Quantitative Finance

ISSN: 1469-7688 (Print) 1469-7696 (Online) Journal homepage: https://www.tandfonline.com/loi/rquf20

Encoding of high-frequency order information and prediction of short-term stock price by deep learning

Daigo Tashiro, Hiroyasu Matsushima, Kiyoshi Izumi & Hiroki Sakaji

To cite this article: Daigo Tashiro, Hiroyasu Matsushima, Kiyoshi Izumi & Hiroki Sakaji (2019): Encoding of high-frequency order information and prediction of short-term stock price by deep learning, Quantitative Finance, DOI: 10.1080/14697688.2019.1622314

To link to this article: https://doi.org/10.1080/14697688.2019.1622314

Published online: 09 Jul 2019.

Quantitative Finance, 2019
https://doi.org/10.1080/14697688.2019.1622314

Encoding of high-frequency order information and prediction of short-term stock price by deep learning

DAIGO TASHIRO, HIROYASU MATSUSHIMA *, KIYOSHI IZUMI and HIROKI SAKAJI

Graduate School of Engineering, The University of Tokyo, Tokyo, Japan

(*Received 2 July 2018; accepted 6 May 2019; published online 9 July 2019*)

Predicting the price trends of stocks based on deep learning and high-frequency data has been studied intensively in recent years. Especially, the limit order book which describes supply-demand balance of a market is used as the feature of a neural network; however these methods do not utilize the properties of market orders. On the other hand, the order-encoding method of our prior work can take advantage of these properties. In this paper, we apply some types of convolutional neural network architectures to order-based features to predict the direction of mid-price trends. The results show that smoothing filters which we propose to employ rather than embedding features of orders improve accuracy. Furthermore, inspection of the embedding layer and investment simulation are conducted to demonstrate the practicality and effectiveness of our model.

Keywords: Mid-price trend forecast; Convolutional neural network; Deep learning

1. Introduction

Is it possible to predict the rapidly fluctuating prices of financial products? In response to this question, many theoretical and empirical studies have been conducted by practitioners and researchers in various fields. In recent years, following the proposal of the Efficient Market Hypothesis and the publication of empirical research refuting it, information technology (especially machine learning and deep learning) has been increasingly applied for financial and economic analyses. These techniques offer the ability to recognize patterns and uncover the knowledge and rules hidden within the data. They have achieved promising results in market forecasting.

While many studies of market forecasts have been conducted, the behavior of the market is changing dramatically. Along with computerization, which has sped up trading on the market, strategies to conduct automated transactions, such as algorithmic trading and high-frequency trading (HFT), have been put to practical use. The order data and transaction data from the market can be observed at a very high sampling frequency, and the volume of data has become enormous. It is expected that this data, called 'high-frequency data', can be leveraged for various purposes.

Research has been conducted to explore the application of deep learning to high-frequency trading data. In particular, many studies using order book data have been reported (Tsantekidis *et al.* 2017, Dixon 2018).

For example, the trend in the mid-price was predicted using a neural network that takes the order prices and quantities of 'asks' (i.e. sell orders) and 'bids' (i.e. buy orders) as reported in the order book information as the inputs. It exhibited better prediction accuracy than machine learning.

Market orders, which represent trader's strong intentions and promises to issue prompt payment, greatly influence the market and there is a strong correlation between the market orders and the market return. In addition, limit orders and cancelation orders may also affect prices and must be considered (Eisler *et al.* 2012, Cont *et al.* 2014). However, it is challenging to deal with order book information using existing methods because it is difficult to determine whether a decrease in the best ask or best bid in the order book is due to a successful order or a cancelation of an order. Hence, a neural network can be used to differentiate successful orders by inputting the order series itself into a model.

In this paper, we propose methods to predict short-term price trends using deep learning based on the high-frequency order series which includes all order types.

First, we propose the order-encoding method that can be used to convert order book information of varying quality

*Corresponding author. Email: matsushima@sys.t.u-tokyo.ac.jp

into input variables for the neural network. Then, we propose an extension of a convolutional neural network (CNN) called an average convolutional neural network (A-CNN) that can be used to acquire a learned model that extracts features to predict price given order book information. Furthermore, we propose an A-CNN+ that is improved to overcome the problems associated with the A-CNN. The proposed approach may ultimately be used to improve the operational performance of algorithmic trading.

The remainder of this paper is organized as follows; Section 2 describes the relevant prior work in this field. In Section 3, we present the proposed method. In Section 4, we describe the experimental validation of the proposed technique on historical price-trend data and compare the results with those obtained by other approaches. In Section 5, we present the results of a simulation of stock price prediction including the simulated improvements in revenue that can be attained using different parameters in the proposed approach. Finally, conclusions are presented in Section 6.

2. Related works

In this section, we describe previous researches involving applications of machine learning to high-frequency trading data. Kercheval predicted the fluctuation in the mid-price by using a support vector machine (SVM) (Hearst *et al.* 1998) based on features such as the bid price and the bid-ask spread (Kercheval and Zhang 2015). Similarly, Fletcher predicted the exchange rate between the Euro and the US dollar using an SVM based on the limit order books (Fletcher and Taylor 2015).

