Machine learning techniques for price change forecast using the limit order book data

James Han[*] Johnny Hong[†] Nicholas Sutardja[‡] Sio Fong Wong[§]

December 12, 2015

Abstract

We study the performance of a multi-class support vector machine (SVM) approach proposed by Kercheval and Zhang (2014) in forecasting mid-price changes using information provided by the limit order book. Various Machine Learning algorithms are tested using data on a typical trading day.

1 Introduction

With the growth of High Frequency Trading (HFT), electronic trading systems have been adopted by many established exchanges, and increasingly more of the volume of daily trades is attributed to high frequency firms. With the advance of technology, the focus of trading has transitioned from a quote-driven market to a order-driven market. The central object to study in the framework of order-driven market is the limit order book, which contains information about traders' intention to buy or sell at a certain price for a particular number of shares.

Kercheval and Zhang (2014) builds a SVM multi-class classifier for forecasting price changes using the limit order book. The predictors are divided into three categories: the basic set, the time-insensitive set, and the time-sensitive set. The basic set contains 10 levels of prices and volumes in the limit order book. The time-insensitive set contains the bid-ask spreads and mid-prices, prices differences, mean prices and volumes, and accumulated differences for the 10 levels. The time-sensitive set contains the price and volume derivatives , average intensity of each type of orders (limit bid/ask, market bid/ask, cancellation bid/ask), relative intensity indicators, and accelerations (market/limit) in the 10 levels. The response variable is a label in one of the three possible classes: upward mid-price change (U), downward mid-price change (D), and stationary mid-price change (S).

2 Overview of Support Vector Machines

The essential idea of SVM is to find the maximum-margin separating hyperplane that divides the data points into classes. Consider the binary classification problem with the training data set $\{(z^{(i)}, y^{(i)})\}_{i=1}^m$, where $y^{(i)} \in \{-1, +1\}$ denotes the label for the ith observation and $z^{(i)}$ denotes the feature vector associated with the ith observation. For simplicity, assume that the data points are linearly separable; that is, there exists (w, b) such that $y^{(i)} \left[\frac{w^T z^{(i)}}{||w||} + \frac{b}{||w||}\right] > 0,\ i = 1, \ldots, m$. Define the geometric margin of (w, b) with respect to the training set $\{(z^{(i)}, y^{(i)})\}_{i=1}^m$:

$$\gamma = \min_{i=1,\ldots m} y^{(i)} \left[\frac{w^T z^{(i)}}{||w||} + \frac{b}{||w||}\right]$$

and the functional margin of (w, b) with respect to $\{(z^{(i)}, y^{(i)})\}_{i=1}^m$ as $\hat{\gamma} = \gamma ||w||$.

The problem of finding the maximum-margin separating hyperplane $w^T z + b = 0$ can be formulated as the following optimization problem:

[*]Department of Statistics, UC Berkeley: hanweijian2012@berkeley.edu
[†]Department of Statistics, UC Berkeley: jcyhong@berkeley.edu
[‡]Department of EECS, UC Berkeley: nsutardj@eecs.berkeley.edu
[§]Department of IEOR, UC Berkeley: siofongwong@berkeley.edu

