Maximising reward in stock trading for top 3 tech stocks

Isha Tyagi
XCS229ii (2021) ; Project Milestone 4
May 23, 2021

1. Abstract

Hedge funds, investment companies and also at personal level, require an automated stock trading strategy which maximizes investment and is profitable. The ever fluctuating stock market is hard to predict. Traditional statistical techniques have not been able to deal with the sequential, volatile and non-linear stock market. Supervised learning has been used widely in predicting stock prices but they are not sufficient to deal with time-delayed rewards. In this project, we make use of the underutilized technique in financial market prediction, Reinforcement Learning, which learns a stock trading strategy by itself that is aimed towards maximizing return on investment. The objective is to devise a novel trading policy based on Reinforcement learning that would be comparative and perhaps better than the already adopted algorithmic trading strategies. Our goal is to maximize profits on stocks of 3 Tech giants, Apple, Amazon and Google. We first aim to find the window size, in days, for training and testing, that would maximize the profits. Our findings suggest a window size of 10 days for Apple and 20 days for Google and Amazon stocks. With a set window size, the algorithm was run to obtain profits with different draw limits and varied buying and sell volume ratio. Buying ratio is the fraction of cash balance that can be used to buy stock. Sell ratio is the fraction of stock inventory that can be sold. The profit ratio was maximised when the draw limit was defined to $20000 coupled with the volume ratio of 0.25. The detailed work has been presented in this paper. Introduction section presents an overview of the topic, recent traditional and machine learning applications in stock trading; why DRL is the need of the hour and how it functions. The Methodology section details the data, model and the training on the collected data. The results section shows our findings based upon tuning the volume ratio. We summarize our results and limitations and how those limitations may be overcome.

2. Introduction

Stock trading is conventionally performed by Supervised learning and modern portfolio theory. But both these techniques do not perform well under certain circumstances, which is when Reinforcement learning performs well. Supervised learning requires a labelled dataset which is not a requirement for reinforcement learning. This is a remarkable advantage as the dataset that we train our model on is usually very large and it would be really cumbersome to label the entire dataset. Supervised learning algorithms predict the probability of future outcomes while reinforcement learning in order to optimize future rewards, utilizes a reward function. Modern Portfolio theory is heavily used but does not perform well in data which is out of sample and it is quite sensitive to the outliers. Modern portfolio theory is unable to incorporate other features and factors which

govern the stock price like, Relative Strength Index, Moving Average Convergence Divergence.

To overcome the limitations posed by supervised learning and modern portfolio theory, Reinforcement learning is utilized. Deep Reinforcement learning maximizes returns along with aversing risks. This conundrum is solved by DRL by maximizing the future expected payoff (reward) over a period of time. Stock trading is a dynamic environment which is sensitive to new slight changes in the market. DRL by utilizing Markov decision process is able to continuously learn from the ever changing market, get feedback and keep optimizing the trading strategy.The feedback makes the DRL continuously improve the performance of its model. DRL can outperform humans and maximizes the portfolio profit over a time period.

2.1 Deep Reinforcement Learning : overview

DRL does the balancing act between exploration and exploitation. If there is no exploration and only exploitation, the algorithm converges at a local optima. The exploration lets the agent try out different policies and gain knowledge upon experience. The agent explores the environment uncharted by humans. Hence, the labelled dataset is not required.

In DRL, experience replay is done. The issue of correlated samples is overcome by experience replay. If the algorithm learns from a batch of consecutive samples, this might result in high variances. This issue is addressed by sampling smaller batches of transitions randomly.

Q-learning by itself is unable to manage a large spatial environment. Q- learning powered by neural networks, that is DRL can also handle data with large space high dimensionality better than the other algorithms.
In totality we'll be utilizing following four concepts to solve our target :

2.1.1 Reinforcement Learning

This unsupervised machine learning technique trains an agent to interact with the environment and subsequently obtain the states and rewards and perform action so as to maximize the total reward.

2.1.2 Deep Reinforcement Learning : DRL estimates Q-value by a neural network. Neural network is applied as a function estimator so that reinforcement learning can be applied on larger data.

2.1.3 Bellman Equation guides the functioning of reinforcement learning algorithms towards maximum reward.

2.1.4 Markov Decision Process models the environment.

2.2 Prior research

Three learning approaches have been recently employed by DRL in financial trading, that are, Actor-only approach, Critic-only approach, actor-critic approach. DRL considers continuous/ discrete state and action.

