A MACHINE LEARNING APPROACH TO INTRA-MARKET PRICE IMPACT MODELING USING NASDAQ LEVEL-2 ITCH DATA

by

Jacob Bruce Brewer

Submitted to Brigham Young University in partial fulfillment

of graduation requirements for University Honors

Department of Computer Science

Brigham Young University

April 2016

Advisor: Dr. Sean Warnick

Department Honors Representative: Dr. Bryan Morse

ABSTRACT

A MACHINE LEARNING APPROACH TO INTRA-MARKET PRICE IMPACT MODELING USING NASDAQ LEVEL-2 ITCH DATA

Jacob Bruce Brewer

Department of Computer Science

Bachelor of Science

This research explores how to leverage system identification and machine learning to predict stock prices using NASDAQ order book data. It begins by providing essential background information of stock market trading mechanics and then gives a brief explanation of how machine learning is used for feedback system identification. The project then applies these principles to create a price impact model of NASDAQ stock prices. After describing detailed results, we show that prediction margins appear to increase for our testing set when we incorporate order book data. This project is a core element of a greater project, which explores the possibility of stock price manipulation and control—currently a great concern to organizations such as the Department of Homeland Security (DHS). Since our findings suggest that stock prices on our sampled

data set are at least slightly more predictable than a baseline algorithm when incorporating order data, it is likely that the growing number of similar high frequency trading algorithms would be affected by a significant change in the order distribution. This would mean that stock prices could be influenced without cost by strategically placing orders.

***Keywords*: control theory, price impact modeling, system identification, terrorism**

ACKNOWLEDGEMENTS

I am very grateful for the mentoring, support and education that made this possible. Thank you: Sean Warnick you for believing in me and for being truly invested in the success of you students. Thank you Galena Chenina, my precious wife, for inspiring and supporting me and even helped me debug my code on one occasion, as I prepared this thesis. I am also grateful for the opportunity to study at Brigham Young University.

This research was supported by the U.S. Department of Homeland Security and the Brigham Young Office of Research and Creative Activities.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

Currently the majority of US equity stock trades that take place are performed by high frequency trading algorithms, accounting for 60-73% of all stock exchange volume [20]. High frequency trades are financial stock trades that are operated entirely by sophisticated computer algorithms that execute purchases or sales of a stock within microseconds. High frequency trading algorithms typically base stock purchase and selling decisions on available financial data, such as the order book, and use machine learning to optimize for the best possible return on investment. However, in the wake of the 2008 financial crisis, regulatory institutions began paying more attention to this trading automation since in many instances high frequency trading has amplified the impact of local crashes, leading to market collapses. However, with current market monitoring and control mechanisms, such “high-frequency-trading induced crashes" are difficult to predict and take preventive measures against.

While investigating this domain, the Department of Justice has indicated the possibility of altering stock prices by placing limit orders (which do not immediately result in a stock trade) on the order book. This is also a concern to organizations like the

Department of Homeland Security (DHS) since, the ability of individuals or organizations to algorithmically drive market prices up or down, without incurring any cost of their own, opens the door to hazardous market manipulation, market volatility and even potential economic attacks on the United States and its citizens [1]. A part of this research project is to explore the potential opportunity of high-frequency traders or terrorists to control US market prices by artificially biasing the market order book.

The dynamics of the order book has been modeled by Biais et al. [6], Cont and De Larrard [18], Hall and Hautsch [4], and Cont et al. [19]. High-frequency trading order strategies were researched in Cvitanic and Kirilenko [13], Guilbaud and Ph [10], and Gatheral and Schied [14]. The possibility for attack on US equity markets using the much-studied phenomenon of information cascades has been researched by Alevy et al. [12], Bikhchandani and Sharma [21] and Devenow and Welch [3]. Recent work by Easley and Kleinberg [8] explores another potential weakness in equity markets related to network congestion and network market stability. This research builds upon the ideas mentioned here and uses control theory and machine learning to first determine order distribution influence on price and second to measure the potential surface of attack through placing dummy orders on the NASDAQ order book.

