Project Title: Reinforcement Learning Based Stock Trading Agent

Submitted By: Dai Yirui, Dong Meirong, Li Duo, Lu Jiahao, Guo Feng

Submitted On: 24 – May – 2020

Executive Summary

Stock market is an important part of the modern-day economy in terms of spending and investment. It creates a platform for businesses to raise additional capitals for expansion, for governments to roll out fiscal and monetary policies to promote greater investment, and for individuals to make personal wealth. However, opportunities are always coupled with risks (Little, 2020), there are many risks presented in stock trading (e.g. political environment and economic risks) and most of them are unpredictable (e.g. Black swan events). Although risks are unavoidable, with thoughtful selections that match one's risk appetite, the risks can be significantly lowered. The aim of the project is to design and build an automated trading agent based on Reinforcement Learning methods to simulate the actions of a sophisticated investor in trading and to empower the agent with the ability to make fair decisions to maximize investment profits and to lower risks. The 4 reinforcement methods used in this project are: SARSA, Deep Q learning, Actor-Critic Network, PPO2. Our team has studied and assessed each method in stock trading carefully and scientifically to conclude their effectiveness in achieving the aim. On top of that, our team has also explored the limitations of these methods in meeting the goal and discussed some possible enhancements to overcome the obstacles.

Contents

1.0 Business Problem Specification

Introduction

Stock plays a pivotal role in the modern economic system in providing a formal platform for buyers to make investments in hopes of making profits, for governments to regulate their economy, and most importantly, for businesses to raise additional money to support growth (Premkumar, 2020).

To trade in the stock market, buyers can take 3 actions which are buy, sell and hold. The actions are intuitive but deciding when to take the actions and how much to invest each time can be an extremely complicated task and requires a lot of reasonings, especially for new investors, because the stock market is full of uncertainties (Little, 2020). Generally, the complexity and risks can be lowered by understanding and following the market pattern for trading. In this project, our team aims to build an automated trading agent based on reinforcement learning methods (e.g. SARSA, DQN, Actor-Critic Network and PPO) to make reasonable decisions based on historical patterns to maximize investment gaining and lower risks. Then new investors can use the agent as a reference point and learning platform for strategic trading.

The agent is also helpful in providing sophisticated buyers a rational basis for decision-making during panic time. When the market is going down consistently and the outlook is uncertain, buyers tend to sell their positions at lower prices to liquidate the stocks. This will further cause a decrease in market price and result in a loss. The use of an automated trading agent would provide a second opinion for buyers to make far-sighted decisions and minimize emotional decisions.

The agent's performance is assessed using Standard and Poor's 500 (S&P) index because S&P tracks the stocks of 500 large-cap companies in the United States and is widely regarded as the best gauge of large-cap equities in the United States (Amadeo, 2020).

Background

Reinforcement learning methods are techniques used to repeatedly feed new information into systems/agents from the available raw data in an iterative process that allows them to maximize the value of a certain predetermined reward.

Recently, there has been a huge increase in the use of artificial intelligence for trading in financial markets such as stock (Forbes Technology Council, 2019), especially after the program AlphaGo defeated the strongest human contemporary Go board game player Lee Sedol in 2016.

The success factors of such systems are usually measured by their abilities to:

- automatically and continuously trade stocks; and
- avoid loss in most time periods and gain as much profit as possible.

2.0 Solution Design

Assumptions

The assumptions of the project are listed below.

- The initial account balance is $50,000. The agent will not be able to buy more stocks after the account balance is negative.
- The dividends are not included in the profits.
- A commission fee of 0.1% is included. If it is less than 1 dollar, commission fee is set to 1 dollar instead.
- As stock price moves across the entire trading hours, in order to have a consistent performance measurement, the team decided to only buy/sell at the last minute of the trading hour, the transaction price will be the daily close price.
- Net worth is equal to daily stock holding value plus account balance. Profit is equal to net worth minus initial account balance ($50,000).
- In order to show the performance of the reinforcement learning model, the system also shows the relative performance. It is based on buying all possible stocks and holding till the end.

Data Preparation

In this project, our team chose the historical data of SPDR S&P 500 ETF Trust (SPY) as the dataset for study. The data can be downloaded from Yahoo Finance (Yahoo Finance, 2020).

The dataset has 7 features which are: trading date, opening price, highest price of a trading date, lowest price of a trading date, closing price, adjusted closing price, and transaction volume.

An adjusted closing price is a stock's closing price on any given day of trading that has been amended to include any distributions and corporate actions that occurred at any time prior to the next day's open. Since the profits (e.g. dividends) are not included in the study, adjusted closing price is also excluded from this study. The exclusion should not have significant impacts on the overall study because adjusted closing price and other features share similar patterns over time.

The full dataset has 6,869 samples and has been split into two subsets: a training dataset with 6,528 samples and a test dataset with 341 samples. The training period is from 29-January-1993 to 31-December-2018 and the test period is from 02-January-2019 to 08-May-2020. The data patterns are illustrated below.

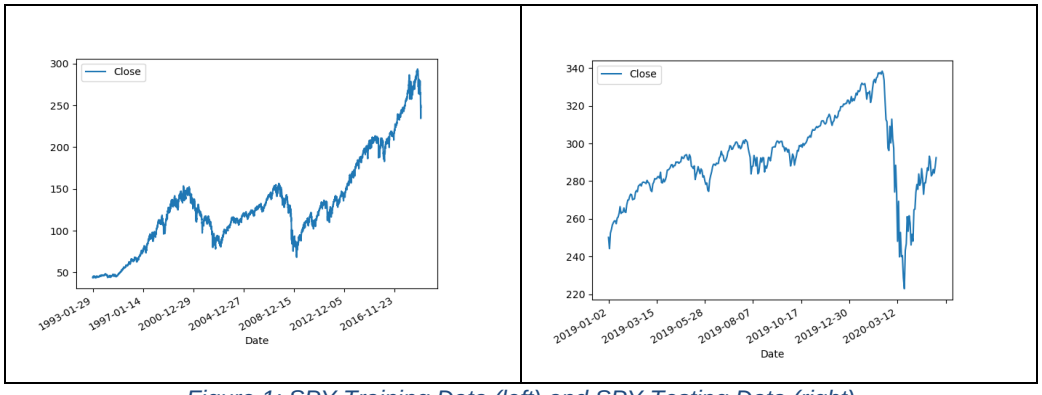


Figure 1: SPY Training Data (left) and SPY Testing Data (right)

