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, and an action as the weight between two stocks, and a reward as portfolio value change or sharpe ratio. In most cases Q-table or neural networks performed way better than the 50-50 reevaluate benchmark when we trained, but when we use Q-table and neural networks to test on unknown datasets, they are not 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, Proof. LEUNG Shing-Yu and Dr. 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 investor's 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 ratio 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 investor to build up a personalised portfolio to optimize their profit in the financial market.

In this project, we are using reinforcement learning to develop an automated trading strategy. The performance of the algorithm is indicated by comparing to the benchmark. For simplifying our situation we just consider the 2-stock portfolio. We also assume that we can buy and sell the stocks at the open and closed price and our behaviour does not affect the stock market. To solve the problem in Reinforcement learning setting the main methodology we used in the report is Q-learning and Deep Q-network.

Reinforcement learning is a learning strategy that can let agent to learn without knowing the rule from the environment beforehand but only the reward of actions it make. 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 apply the techniques of deep learning to bring reniforcement learning to slove a wider range of problem.[1] The most famous one should be a modified version of well known AlphaGo[1], AlphaGo Zero. It involves the technique deep reinforcement learning. Another example is that using deep reinforcement learning, we can train the computer to play atari game[2].[4] With wide application of reinforcement learning, someone proposed to apply reinforcement learning in financial sector especially for portfolio management. Some of the previous works attempted

[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

to tackle this problem. The work of Jin & EI-SAAWY [3] is similar to our goal in this project. They approach the problem by using deep q network but we also try using Q-learning in our project.They consider sharpe ration as one of the factors to consider which is also what we do in Q-learning. Another related work is Jiang, Xu and Liang[2] which they use 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 cryptocurrency market instead of stock market which what we are trying to do.

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 state(row) and action(column) possible in the environment. Within each cell of the table, we learn a value for how good it is to take a given action within a given state. We start by initializing the table to be all zeros, and then as we observe the rewards we obtain by taking various actions, we update the table accordingly.

When updating the Q-table, Bellman equation is used. Bellman equation's concept is that the expected long-term reward for a given action in 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 would end up in.

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

In Q-learning, for every training process, we first check whether the state already exists in the q-learning table, if it exists, we choose the action with the largest Q-value. If it doesn't exist, we build a new empty line of this state and random choose action. Then we update the value in the q-learning table by Q-TABLE learning formula .

As we use q-learning model, we need to discretize the states and actions. For actions, we discretize it by choosing actions as the weight of stock A of the total portfolio value and only keep one decimal. So we define the action vector as $(0, 0.1, 0.2, \ldots, 0.9, 1)$. For the states, we tried two models. We first use the pair of stocks' close price, but it is not general enough to be used as states in q-learning

model. Then we generalize the states by using the pair of stocks' close price change, and only keep two decimals. After that, we generalize our states by using another method. We using the slopes of the linear regression functions of the stocks' each fixed period(3 days, 10 days).

For the reward function, at the first we simply used portfolio value change as a reward. However, as for portfolio management, we should not only consider the profit, but the risk. Thus we use sharpe ratio as the model's reward function. Sharpe ratio formula

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. ϵ 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 approximates Q-score, since in Q-Learning both state and action state need to be discrete and calculating and optimizing Q-score is 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 using 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-batch from M and replay to solve the problem. Plus, we separate the target network and copy the network regularly to solve non-stationary targets problem.

Initialise replay memory D to capacity N
Initialise action-value function Q with random weights θ
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 ϵ 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 θ
    Every C steps reset $\hat{Q} = Q$
  End For
End for

Chapter 3

Data Description

The algorithms will be tested on stock market data or cryptocurrency data. For stock data, we will collect stock history of each day's open price and close price by using Python library Pandas.DataReader. The stock history that we will use for training algorithm is from 2006.6.29 to 2018.6.29 which includes the financial crisis to train the model on non-occasional circumstances.

Stocks are chosen among S & P 500, considering the beta index, duration, and whether they contain some meaningful abrupt price change history. Samples we have chosen are from the top 10 stocks with the highest weight in S & P 500's high-beta index fund .

Constituent	Symbol	Sector
Align Technology Inc	BALGN	Health Care
Micron Technology Inc	MU	Semiconductor
Nvidia Corp	NVDA	Semiconductor
Lam Research Corp	LRCX	Semiconductor
Advanced Micro Devices	AMD	Semiconductor
NetFlix Inc	NFLX	Consumer Discretionary
Applied Materials Inc	AMAT	Semiconductor
KLA-Tencor Corporation	KLAC	Semiconductor / Material
Freeport-McMoRan Inc	FCX	Mining and Metal
Incyte Corp	INCY	Health Care / Pharmaceutical

Table 3.1: Stock Data we used

We chose one stock from 10 high-beta stocks shown above, and another from low-beta stocks to make up our portfolio. In later steps, we choose K number of stocks considering the sectors, plus other foreign companies such as Lotte from South Korea to make the impact of exchange rate into consideration.

Chapter 4

Mathematical Description of the Project

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.

4.1 Mathematical Formalism

We define $p_1, p_2, p_3, ..., p_t, ...$ as the close price of an stock sequences of each day released from the exchange center. Then for another stock, its close price sequence is $q_1, q_2, q_3, ..., q_t, ...$ Let price vector to be (p_t, q_t). We use price price change vector $(\frac{p_t - p_{t-1}}{p_{t-1}}, \frac{q_t - q_t - 1}{q_{t-1}})$ to define the state later. pv_t is the portfolio value at time period t, which is calculated based on the market value of two stocks and two stocks' weight. We define a_t as weight of stock1 at the time period t in the portfolio value. It is calculated as the proportion of stock1's market value in the total portfolio value.We define the initial portfolio as $ 10,000 which is called budget from now on.

