Trading with Deep Learning
- Comparing Techniques
used for Predicative Modelling for Stock Market

Tianxin Li
ltianxin@edu.uwaterloo.ca
University of Waterloo

CS489 Project - Option A Literature Survey

Abstract

This paper aims to explore model-free reinforcement learning techniques for trading stock as compared to traditional stock trading techniques such as using mathematical models or machine learning models to forecast stock prices with manually designed trading policy. Reinforcement learning is used to create optimal trading strategy. The comparison showed that model-free learning technique has potential to vastly outperform traditional stock trading policies and are worth further investigated.

CS489 Project - Option A Literature Survey

Trading with Deep Learning
- Comparing Techniques
used for Predicative Modelling for Stock Market

The advancement in machine learning in recent years has seen many successes in application in stock trading. Deep neural network such as Long-Short Term Memory (LSTM), Gated Recurrent Unit (GRU) and WaveNet etc. enabled the development of effective prediction models for time-series stock data. The recent breakthroughs in Artificial Intelligence (AI) has also sparked applications of reinforcement learning technique in many fields. Such as Learning to Play Atari by DeepMind and Google's famous AlphaGo. There is a lot of potential in teaching AI to exploit vast amount of information in complex environment to learn the best path. Quantitative stock trading is a very well-defined environment for such technique with clear defined reward and relatively simple state transitions.

Introduction

Traditionally, quantitative stock trading is centered around the development of mathematical representation of the stock price movement, constructing trading policies using back-testing to validate hand developed trading policies. The two main steps of optimizations involved in this process are: firstly, optimizing the accuracy of time-series forecasting of stock price movement; secondly, the validity of trading policy that maximize profits and minimize risks by exploiting price predictions.

Though such process has proven to be effective, there are many drawbacks. In particular, there are two drawbacks that we want to investigate and hope that could be addressed in this research. Firstly, the two-step process involves lengthy back-testing, a method commonly used to test how such model works with historical data and real-world live data. This can take a large amount of time in model development. Should the model underperform, researchers iterate over this process to improve existing model or develop new models. Secondly, the second step, developing trading policies can be tedious and very specific to a particular stock or price prediction. Not only this could subject to human biases and errors, it can be time-consuming for development and modification, let alone finding the optimal strategy. Moreover, it could subject to changes responding to volatile market environment and it is extremely difficult to device optimal trading policies that reacts to complex changes. Thus, the various errors and biases introduced by human in the prevalent process for developing stock trading algorithm could be vastly improved with reinforcement learning by using algorithms to learn to interact optimally given available market information.

Background

Machine Learning Models

Logistic Regression is one of the most commonly used machine learning models for supervised classification tasks. They are relatively quick to optimize and train, do not require very large dataset as compared to deep neural networks and offers greater model transparency. It is trained to output binary output of the price of the stock either going up or down given financial information of the company and historical stock price data in sliding windows.

Recurrent Neural Networks

Recurrent Neural Networks (RNNs) and its various adaptations has been widely used in sequence modelling. Due to its advantage in that it could theoretically capture long-term dependencies make it especially useful for time-series data regression and classification such as stock price prediction. RNN does so by recursively apply activation functions on state-to-state transitions so that input sequence information could be carried forward in hidden states. Illustrated below where h_t and h_{t-1} represents hidden states and φ represents non-linearity given input x_t:

$$\mathbf{h}_t = \phi\left(W\mathbf{x}_t + U\mathbf{h}_{t-1}\right),$$

However, training an RNN to accurately accomplish such tasks could be very difficult due to the well-known problem of vanishing/exploding gradient. The recent development in RNN involves modification over RNN architecture to retain 'memory' of long-term and short-term dependencies of data. The most popular modifications include Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU) and Gated Feedback Recurrent Neural Nets (GF-RNN). The motivations are similar, using gated activation functions to replace RNN's state-to-state affine transformation, in this way, the RNN is able to better maintain its gradient being back-propagated. The memory cell C_t of LSTM is illustrated below mathematically. It carries memory information from previous cell and input from new information through input gate i_t^j and forget gate f_t^j, and finally an output gate determining the memories active for the prediction.

