802	0.4563	0.5278	0.5357	0.5238	0.4802	0.4881
SVM	Training sample	1.0000	0.8085	1.0000	0.8175	1.0000	1.0000	0.9018	0.8482	0.7192
	Test sample	0.5893	0.5933	0.4623	0.5139	0.5278	0.5337	0.5119	0.5337	0.5635
	Out-of-sample	0.5159	0.5040	0.5079	0.5079	0.4960	0.5556	0.5159	0.5040	0.4921
$best_1$	Training sample	0.5407	0.5258	0.5069	0.4931	0.5089	0.5327	0.5317	0.5228	0.5030
	Test sample	0.5198	0.4782	0.5079	0.5496	0.5317	0.4940	0.5099	0.5298	0.5119
	Out-of-sample	0.5000	0.5159	0.5357	0.4762	0.4722	0.5278	0.5317	0.5159	0.4722
$best_2$	Training sample	0.5456	0.5169	0.5099	0.4921	0.5218	0.5248	0.5228	0.5089	0.5198
	Test sample	0.5099	0.5437	0.5139	0.5258	0.5298	0.4742	0.5040	0.5317	0.5238
	Out-of-sample	0.5278	0.5317	0.5317	0.4405	0.4921	0.5000	0.5437	0.4841	0.4802
Buy-and-Hold	Training sample	0.5437	0.5456	0.5486	0.5268	0.5040	0.5109	0.5169	0.5308	0.5308
	Test sample	0.5258	0.5159	0.4821	0.5060	0.5516	0.5556	0.5099	0.4841	0.5099
	Out-of-sample	0.4921	0.4722	0.5397	0.5635	0.5476	0.4722	0.4960	0.5238	0.5000

Note: This table shows the accuracy of all the models for FTSE100, of which the SVM is overfitting.

Table A.5: Accuracy for the NKY

		F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
BGSA-SVM	Training sample	0.6726	0.6538	0.5952	0.6260	0.6270	0.6835	0.6528	0.6706	0.6329	0.6875	0.8095
	Test sample	0.6151	0.6190	0.6111	0.6270	0.5933	0.6091	0.6111	0.6290	0.6409	0.6369	0.6190
	Out-of-sample	0.5595	0.5476	0.5357	0.5079	0.5000	0.5397	0.5119	0.5238	0.5317	0.5357	0.5079
RW	Training sample	0.4792	0.5069	0.5149	0.5020	0.4613	0.4960	0.5020	0.5159	0.4861	0.5129	0.5327
	Test sample	0.4563	0.5040	0.5139	0.5020	0.5079	0.4881	0.4782	0.5496	0.4980	0.5278	0.5060
	Out-of-sample	0.4643	0.5119	0.5079	0.5317	0.5635	0.5516	0.5397	0.4603	0.4405	0.5159	0.5873
SVM	Training sample	1.0000	0.8978	1.0000	1.0000	0.9871	0.8591	0.9980	1.0000	1.0000	1.0000	0.8363
	Test sample	0.5456	0.5556	0.4563	0.5258	0.4940	0.5020	0.5060	0.5298	0.5218	0.5595	0.5397
	Out-of-sample	0.4921	0.4921	0.4881	0.4603	0.5317	0.5040	0.5437	0.5516	0.4802	0.5198	0.5278
$best_1$	Training sample	0.4851	0.5079	0.5139	0.5159	0.4782	0.4950	0.5079	0.5129	0.5188	0.5089	0.5079
	Test sample	0.5218	0.4960	0.4901	0.4722	0.5179	0.5099	0.5020	0.4980	0.4742	0.4921	0.5337
	Out-of-sample	0.5079	0.4722	0.4722	0.5397	0.5317	0.4722	0.5159	0.4683	0.5159	0.5516	0.4881
$best_2$	Training sample	0.5010	0.5099	0.5159	0.5149	0.5208	0.4980	0.4861	0.4861	0.4921	0.5139	0.5268
	Test sample	0.5218	0.4960	0.5040	0.4901	0.5179	0.4841	0.5119	0.5099	0.5079	0.5099	0.5040
	Out-of-sample	0.4960	0.5119	0.4960	0.5635	0.4722	0.5238	0.4960	0.5198	0.4643	0.5238	0.5159
Buy-and-Hold	Training sample	0.5050	0.4960	0.5020	0.4792	0.4742	0.4752	0.4851	0.5020	0.5129	0.5079	0.4960
	Test sample	0.4921	0.4762	0.4563	0.4742	0.5139	0.5298	0.5119	0.4861	0.4802	0.5099	0.5357
	Out-of-sample	0.4524	0.4603	0.4881	0.5397	0.5198	0.5040	0.4683	0.4921	0.5278	0.5437	0.5397

Note: This table shows the accuracy of all the models for NKY, of which the SVM is overfitting.

Table A.6: Accuracy for the DAX

		F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
BGSA-SVM	Training sample	0.6190	0.6111	0.7063	0.7252	0.6875	0.7004	0.6300	0.7500	0.6389	0.6667	0.6806
	Test sample	0.6190	0.6052	0.6091	0.6310	0.6389	0.6190	0.6349	0.6290	0.6389	0.6131	0.6091
	Out-of-sample	0.4643	0.5278	0.4881	0.5992	0.5040	0.4921	0.5714	0.5238	0.5198	0.5079	0.5238
RW	Training sample	0.5208	0.5327	0.4732	0.5109	0.4911	0.4950	0.5278	0.4712	0.5179	0.4940	0.4950
	Test sample	0.5020	0.4663	0.5278	0.4980	0.4901	0.4901	0.5556	0.5020	0.4821	0.5119	0.5179
	Out-of-sample	0.5437	0.4603	0.5079	0.4722	0.4881	0.4921	0.4683	0.5119	0.5357	0.4484	0.4683
SVM	Training sample	1.0000	1.0000	0.8710	0.8423	0.8909	1.0000	0.9950	1.0000	0.9980	1.0000	1.0000
	Test sample	0.5397	0.5397	0.5159	0.5853	0.5794	0.5516	0.5675	0.5774	0.5516	0.5198	0.5456
	Out-of-sample	0.4643	0.4762	0.4841	0.5675	0.5000	0.5516	0.5000	0.5198	0.5198	0.5040	0.4921
$best_1$	Training sample	0.5387	0.5198	0.5536	0.5456	0.5397	0.5129	0.5179	0.5288	0.5109	0.5268	0.5198
	Test sample	0.5556	0.5357	0.5238	0.5139	0.4901	0.5099	0.5278	0.4821	0.5218	0.4940	0.5020
	Out-of-sample	0.5159	0.5159	0.4960	0.4841	0.5357	0.5079	0.5317	0.5476	0.4722	0.5040	0.5317
$best_2$	Training sample	0.5298	0.5655	0.5327	0.5079	0.5030	0.5397	0.5069	0.5119	0.5258	0.5119	0.5407
	Test sample	0.5575	0.5119	0.5179	0.5060	0.5218	0.5258	0.5218	0.5218	0.5536	0.5218	0.4623
	Out-of-sample	0.5159	0.5317	0.5595	0.5159	0.5238	0.4683	0.5357	0.5357	0.4206	0.5238	0.5119
Buy-and-Hold	Training sample	0.5675	0.5536	0.5635	0.5367	0.4931	0.5000	0.4940	0.5278	0.5605	0.5635	0.5476
	Test sample	0.5317	0.5119	0.4544	0.4881	0.5337	0.5675	0.5873	0.5595	0.5079	0.5198	0.5456
	Out-of-sample	0.4643	0.4444	0.5317	0.5357	0.5992	0.5754	0.5437	0.4722	0.5675	0.5238	0.4921

Note: This table shows the accuracy of all the models for DAX, of which the SVM is overfitting.

Table A.7: Accuracy for the CAC

		F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
BGSA-SVM	Training sample	0.6935	0.6448	0.6587	0.6607	0.6300	0.6012	0.7897	0.6270	0.6111	0.6052	0.6796
	Test sample	0.6329	0.6052	0.6052	0.6270	0.6091	0.5913	0.5933	0.6270	0.6270	0.6171	0.5873
	Out-of-sample	0.4881	0.5397	0.5040	0.5079	0.5476	0.4762	0.5714	0.5913	0.5833	0.5278	0.5195
RW	Training sample	0.5079	0.5099	0.5129	0.5069	0.5000	0.4970	0.4692	0.4990	0.4861	0.5000	0.5000
	Test sample	0.4563	0.5099	0.4623	0.5020	0.5119	0.5040	0.4603	0.5020	0.4861	0.5119	0.5516
	Out-of-sample	0.4841	0.5278	0.4643	0.4881	0.4841	0.4524	0.5357	0.5397	0.5238	0.5198	0.5159
SVM	Training sample	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.8333	1.0000	1.0000	0.9286	1.0000
	Test sample	0.5119	0.5159	0.4524	0.4623	0.5139	0.4762	0.5913	0.4940	0.4940	0.5258	0.4821
	Out-of-sample	0.4921	0.4405	0.4841	0.5437	0.4921	0.5357	0.5516	0.5278	0.5198	0.5675	0.5159
$best_1$	Training sample	0.5119	0.5258	0.5089	0.5179	0.4970	0.5278	0.5387	0.5248	0.5357	0.5188	0.5208
	Test sample	0.4643	0.5079	0.5218	0.5476	0.5615	0.5476	0.5456	0.5119	0.5397	0.5437	0.5456
	Out-of-sample	0.5635	0.5040	0.5913	0.5317	0.5635	0.4802	0.5556	0.5635	0.5397	0.5516	0.5119
$best_2$	Training sample	0.5456	0.5198	0.4881	0.5050	0.5050	0.5179	0.5427	0.5337	0.5248	0.5179	0.5218
	Test sample	0.4722	0.5139	0.5278	0.5238	0.5377	0.5694	0.5020	0.4901	0.5536	0.5496	0.5079
	Out-of-sample	0.5397	0.5040	0.5714	0.5317	0.5754	0.4841	0.5317	0.5437	0.5357	0.5159	0.4722
Buy-and-Hold	Training sample	0.5079	0.5347	0.5546	0.5308	0.5099	0.4891	0.4831	0.4970	0.5228	0.5208	0.5069
	Test sample	0.5675	0.5159	0.4524	0.4623	0.5139	0.5317	0.5317	0.5099	0.4821	0.4960	0.4821
	Out-of-sample	0.4643	0.4405	0.4841	0.5437	0.5198	0.5437	0.4762	0.4881	0.5040	0.4603	0.5476

Note: This table shows the accuracy of all the models for CAC, of which the SVM is overfitting.

Appendix A.3 The Sharpe Ratio and other tests

In this Appendix, details of the Sharpe Ratio, Chi-square test and volatility analysis in this study are presented.