For n days' close price $p_1, p_2, p_3, ..., p_n$, if a line is $F(x)$, the error $\epsilon = (F(i) - p_i)^2, i = 1, 2, 3, ..., n$. We can find a line $f(x) = a_n x + b_n$ that minimize the sum of error $min \sum_{i=1}^{n} (f(i) - p_i)^2$, 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 $g(x) = k_n x + h_n$

4.2 Transaction Cost

In a 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 share to be float type,

we used formula below to calculate transaction cost.

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

4.3 Benchmark

The model's test data performance was compared against two benchmarks. We used the preceding study by Oliver and Hamza to determine the benchmarks. The first, the do-nothing benchmark, allocates half of its starting budget to each stock half-half and then does nothing. This benchmark acted as a very crude approximation of the market since it represents the raw performance of the two stocks.

The second, the rebalance benchmark, re-evaluates its holdings at the end of every market days, and buys or sells stock to ensure the total portfolio value is split into 50-50 between the two stocks. It is important to note that it maintains a proportion of stock values, not stock shares.

4.4 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 stock 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 random choose two high-beta stock 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 combination 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 beta index. 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.

4.5 Assumption

While building model for portfolio management, we took some assumptions. First, we buy and sell the stock at the close price. 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 sell can be non-integer. Lastly, transaction rate occurred by buying and selling are same, and proportional to the exchanged market value.