The next section begins with a review of basic order book dynamics (Chapter 2). Chapter 3 provides an overview of the core elements of system identification and also provides a comparison of system identification and machine learning. Chapter 4 walks through the data set and the econometric feature selection process for the chosen machine

learning approach to system identification. Chapter 5 describes the machine learning models and baseline algorithms. Chapter 6 describes the final results. Chapter 7 concludes by summarizing what we have learned and describes future work.

Chapter 2

System Description

2.1. Order Book

There are two primary types of stock orders in US equity trading markets – limit orders and market orders [15]. A limit order is a promise to buy or sell a specific number of stocks at a given price. This order sits on the order book until it is either cancelled or until a match is made to buy or sell the stocks by an electronic matching engine. A market order does not sit on the order book; it executes immediately at the best available price. These orders are processed in sequence of when they were received. In the case that a new buy or sell order is placed, and there are multiple limit orders waiting on the other side ready to execute at the set price, the computerized priority rules determine that the order, which has been on the books the longest executes first. Once this match is made, the order executes, stock possession transfers and the broker or dealer makes a small fixed trading commission. The status of the new lowest sell price and highest buy price is updated almost instantaneously for worldwide traders to monitor.

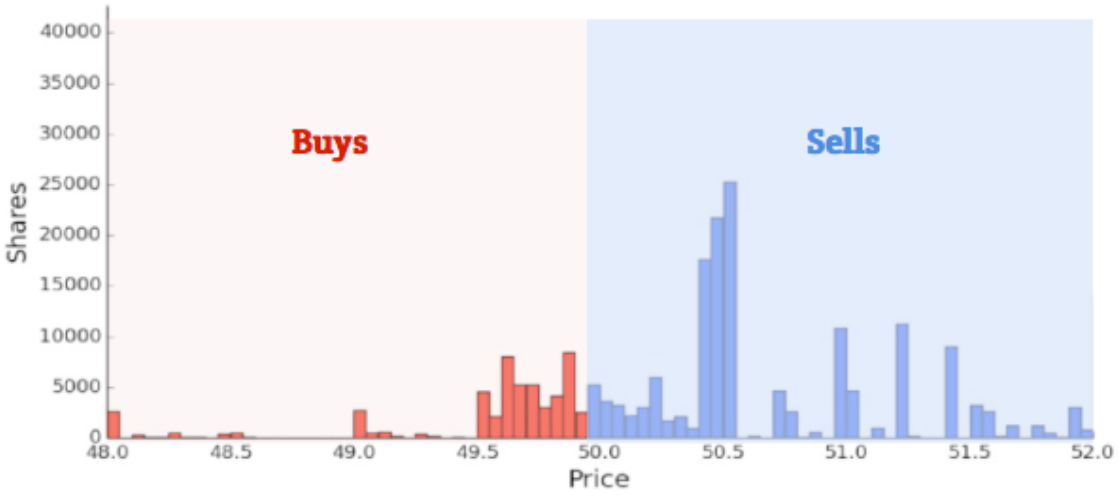

FIGURE 1. The order book for Citibank stock trading on NASDAQ at 10:00 a.m. on March 7, 2014. Red bars indicate the number of shares bid to purchase the stock at each price while blue bars represent the number of shares offered for sale at each price.

The pending limit orders can be visualized using a histogram. Figure 1 shows the order book for Citibank stock traded on NASDAQ at 10:00 a.m. on March 7, 2014 [7]. The red represents the limit orders to buy at set prices along the x-axis. The blue represents limit orders to sell. The y-axis reflects the total number of shares on the order book at the selected price. In this case, a market order to buy 5,000 shares would consume the 5,000 lowest priced shares for sale, represented by the left-most blue bars (at approximately $50.00). Likewise, any arriving market order to sell Citibank would sell to the highest priced order to buy on the order book (shown in red). A market order to purchase 10,000+ shares would be matched to multiple blue bars to meet the large

purchase order. This order execution may even leave a gap of empty space between the blue and the red bars. This gap is usually narrow among highly traded stocks, and wider among less popular stocks and is called the bid-ask spread.

2.2. National Security Concerns

Buying or selling sizable volumes of shares will clearly affect the spread and stock prices. However, in recent years, organizations like the Department of Homeland Security, have become increasingly concerned that prices could be artificially manipulated by placing orders outside of the spread region, that ultimately would not even execute [2]. This means that a trader or an attacker could shift market prices without incurring any cost.