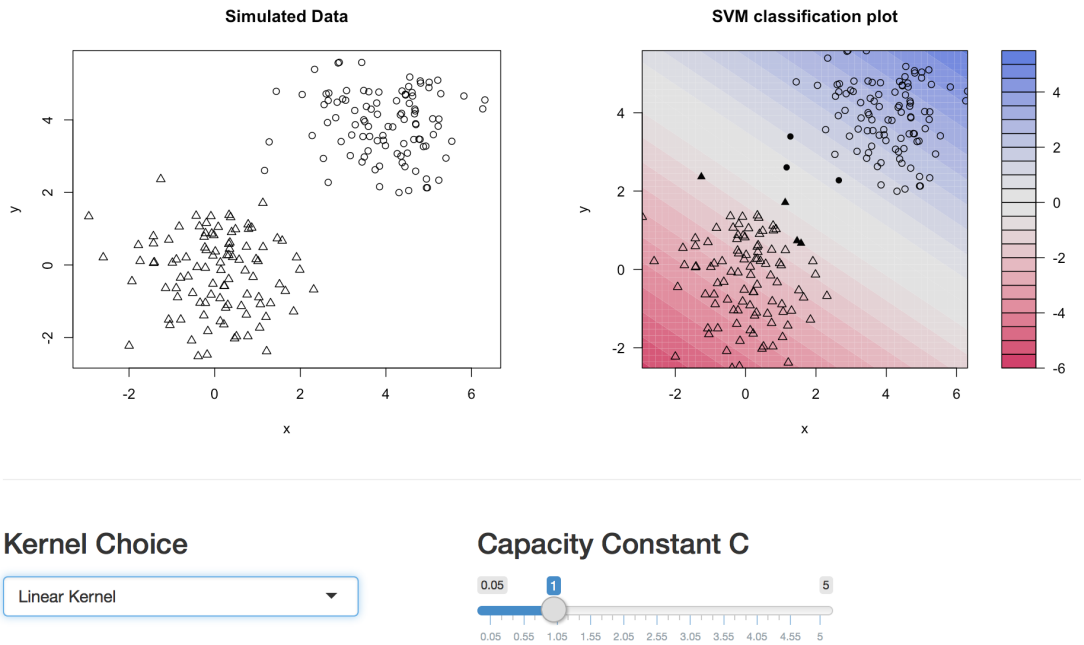

Figure 1: SVM classification (with linear kernel and $C = 1$) on simulated data. 100 points are simulated from a Normal$\left([0,0]^T, I\right)$ distribution (where I denotes the 2×2 identity matrix) and labeled as triangles. Another 100 points are simulated from a Normal$\left([4,4]^T, I\right)$ and labeled as circles. The left panel shows a scatterplot of the simulated data. The right panel shows the SVM classification. The filled triangles and the filled circles represent the support vectors.

$$\begin{aligned} \max_{\gamma,w,b} \quad & \gamma \\ \text{subject to} \quad & y^{(i)}\left[\frac{w^T z^{(i)}}{||w||} + \frac{b}{||w||}\right] \geq \gamma,\ i = 1,\ldots,m. \end{aligned}$$

The trouble with this formulation is that the constraint is non-convex. We can transform the optimization problem into a formulation without the non-convex constraint:

$$\begin{aligned} \max_{\hat{\gamma},w,b} \quad & \frac{\hat{\gamma}}{||w||} \\ \text{subject to} \quad & y^{(i)}(w^T z^{(i)} + b) \geq \hat{\gamma},\ i = 1,\ldots,m. \end{aligned}$$

Now the problem with this formulation is that the objective function $\hat{\gamma}/||w||$ is not convex. We introduce the constraint that $\hat{\gamma} = 1$.

Finally we obtain the following convex optimization problem:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}||w||^2 \\ \text{subject to} \quad & y^{(i)}(w^T z^{(i)} + b) \geq 1,\ i = 1,\ldots,m. \end{aligned}$$

To classify a new observation, we simply compute the sign of $w^T z^{(new)} + b$, where (w, b) is the solution to the optimization problem.