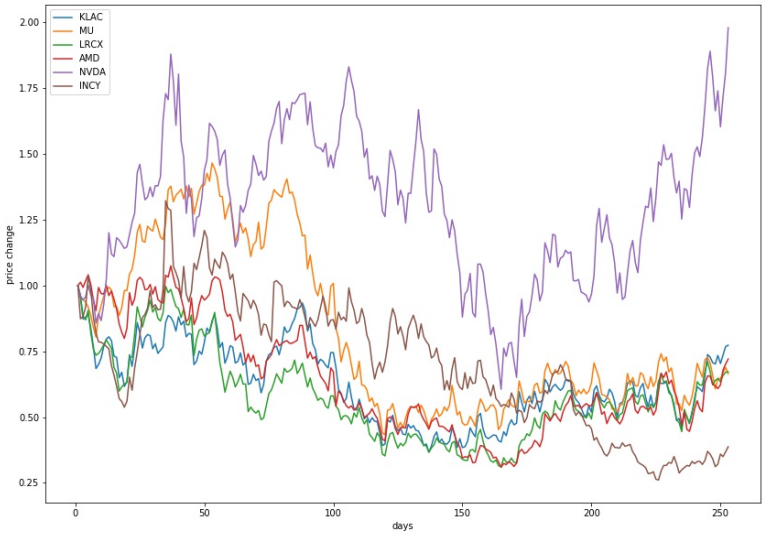


Figure 4.1: High Beta Stock

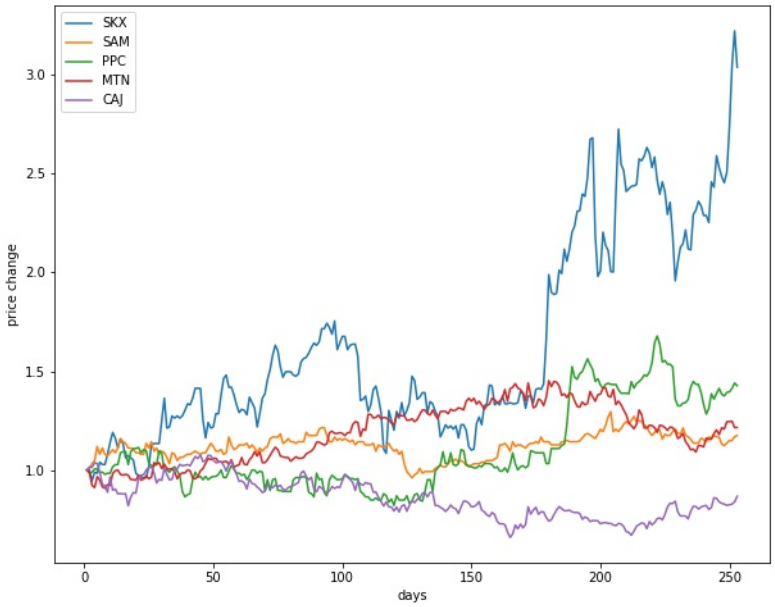


Figure 4.2: Low Beta Stock

Chapter 5

Results and Discussion

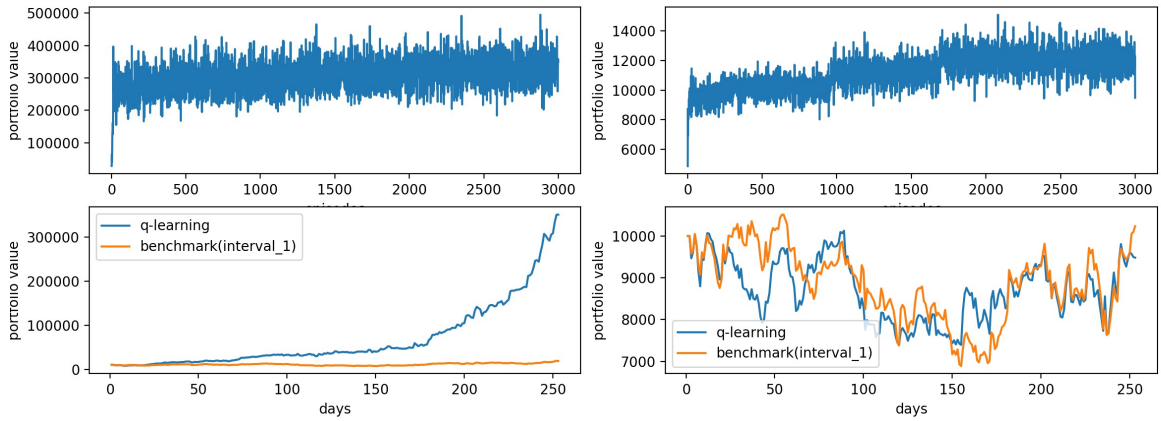
5.1 Q-Learning

5.1.1 Basic Model

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

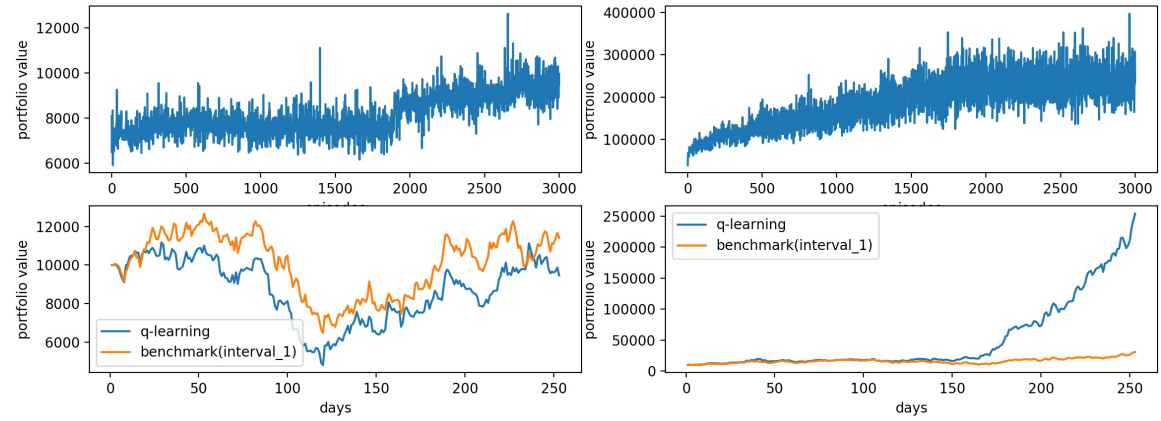
We adjusted the parameters to give better performance. The learning rate(α), which decides the impact of new data on the existing Q-table, was set as 0.0001 based on experiment. The discount factor(γ), which discounts the sum of future reward was set as 0.9. The epsilon(ϵ) was set as 0.9 and for every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

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



(a) Training 1 (KLAC & SKX)

(b) Training 4 (AMD & MTN)



(c) Training 9 (MU & PPC)

(d) Training 10 (NVDA & SKX)

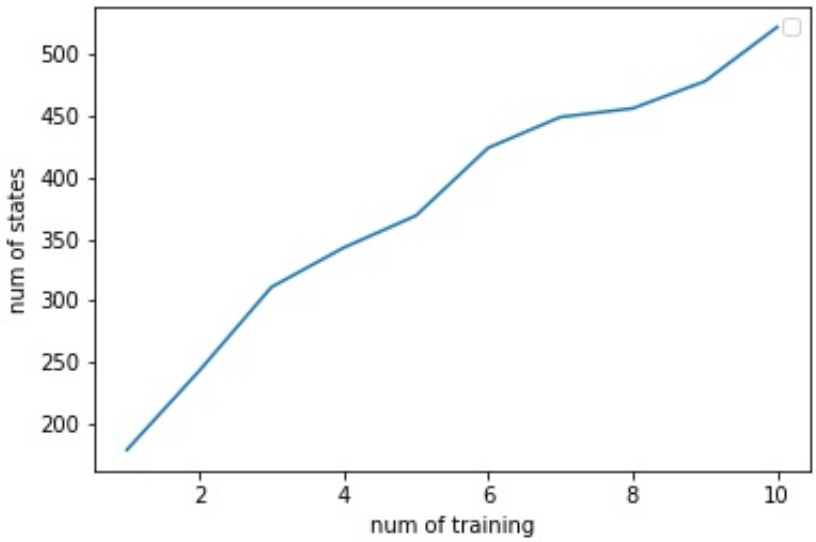


Figure 5.2: Dimension change over training

Using the Q-table we trained, we tested on two test datasets(AMAT & CAJ, FCX & CAJ). Test results using FCX & CAJ dataset usually showed better results. However more training didn't guarantee better test results. For AMAT & CAJ, the best test result was using the Q-table trained with 1 dataset, and further training tables made the test results' portfolio value be decreased. On the other hand, for FCX & CAJ, the best test result was with 8 times training Q-table, and before and after that the final portfolio value is below benchmarks. 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 9 times training, and using 10 times training differed a lot. The last plot which has 3 subplots 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.

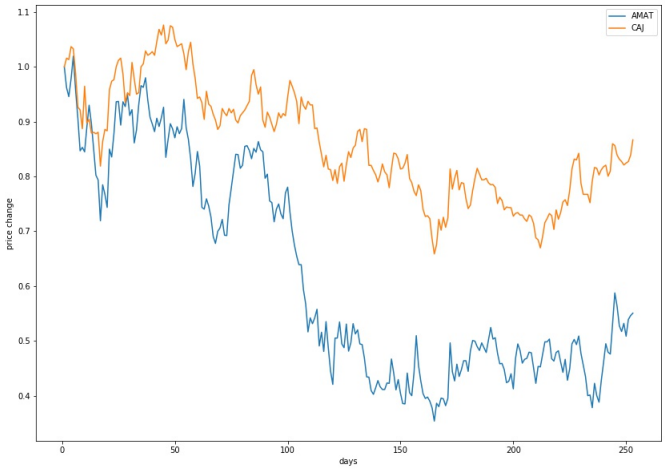
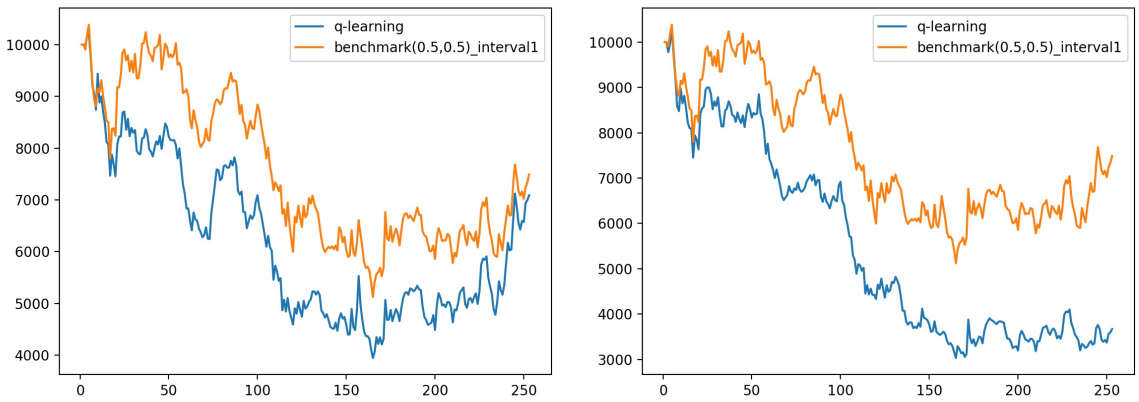