The introduction of algorithmic trading has increased the possibilities for such market manipulation and attacks. In efforts to gain marginal investment advantages, high frequency traders are incorporating as much financial data as possible, including the order book, to guide automated algorithmic trading decisions. For example, when a trading algorithm spots a rapidly increasing number of sell limit orders, this may indicate to the trading algorithm that traders do not want to hold this stock anymore and the price will soon drop. If thousands of automated trading algorithms across the globe simultaneously act accordingly, then this could trigger what is called “a flash crash” even though the responsible limit orders did not even execute. This form of attack, known in

one of its forms as “spoofing,” is potentially responsible for the flash crash of May 2010 [5].

Chapter 3

System Identification and Machine Learning

3.1. System Identification

In order to determine how big the limit-order spoofing threat is, we must first model the behavior of the order book system using system identification. System Identification is the process of using statistics to create models that based on measured data, reveal the dynamics of the system. There are three general steps in system identification as mentioned in Ljung's guide on System Identification [17]:

- Obtaining Useful Data
- Generating a Set of Candidate Models
- Determining Which Candidate Models is Best

Step 1: The Data – Ideally, the data used to identify the system is based on carefully designed experiments that show the input-output behavior of the system under as many circumstances as possible. The topic of experiment design is the study how to maximize the "informativeness" of this data. In our study of the order book dynamics, our research team does not have the funds to directly experiment by placing billion-dollar limit orders on the NASDAQ order book to see how the system behaves, but enough

order book history, with its market fluctuations, is available that our data set is reasonably informative.

Step 2: Candidate Models – A series of candidate models are generated using a variety of possible scientific methods. This step of the process is iterative and will often take the most time and effort.

Step 3: Assessment – There is a process for determining which of the models is the best model and should be further explored. This is typically done through experimentation and simulation. After this step is completed, the process may return to generating more candidate models (Step 2), or terminate if a stopping criteria is met.

3.2. Machine Learning Used in System Identification

Machine learning is similar to system identification, although there are some subtle differences. Both machine learning and system identification involve what is called "learning a problem." Both of them use similar steps: 1) Data 2) Candidate models and 3) Selection Criteria, as discussed earlier. Both machine learning and system identification can be used to predict the future and both are black box models. However, machine learning addresses this problem by refining a model class composed of functions and in the case of system identification the model class is not composed of just functions, but rather dynamic systems. Another difference is that in system identification there is a state variable; in machine learning there is not. Since a descriptive feedback system has a state

variable, it determines not only what a future output will be, but also the future state. Machine learning on the other hand, uses the current state to predict a selected output [9]. Another difference is that system identification is strictly linear, while machine learning can be non-linear.

While machine learning is not identical to system identification, they both serve a similar purpose. In fact, machine learning can often prove to be more predictive of future state and behavior than system identification, even though it does not use explicit state variables [16]. For the purpose of this research we will use a machine learning approach to solving the system identification problem. We do this because stock price changes do not fluctuate according to a known linear model. However, since our purpose is to identify a system that we can control – not merely predict – we will continue to use the term system identification. However, our method for obtaining the desired system will be through machine learning.

Chapter 4

Data & Econometric Feature Selection

4.1. Data Source

The data set that we work with to determine equity market vulnerability is approximately 16 terabytes of NASDAQ Level-2 ITCH data spanning more than a year starting in early 2014. This data contains all stock orders placed on all tickers on NASDAQ at every microsecond. The data records the type of order (buy, sell, cancel, etc.), the quantity of shares, and the time stamp of the order. Using this data we reconstruct the order book for any given stock ticker at any given time. Table 1 shows what the data set looks like.

