Supervised Deep Learning for Optimized Trade Execution

Hua Wanying, Long Zijie, Wang Kunzhen

April 19, 2019

1 Introduction

2 Literature Review

3 Model

In this project, we assume that the optimal execution strategy can be expressed as a pure function of the following 6 variables: t the remaining time before the end of the time horizon, i the remaining inventory to sell, the price level, price trend, limit order book volume mismatch as well as the bid-ask spread at the decision point. Following the convention in [1], we group the 6 input variables into two categories, i.e., the **private variables** consisting of t and i that is specific to the Optimized Trade Execution problem, and the **market variables** consisting of the rest of the four. Output of the model is represented by ***action***, the price at which to place a limit order. The model can be expressed mathematically as

$$action = f(t, i, price\ level, price\ trend, vol\ mismatch, bid\text{-}ask\ spread),$$

where f is an unknown function to be learned.

To estimate the function f, we develop a supervised deep learning model as described below. The model is implemented with *Tensorflow* and *Tensorflow Keras* provided by Google Brain, using *Python*. Implementation of the model can be found in the file *Model.py*.

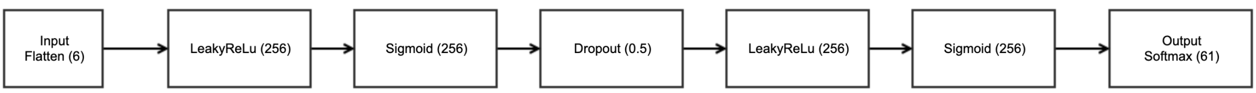

Figure 1: The Supervised Deep Learning Model

- **Input Layer** The input layer consists of simply the 6 parameters of the function f. Detailed definitions, rationales and extractions of these variables are provided in Section 4.2 and 4.3.

- **Hidden Layers** The model is composed of 5 fully-connected hidden layers with 256 neurons each. Activation functions for each layer is, correspondingly, *leakyReLu*, *sigmoid*, *dropout* with a rate of 0.5, *leakyReLu*, *sigmoid*. These activations are chosen after taking into consideration the nature of the problems. For example, noting the sparse activation characteristic of the *leakyReLu* activation and that the outputs are discrete, we chose *leakyReLu* to denoise the training process. Another advantage of the *leakyReLu* is its computational efficiency and ability to avoid dead neurons. The *sigmoid* activation is chosen for its ability to capture non-linear relationships. A *Dropout* layer is chosen in the middle to denoise and speed up the descent.

- **Output Layer** The output layer represents the predicted action given the input. The output variable, ***action***, is discrete for computational efficiency. Moreover, having a discrete output is important to avoid overfitting.

4 Model Training

4.1 Data Description

4.2 Market Variables

4.3 Private Variables

5 Results

6 Remarks

7 Conclusion

References

[1] Yuriy Nevmyvaka, Yi Feng, Michael Kearns. *Reinforcement Learning for Optimized Trade Execution*. Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA, 2006.