FLORIDA STATE UNIVERSITY

COLLEGE OF ARTY SCIENCE

ENSEMBLE METHODS FOR CAPTURING DYNAMICS OF LIMIT ORDER BOOKS

By

JIAN WANG

A Dissertation submitted to the
Department of mathematics
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

2017

Copyright © 2017 Jian Wang. All Rights Reserved.

Jian Wang defended this dissertation on July 31, 2017.
The members of the supervisory committee were:

Jinfeng Zhang

Professor Directing Thesis

Capstick Simon

University Representative

Giray Ökten

Committee Member

Alec Kercheval

Committee Member

Washington Mio

Committee Member

The Graduate School has verified and approved the above-named committee members, and certifies that the dissertation has been approved in accordance with university requirements.

ACKNOWLEDGMENTS

I am greatly indebted to many people during my life to pursue this degree.

First and foremost, I want to express my gratitude to Dr. Jinfeng Zhang,my thesis major professor. Without his patience and insight, it is impossible to complete this thesis. His profound knowledge in statistics and programming, smart vision in research direction and passion for both work and life has benefited me a lot. Dr. Zhang has provided every help I would imagine from a research advisor and it was an enjoyable experience of working with him.

I also deeply appreciate my co-advisor Dr. Giray Ökten, he put a lot of effort on this thesis and provided much insightful advice. Besides the solid knowledge of Monte Carlo methods he taught me, he also helped me so much from the very first day that I joined the department, including selecting courses,career preparation, and job search. I feel very fortunate to have Dr. Ökten as my co-advisor for accomplishing this thesis.

I want to extend my thanks to Dr.Alec N. Kercheval and Dr. as my committee members and every course I had at our department in Florida state university is a wonderful experience and I would like to thank all my instructors for their patience and effort: Dr.Kopriva, Dr. Nichols,Dr. Ewald, Dr. Aldrovandi ,Dr.Fahim, Dr.Gallivan,Dr.Kim, Dr. Sussman and Dr. Zhu. Those courses help me to cultivate solid background of financial math, numerical analysis and programming.

Furthermore, I owe my special thanks to Dr. Penelope Kirby, Dr. Penny LeNoir, Ms.Blackwelder, Mr. Dodaro, Mr. Grigorian and Mr. Wooland, who gave me a lot of instructions and suggestions when I worked as a teaching assistant in our department and mathematics lab. They are all very nice and point out my weak points during work, which gave me a good opportunity to enhance my presentation skills and teaching abilities.

I also want to thank my colleagues and friends at department: Yang Liu, He Huang, Jinghua Yan, Bo Zhao, Jian Geng, Ming Zhu, Wanwan Huang, Yuan Zhang, Wen Huang, Yaning Liu,Qiuping

Xu, Guifang Zhou, Pierre Garreau, Ahmed Islim, Dawna Jones, Linlin Xu, Yuying Tzeng, Szu-Yu Pai, Nguyet Nguyen, Dong Sun, Daozhi Han, Yuanda Chen, Chun-Yuan Chiu, Chenchen Zhou, Yao Dai, Chaoxu Pei, Fangxi Gu, Zailei Cheng, Jian Li, Xinru Yuan, Haixu Wang, Wei Meng and Qi Si for their help, cooperation and meaningful suggestions to my research.

Finally, I want to thank my father, Xinshan Wang, my mother, Hua Yang and my other family members for their supports during my Ph.D study period. They are the origin of all my motivation, confidence and strength.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

According to rapid development in information technology, limit order books(LOB) mechanism has emerged to prevail in today's financial market. In this paper, we propose ensemble machine learning architectures for capturing the dynamics of high-frequency limit order books such as predicting price spread crossing opportunities in a future time interval. The paper is more data-driven oriented, so experiments with five real-time stock data from NASDAQ, measured by nanosecond, are established. The models are trained and validated by training and validation data sets. Compared with other models, such as logistic regression, support vector machine(SVM),our out-of-sample testing results has shown that ensemble methods had better performance on both statistical measurements and computational efficiency. A simple trading strategy that we devised by our models has shown good profit and loss(P&L) results. Although this paper focuses on limit order books, the similar frameworks and processes can be extended to other classification research area.

Keywords: limit order books, high-frequency trading, data analysis, ensemble methods, F1 score.

CHAPTER 1

INTRODUCTION

1.1 High frequency trading system

1.1.1 Evolution of high-frequency trading

Over the last few decades, information technology, including computing speed and memory volume, has made a great development. According to this trend, a new class of trading system, which is called high-frequency trading (HFT), has appeared to today's financial markets. Generally speaking, HFT represents a program trading platform that uses powerful computers to transact a great number of orders at very fast speeds. It becomes more and more attractive because of some key factors:

- **Narrowing Spreads**. In 2001, the unit of quoting prices in U.S. Stock exchanges changed from fractions to decimals. So the minimum spread between the bid and ask prices decreased from 1/6th of a dollar (6.25 cents) to one cent. The change of price unit provides traders better alternatives to seeking spread arbitrages, which results in a strong boost in an algorithmic trading system.

- **Regulation changes**. In 2005, the Securities and Exchange Commission(SEC) passed the Regulation National Market System(Reg.NMS), which improved transparency and competition among different financial markets. Besides, this regulation also required trade orders to be posted nationally instead of at individual exchanges. So traders can be beneficial of profit from small price difference of one security among different exchanges.

High-frequency trading is an extension case of algorithmic trading, which turns over small positions of security very frequently. The U.S. Securities and Exchanges Commission conclude specific characteristics of HFT:

- Submit some orders and cancel them soon after the submission.

- Maintain very few or no overnight positions.
- Maintain a very high turn over rate of a very small positions of a specific security or evena pool of securities. Meanwhile the holding period for each position is also very short.
- Utilize complicated and high-performance computing program to generate, execute or cancel orders.
- Utilize individual data from exchanges and servers that belong to co-location provider that minimize network or other types of latencies.

According to recent survey(Agarwal (2012)), high-frequency trading has taken a great number of share in U.S. and European equity trading volume. As shown in figure 1.1, in U.S., the percentage of HFT in equity turnover by volume maintained growth trend overall from year the 2005 to 2010. For example, in 2010, HFT occupied 56% by volume of the entire equity turnover, increased from 21 % in 2005.

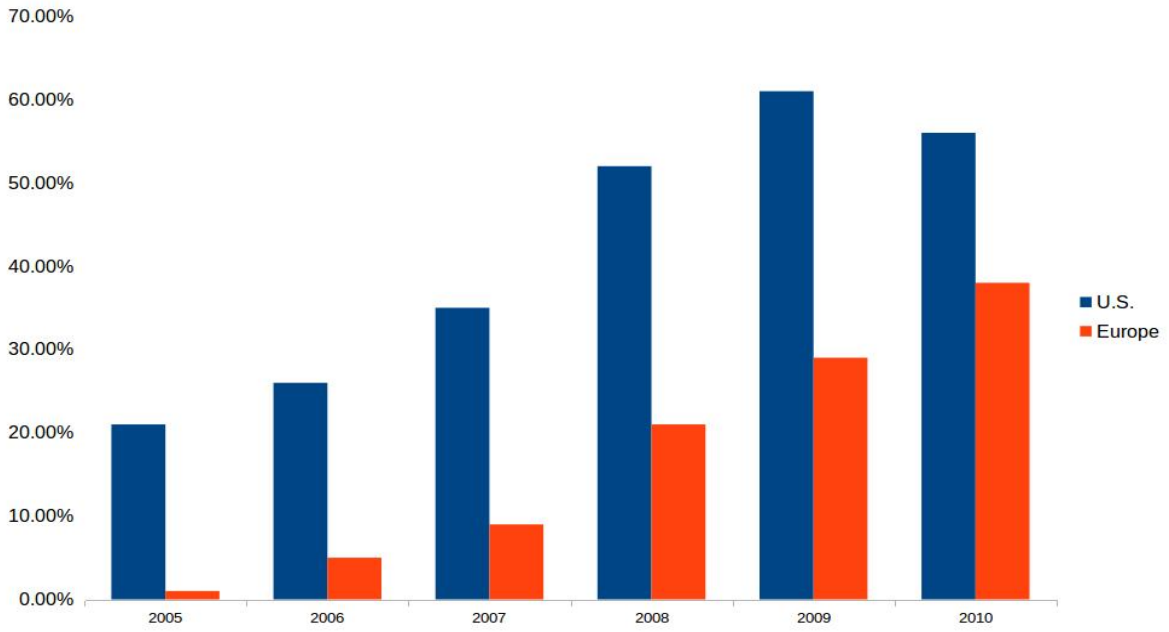


Figure 1.1: High frequency trading as a % of equity turnover by volume, U.S. and by value, Europe 2005-2010

A similar situation happened in Europe. HFT only accounted for 1% of equity turnover by value in 2005. The percentage surged to 38 % in 2010, increased from 29 % just a year ago.

1.1.2 High frequency trading strategies

Generally speaking, there are two main high-frequency trading strategies: passive and aggressive trading strategies. Passive strategies use limit orders which let the brokerage to buy or sell the stocks at a specified level price, while aggressive strategies utilize market orders to buy or sell the stocks immediately. In the following, we give the description of main types of these two trading strategies.

For passive HFT strategies, the first widely used method is passive market making. This method allows the market maker to purchase a company's securities and, at the same time, the market maker is also acted as an underwriter of the securities in a secondary public offering. Since the market maker can place bids of a security before its publication, the real buyers would have to place their buy orders higher than the market maker's bid price. Therefore the market maker can benefit from this higher opening. The second method for passive strategies is arbitrage trading. As described by its name, this method makes a profit by the price difference of the same or related securities. A simple example is as follows: the stock of one company that we called X is trading at \$10 on the New York Stock Exchange(NYSE), while at the same moment it is trading at the price of \$10.05 on the London Stock Exchange(LSE). Given that a trader can trade the stock on both markets with no time lag, he can buy one specific security on the NYSE and sell the same shares on the LSE just after buying. Obviously, a profit of 5 cents without risk can be earned by him. Those arbitrage opportunity will persist until the specialists on the NYSE or LSE adjust their price to eliminate the price difference.

The main type of aggressive HFT strategies consists of the following two types: Momentum ignition and order anticipate. The former one is a strategy that a proprietary trading firm buy or sell volumes of orders that will cause the price of underlying security significantly going up or down shortly. Such quick submission and cancellation of many orders of a stock will trigger other traders' algorithm to buy or sell the same stock more aggressively. After the trend of price movement is made in the market, the momentum maker will benefit from selling the stock at a higher price or buying the stock at a lower price. However it is very difficult to distinguish between momentum ignition and *spoof*, which was defined as illegal according to Dodd-Frank Act. For the order anticipate method, it can be described as a liquidity detection trading which confirms the existence of large institutional buys or sellers in the marketplace and then trades ahead of these

buyers or sellers in anticipation that their large orders will move market prices(Securities and Exchange Commission,2014,p.8). The line between this method and another illegal action called *front-running* can be nuanced. The *front-running* is the unethical practice that a stockbroker trades securities in his personal account based on the knowledge of advance knowledge of pending orders from its customers.

1.2 Limit order book dynamics

According to the above section, we know that under the aggressive strategies situation, it is very likely that the strategy will be deemed as illegal. Therefore choosing passive strategies in high-frequency trading is a good idea. Since most transactions in the passive trading strategies are buying and selling the securities at a particular price, limit order books play an indispensable role in this strategy. Actually, in today's financial market, more than half of stock exchanges now use a limit order book(LOB) mechanism to facilitate trade(Rosu (2009)). Some exchanges, such as Helsinki, Hong Kong, Shenzhen, Swiss, Tokyo, Toronto, and Vancouver Stock Exchanges, now use pure LOBs(Luckock (2001)). Some exchanges use hybrid of hybrid LOBs, which include the New York Stock Ex-change (NYSE), NASDAQ, and the London Stock Exchange (LSE) (Cont et al., 2010).

As described above, we can see that LOBs play an major role in financial trading architectures, it is beneficial for both scholars and practitioners to understand dynamics of LOB. The advantages of capturing dynamics of LOBs include: finding optimal opportunity to execute orders(Obizhaeva and Wang, 2013); improving the performance of electronic trading algorithms(Engle et al., 2006); Obtaining a better understanding of micro market structure for Practitioners(Harris, 2003); Getting a clearer insight into market volatility(Kirilenko et al., 2015).

In this thesis, we first introduce basic definitions of LOBs, including math definition of the bid-ask spread, mid price, bid and ask side depth, spread crossing opportunities and so on. Then we use statistical and data driven methods,especially in the area of machine learning architectures, to model the future arbitrage opportunities of LOBs. We focus on ensemble machine learning algorithms to build our prediction models. As far as we know, there is no evidence shows that these methods were used in LOBs research area. The introduction of ensemble machine learning

algorithms will be given in the following section of this chapter and details can be found in chapter 6.

1.3 Ensemble method for classifiers

In spite of a great amount of research on limit order books, there is only a few literature which utilizes machine learning methods for capturing the limit order books. Furthermore, based on our knowledge, there is little evidence that ensemble methods were allied to this topic. In our research, we try to use AdaBoost(Adaptive Boosting) method, which was deemed as the best classifier off the shelf(Kégl, 2013), and random forest method to predict spread crossing over opportunities of LOBs.

Generally speaking, an ensemble classifier contains a set of individually trained classifiers(such as neural networks or decision trees) to get a better predictive performance by combining the predicting results of each classifier. Some past research shows that ensemble classifier is more accurate than any of the individual classifier which constitutes the ensemble. For example, Hansen and Salamon (1990) and Hashem (1997) have conducted both theoretical and empirical research which demonstrated that a good ensemble of neural networks is more accurate than its primary classifier.

Like other machine learning classifiers, ensemble methods are double edged swords. The main advantage of ensembles is that there is a little probability that all classifiers will make the same mistake. Actually, if each error is made by a minority of the classifiers, you can obtain an optimal classification. Besides, ensembles are very likely to reduce the variance of classifiers. Therefore, if the classification algorithms are sensitivity to small changes in the training data, ensemble methods tend to be very helpful. The disadvantages of ensemble methods are also notable, the biggest might be the lack of interpretation(Bühlmann, 2012). A linear combination of an individual classifier is much harder to interpret than a single classifier.

In our research, we utilize two typical ensemble algorithms, AdaBoost and random forest, to train and predict the labeled spread crossing over in a fixed time interval of LOBs. The price spread crossing opportunities can be labeled as ask price lower, bid spread higher and no crossing over in a fixed time horizon. Therefore our problem is a multi-class classification case, the one against one and one against all methods will be introduced to solve the multi-class classification

problem. The real time data is divided into two part with the percentage of 9:1, to make a training dataset and testing data set respectively. Besides, features with price, volume and order book arrival intensity of each price level are created,so every data sample in training and the testing dataset is represented as a vector of features. Moreover, Precision, recall and F1 score are used to measure the performance of the models, since the existence of arbitrage in a relatively long time interval is rare and our dataset can be treated as an imbalanced data.

Experiments with real time data from NASDAQ show that the ensemble models built in our paper can not only predict the arbitrage opportunities with high accuracy, but also can improve the prediction performance compared with basic classifiers, such as logistic regression, support vector machine and decision trees. We also design a naive trading strategy in the testing time interval and demonstrate the Profit and Loss (PnL) of our models. The cumulative Pnl curve show that the traders can obtain positive return with zero investment, which indicates that the statistical arbitrages can be found in our models.

1.4 Purpose of the dissertation

The research conducted in this dissertation attempts to construct a framework that captures the dynamics of limit order books characterized by metrics such as bid ask spread and spread crossing , builds models to recognize intrinsic patterns hidden in the limit order books with the help of machine learning techniques, automates the prediction process on the evolutions of metrics under study for unseen events, and the prediction results are further exploited to guide trading strategies in real time. To make the work flow in the proposed framework completely automate without human intervention, we streamline all the processes designed in the framework including construction of training data, feature selection, model building, performance evaluation and confidence testing. To keep pace with the rapid changes of limit order book dynamics, the models constructed in the proposed framework are automatically updated by using the most up-to-date transactional events as the training data. Moreover, the application of trading strategies designed on the base of models built in the proposed framework to real situations in a real-time manner is also automated. In order to build a learning model for a given financial asset such as a stock with respect to a certain metric, the framework first automatically constructs a set of training data by extracting a variety of features from the message and order books, which could be from real world circumstances or

from simulation. To improve the efficiency of model training and prediction process on unseen events, the extracted features are further pruned according to their contributions measured in information gains to the performance of the model built. Then the learning model is built by using the ensemble techniques, especially Adaboost and random forest. The performance of the resulting model is validated by precision, recall and F 1 -measure. Finally, the learning model is employed to predict unseen events with respect to the metric in question and the predictions are subsequently used to assist the decision making in the real time trading.

Clearly, the research described in this dissertation can be viewed as an endeavor on integrating together the limit order book modeling in mathematical finance discipline and the machine learning techniques. The contributions of this dissertation are summarized as follows.

1) The framework developed in this dissertation is highly efficient on both training models and classifying unseen data points. For instance, an event can be labelled in far less than a tenth of a millisecond, which is vitally important for helping make real-time trading decisions in real world circumstances.

2) Instead of predicting future event price change, we focus on predicting the price change for future fixed time interval. In practice, the time of future event coming is hard to predict therefore it is very difficult to conduct a trading strategy based on future event. Therefore forecasting the price change based on future time interval is of practical significance.

3) We use real time stock data in a very low latency, the minimum time change can be nano-second. Besides, the data volume that we used is also very high, each stock will have millions number of data. Therefore, the results of our experiment are more credible and convincing.

4) Adaboost and random forest technique are used to improve the prediction performance. The experimental results show that ensemble methods will improve the model efficiency in a significant way compared with the basic machine learning tools.

To better present the above-described contributions, this dissertation is organized as follows.

Chapter 2:Literature Review:Briefly reviews previous modeling methods, mainly statistics and machine learning approaches, on limit order markets, then surveys the ongoing research in simulating the dynamics of limit order books.

chapter 3:Mathematical description and statistical properties of limit order books: List the main math definition of limit order books, such as bid price, ask price, and bid-ask spread. Besides some statistical properties are also mentioned in this chapter.

Chapter 4: Basic machine learning tools: Introducing some classic machine learning tools that we use as benchmarks to the ensemble methods. Those tools include: logistic regression, lasso regression, ridge regression, support vector machine and decision tree.

chapter 5: Ensemble methods for prediction: Introducing properties of ensemble methods,especially for adaboost and random forest.

chapter 6: Model framework and results:In this chapter,we describe the data set that we use to train and test the model, and the framework of our models. Besides, the measurement for model performance such as precision, recall and F1 score are also introduced here. Finally, we also show the profit and loss of a simple trading strategy by using our model to predict the future arbitrage opportunities.

CHAPTER 2

LITERATURE REVIEW

In this chapter, we review the past research in finding the laws of limit order books and applying machine learning tools into financial markets. In the first part we summarize the papers which describe the models and rules of limit order book. The research work of using machine learning methods to capture the dynamics of limit order book is generalized in the second part.

2.1 Limit order book dynamics and modeling

There are a lot of research work on capturing limit order book dynamics, including theoretical and experimental results. Through simulating the effect of different order book depth,Bollerslev and Domowitz (1993) discover that when the order book depth increases, the autocorrelation of spreads will also rise.

Hamao and Hasbrouck (1995) studied the patterns of intra-day trades and quotes for part of stocks on the Tokyo Stock Exchange. They claim that an order that is waiting to be executed has a greater impact on the price movement than the one that is executed immediately. Therefore, after the market orders are executed, the market price is likely to move in the same direction. Besides, they find that there is always some delay in execution when the market orders become the limit orders. After that, they also investigated the limit orders in NYSESuperDOT market and find that the market orders often have worse performance than the limit orders that are placed at or above the best quote.

Maslov and Mills (2001) investigate the order book data from NASDAQ and discover that the number of limit orders placed on the bid sides and the number of limit orders placed on the ask sides are often largely unequal, the disequilibrium will cause the price change in the short term.

Bouchaud et al. (2002) studied three stocks in the Paris Bourse and find that the price of order books obey a power law near the price at the present time and the mean of the distribution is diverging.

Zovko et al. (2002) study around two million orders from the London Stock Exchange. They found that a power law distribution can be applied to the difference between the limit price and the best price. Besides, the volatility of the price is positively correlated with the relative limit price levels.

Potters and Bouchaud (2003) investigate the order book data in NASDAQ and discover that the tail of the price of the incoming order shows a very slow recession. Besides, they also find that rather than the power-law, the link between price response and volume follows a logarithmic distribution in British and Fresh financial market.

In Australian stock market, Hall et al. (2004) find that the order book depth and the market activity will significantly influenced the buy-side pressure as well as the sell-side pressure. Therefore,traders tend to utilize the information inside the order book to infer other market participants trading strategies.

Weber and Rosenow* (2005) studied the order books on the Island financial market. They find that the price impact function, which is used to describe the phenomenon that the stock trading will cause price changes, is convex. They discover that price changes and order flow is strongly anti-correlated, which will reduce the price impact of market orders.

Boehmer et al. (2007) use sample stock trading data in different markets to verify if past execution quality will affect order routing decisions. Their results show that routing decisions are related to the execution quality. The market will receive more orders if their execution costs are low and can fill orders immediately.

Ganchev et al. (2010) and Laruelle et al. (2011) introduce a numerical algorithm to optimally split orders across liquidity dark pools. They also give the experimental validation of the algorithm by using execution data in a dark pool from a brokerage.

A continuous time stochastic model was proposed by Cont et al. (2010) to capture the dynamics of a limit order book. Matrix computation and Laplace transform methods are used to calculate the probability of different events. Numerical results with real-time high frequency data are also provided to show the efficiency of their model to capture the short-term dynamics of a limit order book.

Guo (2013) provide a solution of an optimal placement problem in a limit order book. Two models, with or without price impact, are proposed in their paper. Besides, they give the solutions of both single-period and multi-period situations.

Cont and Kukanov (2013) also study the optimal order placement problem, they transform this problem to a convex optimization problem. They consider the fee structure, the state of order books, the order flow properties and the preference of a trader. Besides, they provide an explicit solution in the single exchange situation and propose a stochastic method in the multiple exchange situation.

2.2 Machine learning methods on capturing limit order book dynamics

There is not much past research in applying the machine learning tools to limit order books or financial applications in general. To my best knowledge, the earliest research in this area can be traced back to Hutchinson et al. (1994). They propose a nonparametric method,learning network, to price and hedge options. Besides, real-time data sample(S&P 500 futures options from 1987 to 1991) was used to test the efficiency of their network pricing method.

Nevmyvaka et al. (2006) study the millisecond limit order data from NASDAQ and propose a reinforcement learning model to conduct a optimized trade execution. Their results show that the reinforcement learning can significantly improve the performance compared with some simpler methods of optimization.

Kercheval and Zhang (2015) propose a support vector machine scheme to predict the movement of price for limit order books. Nano-second time scale data from NASDAQ was used to build their model and feature selection process was conduct to show the importance of each feature. They also design a single buy-low and sell-high trading strategy to verify the practical value fo the model. The result of profit and loss(PnL) through this strategy is promising.

Park and Van Roy (2015) demonstrate a confidence triggered regularized adaptive certainty equivalent (CTRACE) policy for excution and learning on the same time and propose a reinforcement learning algorithm for improving the efficiency in linear-quadatic control problems. Monte Carlo simulation is utilized to show the CTRACE method is better than the certainty equivalent policy.

Sirignano (2016) build a deep neural network algorithm in $\mathbb{R}^d$ space. Around 500 U.S. stocks are trained and tested and a cluster with 50 GPUs is used to accelerate the speed of computing. The experimental results show that their model outperforms the logistic regression model with non-linear features, empirical model, and a standard neural network model.

Some other related literatures include: using hidden markov chain models into financial market, for example,Idvall and Jonsson (2008), Hassan et al. (2007),Hassan and Nath (2005),Landen (2000),Mamon and Elliott (2007),Zhang (2004),Bulla (2006),Rossi and Gallo (2006),Rydén et al. (1998); using artificial neural networks to make prediction in financial markets,e.g.,Trippi and Turban (1992),Kuan and Liu (1995),Walczak (2001), Shadbolt (2002) ;using support vector machine combined with different kernels to predict the financial time series changes. Those work can be found in Cortes and Vapnik (1995),Tay and Cao (2001),Tay and Cao (2002),Cao (2003),Kim (2003), Pérez-Cruz et al. (2003),Huang et al. (2005),Van Gestel et al. (2001),Hazarika and Taylor (2002),Tino et al. (2005),Huang and Wu (2006),Huang and Wu (2008),Fletcher et al. (2009),Chalup and Mitschele (2008); predicting price change in financial market by use of Kalman Filtering. Related work can be found in Fletcher (2007), Julier and Uhlmann (1997),Evensen (2003),Bolland and Connor (1996),Bolland and Connor (1997)

CHAPTER 3

MATHEMATICAL DESCRIPTION AND STATISTICAL PROPERTIES OF LIMIT ORDER BOOKS

In this chapter, we give precise mathematical descriptions of LOBs' trading and exchanging principles. Besides,some basic statistical properties of limit order books, which have been proposed by past research, are also studied and described. Those properties include:size of orders, shape of order books,time of arrival of orders, placement of orders and so on. Study for those properties will help us to get a clearer insight of how to capture the dynamics of LOBs. For example, by knowing the distribution of arrival of orders will help us to build more meaningful features in our prediction models.

3.1 Mathematical descriptions of LOBs

An LOB can be represent as a three dimensional vector by the following definition:

Definition 3.1.1 *An order $x = (p_x, \omega_x, t - x)$ submitted at time t_x which price p_x and size $\omega_x > 0$(respectively, $\omega_x < 0$) is a commitment to sell (respectively, buy) up to $|\omega_x|$ units of the traded asset at a price no less than (respectively, no greater than) P_x.*

For a given LOB, the units of order size and price are defined as follows:

Definition 3.1.2 *The lot size of an LOB is the smallest amount of the asset that can be traded within it. All orders must arrive with a size $\omega \in \{\pm k\sigma | k = 1, 2, ...\}$*

Definition 3.1.3 *The tick size π of an LOB is the smallest permissible price interval between different orders within it. All orders must arrive with a price that is specified to the accuracy of π*

For example, if $\pi = 0.01$, then the largest permissible order price that is strictly less than \$1 is \$ 0.99, and all orders must be submitted at a price with exactly two decimal places.

