Capstone Project Proposal for Training a Stock Price Prediction Model with Neutral Network

Jianming Pan

August 2022

1 The Domain and Background of the Project

Stock price prediction is always a challenging but attracting task. Various techniques have been tried to conquer the market, but few succeed. The key component for stock price prediction task is to fit a conditional distribution function of stock returns in the context of historical data. And then input updated information to make a prediction of stock's return and risk. Based on estimated return and risk, we also need to utilize portfolio optimization to allocate resources on making investment decisions.

The difficulty of such tasks mostly lies in the low signal-to-noise-ratio (SNR) of stock market. Day-by-day transaction in the busy market generates huge amount of data, which is prefect for modern machine learning method. However, it usually takes great effort to extract the valuable part of such data. In this capstone project, I will built a Multi Precision Layer prediction model on the predefined features of stocks. By combining optimization method, the final model could output exact investment decision of any asset pools that customer interested.

2 The Problem Statement

As mentioned before, our major task is to train a model(function) $F : R^{n\times m} \to R^n$, mapping n stocks with m features to their returns. To be clear, predicting stock price is the same as to predict stock return. Because we could choose a start time and set the stock price as P_0, and then at any time later than the start time. The price of stock is $P_0(1 + r)$, where r represents the return in the interval time. By this transformation, our data distributed more like independently.

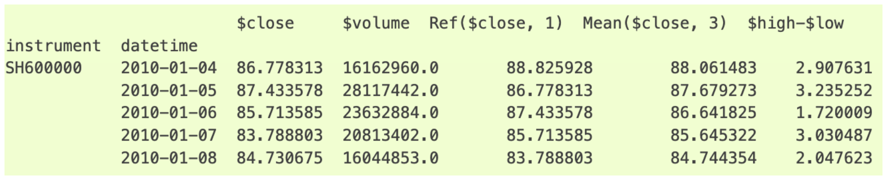
3 Dataset and Inputs

The data used in this project is from the open source Microsoft Qlib Dataset. One can use "pip install pyqlib" command in python console to obtain the qlib.

And paste the following command in the terminal to download the dataset.

python scripts/get_data.py qlib_data –target_dir /.qlib/qlib_data/cn_data –region cn

The dataset named Alpha158 contains 158 pre-calculated features of corresponding stock in China's stock market from 2010-2020. The dataset has two indexs ¡datetime, instrument¿ and 158 columns of features. Features are defined as transformations of stock Close/High/Low/Open price and volume. For example, a feature could be made as the average past 3 days' close price.



```
                          $close     $volume  Ref($close, 1)  Mean($close, 3)  $high-$low
instrument  datetime
SH600000    2010-01-04  86.778313  16162960.0       88.825928        88.061483    2.907631
            2010-01-05  87.433578  28117442.0       86.778313        87.679273    3.235252
            2010-01-06  85.713585  23632884.0       87.433578        86.641825    1.720009
            2010-01-07  83.788803  20813402.0       85.713585        85.645322    3.030487
            2010-01-08  84.730675  16044853.0       83.788803        84.744354    2.047623
```

Figure 1: Example of Alpha158

4 Solution Statement

I will construct a MLP model with one hidden layer to fit the conditional distribution of stock returns. The input of the data is randomly chose from a cross-section normal samplers, which could best average the distribution drift between batches and also retain the datetime feature of the data. The output of the model is the estimated return of each stock with a MSE loss function.

5 Benchmark Model

Traditional stock price prediction task uses a linear model to integral all stock features. So the benchmark of our model is the linear regression model.

6 Evaluation Metrics

For all investment decision, backtest is the best way of determining which model performs better. We will make a backtest program to simulate the true investment environment. As for the metrics, annualized return (AR), sharpe ratio (SR), IC, IR, Maxdrawdown, and net value of simulated portfolio is all we need.

$$AR = \sqrt{\prod_{i=1}^{T}(1+r)^{i}}^{\,T/252} \tag{1}$$

$$SR = \frac{AR}{\sigma(r)} \tag{2}$$

$$IR = SR = \frac{AR - benchmark}{\sigma(r)} \tag{3}$$

$$Maxdrawdown = \max_{i<j}(r_i - r_j) \tag{4}$$

7 Project Design

The aim of this project is to train a model that helps investor make decisions on the stock market. By applying Machine Learning method, we could outperform the traditional method. To brief summary, the project is designed as follows:

(1) Load the Alpha158 from qlib. All data imported as time-series data with pandas DataFrame type. Spilt train set, valid set and test set.

(2) Built dataprocessor.py to process data before training. Make sure the process procedure is different in stage of learning and inference. Because we do not want to introduce future data in the inference stage, while in the learning stage, normalizing all distribution in each batches requires us to mix data together. Also, cross-section norm and NAN data should be included in this script.

(3) Determine the hyperparameters of our model, specifically including batch_size, learning rate, epoch. The net structure of the model is simple but powerful.

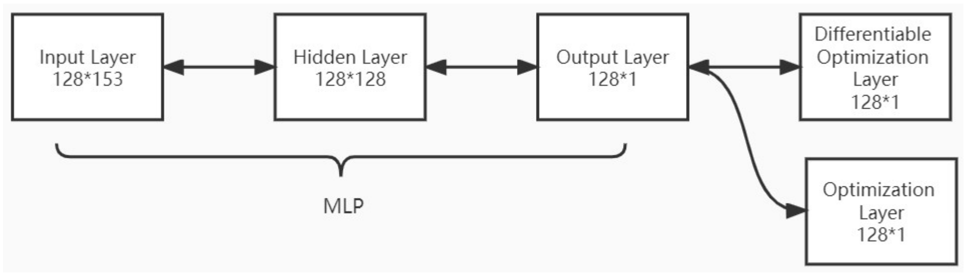


Figure 2: Net structure of the model with batch size of 128

(4) Basically, I will train 3 models. 2 listed above are Machine Learning model. And one for linear model as benchmark. The difference of above 2 models are the optimization layers. Optimization is the last part of investment decision. We will attain a prediction on tomorrow's stock price, but we need to decide how to allocate our money to buy them. For example, we cannot but the one that rises most according to prediction. Because it is risky, all outputs of stock-related information are random variables. So we should think carefully and make a trade-off between risk and return.

Traditionally, people uses a optimization method to determine which stock they would buy based on the information of prediction model. While this optimization method cannot fit in the procedure of machine learning because it is not differentiable. (Math Analysis would be included in the report) However, Agrawal, et al. (2019) came up with a convex problem solver called DPP, which makes the optimization differentiable. And I want to utilize this new method to modify existed MLP model.

(5) After training, I will evaluate all this models' performance in backtest.py, which would compute the metrics listed above and give a fair comparison between all 3 models.

(6) At last, we do hope to make a user-friendly interface to help interested investors make investment decision. I will upload all code to GitHub. Users may download their own data to train their models under the guidance of this project.

8 References

[1] https://github.com/microsoft/qlib.git

[2] Agrawal A, Amos B, Barratt S, et al. Differentiable convex optimization layers[J]. Advances in neural information processing systems, 2019, 32.