2013

Modeling High-Frequency Order Book Dynamics with Support Vector Machines

Yuan Zhang

FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCES

MODELING HIGH-FREQUENCY ORDER BOOK DYNAMICS WITH SUPPORT VECTOR MACHINES

By

YUAN ZHANG

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

Degree Awarded:
Fall Semester, 2013

Yuan Zhang defended this dissertation on November 7, 2013.
The members of the supervisory committee were:

Alec N. Kercheval
Professor Directing Thesis

Xufeng Niu
University Representative

Warren Nichols
Committee Member

Kyounghee Kim
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

Like a long journey, my five years of academic researches in my PhD pursuit is never smooth as its course, full of countless perilous twists and turns, is beset with difficulties and obstacles. I am fortunate and lucky, however, to get many a helping hand in this trip, making it a fruitful and unforgettable journey rather than an aimless and random walk. I would like to take this opportunity, therefore, to acknowledge and express my sincere gratitude to all people who guide me and accompany me through this painful but enlightening journey.

First, Prof. Alec Kercheval, my PhD advisor, firmly believes in and supports the researches in this dissertation. His endless erudition and infectious enthusiasm always inspire me and urge me forward, meanwhile, his resolute patience and consistent help, when I encounter setbacks, transform me into a determined traveller in this rugged journey and let me have the conviction that magnificent spectacle is attainable. Without doubt, the invaluable and enjoyable learning experience I gain under his advice will benefit me for a life time.

Members of my committee, Prof. Warren Nichols, Prof. Xufeng Niu and Pro. Kyounghee Kim, are all extremely generous to me and share with me their broad knowledges without reserve. I enjoy discussing with them as they always provide me with fresh and insightful viewpoints into many problems that frustrate me at times. In addition, their constructive suggestions and detailed comments on the draft of this dissertation are very valuable and tremendously contribute to its finalization.

Prof. Bettye Anne Case and Prof. Giray Ökten, give me the honorable opportunity to study in this prestigious program in USA. I can't never fulfill my goal without the generous help and guidance they have been offering me these years. Dr. Penelope Kirby, Dr. Penny LeNoir, Ms. Blackwelder, Mr. Dodaro, Mr. Grigorian and Mr. Wooland, who have trusted me and helped me to build up and enhance my teaching skills. It's really a precious and memorable experience for me to work with them in the past three years.

In the course of my PhD study, my father Guoguang Zhang falls ill seriously and is hospitalized several times, which fills me with great grief and disruption, and several times I do think of giving up. It is the unconditional love of my father and my mother Yijian Chen as well as their constant encouragement that keeps me forge ahead and empowers me with responsibility and devotion. I am deeply indebted to my parents as they not only profoundly influence my personality, but also shape my outlook on life. Although I may not be able to repay my parents' love, I would still like to dedicate this dissertation to them.

In addition, my uncle Zhongrong Chen and my aunt Wenping Li are very considerate and exceptionally kind to me. They always provide me selfless help and express their sympathy whenever I feel sad and dejected. Their positive and optimistic attitude in face of life's ups and downs set me a good example to follow. My beloved cousins Sally and Rick are affectionately adorable, their heartfelt babble and sweet laughter never fail to refresh and invigorate me, keeping me fully energetic in my work.

Finally, I thank my colleagues and classmates in the department: Yang Liu, He Huang, Bo Zhao, Jian Geng, Ming Zhu, Wen Huang, Yaning Liu, Yu Fan, Guanghong Wang, Quan Yuan, Pierre Garreau, Ahmed Islim, Dawna Jones, Dong Sun, Qiuping Xu, Jian Wang, Yuanda Chen, Fangxi Gu and Corey Harris, for all the help, for all the interesting and insightful discussions we have made. Also, I thank my friends: Xiaoying Zhang, Ching Wah Ho, Yuk Lan Lee, Lixia Zhang, Haolan Zhou, Yu Qiao, Kakit Fung, Joann Loosbrock and Gary Gramjo, for all your supports for me, for accompanying me and delighting my life in such a colorful way.

The greatest joy of life, methinks, is to share happiness with others. I am hoping that these honorable and respectable people who escort me through my journey will accept my gratitude, however scant it is.

TABLE OF CONTENTS

Appendix

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

A machine learning based framework is proposed in this paper to capture the dynamics of high-frequency limit order books in financial markets and automate the prediction process in real-time on metrics characterizing the dynamics such as mid-price and price spread crossing. By representing each entry in a limit order book with a vector of features including price and volume at different levels as well as statistic features derived from limit order book, the proposed framework builds a learning model for each metric with the help of multi-class support vector machines (SVMs) to predict the directions of market movement. Experiments with real data as well as synthetic data establish that features selected by the proposed framework have highly differentiating capability, models built are effective and efficient in predictions on price movements, and trading strategies based on resulting models can achieve profitable returns with low risk.

Key Words: high-frequency limit order book; machine learning; multi-class classifiers; support vector machines (SVMs); feature selection; cross validation; precision, recall, and F-measure; hypothesis tests; trading strategies

CHAPTER 1

INTRODUCTION

1.1 Modeling Limit Order Book Dynamics

The popularity of electronic communications networks (ECNs) has led to the wide adoption of electronic trading systems by many established exchanges including NYSE, NASDAQ, and London Stock Exchange (Cont et. al (2010) [20], Parlour et. al (2008) [63]). As a result, the traditional quote-driven markets have been replaced by order-driven trading platforms: in the former, buy and sell orders and quotes are provided by market makers or dealers, while in the latter, all outstanding orders are aggregated into a limit order book accessible to every market participant. With little human interventions, market orders are executed against the best available prices in the limit order book within nanoseconds rather than minutes as in quote-driven markets (Cartea et. al (2011) [15], Hollifield et. al (2004) [42]). The rapid decrease in order processing time renders trading in financial assets extremely voluminous and highly frequent. For instance, among the trading transactions of US in 2012, high-frequency trading accounted for 84% in stock trades and 51% in equity value (Popper (2012) [68]). Clearly, the characteristics of order-driven trading systems – high-frequency trading and speedy processing time – not only change the dynamics of the markets such as price movement and evolution, but also demand new trading strategies that can capture short-term behavior of underlying assets and make trading decisions in split second (Parlour et. al (1998) [64]). Various researches have been conducted, therefore, to study the limit order book dynamics, attempting to provide guidelines for trading strategies in ever accelerating electronic platforms.

Generally speaking, researches on modeling limit order book dynamics can be grouped into two main categories: statistical modeling and machine learning based methods. In the former approaches, statistical properties of the limit order book for the target financial asset are collected and conditional quantities are then derived and modeled (Cont et. al (2010) [20] Rosu (2005) [71]). On the other hand, machine learning methods mine and represent entries from limit order book in a systematic manner, then categorize the data so that unseen events can be recognized and

classified based on the generalization (Jondeau et. al (2003) [47] Linnainmaa et. al (2008) [51]). It is well established that statistical based methods usually impose strict and typically unvalidated mathematical assumptions on models (Biais et. al (2000) [4]), in addition, parameters used in models may be unobservable (Foucault et. al (2005) [31] Parlour et. al (1998) [64]), and finally, the intensive computation required by statistical methods makes them unlikely to be deployed in real world environments (Bouchaud et. al (2002) [10]). On the contrary, machine learning methods, owing to their data driven characteristics and light-weight computation, have been applied to a variety of financial markets such as liquidity and volatility in bonds and index trading (Hasbrouk et. al (2006) [37] Lo et. al (1997) [52]). For instance, support vector machine (SVM) with multi-kernel technique has been used in tracking the dynamics of foreign exchange markets (Fletcher et. al (2012) [29]).

However, the application of machine learning techniques, especially support vector machines (SVMs) (Vapnik et. al (1995) [22]), in financial markets is still in its infancy. As a supervised learning method that attempts to infer and generalize information from labeled samples, basic SVM, given a set of training data consisting of positive and negative samples, builds a model that analyzes data, recognizes patterns, and assigns new samples into one of the two classes. In order to make positive and negative samples separable with a broad margin, SVMs may map data into high-dimensional feature spaces with the help of kernels. Evidently, basic SVM model is binary and can only distinguish two different categories. To discriminate multiple classes, two methods, reduction and monolithic optimization, can be employed: in the former, a set of binary SVM models is constructed, each for a possible label; while in the latter, a single model is built to differentiate multiple classes. For example, three SVM models are built in (Fletcher et. al (2012) [29]). to label price movement that could be up, down, or stationary. Unfortunately, the models in (Fletcher et. al (2012) [29]) may deliver conflicting labels to the same sample, consequently invalidating the classification of the sample.

Many metrics or indicators have been designed to characterize the dynamics of high-frequency limit order books, among which mid-price – average of best ask and bid prices – and price spread crossing direction – gap between bid and ask prices at different time horizons – are the most promis-

ing (Cont et. al (2010) [20], Fletcher et. al (2012) [29]). In this paper, we employ multi-class SVMs method to capture the dynamics manifested in high-frequency limit order book and predict the movements of mid-price and price spread crossing direction. To build a multi-class SVM model for a metric such as mid-price or price spread crossing, we construct a training data set by using the most up-to-date entries in the high-frequency order book for the target financial asset, which typically consist of prices and volumes at multiple tick levels from both ask and bid sides. The training data set is composed of samples with different classes (or labels) and the distribution of different labels typically follows that of the order book in question. Moreover, each data sample in the training data set is represented as a vector of features that can be price and volume at different levels as well as their derivatives. To reduce the time spent on building models and testing new data point, which is closely related to the dimensionality of the feature vector (i.e., number of features in the vector), features are selected and pruned based on their contributions to the model performance, measured in information gains. The n-fold cross validation method is used to evaluate the performance of the models in terms of precision, recall, and F-measure. Unlike the SVM models in (Fletcher et. al (2012) [29]), however, which may assign contradicting and invalid label to a sample, the models built with the proposed method work together and label an input sample in a majority-vote manner.

Experiments with real data from NASDAQ demonstrate that the multi-class SVM models built in this paper not only predict various metrics with high accuracy, but also deliver predictions efficiently. By further comparing our models against baseline models such as randomized predictor, we show that the improvement in performance by the models built in the proposed framework is dramatic and statistically significant, consequently, the trading strategies built on the trained models can achieve satisfactory payoffs with low risks.

1.2 Simulating Limit Order Book Dynamics

The basic building block for modelling limit order book dynamics by either stochastic methods or machine learning techniques is, without any doubt, the data contained in the limit order book itself, which could be obtained directly from the real world trading activities. However, financial data such as limit order books from actual transactions may not recorded comprehensively owing to

historical or technical reasons. For instance, the Stock Market Crash of 1929, the largest financial crisis of the 20th century that signaled the beginning of the ten-year-long Great Depression, is analyzed and dissected mainly through simulations in addition to limited available real data. It is also typical that the unavailability of real-world financial data to researchers is simply due to confidential or private issues. For example, the full spectrum of the Flash Crash on May 6, 2010, which plunged the Dow Jones Industrial Average into its largest intra-day point loss in history and was believed to be caused primarily by high frequency trading (Popper (2012) [68]), may not be readily available to most researchers; in this case, simulations allow the recovery of the main events around crash moment leading to new insights into the dynamics and momentum of the market. Thus, simulations on financial data are mainly used to mimic the behavior of trading market in real world environments so that simulated data share the same profile as the real-world data and therefore manifest similar dynamics.

Interestingly, simulated data rather than real-world data are sometimes preferred in modeling financial markets as simulation not only is capable of reproducing scenaria that actually happened in the past, but also has the ability to generate a myriad of variants, making it possible to thoroughly study the interactions of different participants in transactional activities. In this regard, the hot-potato effect – quickly buy and then resell contracts to each other among high frequency traders – in the Flash Crash of May 6, 2010 can be easily replicated through simulations. Similarly, the domino effects in stock market – local emerging market crashes evolve rapidly into more severe regional endemics or even global epidemics (Markwat et. al (2008) [57]) – are usually studied and examined via simulations, as a result, the chained panics, a kind of domino effects, spread in Asian crisis of 1997 and stock market crash of October 1987 are investigated comprehensively on various simulation platforms. Besides, the controllable and tunable free variables in simulators also allow the creation of market scenaria that do not happen yet in reality but may very likely occur in the future. It is clear, therefore, that simulations can project the potential directions of market movements in addition to mimic the market dynamics. Furthermore, the scanty collection of data for rarely occurred trading events in real circumstances renders the models built upon it statistically unsound, making the synthetic data generated by simulations absolutely essential to train robust models; additionally, the artificial data from simulations can also be used to further

enhance and verify the capability and adaptivity of learning models trained on real-world data, making the models more agile and robust to unseen situations.

Although simulation models are widely employed to design and analyze large systems in manufacturing, logistics, warfare and other fields, similar modeling techniques, especially event-based and asynchronous discrete-time simulations, are still rarely utilized on financial analysis in general and evaluation of limit order book dynamics in particular (Jacobs et. al (2004) [45]). The simulation methods applied in financial analysis can be broadly categorized into two groups: continuous-time and discrete-time simulations; in continuous-time models, the system being simulated changes its status continuously over time, while in discrete-time models, time advances in non-continuous but discrete increments. Compared to discrete-time simulations, continuous-time financial models may have the advantage of being analytically solvable under certain assumptions on price processes and market evolutions, however, such continuous-time models may be inappropriate when the policies to be evaluated are complex and both detailed micro-structure and macro-structure of the system are to be studied. As dynamics of limit order book are complicate and intricate, demanding careful scrutiny on the micro-structure and interactions of various elements in the system such as price and volume at different tick levels, the framework proposed in this thesis uses discrete-time simulation models to synthesize data. The discrete-time simulation techniques can be further broken down into synchronous and asynchronous modes according to the methods used to advance time steps: in synchronous time models, the clock moves forward by fixed increments such as a second, a minute, or a day and the status of all system constituents is updated at each increment of time; while in asynchronous models, time could leap forward in varying and uneven increments to the next scheduled event. In this thesis, we generate synthetic data to characterize limit order book dynamics largely by using asynchronous discrete-time simulation as it perfectly matches the event-driven feature of the limit order books.

The key idea behind any asynchronous discrete-time simulation model is to constructively create all transactional events in a bottom-up fashion so that the whole gamut of financial market to be simulated can be derived. Based on the methods to generate individual events, the asynchronous discrete-time simulation models may fall into two main categories: agent-based and process-based

methods. In the agent-based mehtod, information in the limit order book including prices and volumes is the results of sequential interactions among agents such as traders and market participants who involve in the transactions under specified circumstances. On the contrary, the process-based simulation directly model each element within limit order book via a mathematical process (Shek (2011) [74]), more specifically, price and volume dynamics can be characterized stochastic processes such as Poisson processes or Brownian motions (Lorenz et. al (2008) [53]), and similarly, the correlation among different processes may be represented accordingly. For instance, the simulation model in (Luckock et. al (2001) [56]) generates a flow of orders with a specified distribution in price instead of mimicking the behavior of each individual trader. As we mainly focus on the aggregate characteristics of order flows at various tick levels rather than the rationality of individual traders, therefore, we employ process-based simulation models to generate artificial data for limit order books.

In the framework proposed by this thesis, the limit order book data are simulated by following with a multi-phase procedure: 1) Parameters Determination: the properties of simulated data to be generated are characterized by a set of parameters that are typically derived from real-world data, and some example parameters includes limit order submission/cancellation rates, market order submission/cancellation rates, price/volume fluctuation ranges, and number of tick levels to be simulated; 2) Order Flows Generation: order flows at differing tick levels are created according to specified arrival rates and fluctuation ranges up to a given period of time; 3) Order Flow Aggregation: events in all order flows from various tick levels are merged into a single limit order book based on their creation time; and 4) Simulated Data Validation: the characteristics of the simulated data are evaluated by computing various statistics of the data such as actual submission/cancellation rates and price/volume fluctuation ranges and comparing them with the specified parameters, so that the quality of the simulated data can be verified. It is reasonable to assume that the synthetic data generated by the above-described procedure possess the properties specified by the given set of parameters should the data pass the quality validation, and therefore could be used the same way as real world data to train and test learning models.

1.3 Aims 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 mid-price or price 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 multi-class support vector machine (SVM) technique, a supervised machine learning method that has been proved to be extremely effective and applied widely in various fields. The performance of the resulting model is validated by a n-fold cross-validation process that evaluates model performance in terms of precision, recall and F_1-measure. In addition, the robustness and consistency of the model performance are verified through a paired resample t-test procedure where the model is compared against some baseline predictors that are constructed without any learning ability. 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. This dissertation designs and implements an automatic framework that can predict effectively and efficiently the desired metrics of limit order book dynamics by employing advanced multi-class SVMs technique, and the models built in the proposed framework not only have a broad prediction coverage – ratio of samples with valid labels over all data, but also manifest superb performance in prediction accuracy and robustness.

2. 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.

3. The novel feature sets devised in this dissertation, including basic, time-sensitive, and time-insensitive features, prove to be effective and possess the ability of generalization and summarization. Additionally, a feature selection process based on information entropy and information gain is designed to construct a minimum and economical set of features that not only trains models with satisfactory performance but also delivers models efficiently.

4. In this dissertation, models built are evaluated with a n-fold cross validation process in terms of precision, recall, and F-measure. Moreover, the performance and robustness of the models are further verified through a significance test procedure where the models trained in the proposed framework are compared against baseline predictors built without any learning capability. The statistical significance tests conducted by using real data as well as simulated data indicates that our models exceedingly outperforms the baseline predictors and the improvement in performance is statistically significant, further justifying the performance of models built in the proposed framework.

5. This dissertation also develops an effective trading strategy based on models built in the proposed prediction framework, which can achieve profitable returns with low risks, demonstrating the effectiveness of the modeling methodology introduced in this dissertation.

6. The discrete-time process based simulation suggested in this dissertation is able to generate synthetic limit order book data with desired characteristics, and the artificial data allow us to build learning models for events that rarely occur in real world conditions.

7. The framework proposed in this dissertation is highly scalable and extendable, making it straightforward to integrate other machine learning methods such as Self-organizing Maps (SOMs), Artificial Neural Networks (ANNs) and logistic regression, and multiple methods can work coordinately to predict metrics that describes the limit order book dynamics in various financial markets.

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

- Chapter 2 – Literature Review: this chapter first 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 – Theory and Mathematical Background: this chapter first gives an overview on the fundamentals of machine learning methodologies, then focuses on the descriptions of support vector machine (SVM) techniques including kernel functions, margins, and multi-class SVMs.
- Chapter 4 – Architecture of the Proposed Framework: this chapter discusses the design rationale and details of the proposed framework to automate prediction process, and the discussion covers the construction of training data, feature selection, model building, synthetic data generation and application, performance evaluation, and trading strategy design.
- Chapter 5 – Experiments and Results: this chapter presents the experimental results of the models built by the proposed framework and analyzes their performance; the model performance are compared and evaluated via aspects of training schemes as well as simulated data; meanwhile, the effects of the trading tactics based on the models are also demonstrated.
- Chapter 6 – Conclusions and Future Work: this chapter concludes the dissertation and points out possible directions for our future work.
- Appendix A – This appendix includes the theory and derivations of SVMs, as the supplementary material that provides the background and building blocks for the SVM model.
- Appendix B – This appendix introduces the information theory and the principles of significance test that direct the feature selection and model performance validation.
- Appendix C – This appendix outlines the flow charts and pseudocodes for multi-SVMs prediction procedures and data simulation.

CHAPTER 2

LITERATURE REVIEW

2.1 Limit Order Book Dynamics and Modeling

The limit order book for an asset, the aggregation of all unexecuted limit orders – the ex ante pre-commitments at pre-specified prices and amounts, has an exceptionally simple form, in reality, however, it provides an immensely large action space with numerous associated states, resulting in complex and volatile economic interactions (Shek (2011) [74] Smith et. al (2003) [75]). The fact that a limit order competes against both existing and future submissions as well as its non-linear payoff when execution further renders the limit order markets high-dimensional and extremely dynamic, making it challenging to rigorously model and empirically quantify various causal or hidden relations (Cont et. al (2010) [20], Rosu (2005) [71]). The intensive research efforts in the last decades, nevertheless, do produce new insights into the dynamics of limit order markets. It has been established, for instance, that the price evolution caused by the momentum of buy and sell orders embedded in delicate processes can be simulated and tracked through a variety of time-sensitive metrics and devices (Cartea et. al (2011) [15], Shek (2011) [74],). In this regard, metrics such as mid-price movement (Cont et. al (2010) [20], Huang et. al (2011) [43]), price spread crossing direction (Fletcher et. al (2012) [29]), distribution of price change durations (Cont et. al (2010, 2012) [19, 20]) are all helpful in understanding intraday price dynamics of limit order book. Therefore, developing models on above-mentioned metrics by either stochastic or machine learning methods to capture the limit order book dynamics is essential to guide trading decision making and facilitate high-frequency transactions.

The early statistics-based modeling endeavor treats limit order book as a clearing house in a queuing system and develops tractable formulas for market-clearing prices by assuming that demand and supply schedules follow independent identical Poisson processes (Mendelson (1982) [59]). The same assumption is also used by the microscopic statistical model for limit order book in (Smith et. al (2003) [75]), which can predict price volatility, bid-ask spread, and probability of

filling orders with the help of a continuous double auction mechanism. A stationary equilibrium model for price formation is proposed in (Foucault et. al (2005) [32]) to investigate the incentives of order placement decision; meanwhile, the model tractability is achieved under the premise that limit orders survive for only one period and would be exogenously cancelled should they be unfilled within the time frame (Parlour et. al (2008) [63]). By further allowing investors to dynamically modify limit orders in real time, the model in (Rosu (2005) [71]) is able to characterize the shape of limit order book, the order flow auto-correlation, and the temporal properties of orders and trades. To simulate the price evolution and interactions, a class of Markovian agent based models is designed in (Bovier et. al (2006) [11]) based on real world like trading rules as well as the behavior of market buyers and sellers who could change their opinions on the stock value stochastically, but the unobservable parameters such as the state of opinion vector governing agent's preference make the model difficult to calibrate and verify.

In contrast to all above stochastic models that focus only on describing unconditional distributions of metrics related to limit order markets, the Markovian queuing model suggested in (Cont et. al (2010) [20]) derives closed-form formula by using Laplace transform for conditional quantities such as mid-price transition probabilities and likelihood of order fulfillment before price transitions. Unfortunately, the hypotheses underlying the model of (Cont et. al (2010) [20]) that arrivals of market orders and limit orders at every price level are independent Poisson processes, time intervals between orders are independent and exponentially distributed, and all orders are of the same size are unsupported by empirical studies on high frequency data. For example, time gaps between orders are shown to be non-exponentially distributed and they are inter-dependent (Bouchaud et. al (2008) [10], Hasbrouk et. al (2006) [37]); similarly, order sizes are heterogeneous and diversified instead of homogeneous (Anderson et. al (2009) [2]). To build more realistic models, some of aforementioned constraints are somewhat relaxed. For instance, limit order sizes are heterogeneous in the birth-death stochastic model of (Huang et. al (2011) [43]), while order arrival intensity is no longer independent in the model of (Shek (2011) [74]), instead, it is simulated with Hawkes' process so that limit order arrivals are both inter-related and clustered (Hawkes (1971) [40]). By approximating intraday dynamics of the limit order book with a Markovian jump-diffusion model and characterizing the joint dynamics of limit order queues with functional central limit theorem, the

proposal in (Cont et. al (2012) [19]) is capable of providing analytical tractability for conditional quantities including mid-price transition probabilities. Despite all the above efforts, the largest obstacle for statistical modeling methods to be employed in real world, however, is the intensive computation they demand when calculating and calibrating quantities of interest in modeling processes.

Compared with the long history of stochastic based modeling methods, researches on machine learning techniques for order book dynamics are quite young but keep emerging. Different from statistic models that are typically mathematics driven, machine learning techniques are data-driven at large, and represent data systematically so that a large amount of data can be mined and generalized automatically (Joachims (1999) [46]). In general, machine learning techniques fall into two main categories: *unsupervised learning* and *supervised learning*, in the former, hidden structures and relations in unlabeled data are inferred and learned by the model to cluster data into different groups, and an unseen data point is assigned to the closest group according to certain similarity measures; while in the latter, a model is built based on a set of labelled data and then is used to classify new data samples. Popular unsupervised learning approaches used in researches on limit order markets include artificial neural network(ANN) (Shalbolt et. al (2002) [72]), k-nearest neighbor method (Blazejewski et. al (2005) [7]), and self-organizing maps(SOMs) (Blazejewski et. al (2006) [6]); meanwhile, support vector machines (SVMs) (Vapnik et. al (1995) [22]), logistic regression (Zheng et. al (2013) [82]), and Naive Bayes (Tino et. al (2005) [79]) are among those widely employed supervised learning methods.

By employing the k-nearest neighbor method – an unsupervised learning technique, a local non-parametric model for trade sign inference with three predictor variables is proposed in (Blazejewski et. al (2005) [7]) to differentiate buyer-initiated trades from those by sellers, and experiments show that the average classification accuracy can reach 71.40%. On the other hand, a self-organizing map (SOM), also a unsupervised learning method, is used in (Blazejewski et. al (2006) [6]) to cluster trades based on features including trade size, best bid and ask volumes, and trade direction; the resulting clusters of buyer-initiated and seller-initiated trades coincide with the non-equilibrium market conditions. Besides unsupervised learning strategies, supervised learning methods are also

utilized to study price evolution and order book dynamics. To this end, logistic regression is used to predict the occurrence of intra-trade price jump in (Zheng et. al (2013) [82]) with the help of variables such as limit order volumes and market order sizes. In comparison, the core of the model in (Fletcher et. al (2012) [29]) to predict the EUR/USD price evolution is multi-kernel SVM that derives a weight for each of multiple kernels and tags a given sample based on the weighted sum of kernels; more specifically, three multi-kernel SVMs are trained to label a new data point as up, down or moveless with respect to spread crossing direction. However, a prediction on a given sample is invalid unless the three independent SVMs agree in their classifications, as a result, the prediction coverage – ratio of samples with valid labels over all data – of the model is extremely low. In summary, although machine learning based models outlined above demonstrate encouraging results, they are yet to be integrated into real world applications to provide real-time trading strategies.

The machine learning based modeling method described in this paper provides a framework to automate the prediction process for limit order book dynamics in real time. Treating prediction on various metrics characterizing dynamics of limit order books as a supervised learning problem, the proposed method first reduces the problem at hand into a set of binary classification tasks and then builds models by using multi-class SVMs for each binary task. Different from the work of (Fletcher et. al (2012) [29]), however, which delivers an exceedingly large number of invalid labels, multi-class models in our method work in a coordinated manner when labelling new samples, therefore, dramatically boosting the coverage. In addition, a much richer set of variables is constructed to depict trading activities in the modeling process, resulting in satisfactory prediction performance in terms of effectiveness and efficiency.

2.2 Simulation on Limit Order Book Characteristics

Simulation techniques are vitally important in system development and analysis, and therefore are widely applied in fields such as warfare, manufacturing, and logistics. There are many motivations to use simulation models and synthetic data generated by various simulation techniques. First, real-world data and simulated data are supplementary to each other and they together

provide ample opportunities to thoroughly understand the complex dynamic structure embedded within the system under study (Lorenz et. al (2008) [53], Willmott (2008) [81]). For instance, if the dynamics of the limit order book are investigated by using both simulated data and model built on real data, then the modeling of limit order book provides theoretical tools to direct the simulation so that artificial data from the simulation could truly represent the momentum and dynamic of the limit order book and share the same charateristics as real data; on the other hand, the well-posed synthetic data from simulation can be used to verify the effectiveness of the model (Eiesland (2011) [27], Smith et. al (2003) [75]). Next, free variables in many simulators, which are controllable and tunable by researchers, not only allow the true reproduction of scenaria that appear in real-world circumstance, but also make it straightforward to create essentially an infinite variety of scenaria that do not yet occur in reality but may likely happen in the future. Hence, besides mimicking existing market dynamics, simulations can also project possible directions of market movements. Finally, by creating a large number of variants to trading events that rarely occur in real situations and using these simulated data to train learning models, simulations can actually improve the performance and robustness of the models.

Traditionally, simulation techniques can be divided into two types: continuous-time and discrete-time models; in the former, the status of the system being simulated changes continuously over time, while in the latter, time advances in discrete increments (Jacobs et. al (2004) [45]). As continuous-time models may be analytically solved by imposing certain constraints on the system under study (Haverkort (1998) [39]), therefore, they are usually employed in financial analysis, in contrast, discrete-time models are rarely used simply because they defy close-formed mathematical solutions. Unfortunately, continuous-time models with analytic formula are typically simplistic and dramatically deviate from the reality, as a result, discrete-time models are more appropriate for complex systems such as limit order book dynamics. The discrete-time simulation methods can be further classified into synchronous and asynchronous models according to the methods to advance time steps: the clock in synchronous models always advances by fixed increments in unit of second, minute, or day; while time in asynchronous models moves forward in varying and non-uniform increments to the next scheduled event. In both synchronous and asynchronous models, however, the status of all system constituents, which could be prices as well as volumes at various tick level, for

instance, in limit order book simulation, may be updated at each increment of time, while change in the status may further trigger subsequent event occurrences, moving the simulation from one status to the next.

Generally speaking, an asynchronous discrete-time simulation model in financial analysis works in a bottom-up manner to first generate all individual events and then aggregate them together to build the whole system. In this regard, two methods are mainly employed to generate individual events: agent-based and process-based methods; in the former, events in the system to be simulated are considered as the consequences of interactions among agents such as traders and market makers who participate the transactions; while in the latter, elements within the simulated system are modeled directly via mathematical processes such as Poisson or Brownian motion (Lorenz et. al (2008) [53], (Shek (2011) [74]). When it comes to the simulation for limit order book, for instance, changes of prices and volumes in agent-based models are all due to the behavior of traders and market participants, thus, data in limit order book are actually the aggregation of activities initiated by traders and market participants under specified circumstances such as imposed market balance. On the other hand, in process-based models, price and volume dynamics at different tick levels are delineated directly with stochastic processes such as Poisson and Hawkes, and correlations among different processes can be simulated in the same manner.

In the agent-based simulation framework proposed in (Chiarella et. al (2002) [18]), the behavior of market participants including fundamentalist, chartist and noise traders are calibrated, and the dynamics of prices and spread are represented as a function of tick size, liquidity and the average lifetime of market orders. The simulation of limit order book in (Eiesland (2011) [27]) is performed by using historical data of a particular stock as the baseline, and updating the simulated order book whenever agents initiate transactional activities, moving simulation from state to state, meanwhile, price and volume at a particular order book level are generated randomly and then merge them together along with data from other levels. The asynchronous market simulator proposed in (Daniel (2002) [25]) allows for a wide variety of agents' behaviors: under the self-referential market hypothesis of rationality, the order book is simulated as a double auction market mechanism and order placement, cancellation as well as execution by agents are simulated through multiple wake-up

processes, which are described by sleeping time parameter τ and characterized through Poisson processes.

Instead of attempting to anticipate the rationality of traders, the model in (Luckock et. al (2001) [56]) assumes that the combined net effect of traders activities is a flow of orders that follows a known mathematic distribution. Consequently, bid and ask orders along with transactions prices are all represented with certain well-behaved distributions and the market, characterized as the interactions between supply and demand, is further simulated by Monte Carlo method. By embedding the impacts of agents' behaviors into price processes characterized as Poisson processes, the modeling method proposed and implemented in (Smith et. al (2003) [75], Guo (2005) [33]) simulates all order flows to the markets independently. More specifically, market orders from both ask and bid sides arrive at a constant rate with a fixed size, while limit order prices arrive at a constant rate with random values generated uniformly; in addition, order cancellation rates are assumed to be constant at different price levels. In the simulation model of (Chen et. al (2010) [17]), the arrival of market ask and bid orders as well as limit ask and bid orders is simulated by a 4-variate Hawkes process, and order cancellation is generated with a constant arrival rate; while parameters in the system such as the cross correlation coefficient matrix of Hawkes process are derived from historical order book data.

In the proposed framework, synthetic limit order book data are generated with asynchronous discrete-time process-based models.

CHAPTER 3

MATHEMATICAL MODEL OF SUPPORT VECTOR MACHINES

The theory of machine learning as well as the support vector machines have deep root in the mathematics (Steinwart et. al (2008) [76]). Since their appearance in the early nineties, support vector machines and related kernel-based methods have been successfully applied in divers fields of application. There are mainly three reasons for the success of SVMs: their ability to learn well with only a very small number of free parameters, their robustness against several types of model violations and outliers, and their computational efficiency compared to several other methods (Bishop (2006) [5]).To thoroughly understand the model behind the SVMs, we need to scrutinize the mathematical concepts as well as the theories that the SVMs are build on (Ng (2012) [61]).

In this chapter, we introduce the important mathematical theories and tools that are highly involved with the research field of this paper. In the reference of (Ng (2012) [61], Boyd et. al (2004) [12]), we formally introduce the core technique that implemented by this paper, known as ***support vector machines (SVMs)***. Intuitively, SVMs is a binary classifier that separates two sets of data by maximizing the margin between them. Thus, in the following, after formalizing the definition of margin by functional margin and geometric margin, we go straight to investigate the optimal margin classifier on the linearly separable case. By introducing the kernel function, we further explore the SVMs on the case of non-linearly separable data. In the end, the principles of multi-class categorizing problem by using SVMs are examined and discussed.

3.1 Functional Margin and Geometric Margin

To derive the definition of margins, we firstly consider a linear classifier for a binary classification problem of the data set $T = \{\vec{x}_i, y_i\}$, $(i = 1, ..., m)$. In the sense, each $\vec{x}_i$ represents a single data point in the data set and $x \in \mathbb{R}^n$, and y_i is the label of the class that each $\vec{x}_i$ belongs to with

$y_i \in Y = \{-1, 1\}$. The classifier tends to find the parameters $\vec{w}$ and b, such that the data points belong to the class -1 and 1 are separated by the hyperplane given by $\vec{w} \cdot x + b = 0$. Specifically, the data points who satisfy $\vec{w} \cdot \vec{x_i} + b > 0$ belong to the class 1, while the points with $\vec{w} \cdot \vec{x_i} + b < 0$ should be in the class of -1. Thus, if a data point has a large positive value when it is plugged in the hyperplane function, which means it is far above from the hyperplane, then it is believed to have high confidence to belong to the class of 1. So, if given a training set, it'd be reasonable to find a decision boundary that allows us to make all correct and confident predictions on the training examples. This leads to the formal definition of function and geometric margins.