A.3.1 The Sharpe Ratio

The Sharpe Ratio is a way to examine the performance of an investment by adjusting for its risk (Sharpe, 1970). The Sharpe Ratio is defined as:

$$Sharpe\ Ratio = \frac{annual\ return - risk\ free\ ratio}{standard\ deviation} \qquad (A.1)$$

In recent years, the risk-free ratio has been low for the countries in the study, and thus the risk-free ratio in Chapter 2 is assumed to be zero.

A.3.2 The Chi-square test for BGSA-SVM

In 53 predictions, BGSA-SVM is the best of the six models 17 times. If the BGSA-SVM is invalid, then the expectation it would be the best model should be less than 8.83 (53/6) times. Thus, a chi-square test for BGSA-SVM with the expectation of random models is described as follows.

H0: the distribution of results that BGSA-SVM is the best forecasting model compared with all benchmark models is the same with the distribution of the expectation of random models.

$$\chi^2 = \sum \frac{(real\ number - T)^2}{T} \qquad (A.2)$$

$$\mathrm{T1} = 53/6, \mathrm{T2} = 265/6, \mathrm{T3} = 56/6, \mathrm{T4} = 265/6, \chi^2 = 9.0604$$

Table A.8 shows the upper percentage points of the χ^2 distribution. As shown in Table A.9, the value of χ^2 can reject the H0 at a 99% confidence level, which means that BGSA-SVM is valid.

Table A.8: The Chi-square test BGSA-SVM with the expectation of random models

	Positive	Negative	Sum
BGSA-SVM	17	36	53
The expectation of random models	$53/6 * \mathrm{n}$	$265/6 * \mathrm{n}$	n
Sum	$19 + 53/6$	$34 + 265/6$	106

Note: n tends to $+\infty$.

Table A.9: Upper percentage points of the χ^2 distribution

Degrees of freedom	Pr		
	0.05	0.01	0.001
1	3.84	6.63	10.83
2	5.99	9.21	13.81
3	7.81	11.34	16.27
4	9.49	13.28	18.47
5	11.07	15.09	20.52
6	12.59	16.81	22.46
7	14.07	18.48	24.32
8	15.51	20.09	26.12
9	16.92	21.67	27.88
10	18.31	23.21	29.59

Note: The degree of freedom is 1 for the test.

A.3.3. The relationship between volatility and annualized returns of BGSA-SVM

Chapter 2 runs the regression of BGSA-SVM's annualized return on volatility to investigate their relationship. Table A.10 shows that the p-value of the estimated coefficient of volatility is 0.168 and the R-squared is only 0.037. This means that the volatility is not related to the annualized return of BGSA-SVM and cannot explain the change of the annualized return.

Table A.10: The regression result of the annualized return of BGSA-SVM and volatility

Dependent Variable: the annualized return of BGSA-SVM
Method: Least Squares
Date: 07/12/18 Time: 17:37
Sample: 53 times test
Included observations: 53 after adjustments

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	1.913591	6.079529	0.314760	0.7542
Daily Standard deviation	6.108195	4.365973	1.399045	0.1679

R-squared	0.036960	Mean dependent var	9.885362
Adjusted R-squared	0.018077	S.D. dependent var	15.57350
S.E. of regression	15.43209	Akaike info criterion	8.347781
Sum squared resid	12145.62	Schwarz criterion	8.422132
Log likelihood	-219.2162	Hannan-Quinn criter.	8.376373
F-statistic	1.957328	Durbin-Watson stat	1.984038
Prob(F-statistic)	0.167852		

Note: EViews is used to run the regression.

Appendix B (Chapter 3)

Appendix B.1 Inputs and dataset

In this Appendix, the inputs pool and the start and end dates of every forecast are presented.

B.1.1 Inputs pool

There is some small difference between the inputs for MLP, CNN and LSTM, which are described in Section 3.4. The summary of the inputs is presented in Table B.1.

Table B.1: Input pool for Chapter 3

Name	Description	Formula	Total individual forecasts
Return (k)	Last days' log return	$\log(\frac{C_{t-k}}{C_{t-k-1}})$ Where: $k = 1 \dots 10$	10
H_{t-k} (k)	Daily highest price	H_{t-k} Where: $k = 1 \dots 10$	10
L_{t-k} (k)	Daily lowest price	L_{t-k} Where: $k = 1 \dots 10$	10
G (k)	Last day's Gold price (COMEX)	G_{t-k} Where: $k = 1 \dots 10$	10
O (k)	Last day's Oil price (NYMEX)	O_{t-k} Where: $k = 1 \dots 10$	10
USD/JPY (k)	Last day's USD/JPY exchange rates	U/J_{t-k} Where: $k = 1 \dots 10$	10(only for INDU)
USD/EUR (k)	Last day's USD/EUR exchange rates	U/E_{t-k} Where: $k = 1 \dots 10$	10(only for INDU)
USD/GBP (k)	Last day's USD/GBP exchange rates	U/G_{t-k} Where: $k = 1 \dots 10$	10
GBP/JPY (k)	Last day's GBP/JPY exchange rates	G/J_{t-k} Where: $k = 1 \dots 10$	10(only for FTSE)
GBP/EUR (k)	Last day's GBP/EUR exchange rates	G/E_{t-k} Where: $k = 1 \dots 10$	10(only for FTSE)
Moving average (n)	Moving average	$MA_{5n}(t-1)$ Where: $n = 1 \dots 10$	10
$AR(m)$	Close price predicted by $AR(m)$ model	$\beta_0 + \sum_{i=1}^{m} \beta_i C_{t-i-1}$ Where: $m = 1, \dots, 10$ β_0 and β_i are the regression coefficients	10
Sum Variables			80

Note: C_t is the close price at time t. $MA_n(t)$ is the moving average of n days at time t.

B.1.2 Start day and end day for forecasts

Table B.2 indicates the start & end dates of the in-sample and out-of-sample of every test. The length of the in-sample is 5 trading years (1260 trading days), and the length of the out-of-sample is 1 trading year (252 trading days).

Table B.2: The dataset for tests

		Start date & end date for INDU and FTSE100
F1	In-sample	2000/1/4-2004/11/1
	Out-sample	2004/11/2-2005/10/19
F2	In-sample	2000/12/21-2005/10/19
	Out-sample	2005/10/20-2006/10/6
F3	In-sample	2001/12/10-2006/10/6
	Out-sample	2006/10/9-2007/9/25
F4	In-sample	2002/11/27-2007/9/25
	Out-sample	2007/9/26-2008/9/11
F5	In-sample	2003/11/14-2008/9/11
	Out-sample	2008/9/12-2009/8/31
F6	In-sample	2004/11/2-2009/8/31
	Out-sample	2009/9/1-2010/8/18
F7	In-sample	2005/10/20-2010/8/18
	Out-sample	2010/8/19-2011/8/5
F8	In-sample	2006/10/9-2011/8/5
	Out-sample	2011/8/8-2012/7/24
F9	In-sample	2007/9/26-2012/7/24
	Out-sample	2012/7/25-2013/7/11
F10	In-sample	2008/9/12-2013/7/11
	Out-sample	2013/7/12-2014/6/30
F11	In-sample	2009/9/1-2014/6/30
	Out-sample	2014/7/1-2015/6/17
F12	In-sample	2010/8/19-2015/6/17
	Out-sample	2015/6/18-2016/6/3
F13	In-sample	2011/8/8-2016/6/3
	Out-sample	2016/6/6-2017/5/23
F14	In-sample	2012/7/25-2017/5/23
	Out-sample	2017/5/24-2018/5/10

Note: The data for INDU and FTSE100 are downloaded from Bloomberg.

Appendix B.2 Performance of eight classification categories

This Appendix presents the results of eight categories classification. The accuracy and trading performance of eight classification categories are shown in Appendix B.2.1 Appendix B.2.2 respectively.

B.2.1 Statistical accuracy of eight classification categories of NNs

The statistical accuracy of the eight classification categories is presented in Table B.3 (for

FTSE100) and Table B.4 (for INDU). This study classifies the daily return of the stock index by eight classification categories because in the previous literature the prediction results of more classification categories are better than that of binary classification, especially the eight categories (Olson & Mossman, 2003). However, the 4 classification categories and 16 classification categories are not examined due to the amount of work that would be involved and the limited time available.

Table B.3: Accuracy for eight classification forecasts for the out-of-sample for the FTSE100

	MLP		MLP with AR & MA		CNN		LSTM	
	Classes accuracy	Sign accuracy	Classes accuracy	Sign accuracy	Classes accuracy	Sign accuracy	Classes accuracy	Sign accuracy
F1	0.2738	0.5476	0.3175	0.5357	0.4405	0.5437	0.2865	**0.4841**
F2	0.2421	0.5317	0.25	**0.4921**	0.3214	0.5040	0.3063	0.5159
F3	0.2222	0.5238	0.2024	0.5159	0.3611	**0.4921**	0.2825	0.5040
F4	0.1587	0.5437	0.1825	0.5476	0.2500	0.5794	0.2548	0.5516
F5	0.1548	0.5119	0.1468	**0.4921**	0.2659	0.5278	0.3627	**0.4960**
F6	0.2262	0.5119	0.1746	0.5238	0.3214	0.5079	0.2063	0.5357
F7	0.2103	0.5079	0.2063	0.5119	0.3294	0.5079	0.3389	**0.4921**
F8	0.1944	**0.4841**	0.1786	0.5476	0.4206	0.5317	0.1865	0.5238
F9	0.2103	0.5516	0.2460	**0.4960**	0.4087	0.5516	0.2429	0.5040
F10	0.2619	0.5198	0.2381	0.5238	0.4087	**0.4762**	0.2786	0.5079
F11	0.2500	**0.4960**	0.2540	**0.4762**	0.4087	0.5159	0.2429	0.5595
F12	0.1786	**0.4841**	0.1984	**0.4841**	0.3492	0.5556	0.1865	0.5278
F13	0.3254	0.5278	0.3135	0.5000	0.4762	**0.4921**	0.2381	0.5278
F14	0.3095	**0.4960**	0.3175	0.5357	0.3690	0.5119	0.2619	**0.4841**
Average	0.2299	0.5170	0.2304	0.5130	0.3665	0.5213	0.2625	0.5153

Note: The out-of-sample sign accuracy lower than 50% are in bold. Classes accuracy is the ratio that eight classification forecasts are right. Sign accuracy is the ratio that the sign of eight classification forecasts is right.

