Summer Research Program in Industrial and Applied Mathematics

Sponsor

⟨Magnum Research Limited⟩

Final Report

⟨Portfolio Management using Reinforcement Learning⟩

Student Members

⟨SEO Dayoung⟩ (Project Manager), ⟨*SNU*⟩,
⟨`multiply@snu.ac.kr`⟩
⟨PARK Junggil⟩, ⟨*SNU*⟩
⟨WONG Singlam⟩, ⟨*HKUST*⟩
⟨YANG Yuwei⟩, ⟨*CityU HK*⟩

Academic Mentor

⟨Avery Ching⟩, ⟨`maaching@ust.hk`⟩

Sponsoring Mentors

⟨Joseph Chen⟩, ⟨`joseph.chen@magnumwm.com`⟩
⟨Don Huang⟩

⟨Date: August 8, 2018⟩

Abstract

In this project, we use Q-learning and deep Q-network to train agents that manage a stock portfolio of two stocks. We defined a state as the stock price change history together with the portfolio, and an action as a modification of the weight between two stocks, and a reward as portfolio value change or sharpe ratio. In most cases Q-table or neural networks performed better than the 50-50 reevaluate benchmark during the training session. But when we use Q-table and neural networks to test on unknown datasets, they are not as good as the benchmark.

Acknowledgments

We are highly indebted to Dr. CHING Avery, Dr. CHEN Joseph and Dr. HUANG Don for their guidance and constant supervision as well as for providing necessary information regarding the project. Thanks Dr. KU Albert Yin-Bon, Pf. LEUNG Shing-Yu and Pf. KOOK Wong to hold this meaningful program and the support from Mathematics Department of HKUST and SNU in completing the project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The investors' ultimate goal is to optimize profit or risk-adjusted return in trading system. Investors construct a portfolio for hands-off or passive investment. A portfolio, a collection of multiple financial assets such as stocks, bonds and bills is usually characterized by its constituents(assets included in a portfolio), weights(the proportion of the total budget invested into each asset), and expected return. Portfolio management is the art and science of making decisions about investment mix and policy, matching investments to objectives, and balancing risk against performance. In this research, we are using reinforcement learning methodologies to optimize portfolios.

Our research is supported by Magnum Research Limited. It is a fintech company aiming to use advanced AI techniques to help different types of investors to build up personalised portfolio to optimize their profits in the financial market.

In this project, we are using reinforcement learning to develop automated trading strategies. The performances of the algorithms are indicated by comparing to a benchmark. To simplify our situation we just consider 2-stocks portfolio. We also assume that we can buy and sell stocks at the closed price and our behaviour does not affect the stock market. To solve the problem in the setting of reinforcement learning setting the main methodologies we used in the report are Q-learning and Deep Q-network.

Reinforcement learning is a learning strategy which let agents learn without knowing the rules of the environment beforehand. Only the rewards of actions are needed. The objective is to optimize the total reward it gets, and through the process of exploration and exploitation, the agent will learn gradually.

Reinforcement learning has been applied to many situations. Especially for deep reinforcement learning, which applies the techniques of deep learning to let reniforcement learning to slove a wider range of problems.[1] The most famous one should be a modified version of the well known AlphaGo[1], and AlphaGo Zero. It involves the technique of deep reinforcement learning. Another example is that using deep reinforcement learning, we can train the computer to play Atari games[2].[4] With wide range of applications of reinforcement learning in their minds, people proposed to apply reinforcement learning in financial sector especially for portfolio management.

[1]AlphaGo is a copmuter program that able to win the world class Go player. On 23,25,27 May 2017 AlphaGo win the first ranked Go player Ke Jie

[2]Google DeepMind is able to do this in 2013

Some attempts were made to tackle this problem. The work of Jin & EI-SAAWY [3] is similar to ours in this project. They approached the problem by using deep q-network but we also try using Q-learning in our project. They considered the sharpe ration as one of the factors which is also what we do in Q-learning. Another related work is Jiang, Xu and Liang[2] where they used more advanced techniques like Convolutional Neural Network (CNN), a basic Recurrent Neural Network (RNN), and a Long Short-Term Memory (LSTM). Their problem setting is in the cryptocurrency market instead of the stock market which is our objective.

Chapter 2

Methodology for Q-learning and Deep Q Network

The main methodologies we are using for the research are Q-Learning and Deep Q Network(DQN). Here are some explanations about those methodologies.

2.1 Q-learning

Q-Learning is one of the Reinforcement Learning algorithms that attempts to learn the value of being in a given state, and taking a specific action there. Q-table is a table of values for every possible state(row) and action(column) in the environment. Within each cell of the table, we learn a value measuring how good it is to take a given action at a given state. We start by initializing the table to be all zeros, and then we update the table as we observe the rewards by taking various actions

When updating the Q-table, Bellman equation is used. The concept of Bellman equation is that the expected long-term reward for a given action at a given state is equal to the immediate reward gained from the current action plus the expected reward from the best future action taken at the following state. Q-table is used to estimate the long-term reward. As shown below, Q-value for a given state(s) and action(a) should equal to the current reward(r) plus the maximum discounted(γ) future reward expected according to Q-table for the next state(s) we obtain.

$$Q(s,a) = r + \gamma(\max(Q(s,a))$$

In a Q-learning training process, we first check whether a state already exists in the q-learning table. If it exists, we choose the action that has the largest Q-value. If not, we build a new empty line of this state and randomly choose an action. Then we update the value in the Q-learning table by Q-table learning formula .

To use a Q-learning model, we need to discretize the states and actions. We discretize actions it by choosing them as the weight of stock A of the total portfolio value and up to one decimal. So we define the action vector as $(0, 0.1, 0.2, , 0.9, 1)$. For the states, we tried two models. We first use the pair of stocks' close prices, but it is not well suited as states in a Q-learning model. Therefore, we model the states to be the pair of stocks' close price changes up to two decimals. After that, we model states by another method. We use the slopes of the linear regression functions of the stocks' close prices in each fixed period(3 days, 10 days).

For the reward function, we began simply by using portfolio value changes as a rewards. However, as concerned in portfolio management, we should not only consider the profit, but also the risk. Thus we use sharpe ratio as the model's reward function.

2.1.1 Q-learning Algorithm

```
Initialising $Q(s,a)$ arbitrarily
Repeat (for each episode):
Initialise $s$
Repeat (for each step of a episode):
Choose $a$ from $s$ using policy derived from $Q$ (e.g. $\epsilon$ greedy)
Take action $a$, observe $r$, $s$
```

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a)) \tag{2.1}$$

$$s \leftarrow s' \tag{2.2}$$

Equation (2.1) and (2.2) are the update equations.

2.2 Deep Q Network (DQN)

Deep Q Network uses the techniques from deep learning to approximate Q-values, since in Q-learning both states and actions are discrete so that calculating and optimizing the Q-value are both time and memory consuming. The key is that we apply the deep neural network to approximate the Q-function. We know that neural network is used to find out the right weights by the back propagation process so it can be used to map all state-action pairs to rewards. One standard example for neural network is the Convolutional Neural Network (CNN).

Due to the problem of correlation between states and non-stationary targets, when we train the neural network, we store transition in memory M, and randomly sample mini-batchs from M and replay to solve the problem. Moreover, we separate the target networks and copy the network regularly to solve the problem of non-stationary targets.

```
Initialise replay memory $D$ to capacity $N$
Initialise action-value function $Q$ with random weights $\theta$
Initialise target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
For episode =1, $M$ do
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For $t = 1$,$T$ do
    With probability $\epsilon$ select a random action $a_t$
    otherwise select $a_t = argmax_a Q(\phi(s_t), a\ \theta)$
    Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibath of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $D$

      $Set\ y_j = \begin{cases} r_j, & \text{if episode terminates at step j+1} \\ r_j + \gamma max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-)) & \text{otherwise} \end{cases}$

    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to network
    parameter $\theta$
    Every C steps reset $\hat{Q} = Q$
  End For
End for
```

Chapter 3

Mathematical Setting

For an automated trading robot with reinforcement learning, investment decisions and actions are made periodically. We allocate fixed amount of budget into two stocks, aiming to maximize return while controlling the volatility and considering the transaction cost. These are the mathematical setting of the portfolio management problem.

3.1 Mathematical Formalism

Let $p_1, p_2, p_3, ..., p_t, ...$ be the close price sequences of two stocks respectively on each day released from the exchange center. Let (p_t, q_t) to be price vector. We use the price change vector $(\frac{p_t - p_{t-1}}{p_{t-1}}, \frac{q_t - q_t - 1}{q_{t-1}})$ to define as the states. pv_t is the portfolio value at the time epoch t, which is calculated based on the market value of two stocks and two stocks' weights in the portfolio. We define a_t as the weight of the first stock at the time epoch t in the portfolio value. It is calculated as the proportion of first stock's market value in the total portfolio value. We define the initial portfolio as \$ 10,000 which is called the budget from now on.

