A Survey of High-Frequency Trading Strategies

Brandon Beckhardt[1], David Frankl[2], Charles Lu[3], and Michael Wang[4]

[1]beb619@stanford.edu
[2]dfrankl@stanford.edu
[3]charleslu@stanford.edu
[4]mkwang@stanford.edu

June 6, 2016

Abstract

We survey and implement a number of known high frequency trading strategies. These strategies take advantages of Thesys high frequency data and attempt to trade intraday from frequencies ranging from milliseconds to minutes, some utilizing machine learning techniques. While discussing and implementing these strategies was insightful, we fail to make significant profit after accounting for transaction costs. That is not to say these strategies do not work in general. We also hope to elucidate where our strategies fall short and where to improve them.

[0]BB contributed the correlation model strategy. DF contributed the ETF statistical arbitrage strategy. CL contributed the order book pressure strategy. MW contributed the automated technical strategy search and intelligent market making algorithm. We thank Lisa Borland and Enzo Busseti for all the insightful discussions and trading ideas. We would also like to thank Artem Vasilyev for contributing his deep insight on "recurrent neural networks" and "locality sensitive hashing" for instantaneous exchange latency arbitrage. Finally, Thesys provided the equity data.

1 Introduction

In this paper, we will present five different high frequency trading strategies that we researched and implemented using Thesys data and platform. The strategies are diverse in nature and attempt to capitalize on independent sources of alpha. So, our original intent was to combine such independent predictors into one ensemble trading strategy. However, in our case, a full ensemble proves to be needlessly complex. Alternatively, we can propose to combine our strategies (once they are tuned and profitable) with Markowitz portfolio optimization.

As a result, we decided to individual research and implement five strategies. The first is a method to search for and identify profitable technical analysis strategies that do not overfit. Next, we attempt to statistically arbitrage highly correlated index tracking ETFs. In a similar vein, another strategy models correlation between assets and attempts to profit off asset divergence. Fourth, a strategy to predict price movements from order book pressure dynamics tuned with machine learning techniques manages to turn a profit. Finally, we have an intelligent market making strategy that places its quotes based on a machine learning algorithm trained on order book dynamics.

2 Data and Methods

Since Thesys only provides equity data, our universe of assets are limited to stocks. We choose equities listed on SP500, NASDAQ, and NYSE because they are liquid to facilitate high frequency trading. Code and simulations are run within Thesys's platform, which is hosted on a Jupyter notebook. We utilized Python packages such as Pandas, NumPy, and scikit-learn for our quantitative analysis.

3 Strategy 1: Automated Technical Strategy Search

3.1 Background

This "strategy" is less of an algorithmic *per se* but rather a technique for finding profitable technical analysis strategies, which can then be turned into algorithms. The goal is to define a strategy space - a set of a certain

enumerable strategy type - and search over that space, selecting only the most profitable for trading.

With a naive approach, the primary issue is overfitting. Since we only have historical data with which to search (and select our strategies from), it is possible and indeed highly probable that we select spurious technical analysis signals that look like they have predictive power but perform quite poorly out of sample. In real world trading, this issue becomes a strong concern for technical traders, who psychologically believe they identify profitable patterns on historical data, but rack up huge losses when using the same patterns on forward trading. To mitigate this concern, we seek to find a way to systematically address the issue of overfitting.

This technique was inspired by that of a hedge fund manager Jaffray Woodriff of Quantitative Investment Management.[7] While not much is extensively known about his specific technique, it is know that Woodriff employs random time series in searching for technical analysis strategies and that he combines his technical predictors into an ensemble.[6] Additional help was obtained through a blog post attempting to replicate Woodriff's methods. We extend their work on lower frequency daily data to high frequency tick and order book data.

3.2 Benchmarking on Random Data

To this end, we come up with the approach to benchmark the strategies in our strategies against their performance on random time series data. The goal we have in mind is to select the highest returning strategies in our strategy space that aren't overfit or due to random noise. On random data, we try the strategies on our strategy space and achieve some baseline returns. Since the underlying time series are random, we know even the best performing of these strategies have zero alpha.

So when we run the same strategies on real time series data, the ones here that outperform those on random data will have positive alpha. From this, we conclude that these top performers on real data that beat those on fake data turn significant profit as to be not overfit or due to random noise.

3.3 Generating Realistic Random Time Series

The insight here was to employ the bootstrap, a statistical technique for sampling with replacement.

First, we obtain a day's worth of tick data (bid and ask quotes) for a single stock of interest in our universe. This leads to around one million different tick events which include all changes at the inside order book. For

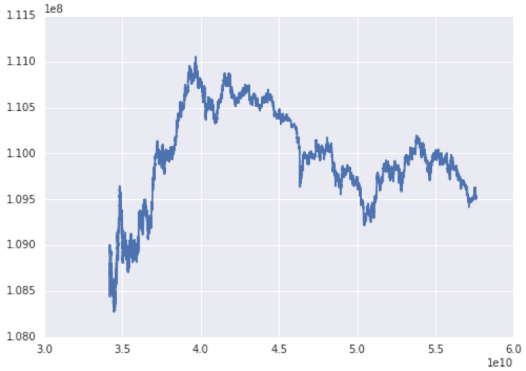
computation simplicity, we then compute each tick's price change as an absolute difference (assumed due to the fine granularity of the returns) in mid market price.

Next, we use the full day's mid market price changes and the time between each tick to produce two empirical probability distribution functions (PDFs). So, we come up with an algorithm to generate and simulated random walk.

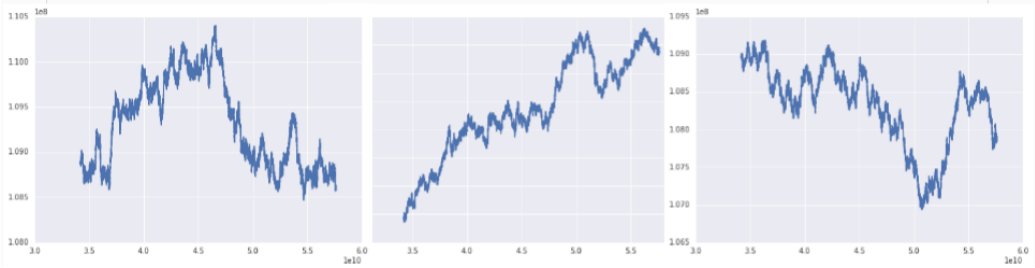
First, initialize $t = 0$ and first price p_0 randomly within some bounds and have a list of trade times as vector T. Furthermore have a list of prices P. While $t \leq$ end of day, draw a time change t_δ from the PDF and a price change p_δ. Store these into the vectors: $T+ = t_{\text{prev}} + t_\delta$, and $P+ = p_{\text{prev}} + p_\delta$. Here $+ =$ denotes the append to vector operation.

When this finishes, we will have a random walk for a day's worth of ticks and corresponding prices modeling on the empirically observed price changes and trade times during the day. In essence, we have shaken up the original day's events on two axes: the price and time. This procedure allows us to rapidly generate realistic looking random price time series.

In apply this procedure, there's the tacit assumption that ticks are IID, which may not be a perfect assumption. One can imagine that the information gleamed from one tick and order would affect subsequent trades also, thereby violating our assumption. More concretely, this assumption makes the produced random time series (which still realistic looking) have less volatility clustering as expected. However, we believe this to not affect the procedure as a whole. Because we are not doing any learning or fitting on the random data itself, we believe this assumption to be a minor flaw.

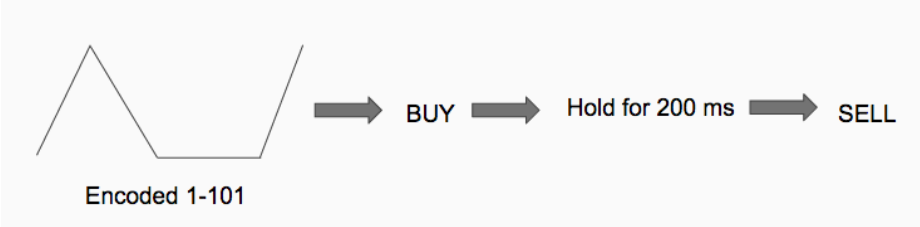


This real series of real AAPL on 2015-01-21 becomes transformed into



3.4 Results on Strategy Space of 4 Tick Price Action

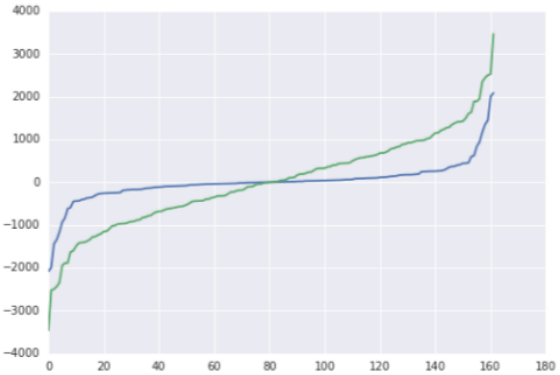
We will now demonstrate the technique on a sample strategy space of all price action strategies of 4 ticks, looking at the sign of the price change within 4 events. We arbitrary impose a holding time of 200ms (but this could be a parameter to tune). The strategy looks like



We simply brute force over all possible strategies. There are $2 * 3^4$ of these, since there are three outcomes at each tick $-1, 0, 1$ (down, neutral, and up) and two resulting actions: buy and short.