Table B.4: Accuracy for eight classification forecasts for the out-of-sample for the INDU

	MLP		MLP with AR & MA		CNN		LSTM	
	Classes accuracy	Sign accuracy	Classes accuracy	Sign accuracy	Classes accuracy	Sign accuracy	Classes accuracy	Sign accuracy
F1	0.3730	0.5357	0.3175	0.5278	0.4206	0.5040	0.2421	0.5040
F2	0.2103	0.5079	0.2500	**0.4960**	0.2103	0.5079	0.2341	0.5516
F3	0.3532	**0.4881**	0.2024	**0.4802**	0.3611	**0.4921**	0.2937	**0.4921**
F4	0.1905	0.5437	0.1825	0.5278	0.2500	0.5794	0.2262	0.5397
F5	0.2421	0.5317	0.1468	0.5000	0.2659	0.5278	0.2302	0.5040
F6	0.2937	**0.4921**	0.1746	0.5357	0.3056	**0.4881**	0.2857	0.5238
F7	0.2659	0.5000	0.2063	0.5040	0.3294	0.5079	0.2302	0.5079
F8	0.1984	0.5516	0.1786	0.5357	0.4246	0.5357	0.2341	0.5238
F9	0.2619	0.5437	0.2460	0.5238	0.4008	0.5516	0.2659	0.5119
F10	0.3690	0.5119	0.2381	**0.4921**	0.3095	0.5873	0.3333	**0.4960**
F11	0.2540	0.5000	0.254	**0.4921**	0.4087	0.5159	0.3770	0.5317
F12	0.3016	0.5040	0.1984	0.5079	0.2619	0.5238	0.3175	**0.4960**
F13	0.2976	**0.4762**	0.3135	**0.4802**	0.4802	**0.4921**	0.3611	0.5516
F14	0.3333	0.5556	0.3175	0.5556	0.3294	0.5238	0.3135	0.5357
Average	0.2817	0.5173	0.2304	0.5114	0.3399	0.5241	0.2817	0.5193

Note: The out-of-sample sign accuracy lower than 50% are in bold. Classes accuracy is the ratio that eight classification forecasts are right. Sign accuracy is the ratio that the sign of eight classification forecasts is right.

The performances of the eight classification categories of MLP, CNN and LSTM are much better than the RW (12.5%). CNN performs the best, since its predictive accuracy for FTSE100 and INDU reaches 36.65% and 33.99% respectively, which is more accurate than that of Chen, Zhou & Dai (2015) (27.2% by LSTM). Additionally, the accuracy of LSTM and MLP is similar. However, the accuracy of LSTM is much lower than that of CNN. The reason for this is due to insufficient inputs lags of LSTM.

Though the results of the eight classification categories forecasts seem successful, this study further analyzes the results and finds that the conclusion is reversed from that of Olson & Mossman (2003). Table B.3 and Table B.4 also show the sign accuracy, which is obtained by converting the eight classification categories to the sign forecast (binary classification forecast). This study finds that the accuracy of converted sign forecasts is lower than that of the binary classification forecasts in Section 3.5.1 (directly obtained binary classifications). This shows that the eight classification categories do not help to improve the sign prediction accuracy. It also shows that the results of the eight classification categories are not more effective than that of the binary classification.

Accuracy of the eight classification forecasts is significantly higher than the uniform random walk because the distribution of daily returns is not uniform. For example, most daily returns cluster in -0.5% to 0% and 0% to 0.5%. NNs concentrate more on learning the distribution of in-samples, which leads to a significant increase in the accuracy of the eight classification categories while the sign accuracy is not improved, but decreased.

This study attempts to adjust the loss function calculation in NNs to solve the problem of class imbalance. In the training process of NNs, when the forecasting results are errors, the penalty values of different classes are determined according to the ratio of this class's proportion over the total sample, and the larger the proportion, the smaller the penalty value. For example, if the return is 3% on a certain day, the predicted error penalty value is larger; while the return is 0.1% on a certain day, the penalty value is smaller when prediction error exists. The NNs try to find the minimum total penalty value. After adopting this method, this study finds that NNs very easily fall into local optimum, resulting in all prediction results be in a certain class. As a result, the prediction is invalid. However, this study holds the view that this method can be further developed to improve the performance of multi-categories classification.

B.2.2 Trading performance of eight categories classification

Table B.5 and Table B.6 show trading performances of the eight classification forecasts for FTSE100 and INDU respectively. The trading performance of eight classification categories is worse than that of the binary classification. In particular, the trading performance of the LSTM is close to that of Buy-and-Hold.

Table B.5: Trading performances of NNs with eight classification categories for FTSE100 (%)

	Buy-and-Hold	MLP	MLP with AR & MA	CNN	LSTM
F1	14.98	14.51	12.20	**24.59**	6.04
F2	10.74	**13.95**	1.58	4.70	11.73
F3	0.88	**9.73**	6.63	-2.79	4.23
F4	-48.84	9.92	**30.65**	18.68	-10.73
F5	**30.09**	18.58	4.41	12.69	-3.91
F6	7.56	15.35	**19.85**	1.98	1.82
F7	-3.23	**8.79**	-7.43	3.23	8.37
F8	6.09	0.36	12.37	**14.49**	13.71
F9	**13.14**	13.03	4.63	12.45	-8.87
F10	**3.11**	1.25	1.50	-1.49	-2.65
F11	-13.37	0.67	-15.05	3.20	**16.58**
F12	14.56	-4.92	-18.46	7.36	**20.04**
F13	**7.08**	-10.43	-4.73	-4.69	-8.44
F14	3.45	-6.19	**14.53**	-3.29	2.42
Average	3.30	**6.04**	4.48	6.51	3.60

Note: The best performance in every forecast period is in bold. The units in the table are %.

Table B.6: Trading performances of NNs with eight classification categories for INDU (%)

	Buy-and-Hold	MLP	MLP with AR & MA	CNN	LSTM
F1	**1.64**	0.27	0.04	0.53	-8.57
F2	14.22	**15.89**	3.34	14.22	10.20
F3	**6.25**	-1.25	-1.10	-6.25	-12.25
F4	-56.21	21.42	-2.45	**46.60**	21.80
F5	**30.32**	-7.57	30.18	**30.32**	16.27
F6	**8.43**	**8.43**	6.96	-8.95	-9.11
F7	**4.55**	2.05	3.17	**4.55**	-11.65
F8	14.65	**15.67**	3.00	-4.50	4.62
F9	14.82	5.50	-12.07	**15.68**	2.49
F10	9.01	-5.32	-2.50	6.25	**9.66**
F11	-4.95	-6.04	2.91	4.95	**47.25**
F12	13.16	**16.23**	13.86	13.16	4.82
F13	**16.75**	15.21	-9.45	-16.75	15.21
F14	**13.21**	8.36	-17.74	**13.21**	8.83
Average	6.13	6.35	1.30	**8.07**	7.11

Note: The best performance in every forecast period is in bold. There are some cases that the return of NNs are the same with Buy-and-Hold, which is due to the failed training (the forecasting results are all in one class). The units in the table are %.

With regard to the performances of the binary classification and eight classification categories, this study draws a different conclusion from Olson & Mossman (2003). This study finds that when using eight classification categories, NNs use most of the capabilities to learn the in-sample distribution of the categories, resulting in a worse average performance than binary classification categories.

Appendix B.3 Trading performances of combination techniques

The annualized trading performances of combination techniques are presented in Table B.7 (for FTSE100) and Table B.8 (for INDU). The trading performance of the simple average is slightly worse than GRR and LASSO. The annualized returns of GRR and LASSO are almost the same for every test period because their daily statistical forecasts are almost the same.

Table B.7: Annualized returns of out-of-sample combination of FTSE100 (%)

	Simple Average	GRR	LASSO
F1	14.93	10.79	10.79
F2	11.55	23.90	22.68
F3	5.87	16.09	16.09
F4	**-1.32**	10.96	10.96
F5	20.98	13.63	13.63
F6	14.85	16.99	16.99
F7	**-1.39**	**-5.28**	**-5.28**
F8	**-5.93**	4.90	5.19
F9	5.81	6.00	6.00
F10	11.30	**-7.87**	**-7.87**
F11	30.28	17.68	17.68
F12	7.68	29.22	29.22
F13	18.61	19.74	21.22
F14	24.86	10.13	10.13
Average	11.29	11.92	11.96

Note: The out-of-sample annualized returns of the simple average, GRR and LASSO lower than zero are in bold. The units in the table are %.

Table B.8: Annualized returns of combination for out-of-sample: INDU (%)

	Simple Average	GRR	LASSO
F1	**-2.92**	10.83	10.83
F2	**-6.84**	**-16.39**	**-16.39**
F3	**-15.32**	15.14	12.86
F4	13.91	23.18	23.27
F5	21.94	21.95	21.95
F6	11.76	11.86	11.86
F7	3.62	6.19	6.19
F8	5.86	2.21	1.72
F9	9.51	28.32	28.32
F10	22.56	8.30	7.90
F11	**-8.88**	8.82	8.82
F12	13.72	19.82	19.82
F13	27.57	10.09	10.09
F14	15.78	**-10.86**	**-10.86**
Average	8.02	9.96	9.91

Note: The out-of-sample annualized returns of the simple average, GRR and LASSO lower than zero are in bold. The units in the table are %.

Appendix B.4 The Chi-square test for MLP, CNN and LSTM

Considering the FTSE100 and INDU together (in total 28 forecasts), Buy-and-Hold, MLP, CNN and LSTM win 2, 8, 6 and 9 times respectively.

Table B.9: The Chi-square test for NNs

	Positive	Negative	Sum
MLP	8	20	28
CNN	6	22	28
LSTM	9	19	28
Buy-and-Hold	2	26	28
Sum	25	87	112
Expected value	5.6	22.4	28

Note: The expected value is the expectation when all the forecasting models are random forecasts.

H0: the distribution of results of the two models is similar.

$$\chi^2 = \sum \frac{(real\ number - T)^2}{T} \tag{B.1}$$

Where:

T is the expected positive times for the benchmark model.

For LSTM, compared with Buy-and-Hold, $\chi^2 = 10.2946$

For MLP, compared with Buy-and-Hold, $\chi^2 = 9$

For CNN, compared with Buy-and-Hold, $\chi^2 = 7.7$

Table B.10: Upper percentage points of the χ^2 distribution

Degrees of freedom	Pr		
	0.05	0.01	0.001
1	3.84	6.63	10.83
2	5.99	9.21	13.81
3	7.81	11.34	16.27
4	9.49	13.28	18.47
5	11.07	15.09	20.52
6	12.59	16.81	22.46
7	14.07	18.48	24.32
8	15.51	20.09	26.12
9	16.92	21.67	27.88
10	18.31	23.21	29.59

Note: The degree of freedom is 1 for the test.

In Table B.10, the value of χ^2 can reject the H0 at a 95% confidence level. NNs is significantly better than Buy-and-Hold. For the comparison between NNs, the value of χ^2 is very small and cannot reject H0. Therefore, in this study, the difference in the performances among NNs is not large. It is necessary to increase the number of trials to

determine which NN is significantly better than others.