Definition 3.1.4 *The lot size σ and tick size π of an LOB are collectively called its resolution parameters.*

Definition 3.1.5 *When a buy (respectively, sell) order x is submitted, an LOB's trade-matching algorithm checks whether it is possible to match x to some other previously submitted sell(respectively, buy) order. If so, the matching occurs immediately. If not, x becomes active*

Active orders in a market make up an LOB:

Definition 3.1.6 *An LOB $\mathcal{L}(t)$ is the set of all active orders in a market at time t.*

An LOB can be treated as a set of queues, each of which contains active bid or ask orders at a specified price.

Definition 3.1.7 *The bid-side depth available at price p and at time t is:*

$$n^b(p,t) := \sum_{\{x \in \mathcal{B}(t) | p_x = p\}} \omega_x$$

The ask-side depth available at price p and at time t, denoted $n^a(p,t)$, is defined similarly using $\mathcal{A}(t)$

The depth available is often demonstrated as multiples of the lot size. Since $\omega_x < 0$ for bid orders and $\omega_x > 0$ for ask orders, it implies that $n^b(p,t) \leq 0$ and $n^a(p,t) \geq 0$ for all prices p.

Definition 3.1.8 *The bid-side depth profile at time t is the set of all ordered pairs(p,$n^b(p,t)$). The ask-side depth profile at time t is the set of all ordered pairs(p,$n^a(p,t)$).*

Definition 3.1.9 *The mean bid-side depth available at price p between times t_1 and t_2 is*

$$\bar{n}^b(p,t_1,t_2) := \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} n^b(p,t)dt$$

The mean ask-side depth available at price p between times t_1 and t_2, denoted as $\bar{n}^a(p,t_1,t_2)$, is defined similarly using the ask-side depth available.

In the following, we define the terms of *bid price, ask price, mid price, and bid-ask spread*

Definition 3.1.10 *The bid price at time t is the highest stated price among active buy orders at time t,*

$$b(t) := \max_{x \in \mathcal{B}(t)} p_x$$

The ask price at time t is the lowest stated price among active sell orders at time t,

$$a(t) := \min_{x \in \mathcal{A}(t)} p_x$$

Definition 3.1.11 *The bid-ask spread at time t is $s(t) := a(t) - b(t)$*

Definition 3.1.12 *The mid price at time t is $m(t) := [a(t) + b(t)]/2$*

More clearly, in an LOB, $b(t)$ is the highest price at which it is immediately possible to sell at least the lot size of the traded asset at time t, and $a(t)$ is the lowest price at which it is immediately possible to buy at least the lot size of the traded asset at time t. Considering prices relative to $b(t)$ and $a(t)$ is helpful in some cases.(Gould et al., 2013)

Definition 3.1.13 *For a given price p, the bid-relative price is $\delta^b(p) := b(t) - p$ and the ask-relative price is $\delta^a(p) := p - a(t)$*

Note that here is difference in signs between the definition of δ for the two sides:$\delta^b(p)$ defines how much smaller that p is less than $b(t)$ and $\delta^a(p)$ illustrates how much larger that p is greater than $a(t)$. It is useful that we compare the properties of orders on both the bid side and the ask side of an LOB.

Definition 3.1.14 *For a given order $x = (p_x, \omega_x, t_x)$, the relative price of the order is :*

$$\delta^x := \begin{cases} \delta^b(p_x) \text{ if the order is a buy order,} \\ \delta^a(p_x) \text{ if the order is a sell order,} \end{cases}$$

Definition 3.1.15 *The bid-side depth available at relative price p and at time t is:*

$$N^b(p,t) = \sum_{\{x \in \mathcal{B}(t) | \delta^x = p\}}$$

The ask-side depth available at relative price p and at time t, denoted $N^a(p,t)$, is defined similarly using $\mathcal{A}(t)$

Definition 3.1.16 *The bid-side relative depth profile at time t is the set of all ordered pairs*$(p, N^b(p,t))$. *The ask-side relative depth profile at time t is the set of all ordered pairs*$(p, N^a(p,t))$.

Definition 3.1.17 *The mean bid-side depth available at relative price p between times* t_1 *and* t_2 *is:*

$$\bar{N}^b(p, t_1, t_2) = \frac{1}{t_2 - t_1} \int_{t_1}^{t_2} N^b(p,t)$$

The mean ask-side depth available at relative price p between times t_1 *and* t_2 *denoted* $\bar{N}^a(p, t_1, t_2)$ *is defined similarly using the ask-side relative depth available.*

Definition 3.1.18 *The mean bid-side relative depth profile between times* t_1 *and* t_2 *is the set of all ordered pairs*$(p, \bar{N}^b(p, t_1, t_2))$. *The mean ask-side relative depth profile between times* t_1 *and* t_2 *is the set of all ordered pairs* $(p, \bar{N}^a(p, t_1, t_2))$

while most traders use the relative depth profile to measure the phenomenon of LOBs, some research has claimed that the order arrival rates relies on relative prices rather than actual prices(Biais et al. (1995),Bouchaud et al. (2002),Potters and Bouchaud (2003),Zovko et al. (2002)), relative depth profiles do not contain information about the absolute prices at which trades occur. Moreover, relative depth profiles provide little information about the bid-ask spread and mid price. Therefore, it is better that we combine the relative depth profiles, bid price($b(t)$) and ask price($a(t)$) together when we consider the problem of LOBs. A completed view of the evolution of an LOB can be obtained if we consider all the information simultaneously.

3.2 Properties of LOBs

During the last two decades, financial market has become more and more computerized. Therefore, it is easier for us to access extensive data on order books. To our knowledge, Biais et al. (1995) is the pioneer to study the computerized data flows of Paris Bourse. After that, lot of related papers provide more empirical findings,statistical properties and modeling views.(See,e.g.Gopikrishnan et al. (2000),Challet and Stinchcombe (2001),Maslov and Mills (2001),Bouchaud et al. (2002),Potters and Bouchaud (2003)), In this section, we give a briefly introduction of some basic empirical study results. Those fundamental statistical properties can be found through observing the real

time data. Some results of observations, such as time of arrival of orders, placement of orders, size of orders, shape of orders books and etc, are essential for calibrating the models of order flows and capturing the dynamics of order books.

3.2.1 Time of arrivals of orders

We compute the cumulative distribution for inter-arrival times of market orders for four stocks: Amazon,Google, Intel and Microsoft. The results are plotted in figure 3.1. From the figure, it is obvious that the Lognormal and Exponential distribution are not good fits. For the Weibull distribution, it has been suggested in Ivanov et al. (2004).From our results, we can see that weibull distribution(red line in our figure) is relatively a good fit to the original data. However, in our case, the Gamma distribution is the best fit for each stock.

In many past literatures, the non-Poisson arrival times models have been introduced to deal with the "irregular" financial data. For example,Engle and Russell (1997) and Engle (2000) have proposed autoregressive condition intensity models, which are beneficial to capturing the processes of orders' submission. Another research area that deal with the non-exponential arrival times relies on the branches of stochastic processes(see,e.g. Clark (1973);Silva and Yakovenko (2007),Huth and Abergel (2012)).

3.2.2 Volume of orders

Some past researches found that the distribution of order book sizes is difficult to measure. A widely accepted rule is power-law distribution.Gopikrishnan et al. (2000) and Maslov and Mills (2001) find that the market orders follows a power law which decay with an exponent $1+\mu \approx 2.3 \rightarrow 2.7$, and limit orders follows a power law which decay with an exponent $1+\mu \approx 2.0$. Challet and Stinchcombe (2001) pay more attention to a clustering effect, that is, "round" size in numbers of shares of orders is often be observed. Moreover, clusters often occur around 100's and 1000's. Till now, those research models have not reach a consensus. It makes sense that the distribution will vary greatly depending on product and market changes

In figure3.2, the distributions of number of orders in 5 minutes of four stocks are plotted. We use their mean value to nomalize the data. The parameter that we used in power law is negative 2.1 and the parameter that we used in exponential distribution is $1+\mu \approx 2.7$, which we have mentioned above. We can see from the figure,in general, the exponential distribution is better than

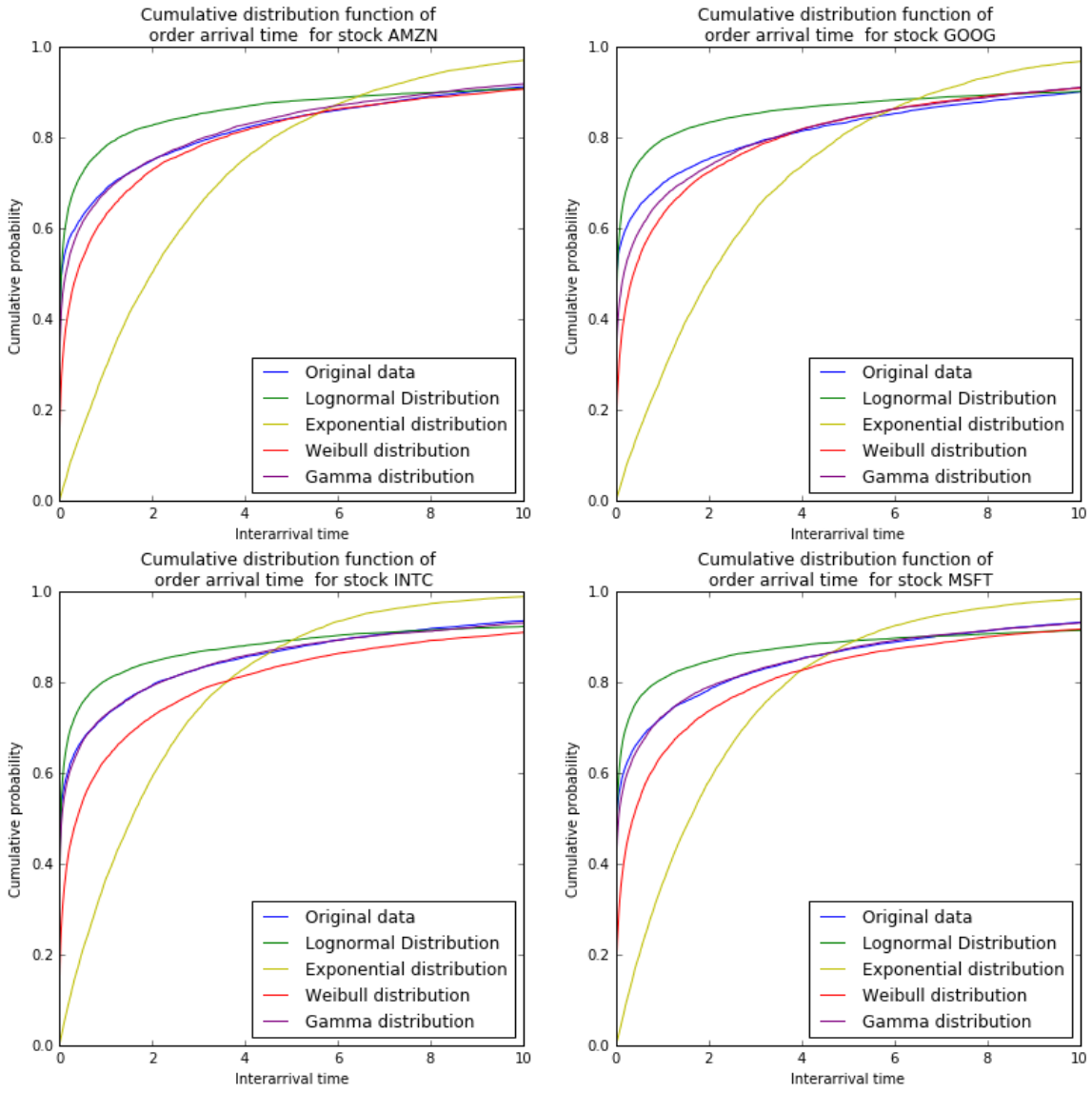


Figure 3.1: Cumulative distribution of inter-arrival time for stock: AMZN,GOOG,INTC and MSFT. In each panel,four distribution, Lognormal, Exponential Weibull and Gamma, are compared with the original dataset. x axis represents the inter-arrival time of market orders and y axis shows the cumulative probability

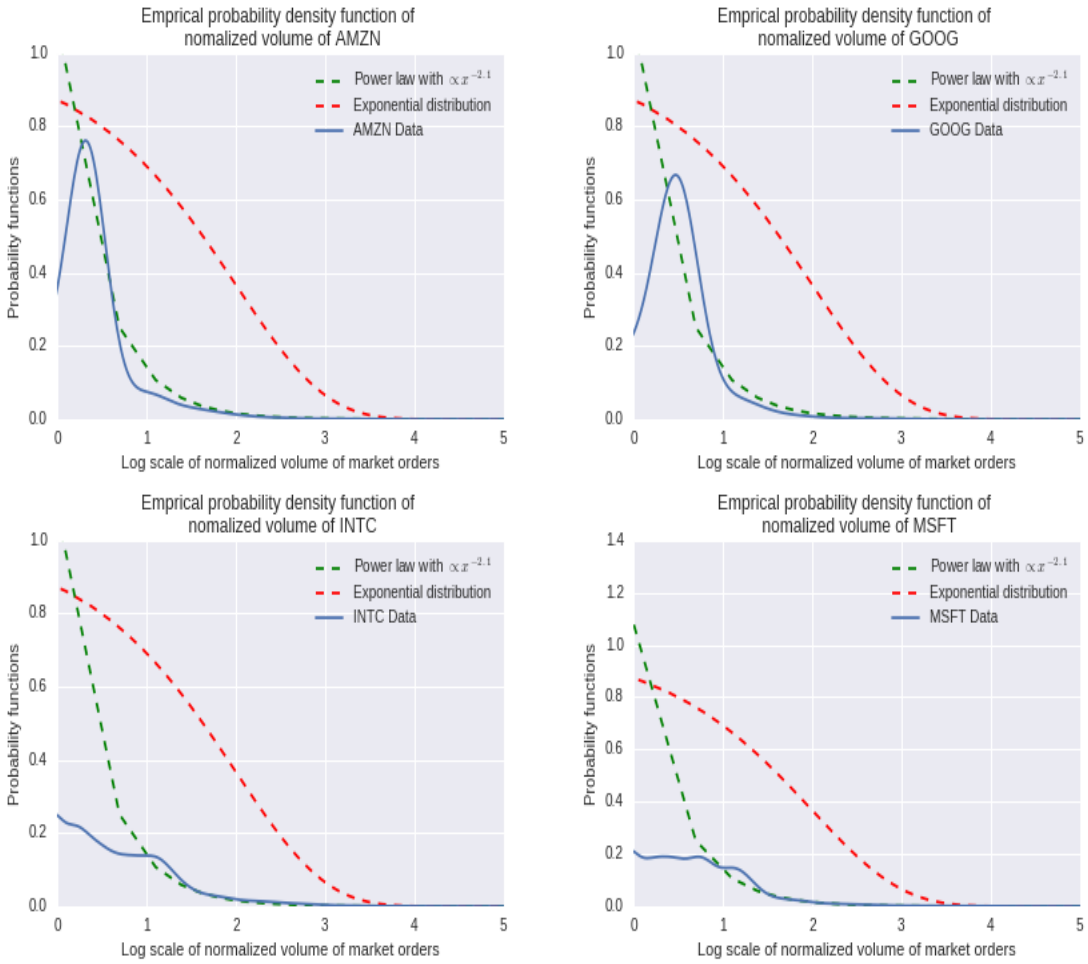


Figure 3.2: Distribution of the number of orders in a 5 minutes of four stocks: AMZN,GOOG,INTC and MSFT. In each panel,power law with parameter negative 2.1 and exponential distribution are compared with the original dataset. x axis represents the log scale of normalized volume of market orders and y axis shows the probability densities of different distributions

the power function for the fitting of empirical data. Meanwhile,for AMZN and Google, this effect is more significant, for INTC and MSFT,there is a large deviation in the initial part.

3.2.3 Placement of orders

On French stocks market, Bouchaud et al. (2002) find that the placement of order books follows a broad power-law properties. After one year, Potters and Bouchaud (2003) claimed the same property on US stocks market. The parameters of the power law are relatively stable: on the Paris Bourse, it is $1 + \mu \approx 1.6$, on the New York Stock Exchange, it is $1 + \mu \approx 2.0$ and $1 + \mu \approx 2.5$ on the London Stock Exchange. Mike and Farmer (2008) uses a Student distribution with 1.3 degree of freedom to fit the empirical distribution for limit order books.

In fig 3.3 we plot the probability distribution of placement of arriving limit orders. From this figure, we can find that the maximum probability of placement is around 0 price difference, which means the new order is more likely to place at the best price level. We also find heavy tails in the probability distribution, which means that there exists some significant price difference between the new order price and the best order price.

3.2.4 Intraday seasonality

The financial market always changes a lot during one trading day. Biais et al. (1995) study the order books in Paris Bourse and observe that the numbers of order book in a given period ΔT vary a lot during the day. Fig 3.4 and fig 3.5 describe the number of limit and market orders coming in a 5-minutes time period. From the chart, we observe a U shape of the numbers during the whole day. The dash line is an ordinary quadratic least square fit of the data). The U shape tell us that the market is active at the beginning and the end of the day, and calm down around the mid day. Potters and Bouchaud (2003) find that the average number for the market orders and limit orders in a ΔT period are highly correlated. In their papers, they use millisecond data sample and we use nano-second data, the result become similar, both market order and limit order figure demonstrate a U shape.

3.2.5 Average shape of the order book

Bouchaud et al. (2002) study the stocks in French capital market and find an extraordinary phenomenon that the maximum of average volume in an order book often deviates from the best

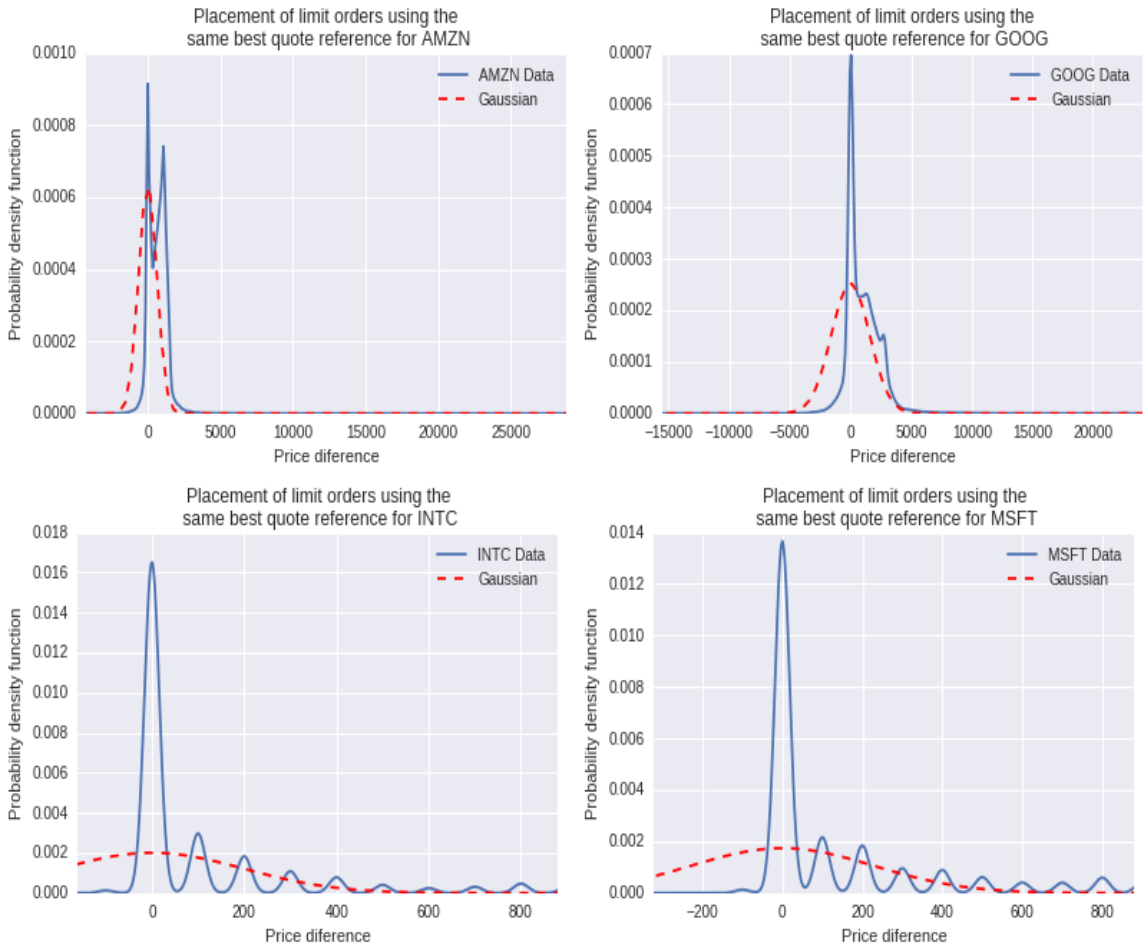


Figure 3.3: Density function of placement of limit orders using the same best quote. Four stocks that we use: AMZN,GOOG,INTC and MSFT. In each panel,Gaussian distribution are compared with the original dataset. x axis represents the price difference between the price of arrival order and the best price before the arrival of this order. y axis shows the probability density function of the order book placement

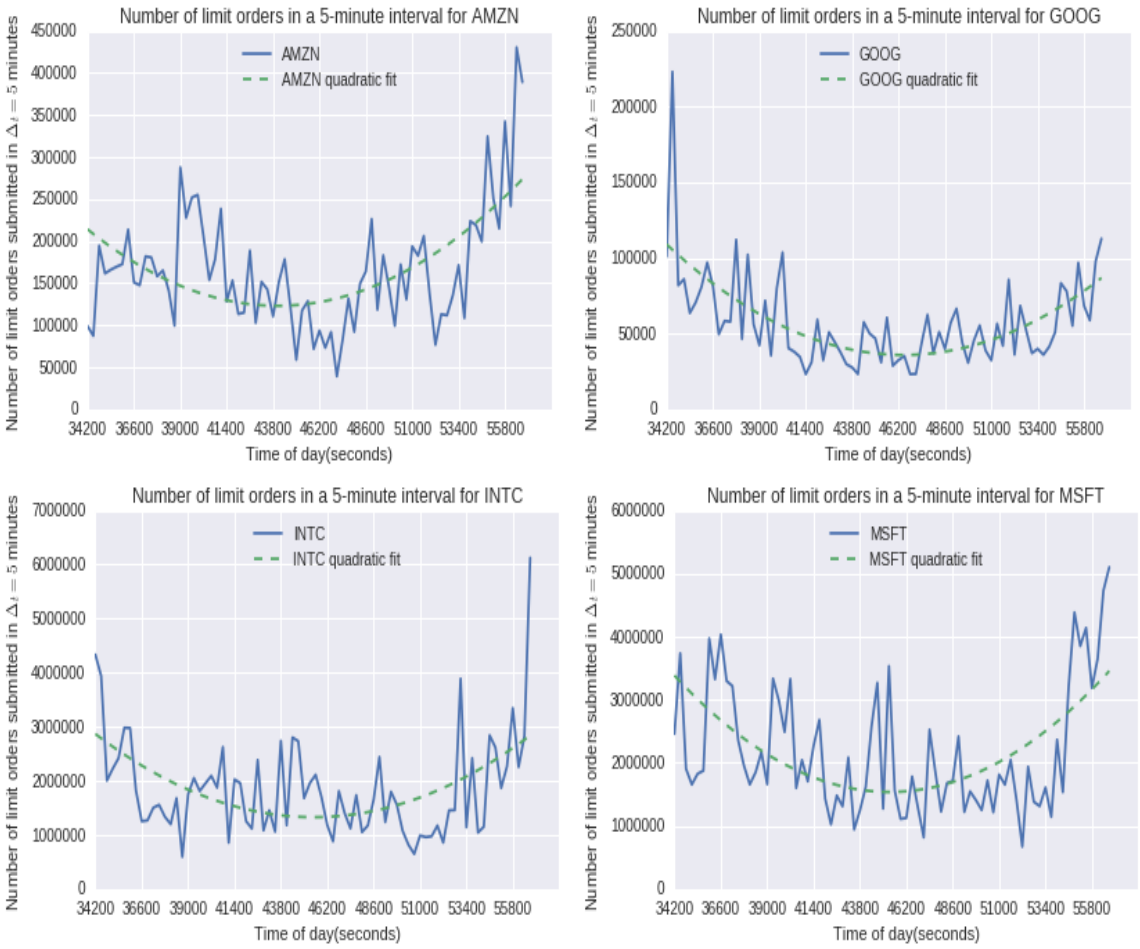


Figure 3.4: Normalized average number of limit orders in a 5-minute time period. Four stocks that we use: AMZN,GOOG,INTC and MSFT. In each panel,least square quadratic function is used to fit to the original dataset. x axis represents the time of day in seconds. y axis shows the number of limit order submitted in a 5-minute time period

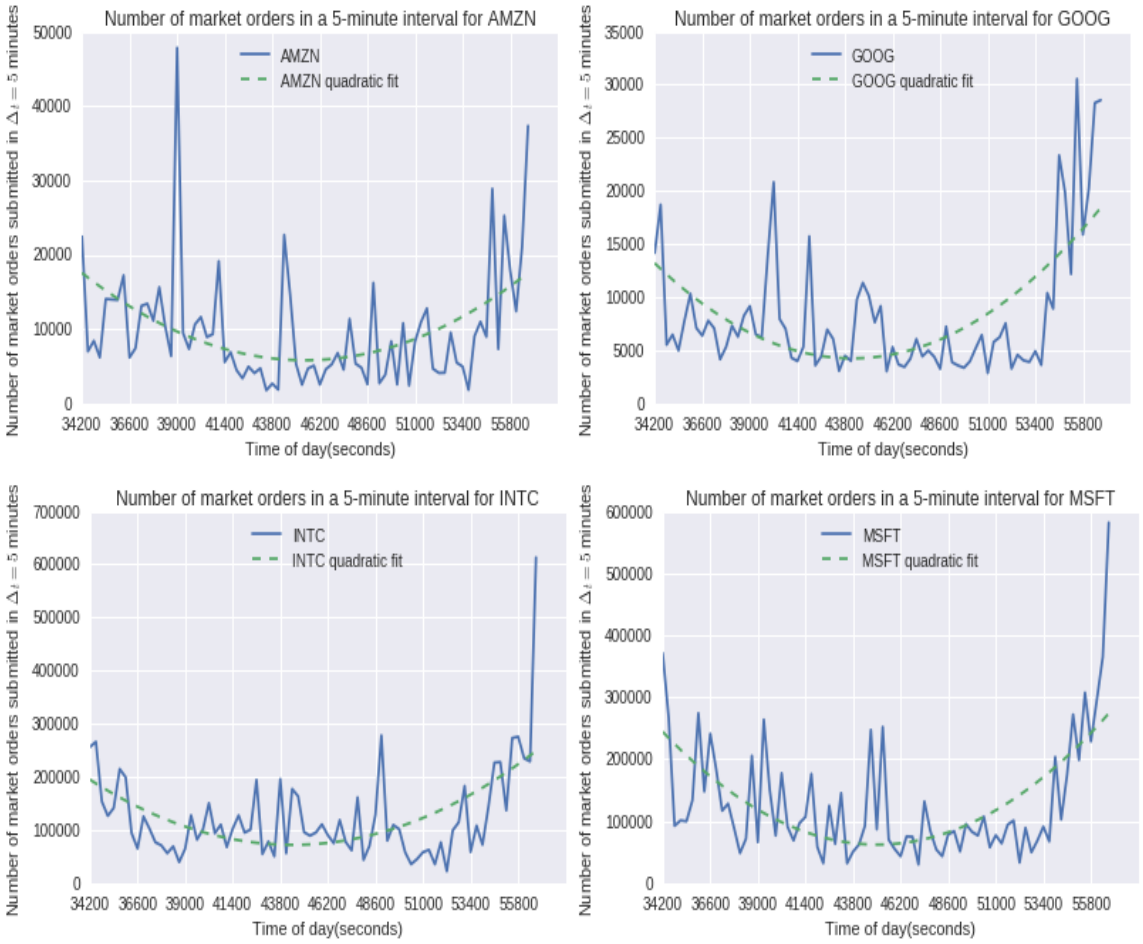


Figure 3.5: Normalized average number of market orders in a 5-minute time period. Four stocks that we use: AMZN,GOOG,INTC and MSFT. In each panel,least square quadratic function is used to fit to the original dataset. x axis represents the time of day in seconds. y axis shows the number of market order submitted in a 5-minute time period

price. In fig 3.6, we plot the average number of five stocks based on different price level. Our results show that when price level increases, the quantity of order books tend to be large, but not always. We can see that in the ask side of INTC, the minimum quantity occurs at the maximum price level. This might be caused by different trading strategies and preference between the US and French financial market.