Tsantekidis pointed out that irrational human behavior cannot be captured in designed features and mathematical models, which limits the accuracy of the approach (Tsantekidis *et al.* 2017). Thus, in recent years, market forecasting has been performed without feature extraction using neural networks based on the raw book orders as inputs. Tsantekidis used a neural network to predict trends in the mid-price (Tsantekidis *et al.* 2017). Tsantekidis predicted the mid-price from the ask/bid prices and the quantity of orders accumulated on the order books using long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997); this approach exhibited better performance than SVM. A single learned model was acquired by LSTM using the order book data of five normalized stocks collected over 10 days. However, the shape and fluidity of the order book can vary depending on the stocks. Furthermore, with regard to the prices in the market, it is considered difficult to normalize the data from the order books because it has power-law characteristics and multiple peaks. Therefore, learned models must be created to every stock and a large amount of data is needed for the learning process.

Further, more serious problems are involved in the classification of the market orders. When market orders are used as explanatory variables, it is possible to dynamically trace the strength and balance of the ask and bid at each point in time. However, when the best ask or best bid on the market order decreases, it is impossible to determine whether it is due to the completion of a market order or an order cancelation. With the emergence of HFT in recent years, cancel orders are becoming frequent, which will make this issue even more difficult to address.

Other studies have used neural networks to predict market prices using current prices, quantities, and order books as the inputs (Dixon *et al.* 2017, Sirignano 2019). Dixon focused on market orders and proposed an extended method to generate features based on the ratio of buy and sell orders from a past order series and add the features to the input vector (Dixon 2018).

However, no studies have successfully predicted market prices by directly inputting a time series of all order types into a CNN as we propose herein.

3. Proposed method

The method proposed in this study is presented as follows: First, we describe the process used to encode market orders. Then, we describe the initial version of the A-CNN. Lastly, we describe the modification of the A-CNN to address its flaws, giving rise to the A-CNN+.

3.1. Order-encoding method

The features of an order are represented as different variables; price, quantity, and time of the order are represented as numerical values (quantitative variables) and buy and sell orders are represented as categorical information (qualitative variables).

In this section, we describe the method used to encode orders before inputting them into the neural network. The order-encoding is done in two stages:

(i) Categorization: convert all qualitative variables to categorical variables belonging to orders.
(ii) Conversion: convert all variables into a single qualitative variable.

The first step is executed as follows for each variable:

order type: The order type (buy, sell, order, limit, and cancel) is a categorical qualitative variable and is used without modification.

price: When qualitative variables are input to a neural network, normalization is generally required. However, the price distribution of the orders is asymmetrical (i.e. a power-law distribution) and multimodal, so it is difficult to calculate standard deviation of it. In addition, preprocessing is difficult because the orders associated with different stock code have different distributions. Therefore, in this paper, we consider the relative price, which is defined as the difference between the price and the mid-price, rather than the absolute price. Figure 1(top) shows the frequency distribution of the relative prices of the orders and the mid-prices at the time of ordering. Figure 1 (bottom) shows the frequency