Introduction to Reinforcement Learning Methods

The section introduces the 4 reinforcement learning models used in this project which are State–Action–Reward–State–Action (SARSA), a simple Actor-Critic Model, Proximal Policy Optimization (PPO) and Deep Q Learning (DQN), to assess the performance of an automated stock trading agent in a simulated trading environment (the prices used are real trading prices). The aim is to maximize the accumulative rewards (the profits) over time.

SARSA

The team first prototyped the agent with the SARSA model. The underlying idea is to build an optimal Quality Table (Q-table) that can guide the action of agents under all situations. The very key concept of building a Q-table is to have a dictionary-like storing place that can map the key - observations of agents (state) to its corresponding value - optimal action under that condition.

To model the problem with SARSA, there are 2 issues. Firstly, the Q-table can have infinite size. Because the stock price as well as market volume can, in theory, reach a very high level that goes beyond the threshold of Q-table, causing the enumeration of all possible combinations to form a Q-table impractical. Additionally, it is meaningless to observe the actual figures of the market. As there is hardly a single benchmark that can be used throughout the time for the agent to measure the price or volume. For example, the same price - 150 of SPY, is considered as high in the period prior to 2008, but is considered as low in 2020.

Our team borrowed the concept proposed by Deshpande (Deshpande, 2017) and discretized the observation space into 15 buckets to handle the problems. At any certain time, the agent is allowed to select only 1 action from the list [“hold”, “sell”, “buy”]. Together with combinations of price and transaction volume, a lookup Q-table of (15, 15, 15, 15, 15, 3) with 6 dimensions is constructed to guide the reinforcement learning agent. The Q-table is initialized with all 0s. A reinforcement learning agent that utilizes the SARSA algorithm is built together with the Q-table.

Open	High	Low	Close	Volume	Action	Q Value
1	0	1	0	-20	0 (Hold)	48.462566424685456
1	0	1	0	-20	1 (Sell)	35.7378352206136
1	0	1	0	-20	2 (Buy)	49.049143012290905

Table 1: SARSA Q-table Example

In the training phase, the agent observes its current state and then makes a step (action) with reference to the Q-table following an Epsilon-Greedy approach as shown in Figure 3 to reach the next state.

$$A_t \leftarrow \begin{cases} \underset{a}{\text{argmax}}\ Q_t(a) & \textbf{with probability } 1-\epsilon \\ a \sim Uniform(\{a_1 \dots a_k\}) & \textbf{with probability } \epsilon \end{cases}$$

Figure 3: Epsilon Greedy

The training parameters are defined as following:

- Epsilon - ε: start from 1 and decay with training iterations and stop at 0.1.
- Q-values in Q-table get updated by the reward given with respect to the action taken from a state. Figure 4 shows the equation that updates the Q value for the state.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\Big].$$

Figure 4: Equation to Updates the Q- value for the State

- Learning rate - α: start from 1 and decay with training iterations and stop at 0.01.
- Discount rate - γ: fixed at 0.98.

Iteratively, a Q-value is populated for each state-action pair in the Q-tables. Theoretically, over the long run, each Q-value has been visited multiple times (approaching infinity) and thus can accurately reflect the expected return of a certain step at its state.

Furthermore, our team has added other hard constraints such as selling is not allowed when there are no shares in hand, and buying is not allowed when there is insufficient balance in the problem scope. If the agent violates the constraints (e.g. attempting to buy when there is insufficient balance), the agent will be forced to take action hold.

However, our team did not include the number of shares in hand as well as the balance in the environment for this model.

The goal is to train the agent so that it can act effectively on price and transaction volume.

Actor-Critic

The method has 2 components: a policy structure (the actor-network) and an estimated value function (the critic-network). The actor takes in the state of the environment, then returns the best action, or a policy that refers to a probability distribution over actions. The critic evaluates the actions returned by the actor using value function approximations. The approximations are then feedback to the actor to update it in the direction of its gradient.

These two networks are trained simultaneously. With time, the critic improves its Q-value prediction, and the actor also learns how to make better decisions, given a state. The method is between on-policy and off-policy because the agent learns off-policy by trial and error with the help of the actor, but the critic-network is always on-policy because the critic must learn about and critique whatever policy is currently being followed by the actor.

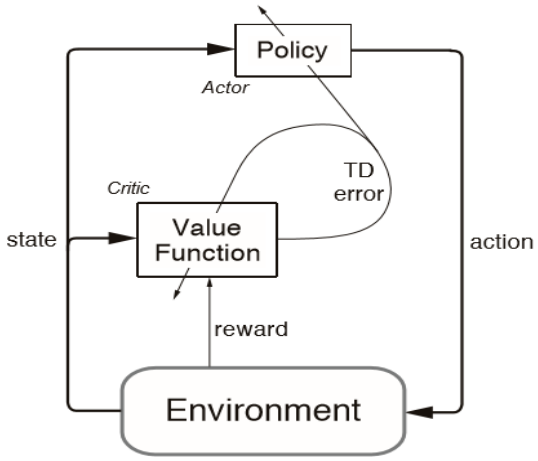


Figure 5: Typical Actor-Critic Architecture

The error is usually evaluated using this function: $\delta t = Rt+1 + \gamma Vt(St+1) - Vt(St)$

R(t+1) is the reward for the action A(t) taken by the actor at step t from S(t). V(t) is the estimated value function implemented by the critic at time t. If the error is positive, it suggests that the tendency or preference to select A(t) should be strengthened for the future, whereas if the error is negative, it suggests the tendency or preference should be weakened.