We may want to allow for a few misclassifications to avoid overfitting, especially under the presence of noise. To deal with either situation, we introduce slack variables $\xi \geq 0$ to the optimization problem:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi^{(i)} \\ \text{subject to} \quad & y^{(i)}(w^T z^{(i)} + b) \geq 1 - \xi^{(i)},\ i = 1,\ldots,m. \\ & \xi^{(i)} \geq 0,\ i = 1,\ldots,m. \end{aligned}$$

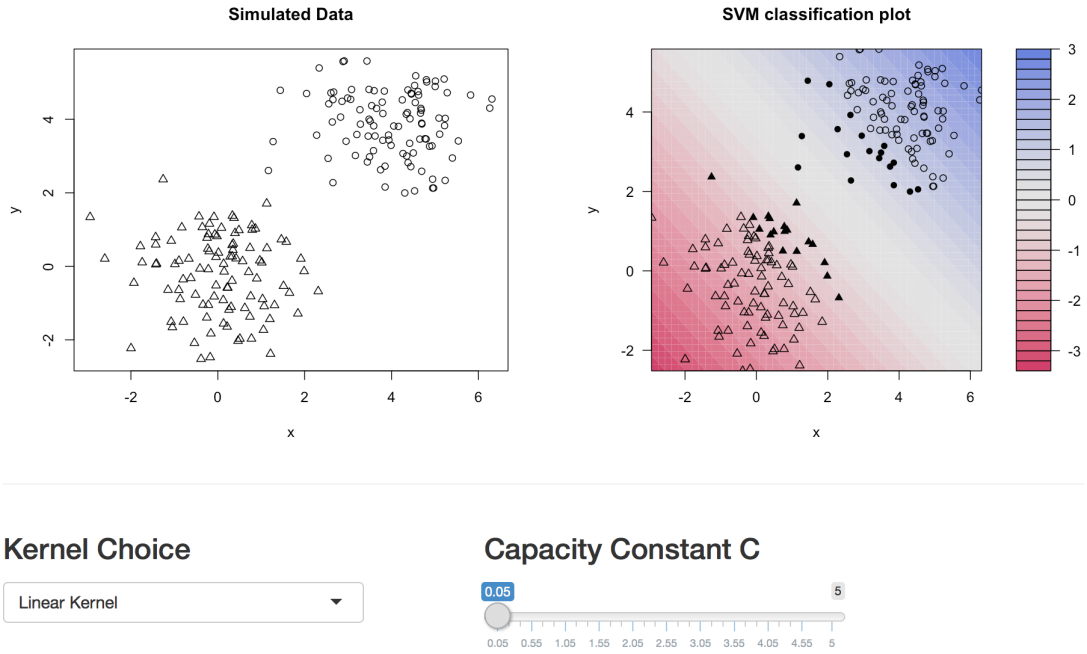

Figure 2: SVM classification (with linear kernel and $C = 0.05$) on simulated data.

where C is a regularization parameter which controls the leeway for misclassification (See Figure 1 and Figure 2).

If the data is not linearly separable, one should consider transformation of the data. We introduce the feature map $\Phi : Z \to V$, where Z is the space for the $z^{(i)}$'s and $(V, \langle \cdot, \cdot \rangle)$ is an inner product space. Our optimization problem becomes the following:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\langle w, w \rangle + C \sum_{i=1}^{m} \xi^{(i)} \\ \text{subject to} \quad & y^{(i)}(\langle w, \Phi(z^{(i)}) \rangle + b) \geq 1 - \xi^{(i)}, \ i = 1, \ldots, m. \\ & \xi^{(i)} \geq 0, \ i = 1, \ldots, m. \end{aligned}$$

Using the Karush-Kuhn-Tucker conditions, the dual problem can be derived:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{m} \alpha^{(i)} - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha^{(i)} \alpha^{(j)} y^{(i)} y^{(j)} \langle \Phi(z^{(i)}), \Phi(z^{(j)}) \rangle \\ \text{subject to} \quad & \sum_{i=1}^{m} \alpha^{(i)} y^{(i)} = 0 \\ & 0 \leq \alpha^{(i)} \leq C, \ i = 1, \ldots, m. \end{aligned}$$

Let $(\alpha^{(1)*}, ..., \alpha^{(m)*})$ be the solution of the dual problem. The solution (w_D^*, b_D^*) can be shown to be

$$w_D^* = \sum_{i=1}^{m} y^{(m)} \alpha^{(i)*} \Phi(z^{(i)})$$

and

$$b_D^* = y^{(j)} - \sum_{i=1}^{m} y^{(i)} \alpha^{(i)*} \langle \Phi(z^{(i)}), \Phi(z^{(j)}) \rangle,$$

where $j \in \{k : 0 < \alpha^{(k)*} < C\}$.