Appendix B.5 The Information Ratio

The Information Ratio is defined as

$$\text{Information Ratio} = \frac{annualized\ return - benchmark\ model\ annualized\ retun}{standard\ deviation} \quad (\text{B. 2})$$

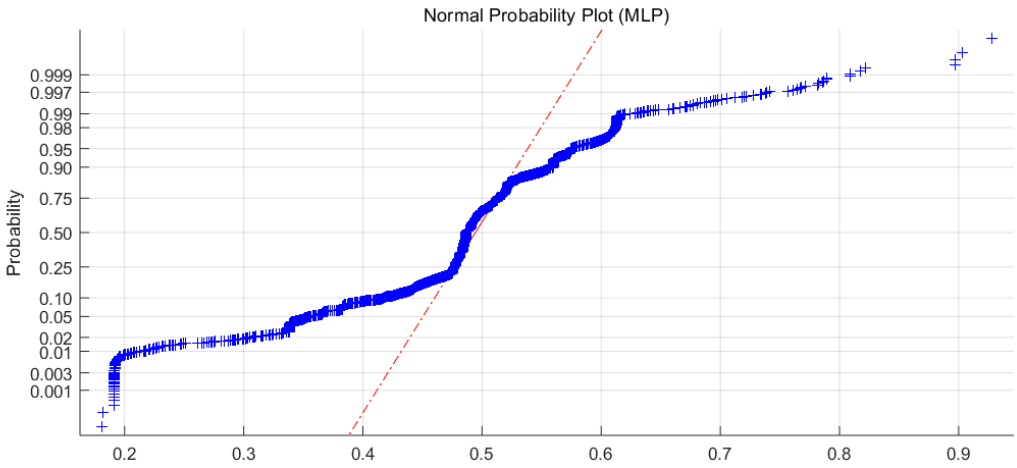
The benchmark model is Buy-and-Hold.

Appendix B.6 Distribution analysis of forecasting probability

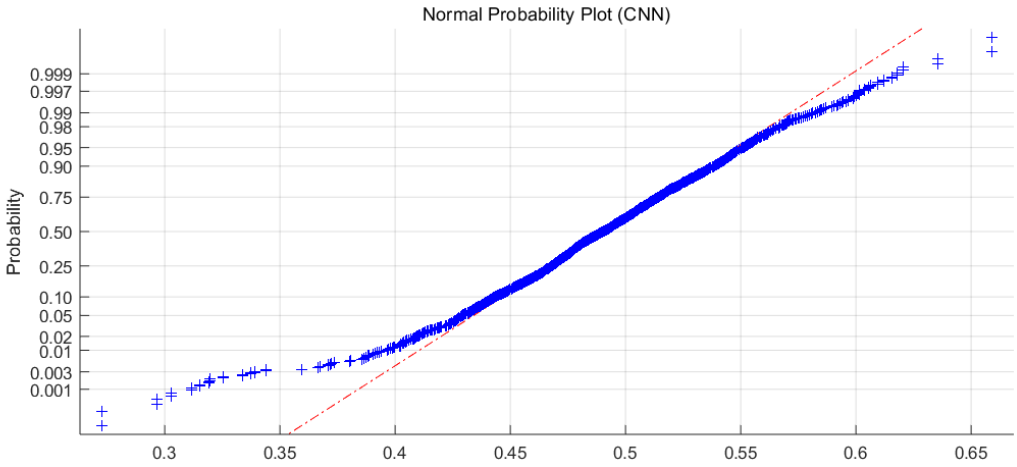
This study uses the normplot function in MATLAB to examine the distribution of the forecasting results. Figures B.1 shows the distribution of the forecasting results of MLP, CNN and LSTM. The data points appear along the red line, meaning the forecasting results are normally distributed. The distribution of LSTM's results is closest to normal distribution. The three distributions for NNs are all left-skewed, especially the forecasting results for MLP and CNN. The left-skewness of CNN is the main reason for CNN failing to pass the Kolmogorov-Smirnov Test in section 3.6.2. However, the results for CNN fit normal distribution very well in the middle part, and thus the results are still considered as a normal distribution.

Figure B.1: Fitting analysis of MLP, CNN and LSTM

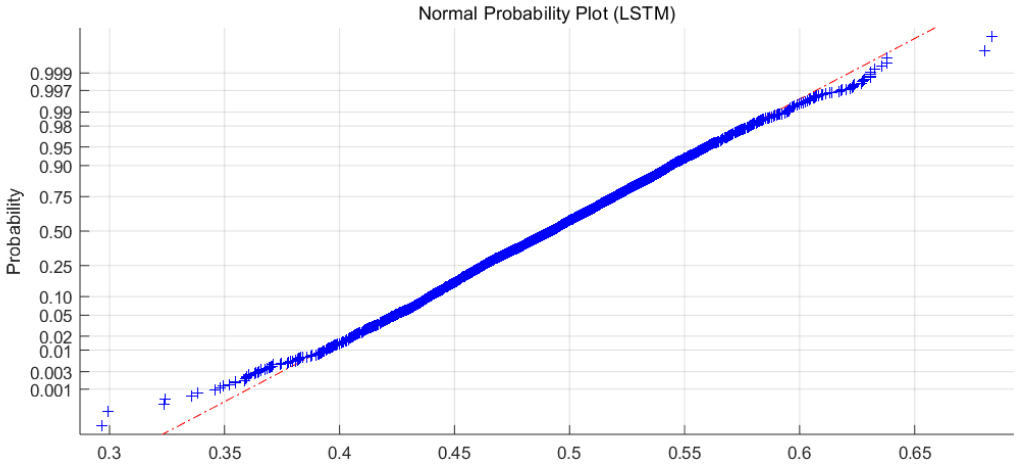
(a) Fitting analysis of MLP



(b) Fitting analysis of CNN



(c) Fitting analysis of LSTM



Note: The vertical axis is the normal distribution density. The horizontal axis is the forecast results. The red line is the regression line of the blue dots. The blue dots are close to a normal distribution if they fit the red line well.

Appendix C (Chapter 4)

Appendix C.1 Literature list

This section presents the literature introduced in Section 4.2.1. A concise overview with relevant literature, their data samples and main results are provided in Table C.1.

Table C.1: An overview of the literature

Approach	Representative studies	Sample	Main results
DIM	Gatev, Goetzmann & Rouwenhorst (1999; 2006)	US CRSP 1962-1997; US CRSP 1962-2002	11% annualized return on actual employed capital
	Perlin (2009)	Brazilian financial market	-
	Broussard & Vaihekoski (2012)	Finnish stock market 1987–2008	12.5% annualized return on actual employed capital
	Jacobs & Weber (2015)	NYSE and AMEX stocks 1960-2008	The profit is from over-reaction to news
CA	Vidyamurthy (2004)	-	-
	Lin et al. (2006)	Selected stocks from Australian stock market 01.2001- 08.2002	-
	Bogomolov (2011)	Australian stock market 1996-2010	1.05% average monthly return of CA-ST
	Caldeira & Moura (2013)	Brazilian stock market 2005-2010	16.38% annualized return
	Li (2014)	38 dual-listed companies in China A-share and Hong Kong H-share	10.8% annualized return
	Rad et al. (2016)	US equity market 1962-2014	0.85% monthly return for CA-ST before transaction costs
Machine learning	Elliott et al. (2005)	-	-
	Huck (2009)	S&P 100 1992 to 2006	13%-57% annualized return
	Huck (2010)	S&P 100 1992 to 2006	16%-38% annualized return
	Avellaneda & Lee (2010)	ETFs 1997-2007	1.1 Sharpe Ratio
	Huang et al. (2015)	Selected stocks on Taiwan stock market 2003-2012	-
Others: stochastic, copula	Do et al. (2006)	-	-
	Do & Faff (2010)	US stock market 1962-2009	Pairs trading' profitability decreases over time
	Mudchanatongsuk et al. (2008)	-	-
	Liew & Wu (2013)	Selected stocks 2009-2012	-
	Krauss & Stübinger (2017)	S&P 100 1990-2014	7.22% annualized return

Note: This table presents the literature on the methods, data and trading performance of pairs trading. The literature is mainly in the stock trading field.

Appendix C.2 Commodities list

This section presents all the commodities used in Chapter 4. The 1980-2018 data are collected from Bloomberg. Daily closing prices are used in this study.

Table C.2: Commodity list

Symbol	Name	Market
BO1	Soybean oil	CBOT
C1	Corn	CBOT
CC1	Cocoa	NYBOT
CL1	Crude oil	NYMEX
CO1	Brent oil	ICE
CT1	Cotton	NYBOT
FC1	Feeder cattle	CME
GC1	Gold	NYMEX
HG1	Copper	NYMEX
HO1	Heating oil	NYMEX
JO1	Orange juice	NYBOT
KC1	Coffee	NYBOT
KW1	Wheat	KCBT
LA1	Aluminium	LME
LB1	Lumber	CME
LC1	Live cattle	CME
LH1	Lean hogs	CME
LL1	Lead	LME
LN1	Nickel	LME
LP1	Copper	LME
LT1	Tin	LME
LX1	Zinc	LME
LY1	Aluminium alloy	LME
MW1	Wheat spring	MGEX
NG1	Natural gas	NYMEX
PA1	Palladium	NYMEX
PB1	Pork bellies	CME
PL1	Platinum	NYMEX
RR1	Rough rice	CBOT
RS1	Canola	WCE
S1	Soybean	CBOT
SB1	Sugar	NYBOT
SI1	Silver	NYMEX
SM1	Soybean meal	CBOT
W1	Wheat	CBOT

Note: Data come from: Chicago Board Of Trade (CBOT), Chicago Mercantile Exchange (CME), Intercontinental Exchange (ICE), Kansas City Board of Trade (KCBT), London Metal Exchange (LME), Minneapolis Grain Exchange (MGEX), New York Board of Trade (NYBOT), New York Mercantile Exchange (NYMEX), Winnipeg Commodity Exchange (WCE). Regarding 35 types of features, the total number of pairs is $35 * 34/2 = 595$.

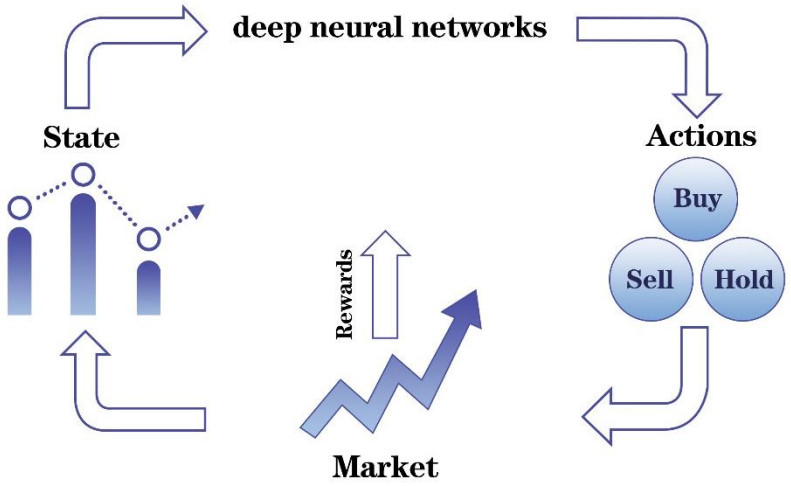
Appendix C.3 The structure of DNN in the DRL

The DNN of DRL in this study is introduced in detail in this section, which includes the training method, layers of DNN, number of neurons and the activation function. In addition, the reasons for using these structures are explained here.