PPO

PPO is another family of policy gradient method for reinforcement learning, it alternates between sampling data through interaction with the environment and optimizing a "surrogate" objective function using stochastic gradient ascent.

It has some of the benefits of trust region policy optimization (TRPO), but they are much simpler to implement, more general, and have better sample complexity (empirically).

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right].$$

Figure 6-1: Surrogate Objective Function

According to the paper proposed by Schuman in 2017 (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017), PPO outperforms other online policy gradient methods, and overall strikes a favorable balance between sample complexity, simplicity, and wall-time.

The underlying principle of the method is formulated as below.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

Figure 6-2: Clipped Surrogate Objective Function

The epsilon is a hyperparameter ϵ. The first term inside the min is L^{CPI}. The second term, $(clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\widehat{A}_t)$ modifies the surrogate objective by clipping the probability ratio, which removes the incentive for moving r_t outside of the interval [1−ϵ, 1+ϵ]. Finally, the minimum of the clipped and unclipped objective was taken, so the final objective is a lower bound (i.e., a pessimistic bound) on the unclipped objective.

$$\hat{A}_t = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_T)$$

Figure 7: Advantage Estimator Function

In the Advantage Estimator Function, t specifies the time index in [0, T], within a given length-T trajectory segment.

A pseudo code implementation of PPO algorithm that uses fixed-length trajectory segments is shown below.

Algorithm 1 PPO, Actor-Critic Style

```
for iteration=1,2,... do
    for actor=1,2,...,N do
        Run policy π_θold in environment for T timesteps
        Compute advantage estimates Â_1,...,Â_T
    end for
    Optimize surrogate L wrt θ, with K epochs and minibatch size M ≤ NT
    θ_old ← θ
end for
```

Figure 8: PPO Pseudocode

Each iteration, each of N (parallel) actors collect T timesteps of data. Then we construct the surrogate loss on these NT timesteps of data and optimize it with minibatch SGD for K epochs.

To speed up the project implementation, instead of implementing the model from scratch, the team leveraged on a pre-built model called PPO2 from an RL package named Stable Baselines, which are a set of improved implementations of Reinforcement Learning (RL) algorithms based on OpenAI Baselines.

Compared with the original PPO algorithm, PPO2 contains several modifications from the original algorithm: its value function is also clipped and advantages are normalized. PPO2 is the implementation of OpenAI made for GPU.

DQN

Because the problem is very complex, it is difficult to enumerate all possible state-action pairs in a Q-table manually like what is being done in Q-learning. DQN can be used to overcome this challenge to determine the possible actions given a state. DQN is a reinforcement learning algorithm that combines Q-Learning with deep neural networks to enable a reinforcement agent to work with complex, high-dimensional environments.

As Convolutional Neural Network (CNN) emphasizes more on feature extraction from data with spatial relationships (e.g. images) while Multiple Layer Perceptron (MLP) is more for tabular datasets. Thus, in this project, MLP (2 layers of 64) is used instead of CNN as the policy in the DQN.

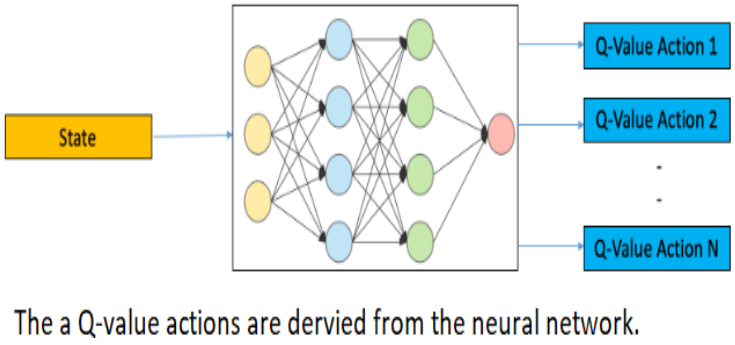


Figure 9: DQN Architecture

Problem Modeling

Action

In the stock market, there are 3 trading actions: “buy”, “sell” and “hold”. The actions can be differentiated into two categories: discrete and continuous. In discrete space, the transaction volume for each action is fixed (e.g. buy 500 shares). In continuous space, the transaction volume should be a range from nearly zero (0%) to the maximum of account balance (100%) for each action.

The team tested different agents with mainly three types of actions:

1. buy all, sell all, hold
2. buy 10% - 100% with 10% interval, sell 10 - 100% with 10% interval, hold
3. buy, sell with continuous range 0-100%, hold

In this project, SARSA, and DQN only supports discrete actions, it is not possible with continuous action space (Hill, Raffin, Ernestus, Gleave, & Kanervisto, 2020). Action Type 1 is used by SARSA and Action Type 2 is used by DQN.

Actor-Critic and PPO2 supports both discrete and continuous actions. The team used Action Type 1 for Actor-Critic and Action Type 3 for PPO2.

Observation

The observation spaces of the stock market can be constructed differently in different reinforcement learning algorithms.

SARSA

For all features, the agent calculates the percentage of change in numbers from the previous state to the current state. To cover all conditions, observations are classified into 15 buckets, shown in Figure 10.

Percentage	<=-20%	<-10%	<-5%	<-4%	<-3%	<-2%	<-1%	<0%
Bucket	-20	-10	-5	-4	-3	-2	-1	0
Percentage	<1%	<2%	<3%	<4%	<5%	<10%	>=10%	
Bucket	1	2	3	4	5	10	20	

Figure 10: Percentage and Bucket

Actor-Critic

The observation space is the fluctuations of the stock market (e.g. the stock price is increasing, decreasing or remaining unchanged in combination with current account balance). The actor uses the n lookback days (defined by the window variable) to observe the fluctuations and predict the next day's action.