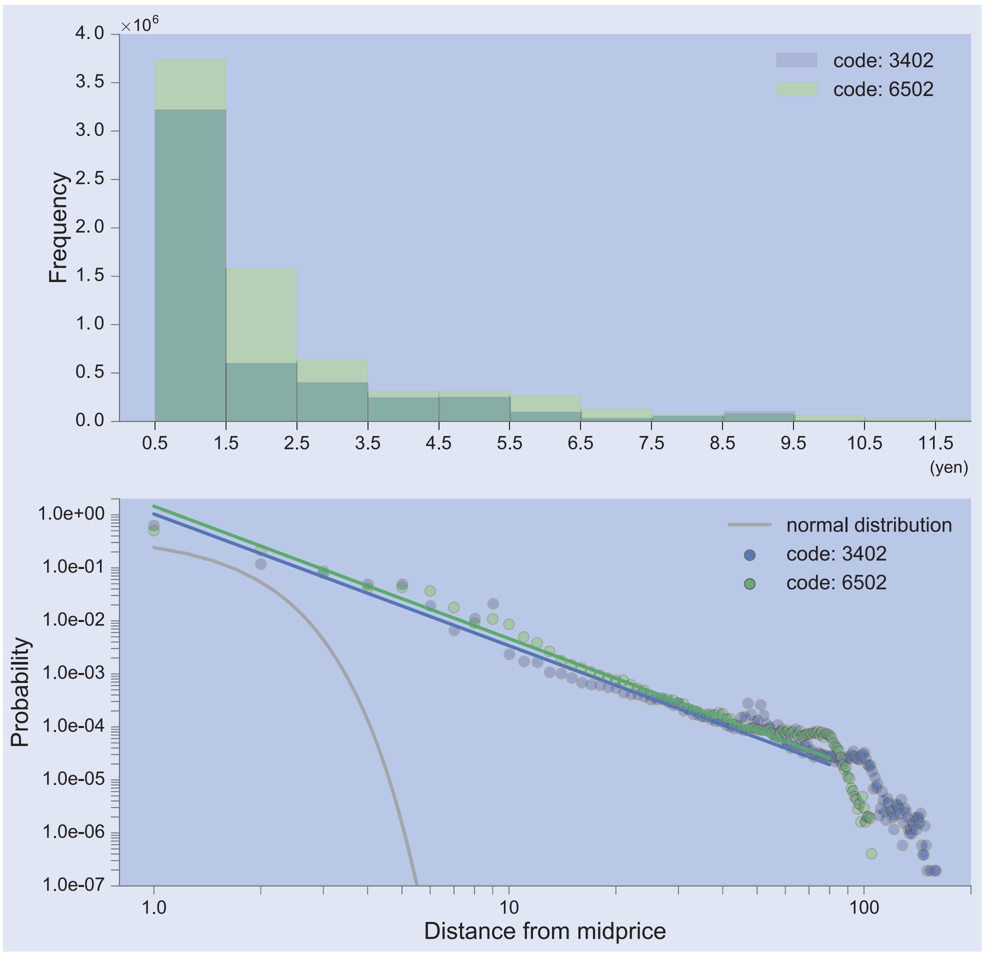

Figure 1. Frequency distributions of the relative prices of limit orders and cancel orders. Top: differences between the mid-price and order price when the orders are placed. Bottom: frequency distributions on a log-log graph: the black curve shows a normal distribution.

distributions of normalized order quantity plotted on a log-log graph. As shown in figure 1 (bottom), although the frequency distributions for stock codes 3402 and 6502 differ, both distributions decrease at prices farther away from the mid-price. Furthermore, it is confirmed that these frequency distributions are similar to power-law distributions, for which averages and standard deviations cannot be identified. Since it is difficult to standardize different price ranges for various stock codes, it is categorized rather than numerical information and used. Also, with relative price, the limit order far from the mid-price is considered to have less impact on price. For these reasons, it is suggested that categorization of order information is superior to normalization.

time difference: When analyzing high-frequency data, it is difficult to classify what kinds of orders have been made. However, assuming that the order type varies with the participants in the order market, it is hypothesized that the accuracy of the market prediction can be improved by associating the information about the participant placing an order with the other order information. To do this, the time-difference relative the previous order is used to classify orders that occur at millisecond intervals as orders made by mechanical traders. Thus, approximate classification information about which transactions are conducted by a particular trader can be included.

Here, by using equation 1, the embedding order, x_t, can calculated at time t.

$$x_t = w_{embed} x_i^{onehot}. \tag{1}$$

The product of the order type (represented by the onehot vector) and the embedding matrix, w, is calculated to derived the embedding vector, x_t, of the order.

The second step in the encoding process is conducted as follows: In order to unify multiple categorical variables and express them as a single variable, the order to be input to the neural network is first normalized. Figure 2 provides a concrete schematic of the order-encoding method. The direct product of plurality of categories represented by matrices is calculated. It can be seen that prices far from mid-price

$$\begin{pmatrix} \text{MarketOrder}^{\text{ask}} \\ \text{MarketOrder}^{\text{bid}} \end{pmatrix} \times \begin{pmatrix} \sim 20 \\ 20 \sim 500 \\ 500 \sim \end{pmatrix}$$

$$\begin{pmatrix} \text{LimitOrder}^{\text{ask}} \\ \text{LimitOrder}^{\text{bid}} \\ \text{Cancel}^{\text{ask}} \\ \text{Cancel}^{\text{bid}} \end{pmatrix} \times \begin{pmatrix} \sim 1 \\ 1 \sim 2 \\ 2 \sim 3 \\ 3 \sim 5 \\ 5 \sim 7 \\ 7 \sim 10 \\ 10 \sim \end{pmatrix} \times \begin{pmatrix} \sim 20 \\ 20 \sim 500 \\ 500 \sim \end{pmatrix}$$

Order type category — Price category [Yen] — Time difference category [ms]

Figure 2. Order-encoding used in this study.

(which have less influence) are categorized together. The time-difference categories are set with the expectation that it is possible to further differentiate automated orders as high-frequency automated orders (if the time difference is less 20 ms) and relatively low-frequency orders (if the time difference is between 20 and 500 ms).

Although even at the same price, limit orders and cancel orders hold different meanings, this feature-representation method of encoding orders to be input to a neural network helps to achieve the objective of dealing with both market orders and limit orders. In addition, this method can make the relationship between the first layer of the neural network and the order information unique, and facilitate the evaluation of the convolution layer.

3.2. *Prediction model using average order embedding*

Convolutional neural networks (CNN) have been successful for tasks, such as document classification, not only image recognition but also in natural language processing (Kim 2014, Johnson and Zhang 2015). In this research, we used CNN for the order time series because CNN has invariance with respect to position.

CNN performs maximum pooling in the sequence direction post convolution. Even if the limit orders or cancel orders concentrate behind the series, recognizing the market order and learning features and patterns of price trends is possible using CNN. Hence, we used a model that uses CNN to convolve local order sequences and extract patterns.

