Summer Research Program in Industrial and Applied Mathematics

Hong Kong 2018

Final Report

A T+0 Stock Trading Strategy Based on Machine Learning Techniques

Sponsor

Charles River Advisors (Hong Kong) Ltd

Industrial mentor

Dr. Hongsong Chou

Academic mentor

Hao Liu

Team members

Chan Hee Cho

Keunwoo Lim

Yuanqiao Lin

Qingyuan Zhang

1 Introduction

Secondary market trading undoubtedly takes up a crucial role in modern financial markets, where exchange-based transaction activities happen multiple times throughout one trading day. That is to say, stocks or indices, namely "liquid products" can be bought and sold again and again for traders' interest. To make maximum profit out of such financial activities, we try to gain as much trading information from the "order book", a snapshot that displays market supply and demand at all times, as possible. To cope with the substantial amount of data, the renowned machine learning techniques are introduced. The ultimate goal is to construct a model that could predict the trend of the market and maximize profit with certain accuracy.

In this report, we handled 'liquid products' in the secondary market. Financial trading activity handling such product is often called T+0 trading. ("T+0" means completing the entire trade verification process on the same day.) We constructed a forecasting model in this market.

There were lots of information affecting the price of each asset. Some studies followed the perspective that such environmental conditions are reflected in each asset price and built model from the history of price. However, little research focused on precise market data features. In this report, not only the price of each asset, we processed various kinds of 'high frequency data'. One of the crucial part for predicting future price is summarizing market condition. We did this work by detecting certain 'features' from the market history.

With five features provided by our industrial mentor, we used four machine learning models, i.e. logistic regression, random forest, convolutional neural network and fully connected neural network. In addition, we calculated the accuracy and efficiency of each model.

2 Construction of features and labels (market condition summarization)

2.1 details about features.

- SRL: Support and Resistance Level

From the past 15 minutes mid-quote of each tick, we calculate the top 95% and bottom 5% cut-off values and use them as two threshold values. And SRL is calculated by the difference between the midQ at time t and the average of two threshold values over the difference between two threshold values. This feature represents the current price's relative position. In normal situation, this value is between -0.5 to 0.5 which means the midQ at time t is between threshold values. If this SRL value is not between -0.5 and 0.5, this will be a strong signal called 'break-out'.

- RTS: Rounded Trade Size

RTS1 = 1 if the trade that happened immediately before t is of a size that is a multiple of 1000 (such as 1000, 2000, 3000, etc.) but not of 5000 and 10000
RTS2 = 1 if the trade that happened immediately before t is of a size that is a multiple of 5000 (but not of 10000)
RTS3 = 1 if the trade that happened immediately before t is of a size that is a multiple of 10000.
RTS1/2/3 = 0 otherwise.
We think this feature can indicate the appearance of institutional investors

-TAI: Trade Arrival Intensity

Define the number of individual trades that happened between the most recent two snapshots immediately before t as N_t, and the average/stdev of N_t between T-Dt and T is Navg/Nstd. Then, this feature is calculated as (N_t-Navg)/Nstd. Dt is usually about 15 minutes or even longer.
We hope that this feature can capture the sudden increase or decrease in trading activity for the

current time point, T.

-OQD: Order Queue Distribution

Assume that the number of limit orders on bid1 (or ask1) at time t is Nb (or Na), and the total number of shares on bid1 (or ask1) at time t is Vb (or Va). Define the average queued order size at time t as OQD1b = Vb/Nb on bid1 and OQD1a = Va/Na on ask1. In a similar way as we defined TAI above, we calculate (OQD1b_t – OQD1b_avg)/OQD1b_stdev and (OQD1a_t – OQD1a_avg)/OQD1a_stdev as two features in the group of "Order Queue Distribution".

-CmpSize: Compared Sized

Define the volume of ask1 and the volume of bid1 as Va and Vb. If Va is relatively bigger than Vb, there is high probability that the price will go up. Compared Size is made to capture this movement. Compared Sized is calculated by (Va – Vb)/(Va + Vb). In our project, Compared Size showed the highest correlation with the label (logarithmic rate of return).