DQN & PPO2

The previous 5 days' stock values and the current account information: Account Balance, Max Net Worth, Shares Hold, Total Sold Shares, Total Potential Sales Values are used to construct the observation spaces.

Reward

SARSA

To differentiate the agent's trading strategy against buy & hold, buy and sell will be rewarded (or punished). The action hold will be excluded from the rewarding function. On top of that, to magnify the effect on trade price difference, our team used 10 as a multiplier over the difference between sell price and buy price. As shown in Table 1, price difference is used to calculate the reward at each action.

Action	**Reward Equation**
Hold	R = 0
Sell	$R = (P_t - P_{buy}) * 10 + (P_t - P_{t-1})$
Buy	$R = P_{t-1} - P_t$

Table 1: Reward Mechanism

Actor-Critic

If the action is to buy new stocks or hold the stocks, there is no reward.

If the action is to sell and the profit is positive, the reward is profit = current_selling_price*share - price_bought*share - commission_cost. Otherwise, the agent will not get any reward.

The cumulative reward is the total profits earned over time in an episode. (The termination condition is to finish all the training data or test data given or when the account balance turns negative.)

The reward mechanism can be improved to encourage buying to prevent the agent from holding the stocks most of the time (or even attempting not to buy at all, just take action hold all the time from beginning to the end so the reward is not negative).

DQN & PPO2

Different rewards are designed to figure out the most effective incentive for reinforcement learning agents to learn.

One simple reward is purely based on the profit. The agent will get a reward if the profit is positive. Similar reward is also set up with net worth, account balance.

Another type is based on a delay modifier to make sure the reward is low in the beginning to incentivize profit sustained over long periods of time. This delay modifier was tested on account balance, profit and net worth.

The team also added a high penalty to avoid getting losses in all periods as people are normally reflected with a systematic human bias: loss aversion, e.g. +1 if the profit is positive, -100 if profit is 0 or negative.

3.0 Performance & Validation

SARSA

Our team selected stock data of SPY between 2019 and 2020 as a testing period. The reinforcement learning agent guided by Q-table is benchmarked against the buy & hold (B&H) strategy. The results produced by agent and B&H strategy share similar trends over time. But the B&H strategy always makes more profits than the agent. The observations are illustrated by figure 11.

Figure 11: Result of whole testing period

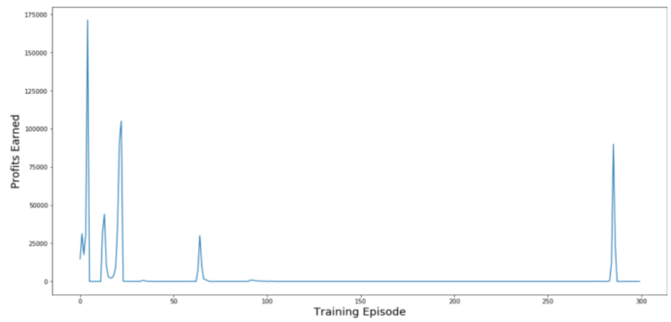
There is still room for the agent to improve. Factors like moving average (MA) and number of shares on hold can be included in the environment. MA makes the agent more farsighted and thus get a better feeling of the current state. On the other hand, by taking the number of shares on hold into account, the agent may learn to sell all shares before price drop happens.

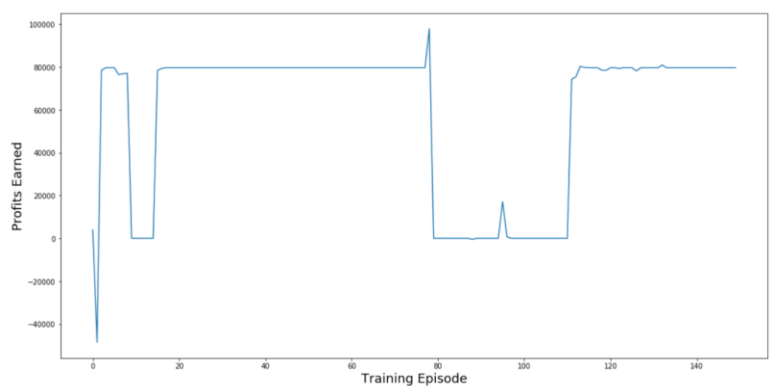
Another important factor to consider is the choice of training data. Price and trade volumes may differ significantly between different time spans. A more sophisticated model may even assign weights to price factor and time recency.

It is also critical to take cautions when customizing reward mechanisms according to the training and testing data. In the period of constant price rising, the B&H strategy is likely to beat all agents that trade actively. Under such situations, it is better to reward more on hold action. Furthermore, if an agent can learn the overall trend through long term MA, it can switch its strategy towards B&H in order to maximize profit in those periods.

Actor-Critic

There are 2 scenarios tested under actor-critic method.

<table>
<tr><th>Scenario</th><th>Performance of All Training Episodes</th></tr>
<tr><td>Scenario 1:
*There is no amount limit.
*There are no service charges for stock transactions.
*The number of shares to buy in each transaction is = 1.
*An episode terminates when all given data are tested.
*The training step = 1 day.
*The training takes more than half a day because the agent is required to test all the data samples in each episode.</td><td> 

