**Practical Deep Learning Approach for Intraday Futures Trading**

**Yuyuan Cui, Ziya Wang**

**Abstract**

This paper provides the modeling approach and preliminary results for using deep learning and reinforcement learning for developing systematic futures trading strategies in intraday timeframe.

**Introduction**

Deep learning has been a popular topic in quantitative financial trading. However, few studies have focused on applying deep learning approach to intraday and tick-level data for trading. As tick-level data provides a much larger dataset for training than daily data, and that various order placement types provide more options for action, there is plenty of opportunities to use complicated deep learning architecture for both prediction and strategy formulation. In this paper, we develop two independent models for China’s IH futures intraday trading based on tick-level high frequency data. The first model is an LSTM model that focuses on short-term price change prediction only. We will then combine the model prediction with a manually formulated simple trading strategy with take-profit and stop-loss thresholds to make trading decisions. The second model is a deep reinforcement learning model whose action space includes buy and sell decisions and hence will output an entire trading strategy on its own. On top of that, we will set up a passive market-making strategy as benchmark. By comparing their performance, we can evaluate the applicability of reinforcement learning and deep learning in intraday futures trading with different degrees of human experience intervention.

**Related Work**

Xiong, Liu, Zhong, Yang and Walid (2018) used deep reinforcement learning to obtain optimal strategy based on daily return data of 30 selected stocks. Xiao Zhong and David Enke (2019) used DNNs to predict daily ETF return based on 60 financial and economic features. Sezer, Ozbayoglua, and Dogdub (2017) used DNNs for optimizing technical indicators for daily stock trading and independently trained models for each of selected 30 stocks.

**Methods**

1. **LSTM for Mid Price Change Prediction**

Due to the sequential nature of time series order book tick data, we apply LSTM model (with XX hidden layers and [output design]) for mid price change prediction problem. The reason why we use price change as our prediction target as opposed to return is that there is no fundamental change in price scale (e.g. due to share split or drastic volatility regime shift) during the entire training and testing data sample, and that our trading horizon is constrained to be intraday. We train the model using XX algorithm. The NN architecture is illustrated as following:

补充画结构图

Where x\_t is the vector of following features as of time t:

wmid\_mid: difference between IH weighted mid and simple mid at time t

wmid\_last: difference between IH weighted mid and last traded price

mid\_lag\_T (T= 1tick, 01s, 05s, 01m, 05m, 10m): difference between current IH mid price and its mid price T time unit before

wmid\_ma\_T (T=05min, 10min): difference between current IH weighted mid and its moving average in past T look-back interval

wmid\_max\_T (T=05min, 10min): difference between current IH weighted mid and its max value in past T look-back interval

wmid\_min\_T (T=05min, 10min): difference between current IH weighted mid and its min value in past T look-back interval

wmid\_bidask\_10min: difference between current IH weighted mid and its 10-minute VWAP (volume-weighted price)

total\_volume\_T (T=10s, 10min): total IH trading volume in past T look-back interval

signed\_volume\_T (T=10s, 10min): trade-direction signed IH trading volume in past T look-back interval

signed\_tick\_T (T=10s, 10min): number of up price changes minus number of down price changes in the past T look-back interval

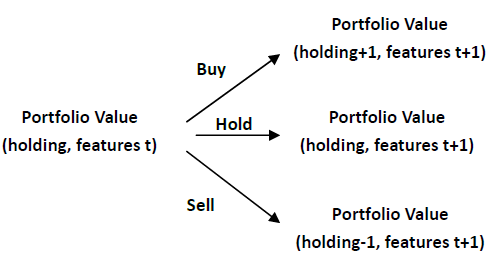
IF\_mid\_lag\_T (T=01s, 05s, 30s, 01min, 05min, 10min): difference between current IF mid price and its mid price T time unit before

IC\_mid\_lag\_T (T=01s, 05s, 30s, 01min, 05min, 10min): difference between current IC mid price and its mid price T time unit before

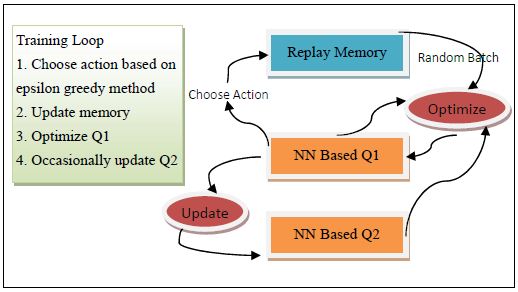
Compared with previous research that only focus on daily returns as predictive features, we take advantage of the most granular level of tick-by-tick data (with average update frequency of less than one second), and construct a wide range of features that encompass information in order book imbalance, trading volume, trading direction, technical time series patter, and cross-asset returns. The reason why we choose not to include cross-asset trade or book information is because based on out-sample testing, the additional predictive power to that already included in cross-asset price change is not meaningful and hence we prefer a simpler model for robustness consideration.

1. **Deep Reinforcement Learning for Intraday Trading**

As a trading strategy’s action space can be conveniently described as either buy, hold or sell, we can train a Deep Q Learning (DQN) model to achieve end-to-end training for a trading model.