The most profitable strategy on fake data performed at a profit of 2083 microdollars per trade whereas the most profitable strategy (a downward momentum strategy!) profited 3461 microdollars per trade. So, our technique posits that this downward momentum technical analysis strategy is predictive out of sample.

If all the strategy are plotted on their profitability as so:



we see the green line (strategies on real data) outperform those on fake

data (blue line) so the difference is presumed alpha.

3.5 Discussion

This technique looks promising though not necessarily on high frequency data. First, technical analysis has an ambiguous underlying mechanism in the high frequency space; it is premised on behavior inefficiencies leading to price patterns, but when trades are dictated by algorithms and computers, do those same patterns have meaning? Additionally, trading on high frequency data requires aggressive trading, often crossing the spread to input market orders, wiping out the 3046 microdollar/trade profit we attained in the sample strategy.

There are also problems with computation intractability associated with are search strategy. Right now, we are simply brute forcing and trying every possible permutation. Obviously, this will require exponentially more computing power as we increase the complexity of our strategy space. One possible iteration include a more intelligent search mechanism such as genetic algorithms.

4 Strategy 2: Exploiting Correlation

4.1 Background

Finding assets that are correlated in some way, building a mathematical model for this correlation, then trading based off of that correlation is common practice in many different trading strategies. We implemented a trading strategy that finds the correlation between two (or more) assets and trades if there is a strong deviation from this correlation, in a high frequency setting. The inspiration for this strategy came from the article *Online Algorithms in High-frequency Trading The challenges faced by competing HFT algorithms*, written by Jacob Loveless, Sasha Stoikov, and Rolf Waeber.

4.2 Algorithm

To create a model representing the correlation between assets, we implemented an exponentially weighted linear regression. This linear regression is meant to model the linear relationship between the asset we are looking to trade Y and the component assets X we are using to find this relationship. The resulting linear combination will be of the form:

$$Y = \beta X + \epsilon \tag{1}$$

Y is a vector of points representing the assets price, X is a $m \times n + 1$ matrix where m is the number of price points we are evaluating and n is the number of assets we're using to estimate Y. The first column of X is the intercept term. ϵ is a vector representing the difference in prices needed to exactly model Y. Since our assumptions for trading are based on the difference between our estimate and the actual asset price, we are acutally not using ϵ, so our resulting algorithm is

$$\text{Estimated\_Y} = \beta X \tag{2}$$

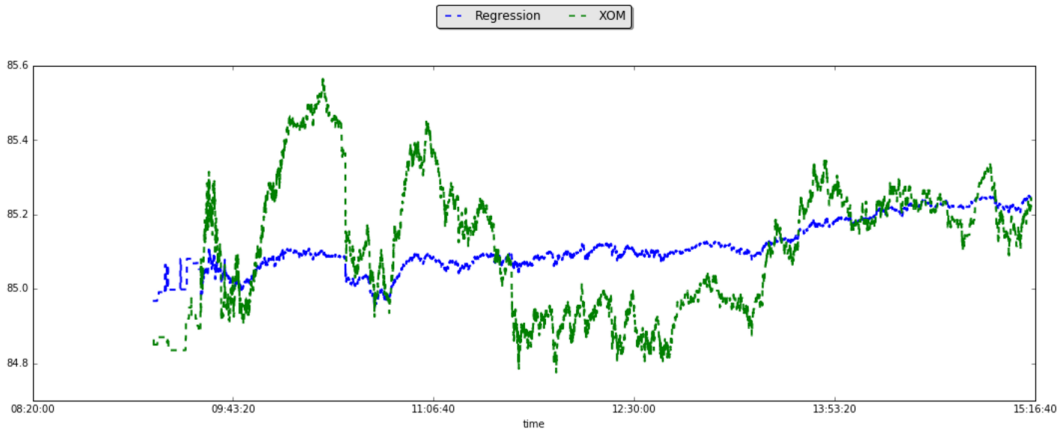
β is calculated using

$$(X^T W X)^{-1}(X^T W Y) \tag{3}$$

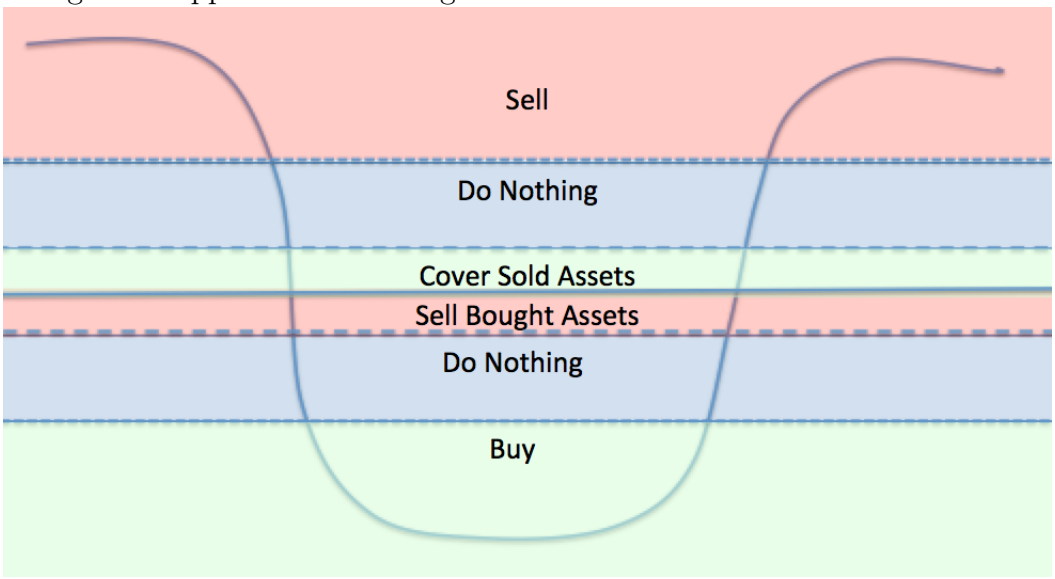
where Y represents the actual price points of the asset we are looking to trade. W is a diagonal matrix that is responsible for exponentially weighting our linear regression. In order to create our weighting scheme, we chose an alpha between 0 and 1 and populate the diagonal matrix using $W[\text{time\_step}][\text{time\_step}] = \alpha^{\text{total\_time\_steps}-\text{time\_step}-1}$. Note that there are faster ways to compute an exponentially weighted linear regression, as noted in the ACM article mentioned in the background section above, however for this project we choose to calculate the regression this way due to its simple implementation.

4.3 The Exploit

Once you can model the correlation between asset prices and find the "line of best fit" between the two, many options become available based on the assumptions that are made. The main assumption we based our algorithm from is that the asset we are trading, Y, will closely follow our regression, Estimated_Y, if Y is highly correlated with the assets that are used to create the regression. Under this assumption, if we see the price of Y deviating from the price of Estimated_Y, we assume that there will be a convergence in the near future. An example of this expected convergence can be found in the figure below:



This chart represents Exxon Mobile (XOM) vs a regression between Exxon Mobile and Chevron Corporation (CVX). Unfortunately, most assets (including these two) generally don't follow such a mean-reverting correlation, however this example taken from March 25th, 2015 highlights a trend we hoped to exploit.
Our general approach for trading can be found below:



This approach buys when it sees a far divergence of the asset price downward relative to the regression price, and sells when there is a high divergence upward. The algorithm sells all assets that were most recently bought when the asset has almost reverted to the regression price (while moving upwards), and covers the sold assets when reverting downwards toward the regression price. It is important to note that we only "Sell Bought Assets" when we are

reverting to the regression from below the regression (in other words moving up in price), and only "Cover Sold Assets" when reverting to the regression from above the regression (in other words moving down in price). Covering sold assets consists of buying back the assets we just sold, which simulates covering a short. We added a "Do Nothing" threshold because we found there was a lot of trades being executed with resulting profits that were too small to cover the bid-ask spread. The thresholds for determining when to trade will be discussed in the "Parameter Tuning" section.

This exploit can be used with many different approaches. Some of the approaches we looked at were:

- Pairs Trading - this consists of trading between based on divergence and convergence of two highly correlated assets. Many times, algorithms will trade both assets in opposite fashions (sell one, buy the other and vice versa) however during the 10 weeks we were only able to focus on selling just one of the assets.

- Basket Trading - this consists of tracking a regression based on an asset and all of its underlying assets. If we see a difference in the regression based on an underlying assets vs the index fund (for example DIA), then we trade assuming the fund will revert back to the regression.

- Trading ETFs based from the same index - For example trading SPY based on a regression RSP RWL RBP, which are all ETFs based on the S&P500 index.