$$c_t^j = f_t^j c_{t-1}^j + i_t^j \tilde{c}_t^j,$$

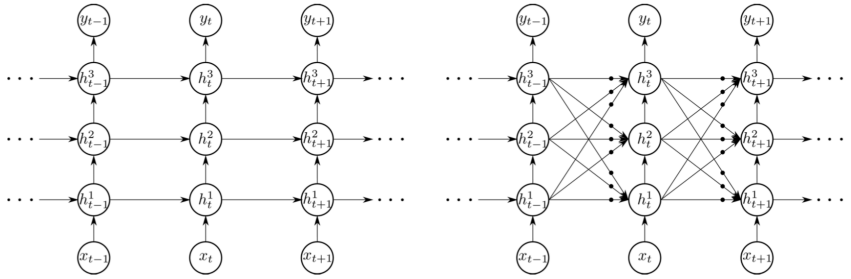
where

$$\tilde{\mathbf{c}}_t = \tanh\left(W_c\mathbf{x}_t + U_c\mathbf{h}_{t-1}\right).$$

$$\mathbf{i}_t = \sigma\left(W_i\mathbf{x}_t + U_i\mathbf{h}_{t-1}\right),$$
$$\mathbf{f}_t = \sigma\left(W_f\mathbf{x}_t + U_f\mathbf{h}_{t-1}\right),$$

$$h_t^j = o_t^j \tanh\left(c_t^j\right).$$

GF-RNN is another attempt to solve the problem of learning multiple adaptive timescales by introducing gated-feedback connections between different layers of stacked RNNs' hidden states, making the architecture able to adaptively learn different timescale with each layer at the same time.



(a) Conventional stacked RNN (b) Gated Feedback RNN

Figure 1 GF-RNN

Reinforcement Learning

Reinforcement learning (RL) has been developing rapidly in the past few years with integration with deep neural networks. It involves an agent exploring the environment through trial and error. The environment has many state and associated rewards / penalties associated with the states, by exploring the environment space, the agent learns the optimal strategy to react to environment. Q-learning is a model-free reinforcement learning method that learns expected utility of actions from state through Markov Decision Process (MDP). It finds the optimal policies that maximize such expected utility. It can be viewed as a very experienced trader learning from millions of trading experiments he went through using various trading policies he came up with either through heuristic algorithms, market knowledge or random selection, he then exploits the strategy that has consistently perform the best for him to maximize profits.

Formally, Q-learning updates the state, action pair Q_{opt} with reward r and next state V_{opt}.

$$\hat{Q}_{opt}(s,a) \leftarrow (1-\eta)\hat{Q}_{opt}(s,a) + \eta(r + \gamma \hat{V}_{opt}(s')$$
$$V_{opt}(s') = \max_{a' \in Actions(s')} \hat{Q}_{opt}(s',a')$$

Where the optimal policy is thus,

$$\pi_{opt}(s) = \arg \max_{a \in act(s)} Q_{opt}(s,a)$$

Q-learning has two inherent advantage in rectifying the problem outlined in introduction. Firstly, it automatically adapts to new market environment should the agent no longer rewarded for the actions he took, thus reducing the need for tedious back-testing process. Secondly, it is 'curious', and sometimes explores the environment, therefore it learns to generalize trading strategies that work well and are likely to converge to an optimal strategy given the large number of trails and errors it can execute, thus covering a larger environment space.

Experiments

Data Exploration

The stock data used in this experiment consist of Standard & Pools 500 daily opening, closing, high, low prices and volume from 2013 February to 2018 February, SEC 10K annual filing of public traded company, including financial information such as gross profit, capital expenditure, liabilities etc. Reddit general market news data is also crawled and tested, however it did not improve model accuracy, more careful feature extraction may be needed to work well.

	Ticker symbol	Security	SEC filings	GICS Sector	GICS Sub Industry	Address of Headquarters	Date first added	CIK
0	MMM	3M Company	reports	Industrials	Industrial Conglomerates	St. Paul, Minnesota	NaN	66740
1	ABT	Abbott Laboratories	reports	Health Care	Health Care Equipment	North Chicago, Illinois	1964-03-31	1800
2	ABBV	AbbVie	reports	Health Care	Pharmaceuticals	North Chicago, Illinois	2012-12-31	1551152
3	ACN	Accenture plc	reports	Information Technology	IT Consulting & Other Services	Dublin, Ireland	2011-07-06	1467373
4	ATVI	Activision Blizzard	reports	Information Technology	Home Entertainment Software	Santa Monica, California	2015-08-31	718877

Figure 2 Sample Company info data

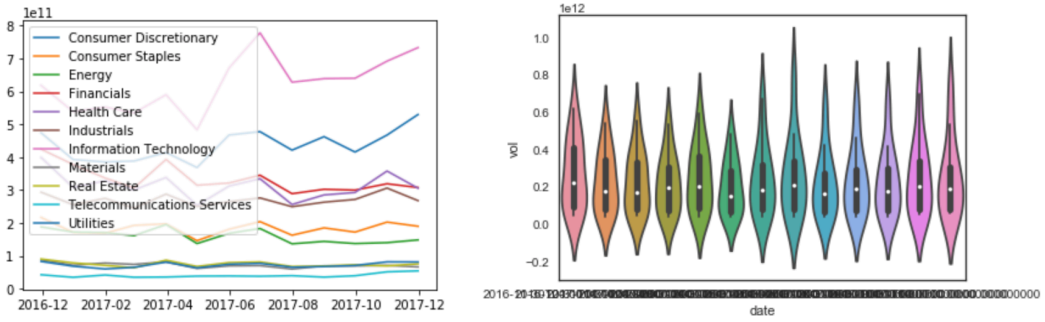


Figure 3 Left: Market Cap by sector. Right: Volume violin plot by sector.