Figure 3.6: Average limit order book. Five stocks that we use: AAPL, AMZN,GOOG,INTC and MSFT.x axis represents the price level:positive axis is ask side and vice versa. y axis shows the average numbers og shares (normalized by mean)

CHAPTER 4

BASIC MACHINE LEARNING SCHEMES

In our research, we use ensemble methods to predict future price change of limit order books. As comparisons, some basic machine learning tools,such as logistic regression, lasso and ridge regression and support vector machine, are used. We call them "basic", not because they are simple, but because they have relatively long history and have already been widely used before. In the following, we give a brief introduction of these models.

4.1 Logistic regression

Logistic regression is a statistical tool which is used to find the fitting model to characterize the relationship between the independent variables and the response variable. Unlike traditional linear regression, the response variable of logistic regression is logit transformation of the probability. More clearly, suppose that we have a binary dependent variable Y, and we try to define the condition probability $Pr(Y = 1|X = x)$, which denoted as $p(x)$. An unbounded range transformation, which is called **logistic** (or **logit**) **transformation** $log\frac{p}{1-p}$, is introduced. The advantage of **logit transformation** is that it is unbounded in both positive and negative direction when $p(x)$ approaches to 1 and 0 respectively. Formally, the formula of logistic regression model is as follows:

$$log\frac{p(x)}{1-p(x)} = \beta_0 + x \cdot \beta \tag{4.1}$$

Solving the above formula for p will give us:

$$p(x; b, \omega) = \frac{e^{\beta_0 + x\cdot\beta}}{1 + e^{\beta_0 + x\cdot\beta}} = \frac{1}{1 + e^{-(\beta_0 + x\cdot\beta)}} \tag{4.2}$$

To solve the binary classification problem, we can assume $Y = 1$ when $p >= 0.5$ and $Y = 0$ when $p < 0.5$. Logistic regression is a widely used tool in applied statistics and discrete data analysis. There are some reasons for this:

- It is a traditional and classical model with a relatively long history, which can be track back to Bliss (1934). It has a wide range of applications in a lot of areas, so there are many related materials and cases that one can refer to.

- It has "exponential family" distribution property, which states that the probability of a specific variable v can be described linearly by $exp\beta_0 + \sum_{j=1}^{m} f_j(v)\beta_j$. If we let the vector v as binary, 0 or 1, and the functions f_j become all identical, then we can obtain a logistic regression. Exponential families take a very important role in statistical and physical research area, so it is beneficial if we turn a problem to logistic regression problem.

- Contingency table plays a critical role in classification problem since if we take a contingency table with two columns(independent variable Y) and many rows(dependent variables x), the contingency table become the classification problem to some extent. Logistic regression gives a linear interpolation for the log-odds, so it is very helpful in the classification problem.

Because the logistic regression deal with the probabilities, rather than just classes, we can use likelihood to fit the model.Assume that we have independent variable Y(denoted by y_i for each data) and dependent variables x(denoted by x_i for each data), the probability is either p, if $y_i = 1$, or $1 - p$, if $y_i = 0$. The Likelihood is defined as follows:

$$L(\beta_0, \beta) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i} \tag{4.3}$$

Take log transformation on both sides, the log-likelihood becomes:

$$\begin{aligned} l(\beta_0, \beta) &= \sum_{i=1}^{n} y_i log p(x_i) + (1 - y_i) log(1 - p(x_i)) \\ &= \sum_{i=1}^{n} log(1 - p(x)) + \sum_{i=1}^{n} y_i log \frac{p(x)}{(1 - p(x))} \\ &= \sum_{i=1}^{n} log(1 - p(x_i)) + \sum_{i=1}^{n} y_i(\beta_0 + x \cdot \beta) \\ &= \sum_{i=1}^{n} -log(1 + \exp^{\beta_0 + x_i \cdot \beta}) + \sum_{i=1}^{n} y_i(\beta_0 + x_i \cdot \beta) \end{aligned} \tag{4.4}$$

To find maximum of the log-likelihood, we can take first derivative of the log-likelihood function with respect to coefficient and set it equal to zero. The result is in the following:

$$\begin{aligned}\frac{\partial l}{\partial \beta_j} &= -\sum_{i=1}^{n} \frac{1}{1+\exp^{(\beta_0+x_i\cdot\beta)}} \exp^{(\beta_0+x_i\beta)} x_{ij} + \sum_{i=1}^{n} y_i x_{ij} \\ &= \sum_{i=1}^{n} (y_i - p(x_i;\beta_0,\beta)) x_{ij}\end{aligned} \tag{4.5}$$

From equation 4.5, we can see that it is a transcendental equation which can not transform to a polynomial equation, so there is no analytical solution. However we can solve it approximately with some numerical tools such as newton method and so on.

4.2 Lasso regression and ridge regression

lasso and ridge regression add different regulation parameter to the ordinary least square regression, so before we describe lasso and ridge regression, we first give a short introduction of ordinary least square regression(OLS). Assume the response variable is Y and the dependent variable is x, the following model:

$$y_i = \beta_0 + x_{i1}\beta_1 + x_{i2}\beta_2 + ... + x_{ik}\beta_k + e_i \tag{4.6}$$

where β_i corresponds to the coefficient of x_i and e_i is the error term with E[e_i]=0
is called least square regression. To make the error term become zero, we take the expectation on both sides and let $E(y_i) = \hat{y}_i$, now equation 4.6 becomes:

$$\begin{aligned}E(y_i) = \hat{y}_i &= \beta_0 + x_{i1}\beta_1 + x_{i2}\beta_2 + ... + x_{ik}\beta_k \\ \hat{y} &= \boldsymbol{X\beta}\end{aligned} \tag{4.7}$$

So the error term $\boldsymbol{e = Y - \hat{Y}}$ and we can define a loss function for the least square regression as $f = \boldsymbol{e'e}$. Our goal is to mimnimized this loss function. To realize this ,we can first define the loss function according to β, x, and y.

$$\begin{aligned}f = \boldsymbol{e'e} &= \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\ &= (y_i - \hat{y}_i)'(y_i - \hat{y}_i) \\ &= \boldsymbol{(y - x\beta)'(y - x\beta)} \\ &= \boldsymbol{y'y - \beta'X'y - y'X\beta + \beta'X'X\beta} \\ &= \boldsymbol{y'y - 2y'X\beta + \beta'X'X\beta}\end{aligned} \tag{4.8}$$

Now let the first derivative of loss function to the coefficient β equal to zero, i.e. $\frac{\partial f}{\partial \beta} = 0$. Since $\boldsymbol{y'y}$ is constant with respect to β, so $\frac{\boldsymbol{\partial y'y}}{\boldsymbol{\partial \beta}}$ is a null vector with the length equal to the number of data sample. The derivative of the second tern is:

$$\frac{\boldsymbol{\partial(-2y'X\beta)}}{\boldsymbol{\partial\beta}} = \boldsymbol{-2X'y} \tag{4.9}$$

For the third term:

$$\frac{\boldsymbol{\partial\beta' X'X\beta}}{\boldsymbol{\partial\beta}} = \boldsymbol{-2X'X\beta} \tag{4.10}$$

Now let the derivative of loss function equal to zero, we obtain that:

$$\frac{\boldsymbol{\partial f}}{\boldsymbol{\partial\beta}} = \boldsymbol{2X'X\beta - 2X'y} = 0 \tag{4.11}$$

So we have:

$$\begin{aligned} \frac{\boldsymbol{\partial f}}{\boldsymbol{\partial\beta}} = \boldsymbol{2X'X\beta - 2X'y} &= 0 \\ \boldsymbol{X'X\beta} &= \boldsymbol{X'y} \\ \boldsymbol{X'X\beta} &= \boldsymbol{X'y} \\ \boldsymbol{\hat{\beta}} &= \boldsymbol{(X'X)^{-1}X'y} \end{aligned} \tag{4.12}$$

The $\hat{\beta}$ is our fitted coefficients which make the loss function smallest. In fact the expectation of $\hat{\beta}$ will equal to β, we give the proof as follows:

$$\begin{aligned} \boldsymbol{E(\beta)} &= \boldsymbol{E[(X'X)^{-1}X'y]} \\ &= \boldsymbol{(X'X)^{-1}X'E(y)} \\ &= \boldsymbol{(X'X)^{-1}X'[X\beta]} \\ &= \boldsymbol{\beta} \end{aligned} \tag{4.13}$$

For the least square regression, if the rank $(\boldsymbol{X})$ is equal to number of independent variables, then the solution system only have one solution $\hat{\beta}$. But when the rank$(\boldsymbol{(}x))$ is greater than the number of independent variables, the system will have infinitely many solutions, which make the solutions lack of credibility. Furthermore, even rank$(\boldsymbol{X})$ equals to the number of independent variables, but when the sample size is close to the number of independent variables, least square regression can have a very poor prediction results.

How to overcome these problems? One way is to add some constraints to the least squares model, that is:

$$\min_{\beta \in \mathbb{R}^p} ||y - X\beta||_2^2 \ subject\ to\ \beta \in C \tag{4.14}$$

for some $C \in \mathbb{R}^p$, where p is the number of independent variables. Lasso regression and ridge regression are two famous examples that choose C to l_1 norm region and l_2 norm region. More clearly:

Ridge regression:

$$\min_{\beta \in \mathbb{R}^p} ||y - X\beta||_2^2 \ subject\ to\ ||\beta||_2 \leq t \tag{4.15}$$

Lasso regression:

$$\min_{\beta \in \mathbb{R}^p} ||y - X\beta||_2^2 \ subject\ to\ ||\beta||_1 \leq t \tag{4.16}$$

We can rewrite equation 4.15 and 4.16 with the KKT conditions and duality as:

Ridge regression:

$$\min_{\beta \in \mathbb{R}^p} ||y - X\beta||_2^2 \lambda ||\beta||_2 \tag{4.17}$$

Lasso regression:

$$\min_{\beta \in \mathbb{R}^p} ||y - X\beta||_2^2 + \lambda ||\beta||_1 \tag{4.18}$$

To our best knowledge, lasso regression was introduced by Tibshirani (1996) and Chen et al. (2001). Hoerl and Kennard (1970) was the first one to propose ridge regression. From figure 4.1 and figure 4.2, we can see that the constraint sets for both ridge and lasso are convex. The l_1 norm causes sparsity of the estimated coefficients, that is, there are $\beta_j = 0$ for many j in the set $1, ..., p$, when the tuning parameter become large. However, this is not the case in ridge regression, which will always generate non-zero estimators, when the tuning parameter changes.

From the KKT conditions(equation 4.18 and equation 4.17), it is feasible to find the regression estimators as a function of λ, which is the tuning parameter. This is called *solution path or regularization path* of the dual problem. Figure 4.3 and figure 4.4 show the regulation path of an example, we can see that lasso estimators will become zero when the tuning parameter rises. However, the ridge estimators will only approach to zero but never touch zero in most cases.

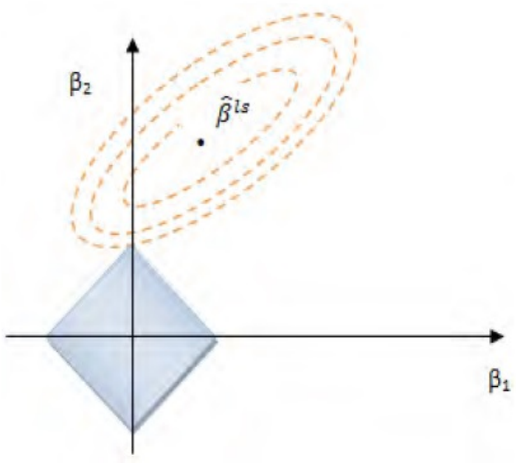


Figure 4.1: Illustration of Lasso constraints, redraw the chart in chapter 3 of Friedman et al. (2001)

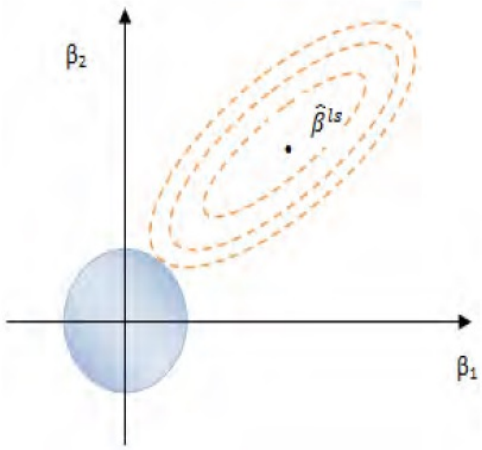


Figure 4.2: Illustration of Ridge constraints, redraw the chart in chapter 3 of Friedman et al. (2001)

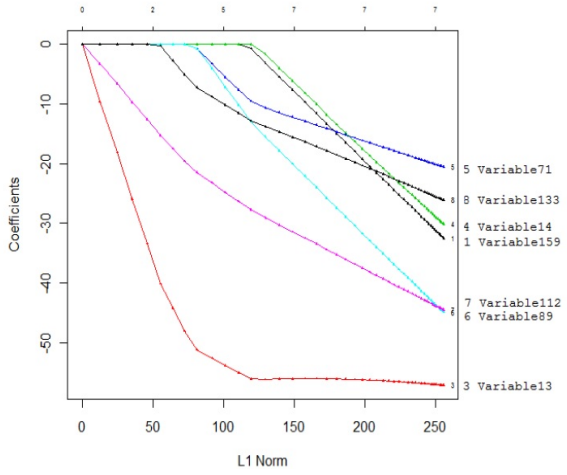


Figure 4.3: An example of the lasso regulation path with the increasing of the tuning parameter, each line represents the lasso solution , redraw the chart in chapter 3 ofFriedman et al. (2001)

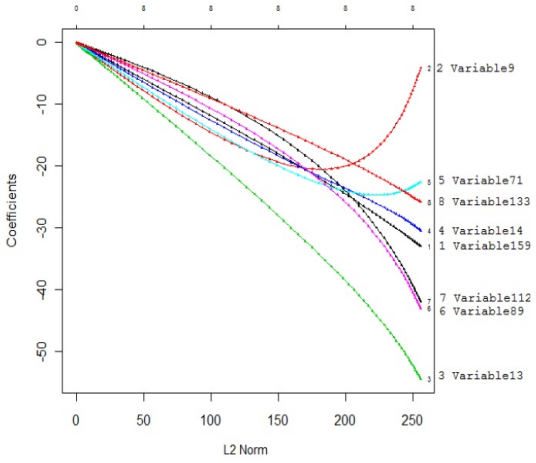


Figure 4.4: An example of the ridge regulation path with the increasing of the tuning parameter, each line represents the ridge solution , redraw the chart in chapter 3 ofFriedman et al. (2001)

4.3 Support vector machine

As far as we known, Boser et al. (1992) first introduced support vector machine(SVM) scheme, which is now widely used in bio-informatics, finance and other areas. Generally speaking, SVM is used to solve a linear two classes classification problem. Suppose there are input variables x and output y and we assume that y only have two values: 1 for positive examples and -1 for negative examples. The relationship between x and y is:

$$f(x) = \sum_{i=1}^{N} \omega^T x + b \tag{4.19}$$

Where ω represents the *weight vector*, and b is known as the bias. The *hyperplane* $\{x : f(x) = \omega^T x + b\}$ divide the space into two side: when $f(x) = \omega^T x + b > 0$, we treat the data in the positive side and vice versa. The boundary that split the space into positive and negative is called the *decision boundary*. Figure 4.5 shows an example of decision boundary of SVM. The black line divide the data into two sets, the red data points are marked as positive and the blue ones are marked as negative. How to find the best decision boundary is the key problem in SVM learning process. To explain this, we need first introduce the definition of margin. When the decision boundary is defined, we denote the closet positive(negative) points to the hyperplane as $x_+(x_-)$. $\hat{\omega}$, the unit vector of ω, is given by $\frac{\omega}{||\omega||}$ where $||\omega||$ is the norm of ω. The margin of a decision boundary f with respect to a dataset $\mathbb{D}$ can be defined as:

$$m_{\mathbb{D}}(f) = \frac{1}{2}\hat{\omega}^T(x_+ - x_-) \tag{4.20}$$

Also we assume that the distance of x_+ and x_- to the decision boundary f is equal which means:

$$f(x_+) = \omega^T x_+ + b = af(x_-) = \omega^T x_- + b \tag{4.21}$$

for some fixed number $a > 0$ and x_+,x_- can also called support vectors. To solve the margin, we set $a = 1$ in euation 4.21, add the two formula together and divide by $||\omega||$, we can get:

$$m_{\mathbb{D}}(f) = \frac{1}{2}\hat{\omega}^T(x_+ - x_-) = \frac{1}{||\omega||} \tag{4.22}$$

The goal of support vector machine scheme is to maximize the margin $\frac{1}{||w||}$, which is same as minimizing $||w||^2$. So the support vector machine problem can be transform to the following quadratic optimization problem:

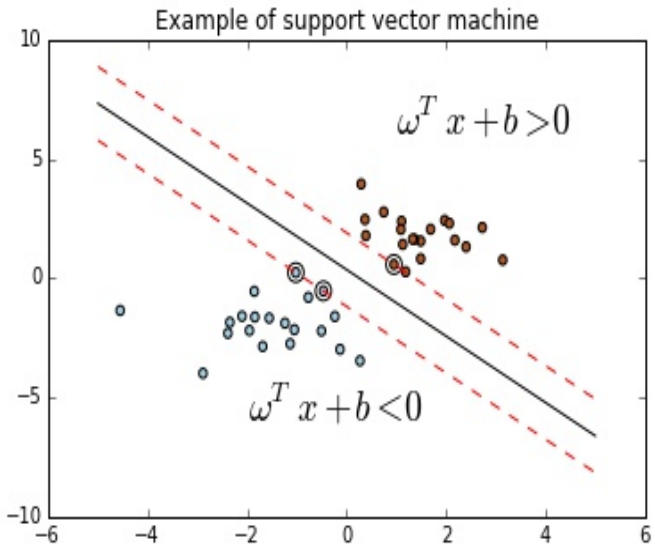


Figure 4.5: Example of a linear support vector machine classifier. The black line is the decision boundary and the circled data are the support vectors.

$$\begin{aligned} \underset{\omega,b}{\text{minimize}} \quad & \frac{1}{2}||\omega||^2 \\ \text{subject to} \quad & y_i(\omega^T x_i + b) >= 1,\ i = 1, \ldots, n. \end{aligned} \tag{4.23}$$

In the ideal case, the above equation can perfectly separate the data if they are linearly separable. However in practice, data is often not linearly separable, to allow errors we can replace the right side of equation 4.23 with the following:

$$y_i(w^T x_i + b) \geq 1 - \xi_i,\ i = 1, \ldots, n. \tag{4.24}$$

where $\xi_i \geq 0$ are called *slack variables*. Under this condition, some misclassifications are allowed if the example lies in the margin and we call this kind of error a margin error. To penalize the margin errors and misclassification, equation **??** can be transformed to:

$$\begin{aligned} \underset{\omega,b}{\text{minimize}} \quad & \frac{1}{2}||\omega||^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(\omega^T x_i + b) \geq 1 - \xi_i,\ i = 1, \ldots, n, \xi_i \geq 0 \end{aligned} \tag{4.25}$$

This formula is called *soft margin* SVM, which was proposed by Cortes and Vapnik (1995). For solving this formula, we can use a lagrange multipliers method to obtain the dual form which utilize the inner product.

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j x_i^T x_j \\ \text{subject to} \quad & \sum_{i=1}^{n} y_i \alpha_i = 0,\ 0 \leq \alpha_i \leq C \end{aligned} \tag{4.26}$$

Here we use the formula $\omega = \sum_{i=1}^{n} y_i \alpha_i x_i$.

As we mentioned before, in practice, the data is often non-linearly separable. To handle this problem, we often map our data from the input space $\mathcal{X}$ to a feature space $\mathcal{F}$. In the space {, the data may be close to linear separable and the discriminant function in this space is:$f(x) = \omega^T \Phi(x) + b$. Suppose the weight vector can be represented by a linear combination of the training samples,i.e. $\omega = \sum_{i=1}^{n} \alpha_i x_i$ then:

$$f(x) = \sum_{i=1}^{n} \alpha_i x_i^T x + b \tag{4.27}$$

The above formula becomes the following form in the space {

$$f(x) = \sum_{i=1}^{n} \alpha_i \Phi(x_i)^T \Phi(x) + b \tag{4.28}$$

If we define a kernel function $k(x, x')$ as

$$k(x, x') = \Phi(x)^T \Phi(x') \tag{4.29}$$

then the equation 4.28 can be written in terms of the kernel function:

$$f(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i) + b \tag{4.30}$$

The following is an example: let $\Phi(x) = (x_1^2, \sqrt{2} x_1 x_2, x_2^2)^T$, if we use the kernel function: $k(x, x') = (x^T x)^2$,then:

$$\begin{aligned} \Phi(x)^T \Phi(z) &= (x_1^2, \sqrt{2} x_1 x_2, x_2^2)^T (z_1, \sqrt{2} z_1 z_2, z_2^2) \\ &= x_1^2 z_1^2 + 2 x_1 x_2 z_1 z_2 + x_2^2 z_2^2 \\ &= (x^T z)^2 \end{aligned}$$

We can see that the mapping function Φ maps the data x in the low dimensional space to the data $\Phi(x)$ in the high dimensional space. The inner product of the high dimensional data is equal to the mapping results of the inner product of the low dimensional data under the kernel function.The following are some widely used types of kernel function.
Linear kernel: mapping function Φ is identity,i.e. no mapping

$$k(x, z) = x^T z \tag{4.31}$$

Polynomial kernel(with degree d):

$$k(x, z) = (x^T z)^d \; or \; (1 + x^T z)^d \tag{4.32}$$

Radial Basis function kernel:mapping the variable x to an infinite dimensional feature space.

$$k(x, z) = exp[-r||x - z||^2] \tag{4.33}$$

There has

4.4 Decision tree

Ensemble models combine some weaker classifiers to become a strong classifier. Decision tree is a commonly used weaker classifier. We also use decision tree result as the reference to our ensemble models. In the following, we give a brief introduction of decision tree learning scheme. Decision tree learning is a method to approximate discrete target function. The learned function can be treated as decision processes which apply the if-then rules. Decision trees classify examples by sorting them from the top root node to the bottom leaf node. Each node in the tree represents a test of one characteristic of the example and each branch from that node specifies one of the possible results for this characteristic

Figure 4.6 gives us a illustration of a typical decision tree learning process. Given an instance (Outlook =Sunny,Temperature =Hot, Humidity=High and Wind =Weak), it will go down to he leftmost branch of this decision tree and will be classified negative,which will not go to tennis on this day.