Definition 3.1.1 (**Functional Margin**). *Given a training example $(\vec{x_i}, y_i)$, the functional margin of $(\vec{w}, b)$ with respect to the training example is defined to be*

$$\hat{\gamma_i} = y_i(\vec{w} \cdot \vec{x_i} + b) \tag{3.1}$$

Given a training set $T = \{(\vec{x_i}, y_i); i = 1, ..., m\}$, the function margin of $(\vec{w}, b)$ with respect to the training set T is defined to be

$$\hat{\gamma} = \min_{i=1,...,m} \hat{\gamma_i} \tag{3.2}$$

By the equation 3.1, if some data point $\vec{x_i}$ indeed has the label $y_i = 1$, then for the functional margin to be large such that the prediction can be correct and confident, the value of $\vec{w} \cdot \vec{x_i} + b$ should be positively large. On the other hand, if $y_i = -1$, then $\vec{w} \cdot \vec{x_i} + b$ should be negatively large to make the functional margin be large. In sum, when $y_i(\vec{w} \cdot \vec{x_i} + b) > 0$, then prediction result is correct. Thus a large functional margin implies a confident and correct prediction.

However, by scaling the parameter $\vec{w}$ and b in the function $\vec{w} \cdot \vec{x_i} + b$, we can actually make the functional margin $\hat{\gamma_i}$ arbitrarily large without changing the hyperplane. Hence, this property of functional margins makes it not a good measure of prediction confidence. In this regard, the

normalization condition is imposed to derive another measure to calibrate the actual distance between the data point to the hyperplane.

Definition 3.1.2 (**Geometric Margin**). *Given a training example $(\vec{x}_i, y_i)$, the geometric margin of $(\vec{w}, b)$ with respect to the training example is defined to be*

$$\gamma_i = \frac{y_i}{\|\vec{w}\|}(\vec{w} \cdot \vec{x_i} + b) \tag{3.3}$$

Given a training set $T = \{(\vec{x}_i, y_i); i = 1, ..., m\}$, the geometric margin of $(\vec{w}, b)$ with respect to the training set T is defined to be

$$\gamma = \min_{i=1,...,m} \gamma_i \tag{3.4}$$

It is clearly from equation 3.3 that, the geometric margin is invariant to rescaling the parameters $\vec{w}$ and b since the norm of $\vec{w}$ is imposed. Thus, the geometric margin can serve as the measure of confidence in the classification. Particularly, when $\|\vec{w}\| = 1$, the geometric margin would equal to the functional margin.

3.2 The Optimal Margin Classifier and Linearly Separable Case

With the definitions of margins as introduced, it is a natural concern to find a decision boundary for a training data set, by which the geometric margin that separating positive training examples and negative examples is maximized; since in this situation, the classification reflects a set of precise and confident predictions on the training set.

To derive the model of SVMs, we start with the case on the optimal margin classifier on the linearly separable data set. For a linearly separable training data set $T = \{(\vec{x}_i, y_i); i = 1, ..., m\}$, we impose the following optimization problem:

$$\begin{aligned} &\max_{\vec{w},b} \frac{\hat{\gamma}}{\|\vec{w}\|} \\ &\quad s.t.\ \ y_i(\vec{w}\cdot\vec{x_i}+b) \geq \hat{\gamma}, \quad i = 1, ..., m \end{aligned} \tag{3.5}$$

Here, since the functional margin and geometric margin are related by $\gamma = \hat{\gamma}/\|\vec{w}\|$, thus, by maximizing the $\hat{\gamma}/\|\vec{w}\|$, we are actually maximizing the geometric margin over the training set T. Recall the scalability of parameters $\vec{w}$ and b, we can always rescale them to let the functional margin with respect to the training set equal to 1, i.e.,

$$\hat{\gamma} = \min_{i=1,...,m} \hat{\gamma}_i = 1 \tag{3.6}$$

Then after applying equation 3.6 to the problem 3.5 and noticing that maximizing $\hat{\gamma}/\|\vec{w}\| = 1/\|\vec{w}\|$ is actually to minimize the term $\|\vec{w}\|^2$, thus we have the following optimization problem:

$$\begin{aligned} &\min_{\vec{w},b} \frac{1}{2}\|\vec{w}\|^2 \\ &\quad s.t.\ \ y_i(\vec{w}\cdot\vec{x_i}+b) \geq 1, \quad i = 1, ..., m \end{aligned} \tag{3.7}$$

In the form of problem 3.7, we transform the problem into a convex quadratic optimization problem with linear constraints. Thus, we can employ the Lagrange duality to find the dual form of the problem, from which, the efficient algorithm to solve the problem can be derived, and also, it allows to use the kernel functions to tackle the problem, which is not linearly separable in the original input space, in a higher dimensional feature space.

Now, rewrite the constraints into $g_i(\vec{w}) = -y_i(\vec{w} \cdot \vec{x_i} + b) + 1 \leq 0$, we can construct the Lagrangian for the problem 3.7 as:

$$\mathcal{L}(\vec{w}, b, \vec{\alpha}) = \frac{1}{2}\|\vec{w}\|^2 - \sum_{i=1}^{m} \alpha_i [y_i(\vec{w} \cdot \vec{x_i} + b) - 1] \tag{3.8}$$

Because the problem only has inequality constraints, thus there are only α_i's but no β_i's for the Lagrange multipliers. To find the dual form of the problem, we need to first minimize $\mathcal{L}(\vec{w}, b, \vec{\alpha})$ with respect to $\vec{w}$ and b for $\vec{\alpha}$ fixed in order to get $\theta_{\mathcal{D}}$. By setting the derivatives of $\mathcal{L}$ with respect to $\vec{w}$ and b to zero, we have:

$$\nabla_{\vec{w}} \mathcal{L}(\vec{w}, b, \vec{\alpha}) = \vec{w} - \sum_{i=1}^{m} \alpha_i y_i \vec{x_i} = 0 \tag{3.9}$$

which implies that

$$\vec{w} = \sum_{i=1}^{m} \alpha_i y_i \vec{x_i} \tag{3.10}$$

On the other hand, by taking the derivatives with respect to b, we have

$$\frac{\partial}{\partial b} \mathcal{L}(\vec{w}, b, \vec{\alpha}) = \sum_{i=1}^{m} \alpha_i y_i = 0 \tag{3.11}$$

Take the equation 3.10 into the Lagrangian equation 3.8, and by equation 3.11, we have (see details in A.16)

$$\mathcal{L}(\vec{w}, b, \vec{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j (\vec{x_i} \cdot \vec{x_j}) \tag{3.12}$$

Hence, by minimizing $\mathcal{L}$ with respect to $\vec{w}$ and b, we reach the equation 3.12. Now, by fixing $\vec{w}$ and b and imposing the constraints about $\vec{\alpha}$ that α_i for all i and the constraint 3.11, we obtain the dual optimization problem of the original primal problem:

$$\begin{aligned} \max_{\vec{\alpha}} W(\vec{\alpha}) &= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j (\vec{x_i} \cdot \vec{x_j}) \\ s.t.\ \ \alpha_i &\geq 0 \quad i = 1, ..., m \\ \sum_{i=1}^{m} \alpha_i y_i &= 0 \quad i = 1, ..., m \end{aligned} \tag{3.13}$$

Then to maximize the objective function in 3.13 by choosing the α_i's, the dual problem should reach its optimality, thus is solved by the optimal α_i^*. At the same time, guaranteed by Slater's (A.0.2) and KKT conditions (A.0.3), the primal optimization problem is solved by the optimal α_i^*. Particularly, the optimal $\vec{w}^*$ can be found by the equation 3.10, i.e.,

$$\vec{w}^* = \sum_{i=1}^{m} \alpha_i^* y_i \vec{x_i} \tag{3.14}$$

Having found the $\vec{w}^*$, suppose b^* is the corresponding optimal parameter. Then consider the following two equations:

$$\max_{i:y_i=-1} \vec{w^*} \cdot \vec{x_i} + b^* = -1 \tag{3.15}$$

$$\min_{i:y_i=+1} \vec{w^*} \cdot \vec{x_i} + b^* = +1 \tag{3.16}$$

Note that in the equation 3.15, only the data points who have the functional margin equal to -1 should lie on the boundary denoted by the equation. Similarly, the points with functional margin equal to 1 lie on the boundary denoted by the equation 3.16. Actually, the data points which satisfy the above two equations are called the ***support vectors***.

Hence, the optimal b^* can be obtained by using equations 3.15 and 3.16:

$$b^* = -\frac{\max_{i:y_i=-1} \vec{w^*} \cdot \vec{x_i} + \min_{i:y_i=+1} \vec{w^*} \cdot \vec{x_i} + b^*}{2} \tag{3.17}$$

Up to this point, the optimal margin classifier with respect to the training data set T is built. Here, the optimal $\vec{w}^*$ and b^* are related by the optimal α_i^*, which is the solution of the dual problem, the prediction of a new data point can be also given in the form of equation α_i^*. Recall that the classifier is designed to classify a new data $\vec{x}$ into class $y = 1$ when the quantity $\vec{w}^* \cdot \vec{x} + b^*$ is positive and classify to tha class $y = -1$ when it is negative. Thus, the quantity can also be derived and written by referring to equation 3.10:

$$\begin{aligned} \vec{w^*} \cdot \vec{x} + b^* &= (\sum_{i=1}^{m} \alpha_i^* y_i \vec{x_i}) \cdot \vec{x} + b^* \\ &= \sum_{i=1}^{m} \alpha_i^* y_i (\vec{x_i} \cdot \vec{x}) + b^* \end{aligned} \tag{3.18}$$

The last expression of the equation 3.18 shows that the prediction of a new data point can be obtained by only performing the inner product of the vector $\vec{x_i}$ and $\vec{x}$. Furthermore, since we have

$g_i(\vec{w}) = -y_i(\vec{w} \cdot \vec{x_i} + b) + 1 \leq 0$, by the complementary slackness (details and proof in A.0.3), we have:

$$\alpha_i^* g_i(\vec{w}^*) = 0, \quad \forall i = 1, ..., m \tag{3.19}$$

i.e.,

$$\alpha_i^* (y_i(\vec{w}^* \cdot \vec{x_i} + b^*) - 1) = 0, \quad \forall i = 1, ..., m \tag{3.20}$$

From equation 3.20, we can observe that:

$$\begin{aligned} \alpha_i^* > 0 &\Longrightarrow g_i(\vec{w}^*) = 0 \Longrightarrow y_i(\vec{w}^* \cdot \vec{x_i} + b^*) = 1 \\ g_i(\vec{w}^*) < 0 &\Longrightarrow y_i(\vec{w}^* \cdot \vec{x_i} + b^*) > 1 \Longrightarrow \alpha_i^* = 0 \end{aligned} \tag{3.21}$$

Hence, this leads to an important conclusion that the α_i^*'s are non-zero only when the $\vec{x_i}$'s are support vectors. On the other hand, when the inequality constraints g_i's are strictly satisfied, which means the points $\vec{x_i}$'s are far from the separating hyperplane and the margins, the value of corresponding α_i^* equal to zero. Back to the equation 3.18, we can see that the prediction result of a new data point $\vec{x}$ only depends on finitely many terms of α_i^*'s, as well as the inner products with the support vectors, and vectors who are not support vectors don't contribute to the results of prediction.

3.3 Kernel Functions and Linearly Non-separable Case

From the previous sections, we already see how an optimal margin classifier is built under the case of linearly separable training data set T. However, in many practical problems, the training data points are not linearly separable in the input space of $\vec{x_i}$. Thus, to establish the optimal margin classifier, the input data points need to be mapped into a higher dimensional space, in

which the mapped data points may be again separated by the hyperplane given by the following decision function:

$$\vec{w} \cdot \phi(\vec{x_i}) + b = 0 \tag{3.22}$$

where the input data point $\vec{x_i} \in \mathbb{R}^n$, ϕ is a non-linear mapping from $\mathbb{R}^n$ to a higher dimensional feature space $\mathbb{H}^q$.

Comparing to the previous optimization problem 3.7, now we are actually solving the optimization problem in the feature space $\mathbb{H}^q$:

$$\begin{aligned} &\min_{\vec{w},b} \frac{1}{2}\|\vec{w}\|^2 \\ &\quad s.t.\ \ y_i(\vec{w} \cdot \phi(\vec{x_i}) + b) \geq 1, \quad i = 1, ..., m \end{aligned} \tag{3.23}$$

Since all the computation involved with $\vec{x_i}$ are performed through inner product, we obtain the dual problem for 3.23 by following the same derivation of problem 3.7 and incorporating the mapping ϕ:

$$\begin{aligned} \max_{\vec{\alpha}} W(\vec{\alpha}) = &\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \phi(\vec{x_i}) \cdot \phi(\vec{x_j}) \\ &s.t.\ \ \alpha_i \geq 0 \quad i = 1, ..., m \\ &\sum_{i=1}^{m} \alpha_i y_i = 0 \quad i = 1, ..., m \end{aligned} \tag{3.24}$$

Following the same manner, we also have the prediction function after finding the optimal $\vec{w^*}$, α_i^* and b^*, given as:

$$\begin{aligned} \vec{w^*} \cdot \phi(\vec{x}) + b^* &= (\sum_{i=1}^{m} \alpha_i^* y_i \phi(\vec{x_i})) \cdot \phi(\vec{x}) + b^* \\ &= \sum_{i=1}^{m} \alpha_i^* y_i \phi(\vec{x_i}) \cdot \phi(\vec{x}) + b^* \end{aligned} \tag{3.25}$$

We can see that the computation of inner product between the mapped data $\phi(\vec{x_i})$, $\phi(\vec{x_j})$ actually takes an important role in solving the optimization problem as well as the prediction. However, the cost to explicitly compute the mapping $\phi : \mathbb{R}^n \mapsto \mathbb{H}^q$ and then solve the optimization problem in the space $\mathbb{H}^q$ is expensive. In practice, the key to address this issue which is named ***kernel method*** is used to boost up the efficiency and saving the memory allocation space. Instead of calculating the mapping and inner product $\phi(\vec{x_i}) \cdot \phi(\vec{x})$ explicitly, the method employs the ***kernel functions*** to substitute the computation of the mapped inner products. An example of kernel function is given in A.17. Now, we formalize the definition of kernel functions:

Definition 3.3.1. *A pairwise function $\kappa(\vec{x}, \vec{z})$ is a valid kernel if it corresponds to a legal definition of the dot product $\phi(\vec{x}) \cdot \phi(\vec{z})$, i.e., $\kappa(\vec{x}, \vec{z}) = \phi(\vec{x}) \cdot \phi(\vec{z})$, where $\vec{x}$, $\vec{z} \in \mathbb{R}^n$, κ: $\mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$, and ϕ: $\mathbb{R}^n \mapsto \mathbb{R}^q$, $n < q$.*

In other words, $\kappa(\vec{x}, \vec{z})$ is a valid kernel if and only if $\exists \phi$: $\mathbb{R}^n \mapsto \mathbb{R}^q$, $n < q$, s.t. $\kappa(\vec{(x, z)}) = \phi(\vec{x}) \cdot \phi(\vec{z})$. One consequence of this is that kernel functions must be symmetric, since $\phi(\vec{x}) \cdot \phi(\vec{z}) = \phi(\vec{z}) \cdot \phi(\vec{x})$.

For some finite set of m points, $\{\vec{x_1}, ..., \vec{x_m}\}$ and a kernel $\kappa(\vec{x}, \vec{z})$, we define the ***kernel matrix*** to be a square, m-by-m matrix K, so that its (i, j)-entry is given by the value $K_{ij} = \kappa(\vec{x_i}, \vec{x_j})$. From above, we can see that the existence of the mapping ϕ is the key for a valid kernel. Using the definition of kernel matrix, this is guaranteed by the ***Mercer's Theorem***:

Theorem 3.3.1 (**Mercer's Theorem**). *Let $\kappa : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$ be given. Then $\kappa(\vec{x}, \vec{z})$ is a valid kernel if and only if the corresponding kernel matrix is symmetric positive semi-definite for all training sets $T = \{\vec{x_m}\}$, $m < \infty$.*

Mercer's theorem (see proof in A.18) provides a way to verify whether a function is a valid kernel. In practice, There are several kernel functions are commonly used for training the learning model. One is called the ***degree-d polynomial kernel***:

$$\kappa(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j})^d \tag{3.26}$$

When $d = 1$, it is also called ***linear kernel***.

The other one is called ***Gaussian kernel*** or ***Radial basis kernel***, i.e.;

$$\kappa(\vec{x_i}, \vec{x_j}) = \exp(-\gamma \|\vec{x_i} - \vec{x_j}\|^2) \tag{3.27}$$

Intuitively, if $\phi(\vec{x})$ and $\phi(\vec{z})$ are close together, then we might expect $\kappa(\vec{x}, \vec{z}) = \phi(\vec{x}) \cdot \phi(\vec{z})$ to be large. Conversely, if $\phi(\vec{x})$ and $\phi(\vec{z})$ are far apart, for instance, being nearly orthogonal to each other, then $\kappa(\vec{x}, \vec{z}) = \phi(\vec{x}) \cdot \phi(\vec{z})$ will be small. In this sense, we can think of $\kappa(\vec{x}, \vec{z})$ as some measurement of how similar are $\phi(\vec{x})$ and $\phi(\vec{z})$, or of how similar are x and z. This intuition may be supported by the form of the kernel functions above.

Now with the help of the kernel functions, we can map the training data points into a higher dimensional feature space, in which the data points belongs to different categories can be linearly separated by hyperplane. Thus, in the problem of 3.23, all the inner product can be replaced by the kernel function, then the corresponding dual problem in the form of kernel function is:

$$\begin{aligned} \max_{\vec{\alpha}} W(\vec{\alpha}) &= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \kappa(\vec{x_i}, \vec{x_j}) \\ s.t.\ \ \alpha_i &\geq 0 \quad i = 1, ..., m \\ \sum_{i=1}^{m} \alpha_i y_i &= 0 \quad i = 1, ..., m \end{aligned} \tag{3.28}$$

Consequently, by using the kernel function, we also have the prediction function after finding the optimal $\vec{w^*}$, α_i^* and b^*, given as:

$$\begin{aligned} \vec{w^*} \cdot \phi(\vec{x}) + b^* &= (\sum_{i=1}^{m} \alpha_i^* y_i \phi(\vec{x_i})) \cdot \phi(\vec{x}) + b^* \\ &= \sum_{i=1}^{m} \alpha_i^* y_i \phi(\vec{x_i}) \cdot \phi(\vec{x}) + b^* \\ &= \sum_{i=1}^{m} \alpha_i^* y_i \kappa(\vec{x_i}, \vec{x}) + b^* \end{aligned} \tag{3.29}$$

The application of kernels to support vectors machines significantly broader the usage of the model. By using the kernel technique, the nonlinearly separable data points can be well separated in the higher dimensional feature space.

3.4 Soft-margin Support Vector Machines

The derivation of the SVMs as presented so far assumed that the data is linearly separable. While mapping data to a high dimensional feature space via ϕ does generally increase the likelihood that the data is separable, however, the situation is not always guaranteed. On the other hand, when the training data contains the outliers, then the hyperplane separating the data points may be significantly affected by the position of the outliers. To make the optimal classifier work for

non-linearly separable dataset as well as less sensitive to outliers, the optimization problem should be reformulated:

$$\begin{aligned} \min_{\vec{w},b,\xi_i} \ & \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{m}\xi_i \\ & s.t. \ \ y_i(\vec{w}\cdot\vec{x_i}+b) \geq 1-\xi_i, \quad i=1,...,m \\ & \xi_i \geq 0, \quad i=1,...,m. \end{aligned} \tag{3.30}$$

Then this optimization problem allows to have the functional margin less than 1, however, if an data points has function margin $1-\xi_i$, with $\xi_i > 0$, then the optimization objective should increase the cost by $C\xi_i$. ξ_i's are called ***slackness variables***. The parameter c controls the relative weighting between the goals of making the $\|\vec{w}\|^2$ small — or maximize the margin, and the goal of ensuring that most of the training examples have functional margin at least 1.

By rewriting the problem 3.30 into standard form:

$$\begin{aligned} \min_{\vec{w},b,\xi_i} \ & \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{m}\xi_i \\ & s.t. \ \ 1-\xi_i - y_i(\vec{w}\cdot\vec{x_i}+b) \leq 0, \quad i=1,...,m \\ & -\xi_i \leq 0, \quad i=1,...,m. \end{aligned} \tag{3.31}$$

The dual problem can be summarized as (see details in A.26):

$$\begin{aligned}
\max_{\vec{\alpha},\vec{\beta}} W(\vec{\alpha}) &= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j (\vec{x_i} \cdot \vec{x_j}) \\
s.t.\ \ \alpha_i &\geq 0 \quad i = 1, ..., m \\
\beta_i &\geq 0 \quad i = 1, ..., m \\
\alpha_i + \beta_i &= C \quad i = 1, ..., m \\
\sum_{i=1}^{m} \alpha_i y_i &= 0 \quad i = 1, ..., m
\end{aligned} \tag{3.32}$$

Furthermore, according to KKT conditions, the complementarity requires that for any primal optimal $(\vec{w}^*, b^*, \vec{\xi}^*)$ and dual optimal $\vec{\alpha}^*, \vec{\beta}^*$,

$$\begin{aligned}
\alpha_i^* (1 - \xi_i^* - y_i(\vec{w}^* \cdot \vec{x_i} + b^*)) = 0, \quad i = 1, ..., m \\
\beta_i^* \xi_i^* = 0, \quad i = 1, ..., m
\end{aligned} \tag{3.33}$$

From the first condition above, we can see that if $\alpha_i^* > 0$, then in order for the product to be zero, then $1 - \xi_i^* - y_i(\vec{w}^* \cdot \vec{x_i} + b^*) = 0$. It follows that

$$y_i(\vec{w}^* \cdot \vec{x_i} + b^*) \leq 1 \tag{3.34}$$

Since $\xi_i \geq 0$ by primal feasibility. Similarly, if $\beta_i^* > 0$, then $\xi_i^* = 0$ to ensure complementary slackness property. From the primal constraint, $y_i(\vec{w} \cdot \vec{x_i} + b^*) \geq 1 - \xi_i$, then we have

$$y_i(\vec{w}^* \cdot \vec{x_i} + b^*) \geq 1 \tag{3.35}$$

Notice that, since $\beta_i^* > 0$ is equivalent to $\alpha_i^* < C$ because $\alpha_i^* + \beta_i^* = C$. Thus we can summarize the KKT conditions as follows:

$$\alpha_i^* < C \Rightarrow y_i(\vec{w}^* \cdot \vec{x_i} + b^*) \geq 1 \tag{3.36}$$

$$\alpha_i^* > 0 \Rightarrow y_i(\vec{w}^* \cdot \vec{x_i} + b^*) \leq 1 \tag{3.37}$$

or equivalently,

$$\begin{aligned} \alpha_i^* = 0 &\Rightarrow y_i(\vec{w}^* \cdot \vec{x_i} + b^*) \geq 1 \\ 0 < \alpha_i^* < C &\Rightarrow y_i(\vec{w}^* \cdot \vec{x_i} + b^*) = 1 \\ \alpha_i^* = C &\Rightarrow y_i(\vec{w}^* \cdot \vec{x_i} + b^*) \leq 1 \end{aligned} \tag{3.38}$$

Finally, we further simplify the dual problem by observing that each pair of constraints of the form:

$$\beta_i \geq 0 \qquad \alpha_i + \beta_i = C \tag{3.39}$$

is actually equivalent to single constraint $\alpha_i \leq C$. Thus, we actually just solve the optimization problem:

$$\begin{aligned} \max_{\vec{\alpha},\vec{\beta}} W(\vec{\alpha}) &= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j (\vec{x_i} \cdot \vec{x_j}) \\ s.t.\ \ & 0 \leq \alpha_i \leq C \quad i = 1, ..., m \\ & \sum_{i=1}^{m} \alpha_i y_i = 0 \quad i = 1, ..., m \end{aligned} \tag{3.40}$$

Applying the feature mapping ϕ and the kernel function that $\kappa(\vec{x_i}, \vec{x_j}) = \phi(\vec{x_i}) \cdot \phi(\vec{x_j})$, the optimization can adapt to the situation of mapping input data into a higher dimensional feature space and then perform the the linear separation. In this case, the above optimization actually is:

$$\begin{aligned} \max_{\vec{\alpha},\vec{\beta}} W(\vec{\alpha}) &= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \kappa(\vec{x_i}, \vec{x_j}) \\ s.t.\ \ & 0 \leq \alpha_i \leq C \quad i = 1, ..., m \\ & \sum_{i=1}^{m} \alpha_i y_i = 0 \quad i = 1, ..., m \end{aligned} \tag{3.41}$$

Up to this point, an optimal binary classifier is built for non-linearly separable input data, which also has tolerance and stable to allow data outliers.

3.5 Mechanics of Multi-class SVMs

Obviously, the binary SVMs is not suitable for the multi-class classification problem proposed by this paper, in which case $Y = \{l_1, ..., l_k\},\ k > 2$. Thus, multi-class SVMs models are built to labelled the 3 different classes given by the prediction metrics. Two approaches are used when

building the models: *multi-class to binary reduction* (Hastie et. al (1998) [38]) and *multi-class optimization* methods (Crammer et. al (2001) [23]).

In the multi-class to binary reduction method, the learning problem is reduced to a set of binary classification tasks and a binary classifiers is independently built for each label l_k with *one-against-all* (Hastie et. al (1998) [38]) or *one-against-one* (Knerr et. al (1990) [49]) training technique. For one-against-all training, it constructs k binary-SVM models for each of the classes. The ith SVMs is trained with all of the examples in the ith class as positive labelled examples, while remaining samples are assigned with negative labels. After all the binary SVMs are built, k quadratic programming problems are solved and k decision functions are obtained $\vec{w_i} \cdot \phi_i(\vec{x}) + b_i$ and $i = 1, ..., k$. The class of $\vec{x}$ is determined by the largest value of the decision function:

$$\mathscr{C}(\vec{x}) \equiv \underset{i=1,...,k}{\arg\max}((\vec{w_i}) \cdot \phi_i(\vec{x}) + b_i) \tag{3.42}$$

For one-against-one method, $k(k-1)/2$ binary SVMs models are built, where each one is trained on data from 2 classes. Thus $k(k-1)/2$ decision functions are obtained, the sign of $(\vec{w}_{ij}) \cdot \phi_{ij}(\vec{x}) + b_{ij}$ determines whether $\vec{x}$ is in the ith or in the jth class, i.e. if $(\vec{w}_{ij}) \cdot \phi_{ij}(\vec{x}) + b_{ij} > 0$, the vote for the ith class is added by one, otherwise the jth class is increased by one. By this ***majority vote*** manner, the voting result eventually decides which class an unseen data point belongs to.

CHAPTER 4

ARCHITECTURE OF THE PROPOSED FRAMEWORK

In this section, we first describe the dynamics of high-frequency limit order book and key metrics to characterize price evolution, then present the rationale behind our design on the framework to automatically predict price movements with machine learning techniques. At the same time, we also discuss various aspects of multi-class SVM – the workhorse of the proposed framework, and delineate our tactics to validate models built with SVMs and methods to evaluate their performance. As a supplementary validation method, the generation of the synthetic limit order book data are also discussed and introduced in this chapter.

4.1 Limit Order Book Dynamics and Metrics

All the records of high-frequency trading activities up to current moment are organized into a database with two major components: message book and order book, the former stores basic information on each trading event such as occurring time and transaction type, while the latter keeps unexecuted limit orders for both bid and ask. Sample message book and order book extracted from stock AAPL in NASDAQ are depicted in Table 4.1. It is clear from Table 4.1 that each row of message book represents a trading event that could be order submission, cancellation, or execution as shown in column "Event Type". The arrival time of an event given in column "Time", measured from the midnight of the day, is in seconds and nanoseconds; while the order price, in the unit of US dollar, and its number of shares as well as transaction direction (ask or bid) are presented in Columns "Price", "Volume", and "Direction", accordingly. On the other hand, each entry of order book, also shown in Table 4.1, groups ask and bid events on n different price levels (typically $n = 10$) along with their volume sizes. More specifically, ask prices are ordered ascendingly while bid prices descendingly, then ask and bid prices are paired together. Take the $(k-1)$-th row of order book as an example: the first pair of ask/bid prices is 585.69/585.44 while the second pair is

585.71/585.40, and it is straightforward that the first pair is the best ask/bid prices, that is, the lowest ask price and the highest bid price.

Table 4.1: Data sample: AAPL order book and message book. The line index is denoted in the first column on the left in term of k. The prices are in dollars ($), and the volumes are in unit of shares. On each price level, a pair of price and volume are shown on both ask and bid side.

Message book

	Time(sec)	Price($)	Volume	Event Type	Direction
$k-1$	34203.011926972	585.68	18	execution	ask
k	34203.011926973	585.69	16	execution	ask
...	...	...	...	...	...
$k+4$	34203.011988208	585.74	18	cancellation	ask
$k+5$	34203.011990228	585.75	4	cancellation	ask
...	...	...	...	...	...
$k+8$	34203.012050158	585.70	66	execution	bid
$k+9$	34203.012287906	585.45	18	submission	bid
$k+10$	34203.089491920	586.68	18	submission	ask

Order book

	Ask^1		Bid^1		Ask^2		Bid^2		Ask^3		Bid^3		...
	Price	Vol.	Price	Vol.	Price	Vol.	Price	Vol.	Price	Vol.	Price	Vol.	...
$k-1$	585.69	16	585.44	167	585.71	118	585.40	50	585.72	2	585.38	22	...
k	585.71	118	585.44	167	585.72	2	585.40	50	585.74	18	585.38	22	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...
$k+4$	585.71	118	585.70	66	585.72	2	585.44	167	585.75	4	585.40	50	...
$k+5$	585.71	118	585.70	66	585.72	2	585.44	167	585.80	100	585.40	50	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...
$k+8$	585.71	100	585.44	167	585.80	100	585.40	50	585.81	100	585.38	22	...
$k+9$	585.71	100	585.45	18	585.80	100	585.44	167	585.81	100	585.40	50	...
$k+10$	585.68	18	585.45	18	585.71	100	585.44	167	585.80	100	585.40	50	...

It is evident from Table 4.1 that a new entry in the message book typically causes one fresh record to be added into the order book. For instance, the transaction event at k-th row of message book of Table 4.1, the execution of an ask order at the price $585.69 with 16 shares, exactly cancels out the best ask price and its volume in Row $k-1$ of order book, making the next best ask price, $585.71, become the new best ask price as shown in Row k of the order book. It can also be observed from message book that multiple trading events could arrive within milliseconds as demonstrated from Row $k-1$ to $k+10$, leading to the drastic fluctuation of prices and volumes in the order book.

Although a variety of metrics have been designed to capture the price fluctuation, our experiences and experiments point out that mid-price movement and price spread crossing are among the most promising metrics. The moving direction of the mid-price, defined as the mean of the best ask price P_t^{ask} and best bid price P_t^{bid} at time epoch t, $P_t^{mid} = \frac{1}{2}(P_t^{ask} + P_t^{bid})$, indicates the actual transaction price as well as the potential trending. Affected by many factors including interchanges between counter-investors, market momentums from multiple paths and liquidities in particular timing and environment, some of which are even hidden or unobservable, mid-price movement actually embodies the current as well as historical price information and projects the trajectory of future price into a probability space. As a result, mid-price movement has very complex but close relationship with market dynamics. The three possible scenarios of mid-price movements — upward, downward and stationary can be clearly illustrated by using the order book at Table 4.1. Compared to Row $k-1$, the mid-price at Row $k+4$ increases to (\$585.71+\$585.70)/2 = \$585.705 from (\$585.69+\$585.44)/2 = \$585.565 due to upward movement of both the best ask price and the best bid price. While the mid-price keeps stationary as time advances from Row $k+4$ to $k+5$ owing to the motionless best ask/bid prices, but it moves downward to $(\$585.68 + \$585.45)/2 = \$585.565$ when transactions further proceed to Row $k+10$ where best bid/ask prices decrease using Row $k+5$ as the baseline. Of course, the mid-price movement could interact with the price dynamics in many other ways, in this regard, the mid-price may move monotonically along with the best price of one side should the best price of the other side keep unchanged, which is demonstrated by the up movement of the mid-price from Row $k-1$ to k and its down movement from Row $k+5$ to $k+8$. Supposed that mid-price movement can be correctly predicted, then long positions can be taken at the price \$585.69 when the event of Row $k-1$ occurs, subsequently at Row $k+4$, profits can be harvested by selling the asset at price \$585.70, obviously one tick higher than the buying price. Thus, as a statistical arbitrage indicator, should future mid-price movement be accurately predicted, proper trading strategy could be devised to accomplish positive expected returns.

On the other hand, the bid-ask spread, the difference between the best ask and bid prices $S_t = P_t^{ask} - P_t^{bid}$, can also be used to track the price change. An arbitrage opportunity may occur when price movement is significant enough to cross the bid-ask spread in time period of $(t,\, t+\Delta t)$. Similar to mid-price, price spread crossing has very strong correlation with market dynamics, which

can be depicted with three scenarios: 1) an upward price spread crossing occurs when the best bid price at $t + \Delta t$ is greater than the best ask price at time t ($P^{bid}_{t+\Delta t} > P^{ask}_{t}$); 2) a downward price spread crossing happens when the best ask price at $t + \Delta t$ is less than the best bid price at time t, ($P^{ask}_{t+\Delta t} < P^{bid}_{t}$); and 3) no price spread crossing takes place if $P^{ask}_{t+\Delta t} \geq P^{bid}_{t}$ and $P^{bid}_{t+\Delta t} \leq P^{ask}_{t}$. It is obvious that when an upward price spread crossing appears as indicated by Row $k-1$ and $k+4$ of the order book in Table 4.1, — $P^{ask}_{t_{k-1}} < P^{bid}_{t_{k+4}}$ — profit is guaranteed if long position is taken on the asset with best ask price \$585.69 at time $k-1$ and then is short sold later with a higher best bid price \$585.70 at time $k+4$. In the same manner, an downward price spread crossing, demonstrated at the $(k+5)$-th and $(k+10)$-th rows of order book where $P^{bid}_{t_{k+5}} > P^{ask}_{t_{k+10}}$, can be exploited by short selling the asset at time $k+5$ with the best bid price \$585.70 and then buying it back with a lower best ask price \$585.68 at time $k+10$.

Compared to mid-price, the spread crossing offers a much stronger signal for arbitrage opportunities as it offers a secure position for profits without any risk. Therefore, these two metrics can work independently or coordinately to provide guidance for trading strategies depending on particular scenarios. For instance, the mid-price direction is good for statistical arbitrage with bounded loss that may be essential for risk control, while the spread crossing excels at detecting risk free arbitrage opportunities. Of course, the premise for the arbitrage opportunities described above is that the directions of mid-price movement and price spread crossing can be predicted accurately, which is the main task tackled in this paper.