4.4 Parameter Tuning

The parameters to be tuned for this algorithm are:

- The amount to weight old data (α). The amount by which to weight the data is a value between 0 and 1 exclusive (1 would be weighting everything the same). Our results showed that the closer to 1, the better. Most of the time we used .9999 as our α.

- The threshold at which to execute a trade (Buy or Sell). This is one of the most important parameters to tune and we still feel a lot more exploration needs to go into determining this value. To start, we used an exponentially weighted average of the difference between Y and the Estimated_Y (using the same α) over the data we trained on. We realized that just using the exponentially weighted average didn't produce as many trades as we hoped for at certain time period, so we generally divided the value by 1.2 or 1.6 in order to execute more trades.

- The threshold at which to cover a trade. At this point we have just divide whatever value our threshold to execute a trade is by 2, so that we can create the "Do Nothing" space. Again, this value should be tuned further but we found promising results with these starting points.

- Amount to trade. We only traded in sizes of 100 shares, however it would be prudent to base the amount traded on some measure of certainty. This could be the distance by which the price of Y deviates from Estimated_Y, however it is not as simple as trading more when the price differs further. There is an upper bound at which we'd want to trade for many reasons, one of them being the presence of a potential sweep, which may alter the price of one asset for an extended period of time. In this case we wouldn't want to trade at all, even with strong signals of deviation.

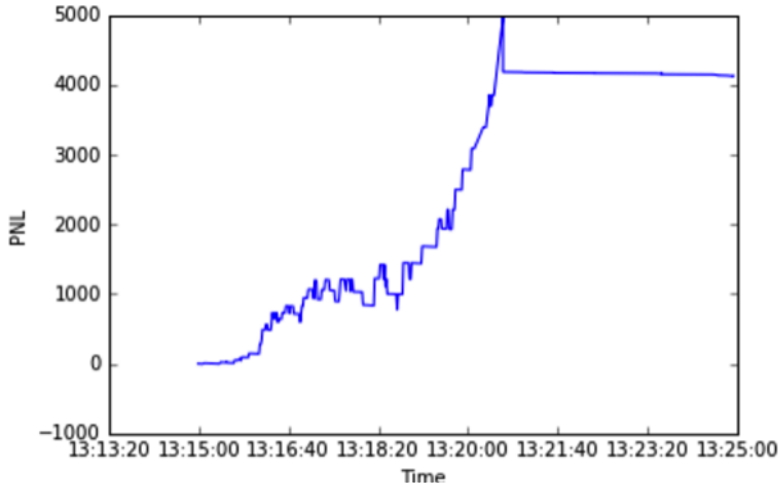
- The amount of data to train on. This amount varied by the time interval we were using, primarily due to the time it takes perform computations on the data. When operating on a time interval of 1 seconds, we generally used 2-4 hours of data to train on. When operating on the timescale of 100000 microseconds (1/10 of a second), we tried to cut the amount of time to train the data proportionally.

- The time interval at which to operate on. As mentioned above, the time interval to operate on was based on a trade off between a more reactive trading system and one that operates on a broader scope on data, due to the time constraints. We chose to operate on a timescale of 1 seconds because it seemed to be a good balance of the two.

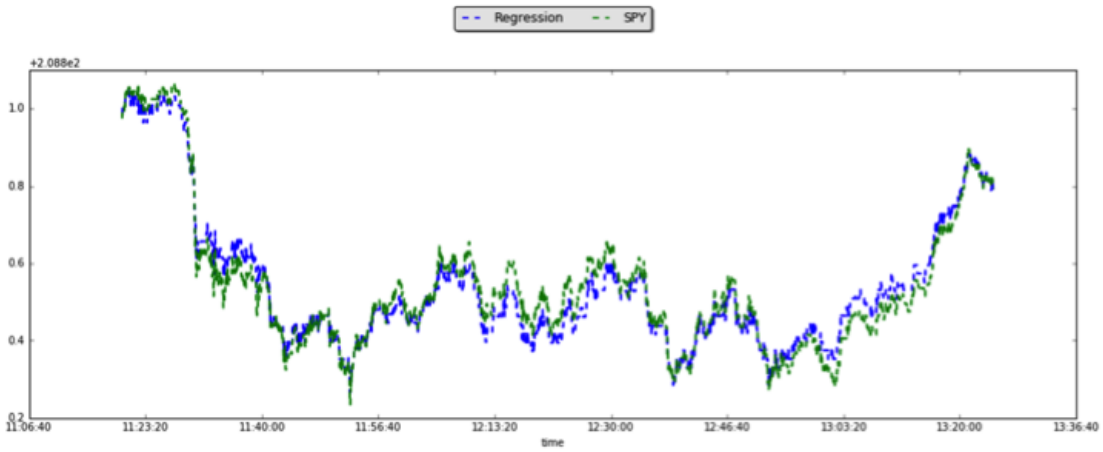
- The assets to trade. We looked at many different assets to trade, most of which we chose because they were highly correlated at some point in time. Some examples of assets we traded were (trading the first asset) XOM vs CVX, GOOG vs. GOOGL, SPY vs. RSP RWL PBP, DIA vs underlying assets.

4.5 Results

The most promising result we got using the strategy is below:



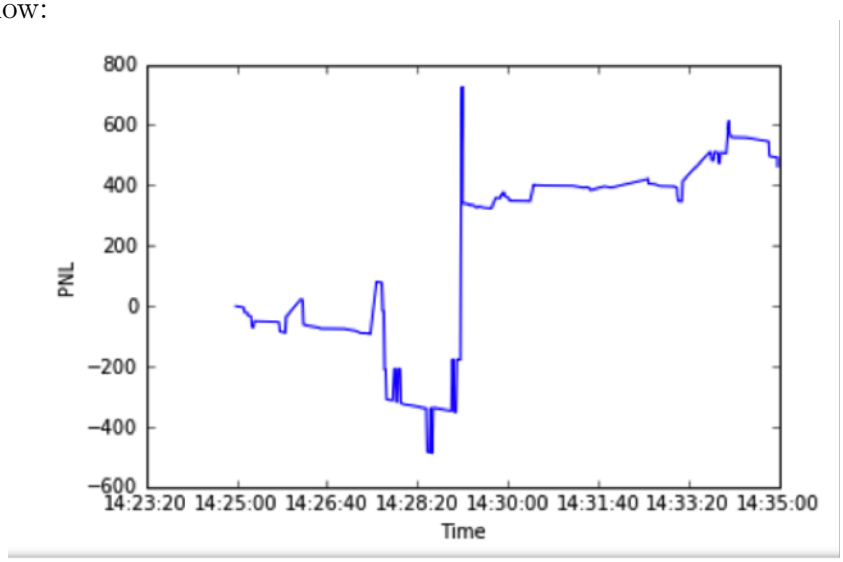
These results took place on March 19th, 2015 trading SPY with a regression relative to RSP, RWL, and PBP. The first regression was based on data from 11:10am to 1:15pm with an interval of 1 second. The algorithm ran for 600 iterations, at each time step shifting the start and end times to run the regression up 1 second. In other words, our algorithm started trading at 1:15pm and ran for 600 seconds (10 minutes). The corresponding prices were:



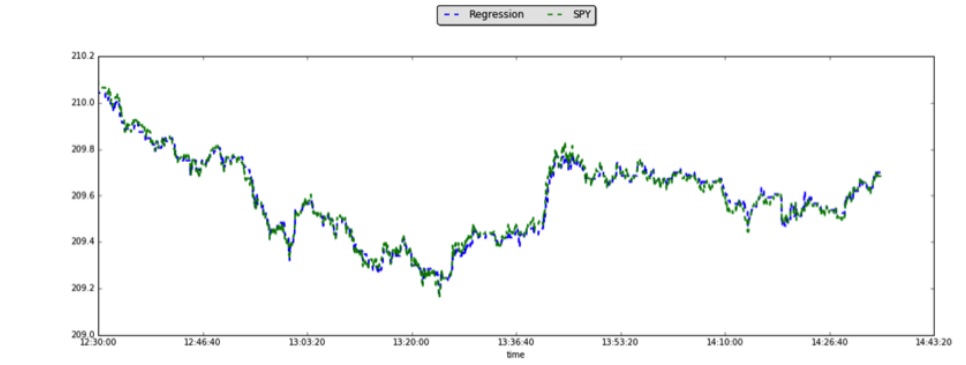
At first glance, it is apparent that the stock price went up quite a bit in the time period we traded, however our algorithm does not operate on signals based on change in price relative to its own price in the past, it is comparing its price at a given time step relative to the regression. As you can see by the prices, the regression was slightly above the price of SPY for most of the trajectory upward. It is possible the other S&P500 ETFs were reacting quicker to changes in the index than SPY was, which allowed us to predict where the price for SPY would move, although there are many other

factors that could account for this foresight.

