Price Trend Prediction Using Deep Learning

Zhiwei Ma

June 8, 2018

Abstract

I summary previous research in applying machine learning algorithms to predict price movement using high frequency limit order data. I also implement a Convolutional Neural Network and deep Recurrent Neural Networks to predict the next price movement. Results show that deep learning models have predictability of price movement using previous limit order data.

1 Introduction

Predicting the price movement in short term can be useful in many aspects. For instance, market makers can use the prediction to avoid adverse price selection and keep providing liquidity to the market (Chang, 2018). In addition, it gives suggestions to the investors of high frequency trading, since they rely on investment under short term horizon.

Given the high frequency limit order book (LOB), the price movements are reflected in the mid-price. In recent years, some research has been published on the predictability of mid-price movements using high frequency LOB. Kercheval and Zhang (2015) trained a Support Vector Machine (SVM) based on a set of handcrafted features, such as spread, derivative and intensity. However, such handcrafted features may lose some important information in the sequential data, while contain some useless and even noisy one. Therefore, we introduce a well known deep learning technique called Convolutional Neural Network (CNN). One key advantage of using a CNN is that it automatically identifies features which are useful for recognition and thus avoid handcrafted features. Some successful work can be found in Tsantekidis et al. (2017b) and Doering et al. (2017).

Beside CNN, we also implement the Recurrent neural network (RNN) to the high frequency limit order data. RNN has very successful applications in aspects like language modeling, translation as well as time serious prediction. An extension of RNN is the Long Short-Term Memory (LSTM), which enables an RNN to correlate temporally distant events by solving the problem of vanishing gradients. We apply both basic RNN and LSTM to predict the price movement. Previous research using RNN in LOB includes Tsantekidis et al. (2017a), Dixon (2018) and MMkinen et al. (2018).

The remainder of this report is outlined as follows. Section 2 introduces both input (real-time order book features) and output data (movement direction labels). Then, Section 3 presents the network models used in this report and their implementation. Section 4 provides the empirical results and finally, Section 5 concludes this work.

2 High Frequency Limit Order Data

The input data consists of the best order for each side of the LOB (bid and ask). Each order is described by 2 values, the price and corresponding volume. Thus, we have four values for each timestamps. For market, we have two assets: a and b. For each asset, we collect data of five consecutive days. The LOB is updated four to five times per second.

2.1 Mid-Price

We want to predict the direction towards which the price will change. In this work, the term price is used to refer to mid-price, which is defined by the mean of best bid and ask price at time t:

$$p_t = \frac{p_a(t) + p_b(t)}{2}, \tag{1}$$

here $p_a(t)$ and $p_b(t)$ represent the best bid price and ask price at time t. More specifically, I want to predict the mid-price direction for the next event (the next timestamps). Predicting the next mid-price movement is suited for markets which often experience a consistent surge of activity at particular times when participants enter into and close out their positions, like futures markets. In addition, since at different time within one day, book updates at different frequencies, predicting in event time will be far more useful and robust.

The directions of mid-price movement are classified into three categories: {*Downward*, *Stationary*, *Upward*}, which are determined simply by the sign of

$$\Delta_t = p_{t+1} - p_t. \tag{2}$$

In some past literature (e.g. Tsantekidis et al. (2017b)), the authors set a threshold α for the return rate to determine the direction: only when the absolute value of return accesses α, will the price change be viewed as a move. However, for our data, mid-price hardly changes and almost all price changes stay in $\{\pm 0.5, \pm 1\}$. Therefore, it is not necessary to hold such thresholding in our work.

2.2 Subsample

In a single day, we have data in there time period: (*I*) 21:00:00-23:30:00, (*II*) 09:00:00-11:30:00, (*III*) 13:30:00-15:00:00 (except for day 4, there is only period (*I*)). I find that in the first several minutes of period (*I*) and (*II*), the price changes drastically compared with rest of the day, which may indicate a different distribution. It is necessary to train a separate model using data in these periods with rapid fluctuation. However, since we don't have enough data (only five days) to train the separate model, I have to exclude data at the first 20 minutes of period (I) and the first 5 minutes of period (II), and train and test our model for the rest of time.