4.2 Design Rationale for the Proposed Framework

As the trading on an asset goes on, information on its limit orders accumulates and the order book becomes voluminous, rendering it nearly impossible to manually analyze and extract any useful patterns from the data. It is thus desired and natural to resort to machine learning techniques to automatically discover the intricate relations hidden in the massive data; in addition, the generalization ability from data intrinsic to machine learning based modeling methods also makes it realistic to predict the evolution of various metrics in an effective and efficient manner. To build a learning model for a given metric such as mid-price movement, a set of labelled samples,

termed training data, should be prepared, in which each data point is characterized with a vector of features. With the help of machine learning strategies such as SVMs, the model for the specified metric is constructed by learning the hidden structure in the training data represented with feature vectors. Before applied to unseen data for predictions, the model is subject to a validation procedure to ensure its soundness and robustness. To keep up with the rapidly changed dynamics of the limit order book, the training data are frequently updated so that models can be refreshed to learn and subsume any new characteristics appeared in the data.

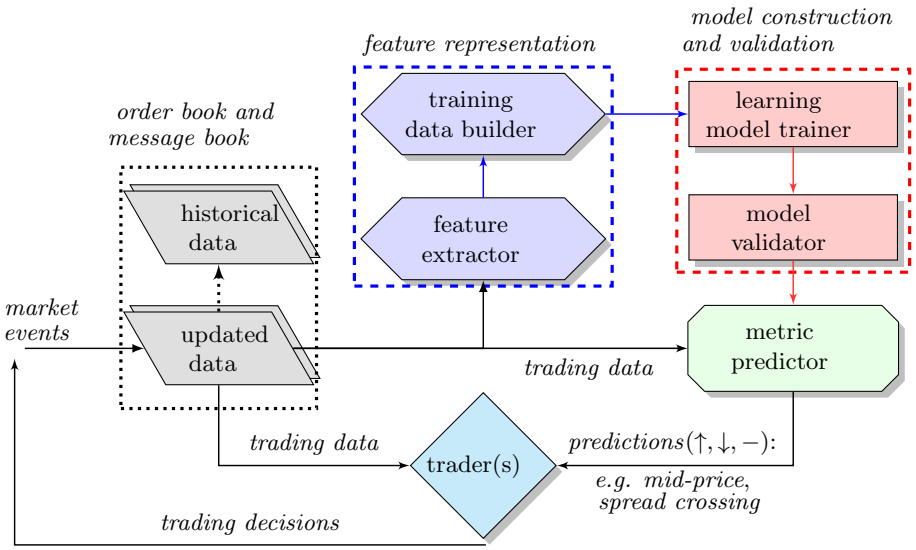

Figure 4.1: Architecture of framework for predicting order book dynamics

Generally speaking, separate learning models should be built for different metrics depicting limit order book dynamics due to their unique and diverse characteristics. To build a machine learning model for a specific metric, the following four-phase process is employed.

- **Features representation**: the information on order book and message book is converted into a format suitable for machine learning methods to manipulate.
- **Learning model construction**: a machine learned model is created for the prediction metric based on the training data automatically collected from both order book and message book.

- **Learning model validation**: the model built by learning methods is evaluated and validated using particular performance measurements.
- **Unseen-data classification**: the constructed learning model automates the prediction on unseen data from real time trading activities.

In the above described four-phase process, which is the foundation of the proposed framework automating the predictions on metrics related to order book dynamics and architecturally outlined in Figure 4.1, the first three phases are conducted offline while the last phase works online to provide predictions on metrics for real time trading events. More specifically, in the architecture of Figure 4.1, the "order book and message book" module is responsible for collecting information on all trading activities including historical and up-to-date data. Such trading information is analyzed and manipulated by the "feature representation" module so that each trading event is expressed as a vector of features such as price, volume, and transaction type; subsequently, a subset of events is selected to form the training data for each metric we are interested in. Using the training data as its input, the module "model construction and validation" builds a model for the metric in question with the help of machine learning techniques, and then the model is validated through specified measurements. After performance evaluation, the model is used to label newly arriving trading events with respect to the specified metric, and the predictions delivered by the model can act as guidance for trading strategies. Details of the main modules in Figure 4.1 will be provided in subsequent descriptions.

4.3 Feature Extraction and Training Data Preparation

The continuous stream of trading activities for an asset is collected into message book and order book as shown in Table 4.1. However, data stored in message book and order book are not suitable for machine learning to digest directly, module "feature representation" is therefore designed to convert the data into the format understood by machine learning methods. More specifically, each data point is represented by a set of features extracted or derived from the data. It is evident that such a conversion would be extremely labor-intensive and require a significant amount of time should it be performed manually due to the massive volume of the data. Thus, the objective of

module "feature representation" is to automate the extraction of features for every data point in the message and order books. With well-formatted feature vectors for data points in place, the module "feature representation" further randomly samples some data points to construct the training data set for each metric used to characterize the dynamics of the limit order book. Generally speaking, the set of training data can be expressed as $T = \{\vec{x_i}, y_i\},\ (i = 1, ..., m)$, where $\vec{x_i} \in \mathbb{R}^p$ is the feature vector for the ith data point and $y_i \in Y = \{l_1, ..., l_k\}$ is its true label identifying the category this data point belongs to. Suppose we are interested in predicting the direction of mid-price movement, for instance, its label set could be $Y = \{upward, downward, stationary\}$.

To better profile the original data and fully represent all label categories of the metric in question, the population in the training data set, although randomly sampled, should follow the distribution of metric labels in the entire collection of data. For example, assuming that the ratio of labels for metric mid-price (i.e., up, down, and stationary) of stock AAPL shown in Tables 4.1 is 1:1:2 and a set of training data with size 4000 is constructed, then ideally the numbers of samples in the training data set with label *upward, downward* and *stationary* should be 1000, 1000, and 2000, respectively. It is typical that each metric has its own unique label set and label distribution, making it necessary to build different training data sets for distinct metrics. Meanwhile, the size of the training data set for different metrics may vary as well depending on the characteristics of the data and the metrics. In this regard, the size of a training data set for a metric is determined in the proposed framework by a cross-validation process that monitors the relationship between size of training data and the performance of the model built on the corresponding training data to avoid overfitting – a phenomenon that exaggerates random error or noise rather than describes statistical patterns of underlying data. As pointed out before, the dynamics of the limit order book could change dramatically within an extremely short period of time, rendering it quite likely that characteristics of order book manifested at the current moment may not be well captured by previously built training data. Hence, the module "feature representation" frequently reconstructs the training data by replace the oldest data points with the most up-to-date ones to ensure the refreshness of the training data.

Table 4.2: Feature vector sets by categories: basic set, time-insensitive set and time-sensitive set. i denotes the level index of limit order book. Each element or dimensional of the feature vector is called an attribute of the vector.

Basic Set	Description(i = *level index*)
$v_1 = \{a_1^k\}_{k=1}^{4n} = \{P_i^{ask},\ V_i^{ask},\ P_i^{bid},\ V_i^{bid}\}_{i=1}^{n},$	*price and volume(n levels)*

Time-insensitive Set	Description(i = *level index*)
$v_2 = \{a_2^k\}_{k=1}^{2n} = \{(P_i^{ask} - P_i^{bid}), (P_i^{ask} + P_i^{bid})/2\}_{i=1}^{n},$	*bid-ask spreads and mid-prices*
$v_3 = \{a_3^k\}_{k=1}^{2n} = \{\lvert P_{i+1}^{ask} - P_i^{ask}\rvert, \lvert P_{i+1}^{bid} - P_i^{bid}\rvert\}_{i=1}^{n},$	*price differences*
$v_4 = \{a_4^k\}_{k=1}^{4} = \{\frac{1}{n}\sum_{i=1}^{n} P_i^{ask},\ \frac{1}{n}\sum_{i=1}^{n} P_i^{bid},\ \frac{1}{n}\sum_{i=1}^{n} V_i^{ask},\ \frac{1}{n}\sum_{i=1}^{n} V_i^{bid}\},$	*mean prices and volumes*
$v_5 = \{a_5^k\}_{k=1}^{2} = \{\sum_{i=1}^{n}(P_i^{ask} - P_i^{bid}),\ \sum_{i=1}^{n}(V_i^{ask} - V_i^{bid})\},$	*accumulated differences*

Time-sensitive Set	Description(i = *level index*)
$v_6 = \{a_6^k\}_{k=1}^{4n} = \{dP_i^{ask}/dt,\ dP_i^{bid}/dt,\ dV_i^{ask}/dt,\ dV_i^{bid}/dt\}_{i=1}^{n},$	*price and volume derivatives*
$v_7 = \{a_7^k\}_{k=1}^{6} = \{\lambda_{\Delta t}^{la},\ \lambda_{\Delta t}^{lb},\ \lambda_{\Delta t}^{ma},\ \lambda_{\Delta t}^{mb},\ \lambda_{\Delta t}^{ca},\ \lambda_{\Delta t}^{cb}\ \}$	*average intensity of each type*
$v_8 = \{a_8^k\}_{k=1}^{4} = \{\mathbf{1}_{\{\lambda_{\Delta t}^{la} > \lambda_{\Delta T}^{la}\}},\ \mathbf{1}_{\{\lambda_{\Delta t}^{lb} > \lambda_{\Delta T}^{lb}\}},\ \mathbf{1}_{\{\lambda_{\Delta t}^{ma} > \lambda_{\Delta T}^{ma}\}},\ \mathbf{1}_{\{\lambda_{\Delta t}^{mb} > \lambda_{\Delta T}^{mb}\}}\},$	*relative intensity indicators*
$v_9 = \{a_9^k\}_{k=1}^{4} = \{d\lambda^{ma}/dt,\ d\lambda^{lb}/dt,\ d\lambda^{mb}/dt,\ d\lambda^{la}/dt\},$	*accelerations(market/limit)*

Remarks:

* a_1^k $(k = 4i - j)$ denotes the ask price, volume; bid price, volume at level i for $j = 3, 2, 1, 0$, respectively.
* a_2^k $(k = 2i - j)$ denotes the bid-ask spread and mid-price at level i for $j = 1, 0$, respectively.
* a_3^k $(k = 2i - j)$ denotes the price differences for ask and bid at level i for $j = 1, 0$, respectively. $(P_{n+1} = P_1)$
* a_6^k $(k = 4i - j)$ denotes derivative of ask price, volume; bid price, volume at level i for $j = 3, 2, 1, 0$, respectively.

In the training data set $T = \{\vec{x_i}, y_i\},\ (i = 1, ..., m)$, the most important and critical component is $\vec{x_i} \in \mathbb{R}^p$, the feature vector for i-th data point. To expedite the construction of feature vectors and characterize dynamics of order book in various aspects, we device and outline in Table 4.2 three different sets of features: *basic*, *time-insensitive*, and *time-sensitive*, all of which can be derived automatically from the data. Also to provide a way for accessing each attribute in the feature vector, the remarks of Table 4.2 list the formula to represent each individual attribute with respect to each feature vector. Evidently, features in the basic set simply includes prices and volumes at both ask and bid sides up to n different levels, which can be directly fetched from the order book shown in Table 4.1. In comparison, features in the time-insensitive set cannot be obtained directly from the message and order books but easily derived from the information embedded in each data point itself. Of this, bid-ask spread and mid-price, price fluctuation ranges, as well as average price and volume at different price levels are calculated in feature sets v_2, v_3, and v_5, respectively; while

v_5 is designed to track the accumulated differences of price and volume between ask and bid sides, which, in turn, simulates the equilibrium of demand and supply.

By further taking historical statistics of limit order book into consideration, we device the features in the time-sensitive set presented in Table 4.2. Particularly, in feature set v_6, derivatives dP_i^{ask}/dt and dP_i^{bid}/dt measure the changing speed in ask/bid prices at price level i, and the changing rate for ask/bid volumes is computed in a similar manner. The average intensity – defined as the arrival rate of a certain trading type – is calculated in feature set v_7 for limit ask/bid orders (denoted as $\lambda_{\Delta t}^{la}$ and $\lambda_{\Delta t}^{lb}$), market ask/bid orders (denoted as $\lambda_{\Delta t}^{ma}$ and $\lambda_{\Delta t}^{mb}$), as well as cancellation ask/bid orders (denoted as $\lambda_{\Delta t}^{ca}$ and $\lambda_{\Delta t}^{cb}$). In contrast, features in v_8 focus on the discrepancy between short term and long term intensities for different trading type denoted as $\lambda_{\Delta t}^{type}$ and $\lambda_{\Delta T}^{type}$, respectively, where *type* can be limit ask/bid orders as well as market ask/bid orders (denoted as la, lb, ma, and mb). By defining an indicator function $\mathbf{1}_{\{f(.)\}}$ to be 1 if $f(.)$ is true and 0 otherwise, then function $\mathbf{1}_{\{\lambda_{\Delta t}^{type} > \lambda_{\Delta T}^{type}\}}$ obviously determines whether trading type in question experiences a more intensive activities currently. For example, should $\mathbf{1}_{\{\lambda_{\Delta t}^{ma} > \lambda_{\Delta T}^{ma}\}}$ assumes value 1, it is clear that more ask executions activities than usual occur and mid-price and spread crossing may go upward with high expectation. The acceleration of a trading type presented as the derivative of its intensity is captured in feature set v_9. For instance, if both $d\lambda^{ma}/dt$ and $d\lambda^{lb}/dt$ have positive sign, then market ask orders and limit bid orders arrive faster, which could drive up the bid-ask spread and may subsequently lead to an upward spread crossing.

Armed with the features described in Table 4.2, the feature vector $\vec{x_i}$ for the i-th data point in the training data set can be formed by simply selecting one or more feature sets and then concatenating them together. It seems intuitively that a data point could be represented more thoroughly should a larger set of features is used, unfortunately, that is not the case. First, some features may depict similar characteristics of the underlying data, making them redundant to each other. Next, some features may not capture the intrinsic hidden patterns in the data for certain metrics, as a result, their inclusion in the feature vectors can only introduce noise in model training. Finally, a sizeable feature vector for a data point also means a demanding computational intensity in model construction as well as real-time prediction process, which is heavily determined by the dimension-

ality of the feature space – the number of attributes in the feature vector. Therefore, it is beneficial to select a minimum set of attributes in the feature space, termed economized feature set, such that a good balance between effectiveness and efficiency can be attained.

In the proposed framework, attributes in feature vectors of samples for a metric are selected according to their contributions to the performance of the resulting model, which are measured in terms of information gain (Cover et. al (2006) [21]). It is well accepted in information theory that information in a system is characterized with *entropy* – a measure of uncertainty or unpredictability of a system, which can be computed as $H(Y) = -\sum_{y \in Y} p(y) \log_2(p(y))$ for a given variable Y (e.g., a training data set) with various values ($y \in Y$), where $p(y)$ is the probability for y belongs to a class of Y, computed as the proportion of class in Y. If samples in the training data Y are partitioned with respect to an attribute X, then the entropy of Y, after observing feature X, becomes $H(Y|X) = \sum_{x \in X} p(x) \sum_{y \in Y} p(y|x) \log_2(p(y|x))$. Consequently, information gain of Y contributed by X is defined as $IG(X) = H(Y) - H(Y|X)$, a measure on the reduction of uncertainty in Y given the information about attribute X. It is evident that a larger $IG(X)$ means a more significant drop in the uncertainty of Y given X. Thus, by ordering ascendingly all attributes in feature vectors of training set Y according to their information gain, the attribute with the highest information gain can be easily found and selected as the most significant contributor to the performance of the model to be built. The same process can be repeated until an economized feature set is formed such that the performance of the model built upon it is comparable to that delivered by the model constructed with the full feature set.

4.4 Learning Model Construction and Validation

In the proposed framework, models are built with Support Vector Machine (SVM) techniques based on the training data prepared by the module "feature representation", which can be represented as $T = \{\vec{x_i}, y_i\}$, $(i = 1, ..., m)$, where each data point $\vec{x_i} \in \mathbb{R}^p$ has a p dimensional feature vector and a true label (or class) $y_i \in Y = \{l_1, ..., l_k\}$. In the simplest case where $Y = \{l_1 = +1, l_2 = -1\}$, a binary SVM classifier is constructed to classify data points of T into two clusters separated by a hyperplane and at the same time to generalize relations hidden

in the data so that unseen examples can also be recognized. Expressed as $\vec{w} \cdot \phi(\vec{x}) + b = 0$, the hyperplane maximizes the summation of its shortest distances to the closest positive and negative examples, here, $\vec{w} \in \mathbb{R}^q$ is the weight vector and b is the bias. Through $\vec{w}$ and b learned from the training set T, the prediction problem can be represented as $y_i = \vec{w} \cdot \phi(\vec{x_i}) + b$, where $\phi(x)$ represents a non-linear mapping of the input instances $\vec{x_i} \in \mathbb{R}^p$ into a higher dimensional feature space $\mathbb{H}^q$ making data points linearly separable and $y_i \in \{l_1 = +1, l_2 = -1\}$ is the corresponding prediction.

The mapping $\vec{x} \in \mathbb{R}^p \mapsto \phi(\vec{x}) \in \mathbb{H}^q$ is performed by a kernel function $\kappa(\vec{x_i}, \vec{x_j}) = \phi(\vec{x_i}) \cdot \phi(\vec{x_j})$. The objective function for the binary SVM is:

$$\begin{aligned} \min_{\vec{w},b,\xi_i} \ & \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i}^{m}\xi_i \\ & y_i(\vec{w} \cdot \phi(\vec{x}) + b) - 1 + \xi_i \geq 0, \forall_i \\ & \xi_i \geq 0, \ i = 1, \ \ldots \ , n, \end{aligned} \tag{4.1}$$

In practice, the objective function is transformed into its Wolfe dual form – a Quadratic Programming (QP) optimization that maximizes:

$$\begin{aligned} & \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j}^{m}\alpha_i\alpha_j\phi(\vec{x_i}) \cdot \phi(\vec{x_j}), \\ & where \ 0 \leq \alpha_i \leq C \ \forall_i, \sum_{i=1}^{N}\alpha_i y_i = 0 \end{aligned} \tag{4.2}$$

The solution α_i of problem 4.2 are then used to compute $\vec{w} = \sum_{i=1}^{m}\alpha_i y_i \phi(\vec{x_i})$, and by grouping data points with their corresponding $\alpha_i > 0$ into the set of *support vectors* $\mathbb{S} = \{s_1, s_2, ...\}$, bias b can then be calculated as $b = \frac{1}{|\mathbb{S}|}\sum_{s\in\mathbb{S}}(y_s - \sum_{n\in S}\alpha_n y_n \phi(\vec{x_n}) \cdot \phi(\vec{x_s}))$. The label for an unseen example $\vec{x}$ can be $+1$ if $\sum_{i=1}^{|\mathbb{S}|}\alpha_i y_i \vec{s_i} \cdot \vec{x} + b$ is positive, and -1 otherwise. It is clear that in feature space $\mathbb{H}^q$, $\kappa(\vec{x_i}, \vec{x_j})$ can be fitted into the objective function through inner products without explicit computation of ϕ. The kernel functions mainly used in the proposed framework are polynomial kernel $\kappa(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j} + 1)^z$ and radial basis function kernel $\kappa(\vec{x_i}, \vec{x_j})$ =exp$(-\|\vec{x_i} - \vec{x_j}\|^2/\sigma^2)$ (Vapnik et. al (1995) [22]).

When label set $Y = \{l_1, ..., l_k\}$ has more than two members ($k > 2$), the classification problem is no longer binary, instead, a multi-class categorization problem should be solved. In this paper, we reduce the multi-class learning problem into a set of binary classification tasks and build a binary classifier independently for each label l_k with *one-against-all* or *one-against-one* methods (Hastie et. al (1998) [38], Knerr et. al (1990) [49]). In one-against-all training method, k binary SVM models are constructed, one for each class in Y; to this end, when training the ith SVM ($i = 1, ..., k$), examples in the ith class are treated as positive while samples in other classes as negative. After k binary SVMs are built, an unseen data point is assigned to the class with the largest value generated by these k binary models. In contrast, one-against-one training method builds $k(k-1)/2$ binary SVM models, each of which is trained on data from 2 classes and therefore can only distinguish between two different labels. When labelling an unseen data point, these $k(k-1)/2$ models work coordinately as follows: each model votes on the two labels it recognizes, then votes from all models are aggregated and the label with the majority vote acts as the final tag for the given data point.

Before a newly built learning model for a metric is actually applied to predict out-of-sample data, it is subject to evaluation and validation with the help of an n-fold cross validation process that works as follows: the training data set T is first divided into n equally-sized partitions $\{T_i, i = 1, ..., n\}$, each of which has the same distribution as T with respect to the metric in question, and then the following steps are repeated n times guided by iteration index $i = 1, ..., n$.

- Training: Partition T_i is set aside as validation (test) set while the remaining $n-1$ partitions are combined to form a new training set $T_i' = \cup_{j \neq i} T_j$, which is used to build learning model L_i.
- Predicting: Model L_i is employed to label samples in validation set T_i and a sample is predicted correctly if its assigned label by L_i agrees with its true label.
- Measuring: Performance measures such as accuracy, precision, recall and F_β-measure are computed for model L_i.

In the n-fold cross validation outlined above, the prediction accuracy is defined as the ratio of the number of correctly predicted examples over the size of validation set. The precision (denoted as P) of a class is the ratio between the number of correctly labelled samples over the total number of samples assigned to the class; while the recall (denoted as R) of a class is the ratio of correctly

predicted examples over the total number of examples whose true labels belong to the class. The harmonic mean of precision and recall is computed by the F_β-Measure: $F_\beta = (1+\beta^2)PR/(\beta^2 P+R)$, and when $\beta = 1$, meaning that both precision and recall are equally important, we obtain the F_1-measure $F_1 = 2PR/(P + R)$. When β varies, more weights are assigned to precision or recall due to possible requirements of practical problems. In the n-fold cross validation process, the average of above measures from n iterations are used as the overall measurements for the learning model.

To justify that the performance of models built in the proposed framework is much better with statistical significance compared against certain baseline predictors, we resort to the paired sample t-test that determines whether a statistical hypothesis should be accepted at a given confidence level. Supposed that we have two classifiers, $\mathcal{C}_1$ and $\mathcal{C}_2$, and would like to compare their performances in terms of measurement M based on a data set S with size n. Then we can conduct a series of k tests and in each trial, the data set S is splitted into two parts: training set T with size n_1 and verification set V with size n_2 (obviously, $n_1 + n_2 = n$), subsequently, classifiers $\mathcal{C}_1$ and $\mathcal{C}_2$ are trained on data set T and their performance measurements m_1 and m_2 are evaluated on data set V. By denoting the difference between the performance measurements of the two classifiers at trial i as $d_i = m_{1i} - m_{2i}$, we obtain a sequence $\mathscr{D} = \{d_1, d_2, d_3, ..., d_k\}$. Assuming that the mean and standard deviation of $\mathscr{D}$ are μ_d and s_d, respectively, we can define the null hypothesis as $H_0 : \mu_d = 0$. In order to determine whether the null hypothesis should be accepted at the specified confidence level α, we compute $t = \frac{\mu_d}{s_d\sqrt{\frac{1}{J}+\frac{n_2}{n_1}}}$ and further derive the p-value $p = \mathbb{F}(t^2, 1, N-1)$ via the cumulative $\mathbb{F}$ distribution function, where $N = km$ with m trials are performed, J is the number of independent data sets that are sampled; then, the null hypothesis should be accepted if $p > \alpha$, implying that the difference in performances of classifiers $\mathcal{C}_1$ and $\mathcal{C}_2$ are statistically insignificant, otherwise, it should be rejected.

4.5 Simulation and Synthetic Data Generation

The simulated data, as introduced previously, not only give a feasible way to replicate the rare circumstances, which are worthy of further investigation, but also provide a platform, through which reproduction of scenarios that appear in real world can be made by characterized parameters

in the simulators. Besides training and testing the model through real data, the model proposed is also implemented against synthetic data that are generated from limit order book simulation. The goal of the simulation serves two aspects: Firstly, the simulated data mimic the real data such that the profiles of simulated data and real data are somehow similar or matched. In this case, the model trained by simulated data can provide validation and comparison to the model trained from real data, this further justifies the performance of the model. Secondly, controlled by certain free parameters, the simulated data should be able to create desired dynamic limit order book metrics embedded within various scenarios. Thus, by simulating the order book data with characteristics, the dynamic model proposed in this paper can be further validated and tested against certain specified circumstances. The simulation steps for limit order book mainly follow four main phases:

- Parameters Determination

  The properties of simulated data to be generated are characterized by a set of parameters that are typically derived from real-world data. The parameters include limit order submission/cancellation rates, market order submission/cancellation rates, price/volume fluctuation ranges, and number of tick levels to be simulated; The parameters can be customized hence give the flexibility for tuning the characteristics of the simulated limit order book data.

- Order Flows Generation

  Order flows at differing tick levels are created according to specified arrival rates and fluctuation ranges. Limit order arrival of each level of order book is simulated by Poisson process. The intensity of each price level is calibrated from real data. On the other hand, the market ask and bid order can be also simulated by two independent Poisson processes. Finally, the cancellation rate of each price level is proportional to the order volume in the level. In sum, on each price level, it has its own submission as well as cancellation processes. For the best ask and best bid, two more processes are used for simulating the execution.

- Data Flows Aggregation

  Events in all order flows from various tick levels are merged into a single limit order book based on their creation time, this exactly follows the event-driven characteristics of the limit order book. Thus, by the specified duration of time, all the events are aggregated in a timely manner, and eventually form the complete order flow that reflects the time series.

- Simulated Data Validation

  The characteristics of the simulated data are evaluated by computing various statistics of the data such as actual submission/cancellation rates and price/volume fluctuation ranges and comparing them with the specified parameters, so that the correctness of the simulated data in terms of arrival intensities can be verified. The main validation measurement are mean square error (MSE) and the cosine similarity.

The following contents of this section introduce the procedures and details of synthetic data generation. Let $\hat{T}$ denote the whole duration of the simulation period in seconds, which follows the length of a complete trading day in real data. K is the length of price range with tick as unit size. For example, to cover a price range around [\$577.33, \$592.33], we take $K = 1500$, and 1 tick = \$0.01. Starting from 1 to 1500, Each price takes 1 grid of the 1500 grids from low to high, which corresponds to the lowest possible bid price to the highest possible ask price. Let i be the grid index for the price, thus $i = 0$ corresponds to the price \$577.33 and $i = 1500$ is the price \$592.33. The grid $q_i = \{\hat{P}_i, \hat{V}_i\}$, stores the price $\hat{P}_i$ and volume $\hat{V}_i$ on the grid. New orders can arrive in any grid from $i = 1$ to $i = 1500$, corresponding to the price range [\$577.33, \$592.33]. And by dividing $\hat{T}$ into $\hat{M}$ equal periods of $\hat{\delta t}$, the order book data of $\hat{T}$ are generated by combining $\hat{M}$ windows of data simulated in each $\delta\hat{t}_p$, $p = 1, 2, ..., \hat{M}$, i.e., $\hat{T} = \hat{M}\hat{\delta t}$.

Table 4.3: Parameters used in the overall simulation. $\hat{T}$ is the total length of simulation duration in seconds. $\hat{M}$ is the number of each small simulation window. $\hat{\delta t}$ is the duration of each simulation window in seconds. K is the number of grids in simulation, each grid has its own processes to generate events. Price range gives the overall range for the possible prices lie on the grids.

$\hat{T}$ (s)	$\hat{M}$	$\hat{\delta t}$ (s)	K	Price Range (\$)
25100.66	251	100	1500	[577.33, 592.33]

The simulation is performed during each simulation window $\delta\hat{t}_p$. In the first window $\delta\hat{t}_1$, the best ask price and bid price are initialized by randomly generating within the price range and placed into the corresponding grids $q_{i_a^*}$ and $q_{i_b^*}$. Thus for any grid q_i, the tick distance j to its opposite best quote is calculated as $j = |i - i_a^*|$ or $j = |i - i_b^*|$, depending on if it's on bid side ($i \leq i_b^*$) or ask side ($i \geq i_a^*$). When the simulation starts, all grids q_i ($i = 1, ..., 1500$) generate their own

order arrival and cancellation events by using Poisson processes independently with intensity which varies by j. The time stamp of each event is recorded. Besides limit order arrival and cancellation, another two independent Poisson processes are used to generate the events of execution, i.e., the market order arrival, for $q_{i_a^*}$ and $q_{i_b^*}$, respectively. For the grids q_i that $i_b^* < i < i_a^*$, both limit ask order and bid orders are opened for placement.

Table 4.4: Formula and corresponding events on grid q_i with respect to j. i is the index for the grid and j is the tick distance to its opposite best quote. The formula are given to find j with respect to different i. The possible events refer to the events could happen on the grid, each type of event is controlled by an independent Poisson process.

i	j	possible events occur
$i = i_b^*$	$j_b = \lvert i_b^* - i_a^* \rvert$	limit bid order arrival, cancellation, execution
$i = i_a^*$	$j_a = \lvert i_a^* - i_b^* \rvert$	limit ask order arrival, cancellation, execution
$i < i_b^*$	$j_b = \lvert i - i_a^* \rvert$	limit bid order arrival; cancellation
$i > i_a^*$	$j_a = \lvert i - i_b^* \rvert$	limit ask order arrival; cancellation
$i_b^* < i < i_a^*$	j_a and j_b	limit ask and limit bid order arrival

Note that, some events may cause the grids of $q_{i_a^*}$ and $q_{i_b^*}$ moving, such as the execution or cancellation on the best quotes, or new limit order submission on grids q_i ($i_b^* < i < i_a^*$), thus, we keep tracking $q_{i_a^*}$ and $q_{i_b^*}$ and updating the indices i_a^* and i_b^* once the events mentioned above occur. Therefore, parameter j — the distance to opposite best quote would also be updated for each grid. Hence the intensities of limit order arrival and cancellation on each grid are subject to change according to j. After $\hat{\delta t}$ ends, all events generated among all grids are merged according to the time stamp in an increasing order. Thus, we have recorded all events that happen in $\hat{\delta t}$, and each of which represents an order placement: submission, cancellation, execution in the order book.

From the introduction above and Table 4.4, the intensity of Poisson process that generates events on grid q_i depends on the index i and the distance j. On q_i that $i < i_b^*$ or $i > i_a^*$, two independent Poisson processes simulate the limit order arrival and cancellation process respectively. The intensities of arrival process and cancellation processes for each grid are calibrated from the real data: using the tick size distance j to the opposite best quote as parameter, the limit order arrival rate is calibrated as $\mu(j) = N_l(j)/\hat{T}$, where $N_l(j)$ is the number of limit order arrival during

the whole day $\hat{T}$ in real order book data. j is the tick size distance to the opposite best quote. In this manner, all $\mu(j)$'s can be calibrated from the real data statistics for ask and bid respectively, and by linear interpolation, we obtain full distribution of $\Lambda^{ask} = \{\lambda^{ask}(j)\}$ and $\Lambda^{bid} = \{\lambda^{bid}(j)\}$ with $j = 1, ..., 1500$. Figure 4.2 shows the interpolated arrival rates for ask and bid with respect to different j. In Figure 4.2, the x-axis is the tick distance on each price level to its opposite best quote, and the y-axis gives the magnitude of the average intensity in the unit of orders/second. It is clear that the distribution of the arrival rates tend to follow gamma law, which is disagree with the assumption in (Cont et. al (2010) [20]), which indicates the distribution should be power law. Now with given j, we can retrieve the limit order arrival rate for q_i, denoted as $\hat{\lambda}_i^{ask}(j)$ or $\hat{\lambda}_i^{bid}(j)$, according to Λ^{ask} and Λ^{bid}.

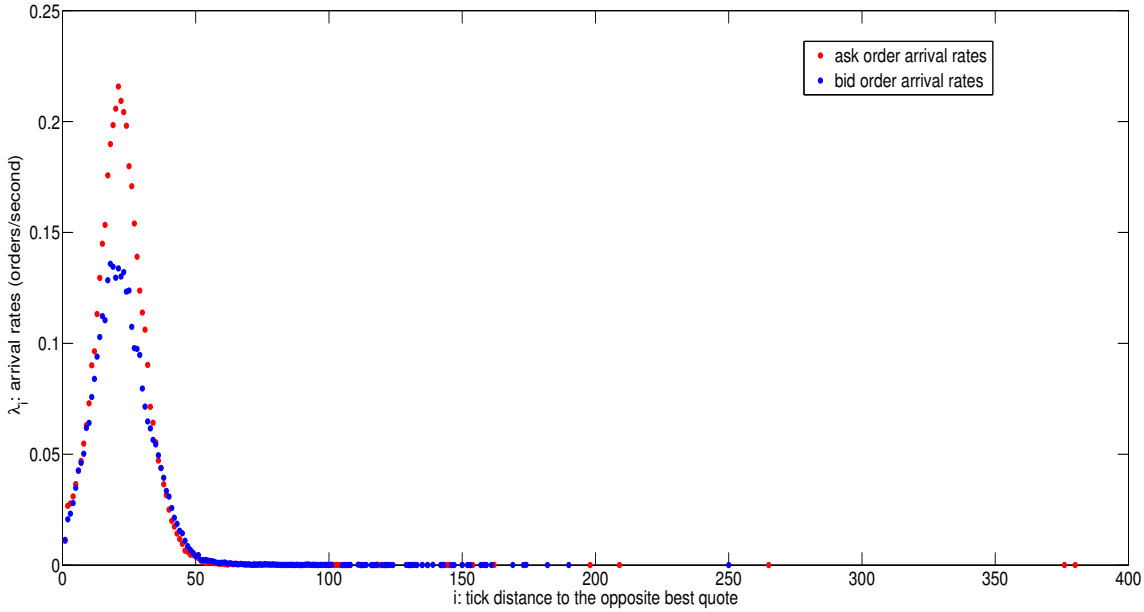

Figure 4.2: Arrival rates of limit ask and bid orders. Calibrated based on the ticker AAPL. The y-axis is the arrival rates given as the average intensity in orders/second. The x-axis denotes the distance i to the opposite best quote measured in tick size. 1 tick = \$0.01.