The SVM classification for a new observation z^{new} is

$$\hat{y}^{(new)} = \text{sign}\left(\sum_{i=1}^{m} y^{(m)} \alpha^{(i)*} \langle \Phi(z^{(i)}), \Phi(z^{(new)}) \rangle + b_D^* \right).$$

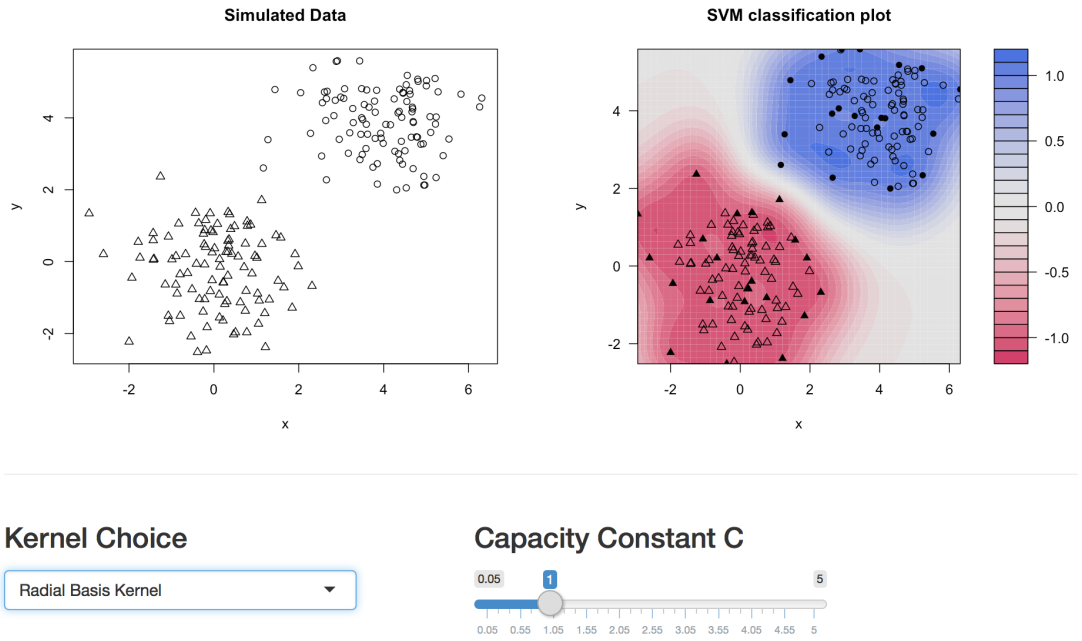

Figure 3: SVM classification (with radial basis kernel and $C = 1$) on simulated data. Compared to linear kernel (Figure 1) and Figure 2, the decision boundary is highly nonlinear.

We end our discussion about SVM with four remarks:

1. Let $\mathcal{K} : Z \times Z \to \mathbb{R}$ be defined as $\mathcal{K}(z, z') = \langle \Phi(z), \Phi(z') \rangle$. The function $\mathcal{K}$ is known as a *kernel* (or a similarity function). To solve the dual problem or computing the SVM classification for a new observation, computing the feature map explicitly is not necessary once the kernel is known. We can deal with high-dimensional (or infinite-dimensional) feature vectors without explicitly computing them. This is commonly referred as the *kernel trick*.

2. The solution depends only on $(z^{(i)}, y^{(i)})$ with $\alpha^{(i)*} > 0$. These observations are therefore known as *support vectors*.