[14783.087698, 31274.237371999978, 17471.836923000003, 30648.775684000004, 171297.39130500008, -7.794305000000207, 16.913004000
000072, -1.3124920000000344, -1.9868780000000044, 1.2699970000000036, 6.981860000000154, -56.662846999999914, 32830.73176700000
5, 44085.147555, 10872.486230000002, 3172.3543629999995, 2363.123108, 2283.2725, 4052.840018000001, 9281.599387000002, 35610.09
5003999995, 91506.05383400006, 105167.61052599995, 251.03313499999882, 71.84452599999975, 59.09137300000013, 30.99311099999987,
-1.961911999999984, -14.219397999999984, 81.38380399999978, -7.680537000000214, 25.96297000000031, 57.18175799999942, 468.59051
00000012, 780.8864070000017, 196.61037399999975, 111.26375300000012, 65.82433399999995, 28.474321000000174, -15.57190099999991
2, 1.0769180000000063, 1.1712329999999724, 0.9331139999999891, 7.338127, 6.187470999999974, -6.611215999999704, 0.390649000000
0389, -0.10500399999999388, 0.061889000000007854, -4.566232000000028, 6.551887000000008, -5.450622999999979, 6.93937499999997,
-1.622520000000086, -4.109381999999954, 25.141934999999975, 14.630594000000003, 6.260505999999964, 43.110769000000104, 22.80239
8999999795, 36.494505000000009, 80.7217880000002, -70.02898800000033, 7949.331748999995, 29917.231924000018, 10448.473775999997,
1723.0506050000004, 1123.501878, 693.894997, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 88.625, 0, 0, 0, 61.552498, 0, 0, 0, 72.6875,
122.742493, 882.8900060000001, 754.7731249999999, 836.467521, 260.3125, 450.88001199999997, 259.563752, 227.162506, 150.840622,
165.032501, 0, 0, 222.082489, 88.032501, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 141.171249, 0, 65.53125, 0, 20.84375, 16.25, 0, 80.662498, 327.488739, 12111.00
995, 89925.74679299998, 23184.323288999993, -0.8130779999999902, 11.412511000000001, 6.875626000000025, 0.6725000000000136, -1.1
25, 0, 0, 0, 0, 0, 0, 0, 0]
The agent prefers not to make transactions, so the profit is not negative.</td></tr>
<tr><td>Scenario 2:
*There is an initial amount of $50,000 given. The agent is prohibited from buying stocks if the account balance is less than the stock price.
*There are service charges for stock transactions
*The number of shares to buy in each transaction = 100.
* An episode terminates when all given data are tested, or the account balance is negative.
*The training step = 1 day.
*The training time is reasonable because most of the episodes are terminated by negative account balances.</td><td> 

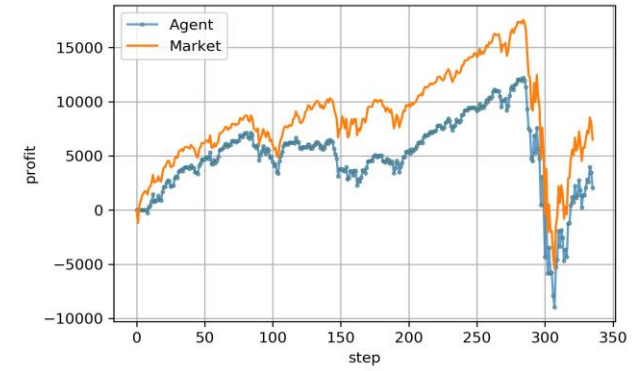
[3875.4671819999494, -48539.103494499956, 78533.53374949998, 79613.28125, 79613.28125, 79613.28125, 76451.6317485, 76837.617116
50001, 77079.5, 0, 0, 0, 0, 0, 0, 78390.5436865, 79341.0390625, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.2812
5, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.2
8125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 7961
3.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 7
9613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.2812
5, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.2
8125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 97884.07624500006, 0, 0, 0, 0,
0, 0, 0, 0, 0, -476.5674995000085, 0, 0, 0, 0, 0, 0, 17013.02899800001, 695.5860014999998, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 74255.06732450002, 75418.37262399995, 80331.3721885, 79739.31024900002, 79613.28125, 79613.28125, 79613.28125, 78527.5917
505, 78464.270243, 79613.28125, 79613.28125, 79287.9487485, 79613.28125, 79613.28125, 79613.28125, 78140.88825600002, 79613.281
25, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 80891.96674050002, 79613.28125, 79613.28125, 79613.28125, 79613.28125,
79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.28125, 79613.2812
5, 79613.28125, 79613.28125, 79613.28125]
The agent tends to buy and hold so it will get more rewards.</td></tr>
</table>

In scenario 1 because there are fewer restrictions (e.g. no-account balance and no service charges), the agent tends to hold the stocks (with no transactions at all or very limited number of transactions). Therefore, for scenario 1, there are usually no profits or only low profits received from the test data.

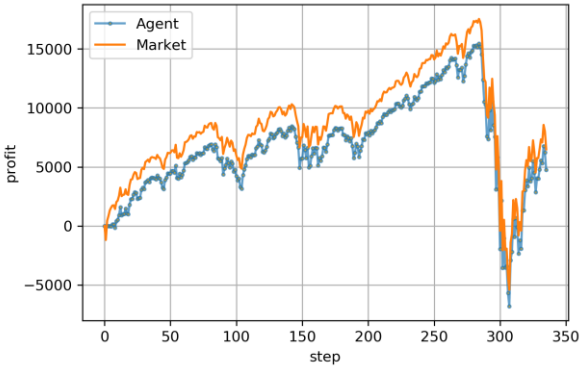
In scenario 2, when there are more restrictions added, the agent tends to buy and hold because the policy gives it more rewards. But the policy causes the agent to make a loss from the test data. This is because the training data has a very wide time range from 1993-2018 and it is wise for the agent to buy at the low points (e.g. in 1993 at about $40++), hold for a number of years and sell at the high points (e.g. in year 2018 at about $200+). But the test data has a very short time range from 2019 to 2020, therefore the policy is not working well. One way to overcome this problem is to use an equal number of data samples in both training and test sets (e.g. training dataset from year 2017-2018, test dataset from year 2019-2020). In this case, different data sizes are used because the aim is to make the agent run continuously with no terminal state and to be as close to real environments as possible (because there is no terminal state to the stock market in the real environment).