Figure 5.3: TEST SET1 (AMAT & CAJ) :PRICE CHANGE



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

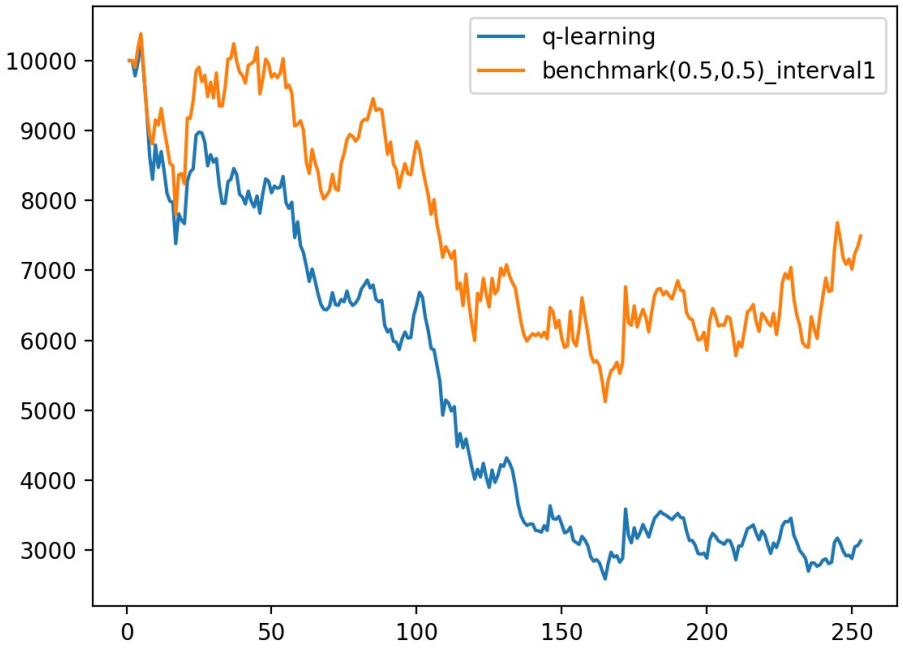


Figure 5.5: TEST SET1 (AMAT & CAJ) : result with 10 training Q-table

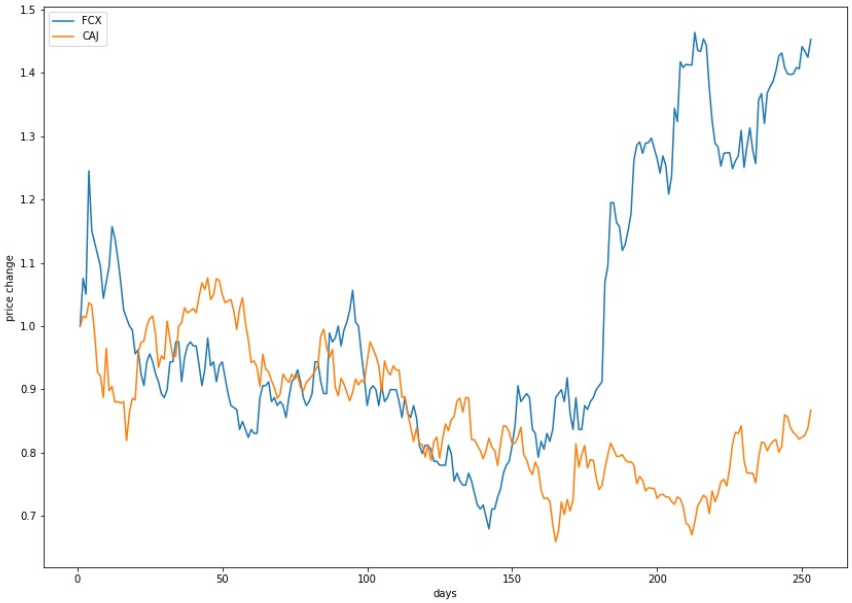
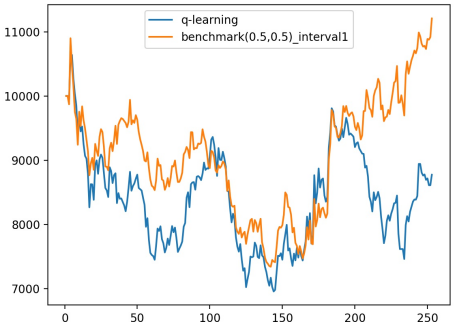
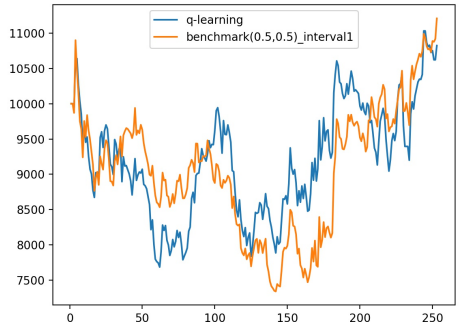