3. For multiclass (say B classes) classification, we can follow a one-versus-all approach: construct an SVM binary classifier for each class that separate the class from the rest of the observations (that is, for $k = 1, ..., B$, solve the optimization problem for the kth class to get $\alpha_k, b^*_{D,k}$). The classification for a new observation $z^{(new)}$ is

$$\arg \max_{k=1,...,B} \left[\left(\sum_{i=1}^{m} y^{(m)} \alpha_k^{(i)*} \langle \Phi(z^{(i)}), \Phi(z^{(new)}) \rangle \right) + b^*_{D,k} \right].$$

4. One advantage of using SVM is that SVM is readily available in different packages/libraries for various common programming languages: Python, R, C++, etc.

For more information on SVMs, see the tutorial paper by Burges (1998) or the monograph by Steinwart and Christmann (2008).

3 Overview of random forests

Random forests are considered as a type of ensemble learning methods, meaning that multiple models are constructed and combined to solve a learning problem. The three key ideas in random forests are classification trees, bootstrap aggregation (also known as *bagging*), and random selection of features.

A classification tree partitions the feature space into pieces and assigns a label to each of the pieces. More concretely, the tree begins with a root node, which is one of the features, and splits the space of the feature into multiple parts. For each part, we select another feature to be a node and perform another split. The newly added feature is selected based on the maximum information gain from the remaining

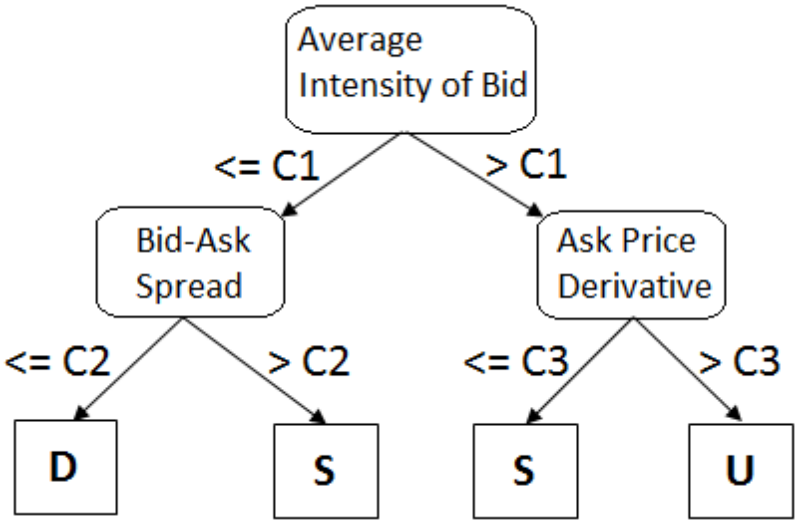

Figure 4: Example of the structure of a simple decision tree

basket of unused features. The maximum information gain of a feature is the maximum reduction in information impurity achievable by splitting the tree based on some criterion of the feature. Information impurity is a measure of how equally distributed are the labels of a certain branch of the tree. For example, a well classified branch containing the same label for all of the data points in the branch would have an information impurity of zero, and a poorly classified branch with the three labels equally distributed would have a large information impurity. The method we chose uses the Gini information impurity measure:

$$i(N) = P(S)P(U)P(D)$$

This process continues recursively until a minimum node size (the number of observations corresponding to the node) or information gain is reached. At that point, the node is considered to be a leaf node, and the most popular label of the data points that reaches this node is assigned as the label of the node.

Classification of a new data point is done by starting at the root node and following the path of the decision tree, and at each node, going to the branch for which the features of the new data point satisfy the branching criterion. Once a leaf node is reached, the new data point is labeled as the label of the node.