However, no matter how well a model is trained using past data, the model might still fail in the real environment because stock market prices are highly unpredictable and volatile and there are no consistent patterns in the data that allow agents to model stock prices over time near-perfectly. Therefore, the automated trading agent can only be used as a reference to help traders in their transactions.

PPO2



a) 20,000 training steps



b) 40,000 training steps

Figure 12: Profit Reward with Delay Modifier and Previous 5 Days Data

In the PPO model, it is observed that, when there are fewer training steps, the agent tends to buy and sell more frequently. When there are more training steps, the agent prefers B&H strategy. This means the training step is an important parameter. Only when there are enough training samples, then the agent can discover the correct trading strategy from the dataset.

Two tests were conducted with one using account balance as a reward signal and another with profits as a reward signal. The two tests produced similar results.

Here is the quick summary at the end:

- 20,000 training steps - Shares held: 184 (Total sold: 1905) Total commission costs: $1,180 Profit: $2,039.00.
- 40,000 training steps - Shares held: 193 (Total sold: 0), Total commission costs: $49 Profits: $4,740.00.

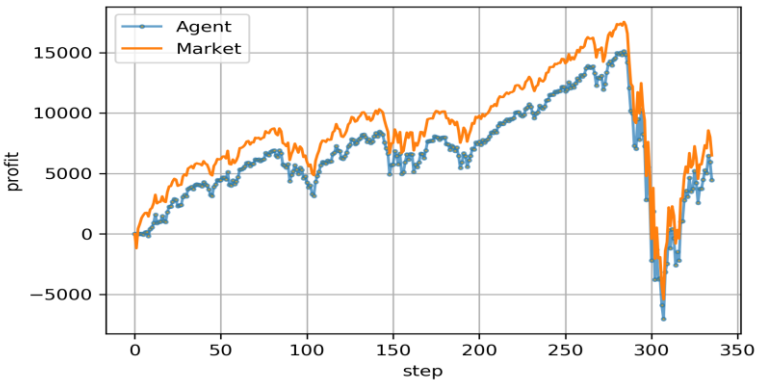
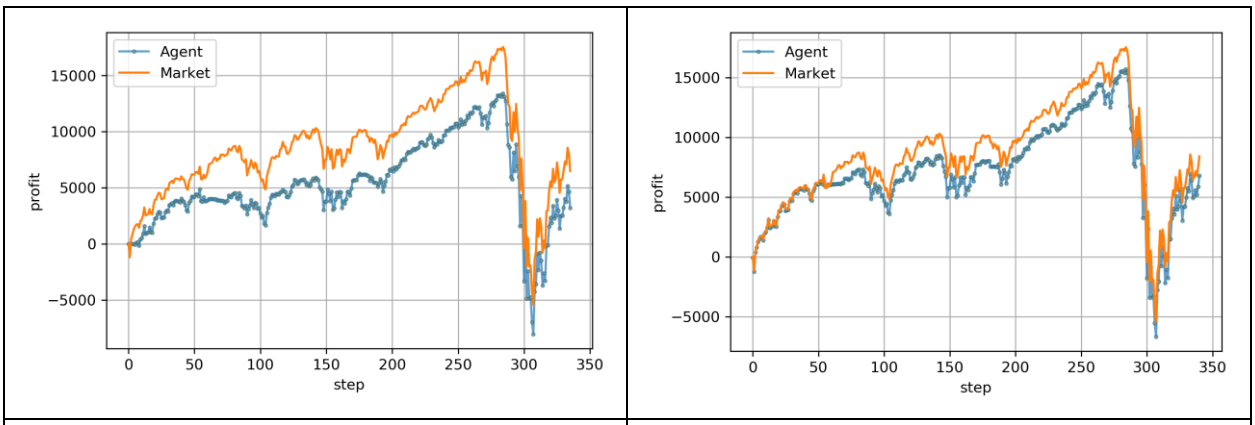


Figure 13: Profit reward without delay modifier - Shares held: 192 (Total sold: 193) Total commission cost: $165.00; Profit: $4,454.00

Then the influence of the delay modifier on overall performance is examined. Figure 13 shows a profit reward without delay modifier which is quite similar to Figure 12 which shows a profit reward with delay modifier. This means the delay modifier has limited impact on the overall performance. But without a delay modifier, the agent will try to trade more in the beginning.



a) previous 5-days data	b) previous 1-day data

Figure 14: Profit Reward and High Loss Penalty with Delay Modifier with Different Observation (5 days vs 1 day)

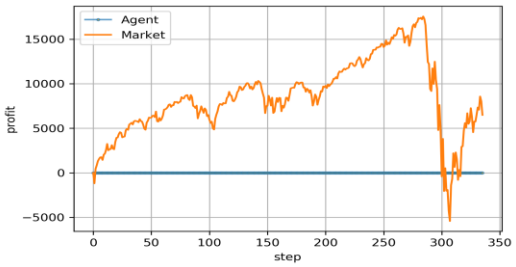
5 Days - Shares held: 188 (Total sold: 1893) Total commission cost: $1,176.00 Profits: $3,202.00

1 Days - Shares held: 194 (Total sold: 1038) Total commission cost: $671.00 Profit: $6,800.00

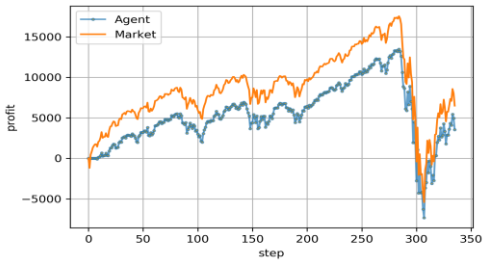
Moreover, our team created 2 models based on different look back windows: 5 days data vs 1 day of historical data, to observe the influence of the look back window on the overall results. Surprisingly, the data from 1-day get a closer trend to B&H shown in Figure 14. The summary showed based on one day more profits were achieved in the end. A possible reason is the 5-days agent can only start trading when there are 5-days' data, this consequently affects the buying

price as in Figure 14 the starting price is lower than following days. The 1-day agent thus gets an advantage on buying the stock earlier at lower prices.