C.3.1 Deep learning for RL model

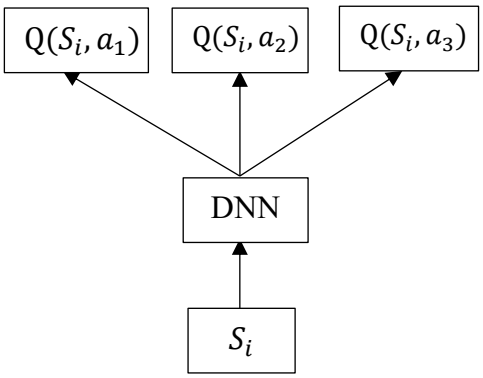
The problem with RL is mainly caused by the limitation of the Q matrix. This can be solved by using the DNN to replace the RL model's brain, that is, using DNN instead of Q matrix to help the RL model make decisions, which is shown in Figure C.1.

Figure C.1: The training process and decision-making process of the DRL model



Note: This figure shows the structure of DRL for pairs trading. The DNN is trained through iteration. DNN decides the action of the agent. Then agent trades in the market (environment), obtains the position state of itself and the reward of this action.

Figure C.2: The states calculated by DNN in the DRL model



Note: S_i is the state at time i. a_1 denotes action (1). DNN replaces the Q matrix and estimates the expected reward for the actions under the states and environment.

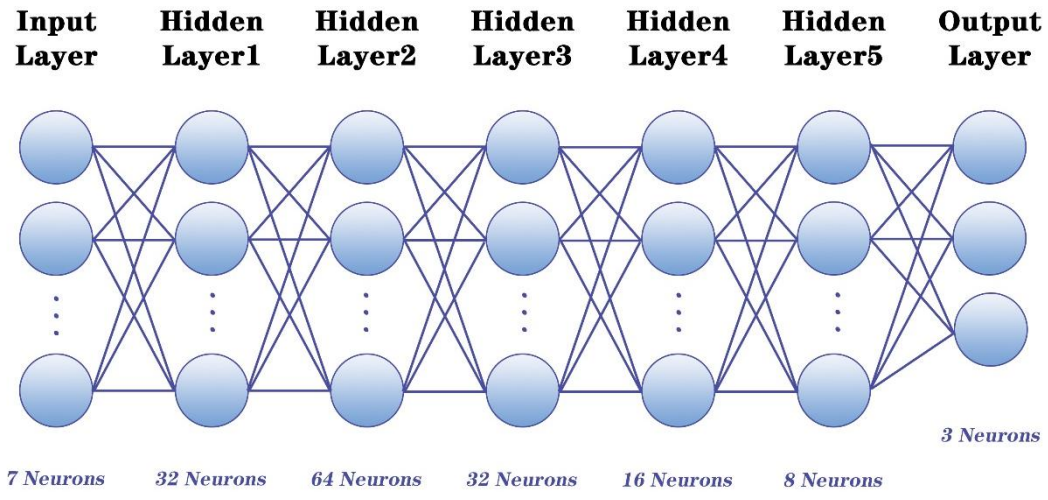
As shown in Figure C.2, in the DRL, DNN calculates the expected value of the various actions of rewards (returns). Thus, it is not necessary to seek the expectations of the corresponding states and actions in the Q matrix, as in the RL model. The DRL model only needs to input the state, including the current price and current long or short position of the asset (and 5 lags are used here).

In terms of the in-sample and out-of-sample, some fixed-length trading periods are randomly picked from a period. The game ends when it reaches the end time in each iteration. The final reward is calculated with returns, the number of transactions and daily volatility. The trained DRL model is tested in the out-of-sample.

C.3.2 Deep Neural Networks

This study uses a 7-layer MLP as the brain of the DRL. MLP is a kind of DNN architecture, which is also one of the most widely used neural network topologies. MLP has the capability to approximate arbitrary functions (Principe, Euliano & Lefebvre, 2000). This suggests that MLP has promise when it comes to problems of non-linear dynamics and function mapping. The structure of the MLP in the DRL of this study is shown in Figure C.3.

Figure C.3: MLP's structure in the DRL



Note: The input layer has 7 features which include 5 lags of price spread, 1 feature of the position state and 1 feature of the standard deviation. The output layer has 3 neurons, which represent the rewards of three trading actions. There are 5 hidden layers with 32, 64, 32, 16 and 8 neurons respectively, which provide sufficient depth for evaluating the rewards based on environment and state.

This research uses the BP algorithm for network training. Additionally, this study uses AdamOptimizer to adjust the connection weights. The activation function is a linear function. There are two reasons for using a linear function here. First, the computing result of output is the reward of each action, and the final reward calculation result can be negative or positive. Thus, an activation function that is symmetrical about the origin is required. Second, there is no limit to the size of the reward. The linear function meets these two requirements. While in the training process, the linear function may cause the connection weights of the MLP in the DRL to fail to converge, leading to the problem that the estimated reward of the MLP infinitely increases. This kind of situation rarely occurs, and this research completely avoids it through employing the pre-training technique.

Appendix C.4 Details of the co-integration test

Table C.3 shows the number of pairs with the co-integration or non-cointegration relationship in every trading period. The number of pairs with a co-integration relationship is not stable. The total number of pairs in earlier years is fewer than in recent years because some commodities do not have data in earlier years.

Table C.3: Summary of co-integration analysis

Year	Co-integration	Non-cointegration	Year	Co-integration	Non-cointegration
1980-1982	8	128	1999-2001	82	513
1981-1983	28	125	2000-2002	59	536
1982-1984	25	146	2001-2003	45	550
1983-1985	3	168	2002-2004	35	560
1984-1986	20	151	2003-2005	40	555
1985-1987	9	267	2004-2006	68	527
1986-1988	43	233	2005-2007	24	571
1987-1989	28	323	2006-2008	45	550
1988-1990	18	333	2007-2009	25	570
1989-1991	15	363	2008-2010	35	560
1990-1992	26	352	2009-2011	23	572
1991-1993	42	336	2010-2012	40	555
1992-1994	13	365	2011-2013	77	506
1993-1995	13	365	2012-2014	38	557
1994-1996	53	325	2013-2015	19	576
1995-1997	45	333	2014-2016	41	554
1996-1998	13	582	2015-2017	65	529
1997-1999	83	512	2016-2018	37	556
1998-2000	46	549			

Note: The co-integrated pairs are selected in the portfolio. The return of the portfolio with more pairs will be more stable.

This study chooses a 95% confidence level to judge whether the pairs are co-integrated. If the p-value is lower than 5% in the Engle-Granger test, this study considers that the pair is co-integrated. It is not wise to use a more restricted confidence level to select pairs, as that would cause a lower expected level of return and higher risks in the out-of-sample.

The pairs with the co-integration relationship in a period are formed as a portfolio with equal weights. For example, in 1980-1982, 8 pairs are selected by CA. For each pair, this study calculates the return and recorded trading actions in 1980 and 1981 (for the in-sample) and 1983 (for the out-of-sample) by ST and DRL. Then, the daily return and annualized return of the portfolio can be calculated.

Appendix C.5 Return of CA-DRL without considering the in-sample performance

Table C.4 presents the results of the CA-DRL trading model without any selection in the in-sample. The average annualized return of CA-DRL that does not consider the in-sample performance is a bit lower, at 11.64% (the return of CA-DRL considers the in-sample performance to be 12.56%). However, only one negative result occurs in 37 periods.

Table C.4: Annualized returns of CA-DRL without making any selection in in-sample.

	CA-DRL			CA-DRL	
Year	In-sample	Out-of-sample	Year	In-sample	Out-of-sample
1980-1982	0.2435	0.1776	1998-2000	0.3807	0.0907
1981-1983	0.2689	0.0560	1999-2001	0.1233	0.0679
1982-1984	0.1368	0.0836	2000-2002	0.2930	0.0930
1983-1985	0.2172	0.1063	2001-2003	0.3351	0.1385
1984-1986	0.3996	0.2193	2002-2004	0.2870	0.1559
1985-1987	0.4343	0.2270	2003-2005	0.4256	0.1772
1986-1988	0.2422	0.1258	2004-2006	0.3705	0.1322
1987-1989	0.2635	0.0281	2005-2007	0.4275	0.0966
1988-1990	0.3137	0.1044	2006-2008	0.2993	0.0799
1989-1991	0.2329	0.0735	2007-2009	0.3221	0.1382
1990-1992	0.3338	0.1243	2008-2010	0.5425	0.1656
1991-1993	0.3240	0.0569	2009-2011	0.4006	0.2057
1992-1994	0.1804	0.0873	2010-2012	0.4745	0.1811
1993-1995	0.3181	**-0.1052**	2011-2013	0.2884	0.0935
1994-1996	0.1635	0.1171	2012-2014	0.1575	0.1548
1995-1997	0.1152	0.0560	2013-2015	0.3301	0.1311
1996-1998	0.2196	0.0230	2014-2016	0.3728	0.1338
1997-1999	0.3478	0.1862	2015-2017	0.2518	0.1382
1998-2000	0.3807	0.0907	2016-2018	0.2104	0.1868
1999-2001	0.1233	0.0679	Average	0.2986	0.1164

Note: The negative returns per test are marked in bold. There is only one negative return in the 1993-1995 out-of-sample test.

Bibliography

Abraham, A., Nath, B., & Mahanti, P. K. (2001, May). Hybrid intelligent systems for stock market analysis. In *International Conference on Computational Science* (pp. 337-345). Springer, Berlin, Heidelberg.

Adebiyi, A. A., Adewumi, A. O., & Ayo, C. K. (2014). Comparison of ARIMA and artificial neural networks models for stock price prediction. *Journal of Applied Mathematics*.

Alexander, C. (2001). *Market models: A guide to financial data analysis*. John Wiley & Sons.

Armano, G., Marchesi, M., & Murru, A. (2005). A hybrid genetic-neural architecture for stock indices forecasting. *Information Sciences*, 170(1), 3-33.

Asadi, S., Hadavandi, E., Mehmanpazir, F., & Nakhostin, M. M. (2012). Hybridization of evolutionary Levenberg–Marquardt neural networks and data pre-processing for stock market prediction. *Knowledge-Based Systems*, 35, 245-258.