TABLE 1. NASDAQ Order Data 2014

Type	Seconds	Order Number	Side	Shares	Price ($)	Shares Left
A	14510.5014246	99691	B	1	1000	1
F	34214.5461243	10398669	S	18	554.35	18
D	34222.2947349	31938698	S	263	554.9	0
D	21399.422182	343764	B	100	551.85	0

A	31201.120316	17189918	B	35	553.54	35
D	34201.1848943	31249690	S	152	555.44	0
…	…	…	…	…	…	…

For the purposes of this project we use a subset of this data for 10 tech companies of interest including Google, Amazon, Microsoft, Apple, Facebook, LinkedIn, Twitter, Intel, HP, and Baidu. We then select 10 random days to observe these stock tickers. Since there are 10 stocks on 10 days we have a total of 100 trading days to work with, which comes out to approximately 1.6GB. Each day of trading has on average 500,000 orders per day per stock, which amounts to 5 million orders per day for the ten companies we chose to work with. Several of these companies are considered to be competitors to one another (for example HP and Intel) and were therefore useful for competitor stock price analysis and predicting. If we then aggregate the total order placements and account for order execution, then we can use the discrete orders seen in Table 1 to create the histogram shown in Figure 1.

Not only can we create the histogram pictured in Figure 1, but also we can make similar histograms for machine learning purposes. These histograms will also have price as the x-axis, but instead they will have unique orders, total age and other measures as the y-axes. We can also create even more histograms for intra-day market order executions for the given day, since market orders execute immediately, and would otherwise be lost data on a limit order distribution histogram. We then represent each of these respective

histograms as a physical object or as a statistical distribution. These distribution histograms will be used for the feature selection process that will be outlined in the coming section.

4.2. Feature Selection

Representing each of these histograms as a material object, we calculate distribution features for machine learning training purposes. Some of the distribution features include:

- Quintiles
- Median
- Center Of Mass
- Maximum Value
- Mode

We also calculate distribution features such as "percentage-of-buy-orders-within-10%-of-highest-buy-price," since economic theory of price and demand suggests that order mass is repulsive. This means that an exploding number of sell limit orders will likely result in a price drop and vice-versa. We also calculate a number of non-distribution features including the bid-ask spared length along with the hour of day, in case there is a correlation between times of day and certain price movements.

The distribution metrics are calculated individually for each order histogram type (buy, sell or other) and then collectively. They are also computed for the list of

competitor stocks, because economic substitution laws suggest that Microsoft prices might increase when Apple is on a decline [11]. Competitor prices could also be relevant in the case that the market (or specific industry) is broadly moving up or down. For example, if the precious metals industry as a whole is declining, then it is likely that Royal Gold Inc. (RGLD) will follow the same trends and decline as well. That means that the total feature space comes out to:

$$F = t * f * w + n$$

Where t is the number of orders types, f is the number of distribution features, w is the number of weights, and n is the number of non-distribution features. We can then augment this feature space further by incrementing backwards in time for the given order-book time instance, and record T order book snapshots. This allows us to calculate discrete time derivatives for each distribution feature leading up to the order-book time instance. Thus our final feature space comes out to 6,000+ features as given by:

$$F = t * f * w * c * T + n$$

To further improve the feature set, the features should be normalized and standardized. Normalization is commonly understood by those familiar with machine learning, however, standardization is used less frequently. To put things simply, standardization brings all the features on the same range of possible values. This is useful since it is completely arbitrary that Microsoft's stock prices are half the price of Apple stock prices, even though Microsoft is a bigger and more profitable company. Stock prices, and therefore many of the distribution features, depend on how many shares a

company decides to issue. Microsoft has simply decided to issue more shares of its stocks and thus its price is also lower.

To account for extreme outliers on the buy or sell side, the histograms can be truncated by moving the outermost 2% of orders inward, thus eliminating outlier effects. Here is an example of some of the features that our final feature space included:

- sellMeanWeightedWithShares
- buyMedianWeightedWithCounts
- cancelsKurtosisWeightedWithShares
- buyAgesQuintile50
- buyAgesTruncatedQuintile98
- buyMedianWeightedWithCounts
- allAgesTruncatedQuintile2
- …

Lastly, we label each of these training instances, each of which contains all the mentioned features. To make things simple, rather than predicting stock prices on a continuous scale we discretized the output by classifying each instance as either having gone up, down or stayed roughly the same t seconds in the future. For the test in this project we used $t = 5.00$. That way performance is easier to measure.