DQN



a) 40,000 training steps



b) 4,000 training steps

Figure 15: Profit reward with delay modifier with different training steps on DQN and previous 5 days data

When we applied the similar setup on DQN with 40000 training steps, the result was a stable line. To figure out the reason, different parameters were configured and finally, it was found the training steps can be the factor. If the training steps were decreased, a trend similar to other reinforcement learning agents was presented. A possible explanation is the ultimate strategy based on profit reward is to hold the account balance in the volatile stock market. Similar result was also presented when using 40000 training steps in PPO2 with 1-day data.

Here is a quick summary on Figure 15:

a) Shares held: 0 (Total sold: 0) Total commission cost: $0.00. Profit: $0.00.

b) Shares held: 182 (Total sold: 0) Total commission cost: $58.00 Profits: $3,550.

4.0 Findings & Possible Improvements

Genetic Algorithm

The team has also tried Genetic Algorithm (GA) together with Regression Neural Network (RNN) to predict stock price and use that result to guide the reinforcement learning agent. In the work *Stock Market RNN Genetic Algorithm* (Campos, 2020), the author uses a deep regression network to learn features from stock statistics and another genetic algorithm to evolve this network. Table 2 shows elements used by the GA to configure the network.

Network Component	**Available Elements**
Number of hidden layers	Range (0, 11)
Type of hidden layer	Function: ['Dense', 'GRU', 'LSTM'], Activation: ['selu', 'relu', 'tanh', 'linear'], Number of Neurons: Range (1, 11)
Learning optimizer	['adam', 'adagrad', 'nadam', 'rmsprop']

Final Activation	['selu', 'relu', 'tanh', 'sigmoid', 'linear']

Table 2: Reward Mechanism

At beginning, there is a population of 20 networks and each of them is initialized with a random configuration of above elements. All networks will be trained using the same training data and are measured by their loss value. At the end of each iteration, top performers are chosen to create the next generation by randomly remixing network components from the parents. There is a certain chance that a mutation will be involved in the process also. After 5 generations, the final population will be evaluated, and the best model is selected.

Remarkably, training time of the first generation lasts more than 3 hours on a student's computer. Although it is generally a good approach, the team decided not to proceed with GA + RNN in this study due to the time constraint.

LSTM Model for Stock Prediction

In this project, Long Short-Term Memory networks (LSTM) is also explored to study its performance on stock predictions (a time series prediction problem). The intention was to incorporate the LSTM model to reinforce learning agents to improve their ability to predict the next state. However, due to time constraints, the team will leave the change as future improvement.

Findings

These are our findings in the project:

1. Stock market is highly complex and could be impacted by many macro/micro economy factors, simple reinforcement learning based trading algorithms (leveraging on purely daily trade information) cannot fully reflect the market behavior, as a result, the actions of reinforcement learning agents may not be the optimal choices.

2. Training dataset and test dataset do not necessarily have correlation, there could be a chance that the testing data happens to have completely different patterns due to certain unprecedented events. (e.g. COVID-19)

3. The transaction cost: Reinforcement learning agent trades more frequently compared with buy and hold (only transact once at day 1), the longer period, the more transaction cost will occur, which will further lower down the net profits.

4. Environment setting such as reward may affect the reinforcement learning agent's decision too. For example, we have tested a few reward settings (Profit: 1, Flat & loss -100, Profit 100, Flat 0, Loss -100, Profit: 20, Flat: 0, Loss -100), it turned out that the first option gave the best performance. In order to have the optimum settings, extensive trial-and-error tuning needs to be carried on.

5.0 Challenges

Environment Unpredictability

The stock market is a very open and dynamic environment and it can be influenced easily by many factors like government policies, global news, economic environment, etc. In our model, only 6 features (e.g. opening price, closing price). To get a better result, more factors (e.g. political news sentiments, economic outlooks indicators) will be needed.

Defining A Precise Reward Function

Although the reward target is to get as much profit as possible. However, if only using the immediate profit or long-term profit without other modifiers, the agent may give some unexpected action like holding the stock after a certain point.

Data Problem and Exploration Risks

Reinforcement learning might require even more data than supervised learning. The point was highlighted by computer scientist and entrepreneur Andrew Ng during his speech at the Artificial Intelligence Conference in San Francisco 2017: “It is really difficult to get enough data for reinforcement learning algorithms. There’s more work to be done to translate this to businesses and practice” (Lytvynova, 2019).

6.0 Conclusion

In this study, the team uses 4 different models (SARSA, DQN, Actor-Critic and PPO) to train the stock agent to help it to make intelligent trading decisions. Most of the models prefer the buy and hold strategy because the data range given is very wide (more than 20 years of data) and the strategy can indeed generate the highest profits. This means the learning agent processes some capabilities to make reasonable decisions that simulated human beings’ behaviors. However, the models are still not perfect because there are many uncertainties in the stock market, the models can incorporate more features derived from Natural Language Processing (for market sentiment detection) to improve its strategy.

7.0 References

Amadeo, K. (2020, March 13). *The S&P 500 and How It Works*. Retrieved from The Balance: https://www.thebalance.com/what-is-the-sandp-500-3305888

Anspach, D. (2019, January 28). *Why Average Investors Earn Below Average Market Returns*. Retrieved from The Balance : https://www.thebalance.com/why-average-investors-earn-below-average-market-returns-2388519

Bakker, B. (2002, January). *Reinforcement Learning With Long Short-Term Memory*. Retrieved from

https://www.researchgate.net/publication/2395590_Reinforcement_Learning_with_Long_Short-Term_Memory

Campos, H. (2020, May 20). *Stock-Market-RNN-Genetic-Algorithm*. Retrieved from Github: https://github.com/HanCamp/Stock-Market-RNN-Genetic-Algorithm

CI, T. (2020, May 22). *Manual*. Retrieved from Github: https://github.com/ccxt/ccxt/wiki/Manual#exchange-structure

Ciaburro, G. (2018). *Keras Reinforcement Learning Projects.* Packt Publishing.