Rather than using the price change, we can define the tendency of a stock price by a regression line. Let close prices in n successive days be $p_1, p_2, p_3, ..., p_n$. If a regression line is $f(x)$, then the error of regression is $\epsilon = (f(i) - p_i)^2, i = 1, 2, 3, ..., n$. We can find a line $f(x) = a_n x + b_n$ that minimize the sum of error squared $\sum_{i=1}^{n}(f(i) - p_i)^2$, then the line $f(x)$ is the simple linear regression function of these n days close price. And for another stock, the simple linear regression function is $g(x) = k_n x + h_n$

3.2 Transaction Cost

In the real world, buying or selling stocks is not free. The cost includes commission fee, tax, etc. Assuming a transaction cost proportional to the stock market values exchanged in the market, we set the rate to be 0.2%. We used the preceding study by Angelos to determine the rate. Since we assumed the stock shares to be of float type, we used the formula below to calculate the transaction cost.

$$\frac{T}{2} = |\frac{pv_t}{p_t} a_{t-1} - \frac{pv_{t+1} - T}{p_t + 1} a_t| * p_t * rate$$

3.3 Benchmark

The model's test data performance was compared against a benchmark. We used the preceding study by Oliver and Hamza to determine the benchmark. It is called the rebalance benchmark, which reevaluates its holdings at the end of every market days, and buys or sells stocks to ensure the total portfolio value be splited into 50-50 between the two stocks. It is important to note that it maintains a proportion of stock values, not stock shares.

3.4 Dataset and Features

We trained our Q-table and neural network using historical stock data gathered from Yahoo finance using the Python library Pandas.DataReader to automatically download the stocks' histories.

Stock riskiness is quantified by beta index. Beta bigger than 1 indicates that a stock is more volatile than the market, while less volatile stock has a beta smaller than 1. We chose ten stocks from S & P 500's high-beta index fund, and five low-beta stocks from 2000/05/01 to 2001/05/01. Then we randomly chose two high-beta stocks from those ten, and one low-beta stock from those five to make up two stock pairs for test. For the other thirty two possible combinations made up by the remained stocks, we randomly chose ten pairs to train our model.

For testing, we also chose two stocks combination based on the beta indices. We tested on two combinations in order to reduce the impact of the unique behavior of the data. We used AMAT(1.29) and CAJ(0.78), and FCX(2.53) and CAJ(0.78) as test sets.

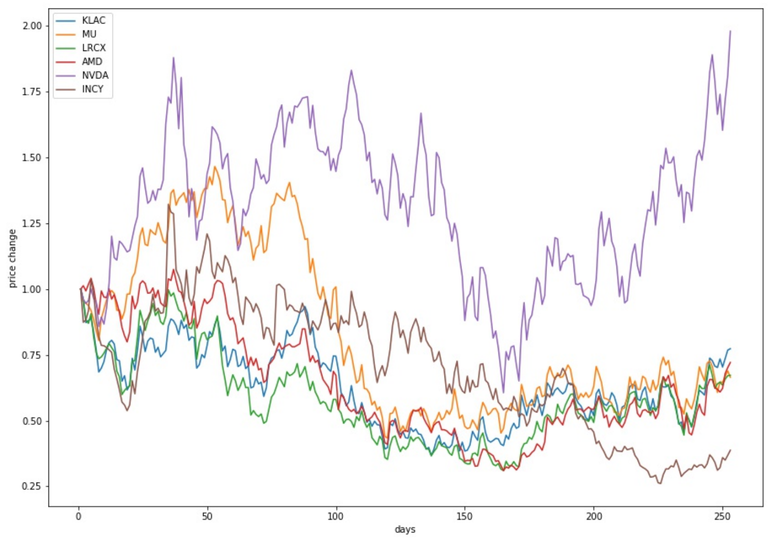


Figure 3.1: High Beta Stock

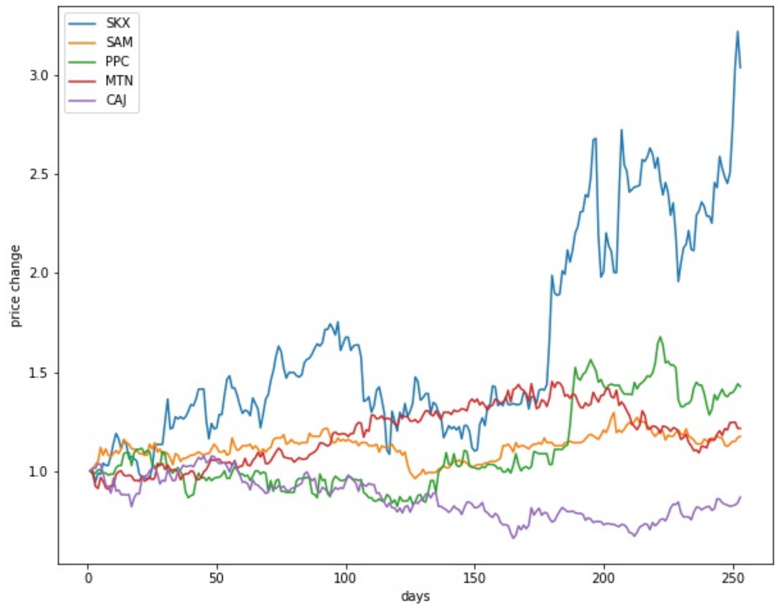


Figure 3.2: Low Beta Stock

3.5 Assumption

While building models for portfolio management, we made some assumptions. First, we buy and sell the stock at the close prices. Second, our behavior of buying and selling does not affect the stock market. Third, we can discretize the stock price and the stock unit to be bought and sold as non-integers. Lastly, the transaction costs of buying and selling are the same, and are proportional to the exchanged market value.

Chapter 4

Results and Discussion

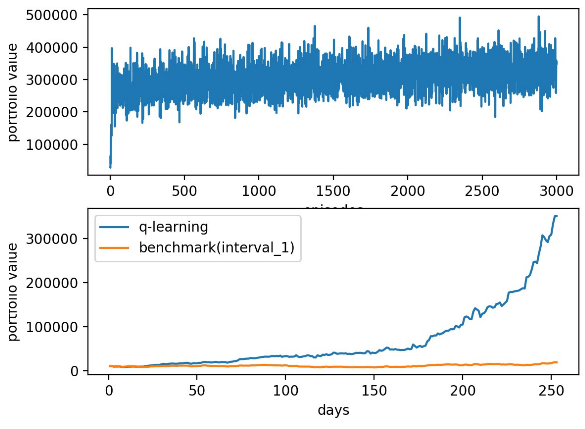
4.1 Q-Learning

4.1.1 The Basic Model

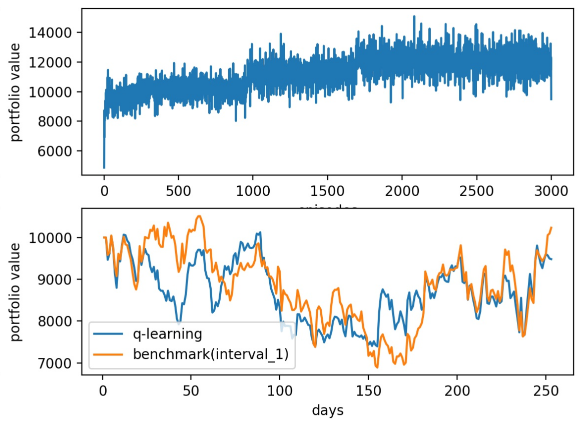
To build the basic model, we used the portfolio value change as the reward function, which means that the agent will choose the action that will increase the expected portfolio value the most. We used a price change pair up to two decimals as a state.

We adjusted the parameters to give better performance. The learning rate(α), which decides the impact of a new data on the existing Q-table, was set as 0.0001 based on experiment. The discount factor(γ), which discounts the sum of the future reward, was set as 0.9. The epsilon(ϵ) was set as 0.9 initially and it is decreased by 1% for every time period until it reaches 0.01. It is done to let the actions be chosen more randomly for exploration during the early state, since the Q-table does not contain much information. However later actions are more likely to be chosen based on the Q-table with a smaller epsilon. With those parameters, we trained the Q-table for 3000 episodes for each dataset.

