Deep Actor-Critic Experiment for Stock Trading Amid Sentiment and Buzz

Omer Gold
Blavatnik School of Computer Science
Tel Aviv University
Tel Aviv 69978, Israel
omergold@cs.tau.ac.il

September 30, 2019

Abstract

In today's stock markets, automatic trading machines are widely used by investment firms, hedge funds, banks, and other various financial players. Recent advances in deep reinforcement learning motivates to model stock trading as a Markov Decision Process, and to study how well can deep reinforcement learning algorithms preform in learning a profitable stock trading strategy.

In this work, we train deep reinforcement learning agents over an identical Actor-Critic architecture, but some of the agents are given additional data, such as sentiment and buzz, which indicate the stock's sentiment and coverage volume in public news sources, respectively. By comparing the results of the different trained agents, we provide an evidence that considering sentiment and buzz data (in addition to price data) may significantly improve the performance of such agents.

1 Introduction

Stock trading gives traders the opportunity to earn (or lose) money by trying to predict upcoming future changes in companies' public value. One of the most popular ways traders can earn money is by selling their stock for a profit if the stock price sufficiently increased from their purchase price. "Investing" usually refers to buying a stock and holding it for a long term (at least several months and mostly years). "Trading" usually refers to a short term buy-and-sell strategy (can be multiple times in one day). In general, trading is known to be of a higher risk than long-term investing. There are evidences that individual investors who make less trading operations over time, have better results on average [5].

To aid individuals, hedge funds, banks, and investment firms in modern stock trading, the use of computer algorithms is rising, for trying to find better short-term trading strategies that could yield higher expected returns. Many of the automatic trading algorithms are designed for "high-frequency trading" (HFT), that is, trading in a very small time resolution of small fractions of a second. Thus, these algorithms are required to have extremely fast running-times, making them able to decide about their actions in small fractional seconds, and therefore, they mostly use deterministic formulas without exploiting expensive machinery such as deep neural networks.

Another trading approach is "mid-term trading", that is deciding on trading actions in a hourly or daily resolution. This allows the use of more complex machinery such as machine learning algorithms, and deep neural networks. The main and older approach to use machine learning (ML) algorithms for stock trading is

Preprint.

Price Prediction: Use ML to predict future stock price, then trade using a predetermined trading strategy, considering the ML price prediction, brokerage fees, taxes, etc.

A different and newer approach, motivated by reinforcement learning (RL) algorithms is

Trading Strategy Learning: Given stocks data, use RL to directly learn how to trade with the aim of maximizing profits.

The latter approach includes "everything" needed for trading, it incorporates price prediction, trading rules, and trading strategy, so that the trained RL-agent could already start trading by itself.

While the Price Prediction approach was liberally investigated (see survey [11] and the many references therein), it seems that the Trading Strategy Learning approach is much less investigated in the academic literature, especially if we restrict ourselves to the state-of-the-art RL algorithms that use neural networks (deep RL); such studies appeared only recently, see [7, 4, 12]. It seems that the emerging field of trading strategy learning using deep RL has a large room for further investigation.

1.1 Contribution

In this work, we use a deep *Actor-Critic* approach to study and experiment day-trading strategy learning. Our main objective is to compare different RL-agents that were trained on the same architecture, but some were given additional data, such as sentiment and "buzz" scores (in addition to price data), which indicate the stock's sentiment and coverage volume in public news sources, respectively.

Specifically, we compare the performance of the following different RL-agents.

1. Agent 1: trained over price data.
2. Agent 2: trained over price, and sentiment data.
3. Agent 3: trained over price, sentiment, and buzz data.

We are not particularly aiming to find a general "superior" trading strategy, but rather comparing the performance of the above RL-agents. We train our agent under the same hyper-parameters and network, thus, such a comparison should give us an indication to whether considering also sentiment and buzz data gives an advantage over taking only price data, for training deep reinforcement agents day-trading strategies, and if so, how significant this advantage is.