Chapter 5

Algorithms Used

5.1. Baseline Algorithm – The Naiive Trader

We benchmark performance against two naïve algorithms. The first makes random guesses of future output and the other algorithm exclusively predicts the majority output of training data. The Majority Vote algorithm is our primary performance benchmark. If we are in fact predicting or manipulating stock prices in a way that gives a market advantage, then we want to measure our performance against the common stock trader. A casual stock trader, who does not leverage algorithmic trading or in depth market research, will purchase a stock and assume that overall the market value will increase over time. Therefore, he or she buys and holds. The trader will not sell or short sell because, he or she knows that there will be ups and downs, but overall the assumption is that he or she will ride out the 'downs' of the market and hopefully end up with an eventual gain. A stock trading control mechanism that exclusively predicts 'up' and purchases accordingly will mimic the behavior of the casual stock trader that buys and holds. This baseline algorithm works best as a benchmark for binary output (up or down only), and is diluted in accuracy when we tested with three outputs (up, down, or remain roughly the same).

Further work would evaluate the significance in prediction variance using statistical p-tests and hypothesis testing. These results could be plotted on a ROC-curve, which is a standard visualization of algorithm accuracy in the field of machine learning. However, for the time being, the Majority Vote algorithm will serve as our primary performance benchmark.

5.2. Selected Machine Learning Algorithms

Here are some of the machine learning algorithms we used with a brief description of the algorithm:

Neural Network – A powerful machine-learning algorithm able to detect hidden features. We hypothesized that it would do best at predicting stock prices for this reason.

Simple Perceptron – Essentially a single layered neural network for comparison purposes.

Grandient Boosting – An algorithm that produces a prediction based on a set of weaker prediction models.

Random Forest – Similar to Gradient Boosting, but used exclusively with randomly generated decision trees.

Naiive Bayes – Leverages probability theory to make classification predictions.

K Nearest Neighbors – A form of lazy learning that finds K similar instances and takes the majority vote of these "neighbors".

Chapter 6

Results

6.1. Initial Attempts Predicting Google

We first tested on one single day of Google on April 4th, 2015 with no competitor stocks or time shift derivatives. The results can be seen in Table 2.

TABLE 2. Initial Results – Predicting Google Stock (GOOG) with 1,000 instances, 226 features. Actual output: 34% 'up', 66% 'down'.

Algorithm	Accuracy	Confidence
Gradient Boosting	71.5%	71.0 - 71.0%
Random Forest	70.1%	69.0 - 70.0%
Naive Bayes	67.8%	67.0 - 67.0%
K-Nearest Neighbors	67.5%	67.0 - 67.0%
Naive Prediction	66.0%	65.0 - 66.0%

These initial tests included 256 distribution features that were not normalized. The majority vote algorithm predicted with 66% accuracy while all four of the four smart

algorithms predicted slightly better, and gradient boosting predicted 5.5 percentage points better than the majority vote algorithm, which is a ~8% accuracy boost. These results looked promising, however as we repeated these test we found that we must have gotten lucky. The pattern did not hold, and often the majority vote algorithm performed best. We continued to attempt predicting Google stock data with data normalization, standardization, truncation, discrete time derivatives and accounting for companies classified as competitors by Morningstar Investment Research, such as Microsoft. However, even with these improvements the accuracy did not appear significantly better than a common stock trader, as represented by the baseline algorithm. In fact, whenever the majority vote was in the 70%+ range the baseline algorithm almost always won. See Table 3 for example:

TABLE 3. Predicting Google Stock (GOOG) with 12892 instances, 1224 features. Actual output: 73.8% 'up', 26.2% 'down'.

Algorithm	Accuracy	Confidence
Majority Vote	73.8%	73.0 - 74.0%
NN (TensorFlow)	73.8%	73.0 - 74.0%
K-Nearest Neighbors	69.4%	68.0 - 70.0%
Random Forest	69.4%	68.0 - 70.0%
Gradient Boosting	67.8%	67.0 - 68.0%

Naiive Bayes	58.4%	57.0 - 58.0%
Random Guess	49.9%	49.0 - 50.0%

6.2. Facebook Prediction

In attempts to improve accuracy, we added a few more machine learning models to the mix and shifted attention to a new stock, Facebook, and immediately found that the prediction margins were much better, as demonstrated in Table 4.

TABLE 4. Initial Results - Predicting Facebook Stock (FB) with 12892 instances, 1224 features. Actual output: 37% 'up', 38% 'down', 25% 'same'.