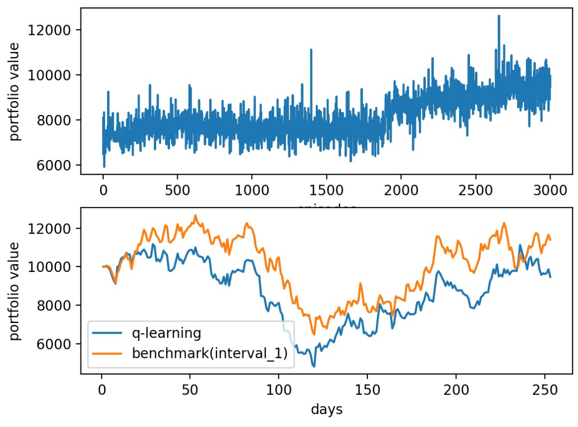
As we trained on 10 training datasets, the performance of the basic model on each dataset does not have clear patterns. Usually, the portfolio values of gains by the model were way better(3 4 times final portfolio value) than benchmark, but training results from the 4th, 7th, 8th, 9th training set were similar or even below that of the benchmark. Each plot below has two subplots. The upper plot shows how the final portfolio value changes over every episode, and using the last episodes data we drew bottom plot which shows the portfolio value change over days. Compared to those of 1 time training with KLAC and SKX data (a) and 10 times training with NVDA and SKX data (d), the results of 4 times training (b) and 9 times training (c) are not good. We did not find any clear reason behind this. The dimension of the Q-table is steadily increasing with more training as shown in figure 5.2.



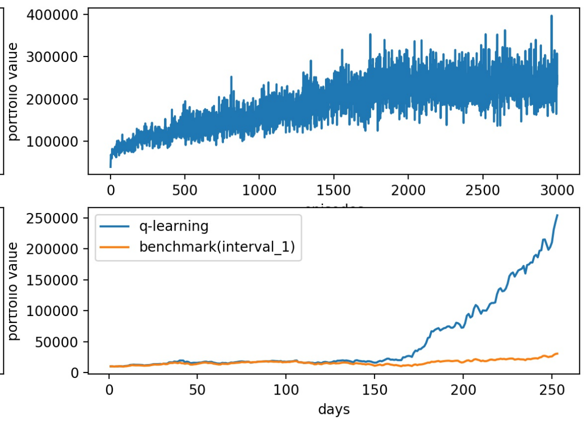
(a) Training 1 (KLAC & SKX)



(b) Training 4 (AMD & MTN)



(c) Training 9 (MU & PPC)



(d) Training 10 (NVDA & SKX)

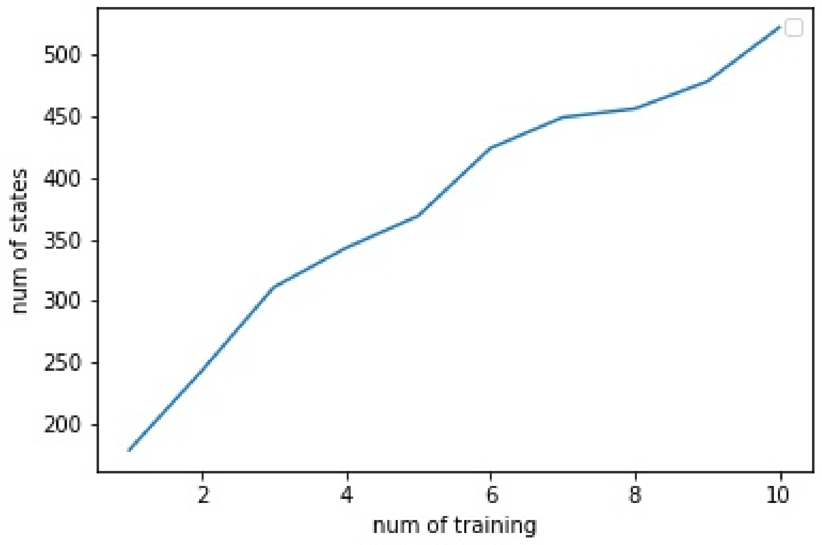


Figure 4.2: Dimension change over training

We trained, we tested on two test datasets(AMAT & CAJ, FCX & CAJ) using the Q-table. Test results using FCX & CAJ dataset showed a greater portfolio value. However more training did not guarantee better test results. For AMAT & CAJ, the best test result was obtained by using the Q-table trained with one dataset, and further training made the test results portfolio values decreased. On the other hand, for FCX & CAJ, the best test result was with the 8 times training Q-table, and before and after that the final portfolio value is below the benchmark. To check the reason why test results keep changing, we analyzed the actions taken in each result. Even the actions taken in the test using the 9 times training, and using the 10 times training differed a lot. Figure 4.9 is drawn using the FCX & CAJ data. The first subplot is comparing the actions taken with 5 and 8 times training, while the second is comparing 8 times and 10 times, and the last is comparing 10 times and 9 times.

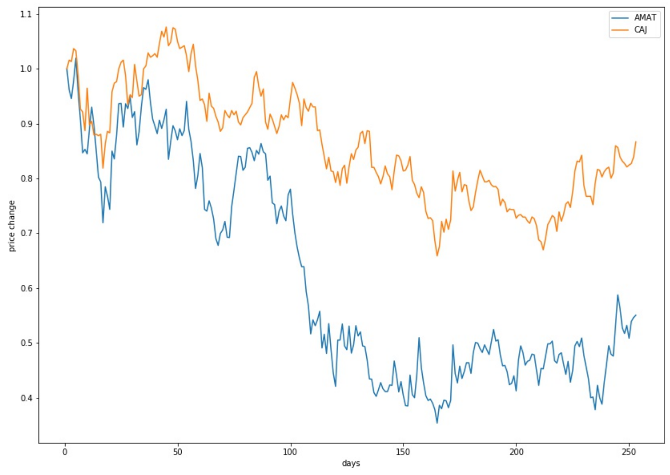
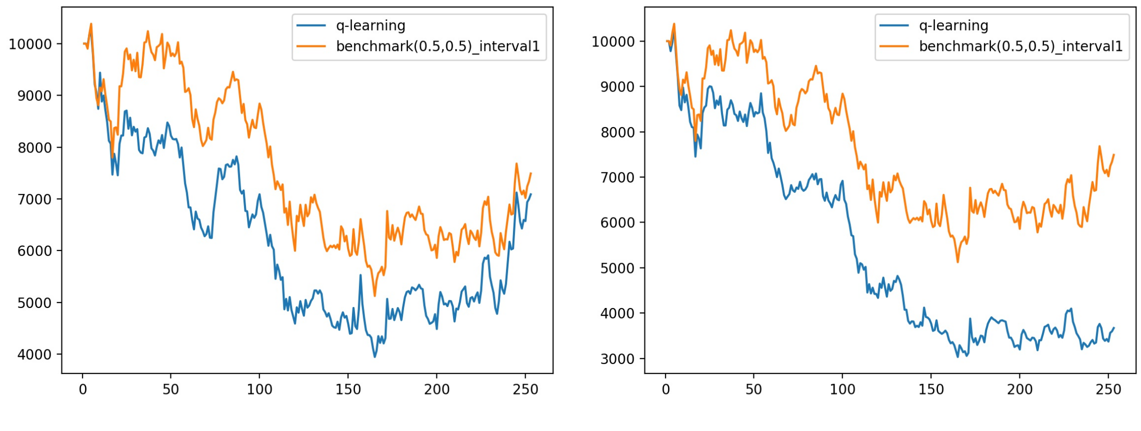