Avellaneda, M., & Lee, J. H. (2010). Statistical arbitrage in the US equities market. *Quantitative Finance,* 10(7), 761-782.

Baba, N., & Kozaki, M. (1992, June). An intelligent forecasting system of stock price using neural networks. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks* (Vol. 1, pp. 371-377). IEEE.

Bahrololoum, A., Nezamabadi-Pour, H., Bahrololoum, H., & Saeed, M. (2012). A prototype classifier based on gravitational search algorithm. *Applied Soft Computing*, 12(2), 819-825.

Baillie, R. T. (1989). Econometric tests of rationality and market efficiency. *Econometric Reviews*, 8(2), 151-186.

Bates, J. M., & Granger, C. W. (1969). The combination of forecasts. *Journal of the Operational Research Society*, 20(4), 451-468.

Bergh, F. V. D., & Engelbrecht, A. P. (2006). A study of particle swarm optimization particle trajectories. *Information sciences*, 176(8), 937-971.

Bertsekas, D. P., & Tsitsiklis, J. N. (1995, December). Neuro-dynamic programming: an overview. In *Proceedings of 1995 34th IEEE Conference on Decision and Control* (Vol. 1, pp. 560-564). IEEE.

Bogomolov, T. (2011, November). Pairs trading in the land down under. In *Finance and Corporate Governance Conference*.

Bogomolov, T. (2013). Pairs trading based on statistical variability of the spread process. *Quantitative Finance*, 13(9), 1411-1430.

Bookstaber, R. (2007). *A demon of our own design: Markets, hedge funds, and the perils of financial innovation.* John Wiley & Sons.

Brockett, P. L., Cooper, W. W., Golden, L. L., & Pitaktong, U. (1994). A neural network method for obtaining an early warning of insurer insolvency. *Journal of Risk and Insurance*, 402-424.

Broussard, J. P., & Vaihekoski, M. (2012). Profitability of pairs trading strategy in an illiquid market with multiple share classes. *Journal of International Financial Markets, Institutions and Money*, 22(5), 1188-1201.

Buehler, H., Gonon, L., Teichmann, J., & Wood, B. (2019). Deep hedging. *Quantitative Finance*, 1-21.

Caldeira, J., & Moura, G. V. (2013). Selection of a portfolio of pairs based on cointegration: A statistical arbitrage strategy. *Available at SSRN 2196391*.

Cao, L. J., Chua, K. S., Chong, W. K., Lee, H. P., & Gu, Q. M. (2003). A comparison of PCA, KPCA and ICA for dimensionality reduction in support vector machine. *Neurocomputing*, 55(1-2), 321-336.

Cao, L., & Tay, F. E. (2001). Financial forecasting using support vector machines. *Neural Computing & Applications*, 10(2), 184-192.

Cao, L. J., & Tay, F. E. (2003). Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on neural networks,* 14(6), 1506-1518.

Cao, Q., Leggio, K. B., & Schniederjans, M. J. (2005). A comparison between Fama and French's model and artificial neural networks in predicting the Chinese stock market. *Computers & Operations Research*, 32(10), 2499-2512.

Chen, A. S., Leung, M. T., & Daouk, H. (2003). Application of neural networks to an emerging financial market: forecasting and trading the Taiwan Stock Index. *Computers & Operations Research*, 30(6), 901-923.

Chen, K., Zhou, Y., & Dai, F. (2015, October). A LSTM-based method for stock returns prediction: A case study of China stock market. *In 2015 IEEE International Conference on Big Data (Big Data)* (pp. 2823-2824). IEEE.

Chen, M. Y., Chen, C. C., & Liu, J. Y. (2013, June). Credit rating analysis with support vector machines and artificial bee colony algorithm. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems* (pp. 528-534). Springer, Berlin, Heidelberg.

Choi, D. J., & Park, H. (2001). A hybrid artificial neural network as a software sensor for optimal control of a wastewater treatment process. *Water research*, 35(16), 3959-3967.

Ciner, C., Gurdgiev, C., & Lucey, B. M. (2013). Hedges and safe havens: An examination of stocks, bonds, gold, oil and exchange rates. *International Review of Financial Analysis*, *29*, 202-211.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3), 273-297.

Damghani, B. M. (2013). The Non-Misleading Value of Inferred Correlation: An

Introduction to the Cointelation Model. *Wilmott*, 2013(67), 50-61.

Deng, Y., Bao, F., Kong, Y., Ren, Z., & Dai, Q. (2016). Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems*, 28(3), 653-664.

Diebold, F. X., & Mariano, R. S. (2002). Comparing predictive accuracy. *Journal of Business & economic statistics*, *20*(1), 134-144.

Ding, X., Zhang, Y., Liu, T., & Duan, J. (2015, June). Deep learning for event-driven stock prediction. *In Twenty-fourth international joint conference on artificial intelligence*.

Do, B., Faff, R., & Hamza, K. (2006, May). A new approach to modeling and estimation for pairs trading. *In Proceedings of 2006 Financial Management Association European Conference* (pp. 87-99).

Do, B., & Faff, R. (2010). Does simple pairs trading still work?. *Financial Analysts Journal*, 66(4), 83-95.

Do, B., & Faff, R. (2012). Are pairs trading profits robust to trading costs?. *Journal of Financial Research*, 35(2), 261-287.

Dorigo, M., & Di Caro, G. (1999). Ant colony optimization: a new meta-heuristic. *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on* (Vol. 2, pp. 1470-1477). IEEE.

Duman, S., Güvenç, U., Sönmez, Y., & Yörükeren, N. (2012). Optimal power flow using gravitational search algorithm. *Energy Conversion and Management*, 59, 86-95.

Dunis, C.L., Likothanassis, S.D., Karathanasopoulos, A.S., Sermpinis, G.S. & Theofilatos, K.A. (2013) A hybrid genetic algorithm–support vector machine approach in the task of forecasting and trading. *Journal of Asset Management*, 14(1), pp. 52-71.

Ekström, E., Lindberg, C., & Tysk, J. (2011). Optimal liquidation of a pairs trade. In

Advanced mathematical methods for finance (pp. 247-255). Springer, Berlin, Heidelberg.

Elliott, R. J., Van Der Hoek, J., & Malcolm, W. P. (2005). Pairs trading. *Quantitative Finance*, *5*(3), 271-276.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, *14*(2), 179-211.

Enke, D., & Thawornwong, S. (2005). The use of data mining and neural networks for forecasting stock market returns. *Expert Systems with applications*, 29(4), 927-940.

Fama, E.F. (1970). Efficient capital markets: A review of theory and empirical work, *The Journal of Finance,* 25 (2), pp. 383–417.

Farmer, J. D., Packard, N. H., & Perelson, A. S. (1986). The immune system, adaptation, and machine learning. *Physica D: Nonlinear Phenomena*, 22(1-3), 187-204.

Fernandez-Rodrıguez, F., Gonzalez-Martel, C., & Sosvilla-Rivero, S. (2000). On the profitability of technical trading rules based on artificial neural networks: Evidence from the Madrid stock market. *Economics letters*, 69(1), 89-94.

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2), 654-669.

Fletcher, D., & Goss, E. (1993). Forecasting with neural networks: an application using bankruptcy data. *Information & Management*, 24(3), 159-167.

Formato, R. A. (2007). Central Force Optimization. *progress in Electromagnetic Research*, 77, 425-491.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4), 193-202.

Funahashi, K. I. (1989). On the approximate realization of continuous mappings by neural

networks. *Neural networks*, *2*(3), 183-192.

Gatev, E., Goetzmann, W. N., & Rouwenhorst, K. G. (1999). Pairs trading: Performance of a relative value arbitrage rule. *Working paper, Yale School of Management's International Center for Finance.*

Gatev, E., Goetzmann, W. N., & Rouwenhorst, K. G. (2006). Pairs trading: Performance of a relative-value arbitrage rule. *The Review of Financial Studies*, 19(3), 797-827.

Gazi, V., & Passino, K. M. (2004). Stability analysis of social foraging swarms. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(1), 539-557.

Giacomini, R., & White, H. (2006). Tests of conditional predictive ability. *Econometrica*, 74(6), 1545-1578.

Granger, C. W., & Ramanathan, R. (1984). Improved methods of combining forecasts. *Journal of forecasting*, 3(2), 197-204.

Guo, X., Singh, S., Lee, H., Lewis, R. L., & Wang, X. (2014). Deep learning for real-time Atari game play using offline Monte-Carlo tree search planning. *In Advances in neural information processing systems,* (pp. 3338-3346).

Guresen, E., Kayakutlu, G., & Daim, T. U. (2011). Using artificial neural network models in stock market index prediction. *Expert Systems with Applications*, 38(8), 10389-10397.

Hassan, M. R., Nath, B., & Kirley, M. (2007). A fusion model of HMM, ANN and GA for stock market forecasting. *Expert systems with Applications*, 33(1), 171-180.

Herlemont, D. (2003). Pairs trading, convergence trading, cointegration. *YATS Finances and Technology*, 33, 1-31.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A. R., Jaitly, N., ... & Sainath, T. (2012). Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29.

Hochreiter, S. (1991). Untersuchungen zu dynamischen neuronalen Netzen. *Diploma, Technische Universität München*, 91(1).

Hong, W.C., (2006). A hybrid support vector machine regression for exchange rate prediction. *International Journal of Information and Management Sciences,* 17(2), pp. 19-32.

Hsieh, T. J., Hsiao, H. F., & Yeh, W. C. (2011). Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm. *Applied soft computing*, 11(2), 2510-2525.

Huang, C. F., Hsu, C. J., Chen, C. C., Chang, B. R., & Li, C. A. (2015). An intelligent model for pairs trading using genetic algorithms. *Computational Intelligence and Neuroscience*.

Huang, W., Nakamori, Y., & Wang, S. Y. (2005). Forecasting stock market movement direction with support vector machine. *Computers & Operations Research*, 32(10), 2513-2522.

Huck, N. (2009). Pairs selection and outranking: An application to the S&P 100 index. *European Journal of Operational Research*, 196(2), 819-825.

Huck, N. (2010). Pairs trading and outranking: The multi-step-ahead forecasting case. *European Journal of Operational Research*, 207(3), 1702-1716.

Huck, N., & Afawubo, K. (2015). Pairs trading and selection methods: is cointegration superior? *Applied Economics*, 47(6), 599-613.

Jackson, J. E. (2005). *A user's guide to principal components* (Vol. 587). John Wiley & Sons.

Jacobs, H., & Weber, M. (2015). On the determinants of pairs trading profitability. *Journal of Financial Markets*, 23, 75-97.

Jasic, T., & Wood, D. (2004). The profitability of daily stock market indices trades based on neural network predictions: Case study for the S&P 500, the DAX, the TOPIX and the FTSE

in the period 1965–1999. *Applied Financial Economics*, 14(4), 285-297.