Data Preprocessing

Before prediction, the training data is standardized and normalized using the same scale without 'contaminating' testing data. This lead to a problem that will be discussed later. For binary classification problem, the n-day window data is labelled with either an increase or decrease in the stock price. Percentage changes is calculated using daily opening price, volatility is calculated using daily low and high price. For RNNs, the historical pricing data is sliced into n-days sliding window with closing price of $n + d$ as labels for regression.

date	2016-11-30 00:00:00	2016-12-31 00:00:00	2017-01-31 00:00:00	2017-02-28 00:00:00	2017-03-31 00:00:00	2017-04-30 00:00:00	2017-05-31 00:00:00	2017-06-30 00:00:00	2017-07-31 00:00:00
Name									
A	2.399403e+09	1.550574e+09	2.171564e+09	1.984559e+09	2.152663e+09	1.734384e+09	2.932022e+09	2.560352e+09	1.951266e+09
AAL	6.481850e+09	5.872229e+09	6.907887e+09	5.070157e+09	7.632871e+09	5.959659e+09	5.999397e+09	6.587511e+09	5.799168e+09
AAP	4.997559e+09	2.657138e+09	2.943253e+09	4.438535e+09	3.459496e+09	2.807702e+09	5.079986e+09	4.512578e+09	4.148781e+09
AAPL	7.929405e+10	6.947346e+10	6.741892e+10	7.634232e+10	7.907886e+10	5.334657e+10	9.928964e+10	1.007900e+11	6.258099e+10
ABBV	1.177476e+10	9.176709e+09	9.655630e+09	7.771697e+09	9.295629e+09	6.735718e+09	8.223911e+09	9.257098e+09	7.220344e+09

Figure 4 Sample preprocessed time-series data on historical stock prices

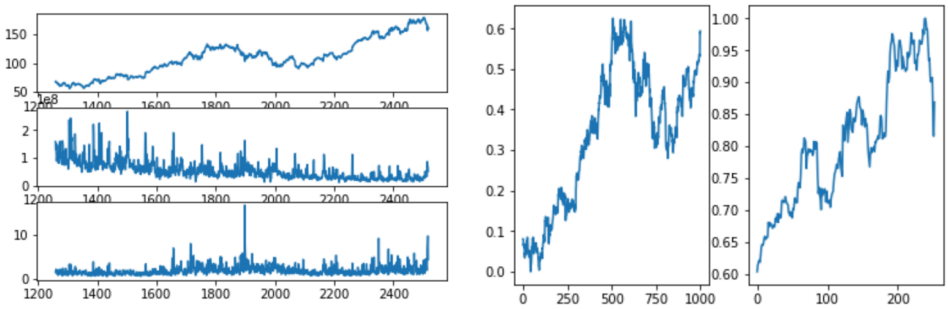


Figure 5 Left:AAPL stock data example (price, volume, volatility) Right: Train-test split

Results and Discussion

The logistic regression model achieved an accuracy of 56%, only marginally better than random guessing. Moreover, as it did not provide much information as to how much the price increase or decrease, it is not very helpful for constructing optimal trading policy. It also have the tendency to give heavy weight on the percentage change of last day, which did not generalize to provide meaningful prediction.

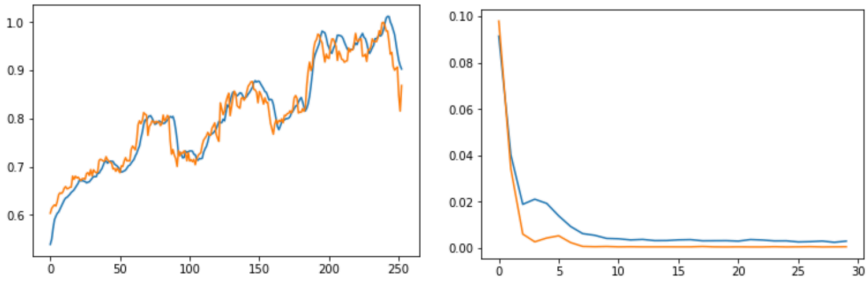


Figure 6 Left:LSTM test price prediction. Right: LSTM training loss vs validation loss

The LSTM model performed well in regression, achieving mean squared loss of lower than 0.01 on test set. However, it seems to learn to merely guessing a similar value of last day closing price instead of equally weighing in volume and company information as observed in the slight shift of price prediction for testing. Adding dropout helped regularize to some extent but a smarter way to formulate loss function could generate a better result. LSTM generalized well for unseen test data, even though test data is in almost entirely different range as compared to training data. It is noted that shuffling training data also reduced testing loss notably. However, given that stock data makes more sense in time-series manner, the model is trained using mini-batch with original sequence maintained.