2.2.1 Critic only approach

This is the most common approach that solves the discrete action space. It utilizes Q-learning, Deep Q-learning. It trains the agent on one stock. It emphasizes learning optimal action selection policy, from the current state, which gives the maximum expected future reward. DQN performs function approximation by a neural network and minimizes mean squared error amongst the Q-values. This approach has a limitation that it functions on finite and discrete state and action spaces. For a large portfolio of

stocks, it's not practical to have finite and discrete spaces and the price is continuous.

2.2.2 Actor-only approach

In this approach, the agent itself learns the optimal policy. The neural network learns the policy rather than learning Q-value. The policy gives the probabilistic distribution of the allowed actions from a given state. This approach is able to play with the continuous action space environment as well. Policy gradient is the technique used mostly which learns the optimal policy directly and maximizes the expected total payoff/ rewards.

2.2.3 Actor-Critic approach

This is a fairly new technique to be applied in the financial market. This approach works by updating the actor and critic network simultaneously. The critic approach approximated the value function. The actor approach tells and updates the probability distribution of the allowed actions. As the updates keep happening, the actor is able to make its actions better along with the critic bettering itself at evaluation of the actions taken. This approach is adaptable to a complex and large environment. The algorithms used in this approach are A2C, PPO and DDPG.

A2C algorithm utilizes duplicates of an agent and they update gradients parallely with divergent dataset. The agents work in the identical environment separately.

PPO technique controls the update of policy gradients. It makes sure that the new policy does not deviate much from the previous policy.

DDPG is the combination of policy gradient and Q-learning frameworks. The function approximation is done by neural networks.

3. Methodology

The code for this project can be found on : https://github.com/ishatyagi22/stock-prediction-rl

3.1 Data collection

In our model, we'll utilize the OHLC data of **3 tech stocks** from January 2011 to December 2018, taken from Yahoo Finance.

The standard OHLC format consists of 5 data points, of which we'll take into account only the four data points, that is the OHLC data. For a stipulated interval, OHLC data gives the information about the prices of the stocks. OHLC is **O**pen, **H**igh, **L**ow and **C**lose for a given stock. In a specific timeframe, Open and close data points are the starting and ending prices. High and low data points are the highest and lowest prices in the interval.

The turbulent dataset is usually not considered. For example, hard from 2008-2011 will not be taken into account. The data from the financial crisis during 2008-2011 is rejected because this could harm the stability of the trading model. Though, this period being such an extraordinary event, could be used to test the robustness of trading models.

3.2 Model

As explained by Mike Wang [8] and Jae Won Lee [9] , In the learning task of stock prediction, it is more natural and effective to represent target values by the successive relative changes in price since the previous time point than the absolute prices after a fixed time horizon which are generally adopted in conventional time series approaches based on supervised learning methods. In this sense, reinforcement learning can be a promising approach for

stock price prediction, representing and leaming the delayed rewards, as well as the immediate rewards, from interactive processes more effectively

Since the state space is large and it is not known what the expected reward of an action at a particular state may be, it seems more fitting to use a “model-free” RL algorithm. Here we try to use the Deep Q Learning algorithm.

Using a neural network to approximate the function for state transition mapping. Since the state space is large, using a neural network to approximate is more efficient.

One of the interesting things about Deep Q-Learning is that the learning process uses 2 neural networks. These networks have the same architecture but different weights. Every N steps, the weights from the **main network** are copied to the **target network**. Using both of these networks leads to more stability in the learning process and helps the algorithm to learn more effectively. In our implementation, the main network weights replace the target network weights every window.

3.2.1 MDP for Stock Price Prediction:

- Action – Buy/Sell/Hold => 3 Actions , result in decreasing, increasing, and no change of the stock shares $\boldsymbol{h}$, respectively.
- States – The sigmoid of the difference between the closing price of 2 consecutive days in a window.
- Rewards – If the action is sell, then the profit is updated as the difference between the sell price and the price stored in data.
- **Policy** $\pi(s)$: the trading strategy at state s, which is the probability distribution of actions at state s.
- **Q-value** (s, a): the expected reward of taking action a at state s following policy π .

3.2.2 Training

The agent retrieves the window size. The data is transformed into a vector containing stock data from the data file. It is done by first splitting each row of the data into lines at each line break. Each line(row) is then split by “,” (csv file) and a vector is received containing only the closing price of the stock at the end of each day.