Figure 5.6: TEST SET2 (FCX & CAJ) : PRICE CHANGE



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



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

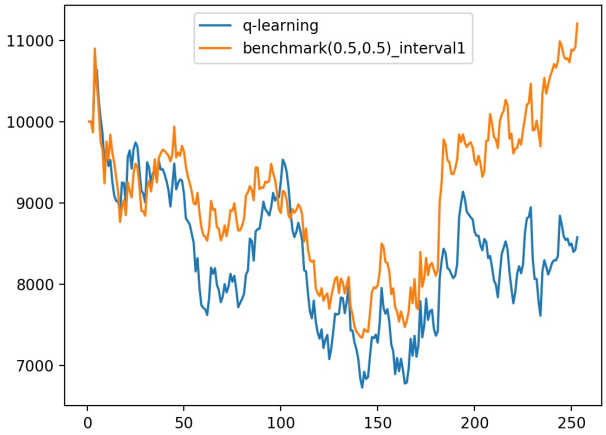


Figure 5.8: TEST SET2 (FCX & CAJ) : result with 10 training Q-table

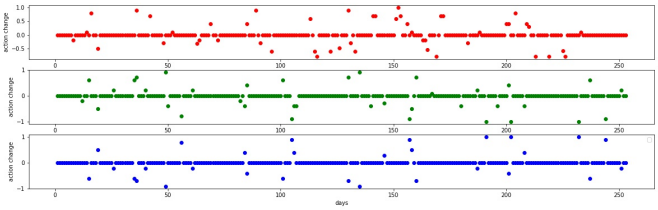


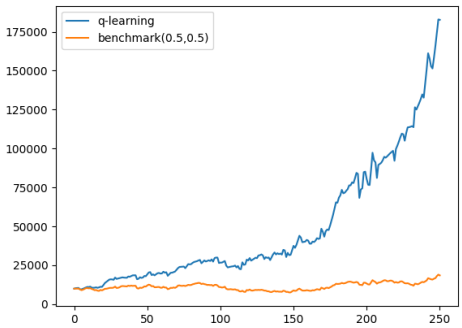
Figure 5.9: ACTION CHANGE : (8 vs 5, 8 vs 10, 10 vs 9)

5.1.2 Linear Regression

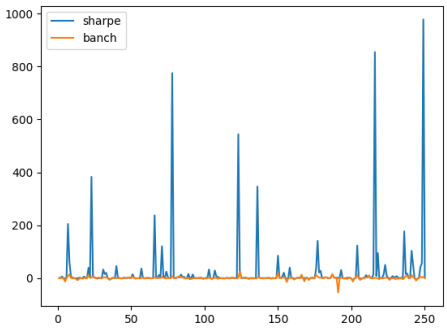
As for portfolio management, we should not only consider the profit, but also the risk, thus we use the sharpe ratio $R = \frac{E(Rp)-Rf}{\sigma Rp}$ as reward function, where $E(Rp)$ is the expected portfolio return , Rf is the risk free rate, and σRp is the portfolio

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' slope as states (a_n, k_n), and to generalize the states, we only keep one decimal of the slope.

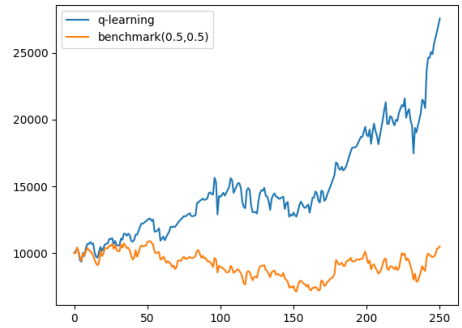
Using 10 pairs's stocks' one year data to train the model



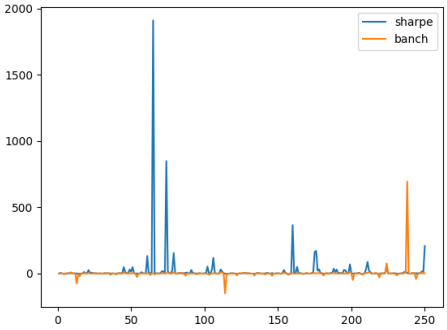
(a) TRAIN 1 (KLAC & SKX)



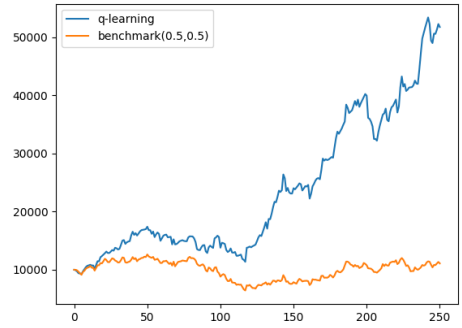
(b) Sharpe Ratio of TRAIN 1



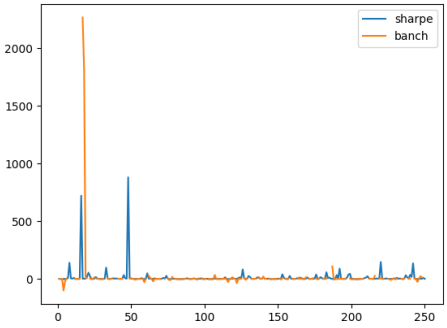
(c) TRAIN 4 (AMD & MTN))



(d) Sharpe Ratio of TRAIN 4



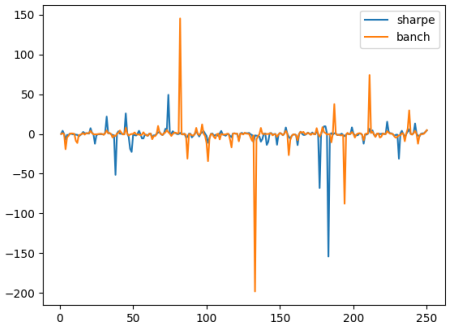
(e) TRAIN 9 (MU & PPC))