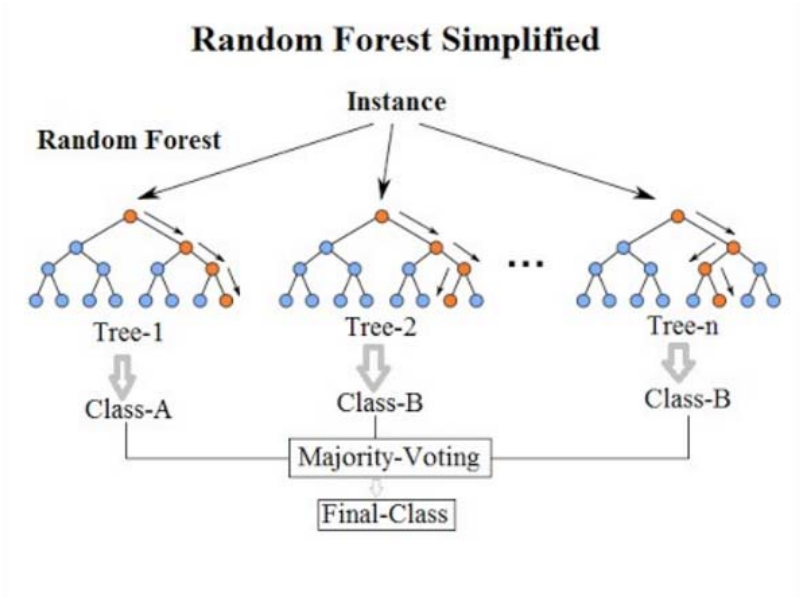
2.2 Detail about Label: logarithmic rate of return

Our label is the logarithmic rate of return. In detail, define the average of midQ at time t+1 to t+10 as midQ_avg. Then the label is calculated by taking natural log to (midQ_avg/midQ_t).

These features and the label are calculated at each tick and we used this data to train our forecasting model.

3 Machine Learning Models

3.1 Random Forest Model Approach



Random Forest model is an ensemble of decision trees.

In decision tree, first we calculate entropy by **H(S)=−∑p(y_i)log2p(y_i)** where p(y_i) is the proportion of the number of elements of y_i class to the number of entire elements in output of S. H(S) tells us how uncertain our dataset is. Then, we calculate Information Gain by **IG(A,S)=H(S)−∑p(t)H(t)**. Information Gain is computed separately on each feature of the current dataset S, whose value indicates how much the uncertainty in S was reduced **after** splitting S using feature A. After compute all the Information Gains of all features, we will then split the current dataset S using the feature which has the highest Information Gain.

In random forest, after we built many decision trees, we use majority voting to get the final decision. One way to avoid overfitting in random forest is to restricting 'maximum depth' in each decision tress in the random forest. In this project, we used this method to prevent overfitting in random forest.

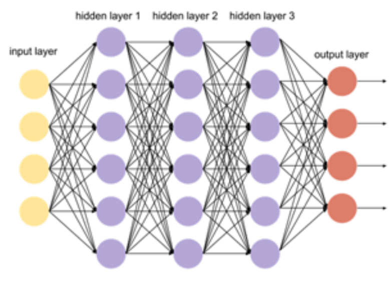
3.2 Logistic regression model approach

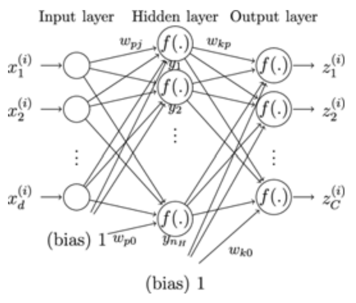
It is similar to linear regression, but its cost function uses logistic function. We used 20 days of previous data to train our models. We used the parameters (x data) based on 9 features, and y data as label. First, we tried to classify the y data into two classes and marked as 1 and 0. Over 75 percent values are marked as 1, and the others are marked as 0. It used 13 features. 9 features are from stock data, and we chose four parameters based on correlation between 9 features, features' statistical property and practice. Its average score was 0.76. And its average accuracy was 0.51. The average number of predicting 1 as 1 is 238. Second, we tried 3 classes, under 25 percent value, over 75 percent, and the middle one marked as -1,1,0. It used 11 features: 9 features are from stock data, and 2 more parameters by same reason.

3.3 FCNN- Fully connected neural network approach

Artificial neural networks (ANNs) are inspired by the biological neural networks, with nodes in the network connected to nodes in different layers, receiving and processing signals and eventually transmitting to the next node. The massive net could then process information like a human brain and perform exceptionally in a variety of tasks, such as computer vision, speech recognition and machine translation.