To improve the efficiency of the learning process, we would like to select the characteristic that is most helpful to classify the samples. We introduce a definition called "*entropy*" which measures

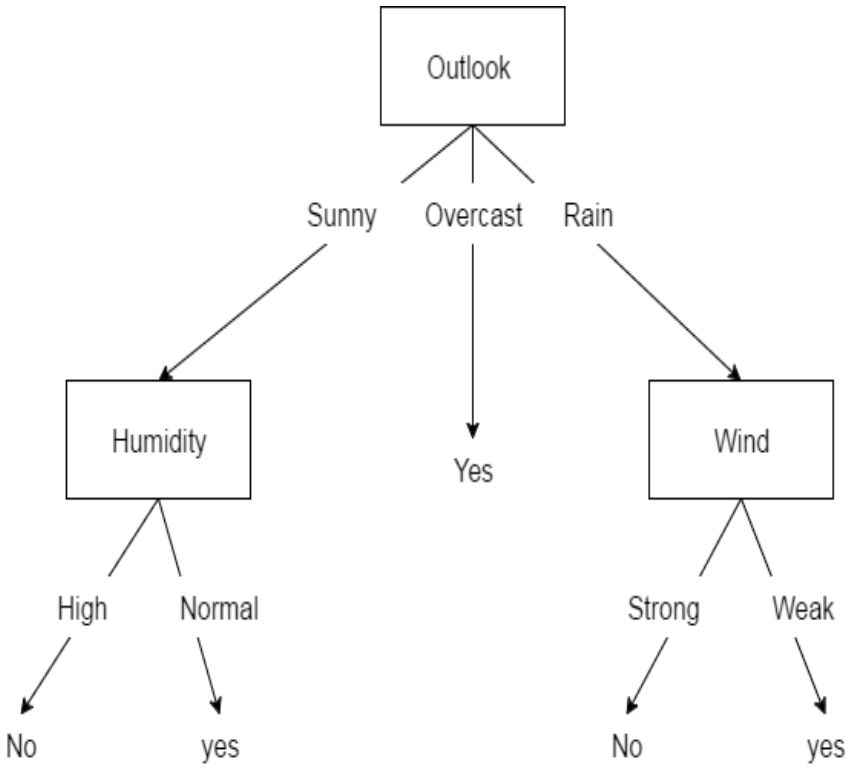


Figure 4.6: Example of decision tree, determine if a person will go out for playing table tennis or not.Nodes are attributes of the weather in a given day and the output in the last layer is yes or not.

the degree of impurity in a given sample set. Suppose there is a data sample S, including positive and negative instances, the entropy of S correspond to this binomial classification is :

$$Entropy(S) = -p_+ log_2 p_+ - p_- log2 p_- \tag{4.34}$$

where the p_+ is the ratio of positive instances in sample S and p_- is the ratio of negative instances in S.For example, is a data set S has 14 instances, in which 9 are positive and 5 are negative(we denote them as $[9+, 5-]$). Then the entropy of S is:

$$\begin{aligned} Entropy([9+, 5-]) &= -\frac{9}{14} log2 \frac{9}{14} - \frac{5}{14} log2 \frac{5}{14} \\ &= 0.940 \end{aligned} \tag{4.35}$$

More generally, if the target value takes more than 2 different classes,then the entropy of S can be written as:

$$Entropy(S) = \sum_{i=1} C - p_i log2 p_i \tag{4.36}$$

where p_i represents the ratio of the number of samples with class i to the whole sample size and C is the number of different classes. We now introduce another new definition that judge the effectiveness of a feature to classify the data sample and call it *information gain*. More formally, information gain regard to a sample data S is:

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)}^{\frac{|S_v|}{|S|}} Entropy(S_v) \tag{4.37}$$

where $Values(A)$ is the collection of all possible values for feature A and S_v is the subset of S where feature A has value $v(S_v = s \in S|A(s) = v)$ Now let see an example of information gain. We still assume a collection S including 14 instances [9+,5-] and the instances that have Wind=Weak contain 6 positive and 2 negative examples, the remainder have Wind = Strong. The information

gain of sorting these 14 samples by the feature Wind can be obtained as :

$$
\begin{aligned}
Values(Wind) &= Weak, Strong \\
S &= [9+, 5-] \\
S_{Weak} &= [6+, 2-] \\
S_{strong} &= [3+, 3-] \\
Gain(S, Wind) &= Entropy(S) - \sum_{v \in Values(A) \in \{Weak, Strong\}} \frac{S_v}{S} Entropy(S_v) \\
&= Entropy(S) - (\frac{8}{14}) Entropy(S_{Weak}) - (\frac{6}{14}) Entropy(S_s trong) \\
&= 0.940 - (\frac{8}{14}) 0.811 - \frac{6}{14} 1.00 \\
&= 0.048
\end{aligned}
\tag{4.38}
$$

CHAPTER 5

ENSEMBLE METHODS FOR PREDICTION

Adaboost and random forest learning algorithms are two machine learning models that we use to improve prediction performance of price change for limit order books. These models are two widely used ensemble methods. In this chapter, we give a brief description of ensemble methods,especially Adaboost and random forest algorithms.

5.1 An introduction of ensemble methods

Ensemble methods are designed for improving the performance of a statistical fitting model. It combines the results of some fitting models, if those models are relatively independent, the combined result will decrease the predicting errors. To be formally, let's consider a real-valued function

$$g : \mathbb{R}^d \to \mathbb{R} \tag{5.1}$$

that is based on data sample $(X_1, Y_1), ..., (X_n, Y_n)$, where X is in a d-dimensional real value space and Y is the response variable based on X. We define a projecting process called "*base procedure*" which will generate a real-value function based on some d-dimensional real input data. To use ensemble methods, we usually do the base procedure several times by changing the input data sample. For example , we can give different weights to dependent variables in each procedure to get different predicting results,$g_1(\hat{\omega_1^T} X), g_2(\hat{\omega_2^T} X), g_3(\hat{\omega_3^T} X), ..., g_n(\hat{\omega_n^T} X)$, where ω_i is the weights in the model in procedure i and n is the number of times that we run the model. Therefore, the results of the ensemble methods $\hat{g}_{ens}X)$ is the linearly combinations of the generated individual function $g_k(\hat{\omega_k^T} X)$

$$\hat{g}_e ns(X) = \sum_{k=1}^{n} c_k \hat{g}_k(\omega_k^T X) \tag{5.2}$$

where C_k is the coefficients of linear combination of the individual function $\hat{g}_k$.The c_k are usually identical, which equal to $\frac{1}{n}$ for linear model. For other models,such as boosting, $\sum_{k-1}^{n} c_k$ will increase when n gets larger.

Why the ensemble models will improve the predictive performance of an individual model? The reasons can be concluded in the following: 1) Re-sampling the data,such as bagging procedure,can be treated as a variance reduction method.Bühlmann (2012) 2) Boosting methods can reduce both bias and variance. Since boosting is a weighted average of some weaker classifiers, so it usually has lower variance. Besides, boosting method pays more attention to the sample of poor predictions, so it often reduces the bias too.

5.2 Bagging

Bagging is a very important ensemble method. Since we will both use bagging method during the process of building Adaboost and random forest scheme, so in this section we give a short introduction of this technique. Suppose we have pairs $(X_i, Y_i), i = 1, ..., n$, where X_i is a d-dimensional real value vector and Y_i is a real value defined as a response variable. If Y_i only takes discrete integers, then it is a classification problem, otherwise, it is a regression problem. Now define a function estimator, which generates result via a given base process

$$\hat{g}(\cdot) = h_n((X_1, Y_1), ..., (X_n, Y_n))(\cdot) : \mathcal{R}^d \to \mathcal{R} \tag{5.3}$$

where h_n is a function that estimates a function according to the given data. The algorithm of bagging is in the following:

Algorithm 5.1: Bagging algorithm,Breiman (1996)

1. Randomly draw n times with replacement from the data $(X_1, Y_1), ..., (X_n, Y_n)$ and define a boostrap sample data as $(X_1^*, Y_1^*), ..., (X_n^*, Y_n^*)$;
2. Derive the boostrapped estimator $\hat{g}^*(\cdot)$, $\hat{g}^*(\cdot) = h_n((X_1^*, Y_1^*), ..., (X_n^*, Y_n^*))(\cdot)$;
3. Repeat step 1 and steo 2 M times, where M is often chosen as 50 or 100, yielding $\hat{g}^{*k}(\cdot)(k = 1, ..., M)$. The bagged estimator is $\hat{g}_{Bag}(\cdot) = M^{-1} \sum_{k=1}^{M} \hat{g}^{*k}(\cdot)$

5.3 Adaboost learning scheme

This section we study the procedure and properties of an important learning scheme, Adaboost, which is deemed as the best off the shelf classifier in the word(Friedman et al. (2000)). The purpose of boosting method is to combine many weaker models together to generate a strong "committee". Freund and Schapire (1995) first introduced the Adaboost scheme.Suppose we have a response

variable Y, which can only take two values, 1 and -1. We also assume a vector of independent variables X and the predict result of a classifier $G(x)$ can also only take one of the two values $1, -1$. The error rate of the predict value $G(x)$ and the true value is:

$$e\bar{r}r = \frac{1}{N}\sum_{i=1}^{N} I(y_i \neq G(x_i)) \tag{5.4}$$

Now we assume that $G_m(x)$ is a sequence of weak classifiers, $m = 1, 2, ..., M$,the strong classifier $G(x)$ is obtained through a weighted majority vote of the results generated by the weak classifiers $G_m(x)$:

$$G(x) = sign(\sum_{m=1}^{M} \alpha_m G_m(x)) \tag{5.5}$$

where $\alpha_1, \alpha_2, ..., alpha_M$ are generated by the Adaboost algorithm. In each process, weights will be updated and those data that are misclassified will get more attention. Algorithm 5.2 shows the main procedure of Adaboost. At beginning, all the initial weights are set to be $\frac{1}{N}$, which means that they are all equal. For each successive step, the weights are individually changed based on the classification result on the last step. For example, At step k, those data that are misclassified by the classifier G_{k-1} will increase their weights,meanwhile those data that are correctly classified will decrease their weights.The factor that we update the weights is $exp(\alpha_m)$ in the algorithm, which is a exponential function of the error rate in the last step.

Algorithm 5.2: Adaboost algorithm, Friedman et al. (2001)

1 Initialize the observation weights ω_i=1/N,i=1,2,...,N;
2 **for** *m=1 to M* **do**
3 Fit a classifier $G_m(x)$ to the training data using weights ω_i;
4 Compute ;

$$err_m = \frac{\sum_{i=1}^{N} \omega_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} \omega_i}$$

5 Compute $\alpha_m = log((1 - err_m)/err_m)$;
6 Set $\omega_i \leftarrow \omega_i \cdot exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]$, $i = 1, 2, ..., N$;
7 Output $G(x) = sign[\sum_{m=1}^{M} \alpha_m G_m(x)]$;

In general, Adaboost is a kind of forward stag-wise additive model that uses the exponential loss function. Algorithm 5.3 shows the procedure of forward stag-wise additive model.

Algorithm 5.3: Forward Stag-wise Additive Modeling, Friedman et al. (2001)

1 Initialize $f_0(x) = 0$;
2 **for** *m=1 to M* **do**
  a) Compute ;
  $(\beta_m, \gamma_m) = \underset{\beta,\gamma}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma))$;
  b) Set $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$

The loss function of Adaboost is :

$$L(y, f(x)) = exp(-yf(x)) \tag{5.6}$$

Substitute into algorithm 5.3, we can get

$$(\beta_m, \gamma_m) = \underset{\beta,G}{\operatorname{argmin}} \sum_{i=1}^{N} exp(-y_i(f_{m-1}(x_i) + \beta G(x_i)) \tag{5.7}$$

This can be transformed to:

$$(\beta_m, \gamma_m) = \underset{\beta,G}{\operatorname{argmin}} \sum_{i=1}^{N} \omega_i^{(m)} exp(-\beta G(x_i)) \tag{5.8}$$

where $\omega_i^{(m)} = exp(-y_i f_{m-1}(x_i))$. To solve equation 5.8, first, assume $\beta > 0$, the solution of $G_m(x)$ in equation 5.8 is:

$$G_m = \underset{\beta,G}{\operatorname{argmin}} \sum_{i=1}^{N} \omega_i^{(m)} I(y_i \neq G(x_i)) \tag{5.9}$$

To prove this, from equation 5.8, we can easily get:

$$\sum_{i=1}^{N} \omega_i^{(m)} exp(-\beta y_i G(x_i)) = e^{-\beta} \sum_{y_i = G(x_i)} \omega_i^{(m)} + e^{\beta} \sum_{y_i \neq G(x_i)} \omega_i^{(m)} \tag{5.10}$$

which can be transformed into the following form if we add and minus term $e^{-\beta} \sum_{y_i \neq G(x_i)} \omega_i^{(m)}$ simultaneously.

$$(e^{\beta} - e^{-\beta}) \sum_{i=1}^{N} \omega_i^{(m)} I(y_i \neq G(x_i)) + e^{-\beta} \sum_{i=1}^{N} \omega_i^{(m)} \tag{5.11}$$

Now put G_m into equation 5.8 and solve parameter β, we can get:

$$\beta_m = \frac{1}{2} log \frac{1 - err_m}{err_m} \tag{5.12}$$

where err_m is the weighted error rate that is minimized:

$$err_m = \frac{\sum_{i=1}^{N} \omega_i^{(m)} I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} \omega_i^{(m)}} \tag{5.13}$$

Now the function $f_m(x)$ is updated:

$$f_m(x) = f_{m-1}(x) + \beta_m G_m(x) \tag{5.14}$$

and the weight in the next step is:

$$\omega^{(m+1)} = \omega_i^{(m)} \cdot e^{-\beta_m y_i G_m(x_i)} \tag{5.15}$$

We know that $-y_i G_m(x_i) = 2 \cdot I(y_i \neq G_m(x_i)) - 1$,so equation 5.15 can be written as :

$$\omega^{(m+1)} = \omega_i^{(m)} \cdot e^{\alpha_m I(y_i \neq G_m(x_i))} \cdot e^{-\beta_m} \tag{5.16}$$

where $\alpha_m = 2\beta_m$ and equation 5.16 is equivalent to algorithm 5.2 line 6, since the $e^{-\beta_m}$ do not change with i and multiply all the weights. So if we normalize the weights, this term will have no impact.

5.4 Random Forest

Breiman (2001) presents random forest method which connects bagging algorithm with random variable selection method. In his paper, he shows that random forest model is competitive compared with arcing and boosting techniques. Besides, random forest focus on variance reduction as well as decreasing bias.

Algorithm 5.4: Random forest, Breiman (2001)

1 **for** *b=1 to B* **do**
  (a) Draw a bootstrap sample Z^* of size N from the training data.
  (b) Grow a random-forest tree T_b to the bootstrapped data, by re- cursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
  i. Select m variables at random from the p variables.
  ii. Pick the best variable/split-point among the m.
  iii. Split the node into two daughter nodes.
2 Output the ensemble of trees $\{T_b\}_1^B$
To make a prediction at a new point x:
3 Let $\hat{C}_b(x)$ be the class prediction of the bth random forest tree. Then $\hat{C}_{rf}^B(x)$= majority vote $\{\hat{C}_b(x)\}_1^B$

5.4 shows the main procedure of random forest technique. In each step, it uses bootstrap method to generate a sample data set, so the generated trees in each step are relatively de-correlated. Therefore the average of results in each step tends to reduce the bias of the model. Since an individual tree model is well known for its noisy property, it is beneficial if we average different tree models. Build different identical distributed(i.d.) tress with low correlation can also reduce variance. We can see that if we average N i.i.d. random variables,each has a variance σ^2, then the variance of the average is $\frac{\sigma^2}{N}$, which will approach to zero when N becomes infinity. In the case that the variables are not i.i.d but i.d, the correlation between each pair is ρ, then the variance of the average is:

$$\rho\sigma^2 + \frac{1-\rho}{N}\sigma^2 \tag{5.17}$$

When N increases, $\frac{1-\rho}{N}\sigma^2$ will decrease to zero. So average will also be beneficial to variance reduction in the i.d. case. Moreover, if through reducing the pairwise correlation ρ, the first term $\rho\sigma^2$ can also been reduced. This can be achieved by random selecting of the input data. Besides the tree models are highly non-linear, so the bagging method during the process in the random forest can improve to reduce the correlation ρ.

When random forest method is used for solving classification problem, it will draw the final solution by a majority vote among the result from each individual tree model. In regression problem, the final result of random forest regression model is simply the mean of results from each tree. In addition, Breiman also make some recommendation about how to choose the size M to select the subset variables. For classification problem, the minimum size of node is one and the default node size is $[\sqrt{p}]$. For regression problem, the minimum size of node is five and the default node size is $[p/3]$, where the notation "[]" represents the rounding function.

CHAPTER 6

MODEL FRAMEWORK AND RESULTS

In this chapter, we focus on describing the framework of our models that apply ensemble method. First, let us review some aspects of limit order books.

6.1 Review of limit order books

According to Roşu (2009),the electronic limit order books prevail in the world's financial market. A lot of company use electronic trading to promote transactions. An order x with volume $\omega_x < 0(\omega_x > 0$,on the other hand) and price p_x is a duty by its owner to buy (sell,on the other hand) up to $|\omega_x|$ units of the security at a price no greater than (no less than, on the other hand) p_x. Orders can match the requirements of the other side when arrive are deemed as market order. Orders can not match the requirements of the other side when arrive are called limit order. The bid price b_t is the leading price among buy orders at time t. The ask price a_t is the cheapest price among the prices that all the active sellers are willing to take for. The difference between a_t and b_t is called bid-ask spread. The mean of a_t and b_t is named as mid price. For the limit order book, each time the ask price will be higher than the bid price, since they can not meet the execution requirements mutually. Figure 6.1 shows a snapshot of a 10-level limit order book. The green arrow shows the mid-price between best bid and best ask price. Besides all the ask prices are higher than all the bid prices. If the price of a new coming sell order (respectively, buy order) is lower(respectively, higher) than the best ask(respectively, bid) price, then the order will be executed, otherwise, it will come into the team of limit order books.

6.2 Data description

In this section, we gibe a detailed description of our dataset which is used to training the models. Our empirical calculations are based on a data set that describes the LOB dynamics for 5 highly liquid stocks traded on Nasdaq during the six-month period of 1 March 2015 to 1 September 2015. 3 The data that we study originates from the LOBSTER database Huang et al. (2015), which

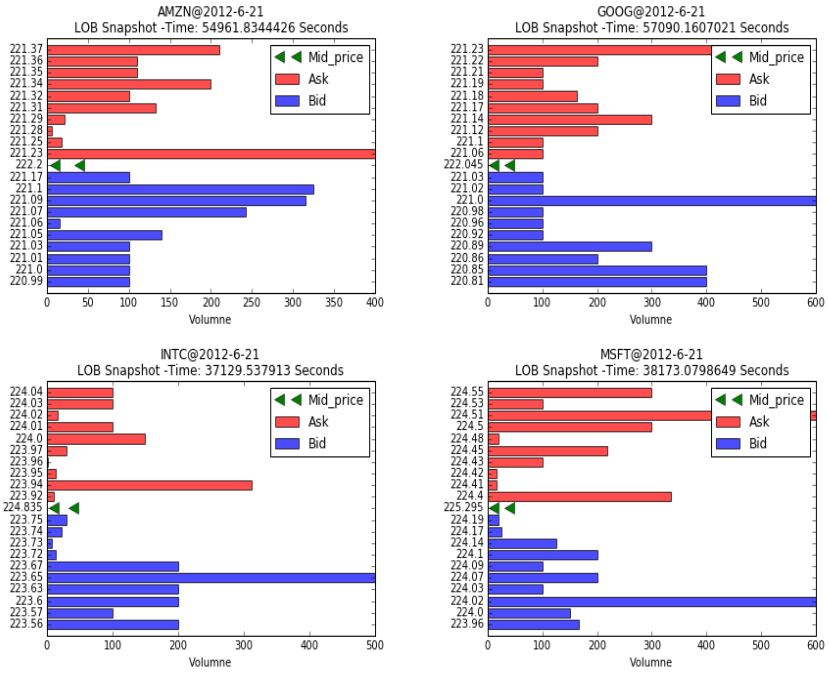


Figure 6.1: Limit order book examples, x axis denotes the volume of each order book and the y axis is the price level of each order book,the left top is AMZN, the right top is GOOG, the left bottom is INTC, and the right bottom is MSFT.Those are snapshots from 2012-06-21

Table 6.1: Limit order book statistics

	AAPL	AMZN	GOOG	INTC	MSFT
Total number of events at the best quotes	400391	269748	147916	624040	668765
Percentage of market order arrivals	1.7%	2.2%	2.4%	3.3%	1.9%
Percentage of limit order arrivals	52.9%	53.1%	52.7%	52.9%	51.7%
Percentage of limit order cancellations	45.4%	44.7%	44.9%	43.8%	46.4%

lists all market order arrivals, limit order arrivals, and cancellations that occur on the Nasdaq platform during 09:30 16:00 on each trading day. Trading does not occur on weekends or public holidays, so we exclude these days from our analysis. We also exclude market activity during the first and last 1000 seconds of each trading day, to remove any abnormal trading behavior that can occur shortly after the opening auction or shortly before the closing auction. On the Nasdaq platform, each stock is traded in a separate LOB with pricetime priority, with a tick size of = \$0.01. Although this tick size is the same for all stocks on the platform, the prices of different stocks vary across several orders of magnitude (from about \$1 to more than \$1000). Therefore, the relative tick size (i.e., the ratio between the stock price and) similarly varies considerably across different stocks. To choose the stocks in our sample, we first create a list of all stocks whose mid price remained below\$50.00 during the sample period of 1 March 2015 to 1 September 2015. We then order these stocks according to their total dollar trade value during this period, and select the first 5 stocks on this list. In descending order of their levels of market activity, these stocks are Apple (AAPL), Amazon (AMZN), Google (GOOG),Microsoft (MSFT), and Intel(INTC). Table 1 lists summary statistics describing trading activity for these 5 stocks during our sample period. Extracting information from the ITCH flow and without relying on third- party data providers, we analyze stocks from different industry sectors for ten full days of ultra-high-frequency intra-day data. The data provides information regarding trades against hidden orders. Coherently, the non-displayable hidden portions of the total volume of a so-called iceberg order is not accessible from data. Our ITCH feed data is day-specific and market-wide which means that we deal with one file per day with data over all the securities. Information (block A in Figure 1) regarding (i) messages for order submissions, (ii) trades and (iii) cancellations is included. For each order,

Table 6.2: Message book example, a sample on 2012-06-12

Timestamp	ID	Price	Quantity	Event	Side
1275386347944	6505727	126200	400	1	1
1275386347981	6505741	126500	300	2	1
1275386347981	6505741	126500	300	1	-1
1275386348070	6511439	126100	17	1	1
1275386348070	6511439	126100	17	3	1
1275386348101	6511469	126600	300	1	-1

Table 6.3: Message book event type, a sample on 2012-06-21

Type	Description
1	Submission of a new limit order
2	Cancellation(Partial deletion)
3	Deletion (Total deletion of a limit order)
4	Execution of a visible limit order
5	Execution of a hidden limit order

its type (buy/sell), price, quantity, and exact time stamp on millisecond ba- sis is available. In addition, (iv) administrative messages (i.e. trading halts or basic security data), (v) event controls (i.e. start and ending of trading days, states of market segments) and (vi) net order imbalance indicator are included.

6.2.1 Limit order and Message order

Message and limit order books are processed for each of the 10 days of the five stocks. More specifically, there are two types of messages particularly relevant for us: (i) add order messages, corresponding to order submissions and (ii) modify order messages, corresponding to updates on the status of existing orders through order cancellations and order executions. Example message and limit order books are illustrated in Tables 6.2 and 3, respectively.

In our data, timestamps are expressed in milliseconds since 1-Jan-1970 and shifted by three hours with respect to Eastern European Time (in the data the trading day goes from 7:00 to 15:25). ITHC feed prices are recorded with a 4 decimal digits precision and in our data decimal point is removed by multiplying the price by 10.000. Currency is in Euro for Helsinki exchange.

Table 6.4: message book direction, a sample on 2012-06-21

Direction	Description
-1	Sell limit order
1	Buy limit order

Table 6.5: limit book direction, a sample on 2012-06-12

Level 1				Level 2				...
Ask		Bid		Ask		Bid		...
Price	Quantity	Price	Quantity	Price	Quantity	Price	Quantity	
126300	300	126100	17	126400	4765	126000	2800	...
126300	300	126100	17	126400	4765	126000	2800	...
126300	300	126100	17	126400	4765	126000	2800	...
126111	291	126000	2800	126200	300	125900	1120	...
126100	291	126000	2800	126200	300	125900	1120	...
126100	291	126000	2800	126200	300	125900	1120	...

The tick size, defined as the smallest possible gap between the ask and bid prices, is one cent. Similarly, orders quantities are constrained to integers greater than one. An example of our LOB is the following:

6.2.2 Features

We follow an event based inflow, as has been used in [31]. This is due to that events (i.e. orders, executions and cancellations) do not follow a uniform inflow rate. Time intervals between two consecutive events can vary from milliseconds to several minutes of difference. Event based data representation avoids issues related to such big differences in data flow. As a result, each of our representations is a vector that contains information for 10 consecutive events. Event based data description leads to a dataset of approximately half a million representations (i.e. 394337 representations). We represent these events using the 144-dimensional representation proposed recently by Kercheval and Zhang [28], formed by three types of features: a) the raw data of a ten-level limit order containing price and volume values for bid and ask orders, b) features describing the state of the LOB exploiting past information and c) features describing the information edge in

the raw data by taking time into account. Derivation of time, stock price and volume are calculated for short and long term projection. Particularly, types in features u 7 , u 8 and u 9 are: trades, orders, cancellations, deletion, execution of a visible limit order and execution of a hidden limit order respectively. The features are listed in the following figure and can be referred in Kercheval and Zhang (2015)

Basic Set	Description(i = *level index*, $n = 10$)
$v_1 = \{P_i^{ask}, V_i^{ask}, P_i^{bid}, V_i^{bid}\}_{i=1}^{n}$,	*price and volume (n levels)*

Time-insensitive Set	Description(i = *level index*)
$v_2 = \{(P_i^{ask} - P_i^{bid}), (P_i^{ask} + P_i^{bid})/2\}_{i=1}^{n}$,	*bid-ask spreads and mid-prices*
$v_3 = \{P_n^{ask} - P_1^{ask}, P_1^{bid} - P_n^{bid}, \lvert P_{i+1}^{ask} - P_i^{ask}\rvert, \lvert P_{i+1}^{bid} - P_i^{bid}\rvert\}_{i=1}^{n}$,	*price differences*
$v_4 = \{\frac{1}{n}\sum_{i=1}^{n} P_i^{ask}, \frac{1}{n}\sum_{i=1}^{n} P_i^{bid}, \frac{1}{n}\sum_{i=1}^{n} V_i^{ask}, \frac{1}{n}\sum_{i=1}^{n} V_i^{bid}\}$,	*mean prices and volumes*
$v_5 = \{\sum_{i=1}^{n}(P_i^{ask} - P_i^{bid}), \sum_{i=1}^{n}(V_i^{ask} - V_i^{bid})\}$,	*accumulated differences*

Time-sensitive Set	Description(i = *level index*)
$v_6 = \{dP_i^{ask}/dt, dP_i^{bid}/dt, dV_i^{ask}/dt, dV_i^{bid}/dt\}_{i=1}^{n}$,	*price and volume derivatives*
$v_7 = \{\lambda_{\Delta t}^{la}, \lambda_{\Delta t}^{lb}, \lambda_{\Delta t}^{ma}, \lambda_{\Delta t}^{mb}, \lambda_{\Delta t}^{ca}, \lambda_{\Delta t}^{cb}\}$	*average intensity of each type*
$v_8 = \{\mathbf{1}_{\{\lambda_{\Delta t}^{la} > \lambda_{\Delta T}^{la}\}}, \mathbf{1}_{\{\lambda_{\Delta t}^{lb} > \lambda_{\Delta T}^{lb}\}}, \mathbf{1}_{\{\lambda_{\Delta t}^{ma} > \lambda_{\Delta T}^{ma}\}}, \mathbf{1}_{\{\lambda_{\Delta t}^{mb} > \lambda_{\Delta T}^{mb}\}}\}$,	*relative intensity indicators*
$v_9 = \{d\lambda^{ma}/dt, d\lambda^{lb}/dt, d\lambda^{mb}/dt, d\lambda^{la}/dt\}$,	*accelerations(market/limit)*

Figure 6.2: Features to build models

6.2.3 Order price,order book volume, and order book type

Here we study some properties of order price,order book volume, and order book type in a given day.