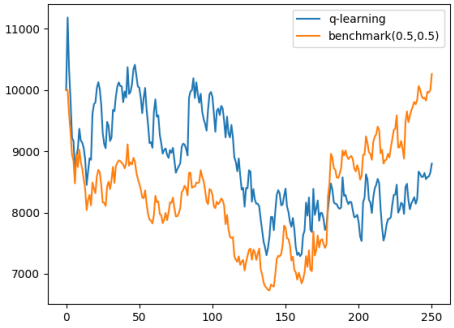
(f) Sharpe Ratio of TRAIN 9

Then we get the test result as follow:

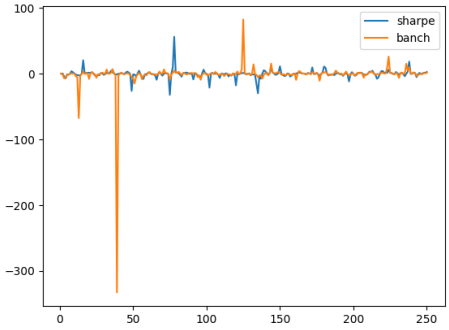
(a) TEST 1 (AMAT & CAJ)



(b) Sharpe Ratio of TEST 1

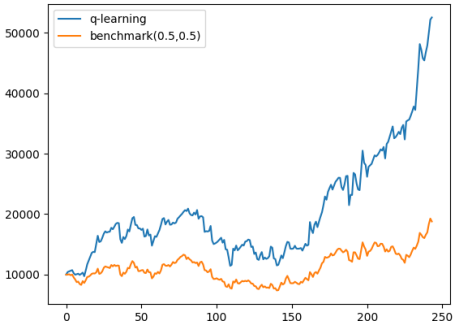


(c) TEST 2 (FCX & CAJ))

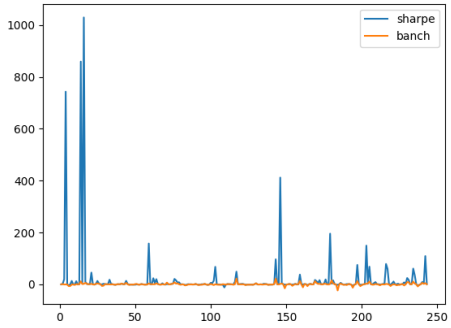


(d) Sharpe Ratio of TEST 2

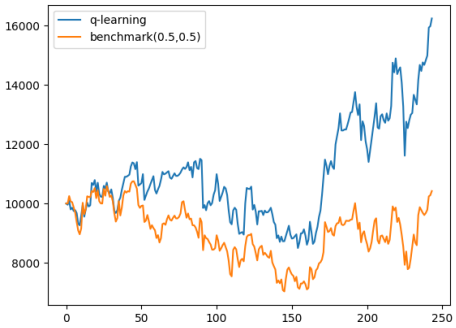
We are thinking that whether more days of a period can have a better result, so we use each 10 days as a period to get the regression functions.



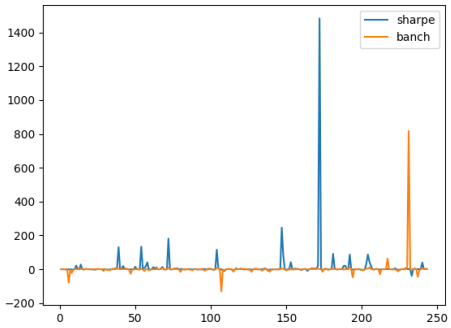
(e) TRAIN 1 (KLAC & SKX)



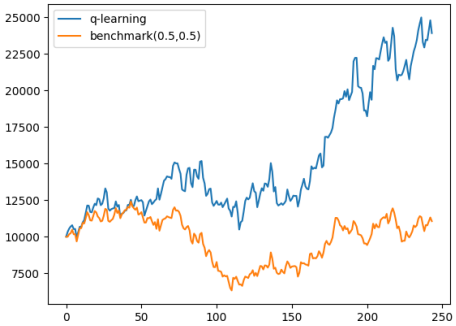
(f) Sharpe Ratio of TRAIN 1



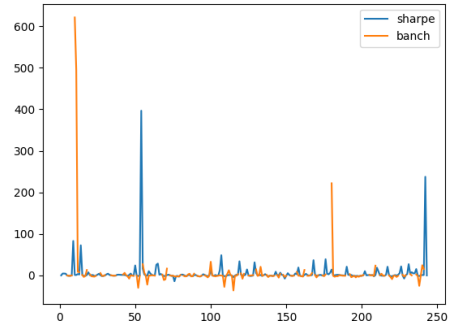
(a) TRAIN 4 (AMD & MTN))



(b) Sharpe Ratio of TRAIN 4

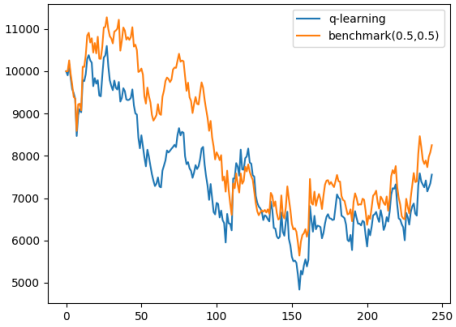


(c) TRAIN 9 (MU & PPC))

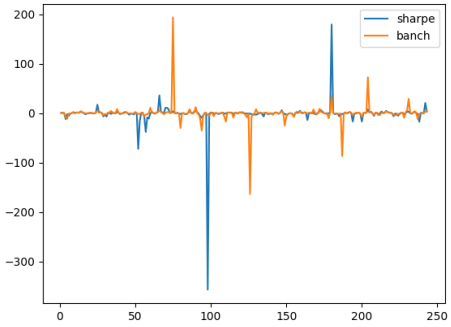


(d) Sharpe Ratio of TRAIN 9

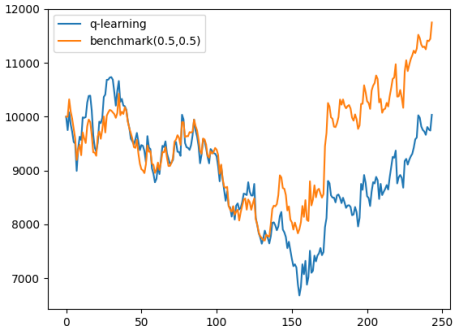
Then we get the test result as follow:



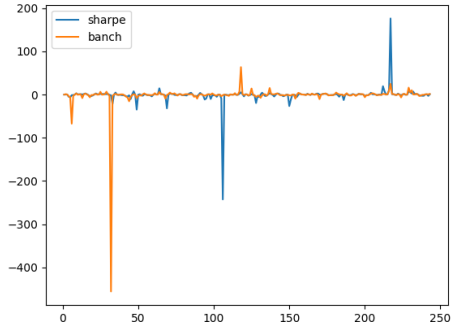
(e) TEST 1 (AMAT & CAJ)



(f) Sharpe Ratio of TEST 1



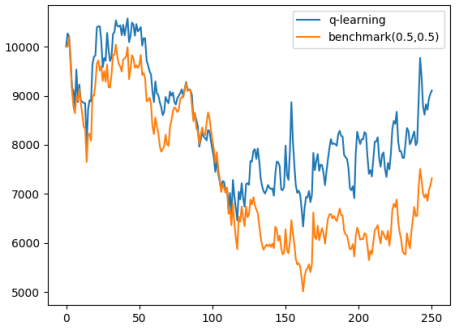
(a) TEST 2 (FCX & CAJ)



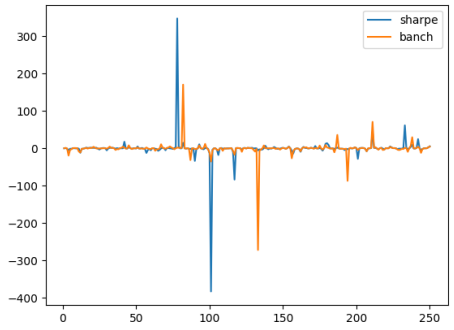
(b) Sharpe Ratio of TEST 2

It is even worse than we use 3 days as a period.

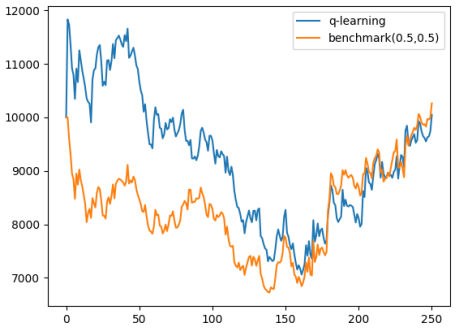
However, if we use 3 days' regression to test in the 10-day training q-table, we get the following testing result:



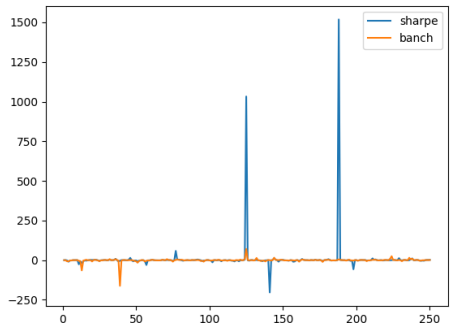
(c) TEST 1 (AMAT & CAJ)



(d) Sharpe Ratio of TEST 1



(e) TEST 1 (AMAT & CAJ)

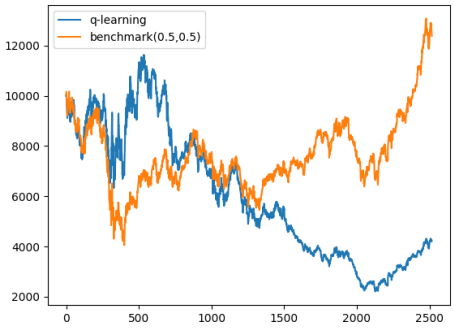


(f) Sharpe Ratio of TEST 1

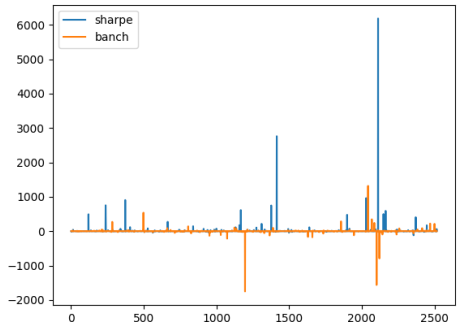
It is better than both of using 10 days as a period to train and test and using 3 days as a period. Maybe training the model by using longer days as a period and test it by using shorter days as a period can get a better model.

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

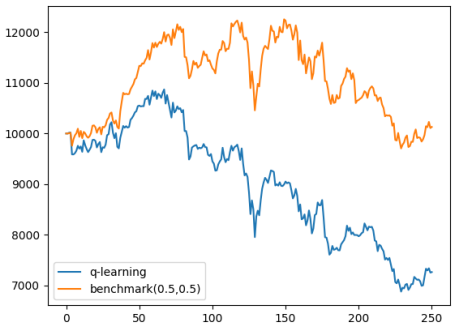
We change the data we use. This time, we use the same one pair of stocks(AMAT & CAJ) to train and test. We use 10 years data to train the model, and use the following one year of data to test it. And still use sharpe ratio as reward function.