The five day data are separated into three sets: training set, validation set and test set. Training set contains the data from first three days; validation set contains data from the fourth day and period (I) of the fifth day; test set contains data from period (II) and (III) of the fifth day. I label the direction for each timestamp and the results are shown in Table 1.

Table 1: The frequencies of price movement.

Asset	Set	*Downward*	*Stationary*	*Upward*	Total
a	Train	3473	211861	3423	218757
	Valid	731	60948	723	62402
	Test	926	50924	952	52802
b	Train	3448	188708	3374	195530
	Valid	582	43260	600	44442
	Test	809	41277	869	42955

As shown in Table 1, there is seriously imbalanced problem, most of observations belong to *Stationary*. To solve the imbalanced problem, observations (sequences of input variables) labeled by the minority class are oversampled with replacement and the majority class observations are undersampled without replacement. Notice that we only subsample the training set and leave the validation and test set untouched. After subsample, train set remains 150,000 observations for each asset, among which the ratio of *Downward*, *Stationary*, *Upward* is 1:3:1.

2.3 Normalization

We apply *MinMaxScaler* to standardize each variables (prices and volumes) to range $[0, 1]$. Also, since drastic distribution shifts might occurs for different days, the standardization of current period's values uses the min and max of the previous period.

3 Neural Network Models

3.1 Convolutional Neural Network

To forecast the mid-price movement by CNN, the N most recent limit orders are fed as input to the network. Therefor, the input matrix for time t is $\mathbf{X_t} = [\mathbf{x_{t-N+1}}, \mathbf{x_{t-N+2}}, \ldots, \mathbf{x_t}]^{\mathbf{T}} \in$

$\mathbb{R}^{\mathbf{N}\times\mathbf{4}}$, where $\boldsymbol{x}_i$ is a 4-dimensional vector describing the best orders on both side. In practice, I choose $N = 100$.

The architecture of the proposed CNN model consists of the following layers:

(1) 2D Convolution with 8 filters of size (4, 4)

(2) 1D Convolution with 8 filters of size (4,)

(3) Max pooling with size (2,)

(4) 1D convolution with 16 filers of size (4,)

(5) Max pooling with size (2,)

(6) Fully connected layer with 32 neurons

(7) Fully connected layer with 3 neurons

ReLU is used as activation function for both the convolutional layers and the first fully connected layer, and softmax is used for the output layer of the network. I use batch size of 50 to train the model. In the meanwhile, I apply batch normalization after each convolutional layer to accelerate the training process. To deal with overfitting problem, I use dropout of 0.5 after the first fully connected layer. I minimize the cross-entropy loss and apply Adam for optimization.

3.2 Recurrent Neural Network and Long Short-Term Memory

Recurrent Neural Network (RNN) are neural networks in which the connection between neurons allow directly cyclical connections. The basic RNN formula is:

$$h_{t+1} = tanh(W^h[h_t, x_t] + \beta), \tag{3}$$

where $[x, y]$ denotes concatenation.

An extension of RNN is the LSTM, which enables an RNN to correlate temporally distant events by solving the problem of vanishing gradients. We apply both basic RNN and LSTM to predict the price movement to see if the 'long memory' improves. For continence, I will call these two models as RNN and LSTM separately.

Similar to CNN, the input of RNN and LSTM is a matrix of $\mathbb{R}^{100\times4}$, here 100 is the length of most recent events. The last output h_T is then fitted into a fully connected layer of 3 neurons and softmax. For both RNN and LSTM, we set the number of layers to be 2 (deep RNN) and number of hidden units per layer to be 32. To reduce overfitting, we apply the dropout operator proposed by Zaremba et al. (2014) with probability of 0.5, which only operates dropout to the non-recurrent connections. For the training, we use batch size of 50 and also apply Adam for optimization.