Similarly, the distribution of cancellation rates with respect to j are given as $C^{ask} = \{\nu^{ask}(j)\}$ and $C^{bid} = \{\nu^{bid}(j)\}$, which are also calibrated from the real data. We have $\nu^{ask}(j) = N_c^{ask}(j)/\hat{T}$ and $\nu^{bid}(j) = N_c^{bid}(j)/\hat{T}$. $N_c^{ask}(j)$ (or $N_c^{bid}(j)$) is the number of cancellation orders during $\hat{T}$ with j tick distance to its opposite best quote. Different from the arrival rates, we want to associate

the cancellation rate with the volume size $\hat{V}_i$ on q_i. Thus by using real data, the average volume $\bar{V}(j)$ is calibrated for each j, then constant proportional parameter $\theta(j) = \nu(j)/\bar{V}(j)$ is derived by linear interpolation. The interpolated $\theta(j)$'s with respect to j for ask and bid order are shown in Figure 4.3. It is obvious that, the parameter θ_i's also tend to follow gamma distribution rather than power law. Hence, by knowing the parameter vector $\Theta^{ask} = \{\theta^{ask}(j)\}$ and $\Theta^{bid} = \{\theta^{bid}(j)\}$, the cancellation rate at grid i can be given as $\hat{\nu}_i(j) = \theta_i(j)V_i(j)$.

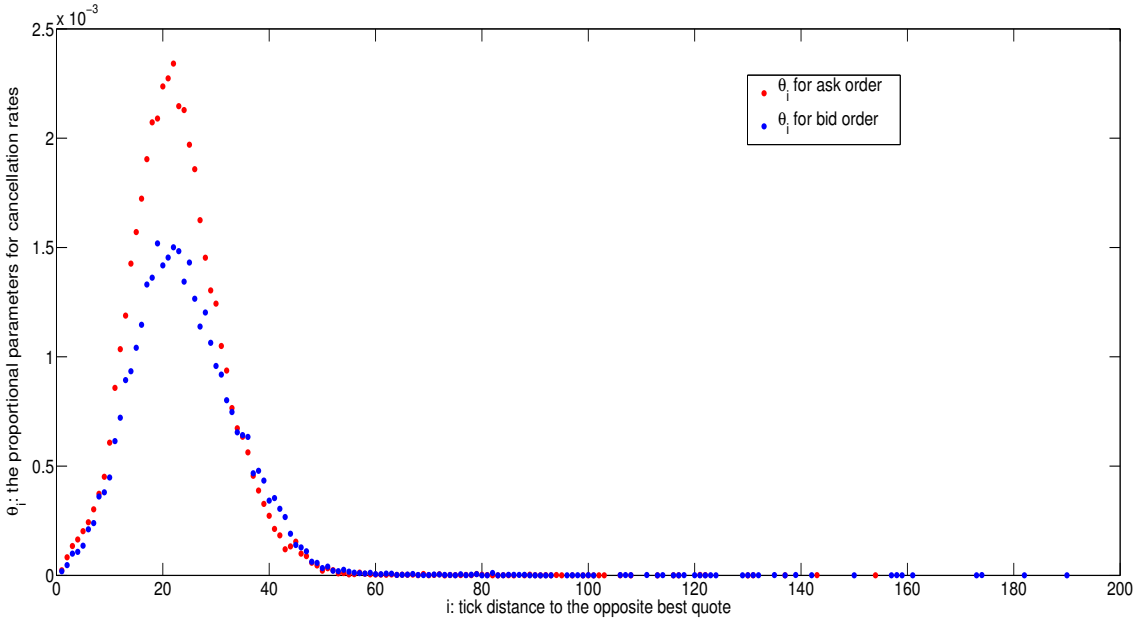

Figure 4.3: Parameter θ_i's that are related to the cancellation rates for ask and bid orders. Calibrated based on the ticker AAPL. The y-axis is the θ_i given in order/second·share. The x-axis denotes the distance i to the opposite best quote measured in tick size. 1 tick = \$0.01.

Particularly, on $q_{i_a^*}$ and $q_{i_b^*}$, another two independent Poisson processes are used to simulate the market order arrival events respectively. To introduce the dynamic of market order arrival, we let the market order arrival rates $\hat{\lambda}_m^{ask}$ and $\hat{\lambda}_m^{bid}$ fluctuate in different simulation window $\delta\hat{t}_p$, $p = 1, 2, ..., \hat{M}$. The average market order arrival rates are calculated for $p = 1, 2, ..., \hat{M}$:

$$\bar{\lambda}_m^{ask}(\delta\hat{t}_p) = N_m^{ask}(\delta\hat{t}_p)/\delta\hat{t}_p \tag{4.3}$$

$$\bar{\lambda}_m^{bid}(\delta\hat{t}_p) = N_m^{bid}(\delta\hat{t}_p)/\delta\hat{t}_p \tag{4.4}$$

$N_m^{ask}(\delta\hat{t}_p)$ and $N_m^{bid}(\delta t_p)$ are the number of market orders happen in $\delta\hat{t}_p$. Then the execution rate fluctuation ranges are given as:

$$\hat{\lambda}_m^{ask} \in [\min_p \bar{\lambda}_m^{ask}(\delta\hat{t}_p), \max_p \bar{\lambda}_m^{ask}(\delta\hat{t}_p)] \tag{4.5}$$

$$\hat{\lambda}_m^{bid} \in [\min_p \bar{\lambda}_m^{bid}(\delta\hat{t}_p), \max_p \bar{\lambda}_m^{bid}(\delta\hat{t}_p)] \tag{4.6}$$

In each new $\delta\hat{t}_p$, $\hat{\lambda}_m^{ask}$ and $\hat{\lambda}_m^{bid}$ will be selected again via uniform distribution on those ranges above. Thus different simulation window $\delta\hat{t}_p$ may have different market order arrival rates. The calibration results from the real data are given in Table 4.5.

Table 4.5: Market order arrival rates calibrated from real data of AAPL. $\hat{\lambda}_m^{ask}$ and $\hat{\lambda}_m^{bid}$ are used to simulate the market ask order and bid order arrival, respectively. $\hat{\lambda}_m^{ask}$ and $\hat{\lambda}_m^{bid}$ follow uniform distribution on the range given in the table and get updated in each new simulation window $\delta\hat{t}_p$.

parameter	$\hat{\lambda}_m^{ask}$	$\hat{\lambda}_m^{bid}$
range (order/s)	$[0.0155, 5.1464]$	$[0.0159, 10.9353]$

On the other hand, we also need to specify the range for the order volume for limit order, cancellation order and market order. Similar to the method of market order arrival rate calibration, we find the minimum and maximum via statistics on the real order book and message book data. Then we let the volume on each new generated event also follow the uniform distribution. To be specifically, we summarize the parameters and the quantities from calibration in Table 4.6.

Table 4.6: Volume fluctuation range for the simulation. The range are calibrated from the real data of AAPL. Each of the parameter represents the volume fluctuation range for a particular type of order. The upper subscript of $\hat{V}$ indicates the ask or bid order, while the lower subscript, given in l, c and m represents limit order, cancellation and market order, respectively. The generated volume follows uniform distribution given by the range in the table.

parameter	$\hat{V}_l^{ask}$	$\hat{V}_l^{bid}$	$\hat{V}_c^{ask}$	$\hat{V}_c^{bid}$	$\hat{V}_m^{ask}$	$\hat{V}_m^{bid}$
range (share)	$[1, 15041]$	$[1, 22444]$	$[1, 3000]$	$[1, 10000]$	$[1, 3290]$	$[1, 2242]$

In the simulation grid q_i, since the generation of volume is randomized by uniform distribution, it could happen that the newly generated execution or cancellation volume may be larger than the existing volume on grid q_i. In this case, we just clear out the volume on grid q_i rather than place a negative volume on the grid. Using $\tilde{V}_m(i)$ and $\tilde{V}_c(i)$ to denote the newly generated market order volume and cancellation volume on grid q_i, $\hat{V}(i)$ to denote the existing volume on grid q_i, the Table 4.7 outlines the situation and the corresponding actions taken by the simulation algorithm. Related to the update of indices for the best ask and bid price grids, i.e. i_a^* or i_b^*, the actions would also cause the update on the distances js then all the intensities related to j.

Table 4.7: Volumes changes and the corresponding actions in simulation taken by algorithms. $\tilde{V}_m(i)$ is the market order volume happens on the grid q_i. $\hat{V}(i)$ is the existing volume size on the grid q_i. The tables outlines the different actions taken by different scenario of $\tilde{V}_m(i)$ and $\hat{V}(i)$.

case	volume change	action
$\tilde{V}_m(i) \geq \hat{V}(i)$	$\hat{V}(i) \leftarrow 0$	update index i_a^* or i_b^*
$\tilde{V}_m(i) < \hat{V}(i)$	$\hat{V}(i) \leftarrow \hat{V}(i) - \tilde{V}_m(i)$	keep index i_a^* or i_b^*
$\tilde{V}_c(i) \geq \hat{V}(i)$	$\hat{V}(i) \leftarrow 0$	update index i_a^* or i_b^*, if $i = i_a^*, i_b^*$
$\tilde{V}_c(i) < \hat{V}(i)$	$\hat{V}(i) \leftarrow \hat{V}(i) - \tilde{V}_c(i)$	keep index i_a^* or i_b^*

At the end of the simulation, all the simulation windows $\delta\hat{t}_p$, $p = 1, 2, ..., \hat{M}$ are aggregated into a complete order book with total duration $\hat{T} = \hat{M}\hat{\delta t}$. To follow the format of the data structure of real data, we truncate the grids according to price level needed. Since there are usually 10 price levels in the real order book data, which means only the best 10 ask prices and best 10 bid prices are shown in the real order book data, thus we keep the 10 grids on ask and bid side of the order book. In this way, a complete order book data with standard format is obtained.

Finally, after the order generation and aggregation, we validate the simulated data by performing the same statistics for the original real data. For some particular parameters, such as the price range, and volume range are expected to be the same as the original data. Ideally, the intensity distribution, such as the order arrival rates and cancellation proportional parameters, should also match the distribution of the original data. In that case, we can verify whether the synthetic data are indeed simulated correctly and the intensities of the real data are being well followed. We have

the order arrival intensity vector calibrated from the real data as $\Lambda = \{\lambda_i\}_{i=1}^{k}$, where λ_i is the arrival rate on the price level that has i ticks distance to the opposite best quote. And the same statistics performed to simulated data yields the arrival intensity vector denoted as $\hat{\Lambda} = \{\hat{\lambda_i}\}_{i=1}^{k}$. We define the mean square error of $\hat{\Lambda}$ over Λ to be MSE $= \frac{1}{k}\sqrt{\sum_{i=1}^{k}(\lambda_i - \hat{\lambda_i})^2}$, where $i = 1, ..., k$. It's obvious that, the less the MSE is, the less the bias for each of the intensities in the simulated data comparing to the original real data. On the other hand, the cosine similarity between $\hat{\Lambda}$ and Λ is defined to be $\frac{\hat{\Lambda}\cdot\Lambda}{\|\hat{\Lambda}\|\|\Lambda\|}$. It is clear that, the larger the cosine value is, the more similarity between the vector $\hat{\Lambda}$ and Λ. When cosine value tends to 0, it means that these two intensity vectors are likely perpendicular to each other, which implies a big difference between the simulated intensities and the original data intensities. In contrast, if the cosine value is close to 1, then it suggests that the two intensity vectors are identical to each other. After the validation phase, the synthetic order book is ready for use.

4.6 Metrics Prediction Based on Built Model

After validation and performance evaluation, the model is put online to provide predictions on unseen data points from real time trading activities. Similar to construction of the training data, each unseen data point is processed and represented with a vector of features, whose dimension is exactly the same as that in the training data. The unseen data point along with its feature vector is fed into the learning model and the latter eventually assigns a label to the data point. For instance, the assigned label could be one of the set $\{upward, downward, stationary\}$ if the direction of mid-price movement is predicted for the given new data point. Obviously, labels tagged by learning models to up-to-date trading events can serve as a signal for traders to make trading decisions. The close relationship between assigned labels by learning model with respect to mid-price as well as price spread crossing and trading decision is depicted in Table 4.8.

The interaction between model-generated prediction on mid-price movement and trading strategy described in Table 4.8 can be better demonstrated with an example. Supposed that at time t, the bid-ask spread S_t is 1 tick and the best ask price is P_t^{ask} with s available shares for the assets, meanwhile, the model predicts an upward movement of the mid-price at $t + \Delta t$ (i.e. $P_{t+\Delta t}^{mid} > P_t^{mid}$

and $\Delta t > 1$). Then, the following trading strategy could enhance the possibility for profit: a market buy order with price P_t^{ask} and s shares should be submitted at time $t+1$. The execution of the purchase results in a higher best ask price and consequently an increasing mid-price, that is, $P_{t+1}^{ask} > P_t^{ask}$ and $P_{t+1}^{mid} = \frac{1}{2}(P_{t+1}^{ask} + P_{t+1}^{bid}) > P_t^{mid}$. After time $t + \Delta t$, the position can exit by selling back the asset when the best bid price is larger than P_t^{ask} or when the best bid price equals to P_t^{bid} depending on whichever occurs first; if it is the former, profit is earned as the asset is sold at a higher-than-bought price, while for the latter, the risk is locked by losing s ticks. The other possible interactions between assigned labels by learning model and trading decision are presented in Table 4.8.

Table 4.8: Example: metrics' labels and corresponding trading decisions. The left side of the arrows indicates the prediction that assigned by the proposed model. The right side of the arrows are the trading strategy according to the prediction given.

Class label		Trading decision
Mid-price direction		
upward	$\longrightarrow$	*long position at* P_t^{ask} *to follow up trending*
downward	$\longrightarrow$	*short position at* P_t^{bid} *to follow down trending*
stationary	$\longrightarrow$	*no move or holding position for entering*
Spread crossing direction		
upward	$\longrightarrow$	*long position at* P_t^{ask} *then short sell at* $P_{t+\Delta t}^{bid}$
downward	$\longrightarrow$	*short position at* P_t^{bid} *then buy back at* $P_{t+\Delta t}^{ask}$
non-crossing	$\longrightarrow$	*no move or holding position for entering*

Similarly, model-delivered labels for spread crossing can also be used to guide trading decisions. if an upward direction of spread crossing is predicted at time t, for instance, then long positions can be taken at the best ask price P_t^{ask}, and sold back at time $t + \Delta t$ with price $P_{t+\Delta t}^{bid}$, which, by prediction, is no less than P_t^{ask}. In short, models trained by machine learning techniques could help traders make better trading decisions should predictions delivered by models are accurate.

CHAPTER 5

EXPERIMENTS AND RESULTS

This section summarizes all the experiments results that used to validate the model proposed in this dissertation. The model efficiency measured by training and prediction time are demonstrated in 5.1. The model performance given by different feature sets and the analysis of feature selection are shown in 5.2. To investigate how the prediction horizon will affect the model performance, experiments of comparisons between different horizons are implemented and the results are shown in 5.3. Moreover, the performance and the training results, given by two different training schemes of multi-class SVMs — one-against-all and one-against-one, are compared and analyzed in 5.4. As a supplementary component for the model validation, synthetic limit order book data generated by simulation are used for model training, the results with cross-validation demonstrate the effectiveness on the simulated data with details in 5.5. Illustrated by the tables of corresponding p-values, the model's performance is further justified by the results of paired t-test in 5.6. Finally, the test of trading strategy built on the model proposed is given in 5.7. With a full trading day length, real data of 5 stocks from NASDAQ are used in the experiments. The prediction time horizon, which specifies how long ahead of current time for the model to project forecasting, is denoted by the notation Δt and measured as the number of events. All experiments are performed based on the system of Linux, with CPU as 2.9GHz Intel Core i7, 8GB 1600MHz memory.

5.1 Overall Performance

Using training data equipped with all features (basic, time-insensitive, time-sensitive) from Table 4.2, the training as well as the prediction time related to the two metrics are outlined in Table 5.1. With the same size of 1500 data points, the training data are constructed by following the distribution of each metrics in the original data. The numbers illustrated are the time that models use for identifying each unseen example. Data in "Training time" and "Testing time" columns are given in seconds and milliseconds respectively. It is clearly from Table 5.1 that, for the same ticker,

the time used to train a model as well as the time for predicting an unseen data point varies by metrics. The differences may due to the number of support vectors of the models trained, which is the main factor relates to computation workload. For instance, the mid-price model of INTC has 171 support vectors while there are 291 for the model of spread crossing, which results in the training and prediction time for mid-price (3.884s and 0.0250ms respectively) are less than those of spread crossing (5.020s and 0.0366ms).

Table 5.1: Models' processing time with different tickers: all features are used for training, training set size = 1500; $\Delta t = 5$ (events)

	Training time(s)		Prediction time(ms)	
Ticker	mid-price	spread crossing	mid-price	spread crossing
MSFT	1.116	2.960	0.0250	0.0180
INTC	3.884	5.020	0.0250	0.0366
AMZN	5.199	5.920	0.0683	0.0210
AAPL	4.246	5.390	0.0567	0.0311
GOOG	3.255	3.640	0.0783	0.0237
Mean	3.540	4.586	0.0507	0.0260

Similarly to the reasons above, given the same metric and training set size, the time used for training and prediction in different tickers are various. Specifically, comparing to 211 support vectors in the spread crossing model of MSFT, 291 support vectors make model training and prediction time of INTC (5.020s and 0.0366ms) are slightly longer than those of MSFT (2.960s and 0.0180ms). In the last row "Mean", the data indicate the efficiency of the models as an overview. Particularly, since the prediction time of AAPL, 0.0311ms, is less than 0.0612ms — the time difference between the upward spread crossing events from the Row $k-1$ to Row $k+4$ in Table 4.1, the model can capture this opportunity and generate corresponding trading decisions for trader.

5.2 Effects and Analysis of Feature Selection

Firstly, we explore the model performance by training the model with different combinations of the features sets. Following the definitions in Table 4.2, four feature set configurations — $\mathcal{F}_0 =$ {basic feature set}, $\mathcal{F}_1 =$ {basic set, time-insensitive set}, $\mathcal{F}_2 =$ {basic set, time-sensitive set} and

$\mathcal{F}_3$ = {basic set, time-insensitive set, time-sensitive set} are shown in Table 5.2 and Table 5.3 for comparisons. By percentage, the first column outlines measurements such as precision (P), recall (R) and F_1-measure (F_1). The second column, denoted by U (up), D (down) and S (stationary), lists the labels for 3 different classes. The columns of $\Delta_{(i,j)}$ denotes the difference between the data in columns "$\mathcal{F}_i$" and "$\mathcal{F}_j$" (latter subtracts former), while the column "Avg." is the average of the last three columns of data in the tables. The mid-price metrics of the ticker AAPL and GOOG are shown in Table 5.2, and the examples of MSFT and AMZN, with the case of spread crossing metrics, are given in Table 5.3.

Table 5.2: Mid-price prediction performance with different feature sets. Label U = Upward, D = Downward, S = Stationary. Training set size = 1500; Δt = 5 (events); $\mathcal{F}_0$ = {Basic features}; $\mathcal{F}_1$ = {Basic and time-insensitive features}; $\mathcal{F}_2$ = {Basic and time-sensitive features}; $\mathcal{F}_3$ = {All features}. $\Delta_{(i,j)}$ = value of column $\mathcal{F}_j$ − value of column $\mathcal{F}_i$. P = Precision, R = Recall, F_1 = F_1-measure. Ticker = AAPL/GOOG

AAPL: Mid-price

	Label	$\mathcal{F}_0$	$\mathcal{F}_1$	$\mathcal{F}_2$	$\mathcal{F}_3$	$\Delta_{(1,2)}$	$\Delta_{(0,1)}$	$\Delta_{(0,2)}$	$\Delta_{(0,3)}$	Avg.
P(%)	U (↑)	83.6	89.5	85.3	84.7	-4.2	5.9	1.7	1.1	2.9
	D (↓)	75.2	88.0	84.9	84.3	-3.1	12.8	9.7	9.1	10.5
	S (–)	79.8	90.0	80.1	81.9	-9.9	10.2	0.3	2.1	4.2
R(%)	U (↑)	76.5	89.5	77.4	76.5	-12.1	13.0	0.9	0.0	4.6
	D (↓)	80.5	88.0	82.9	83.2	-5.1	7.5	2.4	2.7	4.2
	S (–)	81.0	90.0	83.8	82.4	-6.2	9.0	2.8	1.4	4.4
F_1(%)	U (↑)	79.9	89.5	81.4	80.6	-8.1	9.6	1.5	0.7	3.9
	D (↓)	77.8	88.0	83.9	83.4	-4.1	10.2	6.1	5.6	7.3
	S (–)	80.4	90.0	82.0	82.2	-8.0	9.6	1.6	1.8	4.3

GOOG: Mid-price

	Label	$\mathcal{F}_0$	$\mathcal{F}_1$	$\mathcal{F}_2$	$\mathcal{F}_3$	$\Delta_{(1,2)}$	$\Delta_{(0,1)}$	$\Delta_{(0,2)}$	$\Delta_{(0,3)}$	Avg.
P(%)	U (↑)	80.0	85.4	86.7	83.3	1.3	5.4	6.7	3.3	5.1
	D (↓)	86.0	85.7	87.2	84.7	1.5	-0.3	1.2	-1.3	-0.1
	S (–)	79.2	89.4	89.3	85.7	-0.1	10.2	10.1	6.5	8.9
R(%)	U (↑)	82.0	85.0	86.7	82.2	1.7	3.0	4.7	0.2	2.6
	D (↓)	83.0	87.0	87.7	83.5	0.7	4.0	4.7	0.5	3.1
	S (–)	80.0	88.5	89.1	83.1	0.6	8.5	9.1	3.1	6.9
F_1(%)	U (↑)	81.0	85.2	86.7	82.8	1.5	4.2	5.7	1.8	3.9
	D (↓)	84.5	86.4	87.5	84.1	1.1	1.9	3.0	-0.4	1.5
	S (–)	79.6	88.9	89.2	84.0	0.3	9.3	9.6	4.4	7.8

Following the definitions in Table 4.2, four feature set configurations — $\mathcal{F}_0$ = {basic feature set}, $\mathcal{F}_1$ = {basic set, time-insensitive set}, $\mathcal{F}_2$ = {basic set, time-sensitive set} and $\mathcal{F}_3$ = {basic

set, time-insensitive set, time-sensitive set} are shown in Table 5.2 and Table 5.3 for comparisons. By percentage, the first column outlines measurements such as precision (P), recall (R) and F_1-measure (F_1). The second column, denoted by U (up), D (down) and S (stationary), lists the labels for 3 different classes. The columns of $\Delta_{(i,j)}$ denotes the difference between the data in columns "$\mathcal{F}_i$" and "$\mathcal{F}_j$" (latter subtracts former), while the column "Avg." is the average of the last three columns of data in the tables. The mid-price metrics of the ticker AAPL and GOOG are shown in Table 5.2, and the examples of MSFT and AMZN, with the case of spread crossing metrics, are given in Table 5.3.

Table 5.3: Spread crossing model performance. Label U = Upward, D = Downward, S = Stationary. Training set size = 1500; $\mathcal{F}_0$ = {Basic features}; $\mathcal{F}_1$ = {Basic and time-insensitive features}; $\mathcal{F}_2$ = {Basic and time-sensitive features}; $\mathcal{F}_3$ = {All features}. $\Delta_{(i,j)}$ = value of column $\mathcal{F}_j$ − value of column $\mathcal{F}_i$. P = Precision, R = Recall, F_1 = F_1-measure. Δt = 5 (events). Ticker = MSFT/AMZN

MSFT: Spread Crossing

	Label	$\mathcal{F}_0$	$\mathcal{F}_1$	$\mathcal{F}_2$	$\mathcal{F}_3$	$\Delta_{(1,2)}$	$\Delta_{(0,1)}$	$\Delta_{(0,2)}$	$\Delta_{(0,3)}$	Avg.
P(%)	U (↑)	88.7	93.3	89.4	83.0	-3.9	4.6	0.7	-5.7	-0.1
	D (↓)	87.9	95.0	89.7	91.4	-5.3	7.1	1.8	3.5	4.1
	S (–)	84.6	91.5	94.8	92.2	3.3	6.9	10.2	7.6	8.2
R(%)	U (↑)	90.3	95.9	94.6	93.0	-1.3	5.6	4.3	2.7	4.2
	D (↓)	92.1	94.7	92.7	94.7	-2.0	2.6	0.6	2.6	1.9
	S (–)	79.4	88.9	75.7	73.3	-13.2	9.5	-3.7	-6.1	-0.1
F_1(%)	U (↑)	89.5	94.6	92.0	88.0	-2.6	5.1	2.5	-1.5	2.0
	D (↓)	90.0	94.8	91.2	93.1	-3.6	4.8	1.2	3.1	3.0
	S (–)	82.0	90.2	85.3	82.8	-4.9	8.2	3.3	0.8	4.1

AMZN: Spread Crossing

	Label	$\mathcal{F}_0$	$\mathcal{F}_1$	$\mathcal{F}_2$	$\mathcal{F}_3$	$\Delta_{(1,2)}$	$\Delta_{(0,1)}$	$\Delta_{(0,2)}$	$\Delta_{(0,3)}$	Avg.
P(%)	U (↑)	72.5	79.7	76.3	77.4	-3.4	7.2	3.8	4.9	5.3
	D (↓)	72.7	72.8	71.6	73.0	-1.2	0.1	-1.1	0.3	-0.2
	S (–)	65.4	76.2	74.3	70.5	-1.9	10.8	8.9	5.1	8.3
R(%)	U (↑)	77.3	85.4	80.1	80.9	-5.3	8.1	2.8	3.6	4.8
	D (↓)	84.2	84.9	80.3	79.6	-4.6	0.7	-3.9	-4.6	-2.6
	S (–)	65.6	76.4	74.1	73.9	-2.3	10.8	8.5	8.3	9.2
F_1(%)	U (↑)	74.8	82.4	78.2	79.2	-4.2	7.6	3.4	4.4	5.1
	D (↓)	78.0	78.4	80.0	76.3	1.6	0.4	2.0	-1.7	0.2
	S (–)	65.5	76.3	74.2	72.2	-2.1	10.8	8.7	6.7	8.7

Insights can be drawn from the Table 5.2: Firstly, the training data with extended feature sets generally outperform the basic feature set. For example, shown by the data in the column

"Avg.", when extra features added to basic set, the precision, recall, and the F_1-measure of the up class of AAPL are all increased by 2.9%, 4.6% and 3.9% respectively. Secondly, depending on the feature set selections, the performance of the models may vary. Illustrated by the signs of column "$\Delta_{(1,2)}$", the model performance of AAPL is better when choosing $\mathcal{F}_1$ rather than other feature sets, however, for the ticker GOOG, the model trained with $\mathcal{F}_2$ demonstrates a better performance among all the models. This suggests the mid-price movements of GOOG have higher dependency on the time-sensitive features. Thirdly, model performance is unnecessarily positively correlated with the number of features. Redundant dimensions in the feature vector could produce noises and negative interferences that lead to learning bias and undermine the performance. Hence it could be part of the reason why the performance of set $\mathcal{F}_3$ are lower than those of either $\mathcal{F}_1$ or $\mathcal{F}_2$.

Table 5.4: Model performance summary. Feature set = {Basic and time-insensitive features}; P = Precision; R = Recall; F_1 = F_1-measure; F_2 = F_2-measure; $F_{\frac{1}{2}}$ = $F_{\frac{1}{2}}$-measure. Label U = Upward, D = Downward, S = Stationary. Training size = 1500.

Ticker	Label	P(%)	R(%)	F_1(%)	F_2(%)	$F_{\frac{1}{2}}$(%)
MSFT	U (↑)	93.3	95.9	94.6	95.4	93.8
	D (↓)	95.0	94.7	94.8	94.8	94.9
	S (–)	91.5	88.9	90.2	89.4	90.9
INTC	U (↑)	91.9	95.1	93.5	94.4	92.5
	D (↓)	93.9	97.6	95.7	96.8	94.6
	S (–)	92.9	83.7	88.1	85.4	90.9
AMZN	U (↑)	79.7	85.4	82.4	84.2	80.8
	D (↓)	72.8	84.9	78.4	82.2	74.9
	S (–)	76.2	76.4	76.3	76.4	76.2
AAPL	U (↑)	79.9	75.7	77.7	76.5	79.0
	D (↓)	80.1	83.4	81.7	82.7	80.7
	S (–)	94.1	95.0	94.5	94.8	94.3
GOOG	U (↑)	76.0	78.5	77.2	78.0	76.5
	D (↓)	70.1	63.6	66.3	64.8	68.7
	S (–)	88.9	72.4	70.5	75.2	85.0
Avg.	U (↑)	84.2	86.1	85.1	85.7	84.5
	D (↓)	82.4	84.5	83.4	84.3	82.8
	S (–)	88.7	83.3	83.9	84.2	87.5

On the other hand, the comparisons in regard to spread crossing are illustrated in Table 5.3. Similarly, the models trained with extended feature sets $\mathcal{F}_1$, $\mathcal{F}_2$ and $\mathcal{F}_3$ outperform those who are

using basic feature set $\mathcal{F}_0$ only. However, different from the results of Table 5.2, both of MSFT and AMZN reach the best performance when the time-insensitive features are added into the training data set. It can be clearly seen from the columns "$\mathcal{F}_0$", "$\mathcal{F}_1$" and "$\Delta_{(0,1)}$" that, the precision of upward prediction is increased from 88.7% to 93.3%. The recall and F_1-measure are also increased by 5.6% and 5.1% in the same case. The situations for the ticker AMZN are quite similar to those of MSFT. Thus, suggested by the results, the spread crossing metrics show a higher sensitivity to the time-insensitive features other than time-sensitive features.

As a summary, Table 5.4 reports the performance of spread crossing models trained for five different tickers. The training data are built in the of size = 1500 data points and trained with the polynomial kernel $\kappa(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j} + 1)^z$ for $z = 2$. Again, as shown by the column "Label", the data of metric — spread crossing are classified into three categories according to the directions. For each category, measurements as the precision, recall as well as F_β-measures by taking $\beta = 1$, 2, $\frac{1}{2}$ — F_1-measure, F_2-measure and $F_{\frac{1}{2}}$-measure are outlined as the columns of "P", "R" , "F_1", "F_2" and "$F_{\frac{1}{2}}$" respectively. It can be observed from the Table 5.4 that, all of the measurements are beyond 80% except for the spread crossing of AMZN and GOOG, which are also above 75%. Hence, for spread crossing metric, with time-insensitive features added, $\mathcal{F}_1$ shows a promising performance for building learning models. All of the above from the results of Table 5.4 demonstrate the effectiveness of the proposed framework.

Table 5.5: Performance measurement before and after feature selection: original set is the training set before feature selection, and economical set is the training set after feature selection. Original training set = {All features}. Feature number in original set = 82. Training set size = 2000; Δt = 5 (events)

		Original Set					Economical Set				
Ticker	Label	P(%)	R(%)	F_1(%)	F_2(%)	$F_{\frac{1}{2}}$(%)	P(%)	R(%)	F_1(%)	F_2(%)	$F_{\frac{1}{2}}$(%)
AAPL	U (↑)	77.8	74.7	76.2	75.3	77.2	75.1	74.2	74.6	74.4	74.9
	D (↓)	80.4	83.3	81.8	82.7	81.0	80.5	82.3	81.4	81.9	80.9
	S (–)	99.1	98.8	99.0	98.8	99.0	99.3	98.6	99.0	98.7	99.2
GOOG	U (↑)	85.4	86.8	86.1	86.5	85.7	85.5	82.6	84.0	83.2	84.9
	D (↓)	83.0	79.3	81.2	80.0	82.2	79.5	80.5	80.0	80.3	79.7
	S (–)	98.6	99.5	99.0	99.3	98.8	98.6	99.5	99.0	99.3	98.8

After comparing the performance by using different combination of feature sets, we further examine the model in the perspective of feature selection. To perform feature selection, we use the the ranking of information gain of each attributes. By selecting the attributes with information gain despondingly, we eventually determine an economical set, on which the model trained maintains the stable performance as trained by using the original feature set.

Table 5.6: Complete list of the selected attributes for the economical sets. a_l^k denotes the k-th attribute from the vector v_l in Table 4.2, $l = 1, 2, ..., 9$.

Ticker	List of Attributes a_l^k in Economical Set (Mid-price)
AAPL	a_1^4, a_1^2, a_2^3, a_2^{15}, a_2^4, a_2^2, a_2^8, a_1^{32}, a_2^{13}, a_7^1, a_9^3, a_1^{14}, a_1^{12}, a_1^{34}, a_1^{28}, a_9^4, a_1^{10}, a_1^{40}, a_1^8, a_1^{24} a_1^{18}, a_1^5, a_1^{13}, a_1^9, a_1^{25}, a_1^{33}, a_4^2, a_1^{17}, a_1^{21}, a_1^1, a_2^{20}, a_3^2, a_8^2, a_6^2, a_6^4, a_9^2, a_5^2, a_2^7, a_2^{14}, a_1^{39}
GOOG	a_2^{20}, a_2^8, a_4^1, a_1^{10}, a_2^1, a_1^5, a_7^6, a_7^4, a_7^2, a_4^4, a_1^2, a_2^{13}, a_2^{15}, a_9^4, a_1^{28}, a_1^{27}, a_1^{30}, a_1^{25}, a_5^1, a_1^{29} a_1^{31}, a_2^7, a_7^3, a_6^3, a_6^1, a_8^1, a_1^{24}, a_8^4, a_7^1, a_7^5, a_3^1, a_2^{19}, a_1^{20}, a_1^{40}, a_1^8, a_1^6, a_4^2, a_1^{16}

Ticker	List of Attributes a_l^k in Economical Set (Spread Crossing)
AAPL	a_2^{20}, a_1^2, a_4^3, a_4^1, a_6^1, a_2^1, a_6^3, a_2^2, a_9^1, a_9^4, a_7^1, a_1^{29}, a_1^9, a_1^{13}, a_1^{25}, a_1^{37}, a_1^5, a_1^1, a_6^2, a_9^2, a_2^{14} a_1^{17}, a_1^{33}, a_7^7, a_2^{20}, a_7^6, a_1^{21}, a_5^2, a_3^2, a_1^{31}, a_1^{35}, a_1^{39}, a_7^4, a_7^2, a_8^1, a_8^2, a_6^4, a_9^3, a_1^{27}, a_1^{11}, a_1^{19} a_1^{23}, a_1^{15}, a_1^3, a_1^7, a_8^4, a_2^{11}, a_2^8, a_1^{20}, a_2^{10}, a_2^{15}, a_2^9, a_1^6, a_2^3, a_2^{13}, a_1^4, a_1^{34}, a_2^4, a_2^5, a_1^{30}
GOOG	a_1^2, a_1^4, a_2^2, a_2^8, a_2^1, a_2^4, a_7^7, a_2^{14}, a_9^2, a_2^{15}, a_2^{19}, a_2^9, a_3^1, a_1^9, a_2^{17}, a_1^5, a_7^6, a_7^4, a_7^2, a_3^2, a_6^2 a_1^1, a_1^{13}, a_2^1, a_1^{19}, a_5^2, a_1^{27}, a_1^{15}, a_5^1, a_8^2, a_1^3, a_9^1, a_1^{39}, a_9^3, a_1^7, a_6^4, a_2^3, a_2^{15}, a_2^{13}, a_1^6, a_2^8 a_2^{16}, a_1^{22}, a_4^1, a_1^{35}, a_1^{23}, a_2^{11}, a_2^{10}, a_1^{11}, a_1^{17}, a_4^3, a_1^{29}, a_1^{21}, a_2^{20}, a_9^4, a_6^1

By using F_1-mearure as the target measurement, we define the economical feature set as the feature set with minimal number of the feature attributes (dimension of feature vector), but retains greater or equals to 95% of F_1-measure of original full feature set for each labelled class. According to the ranking of information gain and the threshold as 95% of original F_1-measure, the performance before and after performing feature selection are shown in the Table 5.5. 38 and 40 attributes in the economical sets are selected from 86 attributes in the original sets for GOOG and AAPL for the mid-price prediction, respectively. As in spread crossing, 53 and 60 attributes are selected from original attributes for GOOG and AAPL, respectively. The complete list of the selected attributes for the economical sets are outlined in Table 5.6.

