Modelling and Simulation for the Analysis of Securities Markets

Rui Hu[1,2], Vadim Mazalov[1,3] and Stephen M. Watt[1]

[1] University of Western Ontario, London, ON, Canada
{rhu8, vmazalov, Stephen.Watt}@uwo.ca
[2] Quantica Trading, Kitchener, ON, Canada
rui@quanticatrading.com
[3] Amazon Canada, Toronto, ON, Canada
mazalovv@amazon.com

Abstract

Many financial markets today are dominated by automated high-frequency trading. According to some studies, this has accounted for more than half the volume of US equity markets in recent years. High-frequency trading strategies typically adopt powerful computers and communications infrastructure and a variety of algorithms to process a large number of orders at high speed, attempting to profit sometimes a fraction of a cent on every trade. While this has led to significant theoretical and applied research, the area still presents many important challenges. These arise both in the strategy modelling phase, where accurate and efficient prediction of the price movement of securities is required, and in the evaluation phase, where strategies must be examined in a variety of market conditions before being launched in real markets. We investigate these two areas. Our modelling approach is based on clustering in a space of technical indicators, using a weighted Euclidean distance in a manner similar to certain handwriting recognition algorithms. Our evaluation environment is a market simulator that uses historical data or live data and agents to reproduce the fine-grained dynamics of financial markets. This paper outlines our approach to modelling and simulation and how they work together.

1 Introduction

Securities markets have experienced a dramatic transformation since the early 2000s. With the advancements in computer technology, traders no longer need to buy and sell securities using hand signals. Instead, their trading activities are automated by sophisticated computer algorithms, making it possible to make effective decisions to promptly react to every single market event. Powerful computers and ultra-low latency networks are also employed to accelerate data processing and message delivery, enabling them to turn over security positions very quickly in order to profit sometimes a fraction of a cent on every trade. This type of trading is commonly referred as *high-frequency trading*. Despite its increasing use and popularity [1, 3, 5, 6], high-frequency trading today still faces many critical challenges. Among these, we are particularly interested in two sub-problems pertaining to aspects of strategy modelling, where accurate and efficient prediction of securities' price movement is required, and strategy evaluation, where strategies must be examined in a variety of market conditions before being launched in real markets. In this article we present an overview of these problems and summarize our previously published solutions [4, 7].

In order to manage risk exposure and optimize profit, it is important to be able to accurately predict price movement of assets. This is a complex problem in that there are a considerable number of factors that may affect price in securities markets. Moreover, these factors themselves may have complicated cause-and-effect relationships, resulting in a variety of potential outcomes. We are interested in the problem of providing predictions of the price movement in the short term, as opposed to the long term, since the number of contributing factors is smaller, and their values are easier to measure. Also, the

possibility of a impact by outside factors, e.g. an unscheduled news release, is lower and has less effect in the short-term. In particular, we present a trading model that can provide accurate and efficient short-term prediction of *one* change in the price of an asset. Our method monitors the market and sets up a space of observations, then measures how close the real-time data is to the recorded observations. Predictions are made based on numerical indicators computed from data received from the market. This work has been reported in [7].

At the same time, trading strategies must be evaluated for correctness and performance before using them real market. This in practice is carried out through simulators which significantly rely on real-time market data or historical data. While this can provide traders with valuable information, there are a number of pitfalls. First, live market data is not always available, which restricts the use of simulators to certain market hours. In addition, the trading strategies tested do not have any impact to the market as they can only follow the trend and their orders are simply executed based on the current market conditions. Similar issues also exist in back-testing approaches. Last, but not least, existing simulators typically do not provide a standard protocol for interaction with users. Instead, they require skills in specific programming languages and demand trading strategies to be implemented on top of proprietary Application Program Interfaces (APIs). This can restrict the evaluation of trading strategies to a single simulation environment. To address this problem, we present a simulator that can support market simulation research and is suitable for strategy evaluation. The simulator is independent of any particular data feed and can provide a realistic testing environment by reproducing certain phenomena of a real market. Multiple users can connect to the simulation server at the same time, allowing them to not only assess the viability of their trading strategies using pre-defined market conditions, but also to create very specific ones that suit their needs. This work has been reported in [4].

The remainder of the article is organized as follows. In Section 2 we describe an example of prediction model for high-frequency trading. Section 3 presents a simulator that is suitable for evaluation of trading strategies. In Section 4 we conclude the article with a brief discussion.