Sequence $\mathcal{S}$ unified by constant n by padding using the embedding vector $\boldsymbol{x}_t$ of the order x_t is represented as $\boldsymbol{x}_{1:n} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]$. The local matrix $\boldsymbol{x}_{i:i+h}$ was convolved, and a new feature c_i was obtained by applying an activation function. When this convolution was performed on $(\boldsymbol{x}_{1:h}, \boldsymbol{x}_{2:h+1}, \ldots, \boldsymbol{x}_{n-h+1:n})$ with a stride width of 1, a new feature vector $\boldsymbol{c} = [c_1, c_2, \ldots, c_{n-h+1}]^T \in \mathbb{R}^{n-h+1}$ was obtained. By performing maximum pooling on the $\boldsymbol{c}$, a feature amount $\hat{c} = \max(\boldsymbol{c})$ was obtained from one filter. This process was performed for some convolution filters. Features $\hat{\boldsymbol{c}} = [\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_{k_{\text{conv}}}]^T$ obtained via convolution using k_{conv} filters and maximum pooling was input to the entire fully connected layer. Using the SoftMax function, we converted the output class obtained from the fully connected layer.

Several sizes of convolution filters were prepared to change the length of the sequence to extract the pattern based on its size. For example, if the filter was large size, the global relationship of orders was captured, whereas if the filter was small size, the local relationship was captured.

3.3. *Average convolutional neural network: A-CNN*

In the time series data for financial markets, even if a certain price trend is observed, few clear patterns exist in the microstructure, such as order units. In a time series of orders, it is assumed that the order interaction for a local range captured by the convolution filter is small. To reduce these effects by averaging a certain number of orders, we propose using average-CNN (A-CNN) as a prediction model, which uses averaging of order embedding.

Figure 3 shows an overview of A-CNN. A-CNN uses a two-dimensional (2D) embedding matrix (i.e. order embedding) as input. By using proposed order-encoding method, various plural order information in one stock code is calculated as w_{embed}, which is represented as the matrix of embedding size (column) and the number of orders (row) in figure 3. Average pooling was applied to the embedded matrix, $\boldsymbol{x}_{1:n}$, with a window width of $1 \times l_{\text{pool}}$ pool. Padding was performed with a padding size of l_{pad} with respect to

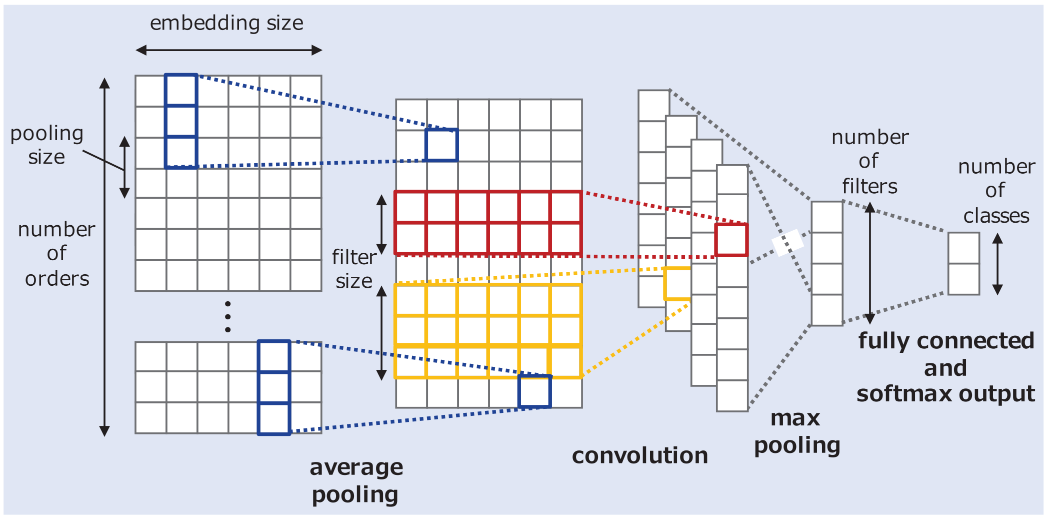

Figure 3. Structure of A-CNN.

the order time series direction prior to pooling. The feature matrix post pooling is represented as $\boldsymbol{x}_{\text{pool}} \in \mathbb{R}^{e\times\lfloor l_{\text{pad}}+n/l_{\text{pool}}\rfloor}$ with the floor function as $\lfloor x \rfloor$. Next, feed forward propagation and learning were conducted similar to CNN.

The effect of averaging also aids in capturing the market order interactions. CNN cannot extract market order features when the interval for the market order is larger than the size of the convolution filter. By adding the averages, extending the length of the order series captured by the filter to $l_{\text{pool}} \times h$ is possible. Based on the above described method, capturing the interactions between the orders of execution with a low occurrence frequency is possible. However, when some market orders are present in the pooling window l_{pool} it is a disadvantage that they are averaged.

Consider an example, wherein using the order-encoding method, order book information was assigned according to each categorized item in figure 2, '1' was assigned to the applicable category, '0' was otherwise assigned, and the 0/1 embedding matrix was input into the neural network. Because the embedding matrix is represented as 0/1, when max pooling is used in the pooling layer, many variables of the matrix through the pooling layer may be 1. Hence, we considered max pooling to be inappropriate, and we used average pooling instead.