3.3 Implementation

For each model and each asset, I run total epochs of 50 due to time limitation. After each epoch, I calculate the loss in validation set and always save the model with the minimal valid loss. For each epoch, if the valid loss doesn't improve, I divide the learning rate by 1.5.

The neural networks are built using several Python libraries. The main library is Pytroch (0.4.0), a python first deep learning framework. For more details, please refer to the README.md document.

4 Results

4.1 Performance Measures

We assign several metrics for this multi-class classification. The first one is the $F1$ score, which is defined as the harmonic mean of precision and recall:

$$F1 = \frac{2}{\frac{1}{recall} + \frac{1}{precise}}. \tag{4}$$

Recall is defined as

$$recall = \frac{TP}{TP + FN}, \tag{5}$$

and precision is defined as

$$precision = \frac{TP}{TP + FP}, \tag{6}$$

here TP, FN and FP denote the numbers of true positive, false negative and false positive respectively. Notice that neither precision or recall takes true negative into account, so $F1$ scores is often used in cases where there are more negatives than positives.

Another metric is Cohen's Kappa:

$$\kappa = \frac{p_0 - p_e}{1 - p_e}, \tag{7}$$

where p_0 is the relative observed agreement among raters, and p_e is the hypothetical probability of chance agreement. Cohen's Kappa is used to measure the concordance between sets of given answers, taking into consideration the possibility of random agreements happening. κ takes value from -1 to 1, and a positive κ indicates that the classifier is better than randomly choosing a label.

Table 2: Performance measures for asset a.

	Downward			*Stationary*			*Upward*			
	Prec.	Recall	$F1$	Prec.	Recall	$F1$	Prec.	Recall	$F1$	κ
RNN	0.22	0.47	0.30	0.98	0.93	0.95	0.19	0.58	0.29	0.28
LSTM	0.11	0.65	0.19	0.99	0.83	0.90	0.14	0.63	0.23	0.18
CNN	0.15	0.52	0.23	0.98	0.87	0.93	0.13	0.60	0.22	0.20

Table 3: Performance measures for asset b.

	Downward			*Stationary*			*Upward*			
	Prec.	Recall	$F1$	Prec.	Recall	$F1$	Prec.	Recall	$F1$	κ
RNN	0.11	0.41	0.18	0.98	0.35	0.52	0.03	0.81	0.05	0.03
LSTM	0.05	0.71	0.09	0.99	0.61	0.75	0.10	0.55	0.17	0.07
CNN	0.17	0.26	0.21	0.97	0.93	0.95	0.14	0.40	0.21	0.19

4.2 Main Results

Table 2 and 3 summary the performance of three models for asset a and b. As illustrated in Table 2, RNN outperforms LSTM and CNN in both $F1$ score and κ for assent a. The relatively low precision of *Downward* and *Upward* is due to the extremely unbalanced test dataset: even if a quiet small proportion of negatives are incorrectly classified as positive, the recall for the positive will be small. The low precision also leads to low $F1$ score. This also explains why RNN get the highest $F1$ score: The *Stationary* recall by RNN is 0.93, which is larger than LSTM, 0.83 and CNN, 0.87. This means that for RNN, much less *Stationary* observations have been incorrectly signed to *Downward* and *Upward*, which improves the precision of RNN for both *Downward* and *Upward*. Though RNN has smaller *Downward* and *Upward* recalls than the other two models, it gets the highest $F1$ score. Compare to RNN, LSTM does not improve the accuracy. This situation also happened in Dixon (2018), where the author found that longer memory benefit was negligible. The reason might be that LSTM gets overfitting. From Table 2, we also find that all three models have κ larger than 0, which indicates that they all have prediction power.

As illustrated in Table 3, CNN performs the best. The two RNN models do poorly in predicting asset b, though they still get positive κ. CNN delivers consistent performance between two assets, which indicates that CNN does identify important features in sequential limit order data. From Table 2 and 3, we find that the recurrent units can still help with prediction. Thus, a combination model of CNN and RNN may be applied here, which first extracts features from CNN and then send them to a RNN. This combination model will be our next interest.