2 A Trading Model

Technical Indicators The number of possible technical indicators that can be extracted from the stock market is potentially overwhelming so careful manual or algorithmic selection of indicators is important. Indicators should not be redundant and should sufficiently describe the asset at the time of an event. In order to limit the available indicators to a manageable number and predict a price change, we examine only the quotes at the current best bid and ask prices. We consider indicators that describe the activity of a single product independently of complementary and supplementary securities. We compute the following indicators for each exchange:

- Weighted rate of change (*ROC*) of the relation of the bid depth (number of shares bid) to the offer depth (number of shares offered).
- n_{cb} (n_{co}) – the number of times an exchange locked the market on the bid (offer).
- n_{lb} (n_{lo}) – the number of times an exchange left the National Best Bid and Offer (NBBO) on the bid (offer).

Within each exchange, each of the indicators is assigned a weight. The weight of an exchange is determined as the average weight of its indicators. We then compute the composite indicators, s_b (s_o), which are the sum of weights of exchanges whose bid (offer) price is equal to the NBBO.

To remove outliers we use the three-sigma rule, assuming that the values of indicators are normally distributed. After the removal of outliers, the values of indicators are normalized by a transformation on the feature values to map them to the range $[0, 1]$: we find the maximum x_i^M and the minimum x_i^m values of an indicator i among all points in the cluster, and then normalize the feature as

$$x_i' = \frac{x_i - x_i^m}{x_i^M - x_i^m}.$$

Algorithm 1 ComputeWeights(X)

Input: X – a cluster of normalized points.
Output: w – a vector of weights of features.

$s \leftarrow \sum_{i=1}^{D} \sigma_i$
{where σ_i is the standard deviation of feature i in the cluster X}
for $i = 1$ to D **do**
  $w_i \leftarrow 1 - \sigma_i / s$
end for
return w

Algorithm 2 Predict(x)

Input: x, a normalized D-dimensional feature point to be classified.
d, a distance threshold. r, distance ratio threshold
Output: Either "predict price change up" or "predict price change down" or "not classified".

{Compute the squared distances to the centroids c^d and c^u of clusters down and up respectively, where w_i^d and w_i^u are the weights of the i-th indicator in the corresponding clusters.}
$d_d \leftarrow \sum_{i=1}^{D} w_i^d (x_i - c_i^d)^2$; $d_u \leftarrow \sum_{i=1}^{D} w_i^u (x_i - c_i^u)^2$

if $d_d < d$ **and** $d_d/d_u < r$ **then**
  return "predict a price change down"
end if
if $d_u < d$ **and** $d_u/d_d < r$ **then**
  return "predict a price change up"
end if
return "not classified"

For a point p to be an outlier, at least one of its coordinates p_o should satisfy $|p_o - s_o| > 3\sigma_o$. The case $p_o - s_o > 3\sigma_o$ implies

$$p_o > \frac{1}{K}\sum_{i=1}^{K} x_{oi} + 3\sqrt{\frac{1}{K}\sum_{i=1}^{K}(x_{oi} - \frac{1}{K}\sum_{j=1}^{K} x_{oj})^2}, \tag{1}$$

where K is the number of points in a cluster and x_{ij} is the i-th feature of the j-th point. Applying inequality (1) to normalized features, we have

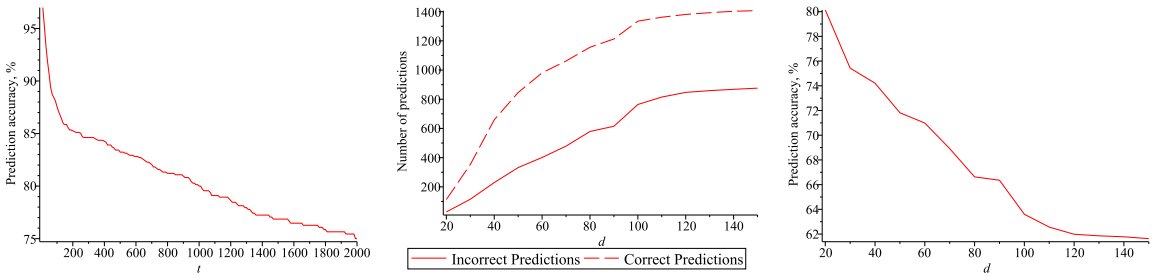
$$\frac{p_o - x_o^m}{x_o^d} > \frac{1}{K}\sum_{i=1}^{K} \frac{x_{oi} - x_o^m}{x_o^d} + 3\sigma_o', \tag{2}$$

where

$$\sigma_o' = \sqrt{\frac{1}{K}\sum_{i=1}^{K}\left(\frac{x_{oi} - x_o^m}{x_o^d} - \frac{1}{K}\sum_{j=1}^{K}\frac{x_{oj} - x_o^m}{x_o^d}\right)^2}.$$

An analogous reasoning can be applied to the inequality $p_o - s_o < -3\sigma_o$.