The states in a window are the sigmoid of the difference in closing price of the previous day from the next day. A day next to the window is also taken to subtract the closing price of the last day in the window from.

At a particular time point, there are three actions: buy, sell and hold are possible. The stock price is updated at t+1. Stock purchased is added to the inventory. Profit made by selling is added to the total profit made.

The action is randomized for exploration and exploitation. When epsilon is zero, the agent will choose the highest reward awarding action and might get stuck at local optima. With every action replay, the value of epsilon is updated from 1 to 0.03. Epsilon value of 0.3 states that the action will take place with 0.3 probability, this is to avoid overfitting.

The reward function after any action and at a particular state is the delta in stock inventory value and the total profit value. New state is achieved after it.

The agent works to maximize the change in the inventory and the total profit earned.

4. Results:

4.1 Optimal window size

Our first aim was to find out the window size for training and testing both, to maximise the profits. We trained our Apple, Amazon and Google stock data with window sizes of 5, 10, 15 and 20days. We tested our data on similar window sizes . The results show that the profit was maximized at 10 days for Apple and 20 days for Google and Amazon stocks (***Fig.1***).

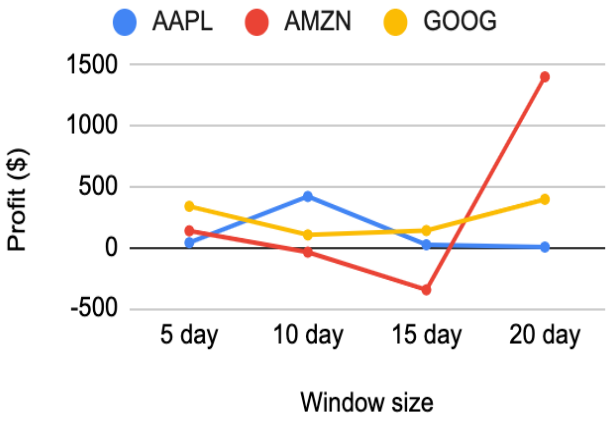

Fig.1 Profit Vs Window size.

4.2 Variation in profit with respect to draw limit and buy/sell volume ratio.

After determining the window size to work upon, the next goal was to see if the profit improves by manipulating the draw amount and buy/sell volume ratio. Draw amount limit of $20000 was set with buying and selling capacity of the agent to be one stock at a time. The buying and selling power was increased at a time to 0.25 and 0.50. 0.25 volume ratio means that 0.25 fraction of the stock inventory would be sold or 0.25 fraction of the amount balance would be used to purchase new stocks when the action is taken.

The buy/sell points and profit earned by Google stocks at the window size of 20 days with varying draw limit and volume ratio are shown in **Fig.2** and ***Table1*** respectively. We did not need to put a limit of maximum draw amount with the constraint of buy/sell one stock because the agent did not draw more than our desired limit of $20000.

The buy/sell points and profit earned by Amazon stocks at the window size of 20 days with varying draw limit and volume ratio are shown in ***Fig.3*** and ***Table1*** respectively.

The buy/sell points and profit earned by Apple stocks at the window size of 20 days with varying draw limit and volume ratio are shown in ***Fig.4*** and ***Table1*** respectively.We did not need to put a limit of maximum draw amount with the constraint of buy/sell one stock because the agent did not draw more than our desired limit of $20000.

Table 1 : Profit earned by each tech stock in 2019 under various conditions.

Stock name	Draw amount not fixed	Draw amount limit $ 20000		
	Volume ratio = 1 stock at a time	Volume ratio = 1 stock at a time	Volume ratio= 0.25	Volume ratio= 0.50
Google	$395	-	$1992	$1856
Amazon	$6161	$1324	$2325	$2602
Apple	$418	-	$15672	$16584

We further evaluated profits on a higher volume ration but no significant increase in profit was observed, so they were dropped from this experiment. The profit ratio tremendously increases when the limit on amount drawn is put. A significant profit is earned when the power of agent to buy/sell is increased to a higher ratio.

Fig. 2 Google buy/sell time points in 2019.Green denotes buy time point and red denotes sell time point. A: Draw amount not fixed and the agent can buy/sell only one stock at a time or hold. A profit of $395 was earned. B: Draw amount limited to $ 20000 and buy/sell volume ratio of 0.25. A profit of $1992 was earned. C: Draw amount limited to $20000 and buy/sell volume ratio of 0.5, A profit of $1856 was earned.