Jurek, J. W., & Yang, H. (2007, April). Dynamic portfolio selection in arbitrage. *In EFA 2006 Meetings Paper*.

Kai, F., & Wenhua, X. (1997, October). Training neural network with genetic algorithms for forecasting the stock price index. *In Intelligent Processing Systems, 1997. ICIPS'97. 1997 IEEE International Conference on* (Vol. 1, pp. 401-403). IEEE.

Kanamura, T., Rachev, S. T., & Fabozzi, F. J. (2009). A profit model for spread trading with an application to energy futures. *The Journal of Trading*, *5*(1), 48-62.

Kao, L. J., Chiu, C. C., Lu, C. J., & Yang, J. L. (2013). Integration of nonlinear independent component analysis and support vector regression for stock price forecasting. *Neurocomputing*, 99, 534-542.

Karathanasopoulos, A., Theofilatos, K. A., Sermpinis, G., Dunis, C., Mitra, S. & Stasinakis, C. (2016) Stock market prediction using evolutionary support vector machines: an application to the ASE20 index. *European Journal of Finance*, 22(12), pp. 1145-1163.

Keating, C., & Shadwick, W. F. (2002). A universal performance measure. *Journal of performance measurement*, 6(3), 59-84.

Kennedy, J. (2011). Particle swarm optimization. *In Encyclopedia of machine learning* (pp. 760-766). Springer US.

Kim K. (2003) Financial time series forecasting using support vector machines. *Neurocomputing*, Volume 55, Issues 1–2, pp. 307–319.

Kim, K. J., & Han, I. (2000). Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. *Expert systems with Applications*, 19(2), 125-132.

Kim, K. J., & Lee, W. B. (2004). Stock market prediction using artificial neural networks

with optimal feature transformation. *Neural computing & applications*, 13(3), 255-260.

Kimoto, T., Asakawa, K., Yoda, M., & Takeoka, M. (1990, June). Stock market prediction system with modular neural networks. *In Neural Networks, 1990., 1990 IJCNN International Joint Conference on* (pp. 1-6). IEEE.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. science, 220(4598), 671-680.

Krauss, C. (2017). Statistical arbitrage pairs trading strategies: Review and outlook. *Journal of Economic Surveys*, 31(2), 513-545.

Krauss, C., & Stübinger, J. (2017). Non-linear dependence modelling with bivariate copulas: Statistical arbitrage pairs trading on the S&P 100. *Applied Economics,* 49(52), 5352-5369.

Kräussl, R., & Sandelowsky, R. M. (2007, January). *The predictive performance of Morningstar's mutual fund ratings*. (Working papers series; No. 963589). SSRN.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *In Advances in neural information processing systems* (pp. 1097-1105).

Kryzanowski, L., Galler, M., & Wright, D. W. (1993). Using artificial neural networks to pick stocks. *Financial Analysts Journal*, 49(4), 21-27.

Kuo, R. J., Chen, C. H., & Hwang, Y. C. (2001). An intelligent stock trading decision support system through integration of genetic algorithm based fuzzy neural network and artificial neural network. *Fuzzy sets and systems*, 118(1), 21-45.

Lai, R. K., Fan, C. Y., Huang, W. H., & Chang, P. C. (2009). Evolving and clustering fuzzy decision tree for financial time series data forecasting. *Expert Systems with Applications*, 36(2), 3761-3773.

Lam, M. (2004). Neural network techniques for financial performance prediction:

integrating fundamental and technical analysis. *Decision support systems*, 37(4), 567-581.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4), 541-551.

Lee, K. C., Han, I., & Kwon, Y. (1996). Hybrid neural network models for bankruptcy predictions. *Decision Support Systems*, 18(1), 63-72.

Leigh, W., Purvis, R., & Ragusa, J. M. (2002). Forecasting the NYSE composite index with technical analysis, pattern recognizer, neural network, and genetic algorithm: a case study in romantic decision support. *Decision support systems*, 32(4), 361-377.

Levine, S., Finn, C., Darrell, T., & Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1), 1334-1373.

Li, C., An, X., & Li, R., 2015. A chaos embedded GSA-SVM hybrid system for classification. *Neural Computing and Applications*, 26(3), pp.713-721.

Li, C., & Zhou, J. (2011). Parameters identification of hydraulic turbine governing system using improved gravitational search algorithm. *Energy Conversion and Management*, 52(1), 374-381.

Li, M. L., Chui, C. M., & Li, C. Q. (2014). Is pairs trading profitable on China AH-share markets? *Applied Economics Letters*, 21(16), 1116-1121.

Li, Y. (2017). Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*.

Liang, Z., Chen, H., Zhu, J., Jiang, K., & Li, Y. (2018). Adversarial deep reinforcement learning in portfolio management. *arXiv preprint arXiv:1808.09940*.

Liew, R. Q., & Wu, Y. (2013). Pairs trading: A copula approach. *Journal of Derivatives & Hedge Funds*, 19(1), 12-30.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., ... & Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lin, Y. X., McCRAE, M. I. C. H. A. E. L., & Gulati, C. (2006). Loss protection in pairs trading through minimum profit bounds: A cointegration approach. *Advances in Decision Sciences*, *2006*.

Liu, J. & Timmermann, A. (2013). Optimal convergence trade strategies. *Review of Financial Studies*, 26(4):1048–1086.

Lo, A. W., & MacKinlay, A. C. (1988). Stock market prices do not follow random walks: Evidence from a simple specification test. *The review of financial studies*, 1(1), 41-66.

Lu, C.J., Lee, T.S., & Chiu, C.C., (2009). Financial time series forecasting using independent component analysis and support vector regression. *Decision Support Systems*, *47*(2), pp.115-125.

Malkiel, B. G., & McCue, K. (1985). *A random walk down Wall Street*. New York: Norton.

Mantero, P., Moser, G., & Serpico, S. B. (2005). Partially supervised classification of remote sensing images through SVM-based probability density estimation. *IEEE Transactions on Geoscience and Remote Sensing*, 43(3), 559-570.

Marcek, D. (2004, July). Stock price forecasting: Statistical, classical and fuzzy neural network approach. *In MDAI* (Vol. 3131, pp. 41-48).

McKinney, W. (2010, June). Data structures for statistical computing in python. *In Proceedings of the 9th Python in Science Conference* (Vol. 445, pp. 51-56).

Min, S. H., Lee, J., & Han, I. (2006). Hybrid genetic algorithms and support vector machines for bankruptcy prediction. *Expert systems with applications*, 31(3), 652-660.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., & Petersen, S. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529.

Minsky, M., & Papert, S. A. (2017). *Perceptrons: An introduction to computational geometry*. MIT press.

Mirjalili, S., Hashim, S. Z. M., & Sardroudi, H. M. (2012). Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. *Applied Mathematics and Computation*, 218(22), 11125-11137.

Mizuno, H., Kosaka, M., Yajima, H., & Komoda, N. (1998). Application of neural network to technical analysis of stock market prediction. *Studies in Informatic and control*, 7(3), 111-120.

Moghaddam, A. H., Moghaddam, M. H., & Esfandyari, M. (2016). Stock market index prediction using artificial neural network. *Journal of Economics, Finance and Administrative Science*, *21(*41), 89-93.

Mostafa, M. M. (2010). Forecasting stock exchange movements using neural networks: Empirical evidence from Kuwait. *Expert Systems with Applications*, 37(9), 6302-6309.

Mountrakis, G., Im, J., & Ogole, C. (2011). Support vector machines in remote sensing: A review. ISPRS *Journal of Photogrammetry and Remote Sensing*, 66(3), 247-259.

Mudchanatongsuk, S., & Primbs, J. A., & Wong, W. (2008, June). Optimal pairs trading: A stochastic control approach. *In 2008 American Control Conference* (pp. 1035-1039). IEEE.

Mukherjee, S., Osuna, E., & Girosi, F., (1997). Nonlinear prediction of chaotic time series using support vector machines, *In Proceedings of the IEEE Workshop on Neural Networks for Signal Processing,* Amelia Island, FL, 511–520.

Nelson, D. M., Pereira, A. C., & de Oliveira, R. A. (2017, May). Stock market's price

movement prediction with LSTM neural networks. *In 2017 International Joint Conference on Neural Networks (IJCNN)* (pp. 1419-1426). IEEE.

Neumann, D. E. (2002). An enhanced neural network technique for software risk analysis. *IEEE Transactions on Software Engineering*, (9), 904-912.

Nevmyvaka, Y., Feng, Y., & Kearns, M. (2006, June). Reinforcement learning for optimized trade execution. *In Proceedings of the 23rd international conference on Machine learning* (pp. 673-680). ACM.

O'Connor, N., & Madden, M. G. (2006). A neural network approach to predicting stock exchange movements using external factors. *In Applications and Innovations in Intelligent Systems XIII* (pp. 64-77). Springer, London.

Olson, D., & Mossman, C. (2003). Neural network forecasts of Canadian stock returns using accounting ratios. *International Journal of Forecasting*, *19*(3), 453-465.

Pang, X., Zhou, Y., Wang, P., Lin, W., & Chang, V. (2018). An innovative neural network approach for stock market prediction. *The Journal of Supercomputing*, 1-21.

Partalidou, X., Kiohos, A., Giannarakis, G., & Sariannidis, N. (2016). The impact of Gold, Bond, Currency, Metals and Oil markets on the USA stock market. *International Journal of Energy Economics and Policy*, 6(1), 76-81.

Perlin, M. S. (2009). Evaluation of pairs-trading strategy at the Brazilian financial market. *Journal of Derivatives & Hedge Funds*, 15(2), 122-136.

Principe, J. C., Euliano, N. R., & Lefebvre, W. C. (2000). *Neural and adaptive systems: fundamentals through simulations* (Vol. 672). New York: Wiley.

Qian, B., & Rasheed, K. (2007). Stock market prediction with multiple classifiers. *Applied Intelligence*, 26(1), 25-33.

Quah, T. S., & Srinivasan, B. (1999). Improving returns on stock investment through neural network selection. *Expert Systems with Applications*, 17(4), 295-301.

Rad, H., Low, R. K. Y., & Faff, R. (2016). The profitability of pairs trading strategies: distance, cointegration and copula methods. *Quantitative Finance*, 16(10), 1541-1558.

Rashedi, E., & Nezamabadi-pour, H. (2012). Improving the precision of CBIR systems by feature selection using binary gravitational search algorithm. *In Artificial Intelligence and Signal Processing (AISP), 2012 16th CSI International Symposium on* (pp. 039-042). IEEE.

Rashedi, E., Nezamabadi-Pour, H. & Saryazdi, S., (2009). GSA: a gravitational search algorithm. *Information sciences,* 179(13), pp. 2232-2248.

Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2010). BGSA: binary gravitational search algorithm. *Natural Computing*, 9(3), 727-745.

Rather, A. M., Agarwal, A., & Sastry, V. N. (2015). Recurrent neural network and a hybrid model for prediction of stock returns. *Expert Systems with Applications*, 42(6), 3234-3241.

Roh, T. H. (2007). Forecasting the volatility of stock price index. *Expert Systems with Applications*, 33(4), 916-922.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088), 533.

Russell, S. J., Norvig, P., Canny, J. F., Malik, J. M., & Edwards, D. D. (2003). *Artificial intelligence: a modern approach* (Vol. 2, No. 9). Upper Saddle River: Prentice hall.

Saad, E. W., Prokhorov, D. V., & Wunsch, D. C. (1998). Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks. *IEEE Transactions on neural networks*, 9(6), 1456-1470.

Salchenberger, L. M., Cinar, E. M., & Lash, N. A. (1992). Neural networks: A new tool for predicting thrift failures. *Decision Sciences*, 23(4), 899-916.

Sarafrazi, S. & Nezamabadi-pour, H., (2013). Facing the classification of binary problems with a GSA-SVM hybrid system. *Mathematical and Computer Modelling*, *57*(1), pp. 270-

278.

Schuldt, C., Laptev, I., & Caputo, B. (2004). Recognizing human actions: a local SVM approach. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on* (Vol. 3, pp. 32-36). IEEE.

Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2015). High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

Sermpinis, G., Dunis, C., Laws, J., & Stasinakis, C. (2012). Forecasting and trading the EUR/USD exchange rate with stochastic Neural Network combination and time-varying leverage. *Decision Support Systems*, 54(1), 316-329.

Sermpinis, G., Stasinakis, C., & Hassanniakalager, A. (2017). Reverse adaptive krill herd locally weighted support vector regression for forecasting and trading exchange traded funds. *European Journal of Operational Research*, 263(2), 540-558.

Sermpinis, G., Stasinakis, C., & Karathanasopoulos, A. (2013), September. Kalman filter and SVR combinations in forecasting US unemployment. *In IFIP International Conference on Artificial Intelligence Applications and Innovations* (pp. 506-515). Springer, Berlin, Heidelberg.

Sermpinis, G., Stasinakis, C., Rosillo, R., & de la Fuente, D. (2017). European exchange trading funds trading with locally weighted support vector regression. *European Journal of Operational Research*, 258(1), pp. 372-384.

Sermpinis, G., Stasinakis, C., Theofilatos, K., & Karathanasopoulos, A. (2015). Modeling, forecasting and trading the EUR exchange rates with hybrid rolling genetic algorithms: support vector regression forecast combinations. *European Journal of Operational Research*, 247(3), pp. 831-846.

Sermpinis, G., Stasinakis, C., Theofilatos, K., & Karathanasopoulos, A. (2014) Inflation and unemployment forecasting with genetic support vector regression. *Journal of Forecasting*,

33(6), pp. 471-487.

Sermpinis, G., Verousis, T., & Theofilatos, K. (2016). Adaptive Evolutionary Neural Networks for Forecasting and Trading without a Data-Snooping Bias. *Journal of Forecasting*, 35(1), 1-12.

Shaw, B., Mukherjee, V., & Ghoshal, S. P. (2012). A novel opposition-based gravitational search algorithm for combined economic and emission dispatch problems of power systems. *International Journal of Electrical Power & Energy Systems*, 35(1), 21-33.

Sharpe, W. F (1970). *Portfolio theory and capital markets*. New York: McGraw-Hill.

Shen, W., Guo, X., Wu, C., & Wu, D. (2011). Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm. *Knowledge-Based Systems*, 24(3), 378-385.

Sheta, A. F., Ahmed, S. E. M., & Faris, H. (2015). A comparison between regression, artificial neural networks and support vector machines for predicting stock market index. *Soft Computing*, 7(8).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1), 1929-1958.

Stasinakis, C., Sermpinis, G., Theofilatos, K., & Karathanasopoulos, A. (2016). Forecasting US unemployment with radial basis neural networks, kalman filters and support vector regressions. *Computational Economics,* 47(4), pp. 569-587.

Subasi, A., & Gursoy, M. I. (2010). EEG signal classification using PCA, ICA, LDA and support vector machines. *Expert systems with applications*, 37(12), 8659-8666.

Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. *In Advances in neural information processing systems* (pp. 1057-1063).

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Swales Jr, G. S., & Yoon, Y. (1992). Applying artificial neural networks to investment analysis. *Financial Analysts Journal*, 48(5), 78-80.

Tam, K. Y., & Kiang, M. (1990). Predicting bank failures: A neural network approach. *Applied Artificial Intelligence an International Journal*, 4(4), 265-282.

Tam, K. Y. (1991). Neural network models and the prediction of bank bankruptcy. *Omega*, 19(5), 429-445.

Tam, K. Y., & Kiang, M. Y. (1992). Managerial applications of neural networks: the case of bank failure predictions. *Management science*, 38(7), 926-947.

Tang, K. S., Man, K. F., Kwong, S., & He, Q. (1996). Genetic algorithms and their applications. *IEEE signal processing magazine*, 13(6), 22-37.

Tarasewich, P., & McMullen, P. R. (2002). Swarm intelligence: power in numbers. *Communications of the ACM*, 45(8), 62-67.

Tay, F. E., & Cao, L. (2001). Application of support vector machines in financial time series forecasting. *omega*, 29(4), 309-317.

Tay, F.E., & Cao, L.J., (2002). Modified support vector machines in financial time series forecasting. *Neurocomputing*, 48(1), pp.847-861.

Tesauro, G. (1995). Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3), 58-68.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1), 267-288.

Ticknor, J. L. (2013). A Bayesian regularized artificial neural network for stock market

forecasting. *Expert Systems with Applications*, 40(14), 5501-5506.

Trafalis, T. B., & Ince, H. (2000). Support vector machine for regression and applications to financial forecasting. *In Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on* (Vol. 6, pp. 348-353). IEEE.

Trippi, R. R., & Turban, E. (1992). *Neural networks in finance and investing: Using artificial intelligence to improve real world performance*. McGraw-Hill, Inc.

Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2017, July). Forecasting stock prices from the limit order book using convolutional neural networks. *In 2017 IEEE 19th Conference on Business Informatics (CBI)* (Vol. 1, pp. 7-12). IEEE.

Van Der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2), 22.

Van Gestel, T., Suykens, J. A., Baestaens, D. E., Lambrechts, A., Lanckriet, G., Vandaele, B., & Vandewalle, J. (2001). Financial time series prediction using least squares support vector machines within the evidence framework. *IEEE Transactions on neural networks*, 12(4), 809-821.

Vapnik V. (1995). The Nature of Statistical Learning Theory. *Springer*, New York.

Vidyamurthy, G. (2004). *Pairs Trading: quantitative methods and analysis* . John Wiley & Sons.

Wang, J. H., & Leu, J. Y. (1996, June). Stock market trend prediction using ARIMA-based neural networks. *In IEEE Int. Conf. neural networks* (Vol. 4, No. 6, pp. 2160-2165).

Wang, J. Z., Wang, J. J., Zhang, Z. G., & Guo, S. P. (2011). Forecasting stock indices with back propagation neural network. *Expert Systems with Applications*, *38*(11), 14346-14355.

Wang, J. J., Wang, J. Z., Zhang, Z. G., & Guo, S. P. (2012). Stock index forecasting based

on a hybrid model. *Omega*, 40(6), 758-766.

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4), 279-292.

Watter, M., Springenberg, J., Boedecker, J., & Riedmiller, M. (2015). Embed to control: A locally linear latent dynamics model for control from raw images. *In Advances in neural information processing systems* (pp. 2746-2754).

White, H. (1988, July). Economic prediction using neural networks: The case of IBM daily stock returns. *In ICNN* (Vol. 2, pp. 451-458).

Wong, F. S., Wang, P. Z., Goh, T. H., & Quek, B. K. (1992). Fuzzy neural systems for stock selection. *Financial Analysts Journal*, 48(1), 47-52.

Wu, C. H., Tzeng, G. H., Goo, Y. J., & Fang, W. C. (2007). A real-valued genetic algorithm to optimize the parameters of support vector machine for predicting bankruptcy. *Expert systems with applications*, 32(2), 397-408.

Xiong, Z., Liu, X. Y., Zhong, S., Yang, H., & Walid, A. (2018). Practical deep reinforcement learning approach for stock trading. *arXiv preprint arXiv:1811.07522*.

Yeh, C. Y., Huang, C. W., & Lee, S. J. (2011). A multiple-kernel support vector regression approach for stock market price forecasting. *Expert Systems with Applications*, 38(3), 2177-2186.

Yetilmezsoy, K., & Demirel, S. (2008). Artificial neural network (ANN) approach for modeling of Pb (II) adsorption from aqueous solution by Antep pistachio (Pistacia Vera L.) shells. *Journal of hazardous materials*, 153(3), 1288-1300.

Yoshihara, A., Fujikawa, K., Seki, K., & Uehara, K. (2014, December). Predicting stock market trends by recurrent deep neural networks. *In Pacific rim international conference on artificial intelligence* (pp. 759-769). Springer, Cham.

Yoon, Y., & Swales, G. (1991, January). Predicting stock price performance: A neural

network approach. *In System Sciences, 1991. Proceedings of the Twenty-Fourth Annual Hawaii International Conference on* (Vol. 4, pp. 156-162). IEEE.

Yoon, Y., Guimaraes, T., & Swales, G. (1994). Integrating artificial neural networks with rule-based expert systems. *Decision Support Systems*, 11(5), 497-507.

Yu, H., Chen, R., & Zhang, G. (2014). A SVM stock selection model within PCA. *Procedia computer science*, 31,.406-412.

Zahedi, J., & Rounaghi, M. M. (2015). Application of artificial neural network models and principal component analysis method in predicting stock prices on Tehran Stock Exchange. *Physica A: Statistical Mechanics and its Applications*, 438, 178-187.

Zhang, Y., & Wu, L. (2009). Stock market prediction of S&P 500 via combination of improved BCO approach and BP neural network. *Expert systems with applications*, 36(5), 8849-8854.

Zhao, L., Cheng, L., Wan, Y., Zhang, H., & Zhang, Z. (2015). A VAR-SVM model for crude oil price forecasting. *International Journal of Global Energy Issues*, 38(1-3), 126-144.

Zhiqiang, G., Huaiqing, W., & Quan, L. (2013). Financial time series forecasting using LPP and SVM optimized by PSO. *Soft Computing*, 17(5), 805-818.

Zhu, X., Wang, H., Xu, L., & Li, H. (2008). Predicting stock index increments by neural networks: The role of trading volume under different horizons. *Expert Systems with Applications*, 34(4), 3043-3054.