Computation of Weights Each indicator within a cluster is assigned a weight, computed as shown in Algorithm 1. The weight of an exchange is computed as the average of the weights of its indicators.



(a) (b) (c)

Figure 1: Prediction results:
(a) Prediction accuracy as a function of t (in milliseconds),
(b) The number of correct and incorrect predictions depending on the distance threshold d, and
(c) Prediction accuracy as a function of the distance threshold d

The Classifier To analyze incoming quotes at high frequency in real-time, the recognition model should be computationally efficient. For the classifier, we have chosen to compute the feature-weighted distance from a test point to the centroid of a cluster, since this is one of the least expensive techniques in artificial intelligence.

This technique is fast in training and classification. To *train* the model, one needs to collect a certain number of points in clusters and find the centroid of each cluster. We are interested in two classes of points: one class for price changes up and another class for price changes down. *Classification* of a quote is performed by computing the squared weighted Euclidean distance from the feature-vector of the quote to the centroid of each training cluster, as shown in Algorithm 2.

As can be observed in the algorithm, we differentiate the notions of classification and prediction. Classification happens with each quote received – a feature vector is formed and the distances to centroids are evaluated. In contrast, a *prediction* is made only if the distances between the sample and the centroids satisfy certain criteria, i.e. if the feature point is relatively close to one of the two centroids. The prediction accuracy can be increased and the number of predictions decreased by reducing the thresholds d and r in Algorithm 2. The necessary values of the thresholds can be determined empirically.

Experimental Setting The experiments were performed on data for MSFT (Microsoft) securities, using quotes from the following exchanges: NYSE Archipelago Exchange (Arca), Better Alternative Trading System (BATS) BZX Exchange, BATS BYX Exchange, Chicago Board Options Exchange (CBOE), Direct Edge A (EDGA), Direct Edge X (EDGX), NASDAQ, NASDAQ OMX BX, National Stock Exchange, and NASDAQ PHLX. The recorded events include: change in bid/offer prices and bid/offer depth. We recorded several days in December, 2011 with the total of 9,389,993 quotes and 4,658 price changes. Training was performed until both clusters had at least 10 points. The value of the weight in computation of the *ROC* was taken as 0.6. After 5 changes in price, parameters of a cluster were recomputed.

We calculated two measures: *on-change* distance accuracy and *prediction* accuracy. The on-change distance measure was counted as correct if the distance to the centroid of the cluster in the direction of the price change was smaller than the distance to the other cluster. In other words, if a change down (up) was recorded and $d_d < d_u$ ($d_d > d_u$), the on-change distance measure was counted as correct.

The prediction accuracy with a prediction interval of t was computed as follows. If the prediction was in the direction of the price change, and the interval between a prediction and the actual change was greater than t, the count of correct predictions was increased by one. If the interval was less than t, the

count was not changed. If the change happened in the opposite direction, the count of wrong predictions was increased by one, independently of the time interval after the prediction. This measure aimed to simulate real-life trading, when execution of a transaction takes a certain amount of time, depending on infrastructure.

Experimental Results The on-change accuracy of the model on the recorded data was 96.25%. The prediction accuracy as a function of t is presented in Figure 1(a) for the distance thresholds $d = 20$ and $r = 1/20$. There are 28 incorrect predictions for any t. We also measured the number of correct and incorrect predictions depending on the distance threshold d with fixed $r = 1/20$ for $t = 1000ms$. The results are presented in Figure 1(b). Figure 1(c) shows the prediction accuracy as the function of d.

3 A Market Simulator

We now present a simulator that can support market simulation research and is suitable for evaluation of algorithmic trading strategies.