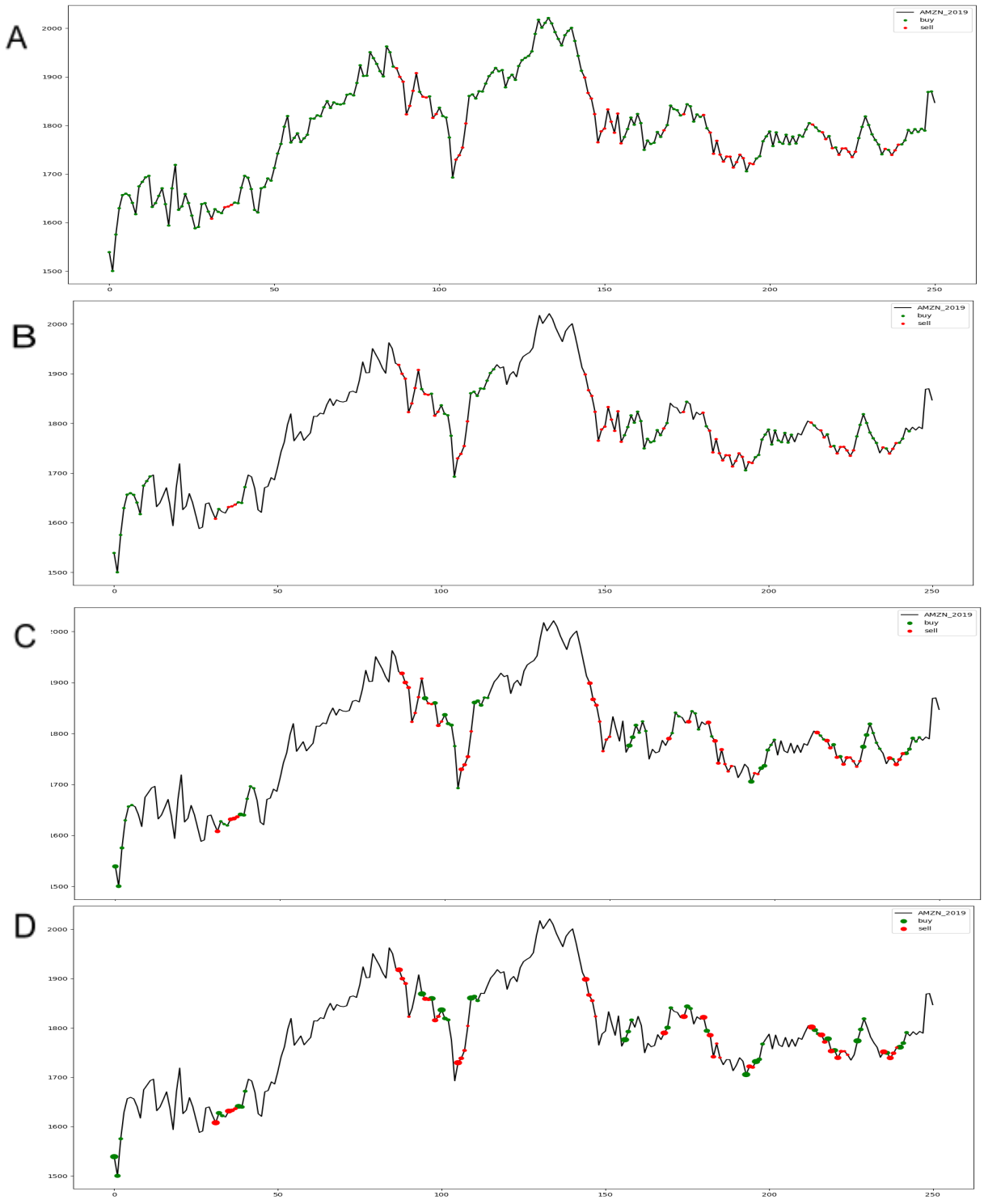

Fig. 3 Amazon buy/sell time points in 2019.Green denotes buy time point and red denotes sell time point. A: Draw amount not fixed and the agent can buy/sell only one stock at a time or hold. A profit of $6161 was earned. B. Draw amount is limited to $ 20000 and the agent can buy/sell only one stock at a time or hold.A profit of $1324 was earned C: Draw amount limited to $ 20000 and buy/sell volume ratio of 0.25. A profit of $2325 was earned. D: Draw amount limited to $20000 and buy/sell volume ratio of 0.5, A profit of $2602 was earned.