We can see from figure 6.3 to figure 6.7, both best ask price and best bid price are decreasing in the given day.

From figure6.8 we can see that for all the five stocks, among all the types, putting a new order book and cancel all the order book accounting for the vast majority of the proportion. In chapter 1,

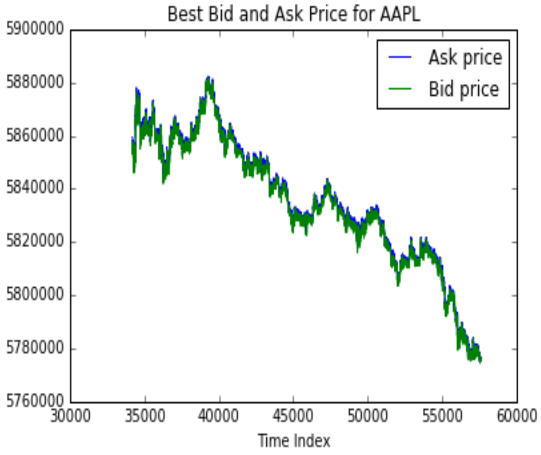


Figure 6.3: Best bid and ask price of AAPL in 2012-06-21

Figure 6.4: Best bid and ask price of AMZN in 2012-06-21

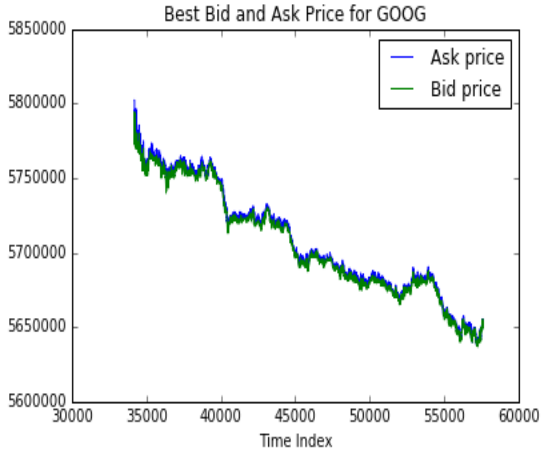


Figure 6.5: Best bid and ask price of GOOG in 2012-06-21

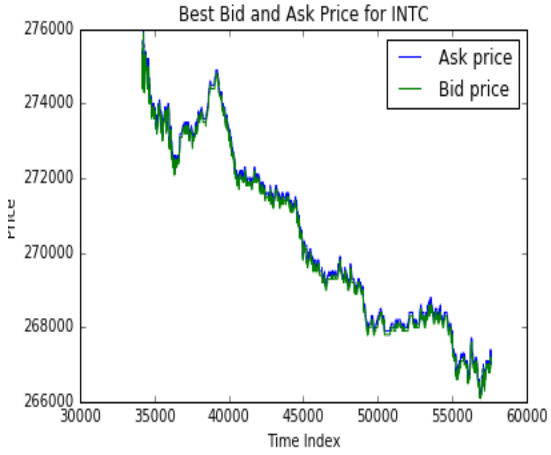


Figure 6.6: Best bid and ask price of INTC in 2012-06-21

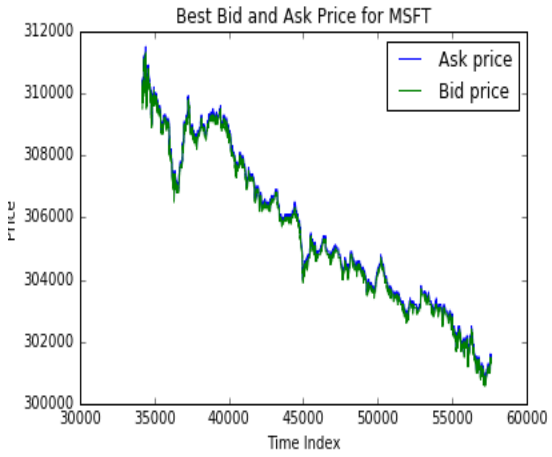


Figure 6.7: Best bid and ask price of MSFT in 2012-06-21

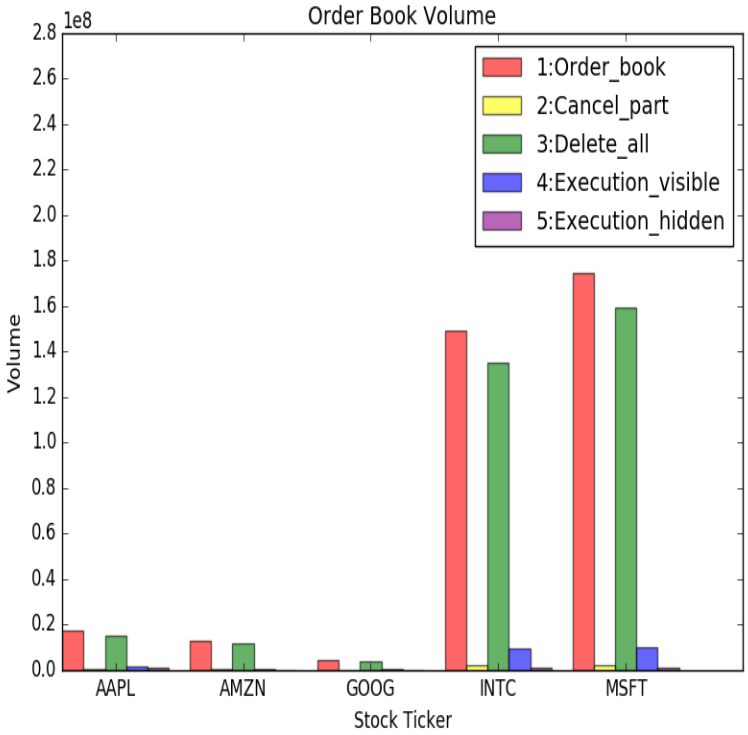


Figure 6.8: Order book volume comparison for five stocks in 2012-06-21

we have mentioned that a momentum strategy will put and cancel order book in a very short time to ignite the momentum from which the executor can beneficial from the spread by other investors.

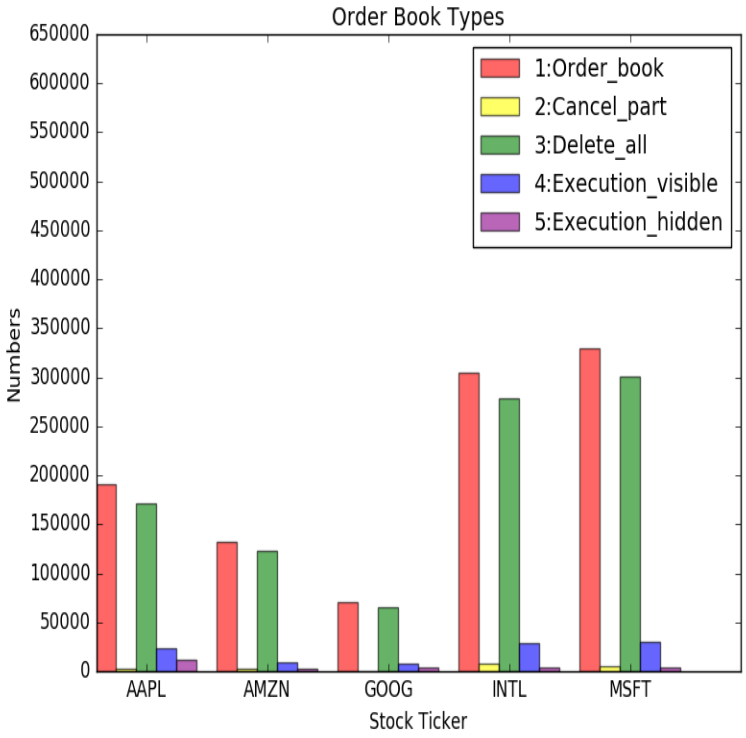


Figure 6.9: Order book type comparison for five stocks in 2012-06-21

Figure6.9 shows the same trend with the situation of order book volume. It is another example to show the arbitrage strategy of high frequency trading companies.

6.3 Model framework

Our model aims to predict the arbitrage opportunities of limit order book price change. there are two kinds of arbitrage in our case: ask price lower arbitrage and bid price higher arbitrage. We know that in any given time, the ask price will always higher than the bid price, so there is no arbitrage. However if the ask price at future Δt time is lower than the current price now, then there exists the arbitrage. How to realize it? The arbitrageur can sell short the stock at current bid price b_t and wait for Δt time, buy the stock from market with the price of ask price $a_{t+\Delta t}$, to

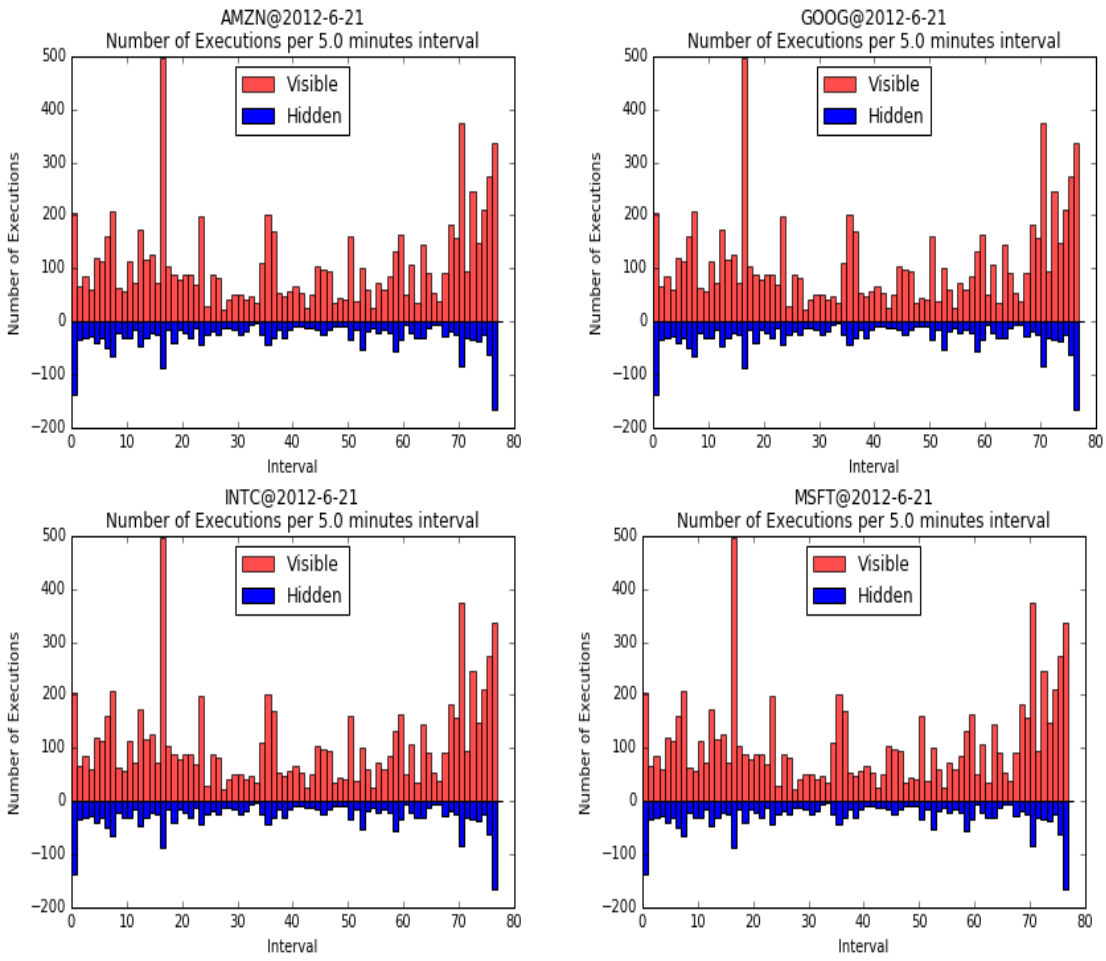


Figure 6.10: Number of trading in 2012-06-21,x axis is the time interval and y axis is the number of executions. Each band in time interval represents 5 minutes.The top left panel shows the result of AMZN, the top right panel shows the result of GOOG,the bottom left panel shows the result of INTC, and the bottom right panel shows the result of MSFT.

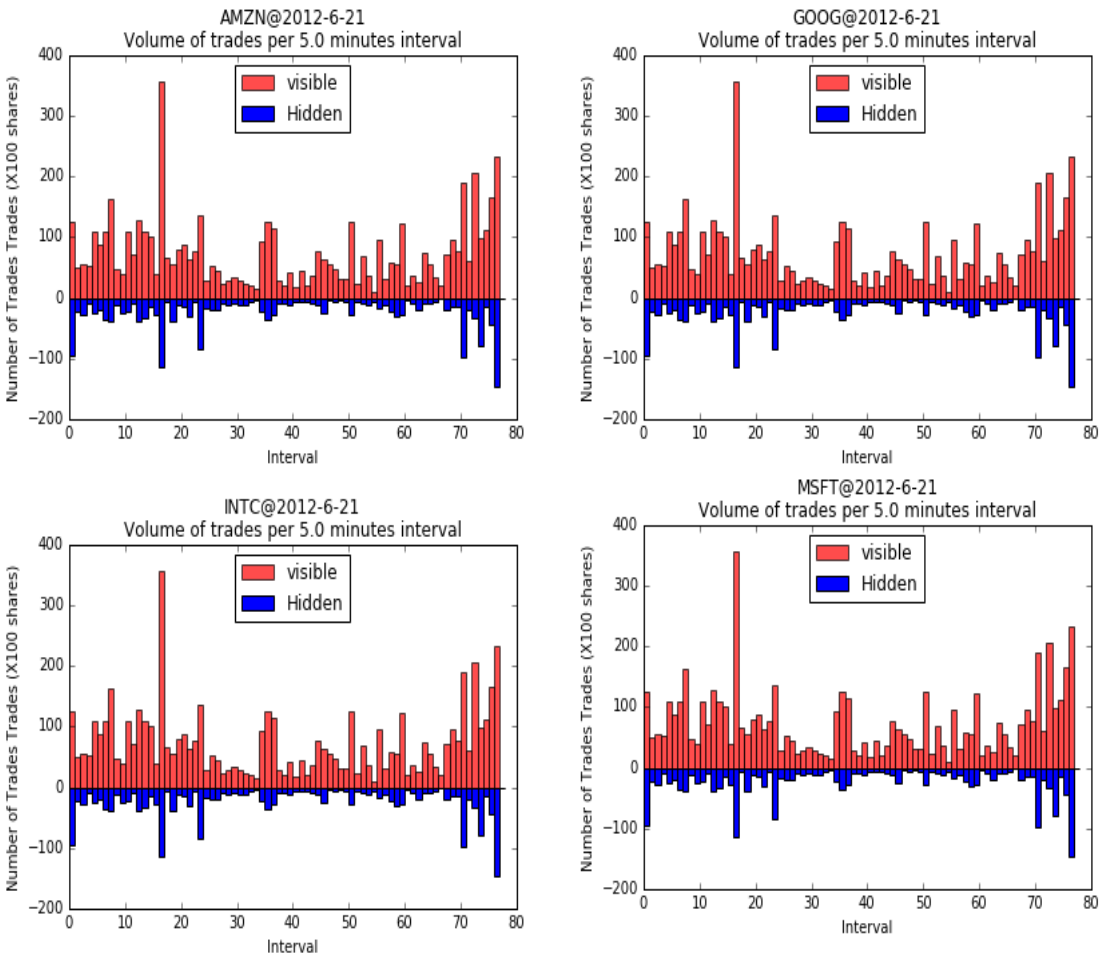


Figure 6.11: Volume of trading in 2012-06-21,x axis is the time interval and y axis is the number of executions. Each band in time interval represents 5 minutes.The top left panel shows the result of AMZN, the top right panel shows the result of GOOG,the bottom left panel shows the result of INTC, and the bottom right panel shows the result of MSFT.

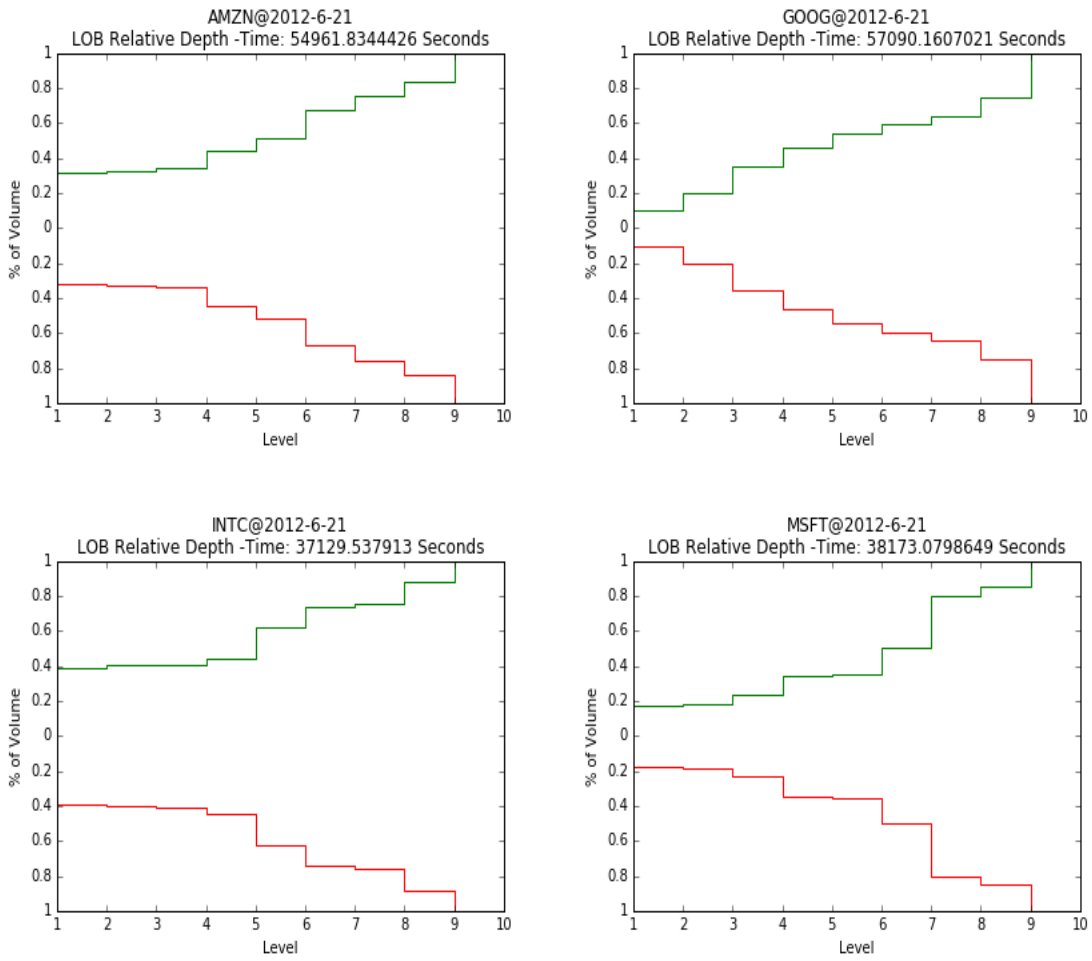


Figure 6.12: Relative depth in 2012-06-21,x axis is the price level and y axis is the percentage of total volume. The top left panel shows the result of AMZN, the top right panel shows the result of GOOG,the bottom left panel shows the result of INTC, and the bottom right panel shows the result of MSFT. .

return the stock required by the short positions, the spread he can earn without risk is $b_t - a_{t+\Delta t}$. On the bid high situation, the arbitrage strategy is similar. Figure 6.13,figure 6.14,and figure6.15 give us examples of ask low arbitrage, bid high arbitrage, and no arbitrage case respectively.

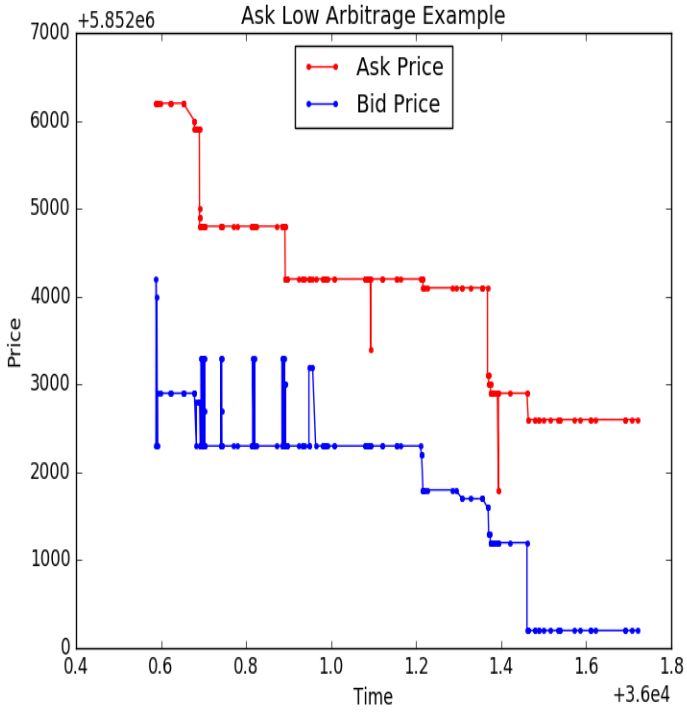


Figure 6.13: Ask low arbitrage example

Instead of predicting the price change of future events, like most past papers did, we focus on predict the price change in a fixed future time interval,e.g. 5 seconds later.

From figure 6.16, we can find that the arbitrage numbers for both bid-high and ask-low opportunities are increasing. For the five seconds time interval, the percentage of arbitrage opportunities among all the transaction is around 2% for each stock listed in the figure. For the ten seconds time interval, the arbitrage percentage increases to around 4%, which almost doubled.

Generally speaking, separate learning models should be built for different metrics depicting limit order book dynamics. To build a machine learning model for a specific metric, the following four-phase process is employed. Features representation: the data in the order book and message

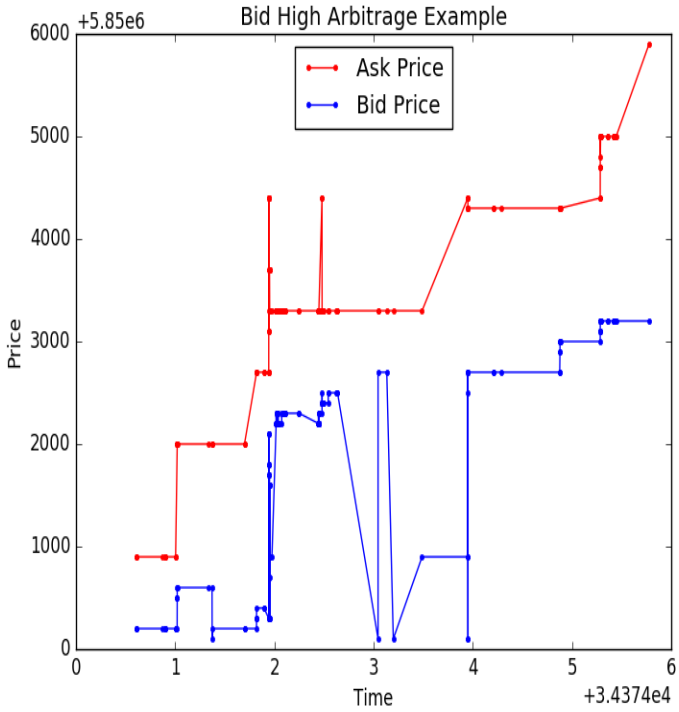


Figure 6.14: Bid high arbitrage example

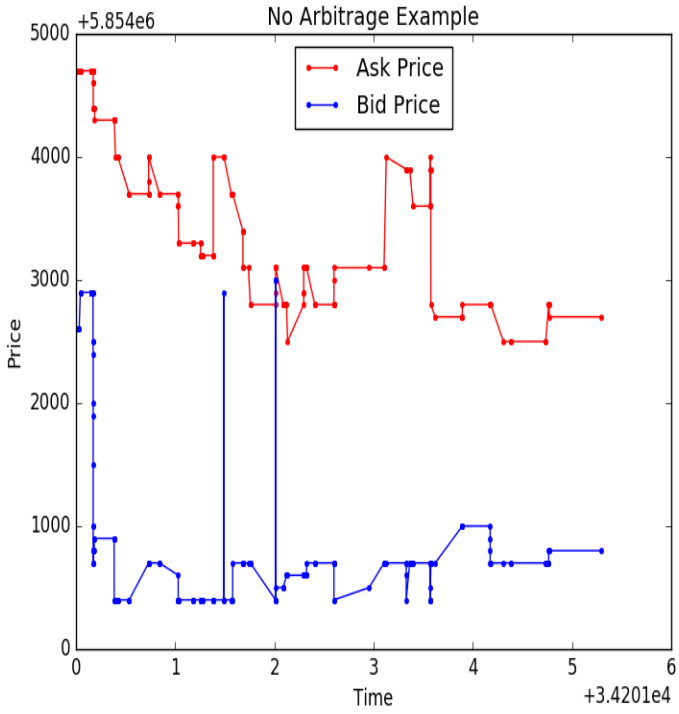


Figure 6.15: No arbitrage example

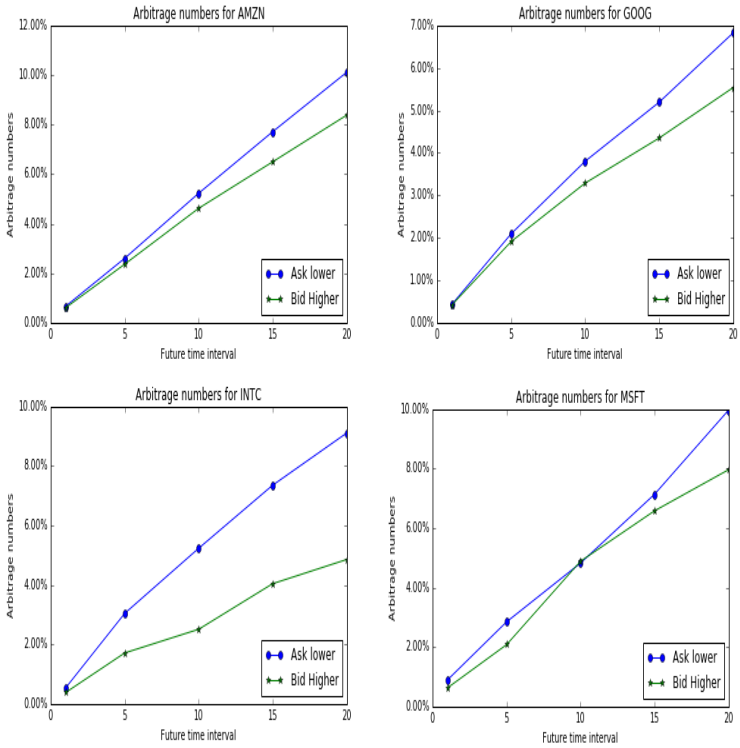


Figure 6.16: Arbitrage numbers based on future time interval

book is converted into a format suitable for machine learning methods to manipulate. Learning model construction: Models of Adaboost and random forest Learning model validation: the model is evaluated and validated using particular performance measurements. Unseen-data classification: the constructed learning model automates the forecasting of the chosen metric in real time.

