Applied Mathematics
&
Statistics

520 Report - Fall 2022

Multiple Kernel Learning on the Limit Order Book

Jason Bohne, Jarryd Sculley, Paul Vespe

Contents

1 Introduction **3**

2 Experimental Design **4**

2.1 Classification Framework 4

2.2 Support Vector Machines 4

2.3 Multiple Kernel Learning 5

3 Data Preparation and Feature Generation **6**

3.1 Kernel and Feature Selection 6

4 Numerical Results **8**

4.1 Evaluation Metrics 8

4.2 Single Kernel Performance 8

4.3 Multiple Kernel Performance 8

5 Conclusion **11**

6 Appendix **12**

6.1 Code Access 12

1 Introduction

In recent years, large portions of trading activity is moving to Electronic Communication Networks (ECNs). As a result, large data sets of limit order book data have come to the forefront of high-frequency finance. Limit order books display available units of an asset for sale by price and size. A market participant that is capable of exploiting order book information may be able to build a profitable trading method.

This paper attempts to build an effective prediction model using the information listed in the top of the order book. While most predictions may focus on the magnitude of the response variable, this paper will look to establish a categorical prediction strategy. In other words, instead of computing the price at time T_{t+1}, the employed model attempts to track the direction of the change in price.

This paper investigates the usage of kernel methods to find patterns that can be exploited for profit. The scope of this application is generally limited to single kernel models; this paper attempts to redefine the problem using a multiple kernel learning (MKL) model. The MKL framework will replace a single kernel model selection with a basis set of kernel functions. From this, a gradient descent algorithm determines the optimal linear combinations of kernels that are suitable for modeling the given information.

In our analysis, features are generated off high-frequency TAQ data spanning a one-month horizon period. Due to the size of available data, support vector machines are trained in batches across the dataset, with evaluation occurring on the subsequent block. For evaluation, the rolling accuracy and weighted precision scores are included. Finally, public code for reproducibility is included on Github in the appendix.

2 Experimental Design

2.1 Classification Framework

A market participant that was informed with the expected price movement could build out a profitable trading strategy. The price of an asset can take one of three actions: trend upwards, trend downwards, or experience no trend. Using this methodology, the classification strategy can be formulated using three actions.

$$P_{t+\delta t}^{Bid} > P_t^{Ask} \text{ Buy the currency (long position)}$$
$$P_{t+\delta t}^{Ask} < P_t^{Bid} \text{ Sell the currency (short position)}$$
$$P_{t+\delta t}^{Bid} < P_t^{Ask},\ P_{t+\delta t}^{Ask} > P_t^{Bid} \text{ Take no position}$$

A potential trading strategy would only utilize predictions from price movements that cross the bid-ask spread.

2.2 Support Vector Machines

Support Vector Machines (SVM) are a common supervised learning algorithm used for classification problems. SVMs are often applied in linear modeling, as well as non-linear, where the inputs are mapped to a higher dimensional space using the kernel trick. Given a set of data, SVMs attempt to fit a decision boundary that minimizes the classification error. The boundary may be a simple line in 2-dimensional spaces but is a hyperplane in higher dimensions. The ability of the kernel trick allows the model to understand the distances between projected data points without having to operate directly in the high-order space.

The optimal selection criterion for the decision boundary relies on a hinge loss function, which quantifies the classification error.

$$\max(0, 1 - y_i(w^T x - b))$$

Including the hinge loss in the optimization, the dual objective can be formulated as

$$\max_{\alpha} -\frac{1}{2}\sum_{(i,j)} \alpha_i \alpha_j y_i y_j K(x_i, x_j) + \sum_i \alpha_i$$

where there is a trade-off between increasing the margin size and ensuring that the features lie on the appropriate side of the margin. These ideas will be extended in the MKL framework, but will still remain relevant for comparative out of sample performance testing.