3.4. A-CNN+

Next, we propose the A-CNN+ model, which is shown in figure 4, as an extension of the A-CNN described in the previous subsection. Multiple averaging matrices are created by performing average pooling on the order embedding matrix using different window widths. A convolution filter is then prepared for each averaging matrix and convolution and maximum pooling are applied. These vectors are input to the entire fully connected layer as in a CNN.

The purpose of this operation is to diversify the size of the context of orders series and to make convolution filters of various sizes. In other words, the larger the window of the average pooling, the more extensively a feature is extracted by the convolution filter. Conversely, the smaller the window for average pooling, the more locally the feature extraction is performed. Here, we hypothesize that the useful pattern for predicting price fluctuations can vary in size within the order series; there may be large trends, like price momentum in a time series from a global point of view, and smaller patterns, such as microeconomic interactions from a local point of view. By using pooling windows and convolution filter windows of various widths, the A-CNN+ can correspond to patterns of various lengths. Based on this way, the model can synergistically capture a larger variety of relationships between orders.

4. Experiment

4.1. Data set

In this experiment, dataset comprised 20 stocks of FLEX-FULL historical data, which include data from the Tokyo Stock Exchange over 245 business days between 1 July 2013 to 30 June 2014. Two classes (Up, Down) and three classes (Up, Down, Neutral) classification problem were set for each issue for which the mid-price was the target of prediction.

This dataset was selected in consideration of the practice of algorithmic trading. The order sequence (event-driven) to be input to the price-prediction model to decide the price trend (the output) is aligned with the order timing (time-driven) via a transaction algorithm. More specifically, as shown figure 5, a delimiter is set every 30 seconds to delimit one encoded ordered sequence and several such order sequences

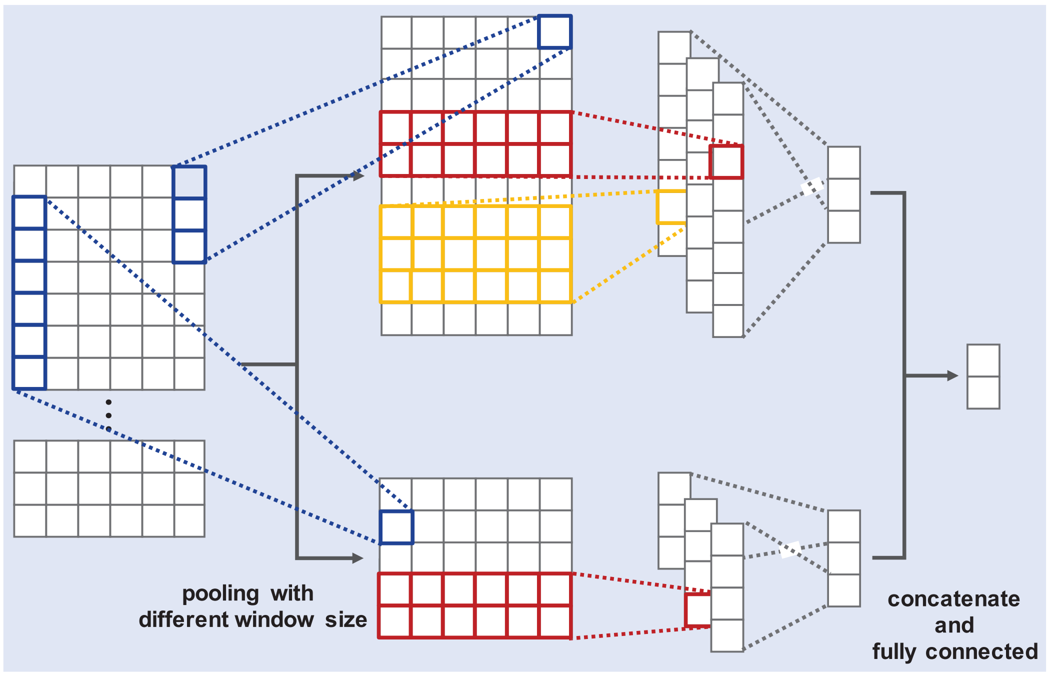

Figure 4. Structure of the A-CNN+.

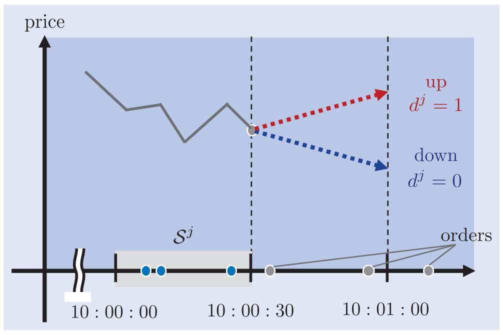

Figure 5. Sampling of a series, $\mathcal{S}^j$, and labeling in two-class classification.

are acquired. It should be noted that although the divided order sequence, $\mathcal{S}^j$, has a variable length, it can be handled effectively on the prediction model side.

As shown figure 5, the output class (the price trend, d^j) corresponding to $\mathcal{S}^j$ was for the time period beginning 30 seconds after the last time point in the input interval. In this experiment, the following classification problems are considered.