Table 5.7: Examples: features selected with top 10 information gain along with the feature type and descriptions. Upper table metrics = spread crossing; Lower table metrics = mid-price; Basic = basic feature; T-insen = time-insensitive feature; T-sen = time-sensitive feature; $\Delta t = 5$ (events), Ticker = AAPL.

Spread crossing

Rank	Info Gain	Type	Description
1	0.4955	T-insen	*mid-price at level 10*
2	0.3883	Basic	*volume at best ask*
3	0.3875	T-insen	*average volume on ask side*
4	0.3793	T-insen	*average price on ask side*
5	0.3663	T-sen	*derivative of best ask price*
6	0.3662	T-insen	*bid-ask spread at best ask and bid*
7	0.3531	T-sen	*derivative of best ask volume*
8	0.3512	T-insen	*mid-price at best ask and bid*
9	0.3471	T-sen	*acceleration of market ask order*
10	0.3405	T-sen	*acceleration of limit ask order*
...	...	...	...

Mid-price

Rank	Info Gain	Type	Description
1	0.4465	Basic	*volume at best bid*
2	0.3873	Basic	*volume at best ask*
3	0.3703	T-insen	*bid-ask spread at level 4*
4	0.3668	T-insen	*bid-ask spread at level 8*
5	0.3244	T-insen	*mid-price at level 2*
6	0.3138	T-insen	*mid-price at best ask and bid*
7	0.3129	T-insen	*mid-price at level 4*
8	0.3039	Basic	*ask price at level 9*
9	0.2995	T-insen	*bid-ask spread at level 7*
10	0.2805	T-sen	*average intensity of limit ask order*
...	...	...	...

The Table 5.7 and 5.8 outline the top 10 features given by the information gain ranking for the ticker AAPL and GOOG. We can see that in the case of AAPL, there is only one basic feature listed in the first 10 information gain ranking while the time-insensitive features take the majority of the kind. The situation is similar for the GOOG. On the other hand, we can also see that the volume values at the best bid and ask level are often selected as time-insensitive feature for both mid-price and spread crossing prediction.

Table 5.8: Examples: features selected with top 10 information gain along with the feature type and descriptions. Upper table metrics = spread crossing; Lower table metrics = mid-price; Basic = basic feature; T-insen = time-insensitive feature; T-sen = time-sensitive feature; $\Delta t = 5$ (events), Ticker = GOOG.

Spread crossing

Rank	Info Gain	Type	Description
1	0.3711	Basic	*volume at best ask*
2	0.3186	Basic	*volume at best bid*
3	0.2512	T-insen	*mid-price at best ask and bid*
4	0.2490	T-insen	*mid-price at level 4*
5	0.2098	T-insen	*bid-ask spread at best ask and bid*
6	0.2008	T-insen	*mid-price at level 2*
7	0.1957	T-insen	*bid-ask spread at level 4*
8	0.1916	T-insen	*mid-price at level 7*
9	0.1675	T-sen	*acceleration of limit bid order arrival*
10	0.1639	T-insen	*bid-ask spread at level 8*
...	...	...	...

Mid-price

Rank	Info Gain	Type	Description
1	0.3356	T-insen	*mid-price at level 10*
2	0.2944	T-insen	*mid-price at level 4*
3	0.2032	T-insen	*average price on ask side*
4	0.1907	Basic	*ask price at level 3*
5	0.1871	T-insen	*bid-ask spread at best ask and bid*
6	0.1593	Basic	*ask price at level 2*
7	0.1568	T-sen	*average intensity of bid cancellation*
8	0.1566	T-sen	*average intensity of market bid order*
9	0.1565	T-sen	*average intensity of limit bid order*
10	0.1559	T-insen	*average volume at bid side*
...	...	...	...

Other crucial features are the mid-price and the bid-ask spread on the best ask and bid level. For instance, both of them are selected as the top 10 features in the spread crossing prediction for AAPL and GOOG. For the time-sensitive features, we can see that 2 out of total 4 time-sensitive features selected for AAPL and the only one time-sensitive feature selected for GOOG are related to the acceleration of orders. On the other hand, the mid-price prediction tends to favor the features related to average intensity of orders — all the time-sensitive features selected in the top 10 table belongs to this kind. The overall distribution statistical results are detailed in the Figure 5.1 and 5.2.

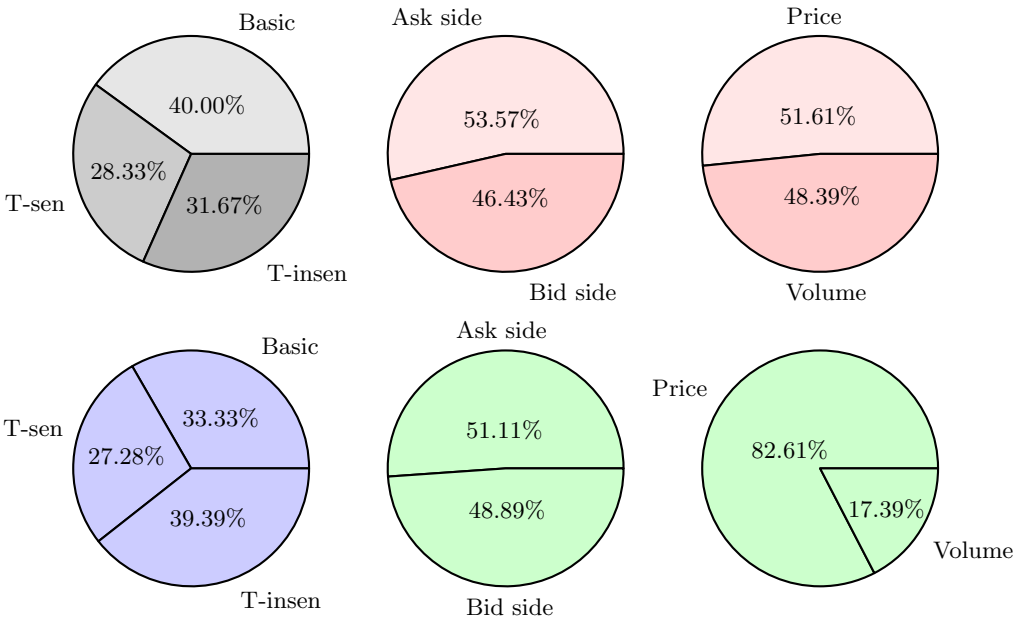

Figure 5.1: Economical training data set feature distribution selected by information gain. The upper pie charts (grey and red) are the distribution of mid-price prediction features, and the lower pie charts (blue and green) is the distribution of spread-crossing prediction features. Basic = basic feature set; T-insen = time-insensitive feature set; T-sen = time-sensitive feature set; Ask side = features only related to ask side; Bid side = features only related to bid side; Price = features only related to price; Volume = features only related to volume. Ticker = AAPL.

The Figure 5.1 and 5.2 clearly show that the distribution of features in the chosen economical set, according to specified principles for the ticker AAPL and GOOG. The upper half of the figures refers to the mid-price economical set, and the lower half of the graph is the case of spread crossing. For both metrics, the distribution by features categories — basic, time-insensitive, time-sensitive; by features related to the ask side or bid side quantities; by features related to price or volume information quantities of the Table 4.2 are interpreted by the pie charts. In Figure 5.1, it is obvious that the basic features are the majority of the mid-price prediction economical features set (40%), while the time-insensitive features dominate the spread crossing economical set by 39.39%. This might suggest that, for AAPL, the time-insensitive features are important for spread crossing metrics just as the basic features to mid-price metrics. On the other hand, features related to ask side and bid side are quite in balance for both mid-price and spread crossing. However, it is quite the opposite for the price and volume related feature distribution in spread crossing case. As shown in the figure, the price related feature highly dominates the economical set (82.61% vs. 17.39%).

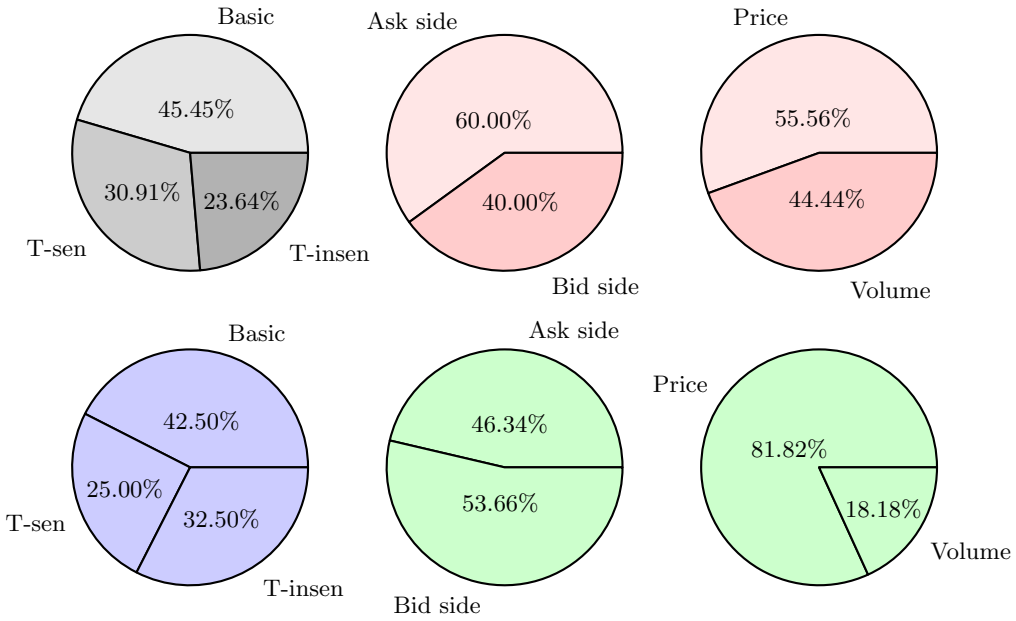

Figure 5.2: Economical training data set feature distribution selected by information gain. The upper pie charts (grey and red) are the distribution of mid-price prediction features, and the lower pie charts (blue and green) is the distribution of spread-crossing prediction features. Basic = basic feature set; T-insen = time-insensitive feature set; T-sen = time-sensitive feature set; Ask side = features only related to ask side; Bid side = features only related to bid side; Price = features only related to price; Volume = features only related to volume. Ticker = GOOG.

In Figure 5.2, the situation of GOOG is similar to that of AAPL in Figure 5.1. Although basic features are still the majority in the economical set of mid-price prediction, the time-sensitive features, take more percentages than time-insensitive features. Hence for GOOG, the time-sensitive feature may indicate more differentiable ability than time-insensitive features for mid-price prediction. This result also match the observation in the Table 5.2. On the other hand, we can see from the pie charts in the middle columns that, the mid-price metrics are more sensitive to features related to ask side while the spread crossing metrics choose more bid-side-related features. Also similar to the AAPL, features involve with price information are the absolute majority in the spread crossing economical set (81.82% vs. 18.18%). This and the case of AAPL conclude that spread crossing metrics rely more on price related features rather then the volume features.

5.3 Impact of Prediction Time Horizons

The time horizon in the prediction problem plays an important role as defining how long we project the forecasting into the future. Intuitively, as the prediction horizon goes longer, more and more factors may affect the prediction results and the accuracy of the prediction may also vary and depend on more uncontrollable parameters. Moreover, the happening frequency of targeted metrics that we try to predict also rely on the implied time horizon as well. In sum, with different time horizons, the performance of predictions may change.

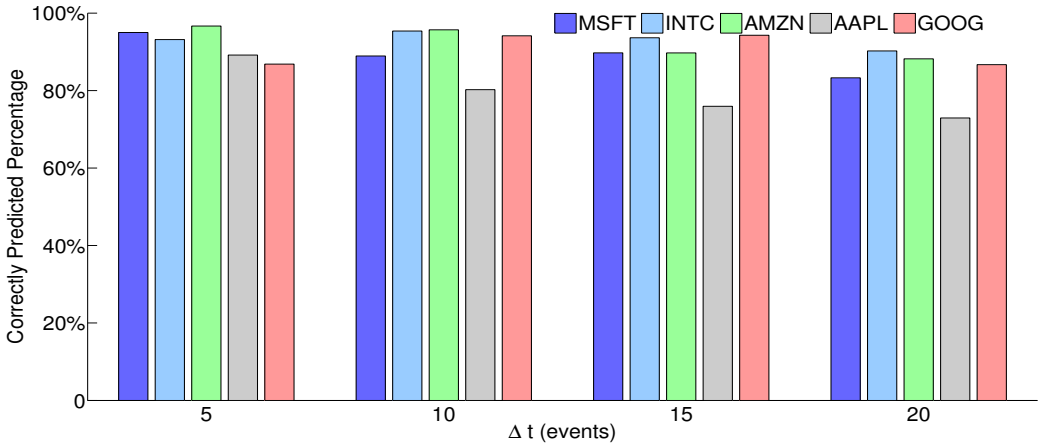

Figure 5.3: Correctly predict percentage in different prediction horizons. Training data set size = 1200, metric = mid-price; Δt = 5, 10, 15, 20 (events)

Using correctly predicted percentage, the metrics are tested on four different time horizons Δt, i.e. 5 events, 10 events, 15 events and 20 events ahead of current time. Figure 5.3 and Figure 5.4 demonstrate the correctly predicted percentages for mid-price predictions and spread crossing predictions respectively, by using feature set $\mathcal{F}_3$ with the size = 1200. Obviously, the correctly predicted percentage for each ticker fluctuates with the time horizon Δt — for the same ticker, the ratio is decreasing as Δt increases. This can be observed explicitly using the case of mid-price metric for AAPL in Figure 5.3 and the spread crossing metric for AMZN in Figure 5.4. One of the reasons behind this should be related to the high volatile characteristics of the asset trading activities, which makes the predictions over a long term are generally harder than those of the short term. However, one can also observe that, some time horizons, for example, the spread crossing metric for INTC at $\Delta t = 10$ and 15 obtain higher percentages than the other time horizons, which suggests

that, for INTC, the spread crossings happen at a frequency with $\Delta t = 10$ or 15 events more possibly than the others, and also that, the features proposed are more suitable for capturing the metrics in such a frequency. Thus choosing a proper prediction horizon for a particular asset may improve the performance thus to achieve better prediction results.

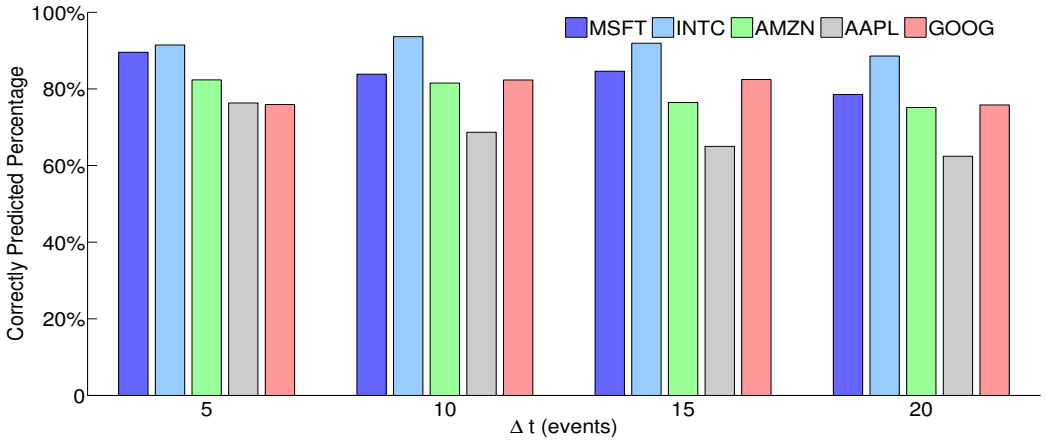

Figure 5.4: Correctly predict percentage in different prediction horizons. Training data set size = 1200, metric = spread crossing; Δt = 5, 10, 15, 20 (events)

5.4 One-against-all vs. One-against-one

Since model training is based on two different methods, namely one-against-all and one-against-one scheme, to investigate the performance difference between these two methods, the same training set is used in the experiment and trained by the two methods separately.

Table 5.9: Models' processing time with different tickers: all features are used for training, Metrics = spread crossing; Training set size = 1000; Δt = 5 (events)

	Training time(s)		Prediction time(ms)	
Ticker	1vs.all	1vs.1	1vs.all	1vs.1
MSFT	5.162	2.640	0.0103	0.0156
INTC	6.940	4.320	0.0089	0.0129
AMZN	6.037	4.800	0.0079	0.0157
AAPL	3.950	3.750	0.0096	0.0190
GOOG	4.090	3.170	0.0093	0.0171
Mean	5.236	3.736	0.0092	0.0161

In terms of traing time and prediction time, the Table 5.9 shows the model performance with respect to the two methods. On all 5 different tickers' training models, the results show that one-against-all scheme requires more time for building the model, but it also has advantage on the prediction speed. The reason of this may due to the model structure built by these two schemes. For one-against-all scheme, each of the k classes and the rest $(k-1)$ classes are separated by k hyperplanes, but in one-against-one scheme, hyperplane is constructed for each pair of classes. Hence, the longer training time for the one-against-all method may result from a larger size on the set of rest $(k-1)$ classes. But regarding to the prediction time, the situation is different — the one-against-all scheme only needs to perform k comparison, while one-against-one scheme needs $k(k-1)/2$. Thus the one-against-all scheme can perform the prediction slightly faster.

Table 5.10: Performance measurements comparison: one-against-all method and one-against-one method. Training feature set = {All features}, P = Precision, R = Recall, F_1 = F_1-measure. Label U = Upward, D = Downward, S = Stationary. 1vs.all = one-against all method; 1vs.1 = one-against-one method; Metric = spread crossing; Training size = 1000; $\Delta t = 5$ (events)

		Precision(%)		Recall(%)		F_1-Measure(%)	
Ticker	Label	1vs.all	1vs.1	1vs.all	1vs.1	1vs.all	1vs.1
MSFT	U (↑)	86.4	88.1	93.8	94.7	89.9	91.3
	D (↓)	96.7	97.9	95.5	94.2	96.1	96.0
	S (–)	89.8	89.3	83.1	86.0	86.3	87.6
INTC	U (↑)	93.3	83.8	96.1	94.6	94.7	88.9
	D (↓)	93.8	93.5	89.7	85.2	91.7	89.2
	S (–)	88.8	88.7	90.1	85.2	89.5	86.9
AMZN	U (↑)	69.1	64.8	80.3	81.8	74.3	72.3
	D (↓)	75.3	81.9	87.8	84.9	81.1	83.4
	S (–)	79.1	86.4	77.9	71.8	78.0	78.4
AAPL	U (↑)	67.6	64.9	70.4	72.4	69.0	68.4
	D (↓)	68.6	68.0	70.4	71.6	69.5	69.8
	S (–)	68.2	69.1	63.7	57.4	65.9	62.7
GOOG	U (↑)	69.0	60.4	71.9	81.8	70.4	69.5
	D (↓)	56.8	62.9	72.4	70.1	63.6	66.3
	S (–)	78.2	82.6	67.3	61.5	72.4	70.5

The training performance are reported in the Table 5.10. For each ticker, the training data sets with the size of 1000 are used in the training. The performance measurements as precision, recall and F_1-measure for the two training methods are reported separately. From the table, once can see

that the performance on one-against-one and one-against-all methods don't show any significant difference. However, for some particular measurement, the performance on different stock does vary. For instance, MSFT has 2% higher F_1-measure in the upward label class when training in the one-against-one method than in the one-against-all method. But one-against-all method outperforms the one-against-one method by 4% on average for the case of F_1-measure for INTC.

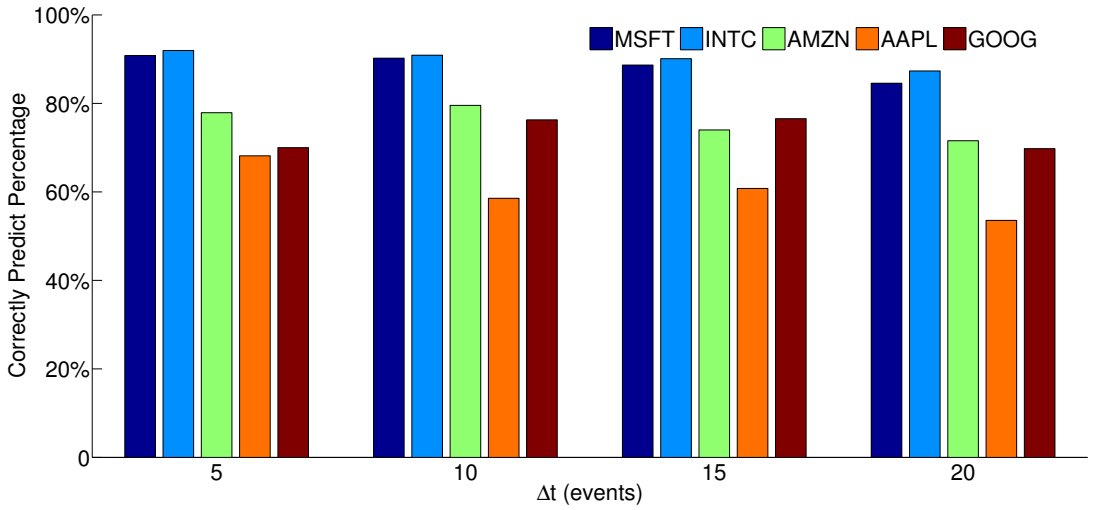

Figure 5.5: Correctly predicted percentage by using one-against-all training method. Training data set size = 1000, metric = spread crossing; Δt = 5, 10, 15, 20 (events)

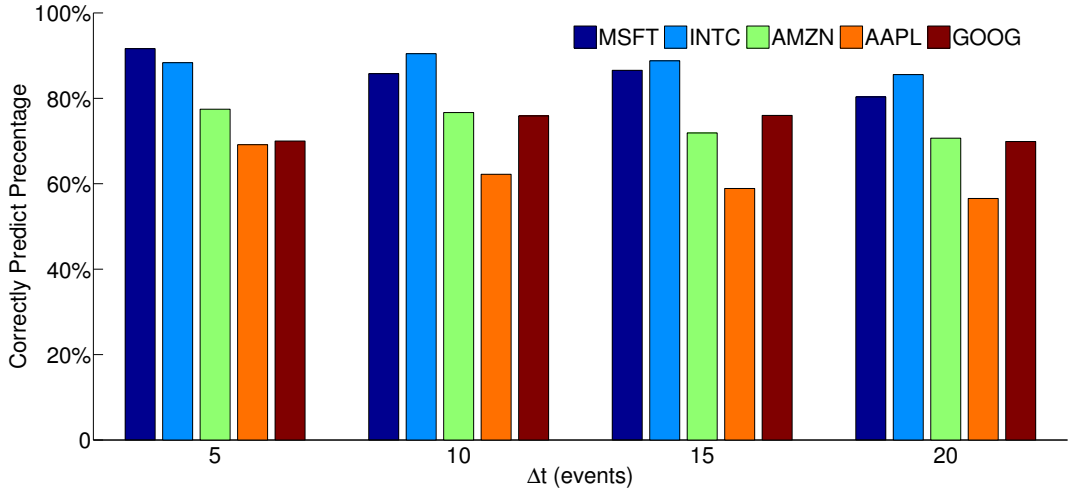

Figure 5.6: Correctly predicted percentage by using one-against-one training method. Training data set size = 1000, metric = spread crossing; Δt = 5, 10, 15, 20 (events)

Figure 5.5 and Figure 5.6 show the prediction accuracy of the training model given as the correctly predicted percentage under one-against-all and one-against-one schemes with different prediction horizons. The graphs illustrate that, under the same method, there are slight differences on prediction accuracy for every ticker. On the other hand, for the same ticker and time horizon, the accuracies over the two training schemes keep stable. Overall, for MSFT and INTC, both of one-against-all scheme and one-against-one scheme predict 90% of the data point correctly. MSFT with prediction horizons equal to 5 or 10 receives better accuracy than others, and INTC gets the best accuracy when the prediction horizons equal to 10 and 15. AMZN and AAPL, especially for AAPL, the accuracy gradually decreases when the prediction horizon becomes longer, this might due to the high volatile on the trading day for the stocks, which makes the prediction over a longer horizon less accurate than a short horizon. With most of the accuracies beyond 70% on the time horizon 5, 10 and 15 for all tickers, both training schemes demonstrate the effectiveness of the learning models.

5.5 Model Performance with Synthetic Data

Table 5.11: Order book simulation data profile based on the real data of AAPL. The arrival rates are in orders/second. The price is in dollars ($) and the volume is in shares. The intervals give the ranges for the parameters.

AAPL

Parameters	Range	Mean	Std.Dev
Market Ask Order Arrival Rate	[0.0155, 5.1464]	1	1
Market Bid Order Arrival Rate	[0.0159, 10.9353]	1	1
Limit Ask Order Book Price	[577.5, 589.65]	583.35	2.87
Limit Bid Order Book Price	[577.33, 588.15]	582.95	2.79
Market Ask Order Volume	[1, 3290]	1575	647.23
Market Bid Order Volume	[1, 2242]	1605	855.14
Limit Ask Order Volume	[1, 15041]	286	150.33
Limit Bid Order Volume	[1, 22444]	449	183.71
Ask Order Cancellation Volume	[1, 3000]	1403	732.53
Bid Order Cancellation Volume	[1, 10000]	5446	2109.01

Synthetic data built by limit order book simulation are used to validate the effectiveness of model proposed. Simulation model and parameters used are based on the statistical method intro-

duced in the section 4.5. As a representative, Table 5.11 shows the parameters for simulation which are obtained from the real order book data of AAPL. By using the parameters in Table 5.11 and the set up in Table 4.3 and 4.5, we simulate order book data with the parameters extracted from the real order book data of AAPL. The Table 5.12 demonstrates the validation results for the simulated data. Low MSE values with magnitude within 10^{-2} and the cosines close to 1 for all the parameter vectors validate the correctness of the simulated data and the intensities are being well followed.

Table 5.12: Synthetic data validation results based on the simulation profile given in the Table 5.11. Original real data ticker = AAPL. Λ_{ask} and Λ_{bid} are the intensity vectors for the limit ask order and limit bid order respectively. Θ_{ask} and Θ_{bid} are the proportional parameter vectors for cancellation rate of ask order and bid order, respectively. MSE denotes for the mean square error and Cosine is the cosine similarity which are described in the section 4.5.

Parameter Vector	MSE (10^{-2})	Cosine
Λ_{ask}	1.1589	0.999892
Λ_{bid}	1.4125	0.999672
Θ_{ask}	0.6577	0.996197
Θ_{bid}	0.4084	0.997044

Table 5.13 demonstrates the model performance on mid-price and spread crossing prediction with synthetic data. In the experiments, 10 data sets given by independently generated synthetic data with size of 2000 data points are trained and then tested on corresponding testing data with size of 2000 data point. The training data sets are constructed by following the class distributions of the synthetic data. Table 5.13 reports the average performance measurements of the 10 trials of experiments. The left subtable outlines the model performance for spread crossing and the right subtable is for mid-price prediction. The prediction horizons Δt are 5, 10 and 20, respectively .

In Table 5.13, It is clearly illustrated that, by precision, recall and F_1-measure, all the performance measurements are above 80% and 75% for spread crossing and mid-price prediction respectively, which implies the models trained by using the synthetic data retain the high performance as models developed from real data. On the other hand, the descending of performance due to a longer prediction horizon — the phenomenon on the metric of predicting real data, also occurs on the synthetic-data-based models. Both of these further validate and justify that, the methodology

of multi-class SVMs modeling actually captures the intrinsic characteristics of the limit order book dynamics in question.

Table 5.13: Performance of the models trained by synthetic data. The synthetic data are generated from the simulation configured by the parameters given in Table 5.11. Δt is the prediction horizon measured by number of events. Training feature set = {All features}, P = Precision, R = Recall, F_1 = F_1-measure. Label U = Upward, D = Downward, S = Stationary. Training data set size = 2000.

Spread Crossing

Δt	Label	P(%)	R(%)	F_1(%)
5	U (↑)	95.1	91.2	93.1
	D (↓)	93.8	92.6	93.2
	S (–)	95.4	95.8	95.6
10	U (↑)	92.3	89.1	90.7
	D (↓)	91.0	90.1	90.5
	S (–)	93.9	96.8	95.3
20	U (↑)	85.7	76.3	80.7
	D (↓)	87.5	88.2	87.8
	S (–)	96.5	97.3	96.9

Mid-price

Δt	Label	P(%)	R(%)	F_1(%)
5	U (↑)	87.2	89.1	88.1
	D (↓)	87.9	90.0	88.9
	S (–)	85.1	82.3	83.7
10	U (↑)	85.5	82.7	84.1
	D (↓)	85.4	82.4	83.9
	S (–)	82.5	78.8	80.6
20	U (↑)	83.7	81.0	82.3
	D (↓)	83.3	81.0	82.1
	S (–)	80.5	76.5	78.4

Table 5.14: Synthetic data performance measurements comparison: one-against-all method and one-against-one method. The synthetic data are generated from the simulation configured by the parameters given in Table 5.11. SP = spread-crossing; MP = mid-price; Training feature set = {All features}, P = Precision, R = Recall, F_1 = F_1-measure. Label U = Upward, D = Downward, S = Stationary. 1vs.all = one-against all method; 1vs.1 = one-against-one method; Training size = 2000; Δt = 5 (events)

		Precision(%)		Recall(%)		F_1-Measure(%)	
Metrics	Label	1vs.all	1vs.1	1vs.all	1vs.1	1vs.all	1vs.1
SP	U (↑)	95.1	95.0	91.2	90.2	93.1	91.5
	D (↓)	93.8	93.2	92.6	92.2	93.2	92.7
	S (–)	95.4	95.3	95.8	96.0	95.6	95.6
MP	U (↑)	87.2	86.3	89.1	90.6	88.1	88.4
	D (↓)	87.9	85.2	90.0	89.2	88.9	87.2
	S (–)	85.1	84.3	82.3	81.2	83.7	82.7

As a supplementary, the Table 5.14 demonstrates the performance of the training model on the simulation data based on one-against-one and one-against all methods. With SP and MP denote

the metrics of spread crossing prediction and mid-price direction prediction respectively, the table gives the performance measurements as precision, recall and F_1-Measure for the modeled trained. One can observed that the both metrics on different multi-class training methods show robust performance on the prediction. Also we can notice that, by referring to the performance of real data AAPL in Table 5.10, the performance of the one-against-all methods still slightly outperforms the one-against-one method, that probably inherit the data characteristics of AAPL.

The Table 5.15 demonstrates the experimental results of the performance of cross training and testing on the real data and simulated data. The notation $M_r(D_s)$ denotes the model is trained by using the real data and testing on the synthetic data. Similarly, $M_r(D_s)$ represents the case that the model is trained via synthetic data while testing on real data. The training data size is the same as the testing data size. In Table 5.15, two groups of experimental results with 1200 and 2000 as the data size are shown. The precision, recall, and F_1-measure for each class of the data are computed for both groups.

Table 5.15: Performance of models by cross training and testing. The synthetic data are generated from the simulation configured by the parameters given in Table 5.11 based on AAPL. $M_r(D_s)$ = model is trained by real data (M_r) and tested on synthetic data (D_s). $M_r(D_s)$ = model is trained by synthetic data (M_s) and tested on real data (D_r). $\Delta t = 5$ (events). Training feature set = {All features}, P = Precision, R = Recall, F_1 = F_1-measure. Label U = Upward, D = Downward, S = Stationary. Training data size = Testing data size = 1200, 2000.

Size = 1200				
Model	Label	P(%)	R(%)	F_1(%)
$M_r(D_s)$	U (↑)	73.5	72.4	72.9
	D (↓)	72.1	76.8	74.4
	S (–)	89.0	65.2	75.3
$M_s(D_r)$	U (↑)	65.5	71.2	68.2
	D (↓)	63.1	61.0	62.0
	S (–)	90.7	78.5	84.2

Size = 2000				
Model	Label	P(%)	R(%)	F_1(%)
$M_r(D_s)$	U (↑)	74.2	74.8	74.5
	D (↓)	68.0	78.7	73.0
	S (–)	92.2	82.7	87.2
$M_s(D_r)$	U (↑)	66.9	70.2	68.5
	D (↓)	70.9	67.5	69.2
	S (–)	91.9	80.5	85.8

Similar to the experiments in Table 5.13, the data sampled in the training set are randomly selected and follow the class distribution that are in the synthetic data and real data. The simulated model M_s is trained on the synthetic data by using the parameters extracted from half day real data, and tested on the rest of the half real data D_r. Similarly, the real data model M_r is trained

on half day real data and tested on the rest of the half synthetic data D_s. From Table 5.15, it is obvious that, all the main performance measurements are around or above 70% except for the downward class in $M_s(D_r)$ from the 1200 size group. Within the same size experiment group, one can observe that, the performance of real data models that are tested against synthetic data are generally 0.2%-16.8% higher than those of models trained by synthetic data while testing on real data.