Figure 4.3: TEST SET1 (AMAT & CAJ) :PRICE CHANGE



(a) TEST SET1 (AMAT & CAJ):1 training (b) TEST SET1 (AMAT & CAJ):3 training

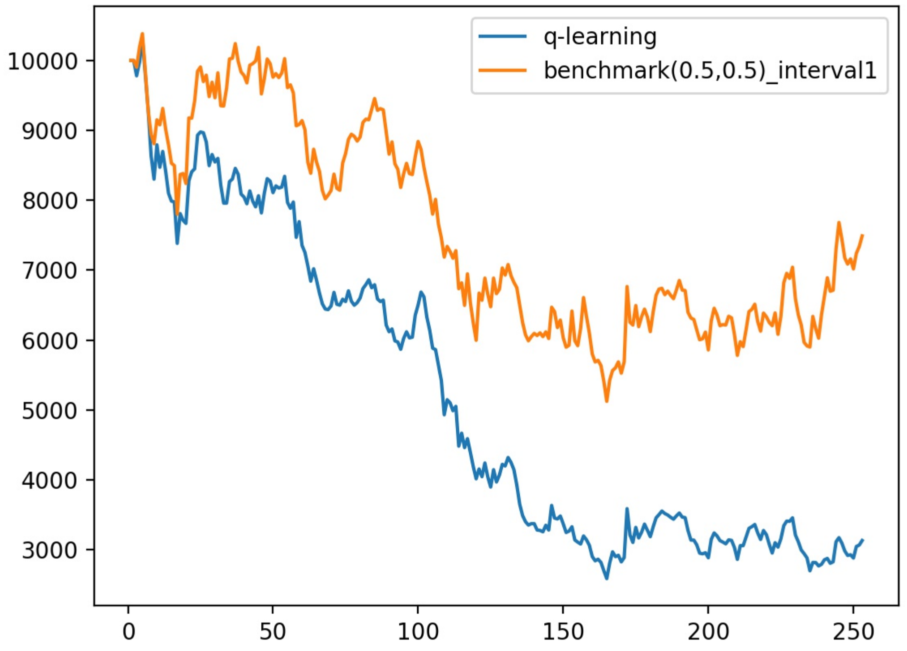


Figure 4.5: TEST SET1 (AMAT & CAJ) : result with 10 training Q-table

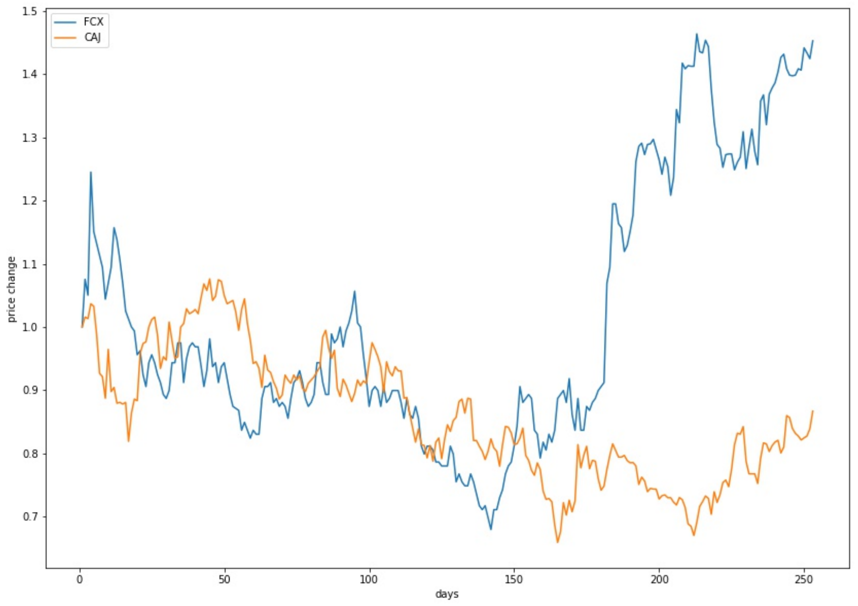
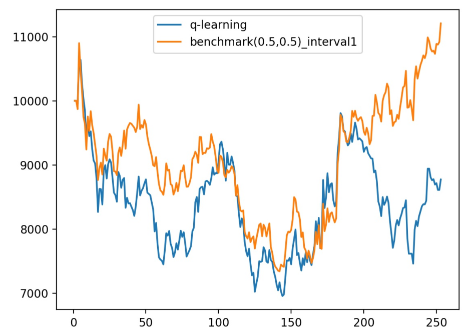
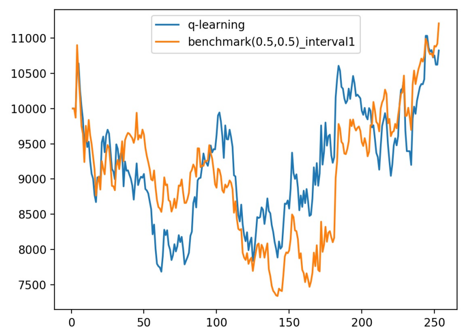


Figure 4.6: TEST SET2 (FCX & CAJ) : PRICE CHANGE



(a) TEST SET2 (FCX & CAJ): 5 training



(b) TEST SET2 (FCX & CAJ): 8 training

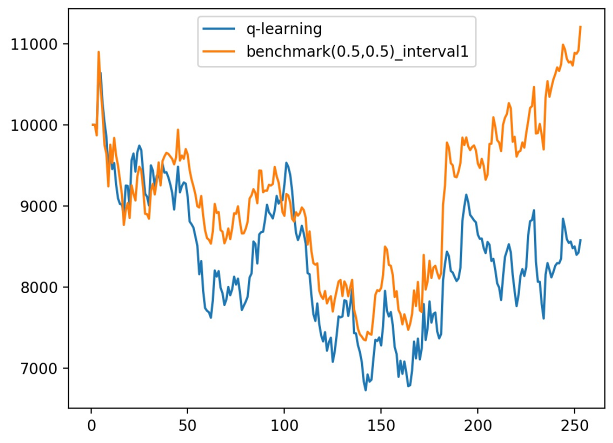


Figure 4.8: TEST SET2 (FCX & CAJ) : result with 10 training Q-table

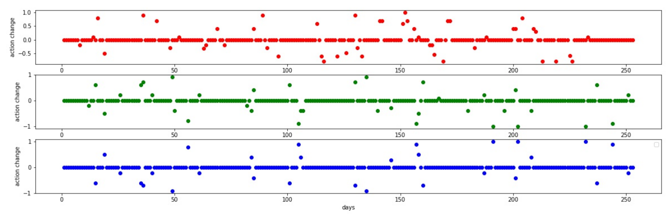
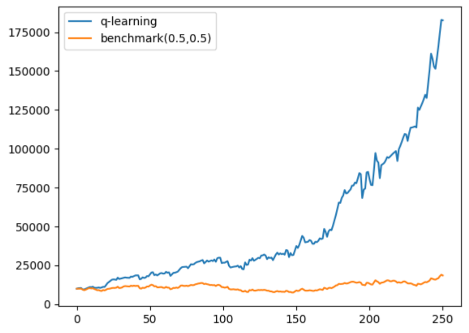


Figure 4.9: ACTION CHANGE : (8 vs 5, 8 vs 10, 10 vs 9)

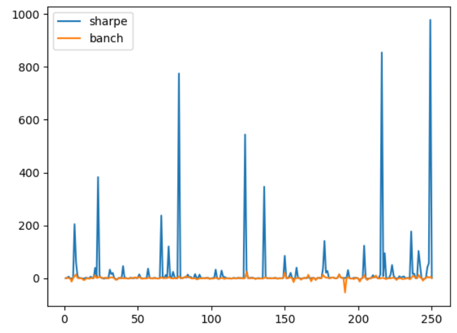
4.1.2 Linear Regression

As for portfolio management, we should not only consider the profit, but also the risk, thus we use the sharpe ratio $R = \frac{E(Rp)-Rf}{\sigma Rp}$ as reward function, where $E(Rp)$ is the expected portfolio return , Rf is the risk free rate, and σRp is the portfolio standard deviation. In our model, we let $Rf = 0$, and $Rf = \frac{pv_t - pv_{t-1}}{pv_{t-1}}$. We use the pairs of two stocks simple linear regression functions($f(x) = a_n x + b_n$, $g(x) = k_n x + h_n$) slope as states (a_n, k_n), and to generalize the states, we only keep one decimal of the slope.

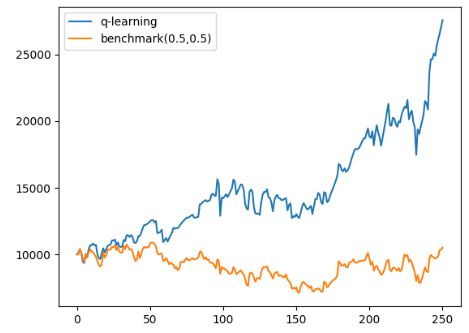
Using 10 pairss stocks one year data to train the model



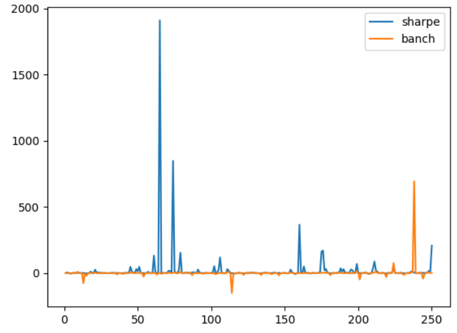
(a) TRAIN 1 (KLAC & SKX)



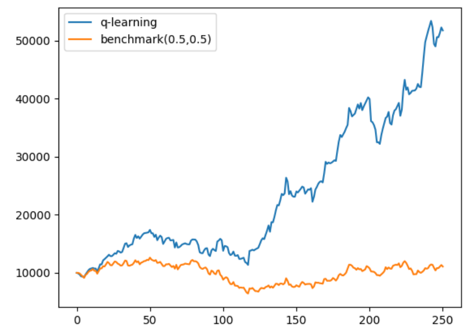
(b) Sharpe Ratio of TRAIN 1



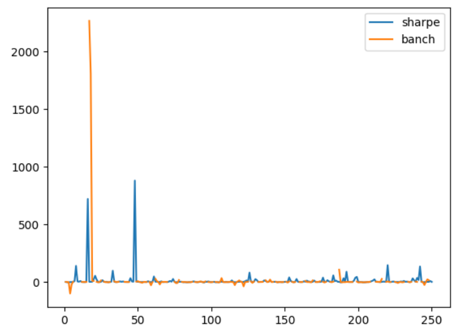
(c) TRAIN 4 (AMD & MTN))