Algorithm	**Accuracy**	**Confidence**
Random Forest	65.6%	65.6 - 66.0%
K-Nearest Neighbors	51.4%	50.0 - 52.0%
Neural Network	46.9%	46.0 - 47.0%
Perceptron	46.7%	46.0 - 47.0%
Naiive Bayes	45.5%	44.0 - 46.0%
Majority Vote	37.9%	37.0 - 38.0%
Random Guess	34.4%	33.0 - 35.0%

With these final improvements that we made, we saw Random Forest storm ahead the rest of the pack in terms of accuracy with 65% accuracy while distinguishing between ‘up’ ‘down’ and the newly added ‘same’ output classes. As stated earlier we were expecting the neural network to perform the best, but it was 20 percentage points below the accuracy of Random Forrest.

We then tested Facebook again, but added a competing social media company, LinkedIn, to the feature space. The results are seen in Table 5.

TABLE 5. Predicting Facebook Stock (FB), referencing LinkedIn (LNKD) in feature space. Uses 12892 instances, 2448 features. Actual output: 20% 'up', 58% 'down', 22% 'same'.

Algorithm	Accuracy	Confidence
Random Forest	67.0 %	65.0 - 69.0 %
K-Nearest Neighbors	59.2 %	57.0 - 61.0 %
Neural Network	58.7 %	56.0 - 61.0 %
Perceptron	58.4 %	56.0 - 61.0 %
Naiive Bayes	31.2 %	28.0 - 33.0 %
Majority Vote	58.8 %	56.0 - 61.0 %
Random Guess	42.6 %	40.0 - 45.0 %

Adding LinkedIn features appeared to boost accuracy even more for Random Forest, which continued to perform best, and all machine-learning models performed much better than the baseline algorithm. Future work will assess the significance of these marginal differences using p-tests and other statistical methods. While we were surprised that the neural network did not perform best, a plot of the neural network training and test set accuracy plotted over epochs revealed something very promising regarding stock price prediction. This plot can be seen in Figure 2.