- two-class classification to predict up/down price
- three-class classification to predict up/neutral/down price

4.2. Results

The data from each issue over the course of one year is divided in a ratio of 7:1.5:1.5 as learning data, verification data, and evaluation data, respectively. Using the data from each of the 20 stock codes, learning was conducted using the learning data and a parameter search was carried out (Appendix). The model with the best performance on the verification data was selected for use with the evaluation data. The F1 scores for each class in the evaluation data were obtained and the average of the F1 scores was used as the evaluation metric.

Figure 6 shows the experimental results including the F1 scores obtained in the two-class mid-price-prediction experiment on the evaluation data. Logistic regression, nonlinear SVM, multi-layer perceptron (MLP), and a CNN were also applied for comparison. Here, the A-CNN and A-CNN+ provided higher scores than the other baseline methods, indicating that the proposed method outperformed the baseline methods for all issues. In some cases, the A-CNN outperformed A-CNN+ but in other cases, the A-CNN+ was better (figure 7).

4.3. Analysis

In this section, we analyze the convolutional layer. There is a vector representing each order with a norm corresponding to

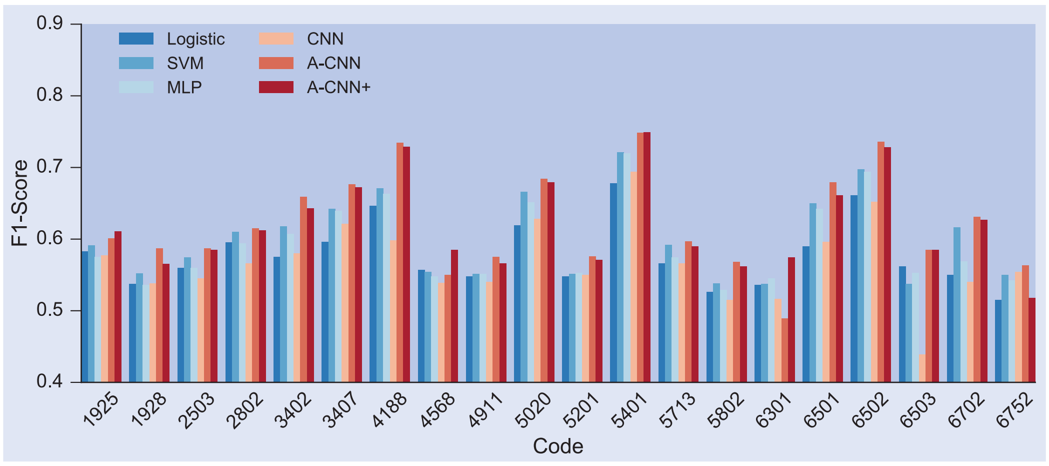

Figure 6. F1 scores of the mid-price prediction for each stock code using two-class classification; the vertical axis represents the F1 score and the horizontal axis denotes each stock code.

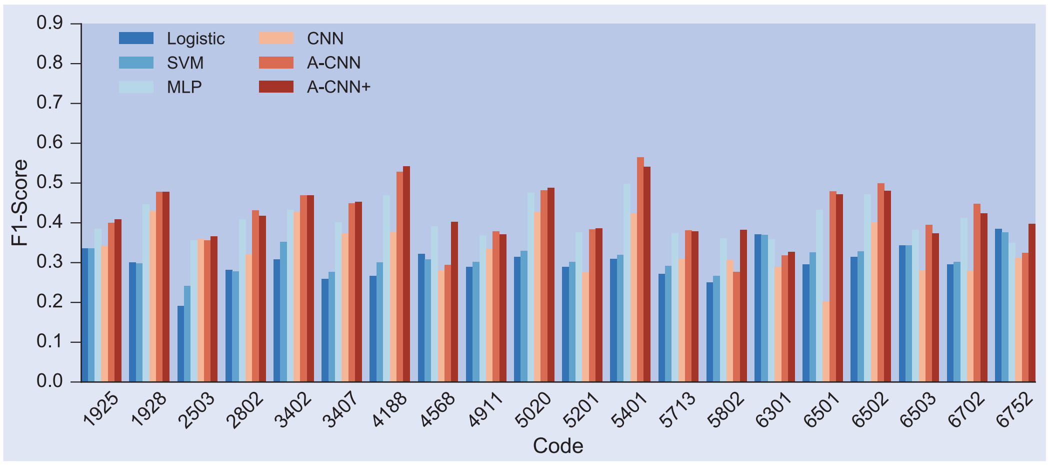

Figure 7. F1 scores of the mid-price prediction for each stock code using three-class classification.