One of the main problems with classification trees (or tree-based methods in general) is that the variance is high, resulting in the instability of the models. To overcome this problem, it is natural to make use of bagging to reduce the variance. Bagging is the process of averaging many models, each of which is trained with a bootstrap sample, to yield a smaller variance. A total of S samples are drawn with replacement from the original dataset, each having the same number of data points as the original dataset. For each of these samples, a set of D features are drawn without replacement from the full set of M features ($D = floor(\sqrt{M})$), and a decision tree is trained using these D features on the sample data. After all of the S decision trees are trained, prediction is done by classifying the new data point using each of the decision trees and taking the mode of the classification results.

One attractive feature of random forests is that feature importance can be extracted. While there is no general consensus on how feature importance is defined, feature importance roughly refers to the contributions of features when building the classification trees in the process of random forests. In the Python package *scikit-learn*, feature importance is computed as the total reduction in information impurity weighted by the proportion of samples reaching that node averaged over all trees of the ensemble.

4 Data and Methodology

The data we used comes from the limit order book and the message book of the SPY index. We used thirty minutes worth of data from 2:30pm to 3:00pm on May 10th, 2012 to train our machine learning models.

Basic Set	Description (i = LOB level)
$v_1 = [0, 39]$	price and volume (n levels)
Time-insensitive set	Description (i = LOB level)
$v_2 = [40, 59]$	bid-ask spreads and mid-prices
$v_3 = [60, 95]$	price differences
$v_4 = [96, 99]$	mean prices and volumes
$v_5 = [100, 101]$	accumulated differences
Time-sensitive set	Description (i = LOB level)
$v_6 = [103, 142]$	price and volume derivatives
$v_7 = [143, 148]$	average intensity of each type
$v_8 = [N/A]$	relative intensity indicators
$v_9 = [149, 152]$	accelerations (market and limit order bids and asks)

Table 1: Feature Vector Set - Basic Set, Time-insensitive Set, and Time-sensitive Set for 10 levels of the LOB

To turn the price movement problem into a classification problem, we need to assign class label to each data point. A data point is a snapshot of the limit order book. Similar to Kercheval and Zhang, we use mid-price movement as class labels, that is, upward, downward, and stationary. Mid-price is defined as the average of the best bid and the best ask. An upward movement indicator (U) is assigned to a data point (a LOB snapshot) if the mid-price at Δ unique timestamp trade events later is larger than the mid-price of the current data point. Similarly, a downward indicator (D) and stationary indicator (S) are assigned if the mid-price is lower and equal to the current, respectively.

The feature space was chosen as the feature vector set presented in Kercheval and Zhang, and is shown in Table 1.

For our analysis, we have defined v_1 to v_7, and v_9 as our entire feature space. Due to computational complexity, v_8 was omitted in our results, but it can be easily added with proper computation power.

The limit order book data file contains 10 level of bid/ask prices as well as the corresponding bid/ask volumes. Each data row represents a snapshot of LOB at a time. The message book data file contains records of trade events. The event types in the data file are defined as follows: A=Add Order; D=Order Delete; CA=Order Executed with Price. Type A events are arrivals of limit bid/ask orders; Type CA events are order executions with market price, or can be viewed as arrivals of market orders; Type D events are order cancellations.

We used K-fold cross validation ($K = 10$) to measure the performance. In K-fold cross validation, the training data is split into K buckets. One bucket is marked as test bucket, and the other $K - 1$ buckets are used to train the classifier. This result is repeated K times until each bucket has been chosen as the test bucket.

To validate our model, performance was measured using three common accuracy measures (averaged for the K buckets): Precision, Recall, and F1-Measure defined in the following equations.

$$\text{Precision: } P = \frac{\#(\text{correctly labeled } y)}{\#(y \text{ in the predictions})}$$

$$\text{Recall: } R = \frac{\#(\text{correctly labeled } y)}{\#(y \text{ in the sample})}$$

$$F_1\text{-measure: } F_1 = \frac{2PR}{P+R}.$$

5 Results

The machine learning techniques were used on SPY data. We tested the results with SVM with Linear and Gaussian kernels and with Random Forest. For $\Delta = 30$ of unique timestamp trade events, the distribution of labels (U,D,S) of (1, 1.1, 2.4) was found. The results and comparisons of the various algorithms are described below.