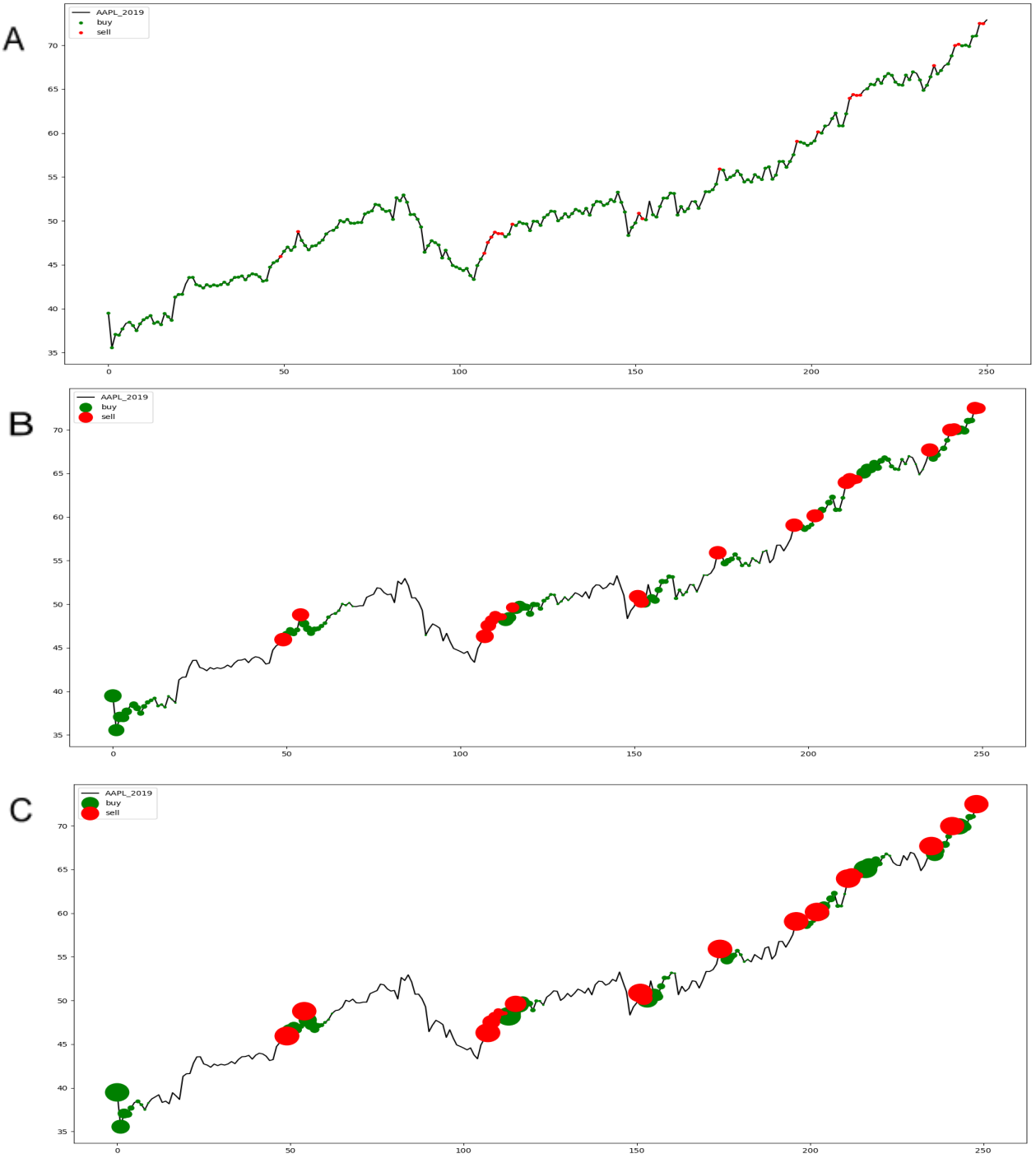

Fig. 4 Apple buy/sell time points in 2019.Green denotes buy time point and red denotes sell time point. A: Draw amount not fixed and the agent can buy/sell only one stock at a time or hold. A profit of $418 was earned. B: Draw amount limited to $ 20000 and buy/sell volume ratio of 0.25. A profit of $15672 was earned. C: Draw amount limited to $20000 and buy/sell volume ratio of 0.5, A profit of $16584 was earned.