2.3 Multiple Kernel Learning

Multiple Kernel Learning is an extension of traditional support vector machines that allows the decision function to be composed as a linear combination of distinct component functions, $f_m(x)$, first proposed by [Rakotomamonjy et al., 2008]. Here each component function corresponds to a different Reproducing Kernel Hilbert Space $\mathcal{H}_m$ with kernel K_m. Under this framework the primal objective can be written as

$$\min_{f_m} \frac{1}{2}(\sum_m \frac{1}{d_m}||f_m||^2_{\mathcal{H}})^2 + C\sum_i \varepsilon_i$$

Constraints on this objective impose the coefficient vector d_m is non-negative and unit norm.

$$y_i \sum_m f_m(x_i) + y_i b \geq 1 - \varepsilon_i,\ \varepsilon_i \geq 0,\ \sum_m d_m = 1,\ d_m \geq 0$$

As with traditional support vector machines, one specifies the kernels that appear within the dual objective, which are polynomial or Gaussian functional forms. MKL algorithms then solve for the optimal coefficient vector prescribing the non-negative linear combination of the candidate kernels. Accounting for the multiple kernels; the dual objective is

$$J(d) = \max_{\alpha} -\frac{1}{2}\sum_{(i,j)} \alpha_i^* \alpha_j^* y_i y_j \sum_m d_m K_m(x_i, x_j) + \sum_i \alpha_i^*$$

As this objective is convex and differentiable, a gradient descent algorithm is suggested by [Rakotomamonjy et al., 2008]. For a fixed coefficient weight vector d_i the differential of the dual $\frac{\partial J}{\partial d}$ can be solved in closed form to be

$$\frac{\partial J}{\partial d_m} = -\frac{1}{2}\sum_{(i,j)} \alpha_i^* \alpha_j^* y_i y_j K_m(x_i x_j)$$

From here estimates for the coefficient vector d_m are iteratively updated by stepping in the direction of descent which is altered to satisfy the non-negativity constraints. Here the step size γ_{opt} is determined by choosing the optimal γ_i in terms of the decrease in duality gap from a set of candidate stepsizes. This iteration is repeated until the duality gap between the primal and

dual objectives is below a certain threshold.

Algorithm 1 Gradient Descent Algorithm (SimpleMKL)

set $d_m = \frac{1}{M}$ for $m = 1 \dots M$
while $\mathrm{P}_{obj} - D_{obj} > \epsilon$ **do**
  Evaluate $J(d)$ using SVM with $K = \sum_m d_m K_m$
  Compute $\frac{\partial J}{\partial d_m}$ and descent direction D
  set $\mu = \arg\max_m d_m, J^* = 0, d^* = 0, D^* = 0$
  while $J^* < J(d)$ **do**
    $d = d^*$, $D = D^*$
    $\nu = \arg\min \frac{-d_m}{D_m}$ $\gamma_{\max} = \frac{-d_\nu}{D_\nu}$
    $d^* = d + \gamma_{\max} D$, $D^*_\mu = D_\mu - D_\nu$ $D^*_\nu = 0$
    Compute J^* using SVM with $K = \sum_m d^*_m K_m$
  end while
  Line search candidate γ_i for $d \leftarrow d + \gamma_{opt} D$
end while

A quality test on the accuracy of the multiple kernel gradient descent algorithm would be to construct the set of kernel basis as an identical kernel and perform the optimization. Ideally the optimal coefficient vector would then be equal allocation of $d_m = \frac{1}{m}$ where m is the size of the kernel basis set. This quality test was performed and passed with results available within our Github.

3 Data Preparation and Feature Generation

3.1 Kernel and Feature Selection

Although the main goal of this paper was to implement an MKL approach in the analysis of our data, it was also of critical importance to engineer a feature set which would give us the best possible predictability. In addition to some of the metrics used by [Fletcher et al., 2010], we also decided to include some other commonly used metrics in limit-order-book analysis which include:

1. *Spread* (S)

The spread is defined as the difference between the current bid and ask price. As we only have top of the book data, this would be the best-bid price and best-ask price in our case. Despite being a simple metric, spread can be a good proxy for market liquidity.

