Prediction of short-term stock price by A-CNN+

COMSE6998 010 Final Project Report

Ruizhe Li (rl3070), Ruochen Nan (rn2498)

December 16, 2020

1 Introduction

1.1 Executive Summary

The goal of this project is using order information and deep learning methods to do the short-term stock price trend prediction. Since a large portion of electronic trading in stocks operates through limit order markets, we want to utilize order features from limit order book which describes supply-demand balance of a market is used as the feature of a neural network.We apply some types of convolutional neural network architectures and the Long Short-Term Memory Units (LSTMs) to order-based features to predict the direction of price trends. The benefits of our project is firstly we utilize properties of market orders. What's more, we tried inception module for CNN to take advantage of multi-level feature extraction. Lastly, we also tried LSTM to gain a higher accuracy.

1.2 Problem Motivation

Electronic order driven markets where buy and sell orders are centralized in a limit order book and executed against the best available orders on the opposite side are increasingly replace the previous quote-driven markets where prices were set by a market-maker. A large portion of electronic trading in stocks operates through limit order markets like NYSE and NASDAQ. In these markets, participants may submit a limit order to buy or sell a certain quantity at a limit price or a market order to buy or sell a certain quantity and this is executed against the best available limit order. Market orders are executed against outstanding limit orders on the opposite side, based on price priority and time priority, which means best available price gets executed first and first in, first out.
More and more stocks and other financial instruments are traded through electronic trading platforms. Therefore, traders now attempt to utilize data from the limit order book once they get those information from the stock exchange.

1.3 Background Work

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. Lots of researchers tried to utilize deep learning methods in short-term stock price prediction. However, their research is mainly based on the data from three perspectives (Bao W et al. 2017).
The first perspective is daily trading data like the price and trading volume. The second perspective is technical indicator like 5, 10, 20 days moving average and William's variable which measures the buying and selling pressure. The third perspective is microeconomic variable like exchange rate and interest rate.
When having data from the limit order book data which is believed to have embedding liquidity information, researchers tends to more focus on order data. They normally use below three types of data: (Kercheval and Zhang 2015, Dixon M 2017) The first tyoe is prices as categorical form. Also they used Volumes.Then the third type is number of different type of orders on both the ask and bid side of the book.
However, in our mainly based paper (Daigo Tashiro et al. 2019), the researcher focus more on order-encoding methods which even doesn't consider the volume of each other so we follow their data processing method but they add time difference data as a way to detect the investment is executed from an analyst or a machine.

2 Approach

2.1 Technical Challenges

Data Collection: Limit order book is collected inside the stock exchange and may not provide to traders in the market, which is probably why traders are not frequently use those data for daily investment in the past. However, now more and more high-frequency trader may collect or buy those limit order book data from the stock exchange. For our project, due to lack of financial resources, we couldn't get the data of FLEX FULL historical data of Tokyo Stock Exchange or

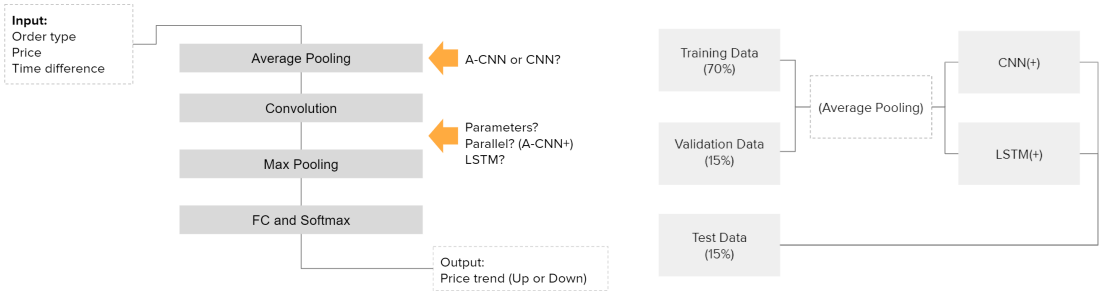
CME FLEX data as previous researcher. However, we luckily to find some free data from NYSE TAQ data and it was also been used in some papers about order books (Rama Cont 2011).
Data Processing: After careful consideration and comparison of different papers, we decided to use order type, price and time difference data since we would like to repeat and further extend the research of our mainly based paper (Daigo Tashiro et al. 2019).
Structure Design: Besides implementation of CNN in our based paper, we also want to explore the performance of LSTM since we could see its excellent performance in stock price prediction using the three main types of data as described above.
Hyperparameter selection: What's more, we need to consider different combination of size of average-pooling, size of filter and number of filter to seek a better performance.

