Market Microstructure in the Big-data Era: Improving High-frequency Price Prediction via Machine Learning

Agostino Capponi\textsuperscript{1}, Shihao Yu\textsuperscript{1}

\textsuperscript{1}Columbia University, IEOR

September 22, 2023

Markov Decision Process and Reinforcement Learning Workshop
Cambridge University
Outline

Introduction

Data

Methodology

Results

Conclusion
Outline

Introduction

Data

Methodology

Results

Conclusion
Price discovery

- Price discovery is key in microstructure and financial markets

- Fundamental question: where does information originate?
  - From which trading venues (NYSE vs Nasdaq)?
  - Via what order types (trades vs quotes, top-of-book vs depth-of-book)?
  - Via which asset classes (spot vs derivatives)?

- The existing empirical microstructure literature typically uses vector autoregressive (VAR) or vector error correction (VEC) models. They share the following features:
  - Structural functional forms with a relatively small set of features
  - In-sample attribution of the information shares
Challenges in the machine age

▶ Trading in the machine age (e.g., in the US equities market)
  ▶ Extremely fast: algorithmic and high-frequency trading; 20% of trades arrive in < 1ms clusters (Menkveld, 2018)
  ▶ A highly fragmented market: 16 public exchanges, internalization, dark pools
  ▶ Voluminous trading data: level-3 order book messages

▶ Challenges:
  ▶ A much-expanded feature set (based on full order books from many markets)
  ▶ Complex non-linear effects and feature interactions (from algorithmic trading strategies such as dice-and-slice, pinging, layering, cross-market/cross-asset arbitrages...)

▶ Need for machine learning (ML) models
Paper in a nutshell

▶ Propose machine learning (ML) models suitable for empirical market microstructure with big data challenges

▶ Apply it to price discovery analysis
  ▶ Which exchange contributes the most to price discovery?
  ▶ Which part of the data feed contributes the most to price discovery?

▶ Key takeaways:
  ▶ ML models designed for processing sequential data, long short-term memory (LSTM), and Transformers, perform much better in predicting short-term midquote returns

  ▶ Nasdaq contributes the most to price discovery. Dropping its data feeds leads to about 6% drop in out-of-sample $R^2$

  ▶ Data feeds beyond the top-of-book are informative. Dropping them leads to about 13% drop in out-of-sample $R^2$
Literature

▲ Empirical market microstructure on price discovery
  ▲ Models: VAR (Hasbrouck, 1991); “information share” via VECM (Hasbrouck, 1995); “component share” (Harris, McInish, and Wood, 2002); “information leadership share” (Putniņš, 2013); VAR + VECM (Hagströmer and Menkveld, 2023);
  ▲ Applications: spot vs futures (Hasbrouck, 2003); cross-border listings (Eun and Sabherwal, 2003); dark vs lit trading (Hendershott and Jones, 2005)

▲ Financial machine learning
  ▲ Cross-section asset pricing (Gu, Kelly, and Xiu, 2020); Mutual funds selection (Li and Rossi, 2020); Robot-advising (Rossi and Utkus, 2020); Corporate bond return prediction (Bali et al., 2020)

▲ Our contribution: apply ML models suitable for empirical microstructure
Outline

Introduction

Data

Methodology

Results

Conclusion
Data

‣ “Direct feeds” from public exchanges
  ▶ Level 3 order-book messages: all add (new limit orders), cancel/Modification of existing orders, and trade messages
  ▶ Timestamped to microsecond precision

‣ 30 constituent stocks of the Dow Jones Index (DJI). 54 trading days interspersed from the year of 2017 to 2021.

‣ For each exchange, we build the entire order book based on the direct feed messages
Limit order book (LOB) market

- Most liquid markets use limit order books (LOBs) for trading
- A limit order book is essentially a collection of unexecuted quotes
  - Each quote specifies the price and quantity the trader is willing to trade
  - New quotes can be continuously added and existing quotes can be canceled, modified, or executed against incoming marketable orders

<table>
<thead>
<tr>
<th>ASKS</th>
<th>Shares</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>249</td>
<td>172.36</td>
</tr>
<tr>
<td></td>
<td>560</td>
<td>172.35</td>
</tr>
<tr>
<td></td>
<td>349</td>
<td>172.34</td>
</tr>
<tr>
<td></td>
<td>525</td>
<td>172.33</td>
</tr>
<tr>
<td></td>
<td>125</td>
<td>172.32</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BIDS</th>
<th>Shares</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100</td>
<td>172.31</td>
</tr>
<tr>
<td></td>
<td>323</td>
<td>172.30</td>
</tr>
<tr>
<td></td>
<td>449</td>
<td>172.29</td>
</tr>
<tr>
<td></td>
<td>364</td>
<td>172.28</td>
</tr>
<tr>
<td></td>
<td>249</td>
<td>172.27</td>
</tr>
</tbody>
</table>
LOB data types

- Level-I: the best bid/ask prices and volumes,
- Level-II: price and aggregated volume across a certain number of price levels
- Level-III: non-aggregated orders placed by market participants