2. *Spread-Change* (SC)

For a given time t, we defined the spread change as the difference in spread between time t and

time *t-1*. This serves as an indicator for how market liquidity is changing. If spread is increasing, the market is considered less-liquid, and if decreasing, is considered more-liquid.

3. *Weighted Spread* (WS)

At time t, we consider the interval $(t-\delta, t)$. We then multiply each bid-price by its associated volume and divide by the total bid-volume in the associated interval to compute weighted bid price (WBP). We calculate the weighted ask-price (WAP) in a similar way. The weighted spread is then defined as

$$WS = WAP - WBP$$

4. *WS Anomaly* (WSA)

After computing the weighted spread for all the times in a given interval, we then calculate a 95% percentile for this interval. If the weighted spread at time t lies greater than this interval, we consider it an anomaly and denote the associated value as 1. Else, we classify it as 0. This can be a useful proxy for the volatility of weighted spread.

In addition to these metrics, we also used some from the paper by [Fletcher et al., 2010], namely: best-bid/ask volume, and change in best-bid/ask volume from the previous observation. Thus, we denote our feature set as the following:

$$F = \Big(V_t, V_t - V_{t-1}, S_t, SC_t, WS_t, WSA_t \Big)$$

Previously stated, the MKL framework will accept a range of different kernels to build the model. As a result, it is necessary to define a set of functions that can solve the selection objective. The set of possible kernel functions is defined as

$$K = \left[exp\left(\frac{-||x-x^{'}||^2}{\sigma_1^2} \right), ..., exp\left(\frac{-||x-x^{'}||^2}{\sigma_n^2} \right), \langle x, x^{'} \rangle \right]$$

where $\sigma_{1...n}$ is the bandwidth parameter of the radial kernel.

The response variable can be labeled in motivation from the previously discussed classification outcomes. The following three labels are created:

$$\text{Label 1: If } P_{t+\delta t}^{Bid} > P_t^{Ask}$$

Label -1: If $P^{Ask}_{t+\delta t} < P^{Bid}_{t}$

Label 0: If $P^{Bid}_{t+\delta t} < P^{Ask}_{t}$

4 Numerical Results

4.1 Evaluation Metrics

For computational feasibility when training, the aggregated features are batched into blocks of 300 observations with evaluation occurring on the subsequent block. This allows us to train and evaluate support vector machines independently across each pair of blocks. As this is a multi-label classification problem evaluation metrics of accuracy and weighted precision will be utilized to determine performance.

Accuracy is defined as:

$$\frac{TP + TN}{TP + TN + FP + FN}$$

Weighted Precision is defined as :

$$\frac{TP}{TP + FP}$$

4.2 Single Kernel Performance

For the single kernel SVM, both the trivial linear kernel and the Gaussian kernel are evaluated. Rolling accuracy and weighted precision for the SVMs across the data batches are included below. Almost universally, the Gaussian kernel outperforms the linear kernel, indicating the presence of nonlinear effects of the features on the short-term trends.

4.3 Multiple Kernel Performance

Following the analysis of [Fletcher et al., 2010] the basis set of functions for the multiple kernel problem consists of 3 and 5 Gaussian kernels with equally spaced bandwidth parameters. The accuracy and precision for a selected date peak after the market opens with a quick decay in performance shortly after. One interpretation for this is there is a short-term trend and stability shortly after the market settles post-open which persists for a certain time period. Overall, the Multiple Gaussian Kernel tends to outperform throughout the day in terms of accuracy, however, has similar performance in precision. One reason for such could be due to an imbalance in classes with greater accuracy in a specific subset.