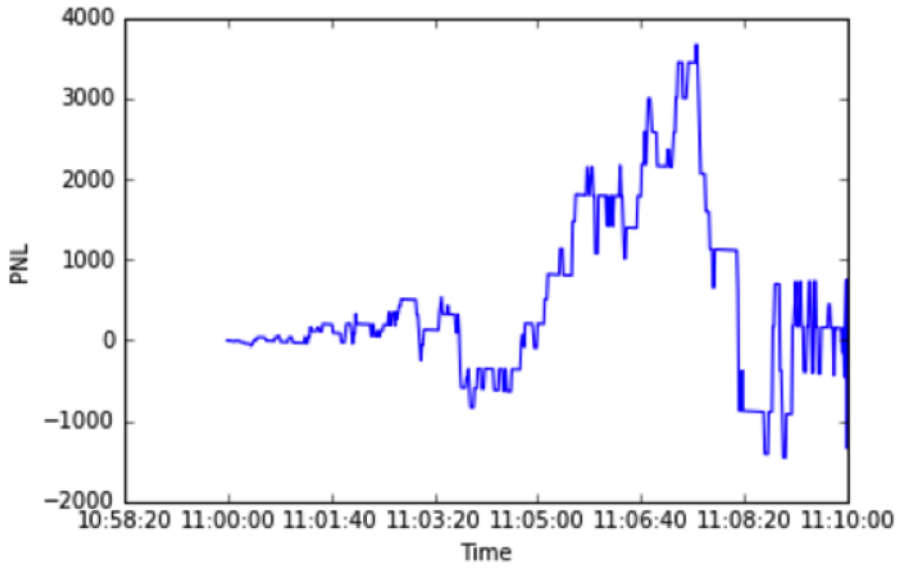
Another positive result using the same ETFs and duration but with the first regression based on data from 12:20pm to 2:25pm on June 26th, 2015 is below:

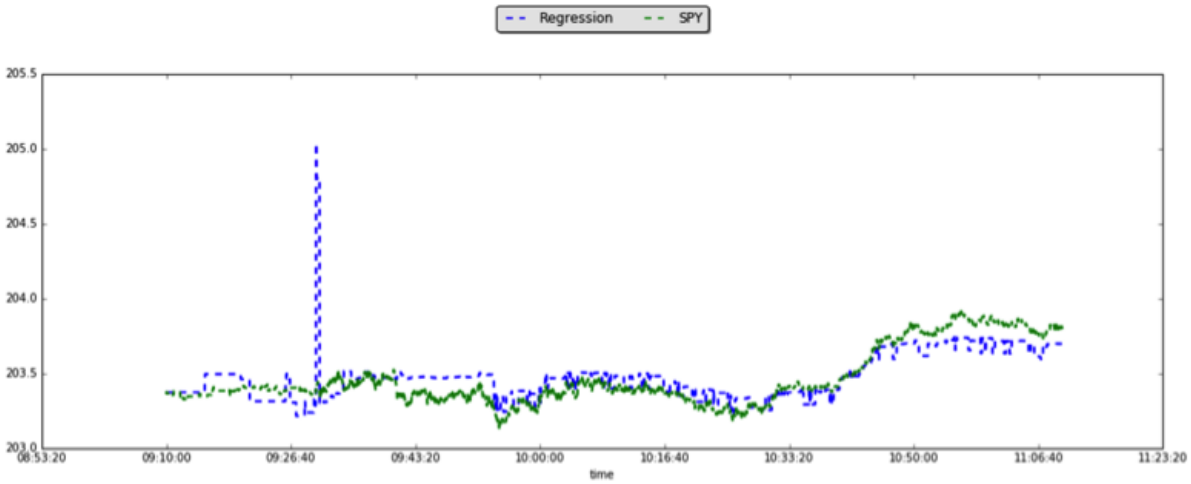


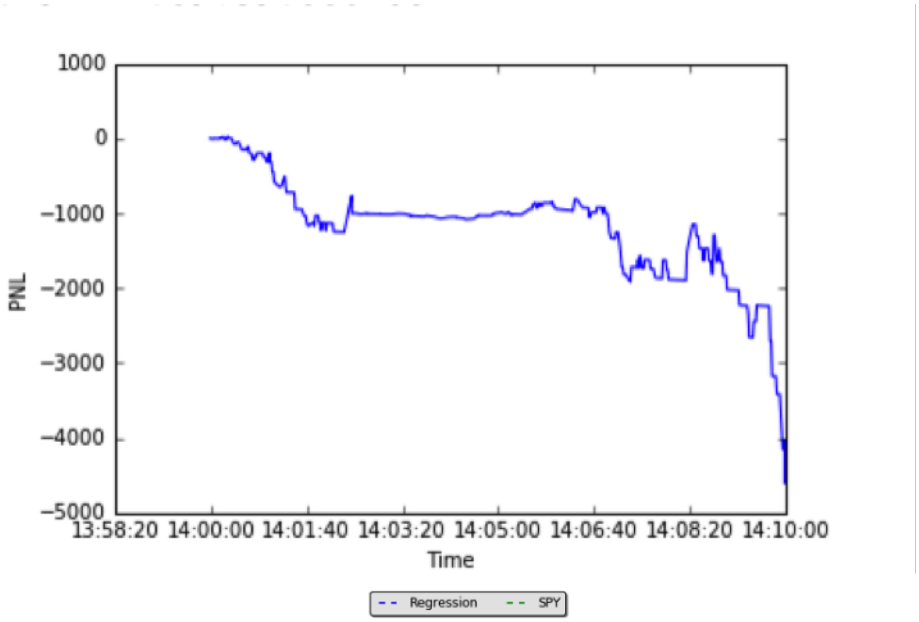
The associated prices were:

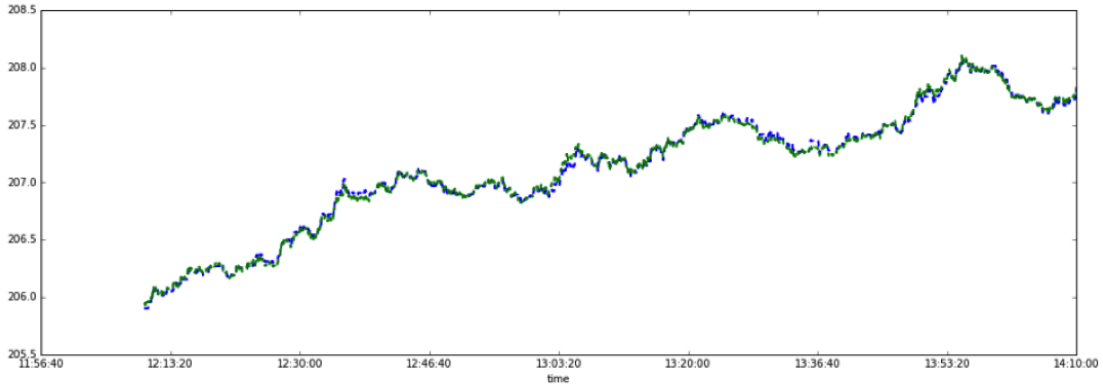


Unfortunately, these positive results could not be replicated at will using these ETFs. Some other results using different dates and the same ETFs can be found below:

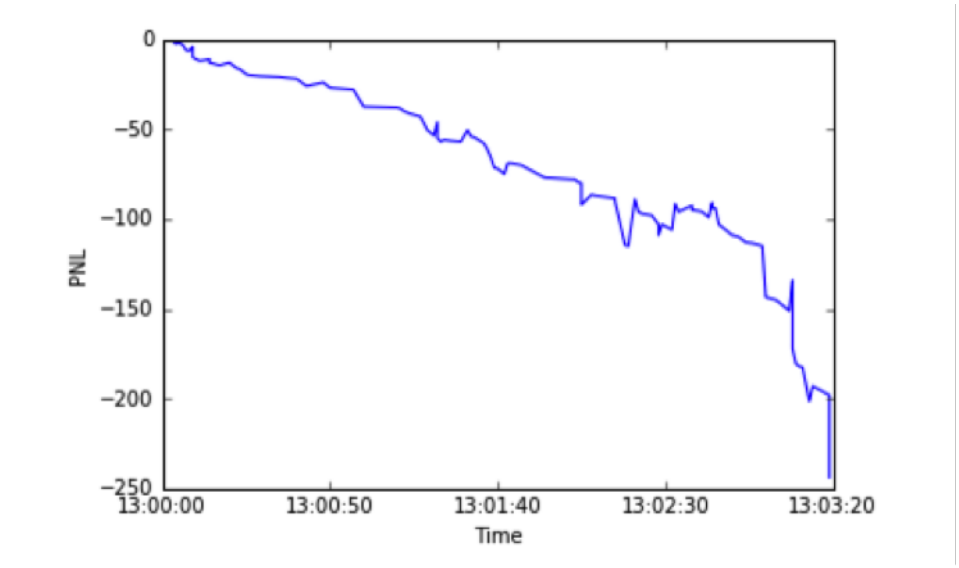

4000
3000
2000
1000
0
−1000
−2000
PNL
10:58:20 11:00:00 11:01:40 11:03:20 11:05:00 11:06:40 11:08:20 11:10:00
Time