2 Problem Statement

We would be given a starting amount of cash and some starting amount of stocks. These amounts will be slightly randomized. Each timestep (day) our agent receives the current stock data, such as price, sentiment, and buzz. Our agent would learn to maximize the value of its portfolio at the end of a given time interval (shares and cash). At each timestep, he could buy, sell, or do nothing.

There are four types of buying and selling: small, medium, strong, and bullish/bearish. Small buy is buying shares using 25% of our current available cash, in medium buy we use 50% of our current cash, in strong buy we use 75%, and in bullish buy we use all our available cash to buy shares of the stock. Similarly, in small, medium, strong, bearish sells, we sell 25%, 50%, 75%, 100% of our current shares, respectively.

We model the stock trading "game" as a Markov Decision Process (MDP) as follows.

- State s: each state is a vector that contains: the amount of available cash, holding amount of each stock, current timestep price of each stock, sentiment and buzz scores of each stock, and the last five days average of these parameters (price, sentiment, and buzz).
- Action a: there are nine possible actions: small buy/sell, medium buy/sell, strong buy/sell, bullish buy, bearish sell, and hold (do nothing). See above for the description of each action.
- Reward $r(s, a, s')$: the difference between the current portfolio value to the starting portfolio value, when action a is taken at state s and arriving at the new state s'. We also give the agent a small penalty if he chose to do nothing (hold).

- Policy $\pi(s)$: the trading strategy at state s. It is essentially the probability distribution of a at state s.
- Action-value function $Q_\pi(s, a)$: the expected reward achieved by taking action a at state s, when following the policy π.

Trading Strategy Goal: Maximize Portfolio Value. The goal is to design a trading strategy that follows the rules mentioned above, and maximizes the value of its portfolio (sum of shares value plus cash value) at a target time t_f in the future. That is maximizing

$$\sum_{t=1}^{t_f-1} r(s_t, a_t, s_{t+1}).$$

Due to the Markov property of the model, the problem can be boiled down to optimizing the policy that maximizes the action-value function Q_π. However, this problem is very hard because the action-value function is unknown to the policy maker and has to be learned via interacting with the environment. Hence, in this work we employ deep reinforcement learning to solve this problem.

3 Data Description and Preprocessing

Qunadl [2] offers various financial data sets. We use the following two datasets.

1. WIKI Prices [3], for obtaining historical daily opening and closing stock prices.
2. FinSentS Web News Sentiment [1], for obtaining daily news sentiment indicators for a stock. This includes
   (a) Sentiment Score: A measure of bullishness and bearishness of equity prices calculated as a statistical index of the news corpus. Scores are defined on a scale of -5 to 5, where -5 and 5 are extremely bearish and bullish indicators, respectively.
   (b) Buzz Score: Normalized value of change in standard deviations of periodic news volume. Buzz scores reflect a sharp change in news volume, thus serving as a risk alert indicator. Defined on a scale of 1 to 10, high buzz score reflects higher volatility.

Since our main objective in this work is to compare the performance of different RL-agents that were trained on the same architecture (but different data), we use one representative stock to work with. We leave the problem of simultaneously trading multiple stock for future work (such a setting should indeed involve more complex learning due to possible strong correlations between companies' data).

Since we want our experiment to be easily reproducible by the public we focus only on free data in this work. While WIKI Prices dataset is free of charge, FinSentS Web News Sentiment dataset gives for free only data on the stock of Apple (symbol: AAPL) and Twiter (symbol: TWTR), for dates starting at 01/01/2013 until 31/12/2017. Since Twiter started publicly trading on 7/11/2013, we have less data for it than for Apple, so we use Apple's stock (AAPL) in this work.

Aligning Sentiment and Buzz Data with Price Data. As mentioned above, we have sentiment and buzz data of AAPL for days from 01/01/2013 to 31/12/2017. The sentiment data includes all days between these dates, even weekends when there is no trading. To make the sentiment and buzz data correspond to the price data, we remove the days where there is no trading, so that the days in the sentiment and buzz data correspond to the days of the price data.