2.2 Goal

The first goal is to explore the effect of average-pooling in our model. We would like to test whether A-CNN model give a better prediction in our data set of market order. Besides, comparing the performance of A-CNN and A-CNN+ to test whether the introduction of inception module work for us. Finally, we change CNN to LSTM to see whether average pooling also provide better prediction when using LSTM.

2.3 Methodology and Solution Diagram

The main methodoglogy and solution diagram is showed in Figure 1.



(a) Methodology (b) Solution Diagram

Figure 1: Methodology and solution diagram.

3 Experiments

3.1 Implementation details

Dataset: TAQ NYSE Equities on Oct 7th, 2019
Hardware: 2vCPUs — 13 GB memory — 1 x NVIDIA Tesla P100 / Tesla T4
Platform: Google Cloud Platform / Colab
Framework: Tensorflow 2
Functionalities: 1) Data processing, 2) Model, 3) Evaluation
Challenges:

- **Dataset:** Hard to find a public dataset perfectly matching the one in the paper.
- **Implementation:** No code provided. Everything from scratch.
- **Intrinsic:** Data processing is hard because price follows the power law.

3.2 Experimental evaluation

This section follows the experiment workflow. We will show how to reproduce our result from data processing, to model building, and finally to evaluation. We also provide our implementation in Jupyter Notebook format in our github repository.

3.2.1 Data Processing

The raw data downloaded is shown in the snapshot below. It has 9 columns: symbol, status , date, time, order type, price, shares, order numbers, and listing market. They are separated by vertical bars. We first divided the aggregated dataset by symbols, which represent the stocks, we selected 4 stocks in the NYSE market, and stored them separately in

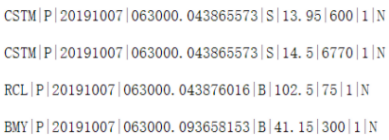


Figure 2: Raw data.

CSV format. After that, we select the columns we need, that are order type, price, and time.
We did the following processing:

1. **Time**

   Calculate the time difference between the current data point and previous data point.

   Categorized and one-hot encoded the time difference.

2. **Order type**

   Order type is categorical by itself. It only has two values, buy and sell. In a later time, the order type is embedded. We used an embedding of size 5, same as in the paper.

3. **Price**

   Calculate the relative price. Absolute price minus the median of all the prices.

   Categorized the relative price and turned them into one-hot encoding.

4. **Feature crossing**

5. **Time series**

   Combine the adjacent data points to form sequences with a window of length 50.

Label: 1 if the current price is higher than previous price, otherwise 0.

3.3 Model building

Based on the A-CNN given in the paper.

- **Input:** Sequence of equity trading records from time T-window size, to time T-1.

  i.e. t in the interval of [T-windowSize, T-1], where windowSize is a hyper-parameter used to control the length of the sequence.

- **Onput:** The probability that the stock price is going up at time T.

3.3.1 CNN

The main structure of CNN models is:

1. Embedding of order-type
2. (with/without) Average pooling
3. Convolutional layers (with/without inception module)
4. Softmax classifier