6.4 Model measurement and numerical results

As we mentioned before, the arbitrage opportunities are rare events among all the placement of order books. For example, the percentages of arbitrage opportunities of five seconds interval is only around 2 %, so the accuracy rate of model is not a good measurement, because if we define all the results of testing sample as no arbitrage, we can still get a very high accuracy rate, which is around 98 %. Therefore we introduce precision, recall and f1 score to deal with this problem.
Some terms here:
Positive (P): Observation is positive, in our case, arbitrage opportunity will occur in the future.
Negative (N): Observation is negation, in our case means there is no arbitrage in the future.
True positive(TP): Observation is positive and is classified as positive, in our case means the detected arbitrage opportunity is real arbitrage opportunity.
False negative(FN):Observation is positive but is classified as negative,in our case means the detected arbitrage opportunity is not an arbitrage opportunity
True negative(TN):Observation is negative and is classified as negative,in our case means the detected no arbitrage case is actually no arbitrage
False positive(FP):Observation is positive but is classified as negative,in our case means the detected arbitrage case is actually no arbitrage.
The precision is:

$$Precision = \frac{TP}{TP + FP} = \frac{positive\ predicted\ correctly}{all\ positive\ predictions} \tag{6.1}$$

The recall is:

$$Recall = \frac{TP}{TP + FN} = \frac{predicted\ to\ be\ positive}{all\ positive\ observations} \tag{6.2}$$

F1 score:

$$F_1 = 2\frac{Precision * Recall}{Precision + Recall} \tag{6.3}$$

The following shows the results of predicting ask low opportunity of stock AMZN.

Table 6.6: AMZN ask-low arbitrage opportunity prediction(5 seconds)

Model	Training time(s)	Training F1 score	Test time(s)	Test Precision	Test Recall	Test F1 score
Logistic regression(Lasso penalty)	408.0	8.1 %	0.003	6.2%	61.5%	11.2%
Logistic regression(Ridge penalty)	12.1	8.0 %	0.02	6.1%	61.5%	11.2%
SVM(Poly 2 kernal,5000 estimator)	100.4	60.6 %	4.1	33.8%	100.0%	50.6%
Decision Tree(no pruning)	4.0	61.2 %	0.005	37.7%	94.2%	53.8%
Adaboost(number of estimate=100)	277.4	99.8 %	0.3	77.7 %	88.6	82.8%
Random forest(number of estimate=100)	49.0	95.6 %	0.16	73.1 %	99.0%	84.1%

Here the training samples are 90000 and test samples are 10000. Computer is 8G memory and Intel Xeon E3.

6.5 Feature importance

According tov

Table 6.7: Feature importance based on random forest model

AMZN		GOOG		INTC		MSFT	
Feature index	Importance	Feature index	Importance	Feature index	Importance	Feature index	Importance
107	3.997699	41	2.497344	16	4.930035	28	3.777894
108	2.531017	3	2.077711	36	3.584072	12	3.668482
116	1.72645	51	1.911522	40	3.271213	16	3.462065
12	1.523849	71	1.860316	14	2.930865	36	3.307528
41	1.489288	111	1.7199	30	2.830366	8	3.134871
52	1.401763	70	1.511696	20	2.694488	26	2.863927
61	1.390584	82	1.503223	22	2.594267	30	2.783838
51	1.296778	42	1.476138	38	2.575397	20	2.782014
1	1.26687	7	1.446197	28	2.565284	24	2.752804
81	1.24307	52	1.413531	32	2.510037	32	2.722132
3	1.240166	53	1.410323	18	2.485869	38	2.660868
9	1.222178	80	1.379994	24	2.388528	81	2.45301
79	1.187279	8	1.3286	82	2.289012	82	2.312318
80	1.168383	84	1.316409	12	2.165716	40	2.276727
53	1.16105	24	1.246773	26	2.119569	34	2.149401
54	1.150503	15	1.223152	4	2.032679	10	1.790342
82	1.150272	55	1.214248	8	2.007556	18	1.765228
118	1.133169	16	1.211959	34	1.950585	14	1.727172
7	1.131877	57	1.199527	81	1.896318	22	1.649081
84	1.099623	1	1.178082	6	1.697322	84	1.515854

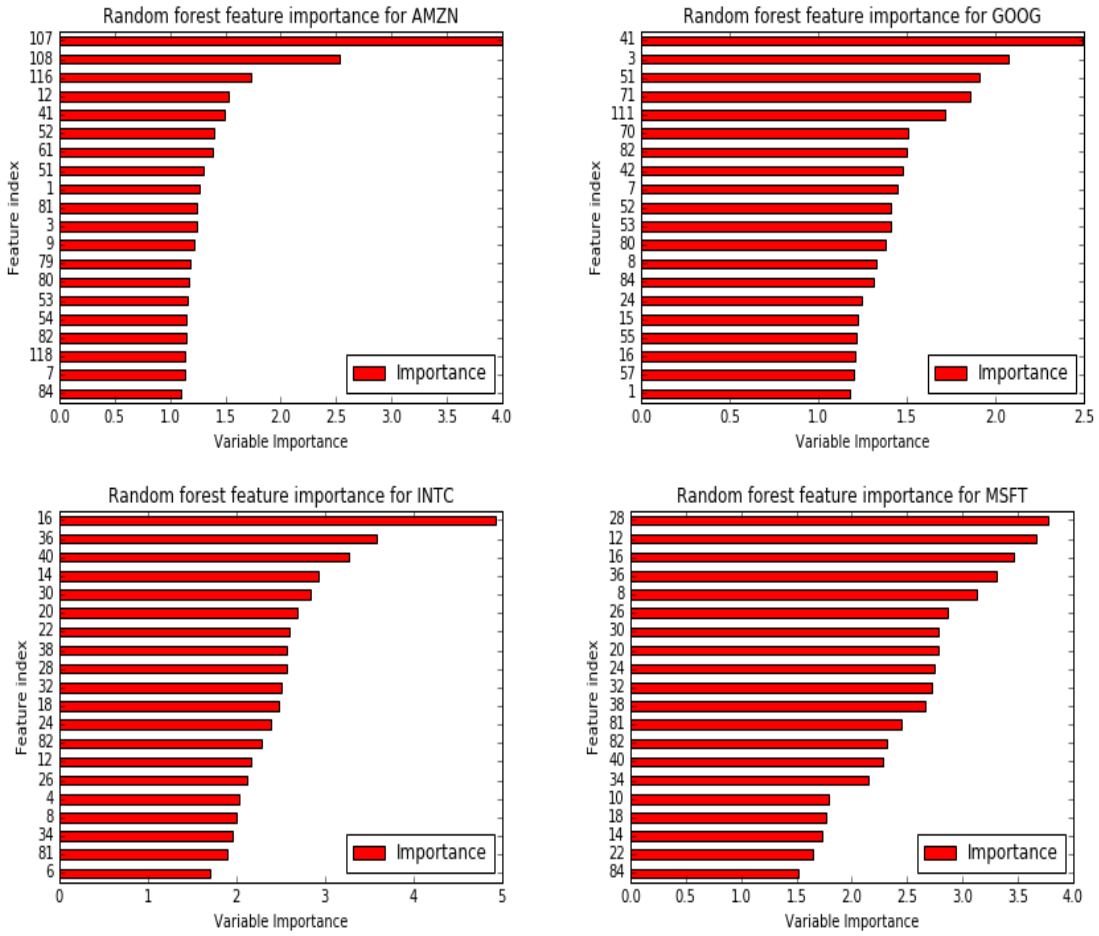


Figure 6.17: Feature importance. x axis is the feature importance and y axis is the corresponding feature index.The top left panel shows the result of AMZN, the top right panel shows the result of GOOG,the bottom left panel shows the result of INTC, and the bottom right panel shows the result of MSFT.

6.6 Trading strategy

According to Zhou et al. (2015),PnL is the profit and loss through transaction which can be written as follows:

$$PnL = \begin{cases} y - c & y >= \alpha, buy\ action \\ -y - c & y <= -\alpha, sell\ short\ action \\ 0 & otherwise \end{cases} \tag{6.4}$$

where y is the net capital gain from transaction, α is significant level and c is the trading cost.In the following we define a simple buy- low and sell high strategy.The details is in algorithm 6.1

Algorithm 6.1: Naive trading algorithm,Breiman (1996)

initialize: PnL=0
for *i =1 to length(test_set)* **do**
    input test_set[i] features into model and get result of Predict[i]
    if *Predict[i]==1(Ask low)* **then**
        Sell short at bid price
        Clear the short option Δt seconds later
        PnL+=$Bid\_price_t - Ask\_price_{t+\Delta t}$
    else if *Predicted[i]==-1(Bid high)* **then**
        Buy at ask price
        Sell at bid price Δt seconds later
        PnL+=$Bid\_price_{t+\Delta t} - Ask\_price_t$
    else
        Take no action
    return PnL

More vivid to show in the following figure 6.18

Figure 6.19 gives us an example on how to conduct an arbitrage trading, when the ask-low cases(green triangles) occur, we can sell short at the current bid price and buy the stock back at the future ask price. Similarly, when the bid-high cases (red triangles)occurs, we can buy the stock at the current ask price and sell the stock to the market at the future bid price.

Figure 6.20 shows the PnL for the four stocks. The red dot represents the bid-high cases and the blue dots represent the ask-low cases. If the dot lies above the x axis,it means that this transaction will earn money and vice versa.

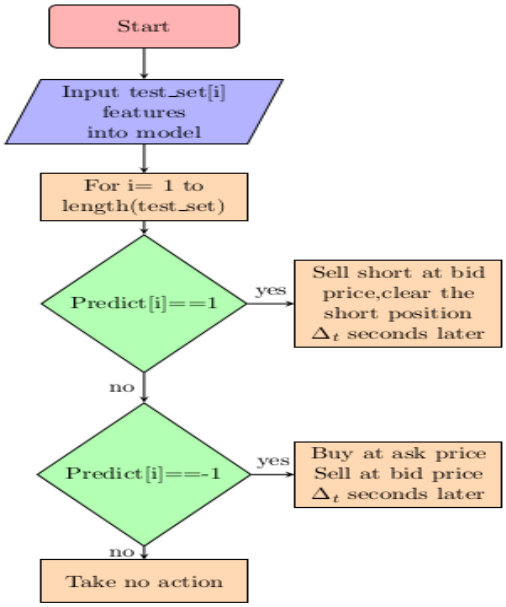


Figure 6.18: Naive trading strategy framework

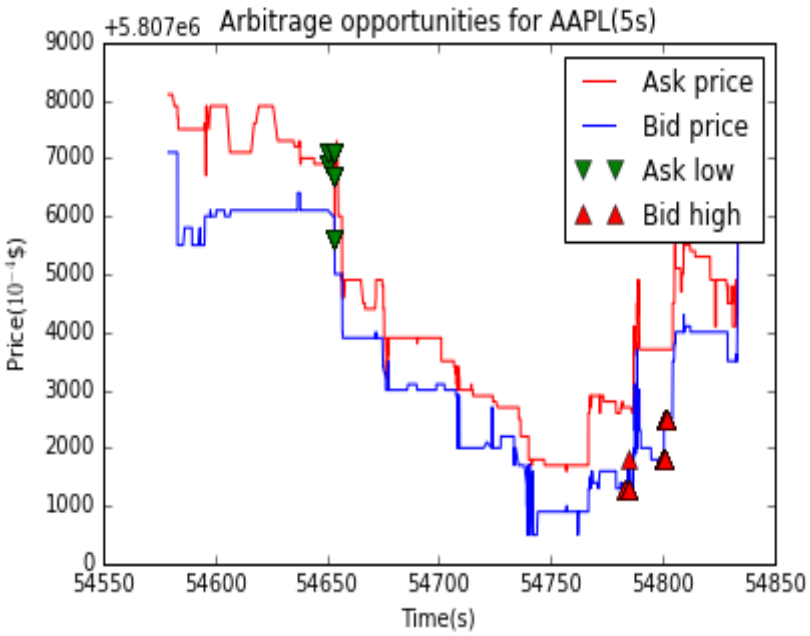


Figure 6.19: Arbitrage opportunities, x-axis is the time elapsed in second and the y axis is the stock price in US dollar.Blue line is bid price and red line is ask price, the triangles are the places where arbitrages occur, the green triangle is ask-low case and the red triangle is the bid high case.

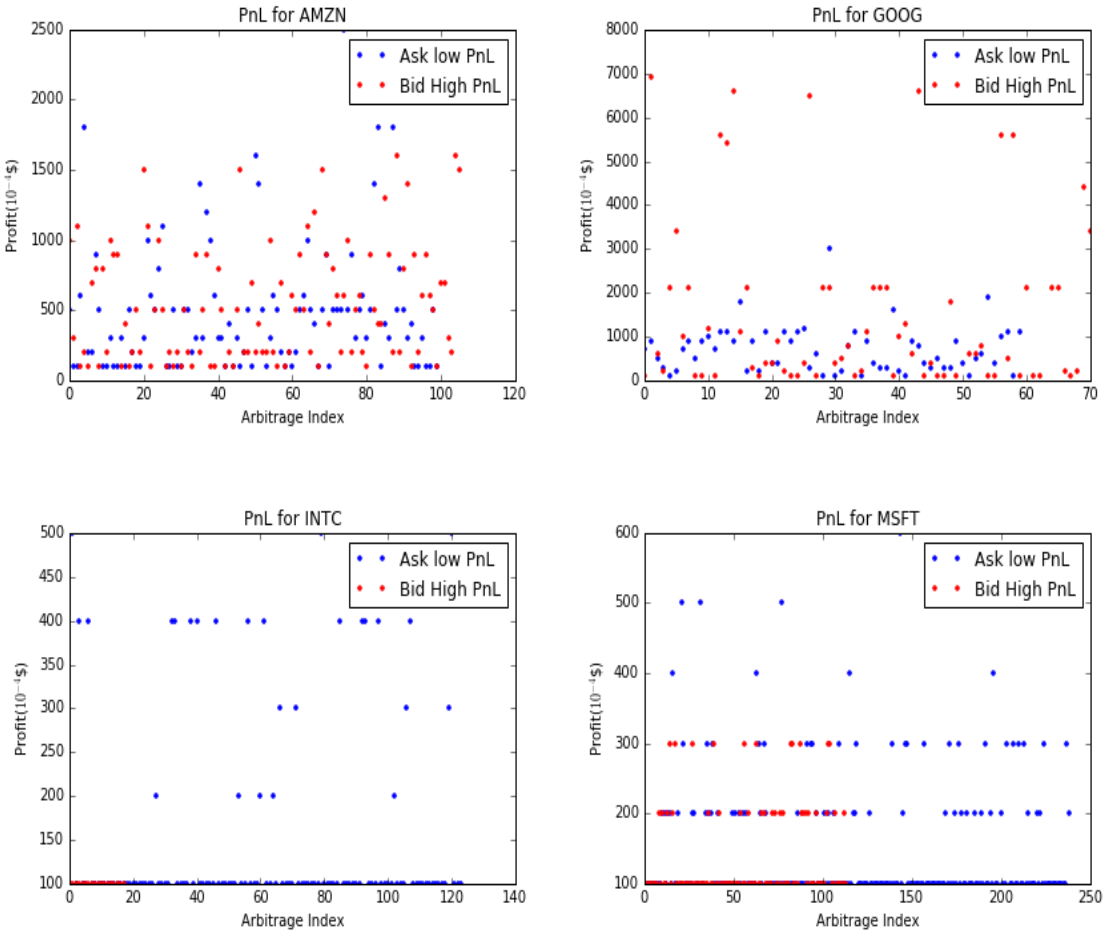


Figure 6.20: Profit and Loss, x axis represents the predicted arbitrage index and y axis is profit or loss for each transaction.The top left panel shows the result of AMZN, the top right panel shows the result of GOOG,the bottom left panel shows the result of INTC, and the bottom right panel shows the result of MSFT.

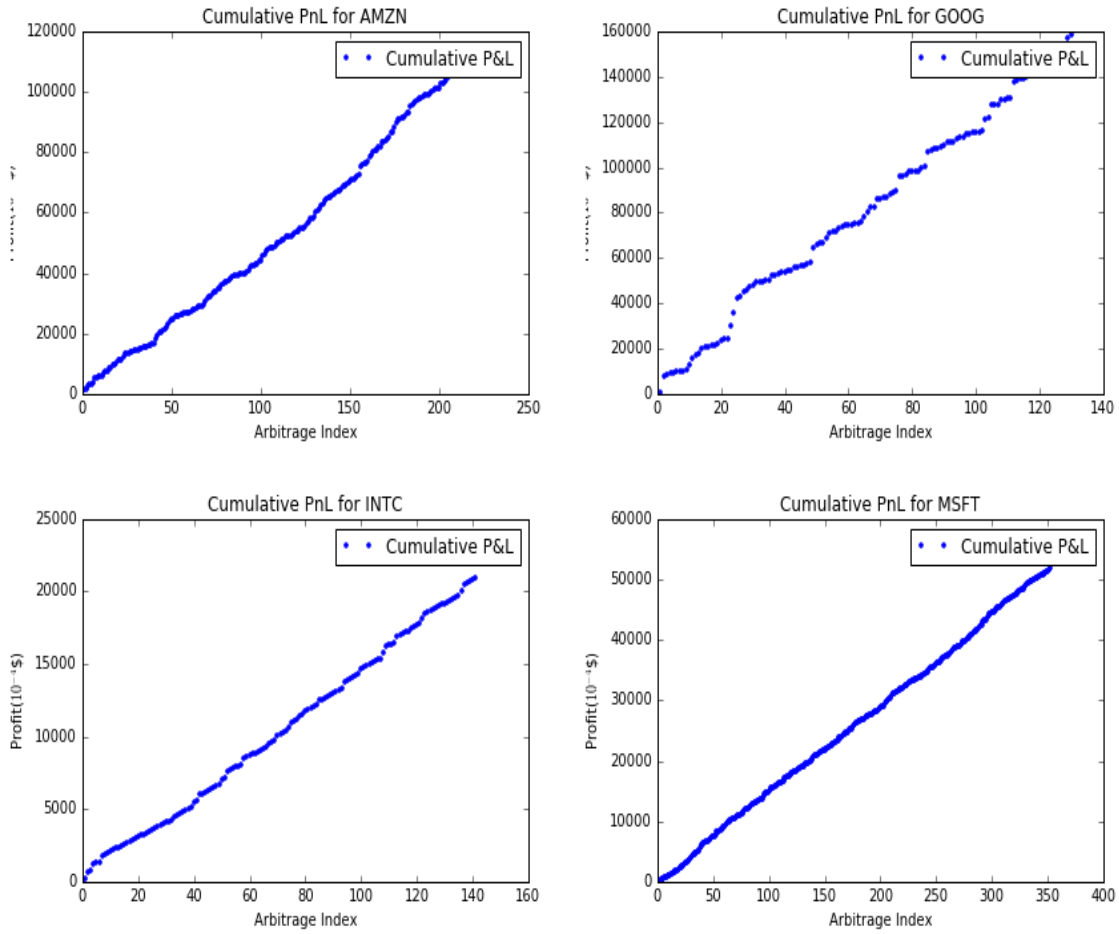


Figure 6.21: Cumulated Profit and Loss, x axis represents the predicted arbitrage index and y axis is profit or loss for each transaction.The top left panel shows the result of AMZN, the top right panel shows the result of GOOG,the bottom left panel shows the result of INTC, and the bottom right panel shows the result of MSFT.

APPENDIX A

TWO CLASSES ARBITRAGE PREDICTION RESULTS

In chapter 6.4, we list the result of ask-low arbitrage opportunity for stock AMZN based on Logistic regression(Lasso penalty), Logistic regression(Ridge penalty),support vector machine,decision tree, Adaboost, and Random forest. In this appendix, we show the results of the other four stocks, which are Apple(AAPL),Google(GOOG),Intel(INTC), and Microsoft(MSFT).

Table A.1: AAPL ask-low arbitrage opportunity prediction(5 seconds)

Model	Training time(s)	Training F1 score	Test time(s)	Test Precision	Test Recall	Test F1 score
Logistic regression(Lasso penalty)	422.9	8.7 %	0.002	5.3%	55.3%	9.7%
Logistic regression(Ridge penalty)	6.5	8.8 %	0.01	5.3%	55.3%	9.7%
SVM(Poly 2 kernal,5000 estimator)	110.3	67.1 %	9.0	40.0%	97.5%	56.7%
Decision Tree(no pruning)	4.3	50.0 %	0.003	30.8%	94.5%	46.4%
Adaboost(number of estimate=100)	390.5	99.8 %	0.21	72.7 %	90.4	80.6%
Random forest(number of estimate=100)	40.5	89.8 %	0.12	67.2 %	99.2%	80.12%

Table A.2: GOOG ask-low arbitrage opportunity prediction(5 seconds)

Model	Training time(s)	Training F1 score	Test time(s)	Test Precision	Test Recall	Test F1 score
Logistic regression(Lasso penalty)	345.3	4.6 %	0.002	0.9%	100.0%	1.7%
Logistic regression(Ridge penalty)	5.2	4.6 %	0.01	0.009%	100%	1.7%
SVM(Poly 2 kernal,5000 estimator)	57.8	35.3 %	3.0	6.1%	100.0%	11.6%
Decision Tree(no pruning)	4.2	53.1 %	0.003	9.6%	84.6%	17.3%
Adaboost(number of estimate=100)	355.1	99.9 %	0.20	74.6 %	100.0	85.4%
Random forest(number of estimate=100)	28.6	97.3 %	0.1	84.2 %	99.0%	91.0%

Table A.3: INTC ask-low arbitrage opportunity prediction(5 seconds)

Model	Training time(s)	Training F1 score	Test time(s)	Test Precision	Test Recall	Test F1 score
Logistic regression(Lasso penalty)	137.7	6.3 %	0.002	3.2%	1.4%	2.0%
Logistic regression(Ridge penalty)	12.1	62.8 %	0.01	3.2%	15.8%	2.1%
SVM(Poly 2 kernal,5000 estimator)	42.9	74.3 %	3.0	3.2%	10.0%	4.8%
Decision Tree(no pruning)	2.3	91.3 %	0.003	56.8%	69.6%	62.5%
Adaboost(number of estimate=100)	198.0	99.9 %	0.20	85.2 %	83.9	84.6%
Random forest(number of estimate=100)	12.5	99.9 %	0.1	65.6 %	80.4%	72.2%

Table A.4: MSFT ask-low arbitrage opportunity prediction(5 seconds)

Model	Training time(s)	Training F1 score	Test time(s)	Test Precision	Test Recall	Test F1 score
Logistic regression(Lasso penalty)	58.6	39.5 %	0.03	15.9%	50.0%	24.1%
Logistic regression(Ridge penalty)	10.2	39.5 %	0.002	15.9%	50%	24.1%
SVM(Poly 2 kernal,5000 estimator)	88.3	77.8 %	6.3	36.4%	94.1%	52.5%
Decision Tree(no pruning)	1.9	88.1 %	0.004	52.2%	15.9%	24.3%
Adaboost(number of estimate=100)	132.3	99.9 %	0.18	59.1 %	100.0	74.28%
Random forest(number of estimate=100)	16.4	99.9 %	0.1	59.1 %	100.0%	74.28%