4 Architecture

We use a deep Actor-Critic algorithm. Our implementation is based on the official PyTorch code [10]. For the neural network architecture and parts of the trading environment implementation we assisted the code given in [6]. In this implementation, the policy network and the value network sit as two different linear-layer heads that split from a main "understanding the world" common neural network, which has one Gated Recurrent Unit (GRU) layer.

For computing the loss, we first compute for each action its reward. That is, the difference between its actual environment reward to the expected reward that is given from the value network. The policy

loss is the sum of log probabilities of all the actions we took, each is multiplied by its reward. The value loss is the sum of L_1-loss between the actual reward of each action we took to its expected value.

5 Experimental Results

Our code is written in Python, using PyTorch [9], and OpenAI's Gym toolkit [8]. We now describe our experiment setting and results.

Data split. Our data is for the trading days from 01/01/2013 to 31/12/2017. We split it as

- Training data - from 01/01/2013 to 31/12/2015.
- Validation data - from 01/01/2016 to 31/12/2016.
- Test data - from 01/01/2017 to 31/12/2017.

Training. We trained the following three agents to maximize returns from day-trading under the same architecture and hyper-parameters.

– Agent 1 is trained over price data.
– Agent 2 is trained over price, and sentiment data.
– Agent 3 is trained over price, sentiment, and buzz data.

We ran each agent through 3,500 epochs on the training data, on NVIDIA Titan Xp GPU. We used Adam optimizer with learning rate 0.001.

In each epoch, we gave each agent a starting amount of cash and AAPL shares, these amounts were slightly randomized so the agent could learn from various starting configurations (the randomization parameters were identical among the three agents). We set a fee of 0.1% for each buy/sell operation, reflecting a standard brokerage fee. We also give a (small) penalty to the agent if he choose to "hold" in order to encourage him to day-trade. Indeed, in our results the agents are buying/selling almost every day, which fits our goal.

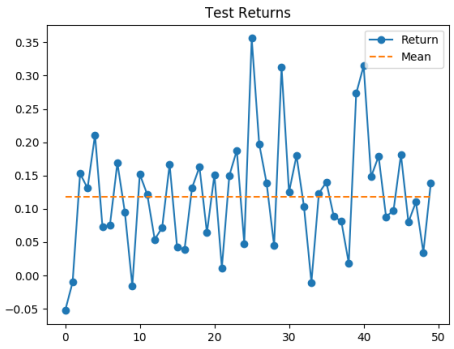
Results. After training and validation of each agent, we executed 50 trading games on the test data (entire 2017 year) of each of the three agents. The results of these games are illustrated in Figure 1. We obtained that the average return per game of Agent 1, Agent 2, and Agent 3 is 11.86%, 16.5%, and 19.12%, respectively (see also the respective figures 1a, 1b, and 1c). Overall, we see that in our case, adding sentiment and buzz data (Agent 3) significantly improves the day-trading average return over using only price data (Agent 1), from 11.86% to 19.12%.

Note that while our agents seemed to learn a decent day-trading strategy (particularly, Agent 3), their performances do not beat other simpler strategies for a "bullish market" (i.e., with an upward trend), such as "buy-and-hold". That is, buying on 01/01/2017 using all the available cash and holding until selling everything on 31/12/2017 would yield nearly 50% return. If one finds good time-points to sell and re-buy during that time, his return can be even higher. It seems that day-trading strategy is a complex task to learn, which involves many difficulties. In the next section, we conclude and propose some directions for future research.

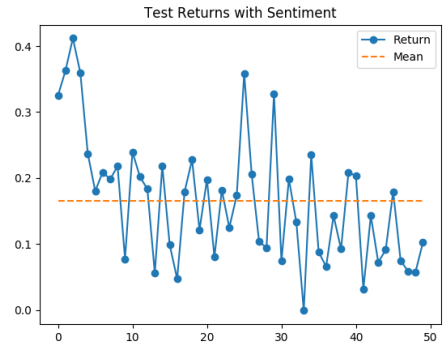
6 Conclusion and Further Research

Our goal in this work is to study how additional stock data, such as sentiment and buzz can affect the performance of the day-trading strategy of deep reinforcement learning agents. We ran experiments for stock trading using three different data settings on the same architecture and hyper-parameters. We found that in our case, adding sentiment and buzz data significantly improves the day-trading average return over using price data only. However, the strategies themselves seems to be far from being superior than other trading strategies. It seems that finding a good day-trading strategy using deep reinforcement learning is a very complex task, which requires further research.