Fully Connected Neural Network serves as a fundamental model for neural network, and to our understanding, the significant characteristic of fully connected neural network lies in that each node in one layer of the network is connected to all the nodes in the previous layer.





The mathematical computation of the model appears straightforward. Each input node is represented as a number x_i, and each node in other layers is composed of a weight w_i and bias b_i, $w_i x_i + b_i$. Eventually, the output layer is constructed. When training this model, we start from the inputs. And then each weight and bias connecting the nodes are modified so as to optimally match the training input and training output. This is called forward propagation. It is equally feasible to use backward propagation, that is, to begin with the outputs and consider the partial derivatives with respect to weight, bias and inputs.

Our group has implemented fully connected neural network using tensorflow in two ways, defining the neural network ourselves and making use of the function tf.contrib.layers.fully_connected(), which allows users to input all the information for a neural network. We tried to train models for 2-class classification and 3-class classification. As mentioned above, we had seven features as input to the neural network, and the first method was to implement multi-class classification, to classify outputs (the price of the stock) into >mean+std (buy signal), <mean-std (sell signal) and between these two (stay signal). For 3-class models, we emphasized on the precision rate, which would indicate the level of correctness of the predictions (eg. for buy signals, the precision rate gives the percentage of buy signals given by the model that matches the real market signal). We have attempted to standardize the input features, initialize the weights. We tried different models for weight and bias initialization and optimized the learning rate. And eventually, the precision rate for buy signal is around 60% and for sell signal is around 50%. For 2-class classification, which predicted the price going up or down, yielded around 50% accuracy.

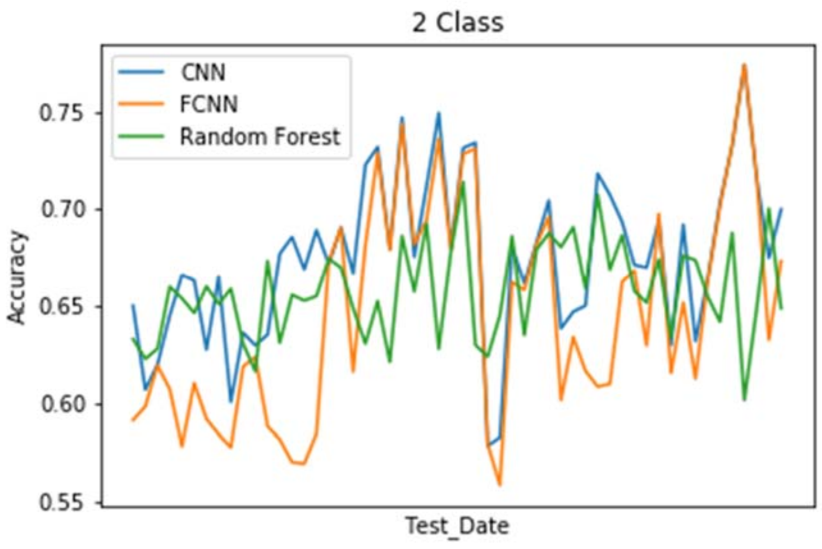
3.4 Convolutional Neural Network (CNN) approach

CNN consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers and normalization layers. What is different from Fully Connected Neural Network is it is not fully connected and using convolutions and layers. It is usually used to analyze image, but it can be applied to our models.