We tested models

1. with/without average pooling
2. with/without inception structures

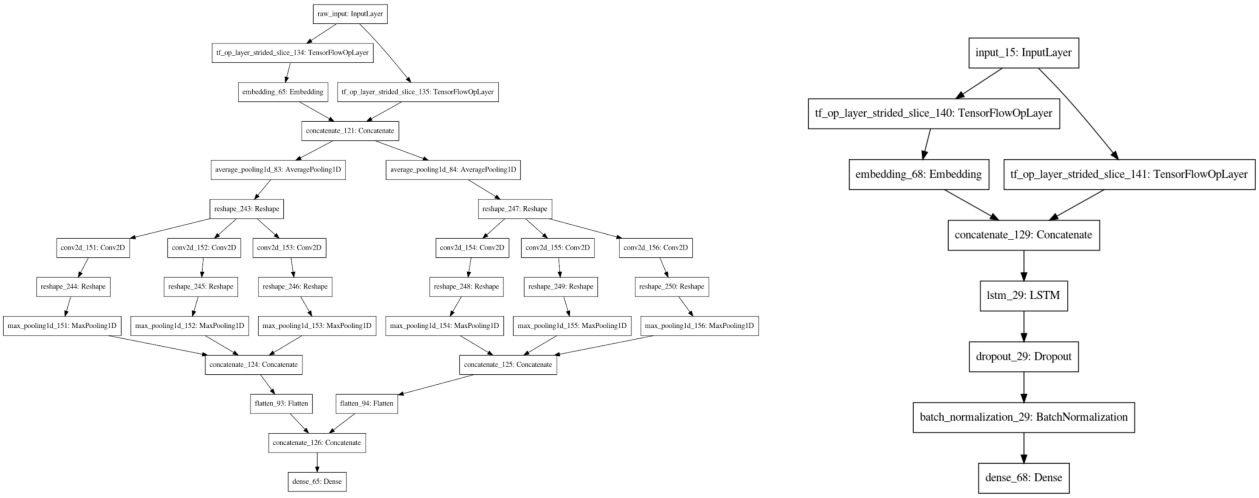
The most complicated model has two large branches, corresponding to different average poolings of size 5 and 10. Under the pooling in each branch, there are three sub-branches, corresponding to convolutional layers with 20 filters in each and having different kernel sizes. Then, the outputs for sub-branches are concatenated together, and the concatenated outputs for the two branches are concatenated again. And finally, they are fed to the output layer with softmax activations.

3.3.2 LSTM

1. Embedding
2. Several LSTM layers
3. Batch-norm and dropout layers used in between
4. Softmax classifier on the top

We tested models

1. with different layers
2. with/without average pooling layers



(a) CNN model (b) LSTM model

Figure 3: Models.

3.4 Evaluation

3.4.1 CNN

Test for Average Pooling

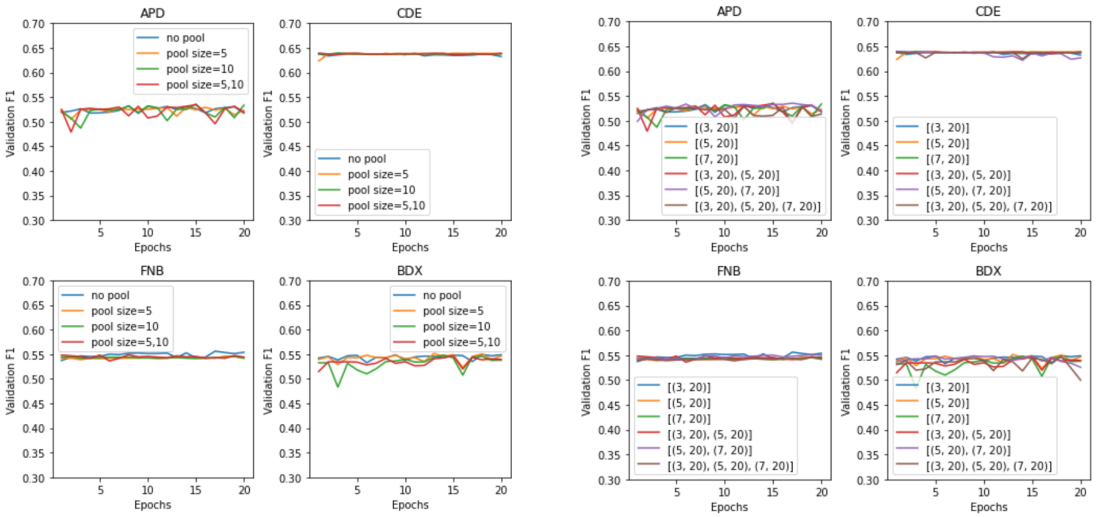
Let's test whether average pooling works on our dataset. The answer is perhaps not. The figure shows the validation accuracies for different stocks. In each subplot, the lines are corresponding to different poolings applied on each model. We tested four models 1) without pooling, 2) pooling of size 5, 3) pooling of size 10, and 4) apply two poolings of size 5/10 parallel.
Result: No big differences among different pooling strategies.
Possible reason: Our dataset only contains the market order data, which has a relatively high frequency. Therefore, applying average pooling to high resolution data is inappropriate. The results are showed in Figure 4.