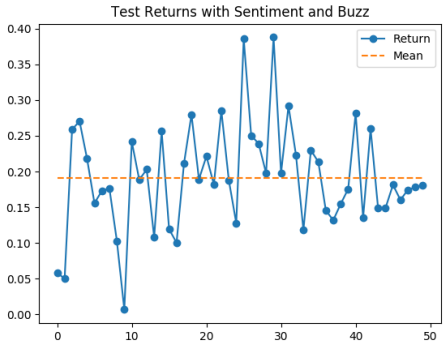
It seems that there are not much academic papers about utilizing deep learning for stock trading. Perhaps, that is due to lack of motivation of researchers to publish promise findings in this field, as



(a) Agent 1: trained over price data only. Achieves 11.86% average return per game.



(b) Agent 2: trained over price and sentiment data (without buzz data). Achieves 16.5% average return per game.



(c) Agent 3: trained over price, sentiment, and buzz data. Achieves 19.12% average return per game.

Figure 1: The average return of each agent.

one may consider to exploit them for personal or cooperate use. Thus, exposing such findings may encourage new improved agents to compete in the same markets, making it harder to profit, and hence the secrecy.

In the academic aspect, it seems that training reinforcement learning agents to trade in the financial markets can be an interesting research problem. Although it has not yet received much (public) attention from the academic community, it may have the potential to push the state-of-the art of other related fields, such as training agents to play multiplayer games.

We finish with some ideas for further research. We propose several directions that we think are worth further investigation, either separately or combined. (i) Use more recurrent layers: since the order of actions is important in trading, it seems that using RNN may be a good architectural fit for the problem. (ii) Continuous action space: let the agent have *all* feasible quantities of cash/stock for buying/selling. (iii) Use different timesteps: performance may vary greatly if we consider longer term or very short-term trading. (iv) Trade multiple stocks simultaneously: this case is particularly interesting due to correlations between the data of different stocks.

References

[1] FinSentS web news sentiment dataset. `https://www.quandl.com/databases/NS1/data`. published by Infotrie.

[2] Quandl. `https://www.quandl.com/`.

[3] Quandl wiki prices dataset. https://www.quandl.com/databases/WIKIP/data.

[4] A. R. Azhikodan, A. G. K. Bhat, and M. V. Jadhav. Stock trading bot using deep reinforcement learning. In *Innovations in Computer Science and Engineering*, pages 41–49, Singapore, 2019. Springer Singapore.

[5] B. M. Barber and T. Odean. Trading is hazardous to your wealth: The common stock investment performance of individual investors. *The Journal of Finance*, 55(2):773–806, 2000.

[6] T. Grek. Reinforcement learning for stock trading. https://github.com/tomgrek/RL-stocktrading.

[7] C. Y. Huang. Financial Trading as a Game: A Deep Reinforcement Learning Approach. *arXiv e-prints*, page arXiv:1807.02787, Jul 2018.

[8] OpenAI. Gym: A toolkit for developing and comparing reinforcement learning algorithms.

[9] PyTorch Official. Pytorch. https://pytorch.org/.

[10] PyTorch Official. Reinforcement learning training example. https://github.com/pytorch/examples/tree/master/reinforcement_learning.

[11] A. Sharma, D. Bhuriya, and U. Singh. Survey of stock market prediction using machine learning approach. In *2017 International conference of Electronics, Communication and Aerospace Technology (ICECA)*, volume 2, pages 506–509, 2017.

[12] Z. Xiong, X. Liu, S. Zhong, H. Yang, and A. Walid. Practical deep reinforcement learning approach for stock trading. *CoRR*, abs/1811.07522, 2018.