Simulator Design The simulator currently consists of a matching engine, a communication interface and a variety of simulated trading agents. The matching engine accepts orders from both logged in users and computerized agents. It maintains a number of order books, each of which records the interest of buyers and sellers in a particular security and prioritizes their orders based on their price and arrival time. This centralized order system continuously attempts to match buy and sell orders. Matching rules are implemented based on continuous double auctions. The matching engine also publishes quote updates to all subscribers, which is handled by the communication interface.

To allow multiple users to interact with our simulator simultaneously and independently, we use the Financial Information eXchange (FIX) protocol [2], which is the *de facto* communications standard in global financial markets. The simulator provides each user a designated port for login and maintains a dedicated channel for communication. Both inbound and outbound messages, such as orders and execution reports, are encoded as FIX format. Using the FIX protocol also provides easy access to our simulator. Most users in both industrial and academic algorithmic trading settings are familiar with the FIX protocol or have it already implemented in order to connect to financial markets. This makes it possible to interact with our simulator with little modification to their systems.

In order to create various market conditions that are suitable for testing, we have developed five pre-defined types of simulated trading agents that represent an important subset of trading entities we observe in the real market. These agents are able to adapt to the market and interact with users' algorithmic trading strategies. The first of these is Market Maker Agent which plays neutrally against the market. Its primary objective is to enhance the liquidity and the depth of the market, resulting in a stable market. In contrast, Liquidity Taker Agent takes liquidity from the market by posting market orders that are often immediately executed at the best available price. By increasing the size or the frequency of the orders, it can potentially cause the quoted prices to change dramatically. Figure 2 shows a comparison of volatility between two simulations. The settings were the same except in the second simulation the Liquidity Taker Agent issues market orders at a higher frequency. Similar to the Market Maker Agent, Liquidity Provider Agent places limit orders to the market. By posting limit orders on both sides without immediately triggering a trade, it adds liquidity to the order book and consequently increases the depth and the stability of the market. We have also developed a Random Agent which uses no information about the market and issues random orders at certain time intervals. This type of agent can be used to create chaos in the simulation environment as well as to investigate the cause of certain market phenomena. The last type of agent is Swift Agent. Compared to the other four agents, a Swift Agent is more sophisticated in that it is able to control the number of open orders it places. This prevents the agent from exposing itself to too much risk, just as human traders would also do in a real market. In addition, the agent is able to monitor the price fluctuations of the simulated market. If the price variation exceeds a certain threshold, the agent will attempt to place more orders on the opposite side to counter the trend.

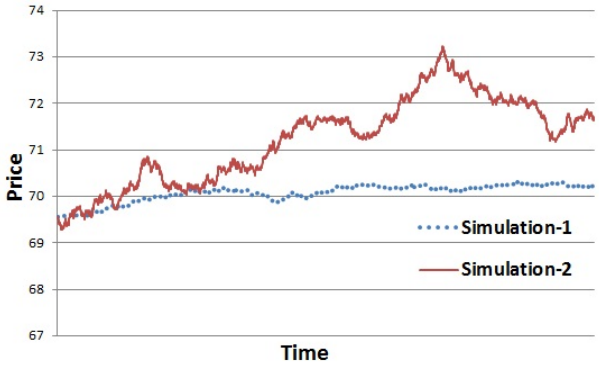


Figure 2: A comparison of two simulations with different liquidity taking.

Software Implementation The simulator is intended to be useful to evaluate trading strategies. It supports a variety of security types, including equities, futures, foreign exchange, and options. The simulator may run a market consisting exclusively of simulated (human or robotic) trading agents using different trading strategies (e.g., to study algorithms or market effects), or external participants (typically human) may log in and interact with the simulated market (e.g., for training). Participants (logged in users or simulated trading agents) can submit both limit and market orders with different time-in-force, allowing them to interact with the simulation environment as if they were trading in a real market. At the same time, the simulator adopts the FIX protocol, which allows multiple users to interact with the simulation environment simultaneously and independently. In contrast to other simulators that require testing trading strategies to be built on top of proprietary APIs, our simulator uses open protocols so can be integrated easily into their systems with little modification. The simulator is able to run in two different settings, each of which is useful in certain scenarios.

The first setting uses simulated trading agents, each of which is able to adapt and react to real-time market events by following selected pre-defined strategies. All of the agents are configurable. By adjusting their configurations, we can create very specific market conditions that would occur only rarely in a real market. In addition, the agent-based simulator is able to run at any time as the data are generated by the computerized agents. The agent-based simulator is useful in that it allows users to create desired market conditions where they can test their trading strategies whenever it is needed.