Test for Inceptron Module

Next, we tested whether our parallel convolutional layers work or not. They are supposed to capture information in different resolutions. We use CNN without average pooling, because pooling has been shown to offer no good result.
Result: Small kernel size models work better than large kernel size models. Model with parallel convolutional layers has a performance similar to the model with small kernels.
Possible reason: Our dataset only contains high frequency orders, thus, larger kernels would miss important features and be less accurate than the small kernel. The model with parallel kernels can capture fine information thus its performance is similar to the small kernel models. The results are showed in Figure 4.

3.4.2 LSTM

LSTM models work surprisingly well on our dataset. It is much better than the CNN structure we used in previous experiments.
Result:

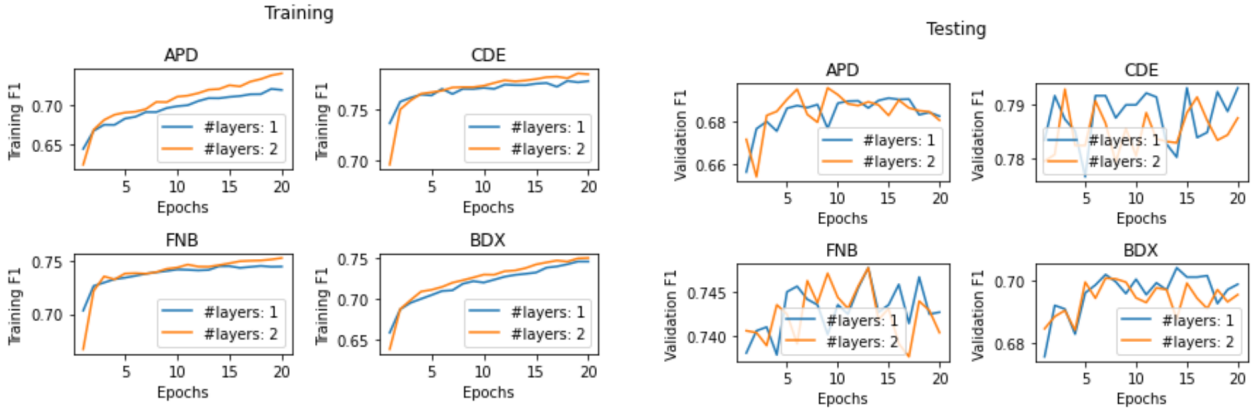


(a) Test for Average Pooling (b) Test for Inception Module

Figure 4: Evaluation.

1. **Training:** The model with 2 layers is more accurate than the model with a single layer.
2. **Validation:** The LSTM model with more layers is easier to be overfitted.

Possible reason: The model with 2 layers has a higher capacity so it can better fit the training data.



(a) Training (b) Validation

Figure 5: LSTM models.

The 1-layer LSTM model is better because it achieves similar accuracy as the 2-layer model and it is trained 30-40 seconds faster in 20 epochs on average.

4 Conclusion

In our experiment, we first did data processing, and built several models with different structures and compared their results.

For CNN models, we can achieve a 60% F1 score. However, average pooling and larger kernel sizes do not work, because our dataset only contains high frequency market orders, where low resolution information is not quite useful. Our parallel-kernel model can give similar performance as small-kernel models. It is expected to give better results on dataset containing both high frequency and low frequency data.

On the other hand, LSTM models can achieve 70-80% F1 score, which is much better than CNN models. For the same reason, average pooling does not work. We compared 1-layer model versus 2-layer model, and found the 2-layer model is easier to be overfitted. 1-layer model is better because it gives similar results and could be trained much faster. But both 1-layer and 2-layer models are much slower than CNN models, thus it could be a bottleneck in a real-world setting.

References

[1] 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, 19:9, 1499-1506, DOI: 10.1080/14697688.2019.1622314

[2] Bao W, Yue J, Rao Y. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PLoS One. 2017 Jul 14;12(7):e0180944. doi: 10.1371/journal.pone.0180944. PMID: 28708865; PMCID: PMC5510866.

[3] Dixon, Matthew. "Sequence classification of the limit order book using recurrent neural networks." Journal of computational science 24 (2018): 277-286.

[4] Cont, R., A. Kukanov, and S. Stoikov. "The price impact of order book events. arXiv. org Quantitative Finance Papers." (2011).

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

[6] NYSE TAQ Data