4 Results

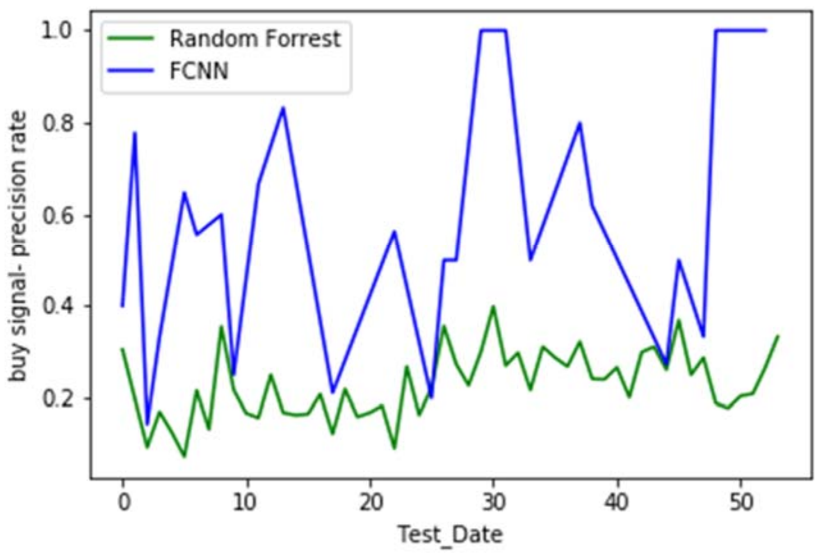
4.1 2-Class Models

In 2-class model, x-axis is the date of 3 months and y-axis is the accuracy of three models. The definition of accuracy is rate of predicting buy sign as buy sign. The result is, accuracy of CNN is 0.67 and accuracy of FCNN, Random forrest is 0.65.



4.2 3-Class Models

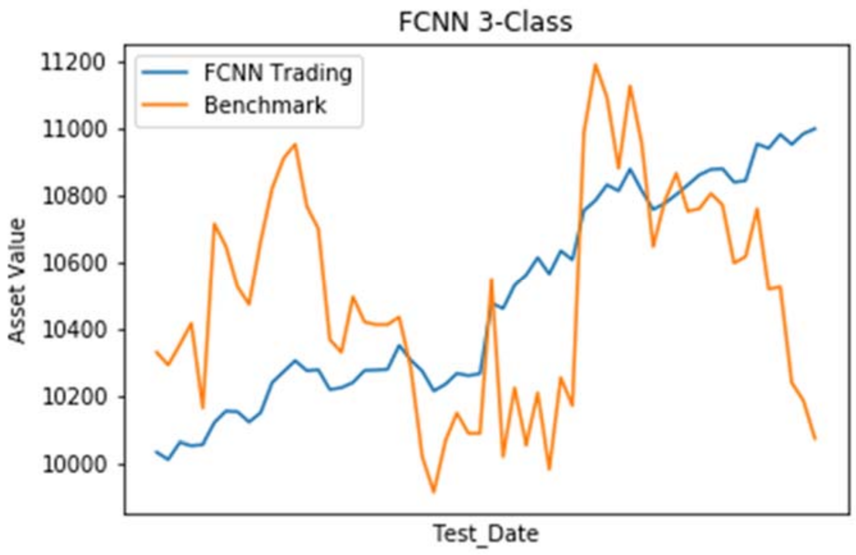
In 3-class model, x-axis is the date of 3 months as well and y-axis is the precision rate of two models. The definition of accuracy is rate of predicting buy sign as buy sign. The result is, precision rate of FCNN is bigger than random forrest, and standard deviation as also.



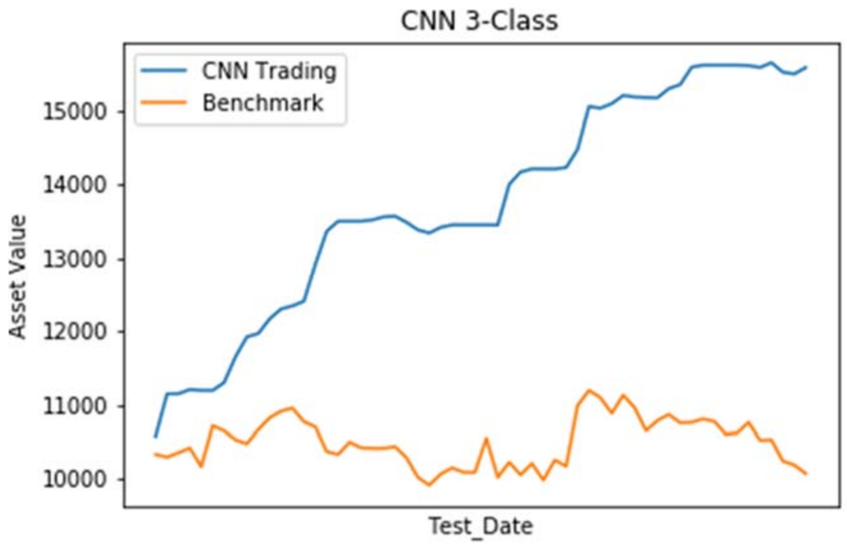
4 Trading Simulation

With the 3-class model, we could get 3 signals: ‘buy’, ‘stay’, ‘sell’. With these signals we made a simple trading bot. The algorithm is simple: buy the stock when the signal indicates that the price will go up, and sell the stock when the signal indicates that the price will go down. In this simulation, we assumed zero transaction fee, zero market impact, zero slippage. Since this simulation didn’t considered transaction fee, this may not appropriate for real trading action. However, this trading results clearly implies that the forecasting model is working properly and this model is quite useful.