Figure 1: LOB data types. Source: Wu et al. (2022)
Direct feeds

▶ “Direct feeds” examples

(a) Add message

(b) Cancel/modification message

(c) Trade message

Figure 2: Direct feeds
Outline

Introduction

Data

Methodology

Results

Conclusion
LOB actions

- LOB is constantly changing due to addition, modification, and execution of orders
  - New quotes can be continuously added, and existing quotes can be canceled, modified, or executed against incoming marketable orders
  - At different price levels

Figure 3: LOB actions. Source: Wu et al. (2022)
Selected Features

▶ **LOB events** and their lagged values (50 lags), *from each exchange*
  ▶ **Trade-BBO-Changing**: Executions moving BBO
  ▶ **Trade-NonBBO-Changing**: Execution not moving BBO
  ▶ **Add-BBO-Improving**: Add orders improving BBO
  ▶ **Cancel-BBO-Worsening**: Cancel orders worsening BBO
  ▶ **Add-at-BBO**: Add orders adding depth at the current BBO
  ▶ **Cancel-at-BBO**: Cancel orders removing depth at the current BBO
  ▶ **Add-\(<=5\text{lvlBBO}$$**: Add orders adding depth \(\leq 5\) levels from BBO
  ▶ **Cancel-\(<=5\text{lvlBBO}$$**: Cancel orders removing depth \(\leq 5\) levels from BBO
  ▶ **Add-\(>5\text{lvl-BBO}$$**: Add orders adding depth \(>5\) levels from BBO
  ▶ **Cancel-\(>5\text{lvl-BBO}$$**: Cancel orders removing depth \(>5\) levels from BBO

▶ **Midquote return**, and their lagged values (50 lags), *from each exchange*

*Example*: suppose a new limit order adding 300 shares at the best bid, then the variable "Add-at-BBO" takes the value of 300, and all other variables are 0
Target and performance evaluation

▶ Target
  ▶ Short-term midquote return (over the next five events)
  ▶ Clock runs in event time

▶ Performance evaluation
  ▶ Out-of-sample $R^2$:
    \[
    R^2(Y, \hat{Y}) = 1 - \frac{\sum_i \left( Y_i - \hat{Y}_i \right)^2}{\sum_i \left( Y_i - \frac{1}{n} \sum_i Y_i \right)^2}
    \]
  
▶ $R^2 > 0$: the model outperforms the out-of-sample mean
Machine learning models

- Simple linear model (OLS)
- Linear models with penalties
  - Elastic net penalties (Elastic Net)
- Tree-based models
  - Random forests (RF)
  - Gradient boosted regression trees (GBRT)
- Artificial neural networks
  - Feedforward or multiplayer perceptron (MLP)
  - Long short-term memory (LSTM)
  - Transformer
Linear model with penalties

- Too many features might lead to overfitting
- One solution is to add penalties to the loss function
- Elastic net penalties: penalize + shrink

\[
\min_w \frac{1}{2n_{\text{samples}}} \| \begin{bmatrix*} \text{features} & \text{Lasso} & \text{Ridge} \end{bmatrix} X w - y \|^2_2 + \alpha \rho \| w \|_1 + \frac{\alpha (1 - \rho)}{2} \| w \|_2^2
\]  

- Solves overfitting but is still linear.
Random forests and boosted regression trees

- Regression tree

**Figure 4:** Tree example. Source: Gu, Kelly, and Xiu (2020)

- Tree splitting captures non-linearities and flexible interactions
- Both random forests and boosted regression trees are ensemble methods
- Combine base estimators to improve generalizability/robustness over a single estimator.
Feedforward networks

- Its non-linear activation function captures non-linearities and flexible interactions
- However, it is not designed for processing temporal sequence
Long short-term memory (LSTM)

- Recurrent neural network (RNN) models are designed to process sequential data like time series.

- Long short-term memory (LSTM) is a gated RNN model that addresses the vanishing gradient problem.

- Uses a series of gate functions to control information flow.

- Captures long-term temporal dependence.
Transformer

- Uses multi-head attention mechanism
- More attention, i.e., weights, given to more important temporal information
- The whole sequence is attended and no loss in temporal information
Training, validation, and testing sample split

- We split each trading day into 13 half-an-hour intervals

- Training (system parameters fitting), validation (hyper/tuning parameters fitting), and testing based on inter-day rolling windows. For example,
  - 09:30 - 10:00 today as training; 09:30 - 10:00 tomorrow as validation; 09:30 - 10:00 the day after as testing

- We consider the following hyperparameters:
  - Elastic-net: $\rho = 0.5; \alpha = (0.1, 0.01, 0.001, 0.0001)$
  - RF: Depth = (2, 4, 6); #Trees = 300; #Features in each split = (3, 5, 10)
  - BRT: Depth = (1, 2); #Trees = (100, 1000); Learning rate = (0.01, 0.1)
  - MLP: units = ((32, 16), (32))
  - LSTM: LSTM units = ((32, 16), (32)); MLP units = ((32, 16), (32))
  - Transformer: # Attention head = (2, 4); Key dimension = (16, 32)
Outline