Figure 1 illustrates the prediction process for asset a. Here we compare the RNN, LSTM and CNN estimated probabilities of next event price *Upward* (red) or *Downward* (green) over a 2 second period from 09:45:22 to 09:47:22. The blue line represents the observed change in mid-price. We observe in this period, CNN correctly predicts the most *Upward* movements while also falsely predicts movements when the mid-price is neutral. RNN and LSTM performs similarly, except that LSTM seems more likely to predict *Downward* movement.

Notice that for all three models, predicting price movement of the next event takes less than 1ms, which is far less than the time interval between two events. Although one can easily change the prediction horizon (e.g. from next one event to the next five event) by changing one parameter in my programming, it is actually not necessary to do so. We can develop a high frequency trading strategy based on our predictive models (e.g. long at best ask price when predicting *Upward* and short at best bid price when predicting *Downward*), and it will be interesting to back-test the performance of our strategy.

5 Conclusions

In this report, Convolutional Neural Network (CNN) and deep Recurrent Neural Networks (RNN) have been implemented to predict the next price movement. The empirical results of two assets show that CNN can extract important features from sequential limit order data, while recurrent unit is still useful to construct the prediction model. For further study, we can build a combination model with CNN and RNN to integrate their advantages. Moreover, trading strategy can be built and a back-testing can be applied.