APPENDIX B

MAIN PART OF CODE

Limit order book

author: Jian Wang

time: 2017-06-17

Model training and fitting

1.Model prepare

In [1]:

```
# -*- coding: utf-8 -*-
"""
Created on Fri Aug 26 00:03:47 2016

@author: jianwang
"""

import sys
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy as sp
from sklearn import linear_model
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import svm
from sklearn import tree
from sklearn import ensemble
import time
import matplotlib.pyplot as plt

#Set default parameters
ticker_list=["AAPL","AMZN","GOOG","INTC","MSFT"]
start_ind=10*3600
end_ind=15.5*3600
data_order_list=[]
data_mess_list=[]
time_index_list=[]
path_save='/media/jianwang/Study1/Research/order_book/'
path_load="/media/jianwang/Study1/Research/order_book/"

## set random seed to produce the same results
```

```
np.random.seed(987612345)

#read the stock ticker
#totally 5 dataset

for i in range(len(ticker_list)):
    #get the path for the csv files
    # name_order is for the order book and name_mess for the message book
    name_order='_2012-06-21_34200000_57600000_orderbook_10.csv'
    name_mess='_2012-06-21_34200000_57600000_message_10.csv'
    # calculate the cputime for reading the data
    t=time.time()
    # header =-1 means that the first line is not the header, otherwise, the first line will be header
    # data_order is for order book and data mess is for message book
    data_order_list.append(np.array(pd.read_csv(path_load+ticker_list[i]+name_order,header=-1),dtype="float64"))
    data_mess_list.append(np.array(pd.read_csv(path_load+ticker_list[i]+name_mess,header=-1),dtype="float64"))
    print("Time for importing the "+ticker_list[i]+" data is:",time.time()-t)
    print("The shape of the order data is: ",data_order_list[i].shape, " of message data is: ", data_mess_list[i].shape)
    # get the time index
    time_index_list.append(data_mess__list[i][:,0])


#print the sample of data
print("Check the original data:")

for i in range(len(ticker_list)):
    print()
    print("The first five sampe of "+ticker_list[i]+" is: ",data_order_list[i][:3])

    # -*- coding: utf-8 -*-

# # save the feature array
# ##get the original order,message and time index data, header =-1 means that did not
# ##read the first column as the name
#%%
# # use a loop to read data
# for ticker_ind in range(len(ticker_list)):
#     data_order=data_order_list[ticker_ind]
#     data_mess=data_mess_list[ticker_ind]
#     time_index=data_mess[:,0]
#     # obtain the reduced order message and time_index dataset, half an hour after the
#     # 9:30 and half an hour before 16:00
#     # data_reduced is used to install the data from 10 to 15:30, take half hour for auction
#     data_order_reduced=data_order[(time_index>= start_ind) & (time_index<= end_ind)]
#     data_mess_reduced=data_mess[(time_index>= start_ind) & (time_index<= end_ind)]
#     time_index_reduced=time_index[(time_index>= start_ind) & (time_index<= end_ind)]

#     test_lower=0
#     # test up is the up index of the original data to construct the test data
#     test_upper=len(data_order_reduced)
#     # data_test is the subset of data_reduced from the lower index to upper index
#     data_order_test=data_order_reduced[test_lower:test_upper,:]
#     data_mess_test=data_mess_reduced[test_lower:test_upper,:]
```

```
#     data_mess_test=data_mess_reduced[test_lower:test_upper,:]
#     t=time.time()
#     feature_array=get_features (data_order, data_mess,data_order_test,data_mess_test)
#     np.savetxt(path_save+ticker_list[ticker_ind]+'_feature_array.txt',feature_array,delimiter=' ')
#     print ("Time for building "+ticker_list[ticker_ind]+" is:",time.time()-t)


# load the feature
#%%
import time
t=time.time()
feature_array_list=[]
for ticker_ind in range(len(ticker_list)):
    feature_array_list.append(np.array(pd.read_csv(path_save+ticker_list[ticker_ind]+'_feature_array.txt',\
                                                   sep=' ',header=-1)))
print(time.time()-t)

# this function used to build the y
# ask_low as 1 bad high as -1 and no arbitrage as 0
# option=1 return ask low, option =2 return bid high, option =3 return no arbi, option =4 return total(ask_low=1,
# bid_high =-1 and no arbi =0)
#%%
def build_y(ask_low,bid_high,no_arbi,option):
    if (option==1):
        return ask_low
    elif option==2:
        return bid_high
    elif option==3:
        return no_arbi
    elif option==4:
        return ask_low-bid_high
    else:
        print("option should be 1,2,3,4")

## save y data
#%%
#time_ind=1
#option_ind=1
#for ticker_ind in range(len(ticker_list)):
#    response=build_y(ask_low_time_list[ticker_ind][time_ind],bid_high_time_list[ticker_ind][time_ind],\
#                                 no_arbi_time_list[ticker_ind][time_ind],option=option_ind)
#    np.savetxt(path_save+ticker_list[ticker_ind]+'_response.txt',response)



## load y data
#%%
response_list=[]
for ticker_ind in range(len(ticker_list)):
    response_list.append((np.array(pd.read_csv(path_save+ticker_list[ticker_ind]+'_response.txt',header=-1))))


## print the shape of the response
```

```
## note it is the total response
#%%
print("The shape of the total response is:\n")

for ticker_ind in range(len(ticker_list)):
    print(response_list[ticker_ind].shape)

# need to get the response from 10 to 15:30
# the shape of the response and the feature array should be equal
response_reduced_list=[]
for ticker_ind in range(len(ticker_list)):
    first_ind = np.where(time_index_list[ticker_ind]>=start_ind)[0][0]
    last_ind=np.where(time_index_list[ticker_ind]<=end_ind)[0][-1]
    response_reduced_list.append(response_list[ticker_ind][first_ind:last_ind+1])

print("The shape of the reduced response is:\n")

## print the shape of reduced response
## response reduced is used for testing and training the model
for ticker_ind in range(len(ticker_list)):
    print(response_reduced_list[ticker_ind].shape)
```

```
Time for importing the AAPL data is: 2.0496621131896973
The shape of the order data is:  (400391, 40)  of message data is:  (400391, 6)
Time for importing the AMZN data is: 1.4024591445922852
The shape of the order data is:  (269748, 40)  of message data is:  (269748, 6)
Time for importing the GOOG data is: 0.7670302391052246
The shape of the order data is:  (147916, 40)  of message data is:  (147916, 6)
Time for importing the INTC data is: 3.408174514770508
The shape of the order data is:  (624040, 40)  of message data is:  (624040, 6)
Time for importing the MSFT data is: 3.4592816829681396
The shape of the order data is:  (668765, 40)  of message data is:  (668765, 6)
Check the original data:

The first five sampe of AAPL is:  [[  5.85940000e+06   2.00000000e+02   5.85330000e+06   1.80000000e+01
    5.85980000e+06   2.00000000e+02   5.85300000e+06   1.50000000e+02
    5.86100000e+06   2.00000000e+02   5.85100000e+06   5.00000000e+00
    5.86890000e+06   3.00000000e+02   5.85010000e+06   8.90000000e+01
    5.86950000e+06   5.00000000e+01   5.84970000e+06   5.00000000e+00
    5.87000000e+06   1.00000000e+02   5.84930000e+06   3.00000000e+02
    5.87100000e+06   1.00000000e+01   5.84650000e+06   3.00000000e+02
    5.87390000e+06   1.00000000e+02   5.84530000e+06   3.00000000e+02
    5.87650000e+06   1.16000000e+03   5.84380000e+06   2.00000000e+02
    5.87900000e+06   5.00000000e+02   5.84270000e+06   3.00000000e+02]
 [  5.85940000e+06   2.00000000e+02   5.85330000e+06   1.80000000e+01
    5.85980000e+06   2.00000000e+02   5.85320000e+06   1.80000000e+01
    5.86100000e+06   2.00000000e+02   5.85300000e+06   1.50000000e+02
    5.86890000e+06   3.00000000e+02   5.85100000e+06   5.00000000e+00
    5.86950000e+06   5.00000000e+01   5.85010000e+06   8.90000000e+01
    5.87000000e+06   1.00000000e+02   5.84970000e+06   5.00000000e+00
    5.87100000e+06   1.00000000e+01   5.84930000e+06   3.00000000e+02
    5.87390000e+06   1.00000000e+02   5.84650000e+06   3.00000000e+02
    5.87650000e+06   1.16000000e+03   5.84530000e+06   3.00000000e+02
```

```
    5.87900000e+06   5.00000000e+02   5.84380000e+06   2.00000000e+02]
 [  5.85940000e+06   2.00000000e+02   5.85330000e+06   1.80000000e+01
    5.85980000e+06   2.00000000e+02   5.85320000e+06   1.80000000e+01
    5.86100000e+06   2.00000000e+02   5.85310000e+06   1.80000000e+01
    5.86890000e+06   3.00000000e+02   5.85300000e+06   1.50000000e+02
    5.86950000e+06   5.00000000e+01   5.85100000e+06   5.00000000e+00
    5.87000000e+06   1.00000000e+02   5.85010000e+06   8.90000000e+01
    5.87100000e+06   1.00000000e+01   5.84970000e+06   5.00000000e+00
    5.87390000e+06   1.00000000e+02   5.84930000e+06   3.00000000e+02
    5.87650000e+06   1.16000000e+03   5.84650000e+06   3.00000000e+02
    5.87900000e+06   5.00000000e+02   5.84530000e+06   3.00000000e+02]]

The first five sampe of AMZN is:  [[  2.23950000e+06   1.00000000e+02   2.23180000e+06   1.00000000e+02
    2.23990000e+06   1.00000000e+02   2.23070000e+06   2.00000000e+02
    2.24000000e+06   2.20000000e+02   2.23040000e+06   1.00000000e+02
    2.24250000e+06   1.00000000e+02   2.23000000e+06   1.00000000e+01
    2.24400000e+06   5.47000000e+02   2.22620000e+06   1.00000000e+02
    2.24540000e+06   1.00000000e+02   2.21300000e+06   4.00000000e+03
    2.24890000e+06   1.00000000e+02   2.20400000e+06   1.00000000e+02
    2.26770000e+06   1.00000000e+02   2.20250000e+06   5.00000000e+03
    2.29430000e+06   1.00000000e+02   2.20200000e+06   1.00000000e+02
    2.29800000e+06   1.00000000e+02   2.18970000e+06   1.00000000e+02]
 [  2.23950000e+06   1.00000000e+02   2.23810000e+06   2.10000000e+01
    2.23990000e+06   1.00000000e+02   2.23180000e+06   1.00000000e+02
    2.24000000e+06   2.20000000e+02   2.23070000e+06   2.00000000e+02
    2.24250000e+06   1.00000000e+02   2.23040000e+06   1.00000000e+02
    2.24400000e+06   5.47000000e+02   2.23000000e+06   1.00000000e+01
    2.24540000e+06   1.00000000e+02   2.22620000e+06   1.00000000e+02
    2.24890000e+06   1.00000000e+02   2.21300000e+06   4.00000000e+03
    2.26770000e+06   1.00000000e+02   2.20400000e+06   1.00000000e+02
    2.29430000e+06   1.00000000e+02   2.20250000e+06   5.00000000e+03
    2.29800000e+06   1.00000000e+02   2.20200000e+06   1.00000000e+02]
 [  2.23950000e+06   1.00000000e+02   2.23810000e+06   2.10000000e+01
    2.23960000e+06   2.00000000e+01   2.23180000e+06   1.00000000e+02
    2.23990000e+06   1.00000000e+02   2.23070000e+06   2.00000000e+02
    2.24000000e+06   2.20000000e+02   2.23040000e+06   1.00000000e+02
    2.24250000e+06   1.00000000e+02   2.23000000e+06   1.00000000e+01
    2.24400000e+06   5.47000000e+02   2.22620000e+06   1.00000000e+02
    2.24540000e+06   1.00000000e+02   2.21300000e+06   4.00000000e+03
    2.24890000e+06   1.00000000e+02   2.20400000e+06   1.00000000e+02
    2.26770000e+06   1.00000000e+02   2.20250000e+06   5.00000000e+03
    2.29430000e+06   1.00000000e+02   2.20200000e+06   1.00000000e+02]]

The first five sampe of GOOG is:  [[  5.80230000e+06   1.00000000e+02   5.79400000e+06   4.96000000e+02
    5.80430000e+06   1.00000000e+02   5.78700000e+06   4.00000000e+02
    5.80500000e+06   1.00000000e+02   5.78500000e+06   5.00000000e+02
    5.80630000e+06   1.00000000e+02   5.78000000e+06   5.00000000e+02
    5.80670000e+06   1.00000000e+02   5.77180000e+06   1.00000000e+02
    5.80960000e+06   5.00000000e+01   5.76940000e+06   1.00000000e+02
    5.80970000e+06   1.00000000e+02   5.76600000e+06   1.00000000e+02
    5.83500000e+06   1.00000000e+02   5.76260000e+06   1.00000000e+02
    5.88000000e+06   1.00000000e+02   5.73200000e+06   2.00000000e+01
    5.89260000e+06   1.00000000e+02   5.70000000e+06   1.00000000e+02]
```

```
 [  5.80230000e+06   1.00000000e+02   5.79400000e+06   1.96000000e+02
    5.80430000e+06   1.00000000e+02   5.78700000e+06   4.00000000e+02
    5.80500000e+06   1.00000000e+02   5.78500000e+06   5.00000000e+02
    5.80630000e+06   1.00000000e+02   5.78000000e+06   5.00000000e+02
    5.80670000e+06   1.00000000e+02   5.77180000e+06   1.00000000e+02
    5.80960000e+06   5.00000000e+01   5.76940000e+06   1.00000000e+02
    5.80970000e+06   1.00000000e+02   5.76600000e+06   1.00000000e+02
    5.83500000e+06   1.00000000e+02   5.76260000e+06   1.00000000e+02
    5.88000000e+06   1.00000000e+02   5.73200000e+06   2.00000000e+01
    5.89260000e+06   1.00000000e+02   5.70000000e+06   1.00000000e+02]
 [  5.80230000e+06   1.00000000e+02   5.79400000e+06   1.96000000e+02
    5.80430000e+06   1.00000000e+02   5.78700000e+06   4.00000000e+02
    5.80500000e+06   1.00000000e+02   5.78500000e+06   5.00000000e+02
    5.80630000e+06   1.00000000e+02   5.78000000e+06   5.00000000e+02
    5.80670000e+06   1.00000000e+02   5.77180000e+06   1.00000000e+02
    5.80960000e+06   5.00000000e+01   5.76940000e+06   1.00000000e+02
    5.80970000e+06   1.00000000e+02   5.76600000e+06   1.00000000e+02
    5.83500000e+06   1.00000000e+02   5.76260000e+06   1.00000000e+02
    5.88000000e+06   1.00000000e+02   5.73200000e+06   2.00000000e+01
    5.89260000e+06   1.00000000e+02   5.70000000e+06   1.00000000e+02]]

The first five sampe of INTC is:  [[  2.75200000e+05   6.60000000e+01   2.75100000e+05   4.00000000e+02
    2.75300000e+05   1.00000000e+03   2.75000000e+05   1.00000000e+02
    2.75400000e+05   3.73000000e+02   2.74900000e+05   2.00000000e+02
    2.75600000e+05   1.00000000e+02   2.74800000e+05   6.61000000e+02
    2.75700000e+05   1.00000000e+02   2.74700000e+05   3.00000000e+02
    2.75900000e+05   8.58900000e+03   2.74600000e+05   7.00000000e+02
    2.76000000e+05   9.59000000e+02   2.74500000e+05   9.00000000e+02
    2.76100000e+05   2.30000000e+03   2.74400000e+05   2.80000000e+03
    2.76200000e+05   2.70000000e+03   2.74300000e+05   3.30000000e+03
    2.76300000e+05   2.00000000e+03   2.74200000e+05   4.06300000e+03]
 [  2.75200000e+05   1.66000000e+02   2.75100000e+05   4.00000000e+02
    2.75300000e+05   1.00000000e+03   2.75000000e+05   1.00000000e+02
    2.75400000e+05   3.73000000e+02   2.74900000e+05   2.00000000e+02
    2.75600000e+05   1.00000000e+02   2.74800000e+05   6.61000000e+02
    2.75700000e+05   1.00000000e+02   2.74700000e+05   3.00000000e+02
    2.75900000e+05   8.58900000e+03   2.74600000e+05   7.00000000e+02
    2.76000000e+05   9.59000000e+02   2.74500000e+05   9.00000000e+02
    2.76100000e+05   2.30000000e+03   2.74400000e+05   2.80000000e+03
    2.76200000e+05   2.70000000e+03   2.74300000e+05   3.30000000e+03
    2.76300000e+05   2.00000000e+03   2.74200000e+05   4.06300000e+03]
 [  2.75200000e+05   1.66000000e+02   2.75100000e+05   4.00000000e+02
    2.75300000e+05   1.00000000e+03   2.75000000e+05   1.00000000e+02
    2.75400000e+05   3.73000000e+02   2.74900000e+05   2.00000000e+02
    2.75500000e+05   1.00000000e+02   2.74800000e+05   6.61000000e+02
    2.75600000e+05   1.00000000e+02   2.74700000e+05   3.00000000e+02
    2.75700000e+05   1.00000000e+02   2.74600000e+05   7.00000000e+02
    2.75900000e+05   8.58900000e+03   2.74500000e+05   9.00000000e+02
    2.76000000e+05   9.59000000e+02   2.74400000e+05   2.80000000e+03
    2.76100000e+05   2.30000000e+03   2.74300000e+05   3.30000000e+03
    2.76200000e+05   2.70000000e+03   2.74200000e+05   4.06300000e+03]]

The first five sampe of MSFT is:  [[  3.09900000e+05   3.78800000e+03   3.09500000e+05   3.00000000e+02
```

```
    3.10500000e+05   1.00000000e+02   3.09300000e+05   3.98600000e+03
    3.10600000e+05   1.00000000e+02   3.09200000e+05   1.00000000e+02
    3.10700000e+05   2.00000000e+02   3.09100000e+05   3.00000000e+02
    3.10800000e+05   2.00000000e+02   3.08900000e+05   1.00000000e+02
    3.10900000e+05   9.34800000e+03   3.08800000e+05   2.00000000e+02
    3.11000000e+05   1.80000000e+03   3.08700000e+05   2.00000000e+02
    3.11100000e+05   4.50000000e+03   3.08600000e+05   4.00000000e+02
    3.11300000e+05   1.00000000e+02   3.08500000e+05   4.00000000e+02
    3.11400000e+05   1.00000000e+02   3.08400000e+05   1.60000000e+03]
 [  3.09900000e+05   3.78800000e+03   3.09500000e+05   3.00000000e+02
    3.10500000e+05   2.00000000e+02   3.09300000e+05   3.98600000e+03
    3.10600000e+05   1.00000000e+02   3.09200000e+05   1.00000000e+02
    3.10700000e+05   2.00000000e+02   3.09100000e+05   3.00000000e+02
    3.10800000e+05   2.00000000e+02   3.08900000e+05   1.00000000e+02
    3.10900000e+05   9.34800000e+03   3.08800000e+05   2.00000000e+02
    3.11000000e+05   1.80000000e+03   3.08700000e+05   2.00000000e+02
    3.11100000e+05   4.50000000e+03   3.08600000e+05   4.00000000e+02
    3.11300000e+05   1.00000000e+02   3.08500000e+05   4.00000000e+02
    3.11400000e+05   1.00000000e+02   3.08400000e+05   1.60000000e+03]
 [  3.09900000e+05   3.78800000e+03   3.09500000e+05   3.00000000e+02
    3.10400000e+05   1.00000000e+02   3.09300000e+05   3.98600000e+03
    3.10500000e+05   2.00000000e+02   3.09200000e+05   1.00000000e+02
    3.10600000e+05   1.00000000e+02   3.09100000e+05   3.00000000e+02
    3.10700000e+05   2.00000000e+02   3.08900000e+05   1.00000000e+02
    3.10800000e+05   2.00000000e+02   3.08800000e+05   2.00000000e+02
    3.10900000e+05   9.34800000e+03   3.08700000e+05   2.00000000e+02
    3.11000000e+05   1.80000000e+03   3.08600000e+05   4.00000000e+02
    3.11100000e+05   4.50000000e+03   3.08500000e+05   4.00000000e+02
    3.11300000e+05   1.00000000e+02   3.08400000e+05   1.60000000e+03]]
80.32640743255615
The shape of the total response is:

(400236, 1)
(269571, 1)
(147766, 1)
(622641, 1)
(667701, 1)
The shape of the reduced response is:

(309538, 1)
(218710, 1)
(118877, 1)
(458160, 1)
(511299, 1)
```

2.train and test data split

In [3]:

```
# -*- coding: utf-8 -*-
# Random split
```

```
#%%-----------------------------------------------------------------
import random
from sklearn.cross_validation import train_test_split

ticker_ind=1
size=100000

# combine the feature and response array to random sample
total_array=np.concatenate((feature_array_list[ticker_ind],response_reduced_list[ticker_ind]),axis=1)[:size,:]



print("total array shape:",total_array.shape)

#split the data to train and test data set
train_x, test_x, train_y, test_y =train_test_split(\
total_array[:,:134],total_array[:,134], test_size=0.1, random_state=42)

# the y data need to reshape to size (n,) not (n,1)
test_y=test_y.reshape(len(test_y),)
train_y=train_y.reshape(len(train_y),)

print("test_y shape:",test_y.shape)
print("train_y shape:",train_y.shape)
```

```
total array shape: (100000, 135)
test_y shape: (10000,)
train_y shape: (90000,)
```

In [77]:

```
np.random.choice(100,3,replace=False
                 )
```

Out[77]:

```
array([35, 57, 46])
```

In [16]:

```
# random generate a given
def random_choice(num, key):
    temp=np.random.choice(num,size=key,replace=False)
    temp_sort=sorted(temp)
    for i in range(len(temp)):
        num[temp_sort[i]]=temp[i]

    return num
```

In [30]:

```
#time series split
#%%-----------------------------------------------------------------------------------------
```

```
ticker_ind=1
size=100000
random_ratio=0.5
# combine the feature and response array to random sample
total_array=np.concatenate((feature_array_list[ticker_ind],response_reduced_list[ticker_ind]),axis=1)[:size,:]

total_array=total_array[random_choice(list(range(size)),int(size*random_ratio)),:]


train_num_index=int(len(total_array)*0.9)

print("total array shape:",total_array.shape)

#split the data to train and test data set
train_x=total_array[:train_num_index,:134]
test_x=total_array[train_num_index:,:134]
train_y=total_array[:train_num_index,134]
test_y=total_array[train_num_index:,134]


# the y data need to reshape to size (n,) not (n,1)
test_y=test_y.reshape(len(test_y),)
train_y=train_y.reshape(len(train_y),)
print("train_x shape:",train_x.shape)
print("test_x shape:",test_x.shape)
print("test_y shape:",test_y.shape)
print("train_y shape:",train_y.shape)
# scale data
#%%

# can use the processing.scale function to scale the data
from sklearn import preprocessing
# note that we need to transfer the data type to float
# remark: should use data_test=data_test.astype('float'),very important !!!!
# use scale for zero mean and one std
scaler = preprocessing.StandardScaler().fit(train_x)


train_x_scale=scaler.transform(train_x)
test_x_scale=scaler.transform(test_x)

print(np.mean(train_x_scale,0))
print(np.mean(test_x_scale,0))

# -*- coding: utf-8 -*-

# set the sample weights for the training model
sample_weights=[]
ratio=len(train_y)/sum(train_y==1)/10
for i in range(len(train_x)):
    if train_y[i]==0:
        sample_weights.append(1)
```

```
    else: sample_weights.append(ratio)
```

```
total array shape: (100000, 135)
train_x shape: (90000, 134)
test_x shape: (10000, 134)
test_y shape: (10000,)
train_y shape: (90000,)
[ -4.92e-14  -1.02e-14  -2.34e-14   6.61e-15   1.28e-14  -1.03e-14
   1.46e-14   9.94e-17  -3.27e-14   2.63e-15  -5.05e-15  -4.60e-15
  -2.69e-14  -8.17e-15  -1.48e-14   1.93e-15   4.11e-14  -6.04e-15
   2.70e-14   1.22e-15  -6.42e-15  -6.08e-15   3.10e-14  -6.34e-16
   3.26e-15  -1.15e-14  -1.09e-14  -9.13e-15  -2.37e-14  -6.47e-15
   3.40e-14   7.49e-15  -1.08e-15   2.01e-14   1.75e-14   3.22e-16
   2.86e-14   1.39e-14   1.19e-14   4.25e-15   5.71e-15   1.93e-14
   6.35e-15   9.73e-15   4.13e-15  -2.84e-14  -9.91e-15   5.47e-15
  -3.39e-15   3.19e-14  -2.40e-15   2.64e-15   2.90e-14  -1.84e-14
  -1.03e-15  -2.86e-14   4.12e-14   1.10e-14   8.49e-15  -1.21e-14
   1.42e-14  -1.12e-14  -4.83e-14   2.68e-15  -9.24e-16   1.17e-13
   5.32e-15  -5.99e-15   2.20e-14   1.31e-14   1.64e-14  -7.45e-14
  -7.04e-14  -2.44e-14   9.75e-14  -4.06e-14  -4.95e-14   3.28e-14
  -3.77e-14   1.91e-14  -8.49e-14   3.35e-15   1.78e-15  -2.86e-15
   1.16e-14   3.62e-15  -8.08e-15   1.10e-14  -4.35e-15  -6.69e-15
  -1.46e-15   3.88e-15  -1.71e-15  -4.36e-15  -1.00e-14  -8.94e-15
  -3.75e-15   7.71e-15  -1.31e-15  -5.85e-15  -7.51e-16   1.23e-15
   6.69e-16   3.25e-15   2.18e-15  -2.14e-15  -1.21e-15   3.25e-15
   1.60e-15   3.21e-15  -1.65e-16   2.54e-15   5.08e-15  -2.97e-15
   1.30e-15   4.76e-16   3.18e-16  -5.45e-16  -2.51e-15  -1.46e-15
  -3.12e-15  -3.95e-15   3.03e-15  -2.01e-15   1.84e-16  -1.15e-15
  -1.18e-14  -7.61e-15  -4.40e-16   3.60e-15   3.57e-16  -1.12e-15
  -3.21e-15   3.84e-16]
[-0.38 -0.17 -0.38 -0.04 -0.38 -0.19 -0.38 -0.03 -0.39 -0.12 -0.38 -0.03
 -0.39 -0.04 -0.37 -0.01 -0.39 -0.04 -0.37  0.   -0.39 -0.05 -0.36 -0.   -0.4
 -0.03 -0.36 -0.   -0.4   0.01 -0.35 -0.   -0.4  -0.02 -0.35 -0.02 -0.41
 -0.03 -0.35 -0.03  0.11  0.04 -0.05 -0.11 -0.16 -0.2  -0.24 -0.26 -0.28
 -0.3  -0.38 -0.38 -0.38 -0.38 -0.38 -0.38 -0.38 -0.38 -0.38 -0.38 -0.08
 -0.15 -0.1  -0.13 -0.13 -0.13 -0.14 -0.13 -0.13 -0.08 -0.11 -0.13 -0.15
 -0.15 -0.19 -0.19 -0.17 -0.17 -0.39 -0.36 -0.18 -0.06 -0.2   0.02  0.04
  0.03  0.02  0.04  0.04  0.04  0.03  0.04  0.03  0.06 -0.    0.01  0.01
  0.01  0.01 -0.    0.01  0.02  0.02  0.01 -0.01 -0.02  0.   -0.01 -0.
 -0.01  0.04 -0.05  0.04 -0.05 -0.    0.   -0.01 -0.    0.   -0.    0.01
 -0.01  0.02 -0.01 -0.09 -0.06 -0.06 -0.09 -0.07 -0.05 -0.02  0.01 -0.
 -0.03]
```

3.Model build

3.1 two classes

In [9]:

```
%matplotlib inline
```

logistic regression

In [10]:

```
train_x_scale.shape
```

Out[10]:

```
(90000, 134)
```

In [11]:

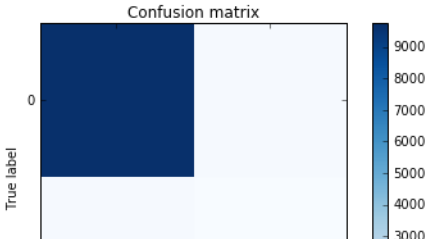
```
#----------------
# logistic l1
#-----------------


from sklearn import linear_model

# set the sample weights for the training model
sample_weights=[]
ratio=len(train_y)/sum(train_y==1)/10
for i in range(len(train_x)):
    if train_y[i]==0:
        sample_weights.append(1)
    else: sample_weights.append(ratio)

        # set the random state to make sure that each time get the same results

time_logistic=time.time()
clf = linear_model.LogisticRegression(C=1, penalty='l1', tol=1e-6,random_state= 987612345)
clf.fit(train_x_scale,train_y)
time_logistic=time.time()-time_logistic

print(time_logistic)

# test the training error
predict_y_logistic =np.array(clf.predict(train_x_scale))
print("train_accuracy is:",sum(predict_y_logistic==train_y)/len(train_y))

# test the score for the train data
from sklearn.metrics import (brier_score_loss, precision_score, recall_score,
                             f1_score)
precision= precision_score(predict_y_logistic,train_y)
recall = recall_score(predict_y_logistic,train_y)
f1=f1_score(predict_y_logistic,train_y)
print("precision is: \t %s" % precision)
print("recall is: \t %s" % recall)
print("f1 score is: \t %s" %f1)

# define a function to prefict the result by threshold
```

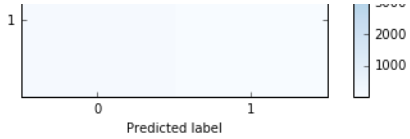
```
# note: logistic model will return two probability
def predict_threshold(predict_proba, threshold):
    res=[]
    for i in range(len(predict_proba)):
        res.append(int(predict_proba[i][1]>threshold))
    return res

predict_y_test_proba =np.array(clf.predict_proba(test_x_scale))

predict_y_test=predict_threshold(predict_y_test_proba,0.5)

# test the score for the train data
from sklearn.metrics import (precision_score, recall_score,
                             f1_score)
print("accuracy is:",sum(predict_y_test==test_y)/len(test_y))
precision= precision_score(predict_y_test,test_y)
recall = recall_score(predict_y_test,test_y)
f1=f1_score(predict_y_test,test_y)
print("precision is: \t %s" % precision)
print("recall is: \t %s" % recall)
print("f1 score is: \t %s" %f1)

%matplotlib inline
## draw chart for the cross table

from sklearn.metrics import confusion_matrix

def plot_confusion_matrix(cm, title='Confusion matrix', cmap=plt.cm.Blues):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(2)
    plt.xticks(tick_marks, [0,1])
    plt.yticks(tick_marks, [0,1])
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')


# Compute confusion matrix
cm = confusion_matrix(test_y, predict_y_test)
np.set_printoptions(precision=2)
print('Confusion matrix, without normalization')
print(cm)
plt.figure()
plot_confusion_matrix(cm)
plt.show()

#---------------
# logistic l2
#-----------------
```

```
from sklearn import linear_model

# set the sample weights for the training model
sample_weights=[]
ratio=len(train_y)/sum(train_y==1)/10
for i in range(len(train_x)):
    if train_y[i]==0:
        sample_weights.append(1)
    else: sample_weights.append(ratio)

        # set the random state to make sure that each time get the same results

time_logistic=time.time()
clf = linear_model.LogisticRegression(C=1, penalty='l2', tol=1e-6,random_state= 987612345)
clf.fit(train_x_scale,train_y)
time_logistic=time.time()-time_logistic

print(time_logistic)

# test the training error
predict_y_logistic =np.array(clf.predict(train_x_scale))
print("train_accuracy is:",sum(predict_y_logistic==train_y)/len(train_y))

# test the score for the train data
from sklearn.metrics import (brier_score_loss, precision_score, recall_score,
                             f1_score)
precision= precision_score(predict_y_logistic,train_y)
recall = recall_score(predict_y_logistic,train_y)
f1=f1_score(predict_y_logistic,train_y)
print("precision is: \t %s" % precision)
print("recall is: \t %s" % recall)
print("f1 score is: \t %s" %f1)

# define a function to prefict the result by threshold
# note: logistic model will return two probability
def predict_threshold(predict_proba, threshold):
    res=[]
    for i in range(len(predict_proba)):
        res.append(int(predict_proba[i][1]>threshold))
    return res

predict_y_test_proba =np.array(clf.predict_proba(test_x_scale))

predict_y_test=predict_threshold(predict_y_test_proba,0.5)

# test the score for the train data
from sklearn.metrics import (precision_score, recall_score,
                             f1_score)
print("accuracy is:",sum(predict_y_test==test_y)/len(test_y))
precision= precision_score(predict_y_test,test_y)
recall = recall_score(predict_y_test,test_y)
f1=f1_score(predict_y_test,test_y)
print("precision is: \t %s" % precision)
```

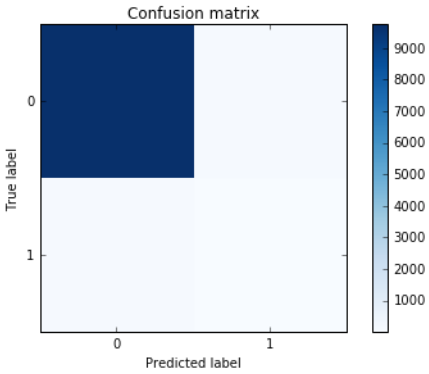
```
print("recall is: \t %s" % recall)
print("f1 score is: \t %s" %f1)

%matplotlib inline
## draw chart for the cross table

from sklearn.metrics import confusion_matrix

def plot_confusion_matrix(cm, title='Confusion matrix', cmap=plt.cm.Blues):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(2)
    plt.xticks(tick_marks, [0,1])
    plt.yticks(tick_marks, [0,1])
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')


# Compute confusion matrix
cm = confusion_matrix(test_y, predict_y_test)
np.set_printoptions(precision=2)
print('Confusion matrix, without normalization')
print(cm)
plt.figure()
plot_confusion_matrix(cm)
plt.show()
```

```
172.18734502792358
train_accuracy is: 0.984222222222
precision is:   0.489117043121
recall is:   0.871250914411
f1 score is:   0.626512361915
accuracy is: 0.9752
precision is:   0.122302158273
recall is:   0.118881118881
f1 score is:   0.120567375887
Confusion matrix, without normalization
[[9735  126]
 [ 122   17]]
```






```
12.392427921295166
train_accuracy is: 0.984144444444
precision is:   0.487474332649
recall is:   0.868960468521
f1 score is:   0.624572480926
accuracy is: 0.9761
precision is:   0.122302158273
recall is:   0.126865671642
f1 score is:   0.124542124542
Confusion matrix, without normalization
[[9744  117]
 [ 122   17]]
```



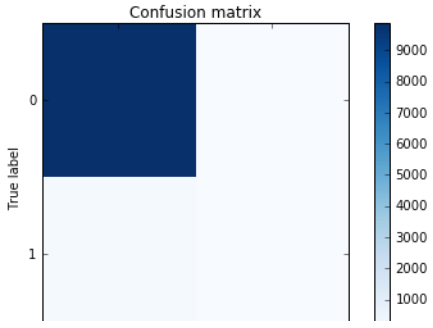
In [12]:

```
#--------------------
# SVM_poly_2
#---------------------

# set the sample weights for the training model
sample_weights=[]
ratio=len(train_y)/sum(train_y==1)/10
for i in range(len(train_x)):
    if train_y[i]==0:
        sample_weights.append(1)
    else: sample_weights.append(ratio)

import time
```

```
from sklearn import svm
# training

# change the depth of the tree to 6, number of estimators=100

t=time.time()
clf = svm.SVC(C=1.0,kernel='poly',degree=2,max_iter=5000,shrinking=True, tol=0.001, verbose=False)

clf.fit(train_x_scale,train_y)

print(time.time()-t)

#testing
# test the training error
predict_y =np.array(clf.predict(train_x_scale))
print("train_accuracy is:",sum(predict_y==train_y)/len(train_y))

# test the score for the train data
from sklearn.metrics import (brier_score_loss, precision_score, recall_score,
                             f1_score)
precision= precision_score(predict_y,train_y)
recall = recall_score(predict_y,train_y)
f1=f1_score(predict_y,train_y)
print("precision is: \t %s" % precision)
print("recall is: \t %s" % recall)
print("f1 score is: \t %s" %f1)

# define a function to prefict the result by threshold
# note: logistic model will return two probability
def predict_threshold(predict_proba, threshold):
    res=[]
    for i in range(len(predict_proba)):
        res.append(int(predict_proba[i][1]>threshold))
    return res


predict_y_test=np.array(clf.predict(test_x_scale))


# test the score for the train data
from sklearn.metrics import (precision_score, recall_score,
                             f1_score)
print("accuracy is:",sum(predict_y_test==test_y)/len(test_y))
precision= precision_score(predict_y_test,test_y)
recall = recall_score(predict_y_test,test_y)
f1=f1_score(predict_y_test,test_y)
print("precision is: \t %s" % precision)
print("recall is: \t %s" % recall)
print("f1 score is: \t %s" %f1)


#draw the crosstab chart
%matplotlib inline
```

```
%matplotlib inline
## draw chart for the cross table

from sklearn.metrics import confusion_matrix

def plot_confusion_matrix(cm, title='Confusion matrix', cmap=plt.cm.Blues):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(2)
    plt.xticks(tick_marks, [0,1])
    plt.yticks(tick_marks, [0,1])
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')


# Compute confusion matrix
cm = confusion_matrix(test_y, predict_y_test)
np.set_printoptions(precision=2)
print('Confusion matrix, without normalization')
print(cm)
plt.figure()
plot_confusion_matrix(cm)
plt.show()
```

```
43.26794648170471
train_accuracy is: 0.988477777778
precision is:   0.606981519507
recall is:   0.9486521181
f1 score is:   0.740295517155
accuracy is: 0.9881
precision is:   0.143884892086
recall is:   1.0
f1 score is:   0.251572327044
Confusion matrix, without normalization
[[9861    0]
 [ 119   20]]
```



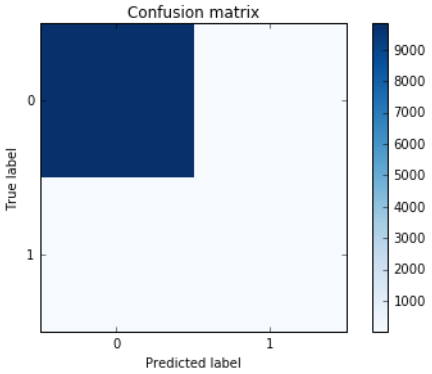
In [13]:

```
#---------------
# decision tree
#-----------------


# set the sample weights for the training model
sample_weights=[]
ratio=len(train_y)/sum(train_y==1)/10
for i in range(len(train_x)):
    if train_y[i]==0:
        sample_weights.append(1)
    else: sample_weights.append(ratio)


from sklearn import tree
# training

# change the depth of the tree to 6, number of estimators=100

t=time.time()
clf =  tree.DecisionTreeClassifier(max_depth=10,random_state= 987612345)
clf.fit(train_x_scale,train_y)

print(time.time()-t)

#testing
# test the training error
predict_y=np.array(clf.predict(train_x_scale))
print("train_accuracy is:",sum(predict_y==train_y)/len(train_y))

# test the score for the train data
from sklearn.metrics import (brier_score_loss, precision_score, recall_score,
                             f1_score)
precision= precision_score(predict_y,train_y)
recall = recall_score(predict_y,train_y)
f1=f1_score(predict_y,train_y)
print("precision is: \t %s" % precision)
print("recall is: \t %s" % recall)
print("f1 score is: \t %s" %f1)

# define a function to prefict the result by threshold
# note: logistic model will return two probability
def predict_threshold(predict_proba, threshold):
    res=[]
    for i in range(len(predict_proba)):
        res.append(int(predict_proba[i][1]>threshold))
    return res
```

```
t=time.time()
predict_y_test_proba =np.array(clf.predict_proba(test_x_scale))
print("test time is:", time.time()-t)
predict_y_test=predict_threshold(predict_y_test_proba,0.5)


# test the score for the train data
from sklearn.metrics import (precision_score, recall_score,
                             f1_score)
print("accuracy is:",sum(predict_y_test==test_y)/len(test_y))
precision= precision_score(predict_y_test,test_y)
recall = recall_score(predict_y_test,test_y)
f1=f1_score(predict_y_test,test_y)
print("precision is: \t %s" % precision)
print("recall is: \t %s" % recall)
print("f1 score is: \t %s" %f1)


#draw the crosstab chart
%matplotlib inline
## draw chart for the cross table

from sklearn.metrics import confusion_matrix

def plot_confusion_matrix(cm, title='Confusion matrix', cmap=plt.cm.Blues):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(2)
    plt.xticks(tick_marks, [0,1])
    plt.yticks(tick_marks, [0,1])
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')


# Compute confusion matrix
cm = confusion_matrix(test_y, predict_y_test)
np.set_printoptions(precision=2)
print('Confusion matrix, without normalization')
print(cm)
plt.figure()
plot_confusion_matrix(cm)
plt.show()
```

```
2.9122314453125
train_accuracy is: 0.993455555556
precision is:   0.760164271047
recall is:   0.997306034483
f1 score is:   0.862735958984
test time is: 0.0032167434692382812
accuracy is: 0.9923
```

```
precision is:   0.58273381295
recall is:   0.81
f1 score is:   0.677824267782
Confusion matrix, without normalization
[[9842   19]
 [  58   81]]
```



In [14]:

```
#----------------------------------------
# Adaboost
#----------------------------------------

# set the sample weights for the training model
sample_weights=[]
ratio=len(train_y)/sum(train_y==1)/10
for i in range(len(train_x)):
    if train_y[i]==0:
        sample_weights.append(1)
    else: sample_weights.append(ratio)


from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier

# training

# change the depth of the tree to 6, number of estimators=100

time_ada=time.time()
clf =  AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=10),n_estimators=100,random_state= 987612345)
clf.fit(train_x_scale,train_y)

print(time.time()-time_ada)

#testing
```

BIBLIOGRAPHY

Anuj Agarwal. High frequency trading: Evolution and the future. 2012.

Bruno Biais, Pierre Hillion, and Chester Spatt. An empirical analysis of the limit order book and the order flow in the paris bourse. *the Journal of Finance*, 50(5):1655–1689, 1995.

Chester I Bliss. The method of probits. *Science*, 79(2037):38–39, 1934.

Ekkehart Boehmer, Robert Jennings, and Li Wei. Public disclosure and private decisions: Equity market execution quality and order routing. *Review of Financial Studies*, 20(2):315–358, 2007.

Peter J Bolland and Jerome T Connor. A constrained neural network kalman filter for price estimation in high frequency financial data. *International Journal of Neural Systems*, 8(04): 399–415, 1997.

PJ Bolland and JT Connor. A robust non-linear multivariate kalman filter for arbitrage identification in high frequency data. *Neural Networks in Financial Engineering (Proceedings of the NNCM-95), eds. AP. N. Refenes, Y. Abu-Mostafa, J. Moody and AS Weigend (World Scientific)*, pages 122–135, 1996.

Tim Bollerslev and Ian Domowitz. Trading patterns and prices in the interbank foreign exchange market. *The Journal of Finance*, 48(4):1421–1443, 1993.

Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

Jean-Philippe Bouchaud, Marc Mézard, Marc Potters, et al. Statistical properties of stock order books: empirical results and models. *Quantitative finance*, 2(4):251–256, 2002.

Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Peter Bühlmann. Bagging, boosting and ensemble methods. In *Handbook of Computational Statistics*, pages 985–1022. Springer, 2012.

Jan Bulla. Application of hidden markov models and hidden semi-markov models to financial time series. 2006.

Lijuan Cao. Support vector machines experts for time series forecasting. *Neurocomputing*, 51: 321–339, 2003.

Damien Challet and Robin Stinchcombe. Analyzing and modeling 1+ 1d markets. *Physica A: Statistical Mechanics and its Applications*, 300(1):285–299, 2001.

Stephan K Chalup and Andreas Mitschele. Kernel methods in finance. In *Handbook on information technology in finance*, pages 655–687. Springer, 2008.

Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001.

Peter K Clark. A subordinated stochastic process model with finite variance for speculative prices. *Econometrica: journal of the Econometric Society*, pages 135–155, 1973.

Rama Cont and Arseniy Kukanov. Optimal order placement in limit order markets. 2013.

Rama Cont, Sasha Stoikov, and Rishi Talreja. A stochastic model for order book dynamics. *Operations research*, 58(3):549–563, 2010.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Robert F Engle. The econometrics of ultra-high-frequency data. *Econometrica*, 68(1):1–22, 2000.

Robert F Engle and Jeffrey R Russell. Forecasting the frequency of changes in quoted foreign exchange prices with the autoregressive conditional duration model. *Journal of empirical finance*, 4(2):187–212, 1997.

Robert F Engle, Robert Ferstenberg, and Jeffrey R Russell. Measuring and modeling execution cost and risk. 2006.

Geir Evensen. The ensemble kalman filter: Theoretical formulation and practical implementation. *Ocean dynamics*, 53(4):343–367, 2003.

Tristan Fletcher. Hybrid evolutionary techniques for fx arbitrage prediction. 2007.

Tristan Fletcher, Fabian Redpath, and Joe DAlessandro. Machine learning in fx carry basket prediction. In *Proceedings of the World Congress on Engineering*, volume 2, 2009.

Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995.

Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

Kuzman Ganchev, Yuriy Nevmyvaka, Michael Kearns, and Jennifer Wortman Vaughan. Censored exploration and the dark pool problem. *Communications of the ACM*, 53(5):99–107, 2010.

Parameswaran Gopikrishnan, Vasiliki Plerou, Xavier Gabaix, and H Eugene Stanley. Statistical properties of share volume traded in financial markets. *Physical Review E*, 62(4):R4493, 2000.

Martin D Gould, Mason A Porter, Stacy Williams, Mark McDonald, Daniel J Fenn, and Sam D Howison. Limit order books. *Quantitative Finance*, 13(11):1709–1742, 2013.

Xin Guo. Optimal placement in a limit order book. In *Theory Driven by Influential Applications*, pages 191–200. INFORMS, 2013.

Anthony D Hall, Nikolaus Hautsch, et al. *A continuous-time measurement of the buy-sell pressure in a limit order book market*. Institute of Economics, University of Copenhagen, 2004.

Yasushi Hamao and Joel Hasbrouck. Securities trading in the absence of dealers: Trades and quotes on the tokyo stock exchange. *Review of Financial Studies*, 8(3):849–878, 1995.

Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12:993–1001, 1990.

Larry Harris. *Trading and exchanges: Market microstructure for practitioners*. Oxford University Press, USA, 2003.

Sherif Hashem. Optimal linear combinations of neural networks. *Neural networks*, 10(4):599–614, 1997.

Md Rafiul Hassan and Baikunth Nath. Stock market forecasting using hidden markov model: a new approach. In *Intelligent Systems Design and Applications, 2005. ISDA'05. Proceedings. 5th International Conference on*, pages 192–196. IEEE, 2005.

Md Rafiul Hassan, Baikunth Nath, and Michael Kirley. A fusion model of hmm, ann and ga for stock market forecasting. *Expert Systems with Applications*, 33(1):171–180, 2007.

Neep Hazarika and John G Taylor. Predicting bonds using the linear relevance vector machine. *Neural networks and the financial markets*, pages 145–155, 2002.

Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Shian-Chang Huang and Tung-Kuang Wu. Wavelet-based relevance vector machines for stock index forecasting. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pages 603–609. IEEE, 2006.

Shian-Chang Huang and Tung-Kuang Wu. Combining wavelet-based feature extractions with relevance vector machines for stock index forecasting. *Expert Systems*, 25(2):133–149, 2008.

Wei Huang, Yoshiteru Nakamori, and Shou-Yang Wang. Forecasting stock market movement direction with support vector machine. *Computers & Operations Research*, 32(10):2513–2522, 2005.

Weibing Huang, Charles-Albert Lehalle, and Mathieu Rosenbaum. Simulating and analyzing order book data: The queue-reactive model. *Journal of the American Statistical Association*, 110 (509):107–122, 2015.

James M Hutchinson, Andrew W Lo, and Tomaso Poggio. A nonparametric approach to pricing and hedging derivative securities via learning networks. *The Journal of Finance*, 49(3):851–889, 1994.

Nicolas Huth and Frédéric Abergel. The times change: Multivariate subordination. empirical facts. *Quantitative Finance*, 12(1):1–10, 2012.

Patrik Idvall and Conny Jonsson. Algorithmic trading: Hidden markov models on foreign exchange data, 2008.

Plamen Ch Ivanov, Ainslie Yuen, Boris Podobnik, and Youngki Lee. Common scaling patterns in intertrade times of us stocks. *Physical Review E*, 69(5):056107, 2004.

Simon J Julier and Jeffrey K Uhlmann. New extension of the kalman filter to nonlinear systems. In *AeroSense'97*, pages 182–193. International Society for Optics and Photonics, 1997.

Balázs Kégl. The return of adaboost. mh: multi-class hamming trees. *arXiv preprint arXiv:1312.6086*, 2013.

Alec N Kercheval and Yuan Zhang. Modelling high-frequency limit order book dynamics with support vector machines. *Quantitative Finance*, 15(8):1315–1329, 2015.

Kyoung-jae Kim. Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1):307–319, 2003.

Andrei A Kirilenko, Albert S Kyle, Mehrdad Samadi, and Tugkan Tuzun. The flash crash: The impact of high frequency trading on an electronic market. *Available at SSRN 1686004*, 2015.

Chung-Ming Kuan and Tung Liu. Forecasting exchange rates using feedforward and recurrent neural networks. *Journal of applied econometrics*, 10(4):347–364, 1995.

Camilla Landen. Bond pricing in a hidden markov model of the short rate. *Finance and Stochastics*, 4(4):371–389, 2000.

Sophie Laruelle, Charles-Albert Lehalle, and Gilles Pages. Optimal split of orders across liquidity pools: a stochastic algorithm approach. *SIAM Journal on Financial Mathematics*, 2(1): 1042–1076, 2011.

Hugh Luckock. A statistical model of a limit order market. *Sidney University preprint (September 2001)*, 2001.

Rogemar S Mamon and Robert J Elliott. *Hidden markov models in finance*, volume 4. Springer, 2007.

Sergei Maslov and Mark Mills. Price fluctuations from the order book perspectiveempirical facts and a simple model. *Physica A: Statistical Mechanics and its Applications*, 299(1):234–246, 2001.

Szabolcs Mike and J Doyne Farmer. An empirical behavioral model of liquidity and volatility. *Journal of Economic Dynamics and Control*, 32(1):200–234, 2008.

Yuriy Nevmyvaka, Yi Feng, and Michael Kearns. Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd international conference on Machine learning*, pages 673–680. ACM, 2006.

Anna A Obizhaeva and Jiang Wang. Optimal trading strategy and supply/demand dynamics. *Journal of Financial Markets*, 16(1):1–32, 2013.

Beomsoo Park and Benjamin Van Roy. Adaptive execution: Exploration and learning of price impact. *Operations Research*, 63(5):1058–1076, 2015.

Fernando Pérez-Cruz, Julio A Afonso-Rodriguez, Javier Giner, et al. Estimating garch models using support vector machines*. *Quantitative Finance*, 3(3):163–172, 2003.

Marc Potters and Jean-Philippe Bouchaud. More statistical properties of order books and price impact. *Physica A: Statistical Mechanics and its Applications*, 324(1):133–140, 2003.

Alessandro Rossi and Giampiero M Gallo. Volatility estimation via hidden markov models. *Journal of Empirical Finance*, 13(2):203–230, 2006.

Ioanid Roşu. A dynamic model of the limit order book. *Review of Financial Studies*, 22(11): 4601–4641, 2009.

Ioanid Rosu. A dynamic model of the limit order book. *Forthcoming in The Review of Financial Studies*, 2009.

Tobias Rydén, Timo Teräsvirta, and Stefan Åsbrink. Stylized facts of daily return series and the hidden markov model. *Journal of applied econometrics*, pages 217–244, 1998.

Jimmy Shadbolt. *Neural Networks and the Financial Markets: Bpredicting, Combining, and Portfolio Optimisation*. Springer Science & Business Media, 2002.

A Christian Silva and Victor M Yakovenko. Stochastic volatility of financial markets as the fluctuating rate of trading: An empirical study. *Physica A: Statistical Mechanics and its Applications*, 382(1):278–285, 2007.

Justin Sirignano. Deep learning for limit order books. 2016.

Francis EH Tay and Lijuan Cao. Application of support vector machines in financial time series forecasting. *Omega*, 29(4):309–317, 2001.

Francis EH Tay and LJ Cao. Modified support vector machines in financial time series forecasting. *Neurocomputing*, 48(1):847–861, 2002.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

Peter Tino, Nikolay Nikolaev, and Xin Yao. Volatility forecasting with sparse bayesian kernel models. In *Proc. 4th International Conference on Computational Intelligence in Economics and Finance, Salt Lake City, UT*, pages 1150–1153, 2005.

Robert R Trippi and Efraim Turban. *Neural networks in finance and investing: Using artificial intelligence to improve real world performance*. McGraw-Hill, Inc., 1992.

Tony Van Gestel, Johan AK Suykens, D-E Baestaens, Annemie Lambrechts, Gert Lanckriet, Bruno Vandaele, Bart De Moor, and Joos Vandewalle. Financial time series prediction using least squares support vector machines within the evidence framework. *IEEE Transactions on neural networks*, 12(4):809–821, 2001.

Steven Walczak. An empirical analysis of data requirements for financial forecasting with neural networks. *Journal of management information systems*, 17(4):203–222, 2001.

Philipp Weber and Bernd Rosenow*. Order book approach to price impact. *Quantitative Finance*, 5(4):357–364, 2005.

Yingjian Zhang. *Prediction of financial time series with Hidden Markov Models*. PhD thesis, Simon Fraser University, 2004.

Nan Zhou, Wen Cheng, Yichen Qin, and Zongcheng Yin. Evolution of high-frequency systematic trading: a performance-driven gradient boosting model. *Quantitative Finance*, 15(8):1387–1403, 2015.

Ilija Zovko, J Doyne Farmer, et al. The power of patience: a behavioural regularity in limit-order placement. *Quantitative finance*, 2(5):387–392, 2002.

BIOGRAPHICAL SKETCH

This is my biography.