Introduction

Data

Methodology

Results

Conclusion
Prediction results

- LSTM and Transformers consistently outperform other ML models
- They capture long-term temporal dependence in the feature time series which can result from algorithmic trading strategies
  - E.g., informed traders slice and dice their orders to minimize price impact

<table>
<thead>
<tr>
<th>Model</th>
<th>MSE-Val</th>
<th>MSE-Test</th>
<th>$R^2_{OOS}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLS</td>
<td>1.0612</td>
<td>1.0657</td>
<td>-0.0657</td>
</tr>
<tr>
<td>Elastic Net</td>
<td>0.9784</td>
<td>0.9815</td>
<td>0.0185</td>
</tr>
<tr>
<td>RF</td>
<td>0.9974</td>
<td>0.9975</td>
<td>0.0024</td>
</tr>
<tr>
<td>GBRT</td>
<td>0.9906</td>
<td>0.9914</td>
<td>0.0086</td>
</tr>
<tr>
<td>MLP</td>
<td>0.9958</td>
<td>0.9959</td>
<td>0.0041</td>
</tr>
<tr>
<td>LSTM</td>
<td>0.8879</td>
<td>0.8961</td>
<td><strong>0.1039</strong></td>
</tr>
<tr>
<td>Transformers</td>
<td>0.8426</td>
<td>0.8509</td>
<td><strong>0.1491</strong></td>
</tr>
</tbody>
</table>
Permutation importance

▶ To assess the importance of a feature or several features, permutate (randomly shuffle the ordering) them in the testing set

▶ Then compare the change in out-of-sample $R^2$

▶ Different from in-sample feature importance

▶ Agnostic to model choice

▶ We use the best-performing model, **Transformers**, for the permutation importance calculations
Permutation importance (exchange)

- Which exchange contributes the most to price discovery?

- Look at the $R^2$ drops when an exchange’s data feed is permutated.

- Nasdaq’s data feed is relatively most important. But the drop in $R^2$ is mild in absolute magnitude.

Table 2: $R^2$ of permuted testing samples. The first line shows the $R^2$ of the original sample. The second lines and so on report the $R^2$ changes when an exchange’s data feed is permutated.

<table>
<thead>
<tr>
<th>Feature Type</th>
<th>Feature</th>
<th>MSE</th>
<th>$R^2_{OOS}$</th>
<th>Drop in $R^2_{OOS}$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>All</td>
<td>0.8492</td>
<td>0.1508</td>
<td>0.0</td>
</tr>
<tr>
<td>Exchange</td>
<td>ARCA</td>
<td>0.8509</td>
<td>0.1491</td>
<td>1.14</td>
</tr>
<tr>
<td></td>
<td>BATS</td>
<td>0.8516</td>
<td>0.1484</td>
<td>1.61</td>
</tr>
<tr>
<td></td>
<td>EDGX</td>
<td>0.8512</td>
<td>0.1488</td>
<td>1.33</td>
</tr>
<tr>
<td></td>
<td>INET</td>
<td>0.8583</td>
<td>0.1417</td>
<td>6.01</td>
</tr>
<tr>
<td></td>
<td>NYSE</td>
<td>0.8511</td>
<td>0.1489</td>
<td>1.26</td>
</tr>
</tbody>
</table>
Permutation importance (LOB levels)

- Which part of the limit order book (e.g., beyond the best five levels, or within the best five levels) contributes the most to price discovery?

- Look at the $R^2$ drop when a different part of the data feeds is permutated

- Data feeds beyond the five best levels have limited information; within five levels much more important

Table 3: $R^2$ of permutated testing samples. The first line shows the $R^2$ of the original sample. The second lines and so on report the $R^2$ changes when part of the data feed is permutated.

<table>
<thead>
<tr>
<th>Feature Type</th>
<th>Feature</th>
<th>MSE</th>
<th>$R^2_{OOS}$</th>
<th>Drop in $R^2_{OOS}$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>All</td>
<td>0.8492</td>
<td>0.1508</td>
<td>0.0</td>
</tr>
<tr>
<td>LOB Level</td>
<td>Beyond five best levels</td>
<td>0.8568</td>
<td>0.1432</td>
<td>5.03</td>
</tr>
<tr>
<td></td>
<td>Beyond top-of-book</td>
<td>0.8693</td>
<td>0.1307</td>
<td><strong>13.36</strong></td>
</tr>
</tbody>
</table>
Outline

Introduction

Data

Methodology

Results

Conclusion
Conclusion

- Machine learning (ML) models, such as long short-term memory (LSTM) and Transformers, designed for processing sequential data have improved prediction performance for LOB midquote returns than other ML models.

- Capture long-term temporal dependence in the feature time series which is needed for analyzing trade in the machine age.

- In terms of price discovery analysis:
  - Nasdaq’s trading contains most information relative to other exchanges. Dropping its data feeds leads to about 6% drop in out-of-sample $R^2$.
  - Data feeds beyond the top-of-book are informative. Dropping them leads to about 13% drop in out-of-sample $R^2$.

- Future extensions:
  - Include other potential features such as order queuing information.
  - More direct evidence of the importance of long-term dependence.