The Q-learning model is adapted from a Kaggle kernel, as compared to machine learning models before, it automatically generates trading policies that prove to generate profits in some of the testing stocks.

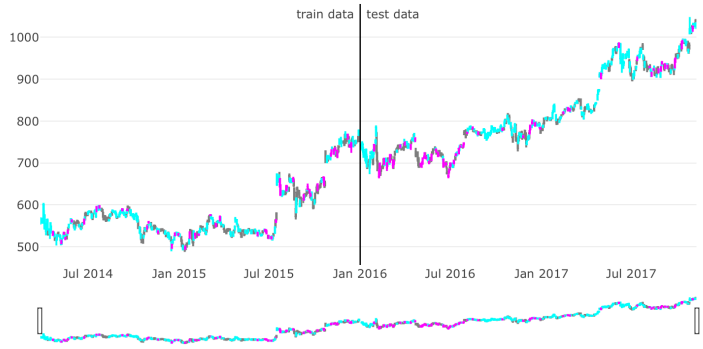


Figure 7 reinforcement learning results

Future Work

There are many possible extensions to the current research in using reinforcement learning for algorithm trading given more time and resources. Here is a list of unverified interesting ideas for future extensions:

- Incorporating more features into the reinforcement learning model could improve the accuracy of the model. Such as features with natural language processing including news indictors, sentiment scores, momentum etc. or more sources for numerical data such as stock indicators, google search index etc.
- Possible ensemble of multiple weak reinforcement learners learned on different features similar to bagging or boosting could improve the accuracy of the ensembled learner.
- Incorporating output of multiple other machine learning techniques as features to the reinforcement learner might improve the accuracy and robustness of the reinforcement learner.
- The economy and market condition should also be added as features as overall market condition can greatly affect the price movement of single stocks.
- The current research in reinforcement learning mostly simplified market environment interaction. It could be interesting to see how the model react to more complex interaction such as shorting, buying on margin in addition to just buy, sell or hold and how it manages portfolio consisting of multiple stocks.

Conclusion

This literature survey explored the machine learning techniques used in stock trading. Price predication for stocks is very challenging and much larger dataset might be needed for robust model to generate consistent profit. Reinforcement learning abstract away the price prediction part but let the agent explore and exploit the best trading policies, it can be a very promising technique for future algorithm traders.

Acknowledgement

I would like to thank Prof. Sun Sun in making this course a very fruitful journey, giving practical advices and interesting studying materials. I would like to thank our teaching assistant Nimesh, Abeer, Shrinu and Nicole for their dedication in answering my doubts and help with explaining assignment problems. Lastly, I would also like to thank the University of Waterloo for providing the resources and learning materials for the course.

Reference

[1] Bengio, Y. S. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, I, 5 (2):157–166.

[2] Bjoern Krollner, B. V. (2010). Financial time series forecasting with machine learning techniques: A survey. *European Symposium on Artificial Neural Networks: Computational and Machine Learning.*

[3] Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2015). Gated Feedback Recurrent Neural Networks. *arXiv: 1502.02367v4*.

[4] Dayan, P., & Niv, Y. (2008). Reinforcement learning: The Good, The Bad and The Ugly. *ScienceDirect*, 18:1-12.

[5] Jiang, Z., & Liang, J. (2017). Cryptocurrency Portfolio Management with Deep Reinforcement Learning. *arXiv:1612.01277v5*.

[6] Lu, D. W. (2017). Agent Inspired Trading Using Recurrent Reinforcement Learning and LSTM Neural Networks. *arXiv:1707.07338v1*.

[7] Molina, G. (n.d.). Stock Trading with Recurrent Reinforcement Learning.

[8] Sezer, O. B., Ozbayoglu, A. M., & Dogdu, E. (2017). An Artificial Neural Network-based Stock Trading System Using Technical Analysis and Big Data Framework. *arXiv: 1712.09592v1*.

[9] Soroka, J. V. (2016). Stock Trading with Reinforcement Learning.

[10] Xiao, C., & Chen, W. (2018). Trading the Twitter Sentiment with Reinforcement Learning. *arXiv:1801.02243v1*.

[11] Zhengyao Jiang, D. X. (2017). A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem. . *arXiv:1706*, 10059.

CS489 Project - Option A Literature Survey