In the second setting, the simulator receives live market data from real exchanges and broadcasts this data to each user. Orders that are submitted by users are executed based on the current market conditions. This type of simulator is claimed by some to be more realistic. Meanwhile, the real market data simulator, in practice, has access to all the products available in the markets, while in agent mode this is not feasible unless there is a further configuration. We provide both these settings to users to evaluate their trading strategies, and give them the freedom to choose the one that is most suitable for their needs.

Useful Scenarios Both the agent-based and the live-data simulator have been adopted by Quantica Trading, a company located in Kitchener, Canada, which develops algorithmic trading software. We have found, informally, the agent-based and live-data simulators to be useful in a number of scenarios. First, some conditions, such as a market crash, may not occur very often in a real market, but they are extraordinarily costly when they do occur and so must be examined. With the agent-based simulator, we can easily reproduce these conditions, and variants, to test strategies. In addition, the simulator has also been found useful for education and training purposes. By executing trades in the risk-free simulation environment, it facilitates trading drills designed for new traders, allowing them to learn how to execute trades and manage risk faster. Moreover, the simulator is also suitable for software demonstration.

With round-the-clock access and risk-free testing, users can present demonstrations of their software applications at their convenience. Last, but not least, a simulator of the type we present also benefits users beyond the high-frequency trading world. High-quality simulation is essential to improve the regulatory environment for North American markets. At the moment, the true impact of regulation cannot be completely understood until it is in effect in the markets. This means that regulation can have unintended consequences or not achieve its desired results. High-quality simulation can help improve this, reducing risk of events such as the "Flash Crash" of 2010 [8].

4 Conclusion

We have explored both the analysis and simulation of financial markets. Our analysis has explored short-term price changes of securities using pattern recognition techniques. This method is based on the distance to the centroid of a set of feature vectors, representing an empirical combination of indicators. The model demonstrates good performance, even with naïve indicators. We have also presented a financial market simulator that supports a full range of security types and allows users to interact as if they were trading in a real market. It uses FIX as the communication protocol, allowing multiple users to interact with the simulation environment simultaneously and independently. We have also presented several types of simulated trading agents which represent a subset of traders observed in real markets. All of these agents are configurable and, by adjusting their parameters, very specific market conditions can be created to explore certain market behaviours. We have found that in a corporate setting that our simulator is useful in a number of scenarios, including system testing, education, training, and policy evaluation.

Acknowledgments

This work was supported, in part, by a CU-I2I grant from the Natural Sciences and Engineering Research Council of Canada. We thank James McInnes, Jonathan Leaver Travis Felker, and Peter Metford for discussions relating to desired function and implementation.

References

[1] Rob Curran and Geoffrey Rogow. Rise of the (Market) Machines. http://blogs.wsj.com/marketbeat/2009/06/19/rise-of-the-market-machines/. [retrieved: Oct 2014].

[2] FIX Trading Community. Financial Information eXchange (FIX) Protocol. http://www.fixtradingcommunity.org/. [retrieved: Oct 2014].

[3] Terry Hendershott, Charles M. Jones, and Albert J. Menkveld. Does Algorithmic Trading Improve Liquidity? *J. Finance*, 66(1):1–33, Feb 2011.

[4] Rui Hu and Stephen M. Watt. An Agent-Based Financial Market Simulator for Evaluation of Algorithmic Trading Strategies. *6th International Conference on Advances in System Simulation*, pages 221–227, Oct 2014.

[5] Rob Iati. The Real Story of Trading Software Espionage. http://www.wallstreetandtech.com/trading-technology/the-real-story-of-trading-software-espionage/a/d-id/1262125? [retrieved: Oct 2014].

[6] Bank of England. Patience and Finance. http://www.bis.org/review/r100909e.pdf. [retrieved: Oct 2014].

[7] Vadim Mazalov Travis Felker and Stephen M. Watt. Distance-Based High-Frequency Trading. *14th International Conference on Computational Science*, 29:2055–2064, July 2014.

[8] US Commodity Futures Trading Commission and US Securities & Exchange Commission. Findings Regarding the Market Events of May 6, 2010. http://www.sec.gov/news/studies/2010/marketevents-report.pdf. [retrieved: Oct 2014].