(d) Sharpe Ratio of TRAIN 4

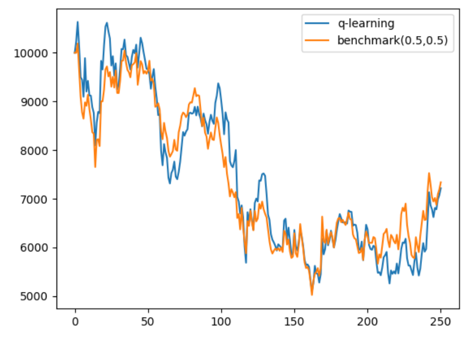


(e) TRAIN 9 (MU & PPC))

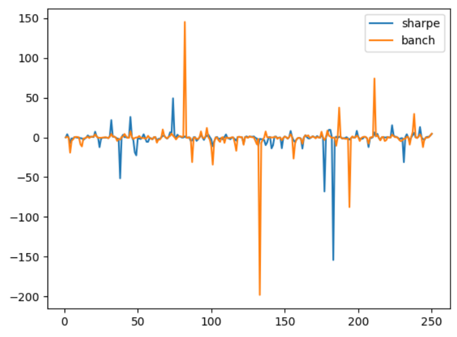


(f) Sharpe Ratio of TRAIN 9

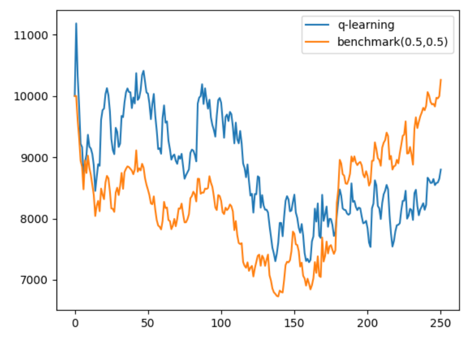
Then we get the test result as follow:



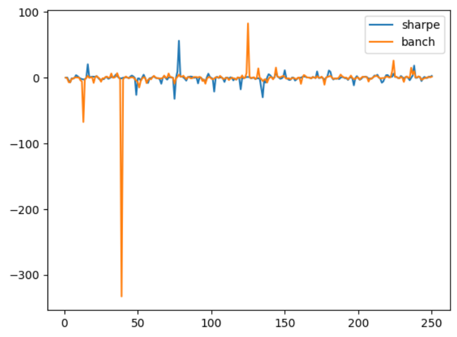
(a) TEST 1 (AMAT & CAJ)



(b) Sharpe Ratio of TEST 1

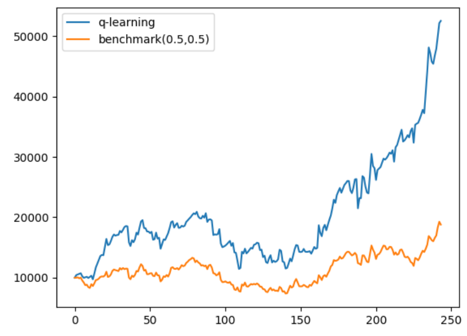


(c) TEST 2 (FCX & CAJ))

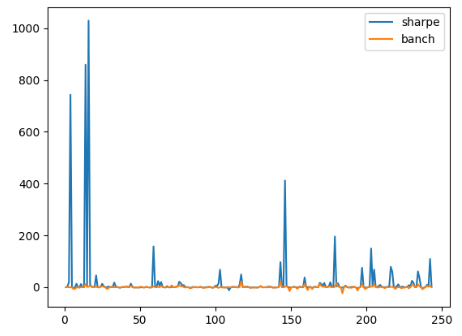


(d) Sharpe Ratio of TEST 2

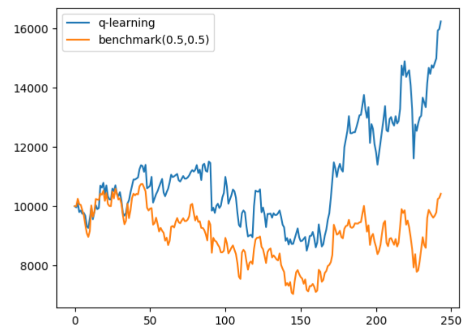
We want to investigate whether more days of a period can have a better result, so we use each 10 days as a period to get the regression functions.



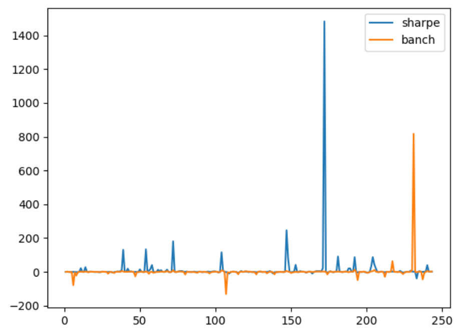
(e) TRAIN 1 (KLAC & SKX)



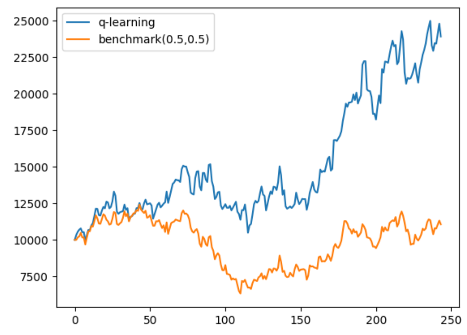
(f) Sharpe Ratio of TRAIN 1



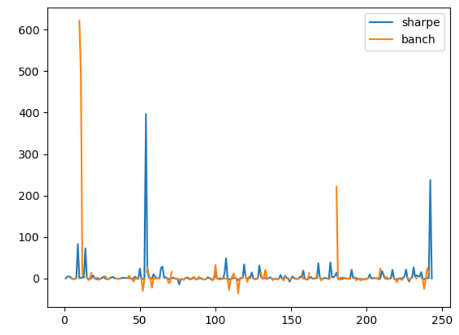
(a) TRAIN 4 (AMD & MTN))



(b) Sharpe Ratio of TRAIN 4

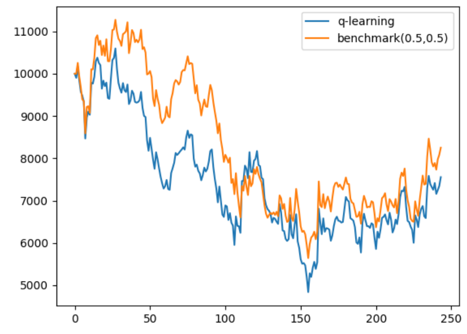


(c) TRAIN 9 (MU & PPC))

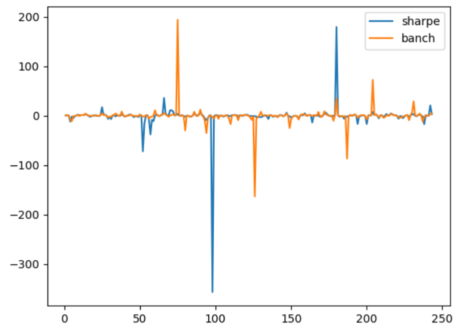


(d) Sharpe Ratio of TRAIN 9

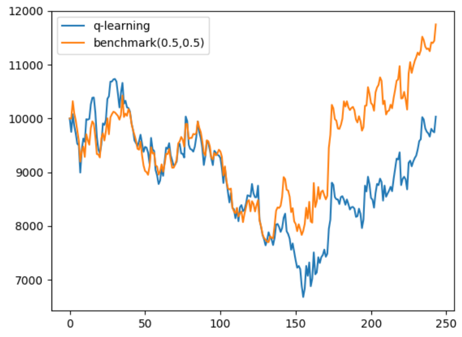
Then we get the test result as follow:



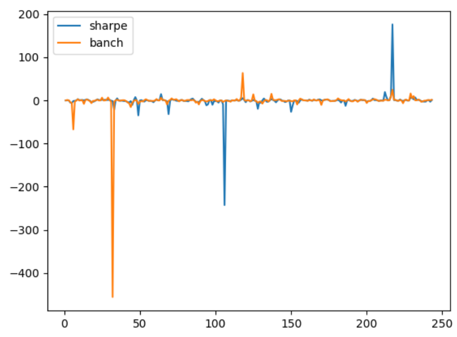
(e) TEST 1 (AMAT & CAJ)



(f) Sharpe Ratio of TEST 1



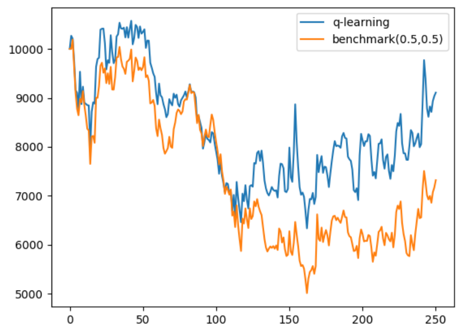
(a) TEST 2 (FCX & CAJ)



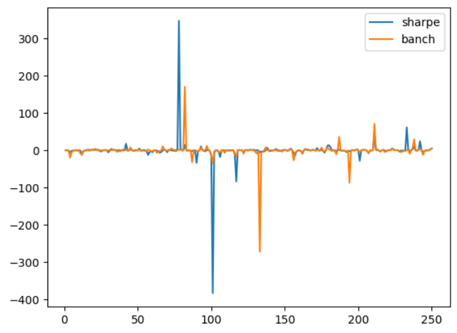
(b) Sharpe Ratio of TEST 2

It is even worse than we've got from 3 days setting as a period.