To characterize the state space, we will use the same features in the LSTM model for comparison purpose. In addition, we add current futures position holding as feature describing the environment. For simplicity, we restrict actions to be either long one additional contract, short one additional contract or no change in position held. As a result, the DNN’s output should be three-dimensional to reflect estimate of value of these three actions. Furthermore, to facilitate calibration, we use market value of position holding 15 minutes since initial setting up a position as “terminal state”. Due to high degree of noise commonly seen in financial data, we choose a Double Q-learning approach to improve robustness and alleviate the risk of overestimating action values. Specifically, we will use the following iterative approach to update the two DNN-based Q(st, at) function estimates, with a cyclic buffer for memory replay.



**Preliminary Results**

1. **Data**

The dataset we use is tick data of IH futures (SSE 50 Stock Index Futures), IC futures (CSI 500 Stock Index Futures) and IF (CSI 300 Stock Index Futures) futures tick data in China’s CFFEX exchange in 2018. The data include all book updates and trade events. The morning trading session is 09:30 AM - 11:30 AM local time, while the after session is 01:00 PM – 03:00 PM local time. We treat data in any look-back period prior to session start (e.g. 9:20 AM or 12:30 PM) as NaN but otherwise treat both sessions equally. For each futures, we will use front-month contract (contract that expires in the next month). As maximum tick frequency is around 0.5 seconds for all these futures, we regularize all tick data with resample frequency of 0.5 seconds. For missing data, we fill last traded price, bid/offer price and size with most recent valid value; and fill traded volume with 0. For back-testing setup, we always close all positions whenever a trading session ends. The model training period is xxx, and the testing period is xxx.

1. **Model Performance** OLS未必用全部的x feature columns, Save y\_hat for trading strategy)

We evaluate the LSTM model using mean-squared error (MSE) of predicted compared with actual [mid or weighted mid] price change in the testing dataset. Model performance is compared with OLS using same set of features as benchmark.

|  |  |  |
| --- | --- | --- |
|  | In-sample MSE | Out-sample MSE |
| LSTM | xx | xx |
| OLS | xx | xx |

Next, we compare the performance of a simple trading strategy guided by the LSTM with the benchmark passive market-making strategy. The LSTM-based strategy places bid order only when the model predicts the price will move up [by xx], places ask order only when the model predicts the price will move down [by xx], and doesn’t not place any order otherwise. The benchmark passive strategy, however, always places both a bid and an offer order. We restrict both strategy to hold a maximum of one contract at a time. The cumulative profit-and-loss (PnL) plot is as follows:

补充pnl图

As our next step, we will evaluate deep reinforcement network model through back-testing performance and comparing it with benchmark passive market-making strategy.

1. **Analysis**

The LSTM model has greater flexibility in capturing both the sequential dependency and complex inter-feature interaction of the high dimensional feature time series. As a result, both in-sample and out-sample MSE are significantly lower than OLS. The market making strategy guided by the model also outperforms purely passive (uninformed) trading strategy. The benchmark strategy trades more frequently and often suffers from adverse selection due to its passive mechanism, and can experience large drawdown even in an intraday horizon. In comparison, the LSTM-based model is more cautious in entering into positions and only trades when it has confidence in predicted direction.

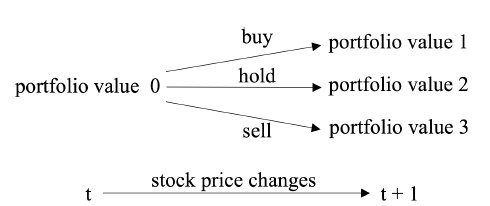
**References**

Sezer, O. B., Ozbayoglu, M., and Dogdu, E. A deep neuralnetwork based stock trading system based on evolutionary optimized technical analysis parameters. Procedia Computer Science, 114:473 – 480, 2017. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2017.09.031. URL

http://www.sciencedirect.com/science/article/pii/S1877050917318252. Complex Adaptive Systems Conference with Theme: Engineering Cyber Physical Systems, CAS October 30 – November 1, 2017, Chicago, Illinois, USA.

Xiong, Z., Liu, X.-Y., Zhong, S., Yang, H., and Walid, A. Practical deep reinforcement learning approach for stock trading, 2018.

Zhong, X. and Enke, D. Predicting the daily return direction of the stock market using hybrid machine learning algorithms. Financial Innovation, 5(1):24, Jun 2019. ISSN 2199-4730. doi: 10.1186/s40854-019-0138-0. URL https://doi.org/10.1186/s40854-019-0138-0.



**Portfolio Value**

**(holding+1, features t+1)**

**Buy**

**Hold**

**Portfolio Value**

**(holding, features t)**

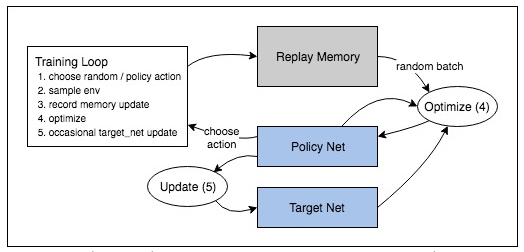
**Portfolio Value**

**(holding, features t+1)**

**Sell**

**Portfolio Value**

**(holding-1, features t+1)**



Training Loop

1. 1. Choose action based on epsilon greedy method
2. 2. Update memory
3. 3. Optimize Q1
4. 4. Occasionally update Q2

Random Batch

1. Replay Memory

Choose Action

NN Based Q1

NN Based Q2