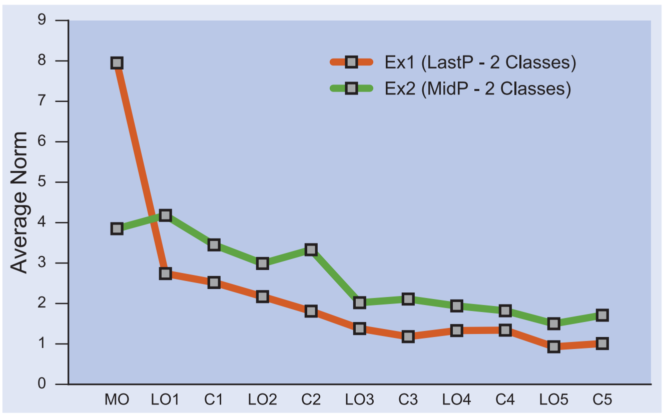

Figure 8. Norm averages of the embedding vectors in the A-CNN model.

the strength of firing of the neural network. Figure 8 shows the average of norms for the two-class classification problem in case of which forecasting targets are execution-price and mid-price. In figure 8, the average of the norms of the embedding vectors from the market sales orders (*MO*), limit orders (*LO*), and cancelation orders (*C*) in each price range are calculated for all the stock codes. The numbers (e.g. *LO*1 and *C*2) represent price categories: 1, 2, 3, 4, and 5 indicate the price categories ∼ 1, 1 ∼ 2, 2 ∼ 3, 3 ∼ 5, and 5 ∼ 7, respectively, as shown figure 2. *LastP* indicates the execution-price and *MidP* indicates the mid-price.

In forecasting execution price, the weight of the market order is relatively high. On the other hand, in the mid-price prediction, orders and cancelations with prices close to the mid-price are weighted the same as market orders.

This is because the prediction of the mid-price is more likely to be affected by the limit order and the cancel order in comparison with to the prediction of the execution price. The mid-price fluctuate by entering a limit order and a cancel order near the best ask and the best bid, but the execution-price change only as a result of entering a market order. Since the mid-price also fluctuates depending on the market order, it is considered that network learned to respond to all types of orders including the market order. This analysis result can be convinced from the property of the execution price and the mid-price, it shows that the proposed method works as intended.

5. Simulation of stock price reduction

The Softmax function was used for the output layer of the neural network, and the probability is available. By making investment using output only when the probability by the model is high, it is possible to increase accuracy and improve performance per transaction. In addition, a threshold was set for the maximum output probability, and the investment behavior was determined.

The higher the probability of this output, the better features are captured and the more reliable output is obtained. As a reason for the above, figure 9 is shown. In figure 9, the bar graph indicates the precision (vertical axis on the left), and the line graph indicates the ratio of the number of samples whose output probability exceeds the threshold value (vertical axis on the right). The threshold values were set to 0.50, 0.55, and 0.60. The chance rate (i.e. vertical axis on the right) indicates the proportion to the total number of samples in the evaluation data. The precision is when a threshold is set for the output of each evaluation data. The precision was calculated using the number of samples whose output exceeded each threshold value. As the threshold value was set higher, the evaluation value increased, and the number of times to output a probability exceeding the high threshold decreased.

When conducting investment simulations, we compared a two-class classification problem with a three-class classification problem using profit. The model with the highest F1 value in the verification data among the A-CNN and A-CNN+ was adopted.

In investment simulation, it was set to cost each time a transaction was made. For simplicity, the spread was uniformly set to 1 yen, and the cost of one transaction was set to 0.5 yen for half the thread. Furthermore, assuming that it operated on all stock codes, we calculated the profit obtained from this portfolio.

The corresponding result is shown in figure 10. Profits were calculated as we varied the thresholds in investment simulations. As a feature of each classification problem, we showed that profit became positive at a certain point as the threshold was increased. As the threshold increased, it went through a point where the profit became maximum, then decayed, and approached 0. The range of positive profit means that profit exceeds the cost of half-spread in one transaction. Although

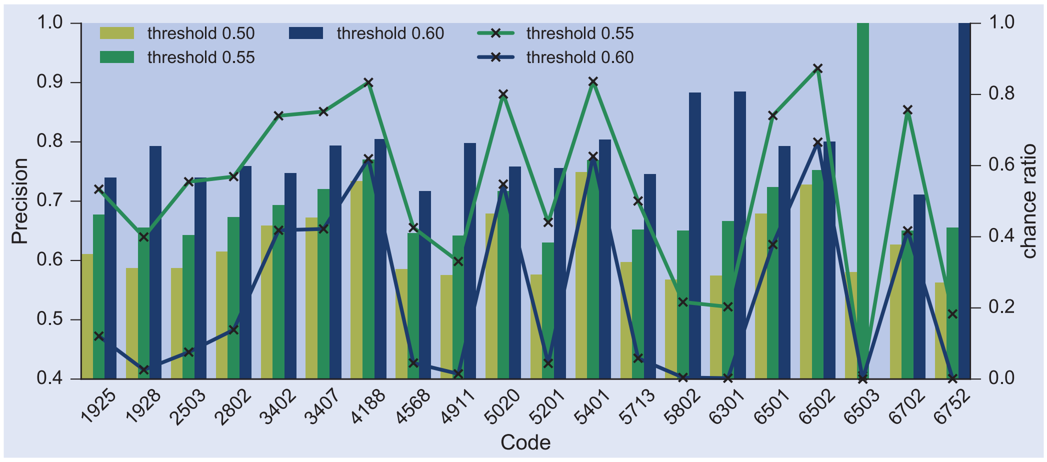

Figure 9. Ratio of the precision and prediction opportunities when the threshold was set to output (two-class classification).