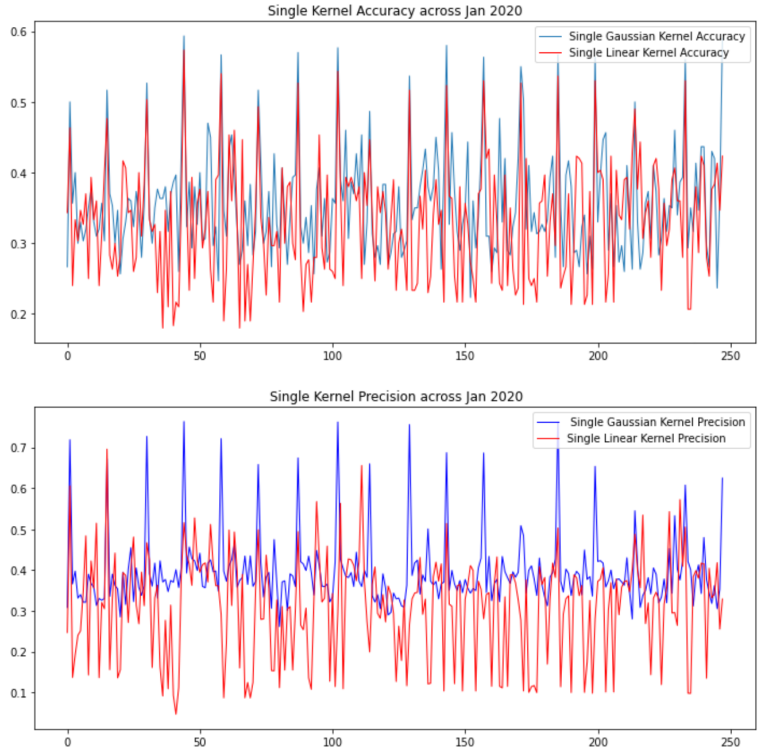

Figure 1: Accuracy and Precision of Single Gaussian vs. Linear Kernel SVM

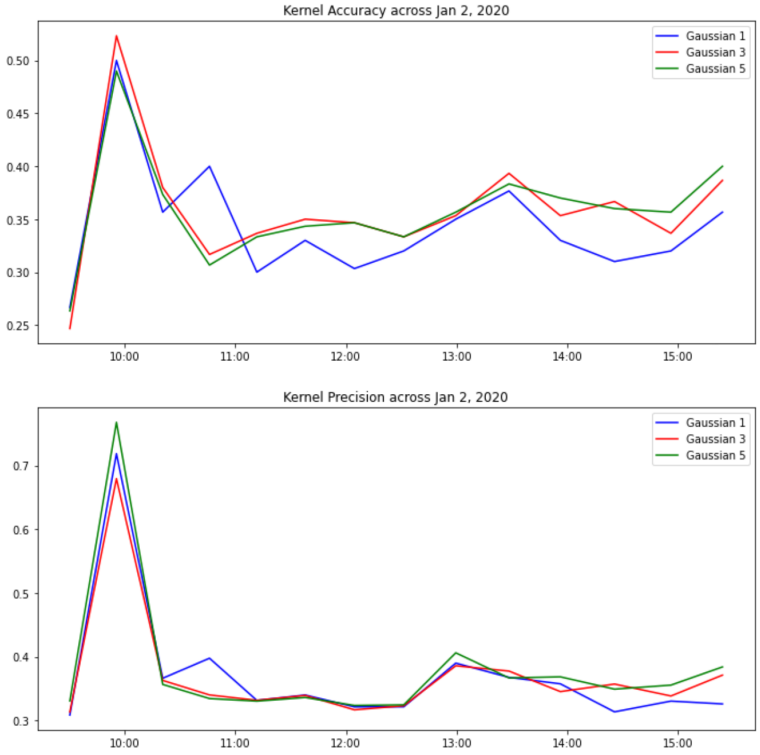

Figure 2: Accuracy and Precision of Single and Multiple Gaussian Kernel SVM

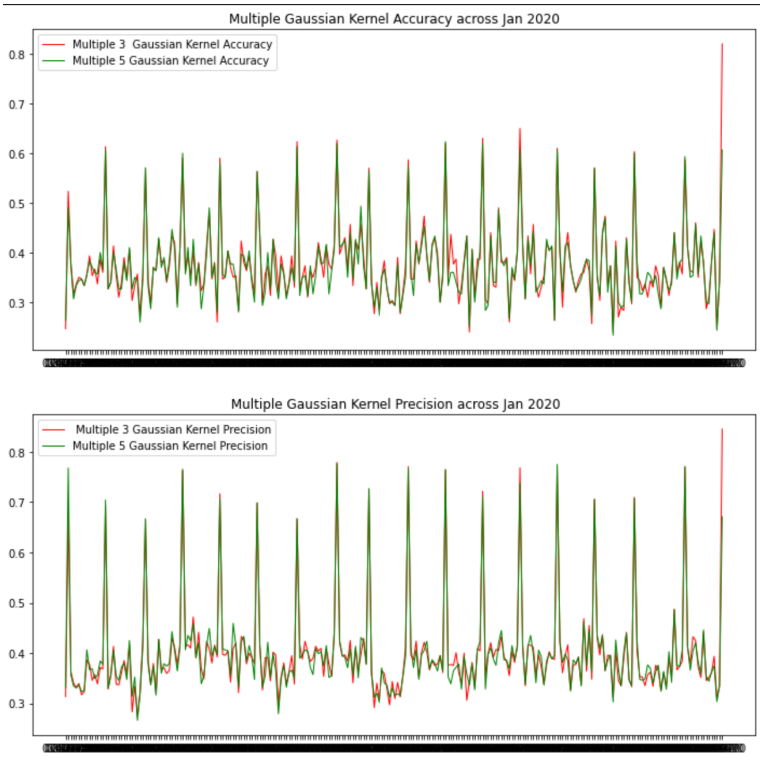

Figure 3: Accuracy and Precision of Multiple Gaussian Kernel SVM

Finally, there are tables specifying the average accuracy and weighted precision attained throughout the evaluation. Overall a single Gaussian kernel seems to outperform the linear kernel. Extending to the multi-kernel case, a significant difference does not appear to be present. Further investigation is required for the distribution of estimates, which can begin by focusing on the estimates' variance.

Kernel	**Accuracy**
Linear	0.33
Single Gaussian	0.36
Multiple Gaussian (3)	0.38
Multiple Gaussian (5)	0.38

Table 1: Sample Mean of SVM Accuracy across Batches

Kernel	**Accuracy**
Linear	0.31
Single Gaussian	0.40
Multiple Gaussian (3)	0.40
Multiple Gaussian (5)	0.40

Table 2: Sample Mean of SVM Precision across Batches

5 Conclusion

This paper attempts to design a profitable price direction model off on limit order book data. Although many approaches focus on the magnitude of price fluctuations, the strategy laid out in this paper applies a classification scheme to label the possible price movements. Single kernel learning methods are a popular approach for classification, but face potential estimation risk in kernel selection. A multiple-kernel framework can avoid estimation risk by establishing a rigorous selection objective given a set of possible kernels. The MKL framework will find the optimal set of kernels and their respective weights that minimize the objective.

In any machine learning task, the process of selecting relevant and useful features is often a problem in and of itself. Although choice of algorithm plays a large role in overall performance and predictability, the quality of features used is equally important. In our case, simple metrics were derived from top of the order book. It would be interesting to see whether additional order book depth increases performance in terms of accuracy or precision.

From our numerical results it is clear that in this application, MKL and the single Gaussian kernel methods perform similarly. That being said, both significantly outperform the naive linear kernel. One could draw the conclusion MKL does not offer an improvement and in fact is a net negative due to the overhead of the additional optimization problem. However future investigation is required to determine the equivalence in performance across a larger set of assets and markets.

6 Appendix

6.1 Code Access

The full repository of all scripts and data utilized in this project can be accessed from our **GitHub**. Note that cleaned and aggregated features can be found **here**

References

[Fletcher et al., 2010] Fletcher, T., Hussain, Z., and Shawe-Taylor, J. (2010). Multiple kernel learning on the limit order book. *Journal of Machine Learning Research - Proceedings Track*, 11:167–174.

[Rakotomamonjy et al., 2008] Rakotomamonjy, A., Bach, F., Canu, S., and Grandvalet, Y. (2008). Simplemkl. *Journal of Machine Learning Research*, 9.