To interpret this, we consider two aspects. Firstly, the synthetic data are generated based on the characterized parameters from the real data, thus the data inherit the dynamic characteristics of real data. The performance of SVMs model trained by the real data shows that, the real data model, which captures the intrinsic characteristics of real data, also works well on its derivation data – the synthetic data. Secondly, shown by the results of $M_s(D_r)$, the simulated data, which describe the dynamics of real data by calibrated parameters, indeed captures the essential characteristics, even not fully, of the real data. When testing the synthetic model against real data, the performance measurements still keep above 65%, even though not as high as that of real data model. On the other hand, by comparing the 1200 size group and 2000 size group, we can see the increasing of the data size leads to the enhancement of all the performance measurements. In summary, the cross training and testing experiments validate the robustness of the model performance of SVMs, which is illustrated by the prediction performance on simulated data generated systematically. More importantly, it provides a justification that, the synthetic data generated by our method indeed be able to incorporate partial dynamic profile from real data.

5.6 Statistical Significance Tests of Performance

The paired t-test is used to further justify the performance of proposed model. With chosen measurements — weighted precision (P%), weighted percentage of correct prediction (C%) and weighted F_1-measure (F_1%) of the three classes, the experiments are designed to reject the hypothesis that, the performance of the model are not statistically significant. In the sense that, the results of the model are systematically produced rather than due to the factors of lucky or randomness.

Table 5.16: Results of paired t-test of spread crossing prediction, $\Delta t = 5$ (events), training data size = 2000. The measurements are weighted precision (P%), weighted percentage of correct prediction (C%) and weighted F_1-measure (F_1%). Cross-validation $n = 10$, Trials = 10; The values in parentheses are the p-values obtained from the paired t-test in scientific notation. Values less than 0.01 represent significance to the 99% confidence level.

	ZeroR			Multi-class SVMs		
Ticker	P(%)	C(%)	F_1(%)	P(%)	C(%)	F_1(%)
AAPL	36.22	59.14	44.21	91.03 (1E-08)	90.25 (8E-07)	90.01 (7E-08)
GOOG	34.11	56.43	43.01	92.00 (5E-09)	91.33 (1E-07)	91.51 (4E-08)

Table 5.17: Results of paired t-test of mid-price direction prediction, $\Delta t = 5$ (events), training data size = 2000. The measurements are weighted precision (P%), weighted percentage of correct prediction (C%) and weighted F_1-measure (F_1%). Cross-validation $n = 10$, Trials = 10; The values in parentheses are the p-values obtained from the paired t-test in scientific notation. Values less than 0.01 represent significance to the 99% confidence level.

	ZeroR			Multi-class SVMs		
Ticker	P(%)	C(%)	F_1(%)	P(%)	C(%)	F_1(%)
MSFT	36.00	52.73	41.50	74.34 (1E-07)	74.65 (1E-04)	72.00 (1E-06)
INTC	41.00	51.63	43.00	75.00 (7E-07)	73.54 (1E-04)	71.20 (2E-06)

Table 5.18: Results of paired t-test of mid-price direction prediction by using simulated data with AAPL statistical profile; $\Delta t = 5$ (events); training data size = 2000. SP = spread crossing; MP = mid-price direction; The measurements are weighted precision (P%), weighted percentage of correct prediction (C%) and weighted F_1-measure (F_1%). Cross-validation $n = 10$, Trials = 10; The values in parentheses are the p-values obtained from the paired t-test in scientific notation. Values less than 0.01 represent significance to the 99% confidence level.

	ZeroR			Multi-class SVMs		
Metric	P(%)	C(%)	F_1(%)	P(%)	C(%)	F_1(%)
SP	37.20	55.23	43.15	95.1 (1E-11)	90.65 (1E-10)	94.75 (7E-11)
MP	36.00	54.33	41.20	88.2 (4E-11)	85.63 (2E-10)	87.20 (5E-10)

ZeroR classifier is chosen to be the benchmark. As the simplest classification method, ZeroR predicts all data points to be the majority class of training data set. Maintaining the correctness of majority class, the performance of ZeroR generally outperforms the random classifier, which

predicts data points to each class with equal probability. Although there is no predictability power in ZeroR, it is useful for determining a baseline performance for other classifiers. The paired t-test is performed between the multi-class SVMs model and the ZeroR. Given a confidence level α, if the hypothesis — performance of multi-class SVMs is equivalent to that of ZeroR classifier, is rejected, then it justifies that, the model performance of multi-class SVMs is statistically significant.

The multi-class SVMs and ZeroR will be trained on the same data set and validated by n-fold cross validation. For both methods, n models are trained by using different combinations of training and testing sets of given data pool, then measurement for the n models are recorded. The measurements from m trials of cross validation are recorded for both methods respectively. Totally $m \times n$ experiments are made. The average of the $m \times n$ difference of measurements $\bar{d}_i$, $(i = 1, ..., m \times n)$ and standard deviation are used to compute the t-value for further finding p-value. The null hypothesis H_0 that $\bar{d}_i = 0$ is rejected if p-value is less than α, otherwise H_0 is accepted.

The experiments results are shown by the Table 5.16 and 5.17 along with the results of simulated data in Table 5.18. The $*$ denotes the statistical significant under the level $\alpha = 0.01$. And the p-values are attached inside parentheses with the corresponding tests. Using AAPL in Table 5.16 as an example, the ZeroR has the 59.14% correctly predict examples on average of $m \times n = 10 \times 10 = 100$ experiments, while multi-class SVMs reaches 90.25%. The p-value for the paired t-test is 8E-07. For the results of synthetic data, the p-values are even smaller since the model performance on simulated data are better than those of real data. The p-values in the demonstrated tables are so far less than the significant level $\alpha = 0.01$, that is, the performance of multi-class SVMs are significantly better than that of ZeroR classifier and the H_0 is rejected.

5.7 Trading Strategy Tests Based on Built Model

As a final test of the effectiveness of the SVM model, we test it with a simple-minded trading strategy against our data. The strategy below is not meant to represent a realistic trading implementation, but rather a simple sanity test to illustrate whether the model produces useful information in the context of sample market data. As an unseen data point comes up, which rep-

resenting a new event occurs, the model performs the prediction and labels the data with upward, downward and stationary under a specified prediction horizon Δt.

Table 5.19: Trading test by using various length of sliding windows based on the prediction of spread crossing, $\Delta t = 5$ (events). In each trading window, the simulated trader goes long \$100.00 on upward spread crossing and short back 5 events later; goes short with \$100.00 if downward spread crossing occurs; no moves if stationary prediction occurs. After the window duration ends, the accumulated profits/loss is computed. Totally 30 windows are included in the test. The mean and standard deviation of the profit of different window lengths are shown in the table. 1 tick = \$0.01; Ticker = AAPL.

AAPL

Duration (s)	Mean (Tick)	Std. Dev. (Tick)	Avg. # of Trades
300	1.5390	1.4352	13.0
600	1.7781	1.1113	22.0
900	1.9781	1.5113	31.0
1800	2.8658	1.8491	68.0
3600	3.5862	1.1083	126.0

Table 5.20: Trading test by using various length of sliding windows based on the prediction of spread crossing, $\Delta t = 5$ (events). In each trading window, the simulated trader goes long \$100.00 on upward spread crossing and short back 5 events later; goes short with \$100.00 if downward spread crossing occurs; no moves if stationary prediction occurs. After the window duration ends, the accumulated profits/loss is computed. Totally 30 windows are included in the test. The mean and standard deviation of the profit of different window lengths are shown in the table. 1 tick = \$0.01; Ticker = GOOG.

GOOG

Duration (s)	Mean (Tick)	Std. Dev. (Tick)	Avg. # of Trades
300	1.5847	0.4429	11.0
600	1.6205	0.4665	19.0
900	1.8001	0.6302	30.0
1800	2.6075	0.4678	59.0
3600	3.2161	0.4851	119.0

For example, if $\Delta t = 5$ events, and the model predicts an upward event at t, then long position is taken by submit a market buy order at the best ask price at t and sell back by best bid price at at $t + 5$. Similarly, if a downward prediction is given, then position is short by best bid price at t and long by best ask price at $t + 5$. No action is taken when stationary prediction is given.

Assume no transaction cost, each trade is made by a principle \$100.00, profit and loss of each trade is accumulated until the trading simulation is ended. With different window durations, the trading simulation are performed on N consecutive sliding windows. Finally, the average and standard deviation of the profits generated by the N windows are computed.

Table 5.21: Trading test by using various length of sliding windows based on the prediction of spread crossing, $\Delta t = 5$ (events). In each trading window, the simulated trader goes long \$100.00 on upward spread crossing and short back 5 events later; goes short with \$100.00 if downward spread crossing occurs; no moves if stationary prediction occurs. After the window duration ends, the accumulated profits/loss is computed. 30 sliding windows are used in the tests of window size 300s, 600s, 900s; 5 sliding windows are used in the test of window size 1800s and 3600s. The mean and standard deviation of the profit of different window lengths are shown in the table. 1 tick = \$0.01; Ticker = Synthetic Data.

Synthetic Data

Duration (s)	Mean (Tick)	Std. Dev. (Tick)	Avg. # of Trades
300	1.5259	1.3422	13.0
600	1.8101	1.2665	26.0
900	2.0013	1.4441	37.0
1800	3.0075	1.4873	73.0
3600	3.8161	1.2835	131.0

Tabel 5.19 and 5.20 show the results for the simulated trading tests of AAPL and GOOG. In the tables, the mean of profits of 30 trading windows with 5 different durations are shown. As the measurement of risk, the standard deviations of the trading windows are reported. We can see that, both standard deviations of AAPL and GOOG are less than the mean, which suggest positive average returns on the trading. Take GOOG as an example, the standard deviations are less than half of the mean on the mean. This suggests the positive expected profits on the tested trading strategy and validate the effectiveness of the proposed model. On the other hand, Table 5.21 gives the trading test results applied on synthetic data. The synthetic data, which is generated based on the parameters calibrated from the real data of ticker AAPL as in Table 5.11. 30 sliding windows are used in the tests of window size 300s, 600s, 900s; 5 sliding windows are used in the test of window size 1800s and 3600s. One can observe that the mean and the average number of trades in each type of window are very close to those as in the real data of Table 5.19, however, all the risk (measured as standard deviations) are generally 0.1-0.3 less than that of real data. This might

due to the better model performance of the model on simulated data as shown in Table 5.13. The profit and risk also conclude the robust performance of SVMs from the simulated data perspective.

The Figure 5.8 and 5.9 demonstrate the windows of simulated trading with duration 1800s. The blue curves are the correct stationary prediction as described by the mid-price of the stock, the green curves are the bid-ask spread evolution with respect to time. By different colors of markers, the prediction results in the form of their correctness and types of errors are labelled directly on the curve. In the unit of dollar, the left and right y-axis are for the mid-price of the ticker and the bid-ask spread respectively. Each time receiving a prediction, the model automatically acts according to the trading strategy mentioned before, thus the corresponding positions are taken. Shown by the green curves, the bid-ask spread can fluctuate dramatically during a very short period of time.

For example, in Figure 5.7, which is the zoom in for the region of purple box in the Figure 5.8, we can see that at the time $t = 1013.2594782$s, an upward prediction denoted as a green arrow is given. Then the long position should be taken at the best ask price \$586.59. At the time of 5 events later, $t = 1013.2594787$s (the prediction turns out to be correct), the asset is sold at the best bid price \$586.67 to realize the profit.

Corresponding to the trading windows of Figure 5.8 and 5.9, the profit-loss curves are also sketched accompanied by the predictions in the Figure 5.10 and 5.11. The profit and loss curves are based on the same strategy above with unit dollar principle. It's intuitive to see that in the places where the profit-loss curves drop down, there are events who are mis-predicted by the model. For example, the profit drops significantly around the time 600s in the Figure 5.10, it is due to an incorrect upward prediction which actually is a down event. For the similar reason, the jump up in the profit curve indicates where the correct predictions are made. Both Figure 5.10 and 5.11 show the positive profits at around 3 ticks and 1.5 ticks respectively when the simulation windows end.

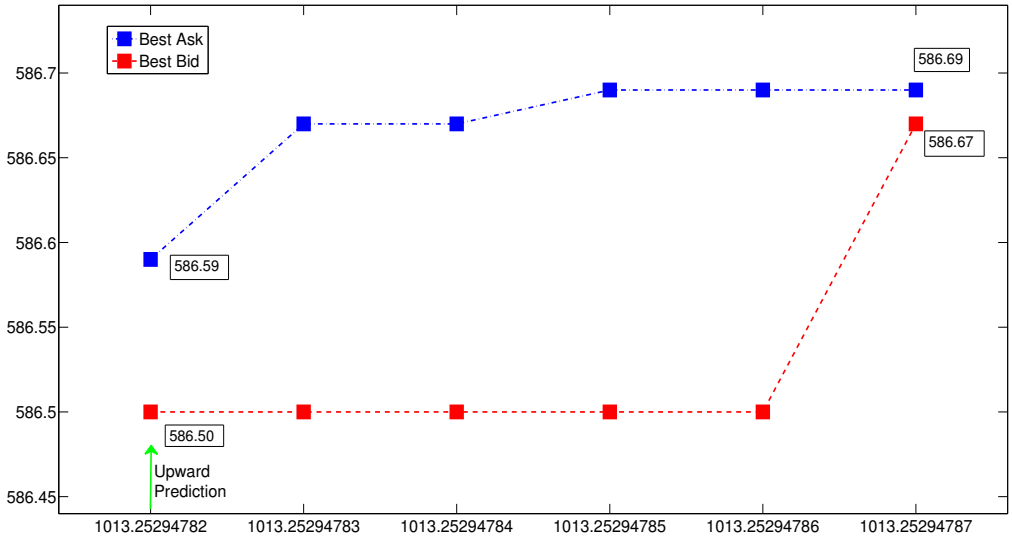

Figure 5.7: Zoom in: upward crossing prediction with the best ask and best bid prices evolution in the purple box of Figure 5.8. y-axis is the stock price in US dollar; x-axis is the time stamps up to nanoseconds; the green arrow indicates an upward prediction occurs at the time of moment. $\Delta t = 5$ (events); Ticker = AAPL.

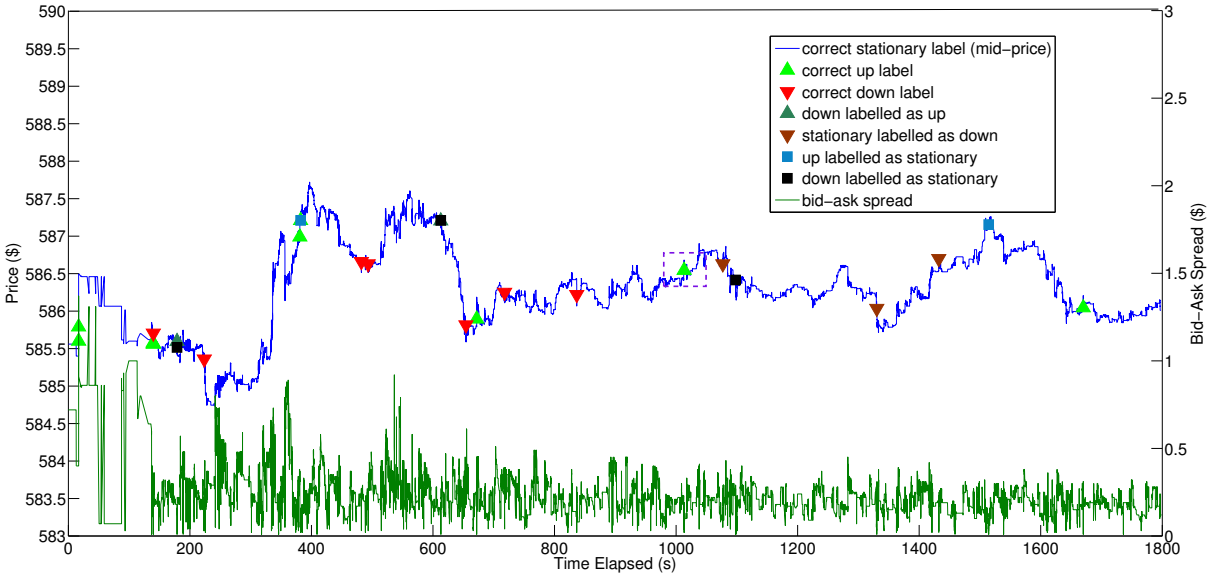

Figure 5.8: Example 1: prediction with labelled signals and bid-ask spread evolution. Left y-axis is the stock price in US dollar; the right y-axis is the bid-ask spread in US dollar. Time duration = 1800 seconds. The purple box contains a time period and the prediction, which will be zoomed in the Figure 5.7. $\Delta t = 5$ (events); Ticker = AAPL.

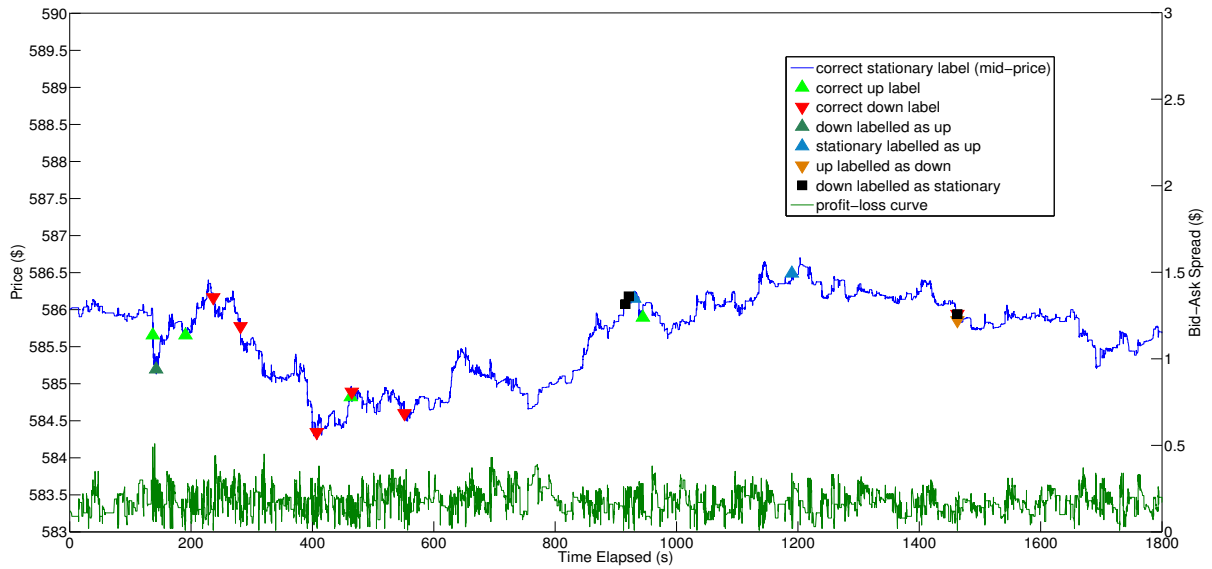

Figure 5.9: Example 2: prediction with labelled signals and bid-ask spread evolution. Left y-axis is the stock price in US dollar; the right y-axis is the bid-ask spread in US dollar. Time duration = 1800 seconds. $\Delta t = 5$ (events); Ticker = AAPL.

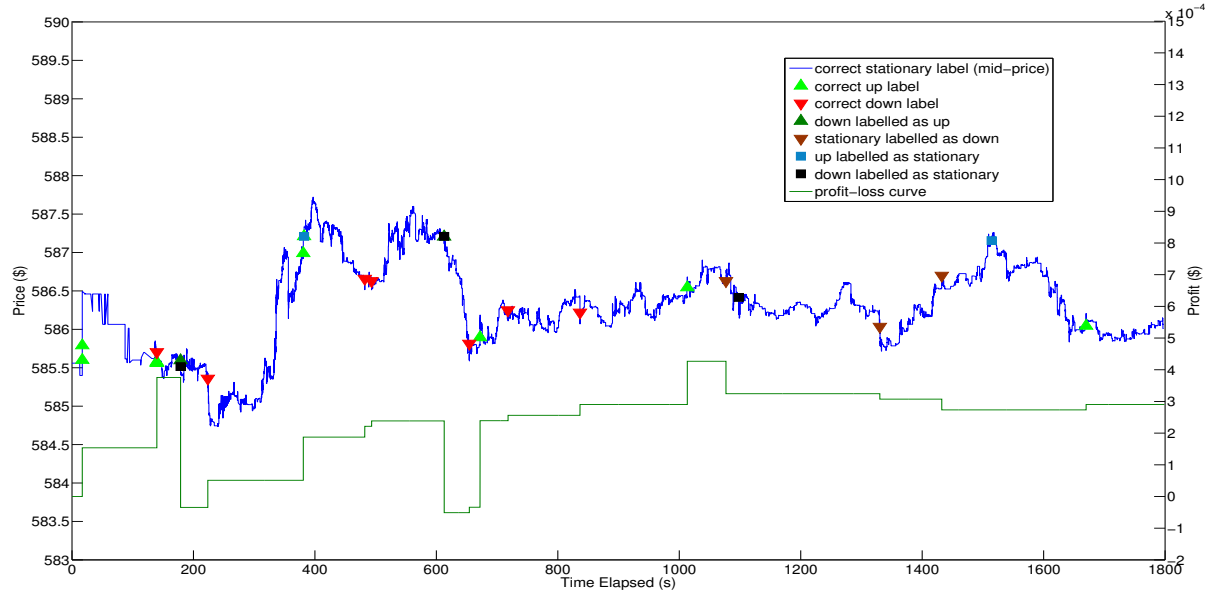

Figure 5.10: Example 1: prediction with labelled signals and profit-loss curve. Left y-axis is the stock price in US dollar; the right y-axis is the bid-ask spread in US dollar. Time duration = 1800 seconds. $\Delta t = 5$ (events); Ticker = AAPL.

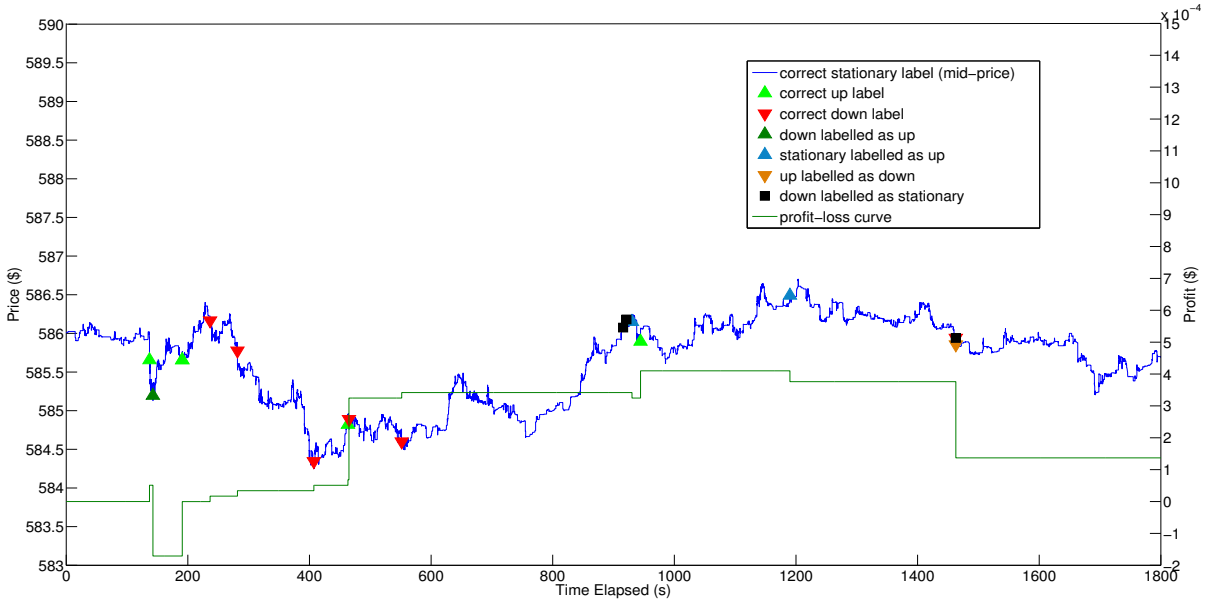

Figure 5.11: Example 2: prediction with labelled signals and profit-loss curve. Left y-axis is the stock price in US dollar; the right y-axis is the bid-ask spread in US dollar. Time duration = 1800 seconds. $\Delta t = 5$ (events); Ticker = AAPL.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

Equipped with machine learning techniques, the proposed framework that automates the prediction process on metrics for high frequency limit order book dynamics is described in detail and evaluated thoroughly with real-world data. For a given metric such as mid-price or price spread crossing, the framework automatically constructs a set of training data by extracting a variety of features from the message book as well as order book and deriving labels for samples in the data set; and subsequently builds a learning model with the help of multi-class support vector machines (SVMs). The feature vector representing each trading event is further enriched with time-insensitive and time-sensitive features, both of which are derived from limit order book. To improve the efficiency in training models and labeling unseen samples, features are selected with the help of information gains so that only those attributes significantly contributing to the performance of the resulting model are retained. The validity and robustness of models are verified with a n-fold cross-validation process that evaluates model performance in terms of precision, recall and F_1-measure. The characteristics of an unseen trading event with respect to the metric in question can then be predicted by using the resulting model and provide valuable guides for real-time trading strategies.

Our experiments with real-world data show that the feature sets designed in the proposed framework are effective. For instance, the models built for mid-price by using only the basic features can attain on average 79.6%, 82.6%, and 80.3% for precision, recall and F_1-measure, respectively; the performance can be further improved by about 6.3% and 3.2% should time-insensitive and time-sensitive features be also accordingly included. Experiments presented in this paper also clearly demonstrate the efficiency of the framework as the time spent on classifying an unseen data point is far less than a tenth of a millisecond. It is established through our experiments that the models trained in the proposed framework not only have a high prediction coverage – ratio of samples with valid labels over all data, but also manifest superb performance in prediction accuracy and robustness, vitally important for helping make real-time trading decisions. Our comparison study

on models built in the proposed framework and baseline ZeroR predictor indicates that the former exceedingly outperforms the latter, and the improvement in performance is statistically significant. Moreover, the trading strategies based on the learning models trained in the proposed framework can achieve profitable returns with low risks, further demonstrating the effectiveness of the modeling methodology introduced in this paper.

It is our future plan to investigate the relationship between model performance and feature set selection as experiments sketched in this paper do reveal that feature selection in fact affects noticeably model performance. For example, time-insensitive features result in better model performance on price spread crossing for MSFT than time-sensitive features; but the opposite is true on mid-price for GOOG. We also plan to devise a much richer set of features by using, for instance, those variables derived from statistical modeling methods and quantify their impacts on model performance; meanwhile, we are studying a variety of metrics including order arrival duration, bid-ask spread, volatility, and are evaluating their effectiveness on helping the construction of trading strategies. Besides SVMs, we are implementing other machine learning modeling methods such as SOMs, ANNs and logistic regression in the framework, so that multiple learning techniques can be easily integrated together and work coordinately to predict metrics for an unseen data point, attempting to further improve modeling performance. Moreover, it is also our desire to apply the proposed framework to other financial markets such as futures, options, and foreign exchanges, and polish the framework accordingly.

APPENDIX A

THEORY AND DERIVATIONS RELATED TO SVMS

Definition A.0.1 (**Convex Set**). *A set C is convex if, for any x, $y \in C$ and $\theta \in \mathbb{R}$ with $0 \leq \theta \leq 1$,*

$$\theta x + (1 - \theta)y \in C$$

The point $\theta x + (1 - \theta)y$ is called a ***convex combination*** of points x and y. And it is fairly obvious that, all of $\mathbb{R}^n$ are convex sets, since for any x, $y \in \mathbb{R}^n$, $\theta x + (1 - \theta)y \in \mathbb{R}^n$.

Definition A.0.2 (**Convex Function**). *A function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is convex if its domain $\mathcal{D}(f)$ is a convex set, and if, for all x, $y \in \mathcal{D}(f)$ and $\theta \in \mathbb{R}$, $0 \leq \theta \leq 1$,*

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y).$$

We say a function is ***strictly convex*** if the strict inequality holds in the Definition A.0.2 for $x \neq y$ and $0 \leq \theta \leq 1$. On the other hand, f is ***concave*** if $-f$ is convex, and similarly, f is ***strictly concave*** if $-f$ is strictly convex.

Given the definitions of convex sets and functions, we now consider the definition of convex optimization problems.

Definition A.0.3 (**Convex Optimization Problem**). *A convex optimization problem is in the form:*

$$\begin{aligned} &\min f(x) \\ &s.t.\ \ x \in C \end{aligned} \tag{A.1}$$

where f is a convex function, C is a convex set, and x is the optimization variable. Specifically in the scope of this paper, we define the problem in a more concrete form:

$$\begin{aligned} &\min f(x) \\ &s.t. \;\; g_i(x) \leq 0, \quad i = 1, ..., m \\ &\qquad h_i(x) = 0, \quad i = 1, ..., p \end{aligned} \tag{A.2}$$

where f is a convex function, g_i are convex functions and h_i are affine functions. x is the optimization variable.

Note that in the Definition A.0.3, the affine functions h_i are defined as the functions that of the form $h_i(x) = a^T x + b$ for some $a \in \mathbb{R}^n, b \in \mathbb{R}$, i.e. the linear function of x.

The ***optimal value*** of an optimization problem is denoted as p^* and is equal to the minimum possible value of the objective function in the feasible region

$$p^* = \inf\{f(x) : g_i(x) \leq 0, i = 1, ..., h_i(x) = 0, i = 1, ..., p\}.$$

The value of p^* is allowed to be $+\infty$ and $-\infty$ when the problem is either *infeasible* (the feasible region is empty) or *unbounded below* (there exists feasible points such that $f(x) \to -\infty$), respectively. Then x^* is an ***optimal point*** if $f(x^*) = p^*$. Note that there can be more than one optimal point, even when the optimal value is finite.

Defining the optimal value and optimal point, then we formalize the definitions of ***locally optimal*** and ***globally optimal*** of convex problems by the Definition A.0.4 and A.0.5:

Definition A.0.4. *A point x is locally optimal if it is feasible (i.e., it satisfies the constraints of the optimization problem) and if there exists some $R > 0$ such that all feasible points z with $\|x - z\|_2 \leq R$, satisfy $f(x) \leq f(z)$.*

Definition A.0.5. *A point x is globally optimal if it is feasible and for all feasible points z, $f(x) \leq f(z)$.*

With the definitions above, now we introduce a very important theorem in regard of convex optimization problems:

Theorem A.0.1. *For a convex optimization problem, all locally optimal points are globally optimal.*

Proof. Suppose that x is a locally optimal point which is not globally optimal, i.e., there exists a feasible point y such that $f(x) > f(y)$. By the definition of local optimality, there exists no feasible points z such that $\|x - z\|_2 \leq R$ and $f(z) < f(x)$. But now suppose we choose the point

$$z = \theta y + (1 - \theta)x \quad \text{with} \quad \theta = \frac{R}{2\|x - y\|_2}$$

Then

$$\begin{aligned} \|x - z\|_2 &= \|x - (\frac{R}{2\|x - y\|_2} y + (1 - \frac{R}{2\|x - y\|_2})x)\|_2 \\ &= \|\frac{R}{2\|x - y\|_2}(x - y)\|_2 \\ &= R/2 \leq R \end{aligned} \tag{A.3}$$

In addition, by the convexity of f we have

$$f(z) = f(\theta y + (1 - \theta)x) \leq \theta f(y) + (1 - \theta) f(x) < f(x)$$

Furthermore, since the feasible set is a convex set, and since x and y are both feasible, $z = \theta y + (1 - \theta)x$ will be feasible as well. Therefore, z is a feasible point with $\|x - z\|_2 < R$ and $f(z) < f(x)$. This is contradicts to the assumption thus completes the proof. □

For application purpose, the convex problems usually can be categorized into several special cases, which can be devised extremely efficient algorithms. The most common special cases are called ***linear programming*** problems and ***quadratic programming*** problems.

Definition A.0.6 (**Linear Programming**). *A convex optimization problem is a linear program (LP) if both the objective function f and inequality constraints g_i are affine functions. i.e., the problems have the form*

$$\begin{aligned} &\min c^T x + d \\ &s.t.\ \ Gx \preceq h \\ &\quad\ Ax = b \end{aligned} \tag{A.4}$$

where $x \in \mathbb{R}^n$ is the optimization variable, $c \in \mathbb{R}^n$, $d \in \mathbb{R}$, $G \in \mathbb{R}^{m\times n}$, $h \in \mathbb{R}^m$, $A \in \mathbb{R}^{p\times n}$, $b \in \mathbb{R}^p$ are defined by the problem, and '$\preceq$' denotes elementwise inequality.

Definition A.0.7 (**Quadratic Programming**). *A convex optimization problem is a quadratic program (QP) if the objective function f is a convex quadratic function and the inequality constraints g_i are affine functions. i.e., the problems have the form*

$$\begin{aligned} &\min \frac{1}{2}x^T P x + c^T x + d \\ &s.t.\ \ Gx \preceq h \\ &\quad\ Ax = b \end{aligned} \tag{A.5}$$

where $x \in \mathbb{R}^n$ is the optimization variable, $c \in \mathbb{R}^n$, $d \in \mathbb{R}$, $G \in \mathbb{R}^{m\times n}$, $h \in \mathbb{R}^m$, $A \in \mathbb{R}^{p\times n}$, $b \in \mathbb{R}^p$ are defined by the problem, and $P \in \mathbb{S}_+^n$ is a symmetric positive semidefinite matrix.

Given the introduction of convex optimization problems as in the previous section, in this section we firstly present the definition of ***Lagrangian***, which is the basis of the Lagrange duality theory. Then follow by the derivation and interpretation, we give the equality conditions for the optimality of primal and dual problems, known as ***Slater's condition***. At last, we outline the

theorem of ***Karush-Kuhn-Tucker (KKT) conditions***, which characterize the optimal conditions for a primal dual optimization pair.

Consider a convex optimization problem of the form:

$$\begin{aligned} &\min f(x) \\ &s.t.\ \ g_i(x) \leq 0, \quad i = 1, ..., m \\ &\qquad h_i(x) = 0, \quad i = 1, ..., p \end{aligned} \tag{A.6}$$

where $x \in \mathbb{R}^n$ is the optimization variable, $f : \mathbb{R}^n \to \mathbb{R}$ and $g_i : \mathbb{R}^n \to \mathbb{R}$ are differentiable convex functions, and $h_i : \mathbb{R}^n \to \mathbb{R}$ are affine functions. Then the generalized ***Lagrangian*** of the problem, is a function $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \mapsto \mathbb{R}$, defined as

$$\mathcal{L}(x, \alpha, \beta) = f(x) + \sum_{i=1}^{m} \alpha_i g_i(x) + \sum_{i=1}^{p} \beta_i h_i(x) \tag{A.7}$$

where $x \in \mathbb{R}^n$ is referred as ***optimal variables*** of the Lagrangian. The vector $\alpha \in \mathbb{R}^m$ with one variable α_i for each of the m convex inequality constraints in the original optimization problem. And vector $\beta \in \mathbb{R}^p$ and each dimension β_i corresponds to an affine equality of the p constraints. The elements such as α_i and β_i are known as ***dual variables*** or ***Lagrange multipliers***.