Deshpande, M. (2017, December 02). *An Overview of Reinforcement Learning: Teaching Machines to Play Games*. Retrieved from Python Machine Learning: https://pythonmachinelearning.pro/an-overview-of-reinforcement-learning-teaching-machines-to-play-games/

Duerson, S., Saleem, F., Kovalev, V., & Malik, A. H. (2005). *Reinforcement Learning in Online Stock Trading Systems*. Retrieved from Semantic Scholar: https://www.semanticscholar.org/paper/Reinforcement-Learning-in-Online-Stock-Trading-Duerson-Khan/be8e61fe568712c799219fb612d190b4e62642ae

Forbes Technology Council. (2019, December 26). *AI Is Trading's Next Key Differentiator*. Retrieved from Forbes: https://www.forbes.com/sites/forbestechcouncil/2019/12/26/ai-is-tradings-next-key-differentiator/#6800de4f425c

Frankel, M. (2017, March 20). *How Much Has Warren Buffett Beaten the Market By?* Retrieved from The Motley Fool: https://www.fool.com/investing/2017/03/20/how-much-has-warren-buffett-beaten-the-market-by.aspx

Hill, A., Raffin, A., Ernestus, M., Gleave, A., & Kanervisto, A. (2020, May 21). *Reinforcement Learning Tips and Tricks*. Retrieved from Stable-Baselines Documentation : https://stable-baselines.readthedocs.io/en/master/guide/rl_tips.html

John Moody, M. S. (1999). *Reinforcement Learning for Trading*. Retrieved from NIPS Processings: https://papers.nips.cc/paper/1551-reinforcement-learning-for-trading.pdf

Kinf, A. (2019, October 15). *Trade and Invest Smarter — The Reinforcement Learning Way*. Retrieved from Towards Data Science: https://towardsdatascience.com/trade-smarter-w-reinforcement-learning-a5e91163f315

King, A. (2019, April 28). *Creating Bitcoin Trading Bots Don't Lose Money*. Retrieved from Towards Data Science: https://towardsdatascience.com/creating-bitcoin-trading-bots-that-dont-lose-money-2e7165fb0b29

Little, K. (2020, February 04). *The Major Types of Risks for Stock Investors*. Retrieved from The Balance: https://www.thebalance.com/major-types-of-risk-for-stock-investors-3141315

Lytvynova, K. (2019, May 14). *Reinforcement Learning Explained: Overview, Comparisons and Applications In Business*. Retrieved from Top Bots: https://www.topbots.com/reinforcement-learning-explained-business-applications/

Meng, T. L., & Khushi, M. (2019). *Reinforcement Learning in Financial Markets*. Retrieved from Data 2019: https://www.semanticscholar.org/paper/Reinforcement-Learning-in-Financial-Markets-Meng-Khushi/d8a545a02aa0e0cbe42607212433b2b4f446e5ab

Nachum, O., Norouzi, M., Xu, K., & Schuurmans, D. (2017, Nov 22). *Bridging the Gap Between Value and Policy Based Reinforcement Learning*. Retrieved from Papers: https://arxiv.org/abs/1702.08892

National Research University Higher School of Economics. (2020, May 22). *Policy-based vs Value-based*. Retrieved from Coursera: https://www.coursera.org/lecture/practical-rl/policy-based-vs-value-based-INklC

Premkumar, D. (2020, March 12). *Why do Stock Markets Exist? And Why is it so Important?* (Trade Brains) Retrieved May 23, 2020, from https://tradebrains.in/why-do-stock-markets-exist/

Saito, S., Wenzhuo, Y., & Shanmugamani, R. (2018). *Python Reinforcement Learning Projects: Eight hands-on projects exploring reinforcement learning algorithms using TensorFlow.* Packt Publishing .

Salloum, Z. (2019, Feburary 9). *Policy Based Reinforcement Learning, the Easy Way*. Retrieved from Towards Data Science : https://towardsdatascience.com/policy-based-reinforcement-learning-the-easy-way-8de9a3356083

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017, August 28). *Proximal Policy Optimization Algorithms*. Retrieved from arXiv.org: https://arxiv.org/abs/1707.06347

Sharma, S. (2018, August 5). *https://towardsdatascience.com/policy-networks-vs-value-networks-in-reinforcement-learning-da2776056ad2*. Retrieved from Toward Data Science: https://towardsdatascience.com/policy-networks-vs-value-networks-in-reinforcement-learning-da2776056ad2

Simonini, T. (2018, July 26). *An intro to Advantage Actor Critic methods: let's play Sonic the Hedgehog!* Retrieved from Free Code Camp: https://www.freecodecamp.org/news/an-intro-to-advantage-actor-critic-methods-lets-play-sonic-the-hedgehog-86d6240171d/

Sutton, R. S., & Barto, A. G. (2015). *Reinforcement Learning: An Introduction .* London: The MIT Press.

Token, 1. (2020, May 21). *1 Token*. Retrieved from 1 Token : https://1token.trade/

trading, R. L. (2020, May 20). *Yuqin Dai, Chris Wang, Iris Wang, Yilun Xu*. Retrieved from MS&E 448 - Big Financial Data for Algorithmic Trading: http://stanford.edu/class/msande448/2019/Final_reports/gr2.pdf

Yahoo Finance. (2020). *SPDR S&P 500 ETF Trust (SPY)*. Retrieved from Yahoo Finance https://finance.yahoo.com/quote/SPY/history/