5. Discussion

5.1 Significance of the result in addressing automated stock trading

Deep reinforcement Learning seems to be a very promising approach to make short term trade decisions. With some tuning the learnt model was able to consistently achieve > 10% annual return (upto 75% in certain cases).

Here we train a neural network directly on the reward. Training directly on the reward to choose actions will tend to focus the algorithm on the situations where the choice of action makes the biggest difference. If you train for prediction first, and then use prediction for control, the training values any improvement in prediction equally, even if it doesn't improve control. We ask what complex model would be able to solve a complex problem, and train the model end-to-end for a long time on data. This yields emergent behavior and effective models.

In general, reinforcement learning will learn to use whatever forecasting edge it has, to generate a maximum reward in a dynamic environment.

5.2 Expansion upon the findings

It would be great to see how more complex Deep learning and Reinforcement learning algorithms could workaround some of the disadvantages of a simpler approach like DQN. These could be summed as:

- Overestimation/Overfitting in certain cases. (AAPL data was overfit in the 9th/10th episode of training).
- The learning rate with DQN is very slow due to the overhead of experience Replay.
- The generated model is sometimes unstable, like it makes only Sell calls, when you do not have inventory it becomes a no-op.

5.3 Limitations and directions of future research.

No estimation of total Investment done by system. The nearest estimate is done by draw. At one time, only a single stock is purchased or sold. This limits the gains which the algorithm can make.

Solution: Buy/ sell number of stocks in proportion to the probability distribution of the action space.

Currently in the model we use a fixed ratio of the number of stocks to buy (called *volume_ratio*). A future approach would be to make the model suggest the number of quantities of a particular security to be bought or sold.

No estimation of total Investment done by system. The nearest estimate is done by total draw.

The current model works best for a single symbol stock trading, we could use correlations among multiple stocks to make better decisions.

The current algorithm purely depends on patterns of stock price movement and making decisions. A number of external factors contribute to movement of stocks, like global financial situations, wars, pandemics etc. An ensemble of models which take into consideration some external inputs like sentiment etc would lead to better decisions.

6. References:

[1] Jae Won Lee, "Stock price prediction using reinforcement learning," ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No.01TH8570),

2001, pp. 690-695 vol.1, doi: 10.1109/ISIE.2001.931880.

[2] Liu, Y. (2019). "Reinforcement Learning Applications in Real Time Trading," Joseph Wharton Scholars. Available at https://repository.upenn.edu/joseph_wharton_scholars/65

[3] Maqsood, H.; Mehmood, I.; Maqsood, M.; Yasir, M.; Afzal, S.; Aadil, F.; Selim, M.M.; Muhammad, K. A local and global event sentiment based efficient stock exchange forecasting using deep learning. Int. J. Inf. Manag. 2020, 50, 432–451. [CrossRef]

[4] Patil, P.; Wu, C.-S.; Potika, K.; Orang, M. Stock market prediction using ensemble of graph theory, machine learning and deep learning models. In Proceedings of the 3rd International Conference on Software Engineering and Information Management, Sydney, Australia, 12–15 January 2020.

[5] Quang-Vinh Dang. Reinforcement Learning in Stock Trading. 2019. Ffhal-02306522f

[6] Ritter, G, “Machine learning for trading”, August 11, 2017.

[7] Thibaut Théate and Damien Ernst. "An Application of Deep Reinforcement Learning to Algorithmic Trading." (2020).

[8] Mike Wang. “ Deep Q-Learning Tutorial : minDQN”. Towards data science, November 17, 2020

[9] Jae Won Lee. “Stock Price prediction using Reinforcement Learning”.ISIE 2001, Pusan, KOREA

[10] Nathan Lambert. “ Fundamental iterative methods on Reinforcement Learning”. Towards data science, Apr 8, 2020

[11] Zhao, F. (2013). Forecast Correlation Coefficient Matrix of Stock Returns in Portfolio Analysis. UCLA. ProQuest ID: Zhao_ucla_0031N_11617. Merritt ID: ark:/13030/m5bv81k5. Retrieved from https://escholarship.org/uc/item/52n659j4

[12] Hongyang Yang, Xiao-Yang Liu, Shan Zhong, and Anwar Walid. 2020. Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy. In ICAIF ’20: ACM International Conference on AI in Finance, Oct. 15–16, 2020, Manhattan, NY. ACM, New York, NY, USA.

7. Presentation Video Link :

https://drive.google.com/drive/folders/11XOSbVr3J28uBUN10RMenTvAPjfzfTIW?usp=sharing