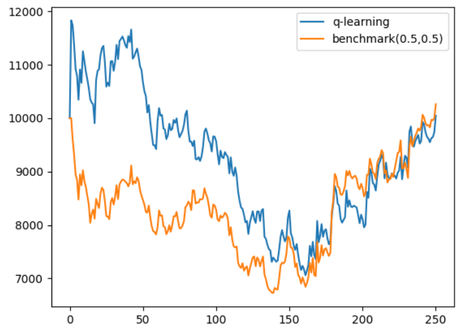
However, if we use 3 days regression to test in the 10-day training q-table, we get the following testing result:



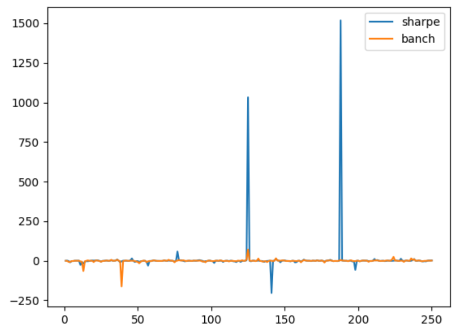
(c) TEST 1 (AMAT & CAJ)



(d) Sharpe Ratio of TEST 1



(e) TEST 1 (AMAT & CAJ)

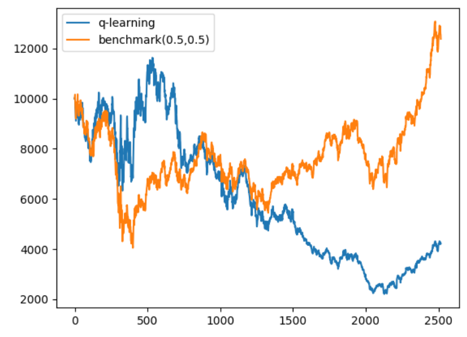


(f) Sharpe Ratio of TEST 1

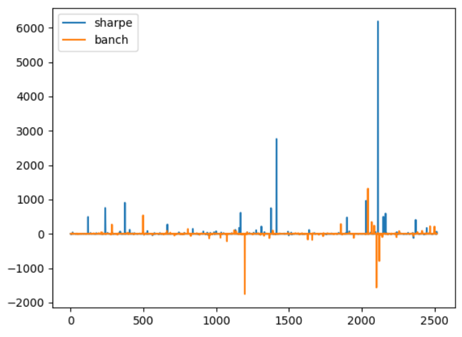
It is better than both of using 10 days as a period to train and test and using 3 days as a period. Maybe training the model by using longer days as a period and test it by using shorter days as a period can get a better model.

Using one pair of stocks 10 years data to train the model (n=3) using sharpe ratio as reward function

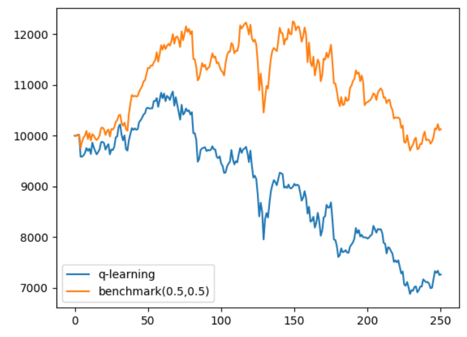
We change the data we use. This time, we use the same one pair of stocks(AMAT & CAJ) to train and test. We use 10 years data to train the model, and use the following one year of data to test it. And still use sharpe ratio as reward function.



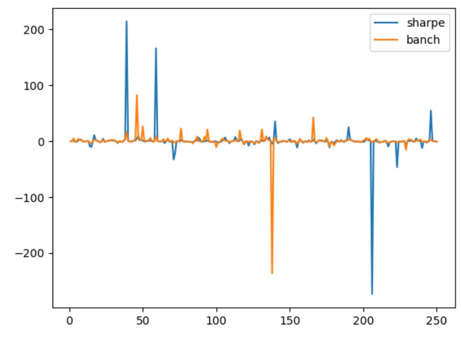
(a) Training



(b) Sharpe Ratio

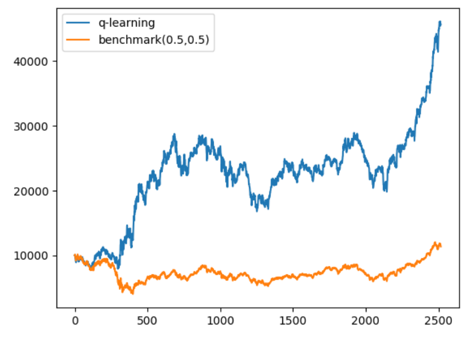


(c) Testing

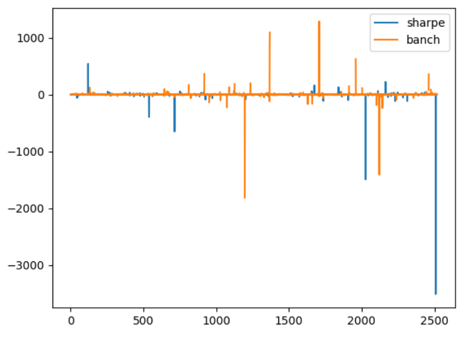


(d) Sharpe Ratio

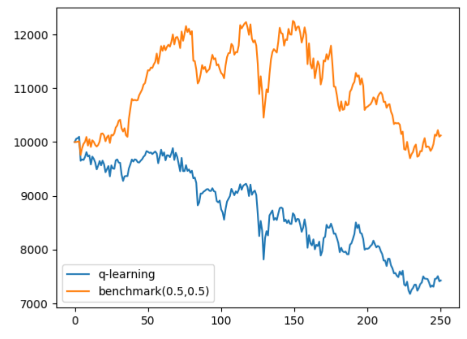
Then we change to use total portfolio value as reward function



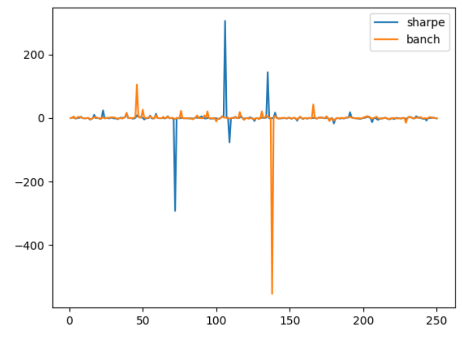
(a) Training



(b) Sharpe Ratio



(c) Testing



(d) Sharpe Ratio

4.2 Deep Q-Network

Q-learning has some obvious problems in our problem setting. Firstly, to implement Q-learning we have to discretize the states, the stock prices. However, if we discretize the stock prices, the result will be inaccurate, which implies that the profit we get may not be optimized. Thus, it is necessary to find an alternative way that can handle the continuous state situation.

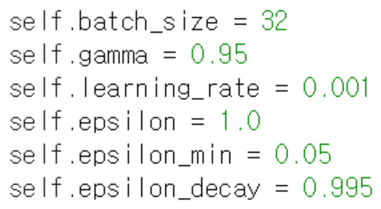
Secondly, it is impossible to include every state and action pair in the Q-table because there are tons of pairs of state and action in the project. Agents should be able to find the action that fits our goal the most even when it runs into entirely new situation. But if a Q-value is not yet available , agents will choose the action randomly. Thus, it is necessary to find another way that helps the agent find the right action without knowing every single Q-value directly.

Deep Q Network is an appropriate method to solve these two big problems.

In our Deep Q Network algorithm, an agent chooses an action randomly with a prescribed probability, which is called an epsilon exploration. It is mainly used in stochastic processed like the situation when stock prices change randomly. Without this epsilon exploration, an agent chooses the action depending only on the specific state that the agent has been trained to react even though better options are highly probable. When neural networks are trained, we use a mini-batch which picks items randomly in the memory that consists of plenty of pairs of state, action, reward, and next-state. In Q-learning, the Q-table is updated and is affected by time correlation. Unlike Q-learning, the algorithm called 'Experience Replay' in Deep Q Network is used to avoid correlation between the memories, especially when the correlation is time dependent

In this project, fully connected layers, whose inputs are vectors, are used instead of convolutional neural networks, whose inputs are matrixes, because states are price changes in this project, and here states can be expressed as vectors more easily than matrixes.