Regression
SPY
205.5
205.0
204.5
204.0
203.5
203.0
08:53:20
09:10:00
09:26:40
09:43:20
10:00:00
10:16:40
10:33:20
10:50:00
11:06:40
11:23:20
time

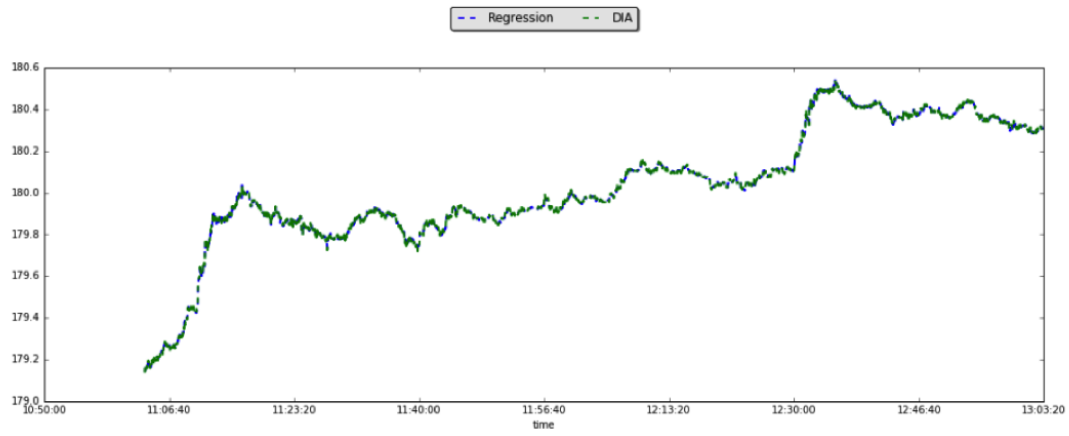




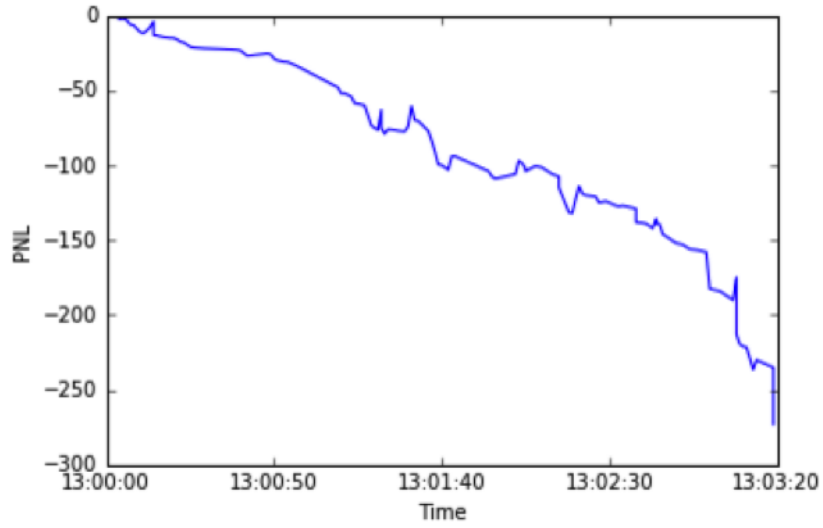
We also looked at trading the Dow (DIA) vs it's underlying assets. There were no positive PNLs from our testing but here's a result from April 14th, 2015 with two hours of training data and 200 seconds of trading.

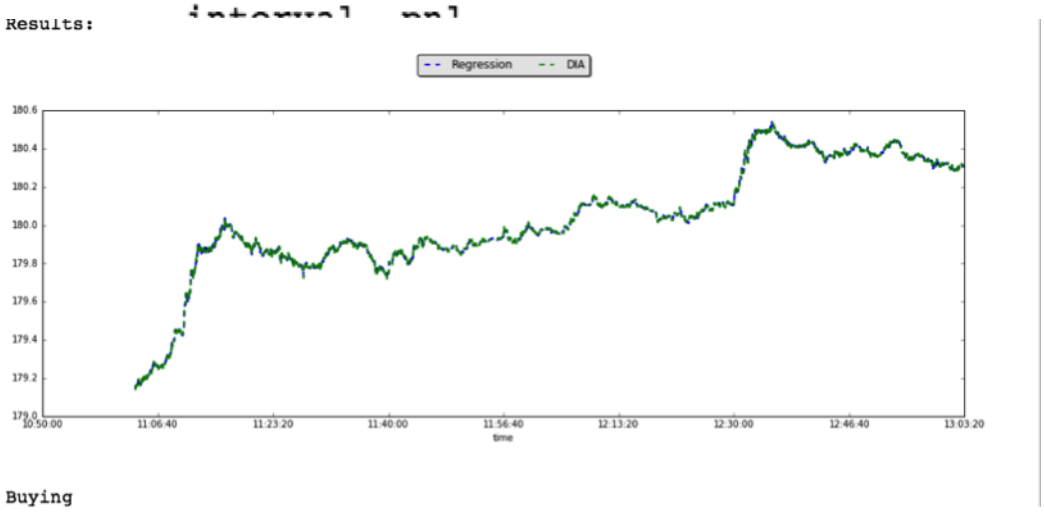


Results:

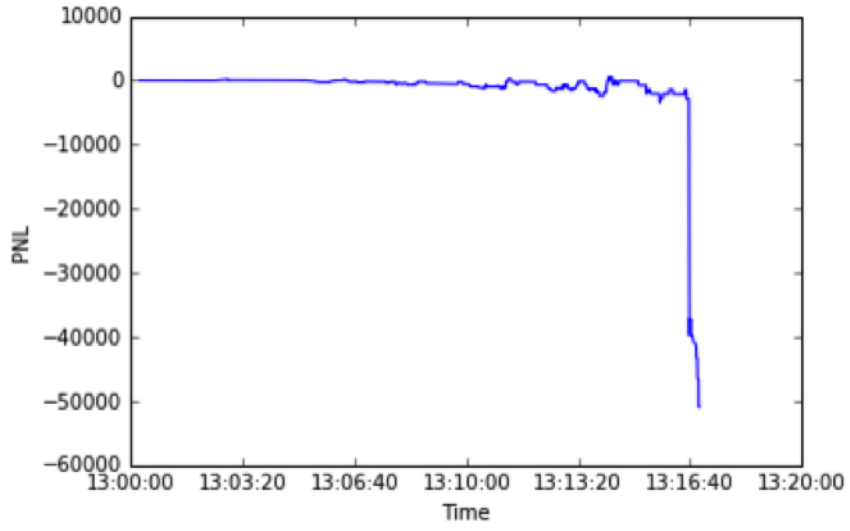


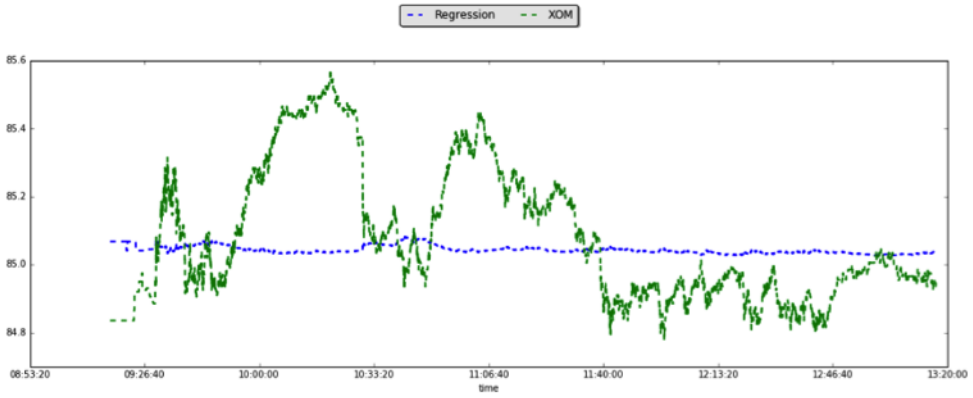
The above results were using a trading threshold of the exponentially weighted average of the difference in price between our asset and regression based on the training data divided by 1.6. In order to make a more strict threshold for choosing when to execute a trade, we ran another test using a threshold of exp weighted average difference/1.2, found below:



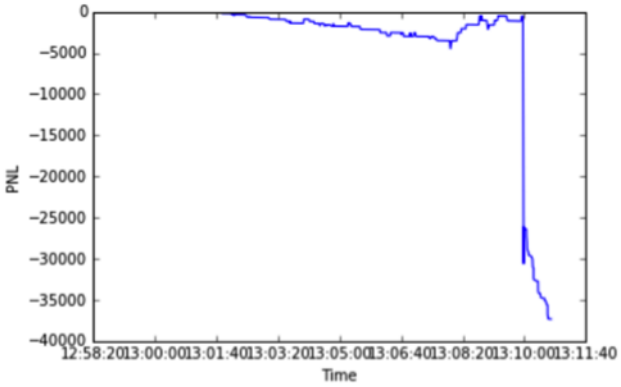


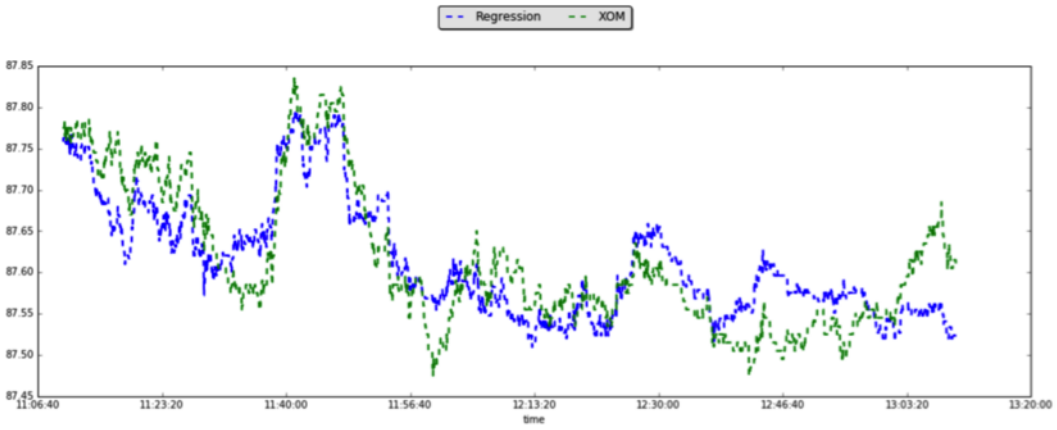
We also tested pairs trading using Exxon Mobile (XOM) and Chevron Corporation (CVX). One result, which used data from 9:00am-1:00pm March 25th, 2015 with 1000 iterations is found below:





A second result, which used data from 11:00am-1:00pm April 20th, 2015 with 1000 iterations is:





4.6 Conclusions

Although we were not able to find an algorithm that consistently turned a profit we were very pleased with these results, which used very naive assumptions and relatively little complex math to hedge risk, assess volatility, and tune assets and parameters. We feel this basic approach to trading has a lot of potential in many fields if enough time is spent tuning all of the complex parameters involved and making computation efficient.

5 Strategy 3: Index Fund Arbitrage

5.1 Background

We implement a trading strategy based on the correlation between an exchange traded index fund and the stock market index which it tracks.

The first index fund was created in 1970, and index funds have become increasingly popular with investors who seek the return profile of a stock market index without owning the individual stocks that make up the stock market index. The fundamental goal of the index fund manager is to minimize tracking error with respect the underlying index.

Stock market indices are generally categorized as price-weighted or capitalization-weighted. The value of a price-weighted index, (e.g. Dow Jones Industrial Average), is computed by summing the value of it's individual components

$$P_I = c \sum_i P_i$$

If the price of a certain component changes by Δp_a, then the price of the tracking index should change by a factor of

$$\Delta P_I = \Delta p_a \frac{p_a}{\sum p_i}$$

We assume that any deviation from expected change in price of the index fund is a temporary market inefficiency, and price will converge to the expected value. By buying the index fund when it is undervalued and selling the index fund when it is overvalued, a trader can exploit this price discrepancy for profit.

5.2 Approach

The Dow Jones Industrial Average (DJIA) is a price-weighted index of 30 large-cap stocks that are picked by the S&P Dow Jones Indices LLC on the basis of reputation, sustained growth, interest to a large number of investors, and accurately sector representation.

The SPDR Dow Jones Industrial Average ETF (DIA) tracks the DJIA, with a net asset value of $11.7 billion as of June 2016.

We implement a trading strategy using DJIA and DIA because it the DJIA is simple to calculate and is composed of a relatively small number of stocks, and the DIA is large and heavily traded.

The simplest trading strategy is to compare daily returns, and buy DIA if its percent return on the day is less than DJIA, and sell otherwise. However, while intraday fluctuations in price are almost always well tracked, a difference in price often lingers for time scales on the order of minutes to hours.

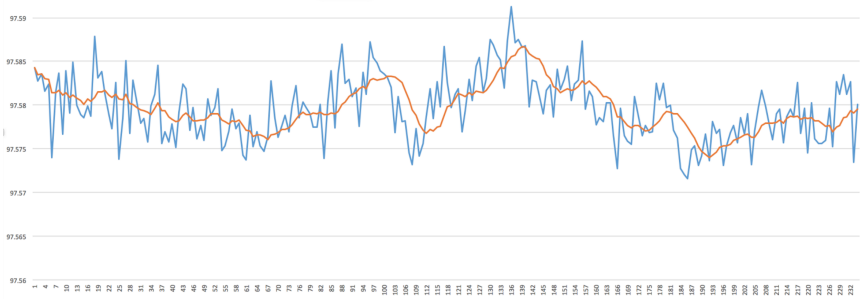
To compensate for this we compute a trailing average ratio between DIA and DJIA. If the current ratio is less than the trailing average, this indicates that DIA is undervalued, even taking into whatever price discrepancy is lingering in the market.

The trailing average ratio is computed as follows, for a given number of ticks n and present time t_f

$$\bar{r} = \frac{1}{n} \sum_{t=t_f-n}^{t_f} \frac{DIA_t}{DJIA_t}$$

We expect the current ratio to equal the trailing ratio, and thus predict

Figure 1: Ratio of DJIA vs. DIA, with trailing average

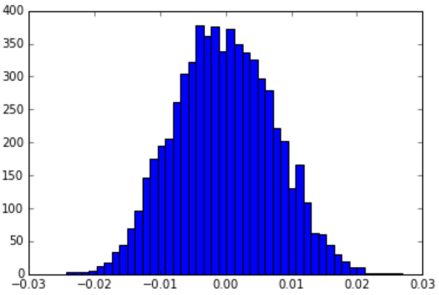


the price of DIA

$$\begin{aligned} r - \bar{r} &= 0 \\ \frac{DIA}{DJIA} - \bar{r} &= 0 \\ DIA &= \bar{r}(DJIA) \end{aligned}$$

By subtracting expected price from current price, we get an estimate for the near-term price change.

Figure 2: Discrepancy between actual and expected price of DIA (measured once per second)

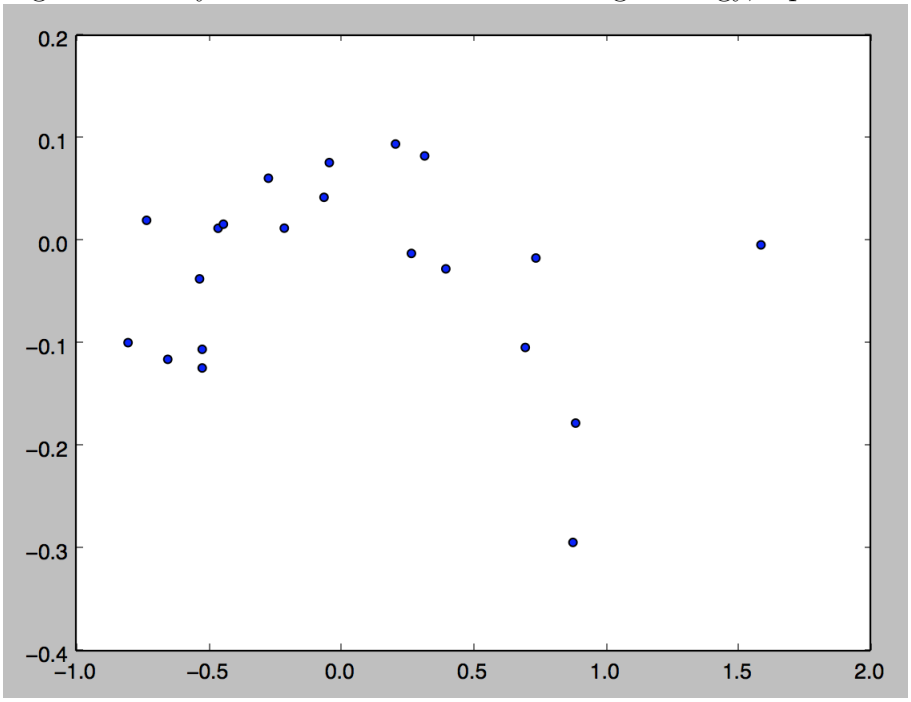


Since the DIA tracks the DJIA relatively closely, a lot of the price inefficiencies are too small to trade on. We only place orders on the largest discrepancies in order to maximize the probability of profit. Through experimentation, we converged on a strategy of placing a buy[sell] order for DIA at the bid[ask] if the current price is $.01 less[more] than the expected price.

As a final optimization, we avoid trading during the beginning of the day (before 10am), to avoid higher than average volatility.

We plot daily return of DIA vs. return on this strategy (in April 2016), as implemented in the Thesys simulator:

Figure 3: Daily return of DIA vs. our trading strategy, April 2016



5.3 Conclusions

This strategy yields decent returns when the market is relatively flat. Future work should be done to reduce losses on days with large market changes.