By intuition, the Lagrangian can be view as a modified version of the objective function to the original convex optimization problem which accounts for each of the constraints. The Lagrange multipliers α_i and β_i can also be thought of costs associated with violating different constraints. In the following, we present the definitions of primal and dual problems and then try to interpret the relationship by introducing the weak and strong duality.

Definition A.0.8 (**The Primal Problem**). *In the optimization problem,*

$$\min_x [\max_{\alpha, \beta : \alpha_i \geq 0, \forall i} \mathcal{L}(x, \alpha, \beta)] = \min_x \theta_{\mathcal{P}}(x) \tag{A.8}$$

The function $\theta_{\mathcal{P}} : \mathbb{R}^n \mapsto \mathbb{R}$ is called the **primal objective**, *and the unconstrained minimization problem on the right hand side is known as the* **primal problem**. *And a point $x \in \mathbb{R}^n$ is* **primal feasible** *if $g_i(x) \leq 0$, $i = 1, ..., m$ and $h_i(x) = 0$, $i = 1, ..., p$. Specially, the vector $x^* \in \mathbb{R}^n$ is used to denote the solution of problem A.8, and $p^* = \theta_{\mathcal{P}}(x^{*)}$ denotes the optimal value of the primal objective.*

Definition A.0.9 (**The Dual Problem**). *In the optimization problem,*

$$\max_{\alpha,\beta:\alpha_i \geq 0,\forall i} [\min_x \mathcal{L}(x, \alpha, \beta)] = \max_{\alpha,\beta:\alpha_i \geq 0,\forall i} \theta_{\mathcal{D}}(\alpha, \beta) \tag{A.9}$$

The function $\theta_{\mathcal{D}} : \mathbb{R}^m \times \mathbb{R}^n \mapsto \mathbb{R}$ is called the **dual objective**, *and the constrained maximization problem on the right hand side is known as the* **dual problem**. *And that (α, β) are* **dual feasible** *if $\alpha_i \geq 0$, $i = 1, ..., m$. Typically, the pair of vectors $(\alpha^*, \beta^*) \in \mathbb{R}^m \times \mathbb{R}^p$ is used to denote the solution of problem A.9, and we let $d^* = \theta_{\mathcal{D}}(\alpha^*, \beta^*)$ denotes the optimal value of the dual objective.*

To interpret the primal problem, note that the primal objective, $\theta_{\mathcal{P}}(x)$, is a convex function of x,

$$\begin{aligned} \theta_{\mathcal{P}}(x) &= \max_{\alpha,\beta:\alpha_i \geq 0,\forall i} \mathcal{L}(x, \alpha, \beta) \\ &= \max_{\alpha,\beta:\alpha_i \geq 0,\forall i} [f(x) + \sum_{i=1}^{m} \alpha_i g_i(x) + \sum_{i=1}^{p} \beta_i h_i(x)] \\ &= f(x) + \max_{\alpha,\beta:\alpha_i \geq 0,\forall i} [\sum_{i=1}^{m} \alpha_i g_i(x) + \sum_{i=1}^{p} \beta_i h_i(x)] \end{aligned} \tag{A.10}$$

And also we have the fact that $f(x)$ does not depend on the value of α and β.

Focus on the bracketed term in the last expression above, one can observe that, if any $g_i(x) > 0$, then maximizing the bracketed expression involves making the corresponding α_i an arbitrary large positive number; however, if $g_i \leq 0$, then the requirement that α_i be nonnegative means that the optimal setting of α_i to achieve the maximum is $\alpha_i = 0$, so that the maximum value is 0.

On the other hand, if any $h_i \neq 0$, then maximizing the bracketed expression involves choosing the corresponding β_i to have the same sign as $h_i(x)$ and arbitrarily large magnitude; however, if $h_i(x) = 0$, then the maximum value is 0, independent of β_i.

To summarize the two cases above, we can conclude that if x is primal feasible (i.e., $g_i(x) \leq 0$, $i = 1, ..., m$ and $h_i(x) = 0$, $i = 1, ..., p$), then the maximum value of the bracketed expression is 0, but if any of the constraints are violated, then the maximum value is ∞. Thus we have,

$$\theta_{\mathcal{P}(x)} = f(x) + \begin{cases} 0 & \text{if } x \text{ is primal feasible} \\ \infty & \text{if } x \text{ is primal infeasible} \end{cases} \tag{A.11}$$

Based on the observation, we can interpret the primal objective $\theta_{\mathcal{P}}(x)$ as a modified version of the original convex objective function in the optimization problem. The only difference is that the infeasible solutions — x's for which violate some constraints, have objective value ∞.

For the dual objective, $\theta_{\mathcal{D}}(\alpha, \beta)$, is a concave function of α and β. To be ready for interpreting the dual problem, we firstly introduce the following lemma:

Lemma A.0.1. *If (α, β) are dual feasible, then $\theta_{\mathcal{D}}(\alpha, \beta) \leq p^*$.*

Proof. By definition,

$$\begin{aligned} \theta_{\mathcal{D}}(\alpha, \beta) &= \min_x \mathcal{L}(x, \alpha, \beta) \\ &\leq \mathcal{L}(x^*, \alpha, \beta) \\ &= f(x^*) + \sum_{i=1}^{m} \alpha_i g_i(x^*) + \sum_{i=1}^{p} \beta_i h_i(x^*) \\ &\leq f(x^*) = p^* \end{aligned} \tag{A.12}$$

□

Implied by the Lemma A.0.1, given any dual feasible (α, β), the dual objective $\theta_{\mathcal{D}}(\alpha, \beta)$ provides a lower bound on the optimal value p^* of the primal problem. Since the dual problem involves maximizing the dual objective over the space of all dual feasible (α, β), it follows that the dual problem can be seen as a search for the tightest possible lower bound on p^*. In this sense, the quantity $(p^* - d^*)$ is called the ***duality gap***. Since α^*, β^* is the dual feasible point, as a consequence, the result given by the lemma below follows:

Lemma A.0.2 (**Weak Duality**). *For any pair of primal and dual problems,* $d^* \leq p^*$.

Specially, if the primal and dual optimization problems satisfy certain technical conditions called ***constraint quailfications***, then an even stronger results — known as ***strong duality*** holds, i.e., $d^* = p^*$. The following theorem gives the most commonly invoked constraint qualifications.

Theorem A.0.2 (**Strong Duality: Slater's Condition**). *Consider the convex optimization problem given in A.6, the strong duality holds if there exists an* $x \in \text{relint}(D)$, *where* $D = \cap_{i=1}^{m} \text{dom}(g_i) \cap \text{dom}(f)$, *such that* $g_i(x) < 0$ *for* $i = 1, ..., m$ *and* $h_i(x) = 0$ *for* $i = 1, ..., p$.

We can see that, the Slater's condition gives a sufficient condition for the strong duality holds, i.e., the solutions of primal and dual problem should equal to each other, $d^* = p^*$. In this case, an original primal problem can always be transformed into its dual problem and solve for the solution. In the later section, we shall prove and verify that, the problem at hand, knowns as the model of ***Support Vector Machines(SVMs)*** is actually a quadratic optimization problem and satisfies the Slater's condition, thus can be solved by its dual form.

In the following lemma, we prove a very important property based on the strong duality, which is knowns as ***complementary slackness***:

Lemma A.0.3 (**Complementary Slackness**). *If strong duality holds for the primal and dual problems pair in A.8 and A.9, then* $\alpha_i^* g_i(x_i^*) = 0$ *for each* $i = 1, ..., m$.

Proof. Suppose that strong duality holds, we observe that

$$\begin{aligned} p^* = d^* = \theta_{\mathcal{D}}(\alpha^*, \beta^*) &= \min_x \mathcal{L}(x, \alpha^*, \beta^*) \\ &\leq \mathcal{L}(x^*, \alpha^*, \beta^*) \\ &= f(x^*) + \sum_{i=1}^{m} \alpha_i^* g_i(x^*) + \sum_{i=1}^{p} \beta_i^* h_i(x^*) \\ &\leq f(x^*) = p^* \end{aligned} \tag{A.13}$$

Since the first and last expressions in the sequence are equal, it follows that every term in the

sequence is also equal. From the expression in the last two lines, we have

$$\sum_{i=1}^{m} \alpha_i^* g_i(x^*) + \sum_{i=1}^{p} \beta_i^* h_i(x^*) = 0 \tag{A.14}$$

However, recall that each α_i^* is nonnegative, each $g_i(x^*)$ is nonpositive, and each $h_i(x^*)$ is 0 due to the primal and dual feasibility of x^* and (α^*, β^*), respectively. Hence, equation A.14 is a summation of all nonpositive terms which equals to 0. If not, there are no compensating positive terms in the summation which would allow the overall sum to remain 0. □

Complementary slackness can be written in many equivalent ways. In the form of:

$$\begin{aligned} \alpha_i^* > 0 &\implies g_i(x^*) = 0 \\ g_i(x^*) < 0 &\implies \alpha_i^* = 0 \end{aligned} \tag{A.15}$$

We can see that whenever any α_i^* is strictly greater than zero, then this implies that the corresponding inequality constraint must hold with equality. This is referred as an ***active constraint***. In the following section, we shall see that, in the case of SVMs, active constraints are also known as ***support vectors***.

To finish this section, we summarize the necessary and sufficient conditions for a primal dual optimization pair optimality, which are known as ***Karush-Kuhn-Tucker (KKT) conditions***:

Theorem A.0.3 (**KKT Conditions**). *Suppose that $x^* \in \mathbb{R}^n$, $\alpha^* \in \mathbb{R}^m$ and $\beta^* \in \mathbb{R}^p$ satisfy the following conditions:*

1. *(Primal feasibility) $g_i(x^*) \leq 0$, $i = 1, ..., m$ and $h_i(x^*) = 0$, $i = 1, ..., p$,*
2. *(Dual feasibility) $\alpha_i^* \geq 0$, $i = 1, ..., m$,*
3. *(Complementary slackness) $\alpha_i^* g_i(x^*) = 0$, $i = 1, ..., m$,*
4. *(Lagrangian stationarity) $\nabla_x \mathcal{L}(x^*, \alpha^*, \beta^*) = 0$.*

Then x^* is primal optimal and (α^*, β^*) are dual optimal. Furthermore, if strong duality holds, any primal optimal x^* and dual optimal (α^*, β^*) must satisfy the conditions 1 through 4.

With KKT conditions, we can better interpret the optimality of the primal problem. At the same time, the conditions can be served as the criteria to justify whether a point x is indeed the optimal point for the primal problem. In the next section, we continue on introducing the mechanics of support vector machines based on the theories of Lagrange duality.

1. **Derivation of Equation 3.13**

   Take the equation 3.10 into the Lagrangian equation 3.8, we get

$$\begin{aligned}
\mathcal{L}(\vec{w}, b, \vec{\alpha}) &= \frac{1}{2}\|\vec{w}\|^2 - \sum_{i=1}^{m} \alpha_i [y_i(\vec{w} \cdot \vec{x_i} + b) - 1] \\
&= \frac{1}{2}\vec{w} \cdot \vec{w} - \sum_{i=1}^{m} \alpha_i y_i \vec{w} \cdot \vec{x_i} - \sum_{i=1}^{m} \alpha_i y_i b + \sum_{i=1}^{m} \alpha_i \\
&= \frac{1}{2}\vec{w} \cdot (\sum_{i=1}^{m} \alpha_i y_i \vec{x_i}) - \vec{w} \cdot (\sum_{i=1}^{m} \alpha_i y_i \vec{x_i}) - b\sum_{i=1}^{m} \alpha_i y_i + \sum_{i=1}^{m} \alpha_i \\
&= -\frac{1}{2}\vec{w} \cdot (\sum_{i=1}^{m} \alpha_i y_i \vec{x_i}) - b\sum_{i=1}^{m} \alpha_i y_i + \sum_{i=1}^{m} \alpha_i \\
&= -\frac{1}{2}(\sum_{i=1}^{m} \alpha_i y_i \vec{x_i}) \cdot (\sum_{i=1}^{m} \alpha_i y_i \vec{x_i}) - b\sum_{i=1}^{m} \alpha_i y_i + \sum_{i=1}^{m} \alpha_i \\
&= \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j (\vec{x_i} \cdot \vec{x_j}) - b\sum_{i=1}^{m} \alpha_i y_i
\end{aligned}$$

   And by equation 3.11, the last term of the above expression is actually zero, thus we have

$$\mathcal{L}(\vec{w}, b, \vec{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j (\vec{x_i} \cdot \vec{x_j}) \tag{A.16}$$

2. **An Example of Valid Kernel Function**

Example A.0.1. *Consider the a two dimensional input space with the mapping* $\phi : \mathbb{R}^2 \mapsto \mathbb{R}^3$*:*

$$\phi(\vec{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

where $\vec{x} \in \mathbb{R}^2$*. Then the inner product of* $\phi(\vec{x})$ *and* $\phi(\vec{z})$ *can be represented as:*

$$\begin{aligned} \phi(\vec{x}) \cdot \phi(\vec{z}) &= (x_1^2, \sqrt{2}x_1x_2, x_2^2)^T(z_1^2, \sqrt{2}z_1z_2, z_2^2) \\ &= x_1^2z_1^2 + 2x_1x_2z_1z_2 + x_2^2z_2^2 \\ &= (\vec{x} \cdot \vec{z})^2 \end{aligned} \tag{A.17}$$

In this case, we can see that the function $\kappa(\vec{x}, \vec{z}) = (\vec{x} \cdot \vec{z})^2$ *can compute the inner product of* $\phi(\vec{x}) \cdot \phi(\vec{z})$ *without explicitly computing the mapping* ϕ*. Thus the function* $\kappa(\vec{x}, \vec{z})$ *is one of the examples of the kernel functions.*

3. **Proof of Mercer's Theorem 3.3.1**

Proof. ($\Rightarrow$) Since κ is a valid kernel, then is has a corresponding map ϕ, such that the kernel matrix $K_{ij} = \kappa(\vec{x_i}, \vec{x_j}) = \phi(\vec{x_i}) \cdot \phi(\vec{x_j}) = \phi(\vec{x_j}) \cdot \phi(\vec{x_i}) = \kappa(\vec{x_j}, \vec{x_i}) = K_{ji}$. Hence K must be symmetric. Moreover, letting $\phi_k(\vec{x})$ denote the k-th coordinate of the vector $\phi(\vec{x})$, we find that for any vector $\vec{z}$, we have

$$\begin{aligned} \vec{z}^T K \vec{z} &= \sum_i \sum_j \vec{z_i} K_{ij} \vec{z_j} \\ &= \sum_i \sum_j \vec{z_i} \phi(\vec{x_i}) \cdot \phi(\vec{x_j}) \vec{z_j} \\ &= \sum_i \sum_j \vec{z_i} \sum_k \phi_k(\vec{x_i}) \phi_k(\vec{x_j}) \vec{z_j} \\ &= \sum_k \sum_i \sum_j \vec{z_i} \phi_k(\vec{x_i}) \phi_k(\vec{x_j}) \vec{z_j} \\ &= \sum_k (\sum_i \vec{z_i} \phi_k(\vec{x_i}))^2 \\ &\geq 0 \end{aligned} \tag{A.18}$$

By the arbitrary of $\vec{z}$, this shows K is positive semi-definite.

($\Leftarrow$) On the other hand, let $T = \{\vec{x_m}\}$, $m < \infty$ be the set of all possible data points. Since the corresponding kernel matrix K is positive semi-definite, it has nonnegative eigenvalues and can be factored as $K = UU^T$. Let $\phi(\vec{x_i}) = u_i$, where u_i is the i-th row of U. This gives the mapping for x_i such that $\kappa(\vec{x_i}, \vec{x_j}) = u_i \cdot u_j$. This completes the proof.

□

4. **Derivation of Dual Problem of Soft-margin SVMs 3.30**

Firstly, we rewrite the problem 3.30 into standard form:

$$\begin{aligned} \min_{\vec{w},b,\xi_i} \ & \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{m}\xi_i \\ & s.t. \ \ 1 - \xi_i - y_i(\vec{w}\cdot\vec{x_i} + b) \le 0, \quad i = 1, ..., m \\ & -\xi_i \le 0, \quad i = 1, ..., m. \end{aligned} \tag{A.19}$$

Then the Lagrangian for the problem is formed:

$$\mathcal{L}(\vec{w}, b, \vec{\xi}, \vec{\alpha}, \vec{\beta}) = \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{m}\xi_i + \sum_{i=1}^{m}\alpha_i(1 - \xi_i - y_i(\vec{w}\cdot\vec{x_i} + b)) - \sum_{i=1}^{m}\beta_i\xi_i, \tag{A.20}$$

Here, it is important to note that $(\vec{w}, b, \vec{\xi})$ collectively play the role of the primal variables. Similarly, $(\vec{\alpha}, \vec{\beta})$ collectively play the role of the dual variables normally used for the inequality constraints. Also notice that there are no affine equality constraints in this problem.

Then the primal optimization is given by:

$$\min_{\vec{w},b,\vec{\xi}} \theta_{\mathcal{P}}(\vec{w}, b, \vec{\xi}).$$

where

$$\theta_{\mathcal{P}}(\vec{w}, b, \vec{\xi}) := \max_{\vec{\alpha},\vec{\beta}:\alpha_i \ge 0, \beta_i \ge 0} \mathcal{L}(\vec{w}, b, \vec{\xi}, \vec{\alpha}, \vec{\beta}).$$

Thus, on the other hand, the dual optimization problem is given by:

$$\max_{\vec{\alpha},\vec{\beta}:\alpha_i\geq 0,\beta_i\geq 0} \theta_{\mathcal{D}}(\vec{\alpha},\vec{\beta}).$$

where

$$\theta_{\mathcal{D}}(\vec{\alpha},\vec{\beta}). := \min_{\vec{w},b,\vec{\xi}} \mathcal{L}(\vec{w},b,\vec{\xi},\vec{\alpha},\vec{\beta}).$$

Noticing that $\theta_{\mathcal{D}}(\vec{\alpha},\vec{\beta}). = \min_{\vec{w},b,\vec{\xi}} \mathcal{L}(\vec{w},b,\vec{\xi},\vec{\alpha},\vec{\beta})$ is an unconstrained optimization problem, where the objective function $\mathcal{L}(\vec{w},b,\vec{\xi},\vec{\alpha},\vec{\beta})$ is differentiable. The Lagrangian is a strictly convex quadratic function of $\vec{w}$, so for any fixed $(\vec{\alpha},\vec{\beta})$. If $(\vec{w}^*,b^*,\vec{\xi}^*)$ minimize the Lagrangian, by KKT theorem, it must be the case that:

$$\nabla_{\vec{w}}\mathcal{L}(\vec{w}^*,b^*,\vec{\xi}^*,\vec{\alpha},\vec{\beta}) = \vec{w}^* - \sum_{i=1}^{m}\alpha_i y_i \vec{x_i} = 0 \tag{A.21}$$

Moreover, the Lagrangian is linear in b and $\vec{\xi_i}$, by setting the derivatives with respect to b and $\vec{\xi}$ to zero, and add the resulting conditions as explicit constraints in the dual optimization problem, we have:

$$\frac{\partial}{\partial b}\mathcal{L}(\vec{w}^*,b^*,\vec{\xi}^*,\vec{\alpha},\vec{\beta}) = -\sum_{i=1}^{m}\alpha_i y_i = 0 \tag{A.22}$$

$$\frac{\partial}{\partial \xi_i}\mathcal{L}(\vec{w}^*,b^*,\vec{\xi}^*,\vec{\alpha},\vec{\beta}) = C - \alpha_i - \beta_i = 0 \tag{A.23}$$

By using the conditions above, we can compute the dual objective as:

$$
\begin{aligned}
\theta_{\mathcal{D}}(\vec{\alpha},\vec{\beta}) &= \mathcal{L}(\vec{w}^*, b^*, \vec{\xi}^*) \\
&= \frac{1}{2}\|\vec{w}^*\|^2 + C\sum_{i=1}^{m}\xi_i^* + \sum_{i=1}^{m}\alpha_i(1-\xi_i^* - y_i(\vec{w}^*\cdot\vec{x_i}+b^*)) - \sum_{i=1}^{m}\beta_i\xi_i^* \\
&= \frac{1}{2}\|\vec{w}^*\|^2 + C\sum_{i=1}^{m}\xi_i^* + \sum_{i=1}^{m}\alpha_i(1-\xi_i^* - y_i(\vec{w}^*\cdot\vec{x_i})) - \sum_{i=1}^{m}\beta_i\xi_i^* \\
&= \frac{1}{2}\|\vec{w}^*\|^2 + \sum_{i=1}^{m}\alpha_i(1-y_i(\vec{w}^*\cdot\vec{x_i}))
\end{aligned}
\tag{A.24}
$$

where the first equality follows from the optimality of $(\vec{w}^*, b^*, \vec{\xi}^*)$ for fixed $(\vec{\alpha},\vec{\beta})$. The second equality uses the definition of the generalized Lagrangian, and the third and fourth equalities follow from equation A.22 and equation A.23, respectively.

At last, we use the equation A.21, and observe that:

$$
\begin{aligned}
\frac{1}{2}\|\vec{w}^*\|^2 + \sum_{i=1}^{m}\alpha_i(1-y_i(\vec{w}^*\cdot\vec{x_i})) &= \sum_{i=1}^{m}\alpha_i + \frac{1}{2}\|\vec{w}^*\|^2 - \vec{w}^*\cdot\sum_{i=1}^{m}\alpha_i y_i\vec{x_i} \\
&= \sum_{i=1}^{m}\alpha_i + \frac{1}{2}\|\vec{w}^*\|^2 - \|\vec{w}^*\|^2 \\
&= \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\|\vec{w}^*\|^2 \\
&= \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{i=1}^{m}\alpha_i\alpha_j y_i y_j(\vec{x_i}\cdot\vec{x_j})
\end{aligned}
\tag{A.25}
$$

Then, the dual problem can be summarized as:

$$\begin{aligned}
\max_{\vec{\alpha},\vec{\beta}} W(\vec{\alpha}) = &\sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i\alpha_j y_i y_j (\vec{x_i} \cdot \vec{x_j}) \\
s.t.\ \ &\alpha_i \geq 0 \quad i = 1, ..., m \\
&\beta_i \geq 0 \quad i = 1, ..., m \\
&\alpha_i + \beta_i = C \quad i = 1, ..., m \\
&\sum_{i=1}^{m} \alpha_i y_i = 0 \quad i = 1, ..., m
\end{aligned} \tag{A.26}$$

5. **Verification of Slater's Condition A.0.2**

For the linearly separable case, we have the following optimization problem:

$$\begin{aligned}
\min_{\vec{w},b} &\frac{1}{2}\|\vec{w}\|^2 \\
s.t.\ \ &y_i(\vec{w} \cdot \vec{x_i} + b) \geq 1, \quad i = 1, ..., m
\end{aligned} \tag{A.27}$$

Note that the objective function $\frac{1}{2}\|\vec{w}\|^2$ is a convex function since the domain of function $f(\vec{w}) = \frac{1}{2}\|\vec{w}\|^2$ is $\mathbb{R}^n$, which is a convex set; and $\frac{1}{2}\|\vec{w}\|^2 = \frac{1}{2}\vec{w}^T A\vec{w}$, with $A = I$ is positive semidefinite. Moreover, to satisfy the Slater's condition (A.0.2) — the sufficient condition for $d^* = p^*$, we only need to find a feasible point $(\vec{w}, b)$, such that the constraints are strictly hold, i.e., $y_i(\vec{w} \cdot \vec{x_i} + b) > 1$. This is true since the training set is linearly separable, meaning that there must exist some $\vec{w}$ and b, such that the functional margins $y_i(\vec{w} \cdot \vec{x_i} + b) > 0$, for all $i = 1, ..., m$. Then by the scalability of $\vec{w}$ and b, we can rescale the chosen $\vec{w}$ and b to let $y_i(\vec{w} \cdot \vec{x_i} + b) > 1$ hold for all i. This completes the verification of Slater's conditions on linearly separable case.

For linearly non-separable case, the optimization problem becomes:

$$\begin{aligned}
\min_{\vec{w},b,\xi_i} &\frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{m} \xi_i \\
s.t.\ \ &y_i(\vec{w} \cdot \vec{x_i} + b) \geq 1 - \xi_i, \quad i = 1, ..., m \\
&\xi_i \geq 0, \quad i = 1, ..., m.
\end{aligned} \tag{A.28}$$

To verify the Slater's condition, notice that the set D defined in A.0.2 of this problem is still $\mathbb{R}^n$. According to the conditions, we only need to find a feasible set of values of $\vec{w}$, b and ξ_i's, such that all inequality constraints strictly holds. Actually, we can choose any $\vec{w}$ and b, and set $\xi_i = \max\{2 - y_i(\vec{w} \cdot \vec{x_i} + b), 1\}$, in this case, $y_i(\vec{w} \cdot \vec{x_i} + b) \geq 2 - \xi_i > 1 - \xi_i$, and $\xi_i \geq 1 > 0$. Thus this completes the verification of the Slater's conditions for the problem.

APPENDIX B

INFORMATION THEORY AND SIGNIFICANCE TEST

In machine learning and statistics, ***feature selection***, also known as ***variable selection***, ***attribute selection*** or ***variable subset selection***, is the process of selecting a subset of relevant features for use in model construction. The central assumption when using a feature selection technique is that the data contains many redundant or irrelevant features. Redundant features are those which provide no more information than the currently selected features, and irrelevant features provide no useful information in any context. Feature selection techniques are a subset of the more general field of feature extraction. Feature extraction creates new features from functions of the original features, whereas feature selection returns a subset of the features. Feature selection techniques are often used in domains where there are many features and comparatively few samples (or data points). Feature selection techniques provide three main benefits when constructing predictive models:

- improved model interpretability,
- shorter training times,
- enhanced generalization by reducing overfitting.

Feature selection is also useful as part of the data analysis process, as shows which features are important for prediction, and how these features are related.

In practice, there are many methods to perform feature selection on the training data set. Essentially, feature selection mainly produce a subset of the original training data set so that the selected subset can take the place of the original subset but still maintains the stability of the training performance. This process is called ***subset selection***. Subset selection evaluates a subset of features as a group for suitability. Subset selection algorithms can be broken into Wrappers,

Filters and Embedded. Wrappers use a search algorithm to search through the space of possible features and evaluate each subset by running a model on the subset. Wrappers can be computationally expensive and have a risk of over fitting to the model. Filters are similar to Wrappers in the search approach, but instead of evaluating against a model, a simpler filter is evaluated. Embedded techniques are embedded in and specific to a model.

In the filter method category, one of the most popular and standard metrics is called the ***information gain***. As a commonly used and accepted feature selection metric in the machine learning. The information gain is built up based on the concept of ***entropy***.

Let X be a discrete random variable, $X \in \mathscr{X}$, and probability density function $p(x) = \Pr\{X = x\},\ x \in \mathscr{X}$. We denote the probability density function by $p(x)$ rather than $p_X(x)$ for convenience. Thus, $p(x)$ and $p(y)$ refer to twp different random variables, and are in fact different density functions, $p_X(x)$ and $p_Y(y)$ respectively.

Definition B.0.10. *The entropy $H(x)$ of a discrete random variable X is defined by*

$$H(X) = -\sum_{x \in \mathscr{X}} p(x) \log(p(x)) \tag{B.1}$$

We will use the convention that the base of the $\log(p(x))$ equals to 2, and also note that, $0 \log 0 = 0$, which is easily justified by continuity since $x \log x \to 0$ as $x \to 0$. Thus adding terms of zero probability does not change the entropy.

Now we have some immediate consequences of the definition:

Lemma B.0.4. $H(X) \geq 0$.

Proof. $0 \leq p(x) \leq 1$ implies $\log(1/p(x)) \geq 0$. □

Lemma B.0.5. $H_b(x) = (\log_b a) H_a(X)$

Proof. $\log_b p = \log_b a \log_a p$

□

The second lemma of entropy enables us to change the base of the logarithm in the definition. Entropy can be changed from one base to another by multiplying by the appropriate factor.

By having the definition of entropy of a single random variable, we now extend the definition to a pair of random variables:

Definition B.0.11. *The joint entropy $H(X,Y)$ of a pair of discrete random variables (X,Y) with a joint distribution $p(x,y)$ is defined as:*

$$H(X,Y) = -\sum_{x\in\mathscr{X}}\sum_{y\in\mathscr{Y}} p(x,y)\log p(x,y) \tag{B.2}$$

We also define the conditional entropy of a random variable given another as the expected value of the entropies of the conditional distribution, averaged over the conditioning random variable.

Definition B.0.12. *If $(X,Y)\sim p(x,y)$, then the conditional entropy $H(Y|X)$ is defined as:*

$$\begin{aligned} H(Y|X) &= \sum_{x\in\mathscr{X}} p(x)H(Y|X=x) \\ &= -\sum_{x\in\mathscr{X}} p(x)\sum_{y\in\mathscr{Y}} p(y|x)\log p(y|x) \\ &= -\sum_{x\in\mathscr{X}}\sum_{y\in\mathscr{Y}} p(x,y)\log p(y|x) \end{aligned} \tag{B.3}$$

The naturalness of the definition of joint entropy and conditional entropy is exhibited by the fact that the entropy of a pair of random variables is the entropy of one plus the conditional entropy of the other. This is proved in the following theorem.

Theorem B.0.4. $H(X,Y) = H(X) + H(Y|X)$.

Proof.

$$\begin{aligned} H(X,Y) &= -\sum_{x\in\mathscr{X}}\sum_{y\in\mathscr{Y}} p(x,y)\log p(x,y) \\ &= -\sum_{x\in\mathscr{X}}\sum_{y\in\mathscr{Y}} p(x,y)\log p(x)p(y|x) \\ &= -\sum_{x\in\mathscr{X}}\sum_{y\in\mathscr{Y}} p(x,y)\log p(x) - \sum_{x\in\mathscr{X}}\sum_{y\in\mathscr{Y}} p(x,y)\log p(y|x) \\ &= -\sum_{x\in\mathscr{X}} p(x)\log p(x) - \sum_{x\in\mathscr{X}}\sum_{y\in\mathscr{Y}} p(x,y)\log p(y|x) \\ &= H(X) + H(Y|X). \end{aligned} \tag{B.4}$$

□

The entropy of a random variable is a measure of the uncertainty of the random variable; it is a measure of the amount of information required on the average to describe the random variable. Here, we formally introduce two related concepts — relative entropy and mutual information, which also known as ***information gain***.

The relative entropy is a measure of the distance between two distributions. In statistics, it arises as an expected logarithm of the likelihood ratio. The relative entropy $D(p\|q)$ is a measure of the inefficiency of assuming that the distribution is q when the true distribution is p. For example, if we knew the true distribution of the random variable. then we could construct a code with average description length $H(p)$. If, instead, we used the code for a distribution q, we would need $H(p) + D(p\|q)$ on the average to describe the random variable.

Definition B.0.13. *The relative entropy between two probability density functions $p(x)$ and $q(x)$ is defined as:*

$$D(p\|q) = \sum_{x\in\mathscr{X}} p(x)\log\frac{p(x)}{q(x)} \tag{B.5}$$

In the above definition, we use the convention that $0\log\frac{0}{q} = 0$ and $p\log\frac{p}{0} = \infty$.

We will then show that relative entropy is always non-negative and is zero if and only if $p = q$. However, it is not a true distance between distributions since it is not symmetric and does not satisfy the triangle inequality. Nonetheless, it is often useful to think of relative entropy as a distance between distributions.

Now, we introduce the information gain, which is a measure of the amount of information that one variable contains about another random variable. It sis reduction in the uncertainty of one random variable due the knowledge of the other.

Definition B.0.14 (***Information Gain***)**.** *Consider two random variables X and Y with a joint probability density function $p(x, y)$ and marginal probability density functions $p(x)$ and $p(y)$. The information gain $I(X;Y)$ is the relative entropy between the joint distribution and the product distribution $p(x)p(y)$, i.e.,*

$$\begin{aligned} I(X;Y) &= \sum_{x\in\mathscr{X}}\sum_{y\in\mathscr{Y}} p(x,y)\log\frac{p(x,y)}{p(x)p(y)} \\ &= D(p(x,y)\|p(x)p(y)) \end{aligned} \tag{B.6}$$

By the definition of the information gain, we have the following theorem:

Theorem B.0.5. $I(X;Y) = H(X) - H(X|Y)$

Proof. We can rewrite the definition of information gain $I(X;Y)$ as:

$$\begin{aligned} I(X;Y) &= \sum_{x,y} p(x,y)\log\frac{p(x,y)}{p(x)p(y)} \\ &= \sum_{x,y} p(x,y)\log\frac{p(x|y)}{p(x)} \\ &= -\sum_{x,y} p(x,y)\log p(x) + \sum_{x,y}\log p(x|y) \\ &= -\sum_{x} p(x)\log p(x) - (-\sum_{x,y} p(x,y)\log p(x|y)) \\ &= H(X) - H(X|Y) \end{aligned} \tag{B.7}$$

□

Thus the information gain $I(X;Y)$ is the reduction in the uncertainty of X due to the knowledge of Y.

By symmetry, it also follows that:

$$I(X;Y) = H(Y) - H(Y|X) \tag{B.8}$$

Thus X says as much about Y as Y says about X.

Since $H(X,Y) = H(X) + H(Y|X)$ in Theorem B.0.4, we have:

$$I(X;Y) = H(X) + H(Y) - H(X;Y) \tag{B.9}$$

Finally, we get:

$$I(X;X) = H(X) - H(X|X) = H(X) \tag{B.10}$$

Thus the information gain of a random variable with itself is the entropy of the random variable. This is the reason that entropy is sometimes referred to as self-information.

By summarizing the results above, we have the following properties given as a theorem:

Theorem B.0.6 (***Information gain and entropy***)**.**

$$I(X;Y) = H(X) - H(X|Y) \tag{B.11}$$

$$I(X;Y) = H(Y) - H(Y|X) \tag{B.12}$$

$$I(X;Y) = H(X) + H(Y) - H(X;Y) \tag{B.13}$$

$$I(X;Y) = I(Y;X) \tag{B.14}$$

$$I(X;X) = H(X) \tag{B.15}$$

Before we go ahead to prove the non-negativity of information gain and conditioning reduces entropy property, we firstly need to introduce an important tool:

Theorem B.0.7 (***Jensen's Inequality***)**.** *If f is a convex function and X is a random variable, then:*

$$Ef(X) \geq f(EX) \tag{B.16}$$

Moreover, if f is strictly convex, then equality in B.16 implies that $X = EX$ with probability, i.e, X is a constant.