Relu and linear functions are used as activation functions, the Mean Squared Error function as a loss function, and the Adam function as an optimizer. In terms of hyperparameters, we chose figures similar to those that are mainly used.



```
self.batch_size = 32
self.gamma = 0.95
self.learning_rate = 0.001
self.epsilon = 1.0
self.epsilon_min = 0.05
self.epsilon_decay = 0.995
```

Figure 4.16: Hyperparameters

In this Deep Q network algorithm, only two stocks, Applied Materials Inc. (AMAT) and Canon Inc. (CAJ), are used through all procedures.

We train and test separately. 10-years data from 08/03/2007 to 08/02/2017 is used for training and 1-year data from 08/03/2017 to 08/02/2018 is used for testing.

A state is a vector that has 58 elements, each of which is a daily price increment, because we use 30-days stock prices for both stocks as a state.

The actions are categorized into 11 options, which are the ratios between two stocks in the portfolio from 0 : 1 to 1 : 0, changing by 0.1 each state. Naturally in this way the reward becomes the profit made by an action after a day.

It implies that an agent chooses the ratio between two stocks which maximizes the profit by considering the stock price changes in 30 days. The starting budget is 10,000$ and transaction fee is 0.2% of total transaction amount same as that in Q-Learning, and we train it for 3,000, 5,000, and 50,000 episodes, respectively.

The graph shows the price change of AMAT and CAJ for 10-years of training period which is from 08/03/2007 to 08/02/2017. After 10 years, AMAT price increases to 1.97 times, and CAJ price decreases to 0.63 times.

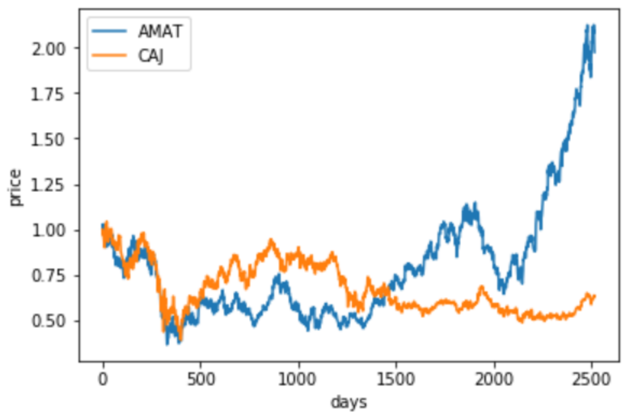


Figure 4.17: Data

Training

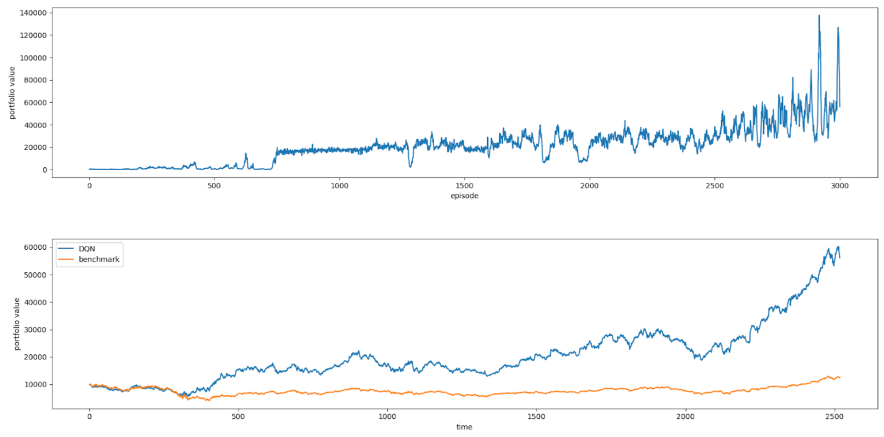


Figure 4.18: Training 1 : With 3,000 episodes

In figure 4.18, the upper graph shows the final portfolio value for every episode. Due to the 0.2% transaction fee, from the beginning to around 700th episode, the portfolio value stays less than 1,000. It suddenly excesses 10,000 at the 747th episode, and keeps increasing and fluctuating continuously until about the 2,500th episode. From the 2,500th episode, it starts to surge dramatically, and at the 2,921th episode the portfolio value peaks at 123,074.

Lower graph of figure 4.18 shows the day by day portfolio value at the final episode, which is the 3,000th one. A benchmark is the strategy that the agent allocate half of total budget to each stock every day. The total portfolio value of the benchmark is 12,433. Even though the final episode that this graph shows is not the episode that makes the profit the most, we can easily notice from this graph that Deep Q Network works much better than the benchmark.

Deep Q Network works well in training, but one weakness is that fluctuation of the portfolio value for every episode is too big as upper the graph shows. We attempted to change the hyperparameters to solve this problem, but it was not effective.

One thing to notice from the upper graph is that from around the 2,500th episode, there is a trend that the portfolio value increases steadily. Thus, we decided to run the same code with 5000 episodes.

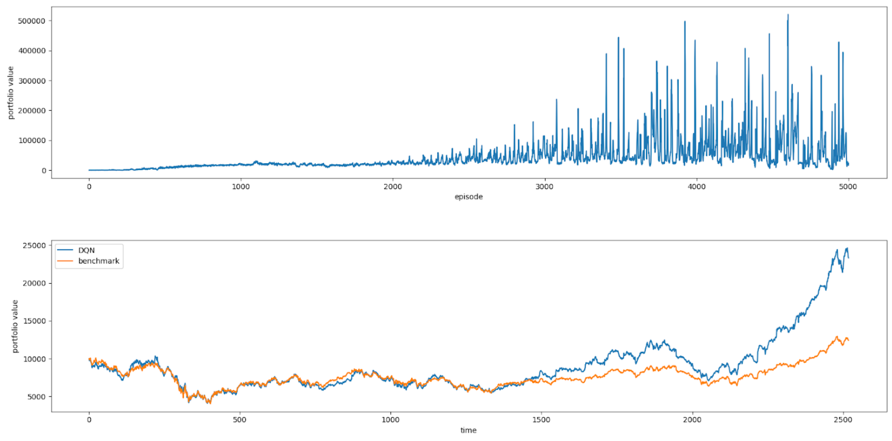


Figure 4.19: Training 2 : With 5,000 episodes

As we guessed, in the upper graph of figure 4.19, after the 3000th episode there are much more peaks than before. The maximum portfolio value is 520,438 at the 4,602th episode.

However, the weakness of Deep Q Network, too much fluctuation, was not solved at all. In the graph below, the result of the 5000th episode is even worse, ending with portfolio value of only 23,321, comparing to the 4,602th episode which ends with 520,438.

Thus, we increased the number of episodes to 50,000 to investigate whether more episodes will solve the fluctuation problem.

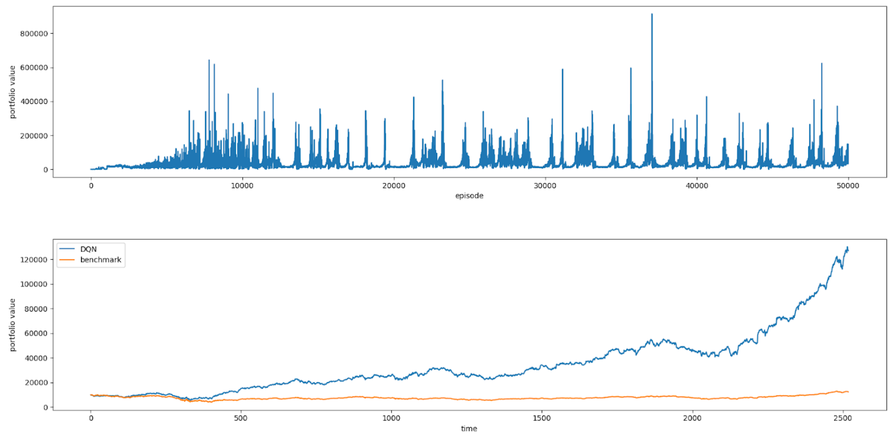


Figure 4.20: Training 3 : With 50,000 episodes

The portfolio value of the last few days still varies too much for every episode, but maximum portfolio value increased again. The maximum portfolio value is 914,586 which happened at the 37,042th episode. As the maximum portfolio value does not increase any more, we decided to stop training.

Overall, training takes about 53 seconds per 100 episodes on average.