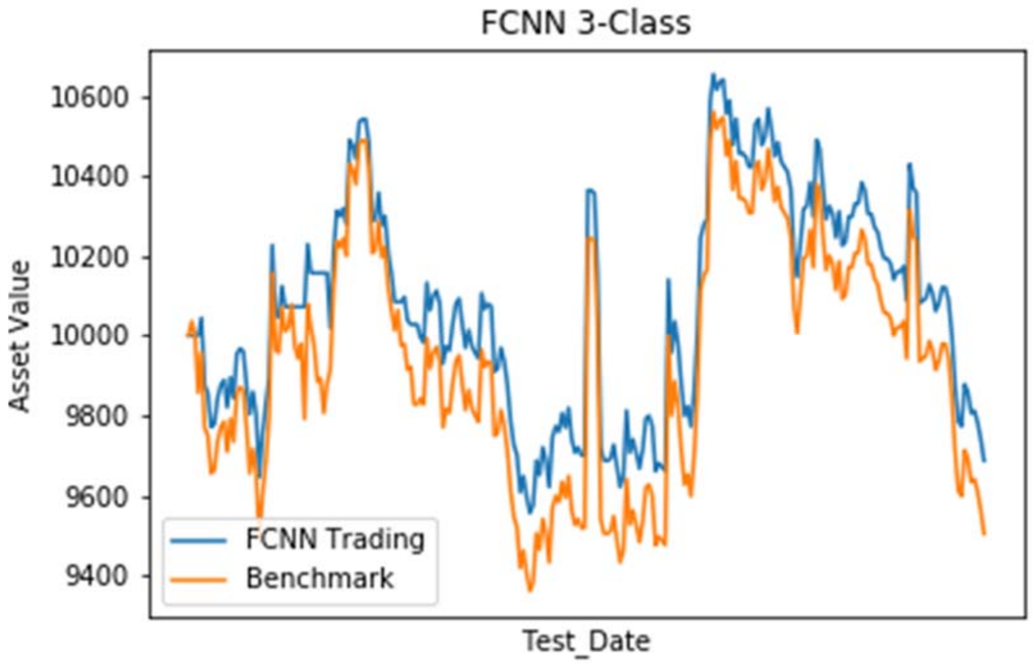
The benchmark is the ‘buy and hold’ strategy which means the stock price. First figure is the trading result using the signals made by the FCNN model and the second figure is the trading result using CNN model.



,



In the second simulation, we considered transaction fee. Since the former one traded very frequently, it was not appropriate with the existence of transaction fee. We tried to lower the percentage of 'buy sign' and 'sell sign'. When we look at the third figure, it looks quite similar with benchmark. But in detail, the model could catch some downtrends and avoid holding the stock. By doing that, the trading bot could beat the benchmark. Even if this work is not that appropriate to earn money itself, it can be used to maximize profit when you decided to hold certain stocks for your investment activity.



6 Future work

We would like to achieve better result using a more complex model in the future. One approach we would like to implement would be Deep Direct Reinforcement Learning model for representing dynamic financial markets introduced by researchers. Initially, Direct Reinforcement Trading is used to incorporate long-time memory into the system to determine the trading policy at time t. It takes into account the return f (price difference between time t and previous time t-1), the decision policy at time t-1, and market features w, b and u. It forms a one-layer Recurrent Neural Network with parameter set {w, b, u}. The next step is to let the neural network learn about market features. This is accomplished with deep learning with a nonlinear mapping on f. The final step aims at reducing uncertainties using fuzzy extensions. Each dimension of the input vector f is assigned with 3 fuzzy degrees. To our understanding, the researchers transformed each input node with 3 membership functions to 3 nodes that were fed to the next layer.

7 BIBLIOGRAPHY

Deng, Y., Bao, F., Kong, Y., Ren, Z. and Dai, Q. (2017). Deep Direct Reinforcement Learning for Financial Signal Representation and Trading. *IEEE Transactions On Neural Networks And Learning Systems, 28*(3), 653-664.

Jiang, Z., Xu, D. and Liang, J. (2017). A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem. Retrieved from: https://arxiv.org/abs/1706.10059