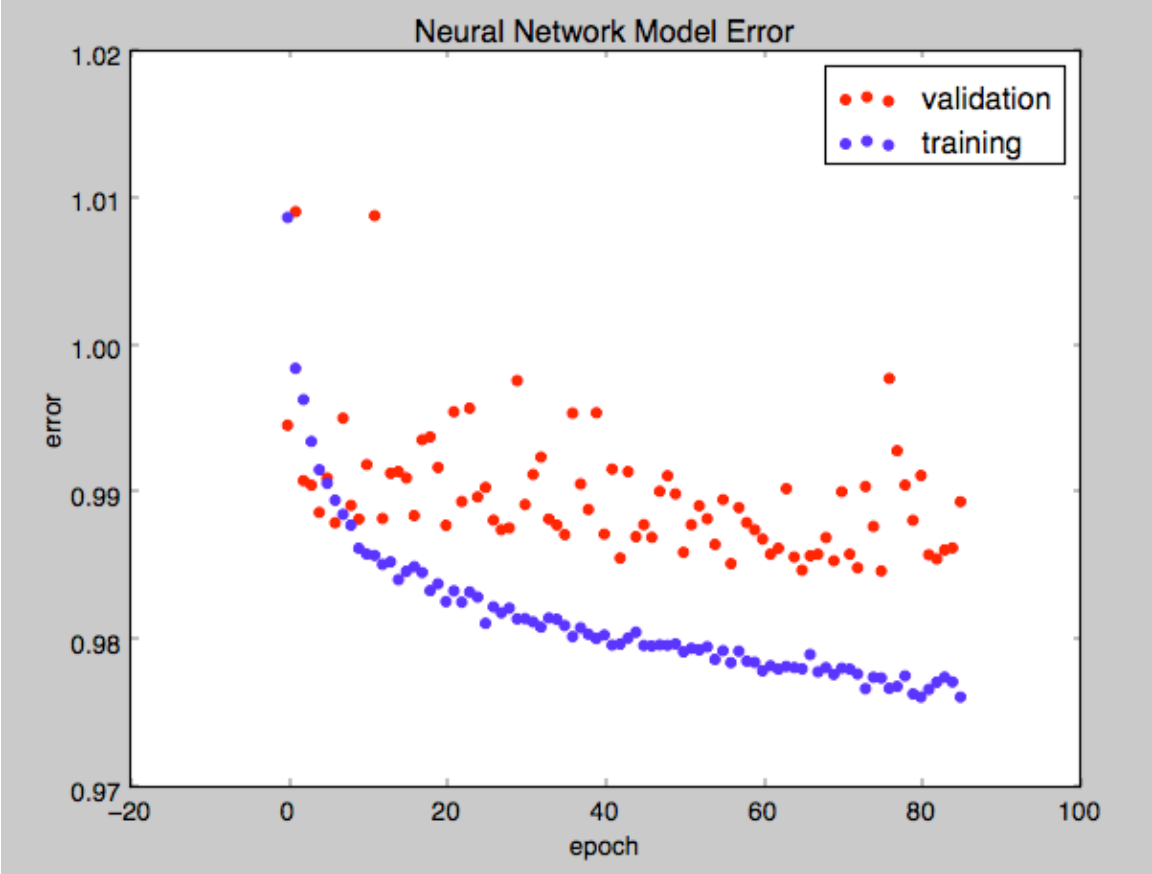

FIGURE 2. Training and Test Set Accuracy for the Neural Network over each Epoch.

From the image above, there is a visible upward blip near the 80th epoch of the validation set and the training set continues to decrease in error. This indicates that some over-fit is taking place, and while the accuracy of the neural network was 10 percentage points below Random Forrest, there is a slight but distinct downward slope in the validation set accuracy. This indicates that the model is in fact learning from the training data and becoming better at predicting the stock price movement of novel data. This is strong evidence, that at least for Facebook stock prices are predictable with a wide error margin. To further determine whether or not this applies to other or all stocks at other times, we will need to continue to test a broader set of stocks across a wider time span. We can however conclude from these results that at least for some stocks order book distributions are very broadly predictive of future stock price movement.

Chapter 7

Conclusions

Predicting future stock prices for companies with low trade volume relative to companies like Microsoft and Google, was much more successful. In our tests we were predicting the stock price at a fixed five seconds into the future. We hypothesize that the reason that highly traded stocks were more difficult is that at such a high trading speed, future price predictability decays proportionally faster. After only two seconds of active trading hours on Google, so many market orders have executed that the most important features (likely the features surrounding the spread region) have already mutated to become indistinguishable from that of two seconds prior.

Based on these observed predictability dynamics, we hypothesize that for each stock there is a window of prediction opportunity that decays. Perhaps the rate of decay is quicker for stocks that are highly traded. This may explain why we were able to predict Facebook well, because with a five second prediction into the future we happened to be

striking within this hypothetical window of prediction opportunity. It could be that the prediction window for Google is two seconds or one second since it is as if higher trading volume makes time go faster. This hypothesis however, would need to be further tested to be verified.

7.1. Further Work

Price Impact Model Improvements

Future improvements will involve feature space improvement, machine learning model improvement and further testing. To improve the feature space, we could add more distribution features pertaining to the histogram such as skewness and other metrics. The feature space could also be scaled by running a PCA algorithm to reduce the feature space. There are also many other machine learning algorithms that could be tested and it would also be worth investigating why the neural networks did not perform well in this study. Optimal parameters for these machine-learning algorithms could be identified using a parameter grid search or other type of parameter exploration wrapper.

Also, to truly determine consistent stock price predictability, we would need to test these prediction algorithms on a much larger data set. This may involve computing a GPU on a super computer and accessing many more of the stock tickers across a much larger time range.

Next Steps to Determine Stock Price Manipulation Threat

The primary object of this class project was to do the first part of this research, the system identification. Later the system identified in this research will be used to design and test a controller that simulates trades to greedily optimize trading return on investment (ROI) based on future stock price predictions. The controller will be tested on the BYU IDeA Labs' Tour De Finance equity trading simulation system, an open source project that dynamically measures high frequency trading algorithm performance for research purposes. If the trading controller simulation results in a statistically significant positive delta in equity trading ROI, then we have shown that order book behavior is predictive of future price movement. In the case that the controller obtains a statistically significant positive delta in trading ROI, then this suggests that the order book shape is in fact predictive of future stock prices. If this is the case, then those features could be manipulated by placing unexecuted limit orders to control stock prices. In the case that trading ROI is not statistically significant, then this suggests that the features that we have learned are not significantly predictive of future prices. This may mean that stock prices cannot easily be controlled by manipulating the order book, and that the threat to this type of economic terrorist attacks is low.