Linear SVM seems to have poor results due to the data being largely linearly inseparable. We instead try SVM with the RBF kernel as alternative separating similarity measure.

The results show that while Upward and Downward Precision accuracy measure is quite high, the Recall accuracy measure is low. What this intuitively means is explained as follows - While we can

Label	Precision	Recall	F1-Measure
Upward (U)	93.8%	20.2%	33.0%
Downward (D)	95.7%	25.6%	40.2%
Stationary (S)	59.2%	98.9%	74.1%

Table 2: SVM Gaussian- Accuracy Measures with entire feature set, $\Delta = 30$

Label	Precision	Recall	F1-Measure
Upward (U)	86.9%	78.3%	82.1%
Downward (D)	87.2%	83.7%	85.4%
Stationary (S)	86.4%	91.6%	88.9%

Table 3: Random Forrest - Accuracy Measures with entire feature set, $\Delta = 30$

predict Upward and Downward correctly most of the time, it is only because we miss a lot of the Upward and Downward trials. Instead we predict Stationary for most of the samples, which is why we see a low Precision metric for Stationary.

Random forest approaches seem to outperform the SVM counterparts and produces quite accurate accuracy measures. In particular, the fact that most of these measures are high shows that the three classes were separable. Some intuition as to why RF has performed better in this scenario can come from the relatively nonlinear nature of the LOB features. While most classification problems that can be separated into linear or polynomial kernels tend to have better performance with SVM, RF is the winner here because the feature space is much more nonlinear.

The aforementioned results had shown accuracy measures upon using the entire feature space. However, it is often unfeasible to be able to run the cross validation model and update various parameters in time if model updates are needed at a certain frequency. Traders may want to find a point that optimally trades off compute time with update frequency. Hence, an economical feature set (a subset of the entire feature space) is desirable for efficiency purposes. An analysis was done to find the relative feature importance, and the following table shows an example of the 50 most important features. Only 50 features are shown for readability purposes. Feature Index can be determined by concatenating and flattening the basic, time insensitive, and time sensitive sets in the full feature vector set from Table 1. For our training dataset, the top ten features with the greatest feature importance are:

1. Feature 30: Volume of the first level of bid
2. Feature 10: Volume of the first level of ask
3. Feature 99: Mean volume of the first ten levels of bid
4. Feature 112: Derivative of the tenth level of ask price
5. Feature 101: Accumulated difference of volumes
6. Feature 132: Derivative of the tenth level of bid price
7. Feature 12: Volume of the third level of ask
8. Feature 97: Mean volume of the first ten levels of ask
9. Feature 11: Volume of the second level of ask
10. Feature 13: Volume of the fourth level of ask

To find the economical set, we simply take the first N features. The following tables show differences in performance between $N = 20$, and $N = 5$.

From Tables 4 and 5, it is clear that picking the best 20 features still gives similar results compared with using the entire feature vector set. However, if we pick too few features, the accuracy drops significantly. Surprisingly enough, it still seems like it $N = 5$ can still be somewhat usable in this scenario.