Proof. We prove this for discrete case distribution by induction on the number of density points.

For a two density point distribution, the inequality becomes:

$$p_1 f(x_1) + p_2 f(x_2) \geq f(p_1 x_1 + p_2 x_2) \tag{B.17}$$

which follows directly from the definition of convex functions. Suppose the theorem is true for distributions with $k-1$ density points. Then writing $p'_i = p_i/(1-p_k)$ for $i = 1, 2, ..., k-1$, we have:

$$\begin{aligned}\sum_{i=1}^{k} p_i f(x_i) &= p_k f(x_k) + (1-p_k)\sum_{i=1}^{k-1} p'_i f(x_i) \\ &\geq p_k f(x_k) + (1-p_k) f(\sum_{i=1}^{k-1} p'_i x_i) \\ &\geq f(p_k x_k + (1-p_k)\sum_{i=1}^{k-1} p'_i x_i) \\ &= f(\sum_{i=1}^{k} p_i x_x)\end{aligned} \tag{B.18}$$

where the first inequality follows from the induction hypothesis and the second follows from the definition of convexity. The proof can be extended to continuous distribution by continuity arguments.

□

We now us these results to prove some of the properties of entropy and relative entropy. The following theorem is of fundamental importance.

]

Theorem B.0.8 (***Information Inequality***)**.** *Let $p(x)$, $q(x)$, $x \in \mathscr{X}$, be two probability density functions. Then*

$$D(p\|q) \geq 0 \tag{B.19}$$

with equality if and only if

$$p(x) = q(x), \ \ \forall x \in \mathscr{X} \tag{B.20}$$

Proof. Let $A = \{x : p(x) > 0\}$ be the support set of $p(x)$, Then

$$
\begin{aligned}
-D(p\|q) &= -\sum_{x\in A} p(x)\log\frac{p(x)}{q(x)} \\
&= \sum_{x\in A} p(x)\log\frac{q(x)}{p(x)} \\
&\leq \log\sum_{x\in A} p(x)\frac{q(x)}{p(x)} \\
&= \log\sum_{x\in A} q(x) \\
&\leq \log\sum_{x\in \mathscr{X}} q(x) \\
&= \log 1 \\
&= 0,
\end{aligned}
\tag{B.21}
$$

where the first inequality follows from Jensen's inequality. Since $\log t$ is a strictly concave function of t, we have the equality if and only if $q(x)/p(x) = 1$ everywhere, i.e., $q(x) = p(x)$. Hence we have $D(p\|q) = 0$ if and only if $p(x) = q(x)$ for all x. □

Based on the theorem above, we have:

Corollary B.0.1 (***Non-negativity of information gain***)**.** *For any two random variables, X, Y,*

$$
I(X;Y) \geq 0 \tag{B.22}
$$

with equality if and only if X and Y are independent.

Proof. $I(X;Y) = D(p(x,y)\|p(x)p(y)) \geq 0$, with equality if and only if $p(x,y) = p(x)p(y)$, i.e., X and Y are independent.

□

Also, we now have the important result given by the following theorem:

Theorem B.0.9 (Conditioning reduces entropy)**.**

$$H(X|Y) \leq H(X) \tag{B.23}$$

with equality if and only if X and Y are independent.

Proof. $0 \leq I(X;Y) = H(X) - H(X|Y)$ □

Intuitively, the theorem says that knowing another random variable Y can only reduce the uncertainty in X. Note that this is true only on the average. Specifically, $H(X|Y = y)$ may be greater than or less than or equal to $H(X)$, but on average $H(X|Y) = \sum_y p(y)H(X|Y = y) \leq H(X)$.

Now, the Theorem B.0.9 gives the support and criteria for performing the feature selection: The larger of $I(X;Y)$, means that the more uncertainty is decreased by given the feature Y, that is, the more information we gain to clear the unpredictability of the training data set. We want to determine which attribute in a given set of training feature vectors is most useful for discriminating between the classes to be learned. Information gain tells us how important a given attribute of the feature vector is. Thus, by calculating the information gain of each individual feature, we can rank the features by their information gain vaue, thus to select the ones with high information gain value.

A paired sample t-test is used to determine whether there is a significant difference between the average values of the same measurement under two different conditions. Both measurements are made on the same data sample and the test is based on the paired difference between these two values. The usual null hypothesis is that the difference in the mean values is zero. In the case of this paper, two learning methods are performed on the same training data set and some specified model performance measurements are obtained. We want to investigate whether the average performance of one of the models is significantly better than the other one, under the same performance measurement. On the other hand, if we can construct a baseline model that serve as the benchmark for evaluating the model performance, then we can compare our model performance with the benchmark model. Further more, if the benchmark model is proved to be generally outperforms the randomized model — which randomly assigns label to an unseen data point with probability $\frac{1}{k}$

given there are total k classes, then this will further prove that our model performance is statistical better than the randomized model with specified confidence level α.

Suppose we want to compare the performance of two classifiers $\mathcal{C}_1$ and $\mathcal{C}_2$ by using the same performance measurement. Also suppose the data set size $\|T\| = n$, then we split T into training set with size of n_1 data points, and n_2 is the size of testing set. Thus $n_1 + n_2 = n$.

Then we first perform N trials of training and testing by using the two classifiers and obtain two series of performance measurement data:

For classifier $\mathcal{C}_1$, the performance measurement values result from N trials are:

$$\mathscr{P}_1 = \{\mu_1, \mu_2, \mu_3, ..., \mu_N\} \tag{B.24}$$

For classifier $\mathcal{C}_2$, the experiments are performed by using the same set up of each set corresponding to the experiments done by using classifier $\mathcal{C}_1$. Also, a series of data under the same measurement are obtained through n tests:

$$\mathscr{P}_2 = \{\nu_1, \nu_2, \nu_3, ..., \nu_N\} \tag{B.25}$$

Then the series of difference between $\mathscr{P}_1$ and $\mathscr{P}_2$ is calculated to denote the performance measurement difference between the two classifiers on each trial, thus we have:

$$\begin{aligned} &\mathscr{D} = \{d_1, d_2, d_3, ..., d_N\} \\ &\text{where} \quad d_i = \mu_i - \nu_i; \end{aligned} \tag{B.26}$$

Then, the mean of the performance difference and the standard deviation are computed as:

$$\mu_d = \frac{1}{N} \sum_{i=1}^{N} d_i \tag{B.27}$$

and

$$s_d = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(d_i - \mu_d)^2} \tag{B.28}$$

Having the average of performance measurement given by the paired learning model as μ_d, the null hypothesis for a paired t-test is formed:

Given significance level α,

$$H_0 : \mu_d = 0 \tag{B.29}$$

and the alternative hypothesis is:

$$H_1 : \mu_d \neq 0 \tag{B.30}$$

If we assume n differences $d_i = \mu_i - \nu_i$ are drawn independently from from a normal distribution, then we can apply the student's t-test, by computing the statistic:

$$t = \frac{\mu_d}{s_d/\sqrt{N}} \tag{B.31}$$

Under the null hypothesis, this statistic has a t-distribution with $N-1$ degrees of freedom.

However, there are many potential drawbacks of this approach. First, the individual difference d_i may not have a normal distribution. Secondly, due to the overlapping of the training set and testing set, it may prevent this statistical test from obtaining a good estimate of the amount of

variation. Thus in order to address the bias embedded with this statistic, a new t-statistic is suggested by Nadeau(1999) [60] and commonly accepted in the machine learning practice field.

The statistic is called ***corrected resample t-value***:

$$t = \frac{\mu_d}{s_d \sqrt{\frac{1}{J} + \frac{n_2}{n_1}}} \tag{B.32}$$

where J is the number of independent sample data sets, and the degree of freedom is used as $N-1$, n_1 is the number for sample training and n_2 is the number of sample for testing. The ratio n_2/n_1 is called the test/train ratio.

After getting the t-value, the p-value is computed by using the $\mathbb{F}$-distribution (Phillips (1982) [66]):

$$p = \mathbb{F}(t^2, 1, N-1)$$

where $\mathbb{F}$ is the cumulative distribution function of $\mathbb{F}$-distribution.

Given a confidence level $\alpha = 0.05$, by comparing the p-value of the t-test , we can conclude that the null hypothesis is rejected when $p < \alpha = 0.05$, i.e., the alternative hypothesis is accepted. In the case of comparing the performance of two learning model, we can say that one is statistical significant better (worse) than the other. Specifically:

- If $p > \alpha$, then null hypothesis H_0 is accepted;
- If $p \leq \alpha$, then null hypothesis H_0 is rejected;

  If $\mu_d > 0$, then under the measurement of performance, classifier $\mathcal{C}_1$ is statistically better than classifier $\mathcal{C}_2$ with $(1-\alpha)$ confidence;

  If $\mu_d < 0$, then under the measurement of performance, classifier $\mathcal{C}_2$ is statistically better than classifier $\mathcal{C}_1$ with $(1-\alpha)$ confidence;

Now, with the tool provided by the paired t-test, we can follow the procedure to measure the difference of two classifiers. Further more, as the philosophy discussed above, we can determine whether the performance of a specified classifier is statistically better or worse than a random generated classifier. In the sense that we can justify whether the model designed obtains the desired performance just by coincidence.

APPENDIX C

RELATED FLOW CHARTS AND PSEUDOCODES

1. **Flow Charts Pseudocodes for Multi-class SVMs Prediction**

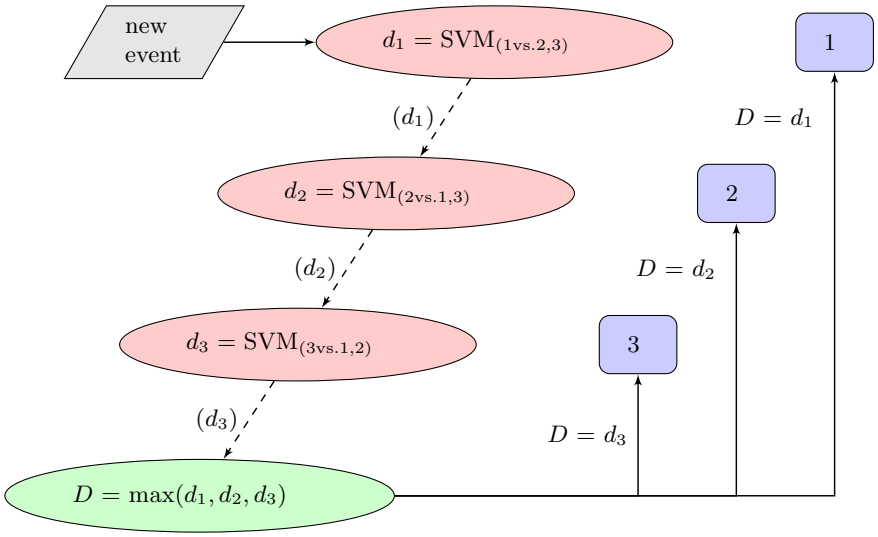

Figure C.1: Flow chart for one-against-all prediction procedure. The red circles represent the SVMs models that perform the classification between two separate categories. The blue boxes are the final labelled classed with the class index. The green circle represents the module that determines the maximum of distances (d_1, d_2 or d_3) for each data point to the hyperplanes that differentiate each class with the rest of the classes.

- i, j, k – the index of class, i, j, k could be 1, 2, 3, that denote upward, downward and stationary class, respectively.
- $\text{SVM}_{(\text{i vs. j, k})}$ – the SVM model classifies the event $\vec{x}$ into the class of i or (j, k), i.e. NOT i by using the decision function 3.29. and returns the distance (d_i) between the data point $\vec{x}$ and the separating hyperplane of class i and NOT class i. The function $\text{SVM}_{(\text{i vs. j, k})}$ indicates that the event $\vec{x}$ belongs to class i if $d_i > 0$, otherwise belongs to either class j or k (i.e. NOT i) if $d_i < 0$.
- $D = \max(d_i, d_j, d_k)$ – the function to determine the maximum distance D among d_i, d_j and d_k. The event $\vec{x}$ will be classified into the class i if $D = d_i$. The distance of d_i is determined by using the decision function 3.29. Notice that the values of d_i, d_j and d_k can be both positive or negative, hence D can be positive or negative depending on the

combination of (d_i, d_j, d_k). Particularly, if two of them have identical values, that is, the tie situation, we can randomly assign the label to the data point to either one of those tie classes, or just specify the label directly according to some restriction principles related to practical situation.

- Decision function for d_i in $\text{SVM}_{(\text{i vs. j, k})}$:

$$\begin{aligned} d_i = \vec{w^*} \cdot \phi(\vec{x}) + b^* &= (\sum_{l=1}^{m} \alpha_l^* y_l \phi(\vec{x_l})) \cdot \phi(\vec{x}) + b^* \\ &= \sum_{l=1}^{m} \alpha_l^* y_l \phi(\vec{x_l}) \cdot \phi(\vec{x}) + b^* \\ &= \sum_{l=1}^{m} \alpha_l^* y_l \kappa(\vec{x_l}, \vec{x}) + b^* \end{aligned} \tag{C.1}$$

- Pseudocodes for the prediction procedure:

main $(\vec{x})$ {

$d_1 = \text{SVM}_{(1\text{vs.}2,3)}(\vec{x})$;
$d_2 = \text{SVM}_{(2\text{vs.}1,3)}(\vec{x})$;
$d_3 = \text{SVM}_{(3\text{vs.}1,2)}(\vec{x})$;
$D = \max(d_1, d_2, d_3)$;

If $(D = d_1)$ {
    $\vec{x}$ belongs to class 1;
}
else if $(D = d_2)$ {
    $\vec{x}$ belongs to class 2;
}
else if $(D = d_3)$ {
    $\vec{x}$ belongs to class 3;
}

}

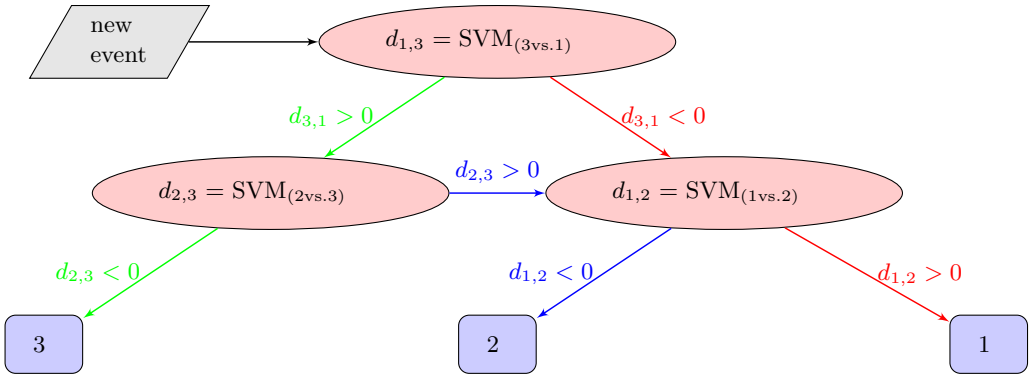

Figure C.2: Flow chart for one-against-one prediction procedure.The red circles represent the SVMs models that perform the classification between each pair of classes. The blue boxes are the final labelled classed with the class index. Each new event that comes into the model goes through this majority vote process and eventually be labelled into one of the 3 classes. This figure shows parts of the possible paths to assign the labels.

- i, j, k – the index of class, i, j, k could be 1, 2, 3, that denote upward, downward and stationary class in each corresponding SVM model.
- $\text{SVM}_{(\text{i vs. j})}$ – the SVM model classifies the event $\vec{x}$ into the class of i or j, given by the decision function 3.29.
- $d_{i,j}$ – the distance of $vecx$ to the separating hyperplane of class i and j; $d_{i,j} > 0$ means $\vec{x}$ belongs to class i, else if $d_{i,j} < 0$ means $\vec{x}$ belongs to class j;
- v_i – number of votes for class i; if $d_{i,j} > 0$, v_i is increased by 1, otherwise v_j is increased by one;
- $V = \max(v_i, v_j, v_k)$ – the maximum of number of votes for each class. If $V = v_i$, then $\vec{x}$ will be labelled as class i;
- The three SVMs models work coordinately and apply the majority vote method by classifying the event $\vec{x}$ into one of the three classes. However, one can notice that the number of votes sometimes can be tie.
- In the Figure C.2, the green, red, and blue paths are good voting processes that can determine the label of $\vec{x}$ without tie situations. But the path, for instance, $d_{3,1} > 0 \rightarrow d_{2,3} > 0 \rightarrow d_{1,2} > 0$ ends up with the situation that $v_1 = v_2 = v_3 = 1$, which is the tie situation with each class gets one vote;
- $D = \max(d_{i,j}, d_{j,k}, d_{k,i})$ – the function that compares all the distances for $\vec{x}$ to three different hyperplanes. This function is used when the tie situation comes up. The label of $\vec{x}$ is i, if $D = d_{i,j}$. Notice that, in this situation, all of $d_{i,j}, d_{j,k}, d_{k,i}$ are negative numbers;

- Decision function for $d_{i,j}$ in $\text{SVM}_{(\text{i vs. j})}$:

$$\begin{aligned} d_{i,j} = \vec{w^*} \cdot \phi(\vec{x}) + b^* &= (\sum_{l=1}^{m} \alpha_l^* y_l \phi(\vec{x_l})) \cdot \phi(\vec{x}) + b^* \\ &= \sum_{l=1}^{m} \alpha_l^* y_l \phi(\vec{x_l}) \cdot \phi(\vec{x}) + b^* \\ &= \sum_{l=1}^{m} \alpha_l^* y_l \kappa(\vec{x_l}, \vec{x}) + b^* \end{aligned} \tag{C.2}$$

- Pseudocodes for the prediction procedure:

```
main (x⃗) {
    d_{1,2} = SVM_{(1vs.2)}(x⃗);
    d_{2,3} = SVM_{(2vs.3)}(x⃗);
    d_{3,1} = SVM_{(3vs.1)}(x⃗);
    D = max(d_{1,2}, d_{2,3}, d_{3,1});

    for (d_{i,j}: (i, j) = (1, 2), (2, 3), (3, 1)) {
        If (d_{i,j} > 0) {
            ++v_i;
        }
        else {
            ++v_j;
        }
    }
    V = max(v_i, v_j, v_k);
    if (No tie in V && V = v_i) {
        x⃗ belongs to class i;
    }
    else if (D = d_{i,j}) {
        x⃗ belongs to class i;
    }
}
```

2. **Flow Charts and Pseudocodes for Data Simulation Procedure**

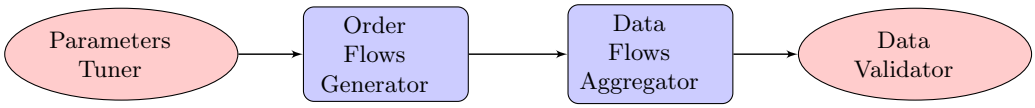

Figure C.3: Order book data simulation work flow. The procedure includes 4 phases of work performed by each module respectively. The modules are corresponding to the phases introduced in the section 4.5

The following pseudocodes describe the general programming processes on the modules of *Order Flows Generator* and *Data Flows Aggregator*.

- *order_book* – the real order book data that used for the parameters extraction and configuration;
- *message_book* – the real message book data that used for the parameters extraction and configuration;
- i – the index of window that are being simulated;
- λ_i – the parameters used in simulating the window i; λ_i includes the market order arrival rates (bid/ask), cancellation rates (bid/ask on each tick distance to opposite best quote), limit order arrival rates (bid/ask and on each tick distance to opposite best quote);
- N – the number of window that need to be simulated; each window is a duration of time with fixed length;
- P_i – the fluctuation range for the prices on each tick level in simulated window i;
- V_i – the fluctuation range for the volume on each tick level in simulated window i, which includes the cancellation volume, limit order volume and execution volume ranges;
- parameters_tuner() – the function that takes real order book and message book as input, and outputs the statistics for the parameters (λ_i, P_i, V_i, etc) used in the simulation; on the other hand, all the parameters can be tuned by configured this function;
- win_simulator() – function that simulates a single window i with specified parameters and returns the organized order flows;
- *order_i* – the output of function win_simulator(), in the format of ordered events and corresponding entries of order book;
- order_aggregator() – function that takes the *order_i* as input, and incorporate s all the *order_i*'s to a timely manner such that a whole complete order book data with the format as the real data is constructed;

- *submission_i, execution_i, cancellation_i* – the output of function event_generator(), multi-dimensional array stores tick distance, the happened time, the price and volume corresponding to each event type;
- event_generator() – function that generates the events of order submission, execution and cancellation during the fixed window duration, each event is output to a data structure associate with tick distance, event type, the happened time, the price and volume. The execution processes are controlled by 2 independent Poisson processes with given intensities, the limit order arrival rates and cancellation rates are calibrated from the real data and given by the output of function parameters_tuner(), such that each tick distance to the opposite best quote has both arrival processes and cancellation process simultaneously;
- event_processor() – function that takes *submission_i, execution_i, cancellation_i* as input, and organizes the events into orders that following time sequence. The output of the function, *order_i*, is the data structure stores exactly the information of all orders happen in the window i in the format of order book data;

main (*order_book*, *message_book*) {

(λ_i, P_i, V_i) = parameters_tuner (*order_book*, *message_book*);

for ($i = 0$; $i < N$; $++i$) {

order_i = win_simulator(i, λ_i, P_i, V_i);

order_aggregator(order_*i*);

}

return ();

}

sub win_simulator(i, λ_i, P_i, V_i) {

(*submission_i, execution_i, cancellation_i*) = event_generator(λ_i, P_i, V_i);

order_i = event_processor(*submission_i, execution_i, cancellation_i*);

return(*order_i*);

}

BIBLIOGRAPHY

[1] Aldridge, Irene *High-frequency Trading: A Practical Guide to Algorithmic Strategies and Trading Systems* Wiley, ISBN 978-0-470-56376-2 2009.

[2] Andersen, Allanm, Cont, Rama, Ekaterina, Vinkovskaya *A point process model for the high-frequency dynamics of a limit order book* Financial Engineering Report 2010.

[3] Avellaneda M. and S. Stoikov *High-frequency Trading in a Limit Order Book* Quantitative Finance, 8(3), 217-224 2008.

[4] Biais, B., D. Martimort and J. Rochet *Competing Mechanisms in a Common Value Environment* Ecnonmetrica 68, 799-837. 2000.

[5] Bishop, C.M. *Pattern Recognition and Machine Learning* Springer, New York 2006.

[6] Blazejewski, A, Coggins, R. *Application of self-organizing maps to clustering of high-frequency financial data* Conferences in Research and Practice in Information Technology 2006.

[7] Blazejewski, A, Coggins, R. *A local non-parametric model for trade sign inference* Physica A: Statistical Mechanics and Its Applications, Vol 348, No.1481-495 2005.

[8] Bottou, L Cotes ,C and Vapnik *Comparison of Classifiers Methods: A Case Study in Handwriting Digit Recognition* Proc. Int. Conf. Pattern Recognition. 1994

[9] Bouchaud, Jean Philippe, Doyne Farmer, Fabrizio Lillo *How markets slowly digest changes in supply and demand* Handbook of Financial Markets: Dynamics and Evolution Elsevier: Academic Press, 57-160. 2008.

[10] Bouchaud, J, M. Mezard,M. Potters *Statistical Properties of Stock Order Books: Empirical Results and Models* Quantitative Finance 2 251-256 2002.

[11] Bovier, A., J. Cerny, o. Hryniv *The Opinion Game: Stock Price Evolution From Microscopic Market Modeling* Int. J. Theor.Appl. Finance 9 91-111. 2006.

[12] Boyd, Stephen., Vandenberghe, Lieven *Covex Optimization* Cambridge UP 2004

[13] Cao, L *Support Vector Machines Experts for Time Series Forecasting* Neurocomputing, 51,321-339. 2003.

[14] Cartea, A and S, Jaimungal *Risk Metrics and Fine Tuning of High Frequency Trading Strategies* SSRN: http://ssrn.com/abstract=2010417, 2012

[15] Cartea, A and J, Ricci *Buy Low Sell High: A High Frequency Trading Perspective* Available at SSRN: http://ssrn.com/abstract=1964781 2011.

[16] Chalup, S. K., Mitschele, A. *Kernel Methods in Finance* Handbook on Information Technology in Finance(pp. 655-687). Berlin:Springer 2008.

[17] Chen, Su., Hu, Chen., Zhou., Yijia *Order Book Simulator and Optimal Liquidation Strategies* Investment Practice Project, Stanford University. 2010.

[18] Chiarella, Carl., Iori, Giulia *A Simulation Analysis of The Microstructure of Double Auction Markets* Quantitative Finance 2002.

[19] Cont, R., Larrard, A. *Order book dynamics in liquid markets: limit theorems and diffusion approximations* 2012

[20] Cont, R., Stoikov, S., Talreja, R. *A Stochastic Model for Order Book Dynamics*. Operations Research, 2010.

[21] Cover, Thomas M., Thomas, Joy A. *Elements of Information Theory* John Wiley & Sons. Inc. 2006.

[22] Cortes, C., Vapnik, V. *Support Vector Networks*. Machine Learning(pp.273-297), 1995.

[23] Crammer,K Singer, Y *On the algorithmic implementation of multi-class kernel-based vector machines* Journal of Machine Learning Research 265-292 2001

[24] Cristianini, N., Shawe-Taylor, J. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods* New York, NY, USA: Cambridge University Press. 2000.

[25] Daniel, Gilles *Asynchronous Simulations of A Limit Order Book* University of Manchester, Thesis 2006.

[26] Dietterich, Thomas G. *Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms* Neural Computation. 1998.

[27] Eiesland, Erik *Simulating the Order Book — A Tool To Discover Trading Strategy* Ostfold University, Thesis 2011.

[28] Farmer,J.D.,L.Gilemot, F.Lillo, S. Mike, A.Sen *What Really Causes Large Price Changes?* Quantitative Finance 4 383-397 2004

[29] Fletcher, T., Shawe-Taylor, J. *Multiple Kernel Learning with Fisher Kernels for High Frequency Currency Prediction*. Computational Economics, 2012.

[30] Fletcher, T., Redpath, F., D'Alessandeo, J. *Machine Learning in FX carry Basket Prediction.* Proceedings of the international conference of financial engineering (Vol. 2, pp.1371-1375) 2009.

[31] Foucault, T., O. Kadan, E. Kandel. *Limit Order Book as A Market for Liquidity.* Review of Financial Studies 18(4) 1171-1217 2005.

[32] Foucault, T. *Order Flow Composition and Trading Costs in a Dynamics Limit Order Market* Journal of Financial Markets 2, 99-134. 1999.

[33] Guo, Ted Xiao. *An Agent-based Simulation of Double Auction Markets* University of Toronto, Thesis 2005.

[34] Grant, Jeremy *High-frequency Trading: Up Against a Bandsaw.* Financial Times, 2010.

[35] Haldane, Andrew G Bank of England *Patience and Finance.* 2010.

[36] Harris, L. *Trading & Exchanges: Market Microstructure for Practitioners* Oxford University Press. 2003.

[37] Hasbrouk, J *Empirical Market Microstructure: The institutions, economics, and econometrics of securities trading.* Oxford, USA: Oxford University Press 2006.

[38] Hastie,M Tibshirani,R *Classification by pairwise coupling* The Annals of Statistics 26(2), 451-471. 1998

[39] Haverkort, B., *Performance of Computer Communication Systems: A Model-Based Approach* John Wiley & Sons, New York 1998.

[40] Hawkes, A.G., *Spectra of some self-exciting and mutually exciting point processes* Biometrika, Vol. 58. 1971

[41] Hazarika, N., Taylor, J. G. *Predicting Bonds Using the Linear Relevance Vector Machines* Berlin:Springer 2002.

[42] Hollifield, B., R. A. Miller, P. Sandas *Empirical Analysis of Limit Order Markets* Review of Economic Studies 71(4) 1027-1063 2004

[43] Huang, H., Kercheval, A. *A Generalized Birth-Death Stochastic Model for High-frequency Order Book Dynamics.* Quantitative Finance, 2011.

[44] Humer, Caroline. *New Age Securities Firms to Testify Against Central Limit Order Book* TheStreet.com 2000

[45] Jacobs, B., Levy, K., Markowitz, H. *Financial Market Simulation* The Journal of Portfolio Management, 3oth Anniversary Issue 2004.

[46] Joachims,T *Making large-scale support vector machine learning practical* Schoelkopf, B., Burges, C., Smola, A. (Eds.), Advances in Kernel Methods Support Vector Learning. MIT Press, Cambridge, MA, pp. 169184 (Chapter 11). 1999.

[47] Jondeau, E., Perilla, A., Rockinger, G. M. *Optimal Liquidation Strategies in Illiquid Markets.* SSRN eLibrary, 2008.

[48] Kim, K *Financial Time Series Forcasting Exchange Rates Using Support Vector Machines* Neurocomputing, 55,307-319. 2003.

[49] Knerr,S., Personnaz, L,. and Dreyfus, G., *Single-layer learning revisited: A Stepwise Procedure for Building and Training a Neural Network* Neruocomputing Algorithms, Architectures and Applications. Springer, 1990.

[50] Kuan,Chung-Ming, Liu, Tung *Forecasting Exchange Rates Using Feedforward and Recurrent Neural Networks* Journal of Applied Econometrics,10(4),347-64. 1995.

[51] Linnainmaa, J., Rosu, I. *Time Series Determinants of Liquidity in a Limit Order Market.* SSRN eLibrary, 2008.

[52] Lo, A., MacKinlay, A., and Zhang J. *Econometrics Models of Limit-order Executions.* SSRN eLibrary, 1997.

[53] Lorenz, Julian., Osterrieder *Simulation of A Limit Order Driven Market* Institutional Investor Journals 2008.

[54] Lu, C. *High Frequency Trading: Price Dynamics Models and Market Making Strategies* Electrical Engineering and Computer Sciences, University of California at Berkekey 2012

[55] Luckock, Hugh. *A Steady-State Model of the Continuous Double Auction* Quantitative Finance 3 385-404. 2003.

[56] Luckock, Hugh., Skipper, Max *Modeling the Limit Order Book: Analytic Results and Monte Carlo Simulations* 2001.

[57] Markwat, T., Kole, E. *Contagion as Domino Effect in Global Stock Markets* Financial Globalisation, Risk Analysis and Risk Management. Volume 33, Issue 11. 2008.

[58] Maslov,S., M.Mills. *Price Fluctuation from the Order Book Perspective-Empirical Facts and A Simple Model* PHYSICA A299 234 2001.

[59] Mendelson, Haim *Market behavior in a cleaning house* Econometrica 1505-1524. 1982.

[60] Nadeau, Claude., Bengio, Yoshua. *Inference of the Generalization Error* Scientific Series, Motreal 1999.

[61] Ng, Andrew *Lecture notes on machine learning* Stanford University, 2012.

[62] Olson, David L and Delen, Dursun *Advanced Data Mining Techniques* Springer, 1st Edition, ISBN 3-540-76916-1 2008.

[63] Parlour, C., Seppi, D. *Limit Order Market: A survey.* Handbook of financial intermediation and banking, 2008.

[64] Parlour, Ch. A *Price Dynamics in Limit Order Markets.* Review of Financial Studies 11(4) 789-816. 1998.

[65] Perez-cruz, F., Afonso-rodriguez, J., Giner, J. *Estimating Garch Models Using Support Vector Machines* Quantitative Finance 3(3), 163-172 2003.

[66] Phillips, P.C.B. *The True Characteristic Function of the F-distribution* Biometrika, 1982.

[67] Platt, John C. *Fast Training of Support Vector Machines Using Sequential Minimal Optimization* Advances in kernel methods,185-208. volume 3 MIT Press, Cambridge, MA, USA. 1999.

[68] Popper, Nathaniel *Times Topic: High-frequency Trading* The New York Times, 2012, Dec 12.

[69] Qi, Z. *Simulation of Multi-Agents In Financial Market* Quantitative Finance, Internship Report 2009

[70] Radcliffe, Roberts C. *Investments: Concepts, Analysis, Strategy* Addison-Wesley Educational Publishers, Inc, ISBN 0-673-99988-2. 1997.

[71] Rosu,I., *A Dynamic Model of the Limit Order Book* University of Chicago, working paper. 2005

[72] Shabolt, J. Taylor, J. G. *Neural Networks and the Financial Markets: Predicting, combining and portfolio optimization* London:, UK:Springer 2002.

[73] Shaw-Taylor, J. Cristianini, N *Kernel Methods for Pattern Analysis* Cambridge University Press, Cambridge. 2004.

[74] Shek, H *Modeling High Frequency Market Order Dynamics Using Self-Excited Point Process.* SSRN: http://ssrn.com/abstract=1668160 2011.

[75] Smith, E., J.D. Farmer,L. Gillemot, S. Krishnamurthy *Statistical Theory of the Continuous Double Auction* Quantitative Finance 3(6) 481-514. 2003.

[76] Steinwart, Ingo., Christmann Andreas *Support Vector Machines* Springer, Information Science and Statistics 2008.

[77] Tay, F., Cao, L. *Modified Support Vector Machines in Financial Time Series Forecasting* Neurocomputing, 48,847-861 2002.

[78] Tay, F., Cao, L. *Application of Support Vector Machines in Financial Time Series Forecasting* Omega, 29 309-317 2001.

[79] Tino, P. Nikolaev, N., Yao, X. *Volatility Forecasting with Sparse Bayesian Kernel Models* International Conference on Computational Intelligence in Economics and Finance 2005.

[80] Walczak, S. *An Empirical Analysis of Data Requirements for Financial Forecasting with Neural Networks* Journal of Management Information Systems 17(4), 203-222 2001.

[81] Wilmott, Paul *Paul Wilmott on Quantitative Finance* Wiley, 2008

[82] Zheng, B., Moulines, E., Abergel, F. *Price jump prediction in limit order book* Journal of Mathematical Finance, 3, 242-255. 2013.

BIOGRAPHICAL SKETCH

Yuan Zhang is currently a PhD candidate majoring in Financial Mathematics at Department of Mathematics at Florida State University in Tallahassee, Florida, USA. His research interests include massive financial data analysis, data mining, high-frequency trading strategies and limit order book dynamics modeling with computer simulation and state-of-art machine learning techniques. He holds a Master in Mathematics from FSU and a Bachelor degree of Quantitative Finance with minor in Applied Mathematics from Shanghai Jiaotong University, China, 2008. He is the member of American Mathematical Society.