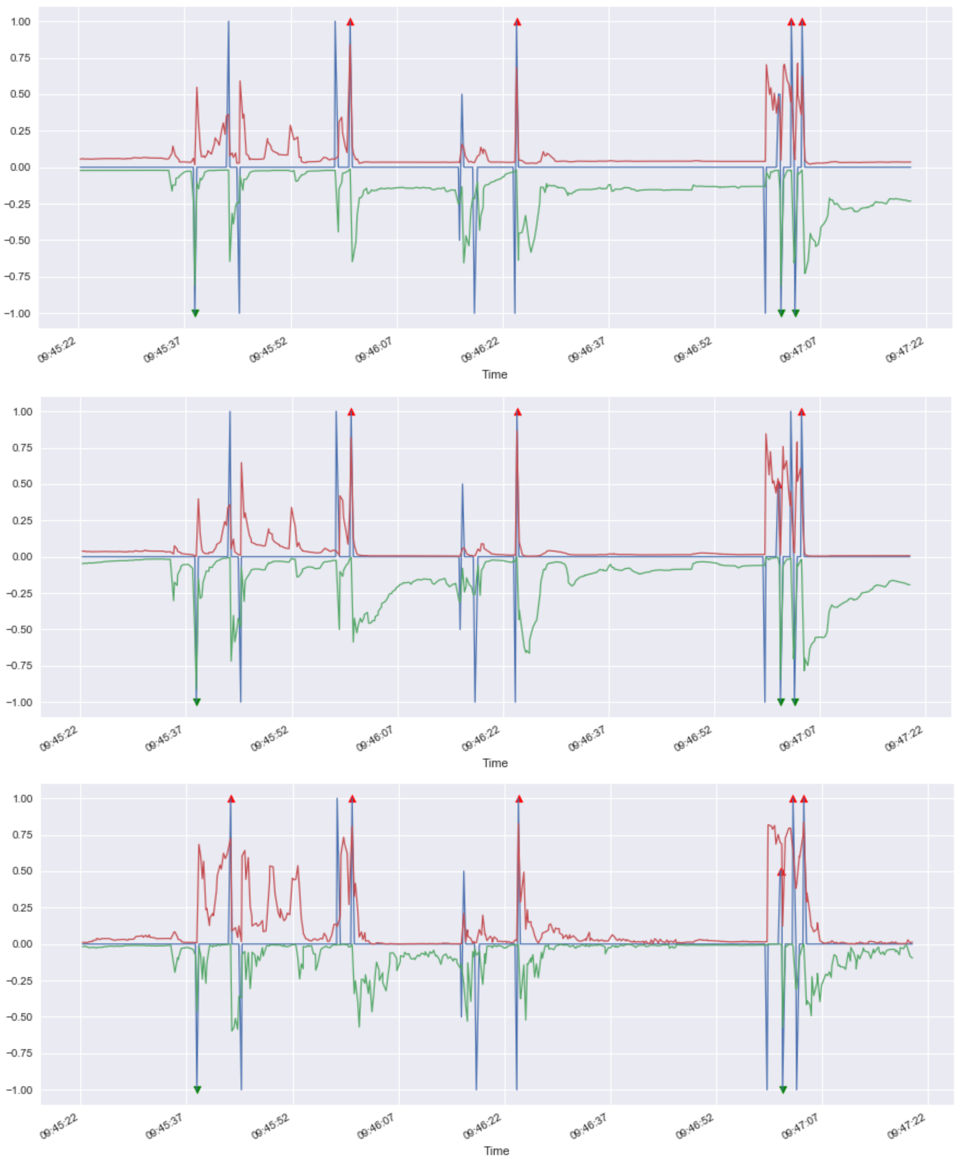

Figure 1: (top) Estimated probabilities of next event price movement *Upward* (red) or *Downward* (green) using RNN. The blue line represents the observed change in mid-price over a 2 second period from 09:45:22 to 09:47:22. The red triangle represents the correct *Upward* label and the green inverted triangle represents the correct *Downward* label. (middle) and (bottom) are using LSTM and CNN separately.

References

Briana Chang. Adverse Selection and Liquidity Distortion. *The Review of Economic Studies*, 85(1):275–306, jan 2018. ISSN 0034-6527. doi: 10.1093/restud/rdx015. URL `http://academic.oup.com/restud/article/85/1/275/3108820`.

Matthew Dixon. Sequence classification of the limit order book using recurrent neural networks. *Journal of Computational Science*, 24:277–286, jan 2018. ISSN 18777503. doi: 10.1016/j.jocs.2017.08.018. URL `https://www.sciencedirect.com/science/article/pii/S1877750317309675`.

Jonathan Doering, Michael Fairbank, and Sheri Markose. Convolutional neural networks applied to high-frequency market microstructure forecasting. In *2017 9th Computer Science and Electronic Engineering (CEEC)*, pages 31–36. IEEE, sep 2017. ISBN 978-1-5386-3007-5. doi: 10.1109/CEEC.2017.8101595. URL `http://ieeexplore.ieee.org/document/8101595/`.

Alec N. Kercheval and Yuan Zhang. Modelling high-frequency limit order book dynamics with support vector machines. *Quantitative Finance*, 15(8):1315–1329, aug 2015. ISSN 1469-7688. doi: 10.1080/14697688.2015.1032546. URL `http://www.tandfonline.com/doi/full/10.1080/14697688.2015.1032546`.

Milla MMkinen, Alexandros Iosifidis, Moncef Gabbouj, and Juho Kanniainen. Predicting Jump Arrivals in Stock Prices Using Neural Networks with Limit Order Book Data. *SSRN Electronic Journal*, apr 2018. ISSN 1556-5068. doi: 10.2139/ssrn.3165408. URL `https://www.ssrn.com/abstract=3165408`.

Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Using deep learning to detect price change indications in financial markets. In *25th European Signal Processing Conference, EUSIPCO 2017*, volume 2017-Janua, pages 2511–2515, 2017a. ISBN 9780992862671. doi: 10.23919/EUSIPCO.2017.8081663. URL `http://www.eurasip.org/Proceedings/Eusipco/Eusipco2017/papers/1570346870.pdf`.

Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Forecasting Stock Prices from the Limit Order Book Using Convolutional Neural Networks. In *2017 IEEE 19th Conference on Business Informatics (CBI)*, pages 7–12. IEEE, jul 2017b. ISBN 978-1-5386-3035-8. doi: 10.1109/CBI.2017.23. URL `http://ieeexplore.ieee.org/document/8010701/`.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent Neural Network Regularization. (2013):1–8, sep 2014. ISSN 0157244X. doi: ng. URL `http://arxiv.org/abs/1409.2329`.