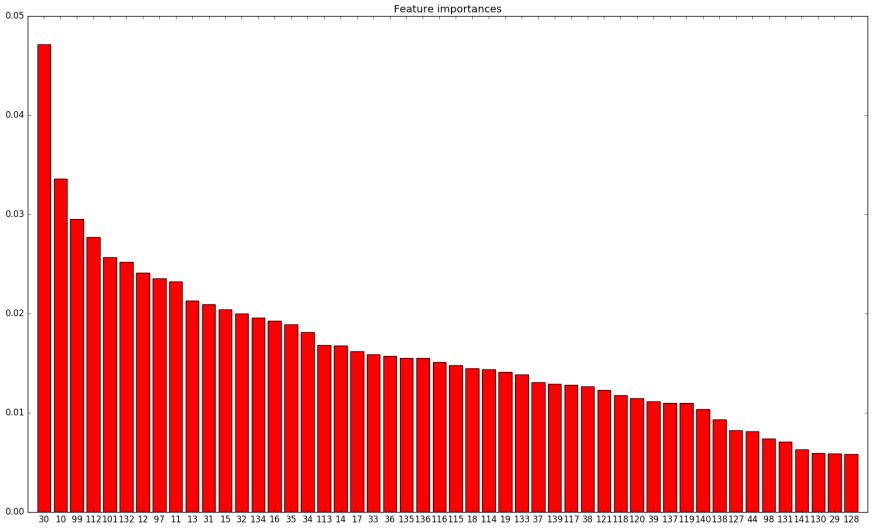

Figure 5: Top 50 Feature Importance vs. Feature Index for SPY 05/10/2012 2:30-3:00pm.

Label	Precision	Recall	F1-Measure
Upward (U)	86.4%	71.3%	77.9%
Downward (D)	83.9%	78.8%	81.2%
Stationary (S)	83.1%	91.6%	87.1%

Table 4: Economical Set Accuracy Measures for $N = 20$, $\Delta = 30$

Label	Precision	Recall	F1-Measure
Upward (U)	81.2%	66.5%	73.1%
Downward (D)	77.4%	76.2%	76.7%
Stationary (S)	81.2%	87.6%	84.3%

Table 5: Economical Set Accuracy Measures for N=5, $\Delta = 30$

Label	Definition
Strong Upward (SU)	$\Delta P_t^{mid} \geq$ 80th percentile of positive mid-price changes
Upward (U)	$\Delta P_t^{mid} > 0$ and $\Delta P_t^{mid} <$ 80th percentile of positive mid-price changes
Stationary (S)	$\Delta P_t^{mid} = 0$
Downward (D)	$\Delta P_t^{mid} < 0$ and $\Delta P_t^{mid} >$ 80th percentile of negative mid-price changes
Strong downward (SD)	$\Delta P_t^{mid} \leq$ 80th percentile of negative mid-price changes

Table 6: Class definitions for the 5-class SVM classifier. The percentiles are computed based on the current training set.

6 Conclusion

We have demonstrated how to use machine learning techniques to classify mid-price movements using Limit Order Book data as features. In particular, we have proposed and demonstrated classification problem that is assumed to solve a stationary prediction problem of predicting mid-price movements at high frequency. Random Forest was most successful in terms of accuracy of mid-price movements compared to its SVM counterparts. This was largely due to the LOB features being not linearly separable. Furthermore, we have shown that an economical set of features can provide similar, almost equivalent results to using the entire feature space.

A potential extension of the 3-class classifier is a 5-class classifier. The five classes in the potential extension are strong upward (SU), upward (U), stationary (S), downward (D), and strong downward (SD). The motivation for extending from three classes to five classes is to capture a fuller picture of the price changes, which enables us to develop a more sophisticated trading strategy compared to the simple instant buy/sell or sell/buy strategy. One possible set of the definitions of the five classes are shown in Table 6. We use the 80th percentile of the positive mid-price changes in the observations in the training set as a threshold to determine whether the upward movement is strong.

References

[1] Burges, Chirstopher J.C. "A Tutorial on Support Vector Machines for Pattern Recognition." Data Mining and Knowledge Discovery 2, 121-167. (1998).

[2] Duba, Hart, and Stork. "Pattern Classification". (2001).

[3] Hastie, Trevor, Tibshirani, Robert, and Friedman, Jerome. "The Elements of Statistical Learning: Data Mining, Inference, and Prediction." Second Edition. (2009).

[4] Kercheval, Alec N., and Zhang, Yuan. "Modeling high-frequency limit order book dynamics with support vector machines." (2014).

[5] Steinwart, Ingo and Christmann, Andreas. *Support Vector Machines*. Springer. (2008).