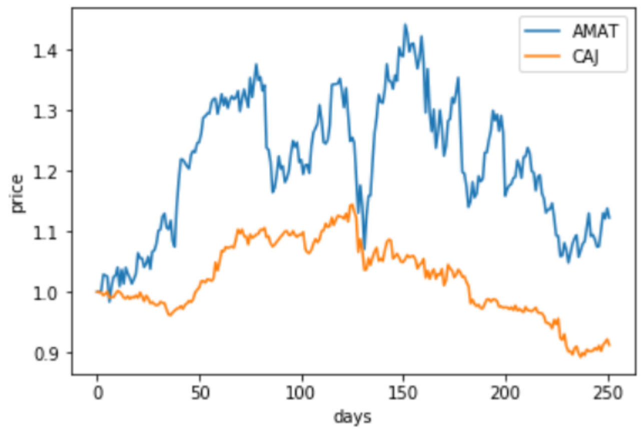


Figure 4.21: Data

For a 1-year testing period, the prices of these two stocks do not change much, and after one year the average is almost the same as the start point.

We chose a model which has maximum portfolio value from each of three results. Thus we choose the 2921th one from the first 3000 episodes to have a portfolio value of 123,074, we chose 4,602th episode's model which has 520,438 portfolio value, and for 50,000 episodes, we chose 37,042th episode's model which has 914,586 portfolio value.

The benchmark remains the same, each stock always has half of the total budget.

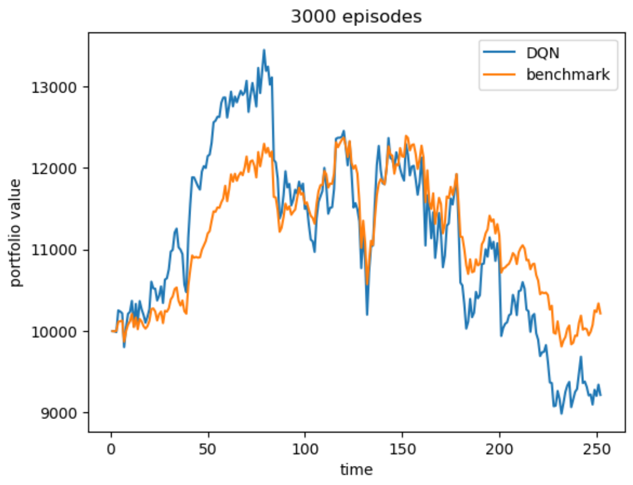


Figure 4.22: Testing (3,000 episodes)

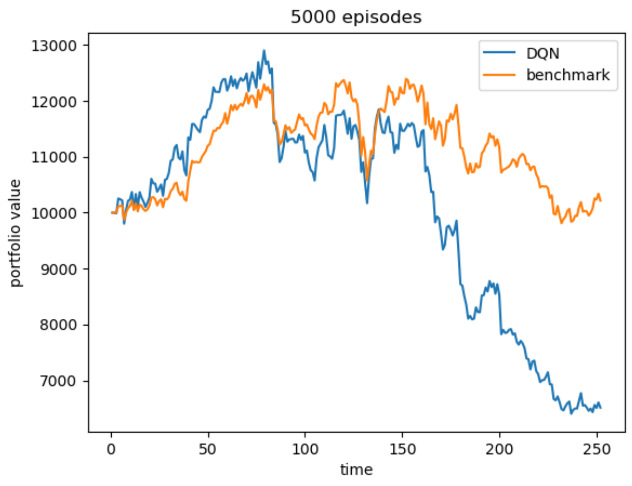


Figure 4.23: Testing (5,000 episodes)

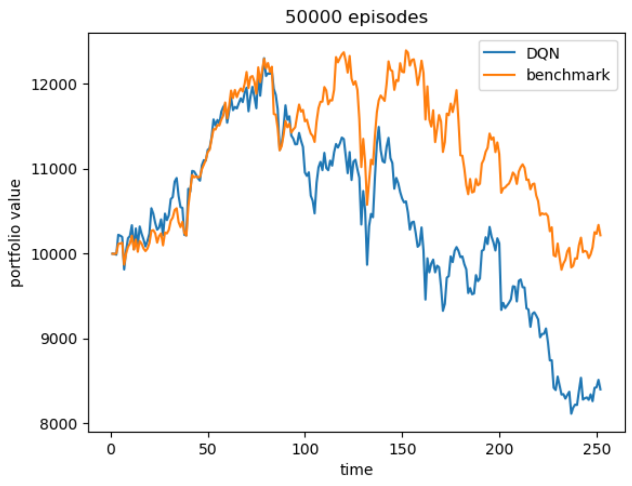
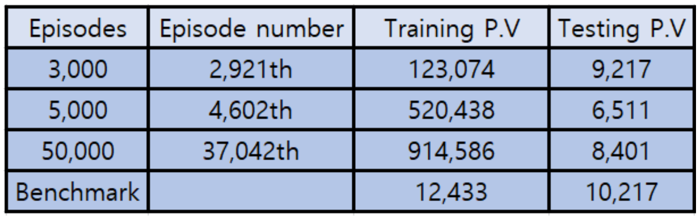


Figure 4.24: Testing (50,000 episodes)

The portfolio value on the last day of a benchmark is 10,217, whereas the portfolio value for testing with the best result among 3,000 episodes is 9,217, a portfolio value for testing with the best result among 5,000 episodes is 6,511, and a portfolio value for testing with the best result among 5,000 episodes is 8,401.

The point of these observation is that no matter how good the result in a training is, it is unrelated to the performance of testing. Moreover, all three results have less portfolio value than the benchmark which just keeps the constant ratio of 50:50.

The chart below summarizes our results for both training and testing by Deep Q-Network.



Episodes	Episode number	Training P.V	Testing P.V
3,000	2,921th	123,074	9,217
5,000	4,602th	520,438	6,511
50,000	37,042th	914,586	8,401
Benchmark		12,433	10,217

Figure 4.25: Comparasion

Chapter 5

Conclusion

For Q-Learning, more training does not always guarantee better performance both for training and testing. In the regression model, training the model by using a longer period but testing with a shorter period gives a better performance.

For Deep Q Network, the reason why the training results are good while testing results are bad is not clear, but one possible interpretation is that the states, which are price changes in the past 30 days, is unrelated to the future price change. If this is true, it explains why training results are good while testing results are bad. From this view, training results are good because when the computer decides the action, it already knows the reward, as it knows the future prices as well. However, when the computer applies the model from training to testing, it does not know the future price and it decides only by the past prices. And if the past prices are unrelated to the future price, the model is useless and this can be the reason why the training result is good whereas the testing result is not good.

Therefore, our conclusion is that it's hard to predict future prices using the history of the past prices. We tried to train the model with various stocks and test with the other stock pairs, and train the model for one stock pair's 10 year data and test with another 1 year. However, both of the test results were not good. We concluded that although how we choose the dataset affects the performance, it is not crucial, as the model itself has some problems.

Chapter 6

Future Works

There are some limitations of our research. First, we just used two stocks to make portfolio. It would be better to build portfolio with various industrial sectors. Second, we assumed that we can buy and sell stocks in non-integer form, which makes our model unrealistic. Third, we defined the states only by using the stock price, and we showed that it is hard to predict the tendency of stock prices by this way. Hence, in a better portfolio model of stocks assets, states should include environmental factors such as market sentiment, average stock prices of each industrial sector. Or building portfolio model using various types of assets such as stocks, bonds, ETF funds considering the inflation, employment rate, market indexes would be better to derive more interesting conclusions.

Appendix A

Exchange Ratio

A.1 Including Exchange Ratio

In the later state of the project, we try to include the exchange ratio into our consideration. The application is that you can include two different countries' stock into the portfolio and settle at the end of the day with one of the stocks' currency in the portfolio. Due to the difference of public holidays and time-zones, we may face the situation that one country's stock market is working while the other one is not. When we encounter such a situation, we will just skip that day in our training model.

The way we calculate the portfolio value is as follows

$$pv_t = pv_{t-1} * [(p_t/p_{t-1}) * w_1 + (q_t/q_{t-1}) * w_2 * cv_t/cv_{t-1})]$$

the portfolio value will be in the currency chosen.

Appendix B

Abbreviations

IPAM. Institute for Pure and Applied Mathematics. An institute of the National Science Foundation, located at UCLA.

RIPS. Research in Industrial Projects for Students. A regular summer program at IPAM, in which teams of undergraduate (or fresh graduate) students participate in sponsored team research projects.

UCLA. The University of California at Los Angeles.

Selected Bibliography Including Cited Works

[1] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, *A brief survey of deep reinforcement learning*, arXiv preprint arXiv:1708.05866, (2017).

[2] X. Du, J. Zhai, and K. Lv, *Algorithm trading using q-learning and recurrent reinforcement learning*, positions, 1 (2009), p. 1.

[3] O. Jin and H. El-Saawy, *Portfolio management using reinforcement learning.*

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, *Playing Atari with Deep Reinforcement Learning*, ArXiv e-prints, (2013).