Bibliography

[1] "DHS Open Source Enterprise Daily Cyber Report." Internet: $http://www.columbia.edu/~joel/DHS/DHS_Cyber_Report_2011-01-11.pdf, [Nov 2, 2015].

[2] "DHS Open Source Infrastructure Report." Internet: http://www.dhs.gov/sites/default/files/publications/nppd/ip/daily-report/dhs-daily-report-2014-10-20.pdf, [Nov 2, 2015].

[3] A. Devenow and I. Welch, "Rational herding in financial economics," *European Economic Review*, vol. 40, no. 3, pp. 603–615, 1996.

[4] A. Hall and N. Hautsch, "Order aggressiveness and order book dynamics," *Empirical Economics*, vol. 30, no. 4, 973–1005, 2006.

[5] A. Kirilenko, A. Kyle, M. Samadi, and T. Tuzun, "The flash crash: The impact of high frequency trading on an electronic market," *Available at SSRN 1686004*, 2014.

[6] B. Biais, P. Hillion, and C. Spatt, “An empirical analysis of the limit order book and the order flow in the paris bourse,” *The Journal of Finance*, vol. 50, no. 5, pp. 1655–1689, 1995.

[7] BYU Information and Decision Algorithms Laboratories (IDeA Labs).

[8] D. Easley and J. Kleinberg, *Networks, crowds, and markets: Reasoning about a highly connected world.* Cambridge University Press, 2010.

[9] D. Fogel, *System Identification Through Simulated Evolution: A Machine Learning Approach to Modeling*. Lexington, MA: Ginn Press, 1991.

[10] F. Guilbaud and H. Pham, “Optimal high-frequency trading with limit and market orders,” *Quantitative Finance*, vol. 13, no. 1, 79–94, 2013.

[11] I. Dierickx and K. Cool, "Asset stock accumulation and sustainability of competitive advantage," *Management Science*, vol. 35, no. 12, pp. 1504-1511, 1989.

[12] J. Alevy, M. Haigh, and J. List, “Information cascades: Evidence from a field experiment with financial market professionals,” *The Journal of Finance*, vol. 62, no. 1, pp. 151–180, 2007.

[13] J. Cvitanic and A. Kirilenko, “High frequency traders and asset prices,” *Available at SSRN1569075*, 2010.

[14] J. Gatheral and A. Schied, “Dynamical models of market impact and algorithms for order execution,” *HANDBOOK ON SYSTEMIC RISK, Jean-Pierre Fouque, Joseph A. Langsam, eds*, pp. 579–599, 2013.

[15] L. Harris, Lawrence and J. Hasbrouck, "Market vs. limit orders: the SuperDOT evidence on order submission strategy," *Journal of Financial and Quantitative Analysis*, vol. 31, no. 02, pp. 213-231, 1996.

[16] L. Ljung, H. Hjalmarsson, and H. Ohlsson, "Four encounters with system identification," *European Journal of Control*, vol. 17, no. 5, pp. 449-471, 2011.

[17] L. Ljung, *System Identification*. Boston, MA: Birkhäuser Boston, 1998.

[18] R. Cont and A. De Larrard, "Price dynamics in a Markovian limit order market," *SIAM Journal on Financial Mathematics*, vol. 4, no. 1, pp. 1–25, 2013.

[19] R. Cont, S. Stoikov, and R. Talreja, "A stochastic model for order book dynamics," *Operations research*, vol. 58, no. 3, pp. 549–563, 2010.

[20] R. Lati. "The Real Story of Trading Software Espionage." Internet: http://www.advancedtrading.com/algorithms/thereal-story-of-trading-software-espio/21840150, Oct. 7, 2009 [Nov 2, 2009].

[21] S. Bikhchandani and S. Sharma, "Herd behavior in financial markets," IMF Staff papers, pp. 279–310, 2000.

[22] Stephen Marsland *Machine Learning: An Algorithmic Perspective*. Taylor & Francis Group, Florida, 2015.