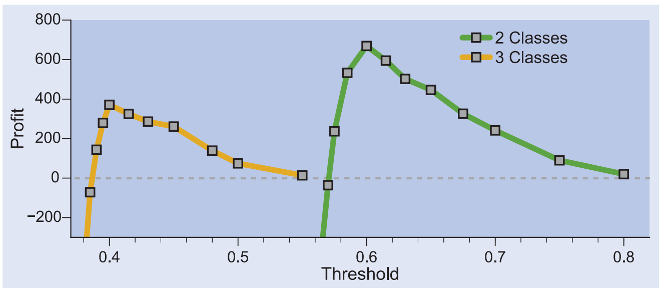

Figure 10. Changes in profit as a function of testing different thresholds. Total profit for all brands with a set cost.

using large threshold the accuracy of the prediction could be increased, the sum of profits declined because the number of transactions become decrease.

From these results, we showed that the proposed order-encoding method and averaging of the embedding matrix are practical. In addition, this result show that the two-class classification can generate larger profit than the three-class classification and can increase the positive profit in many thresholds. This results suggest that the two-class classification was easier to learn than the three-class classification.

6. Conclusion

In this research, in order to predict the short-term trend of stock price, we proposed an order-encoding method and improved CNN models (A-CNN, A-CNN+) which are suitable for capturing features of the orders. Through experiments using large-scale high-frequency trading data, the experimental results showed that the proposed method is superior to other benchmark methods. Furthermore, it was confirmed that the convolutional layer performed as intended. Especially in prediction of mid-price, we confirmed that an improved CNN by the proposed method can learn to capture features of the order book.

The experimental results suggest that the learned model responds strongly to the ordering process in all tasks. In the prediction of mid-price, we found that the limit orders and cancel orders close to the mid-price are important as same as the market orders. Finally, investment simulations were conducted to demonstrate a practical application to buying and selling according to the model predictions by the proposed method. Positive results were achieved through investment simulations, and the practicality of the model is suggested. Furthermore, the performance was improved by setting a threshold for the probability of output, and results show that the two-class classification model outperforms the three-class classification model.

In future works, the incorporation of more detailed order book information will be explored. In this paper, we focused on information on the type and price of orders sampled in fixed time interval. By introducing more fluid information, it is potentially that prediction accuracy is improved. In addition, it is necessary to develop the method to utilize it.

Disclosure statement

No potential conflict of interest was reported by the authors.

ORCID

Hiroyasu Matsushima http://orcid.org/0000-0001-7301-1956

References

Cont, R., Kukanov, A. and Stoikov, S., Price impact of order book events. *J. Financ. Econ.*, 2014, **12**, 47–48.

Dixon, M., Polson, N. and Sokolov, V., Deep learning for spatio-temporal modeling: Dynamic traffic flows and high frequency trading. *Appl. Stoch. Models. Bus. Ind.*, 2017, 1–20.

Dixon, M., Sequence classification of the limit order book using recurrent neural networks. *J. Comput. Sci.*, 2018, **24**, 277–286.

Eisler, Z., Bouchaud, J.P. and Kockelkoren, J., The price impact of order book events: Market orders, limit orders and cancellations. *Quant. Finance*, 2012, **12**, 1395–1419.

Fletcher, T. and Taylor, J.S., Multiple kernel learning with fisher kernels for high frequency currency prediction. *Comput. Econ.*, 2015, **15**, 1315–1329.

Hearst, M., Dumais, S., Osuna, E., Platt, J. and Scholkopf, B., Support vector machines. *IEEE Intell. Syst. Appl.*, 1998, **13**, 18–28.

Hochreiter, S. and Schmidhuber, J., Long short-term memory. *Neural. Comput.*, 1997, **9**, 1735–1780.

Johnson, R. and Zhang, T., Effective use of word order for text categorization with convolutional neural networks. Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 103–112, 2015.

Kercheval, A. and Zhang, Y., Modeling high-frequency limit order book dynamics with support vector machines. *Quant. Finance*, 2015, **15**, 1315–1329.

Kim, Y., Convolutional neural networks for sentence classification. *Empirical Methods Nat. Lang. Process*., 2014, 1746–1751.

Sirignano, J.A., Deep learning for limit order books. *Quant. Finance*, 2019, **19**, 549–570.

Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M. and Iosifidis, A., Using deep learning to detect price change indications in financial markets. Proceedings of the 25th European Signal Processing Conference, pp. 2511–2515, 2017.

Appendix

Parameters

Here, search parameters related to each method used in this paper are described. Parameters were searched using grid search.

Table A1. A-CNN: ReLU was used for the activation function.

Parameter	Range
dimensions of the embedding layer	[3, 5, 10]
Size and num. of filter (size:num.)	[3:20, 5:20, 7:20, 3:5, 5:5, 7:5, 10:5]
Pooling size	[5, 10, 15]

Table A2. A-CNN+: Size of filter is fixed as 20. ReLU was used for the activation function.

Parameter	Range
dimensions of the embedding layer	[3, 5, 10]
Number of filter	[3, 5, 7, 3, 5, 7, 10]
Size of average pooling	[5, 10, 5, 10, 15]