6 Strategy 4: Order Book Pressure

6.1 Background and Literature Review

The limit order book for an asset offers all available information on the prices and quantities traders are willing to buy and sell an asset for at a snapshot in time. Utilizing the information presented in the order book, especially over a time series, gives us valuable insight into market microstructure and may provide signals into short-term price movements.

The order book pressure strategy we explore in this section is, at its core, very simple and follows from basic economic principles of supply and demand. Simply put, if demand (the bid queue) is significantly greater than supply (the ask queue), we expect the price to increase in the short term, and vice versa.

This strategy is built on the work presented in a few previous papers.

The idea of utilizing insights on market microstructure from the limit order book in high-frequency trading is explored in Avellaneda et al.[1]. In a later paper by Avellaneda et al., the use of level-I quotes only is used in a pressure strategy.[2]. A paper by Lipton et al. in 2013 presents the primary basic formula we use to evaluate limit order imbalances:

$$I_t = \frac{(Q_B - Q_A)}{(Q_B + Q_A)}$$

...where Q_B is the size of the bid queue and Q_A is the size of the ask queue. It is hypothesized that the value of I_t is correlated to short-term price changes.[4]

The high-frequency team from the 2015 MS&E 448 class also investigated an order book pressure strategy. They initially attempted using machine learning techniques, specifically SVM and decision tree, to make buy/sell/hold predictions. However, the performance of these techniques, along with other order book imbalance formulae, was limited compared to the simple solution proposed in Lipton et al. Therefore, they ended up a solution which calculated I_t with 8 levels of the order book at three intervals separated by 7.5 seconds, and using these values to determine a buy/sell signal.[5]

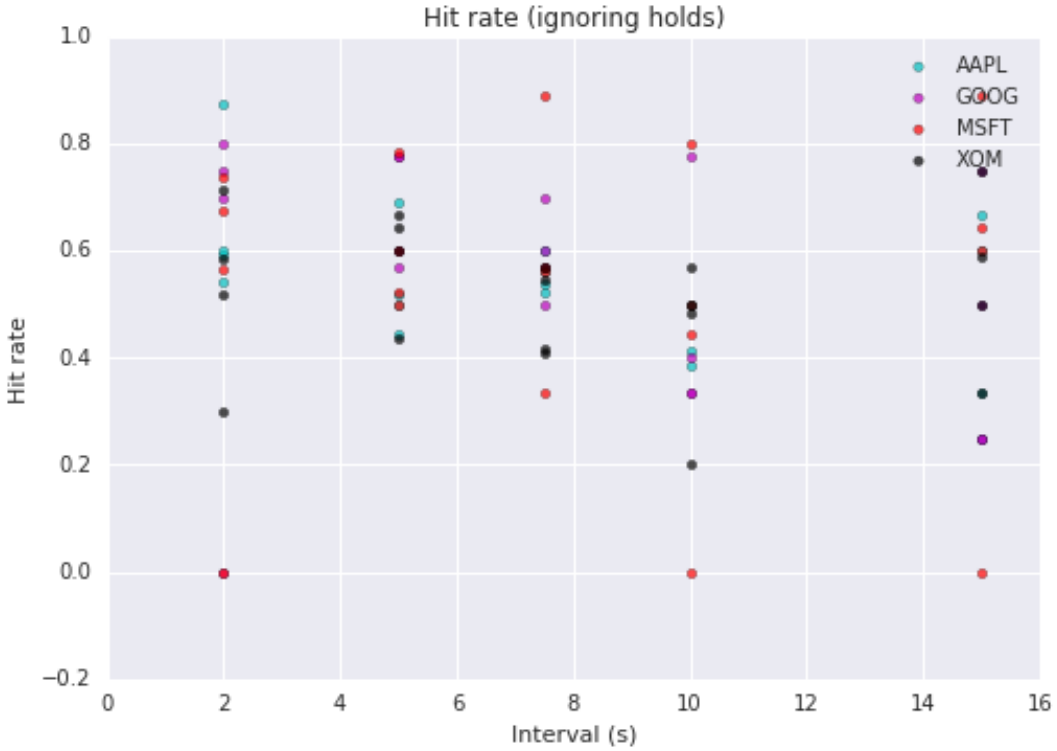
Our strategy continues the work of Aryan et al. from last year, utilizing similar ideas of taking multiple snapshots separated by a small interval and using these I_t values' momentum and instantaneous pressure to predict short-term price movements.

6.2 Signaling Mechanism

We used a similar mechanism as Aryan et al. to surface buy/sell signals; we simply take 3 snapshots of I_t separated by t seconds. We then use logic that ensures that $|I_t|$ is above a certain threshold, and that the change between I_{t-2}, I_{t-1}, and I_t is sufficient.

The main improvement of our signaling model over that of the 2015 team is that it appears that many of the hyperparameters chosen in the previous paper were fairly arbitrary. For instance, they decided on 7.5 seconds and manually tuned the threshold values at which a signal was generated. The interplay between the momentum and current pressure indicators also appeared to be fairly primitive logic. Finally, the method used in the 2015 paper for actually calculating I_t was unclear; the paper mentioned weighting bids and asks further from the spread lower, but it is not clear whether this was used.

To demonstrate the significance of various hyperparameters, consider the difference in hit rates when simply varying the value of t for both intervals:



As such, there were up to 40 values and non-continuous settings that needed to be optimized for the signaling mechanism alone. These included:

- Number of levels to use in the order book for pressure calculation. Both a constant number of levels, e.g. $l = 8$, and a variable number of levels (depending on percentage difference from best bid/ask) were tried.
- Decay function used to de-weight levels further from best bid/ask. This was a discrete choice. We attempted choices ranging from no decay, exponential decay based on level or percentage difference, etc. About 10 functions were tried (each with their own parameters), though this was not the most scientific. We ended up with an inverse function with respect to the percentage difference from the best offer. For instance, on the bid side:

$$Q_b = \sum_{l=0}^{L} weight(\frac{1 - p_l}{p_{best}}) \sum_{o=0}^{l_{orders}} q_o$$

$$weight(d) = \frac{1}{\gamma d + \beta}$$

- Time intervals between I_{t-2}, I_{t-1}, I_t.
- Threshold of current pressure at which to generate signal.
- Thresholds of momentum between I_{t-2}, I_{t-1}, I_t to generate signal.
- Trade-off between momentum indicators and instantaneous pressure indicators. Logic here was also non-continuous.
- Weighting between Q_b and Q_a, as for some decay functions it was imbalanced.

As can be seen, no off-the-shelf optimization algorithm could optimize all these hyperparameters, especially considering the discrete choices. As a result, I implemented from scratch an algorithm which took inspiration from gradient descent and differential evolution. The algorithm did not scan the vast majority of the possible parameter space due to computation time and was not the most scientific, but was quite effective and avoided overfitting significantly.

6.3 Execution Strategy

The execution strategy was significantly more simple than signaling for two reasons. First, it would be computationally intractable to attempt to optimize too much behavior, due to the extreme slowdown when turning on execution. Secondly, an early attempt at order execution gave very good results.

The simple strategy is as follows:

1. When signal is raised, place buy/sell at market price in quantity proportional to signal strength.
2. Place opposite order at 50 basis points' profit price.
3. Add stop loss at 100 basis points' difference in other direction.

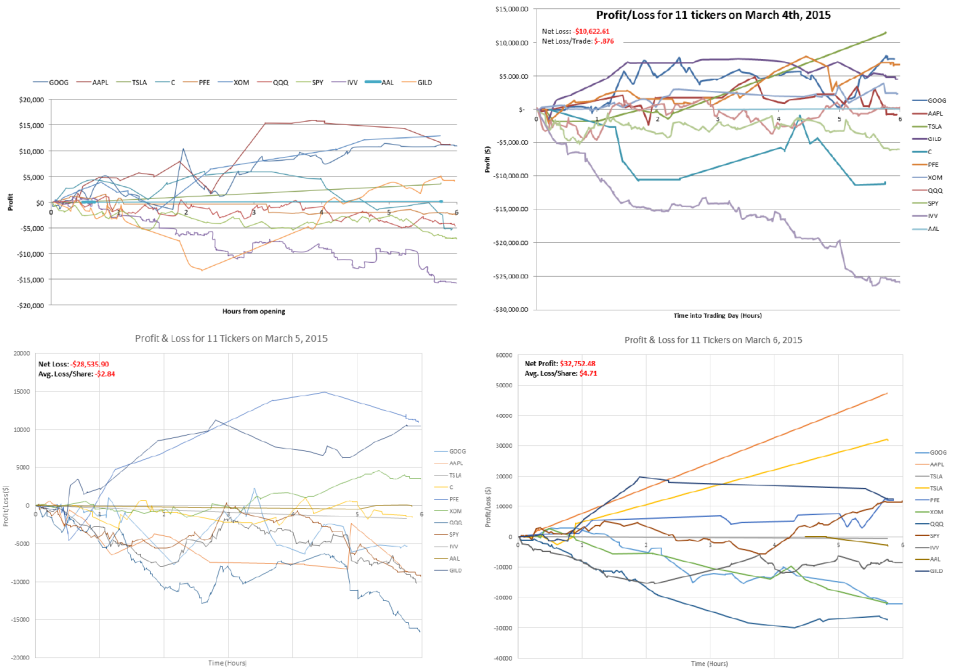
The parameters of quantity and 50/100 basis points were manually tested to decent effect, though could be further optimized.

6.4 Results

When trained on an assorted training and validation set of trading days excluding days used in testing by the 2015 HFT group, and tested on an assorted held-out test set, of which 50 percent of days were those used by the

2015 group, the strategy achieved very good returns (even when paying the spread). The strategy outperformed all iterations of the 2015 team's order book pressure algorithm:

Compared to the performance of the 2015 team's algorithm, which had limited tuning of parameters and access to less training data:



6.5 Discussion and Further Work

Overall, we were fairly pleased with the results this strategy achieved, as it was our only strategy which was able to consistently turn a profit without significant overfitting. It also speaks to how important continuous iteration upon earlier strategies is, as this strategy initially stemmed from the simple idea of counting the difference between size of the bid queue and ask queue, and was then improved upon by the 2015 team. By having access to far more data and greater ease of prototyping, we were able to further tune and improve upon the strategy.

That being said, there are some significant holes in the current implementation. For instance, the hyperparameters were trained to generally be constant, while it is fully reasonable that the optimal parameters vary among different assets. For instance, a more heavily traded asset might enjoy a faster interval time.

As seen in the profit/loss graph above, it is also clear that though the algorithm generally turns a profit, it is possible to tank as well. More sophisticated execution mechanisms and possibly better tuning of buy/sell thresholds could be very effective.

7 Strategy 6: Intelligent Market Making

7.1 Background

A market making strategy seeks to post both bid and offer quotes hoping to get filled on the roundtrip in rapid succession, thereby capturing the spread. The market maker faces both inventory risk (where the price moves before she can complete a roundtrip) as well as adverse selection (where the counterpart is more informed) and the she faces losing trades each time. It is believed that the spread compensates the market maker for these two risks. [3]

As such, our goal is to maximize profit by minimizing these risks. To do this, we ask: where do we place our bid/ask quotes such that we get executions while getting the best prices?

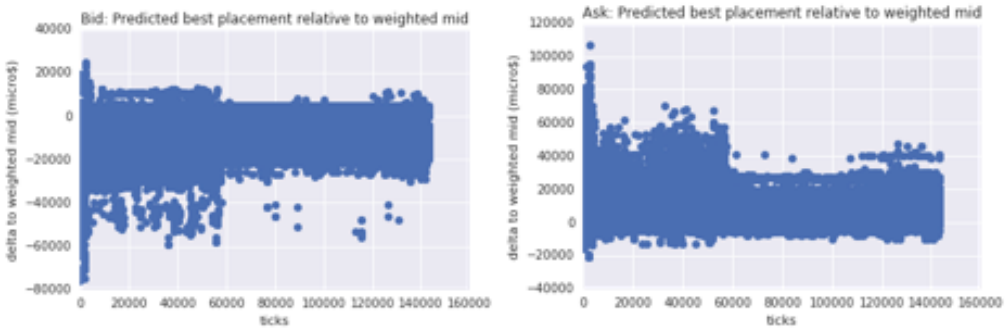
There is a tradeoff here. We can post thin quotes (close to the best bid/ask) and get filled more often or thick quotes (far away) but get less executions. Additionally, if a very aggressive comes and an eats up a lot of liquidity on one side of the book, the both the thin and thick quote could get filled while the thick quote got a better price and attains less risk. Our goal is to predict where to place our orders to get the latter type of execution.

7.2 Machine Learning Model

To make these predictions, we will utilize a machine learning model - the random forest. We choose this because it is off the shelf and easy to tune. We will discuss the disadvantages later.

We aggregate at each execution statistics about the current and previous 2 ticks. These include information about each of the 3 levels of book data (including spread, weighted midpoints, deltas of the bid/ask quotes at each level, volumes at each level, rate of recent trades, etc). We end up with $P = 54$ features. Our response is the best recently executed price on the bid and ask side (represented as the delta to the mid price) within 3 ticks. Thus, we train two separate models: one to predict bids and one to predict asks. There are about $N = 30,000$ training events per day per symbol.

After training a random forest with $n_{trees} = 10$ and default parameters with sci-kit learn on a day of AAPL on 2015-01-21 and running on out of sample next day data, we have the following models:



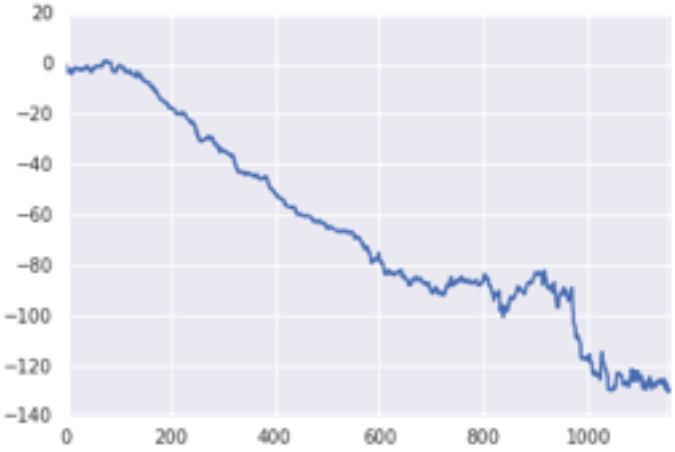
which we note produces some interesting insights. It knows when to produce thick quotes far from the bid ask spread, but on average produces predictions around 3 cents away from the mid. This is in line with typical market making strategies. Insightfully, the model knows to post wide spreads at opening auction.

7.3 Trading Rules

Based on the results of the model trading, we turn the model into a trading strategy and evaluate its performance. To make things easy to test, we train on the previous trading day's data and produce both bid and ask prediction models using the random forest.

During the next day, we stream in the last 3 ticks, featurize the data as we did when training the data, feed the features into the models and produce predicted bids and asks. Then, we only trade if the predicted bid is less than the current best bid (and vice versa for the ask). If we decide to trade, we immediately send a limit order for the predicted price and update the order to match our prediction if the market moves. Once the order executes, we switch sides of the trade. We keep orders to 25 shares to limit market impact. By immediately switching sides, we attempt to buy on one tick then sell on the next, a very risk adverse strategy.

7.4 Results



A graph of results for trading over 30 minutss of AAPL on 2015-01-22 with models trained on the previous day's data is above, which shows it losing a lot of money.

We believe the trading strategy is losing money due to its intense risk aversion. By attempting to buy on one tick and sell the next tick, it does not wish to hold any inventory at all. If the price moves down, we would like to accumulate a long inventory, and getting better and better prices each time, believing the price will mean revert. But, the current strategy forgoes this in favor of selling on the next tick regardless of the price move down!

But, there is reason to believe the model has alpha. Since we are trying to predict the best execution price (which is guaranteed to be further away from the mid than the best bid/ask) and we are indeed getting executions, that means we are getting better prices than a naive marketing maker posting at the best bid and ask.

7.5 Discussion

The intense risk adversity of the market making strategy makes it unprofitable. But, we believe the machine learning models themselves perform well. To make our market making strategy work, we must also intelligently deal with inventory risk (perhaps using stochastic models as in current literature). Currently we have an intelligent way of posting quotes but no smart way of managing the positions.

Additionally, as with any machine learning model, more fine tuning or parameters and optimization would help. As always, more and higher quality features would allow better prediction.

Finally, we note that a random forest may not be a tractable model in practice. Because it is necessary to evaluate many decision trees, it slow to discover the predicted bid/ask. Market makers must react to information quickly so this evaluation time is significant. Many of today's most sophisticated market makers (e.g. Jump Trading, Citadel) use linear regression (simply evaluating a dot product) for quick prediction for this very reason. We used a random forest because it is easy to train and gives good accuracy out of the box.

References

[1] M. Avellaneda and S. Stoikov. High-frequency trading in a limit order book. *Quantitative Finance*, 8(3):217–224, 2007.

[2] Reed J. Avellaneda, M. and S. Stoikov. Forecasting prices from level-i quotes in the presence of hidden liquidity. *Algorithmic Finance*, 1(1):35–43, 2011.

[3] Michael Durbin. *All About High-Frequency Trading*. McGraw-Hill Education, New York, 1 edition edition, August 2010.

[4] Pesavento U. Lipton, A. and M.G. Sotiropoulos. Trade arrival dynamics and quote imbalance in a limit order book. *Quantitative Finance*, 2013.

[5] Vikram Ramakrishnan Thomas Stephens Omid Aryan, Felix Boyeaux. High frequency price prediction: Leveraging limit order book momentum and imbalance. 2015.

[6] QUSMA. Doing the Jaffray Woodriff Thing (Kinda), Part 1, January 2013.

[7] Jack D. Schwager. *Stock Market Wizards: Interviews with America's Top Stock Traders*. HarperBusiness, New York, rev upd edition edition, April 2003.