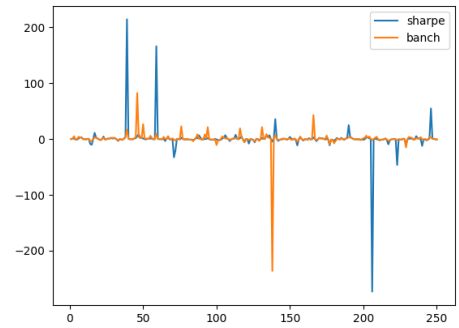
(a) Training



(b) Sharpe Ratio

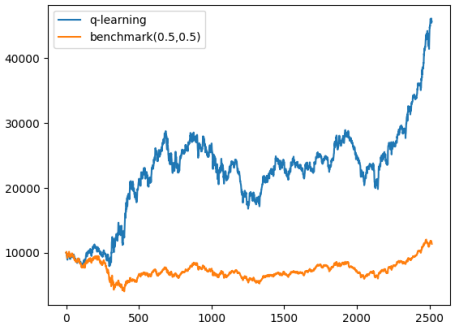


(c) Testing

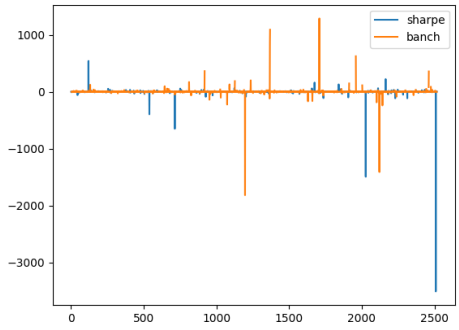


(d) Sharpe Ratio

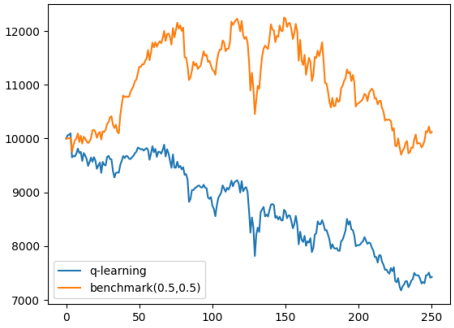
Then we change to use total portfolio value as reward function



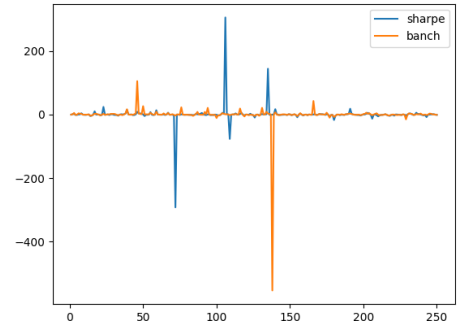
(a) Training



(b) Sharpe Ratio



(c) Testing



(d) Sharpe Ratio

5.2 Deep Q-Network

Q-learning has some obvious problems in our problem setting. Firstly, to implement Q-learning we have to discretize the state, which is the stock price. However, if we discretize the stock price, the result we get 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 find every Q-table value because there are tons of pairs of state and action in the project. Computer should be able to find action that fits our goal the most even when it runs into entirely new situation, but if Q-table values that computer finds are not enough computer will choose the action randomly. Thus, it is necessary to find another way that helps the computer find the right action without having to find out every single Q-table value directly.

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

In our Deep Q Network algorithm, computer chooses action randomly with predetermined probability, which is called 'epsilon exploration'. It is mainly used in stochastic process like this situation where stock price changes randomly, because if without this epsilon exploration after one certain state happens computer choose the

action depending on this specific state solely even though there are high probabilities the other states can happen as well. When neural networks are trained, we use mini-batch which picks items randomly in the memories which consists of plenty of pairs of state, action, reward, and next-state. In Q-learning, Q-table is updated with time and it is affected by time correlation. Unlike Q-learning, in Deep Q Network this algorithm called 'Experience Replay' is used to avoid correlation between each memory, especially with this kind of situation where time is related with, and it makes results better.

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

Relu and linear function are used as an activation function, Mean Squared Error function as a loss function, and 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 5.16: Hyperparameters

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

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

The state is a vector that has 58 elements, each of which is price change between the day before and the day of each stock, because we use 30-days stock price for both stock as the state.

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

It implies that computer chooses the ratio between two stocks which maximizes the profit by considering stock price changes in 30 days. The starting budget is 10,000$ and transaction fee is 0.2% of total transaction amount as same as 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.

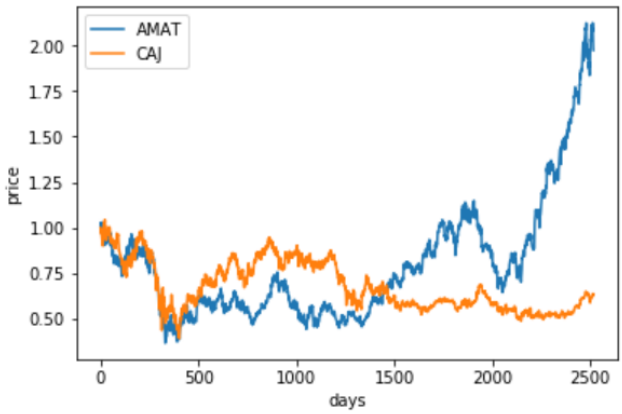


Figure 5.17: Data

Training

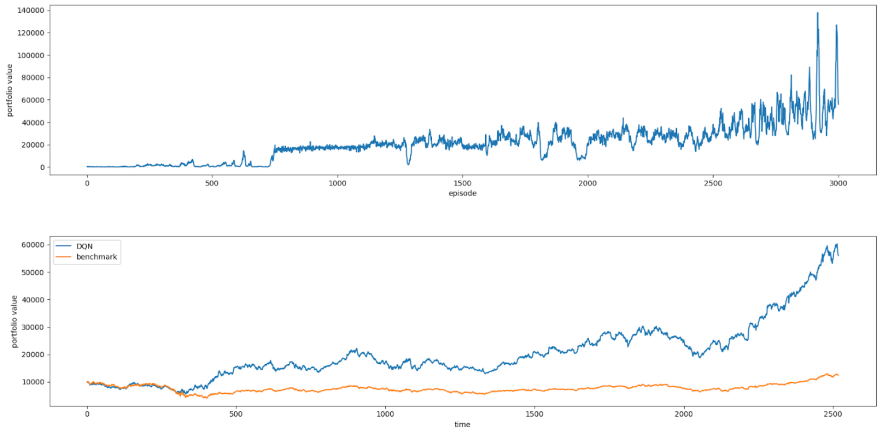


Figure 5.18: Training 1 : With 3,000 episodes

Upper graph shows that the portfolio value of the last day after 10 years of training for every episode. Due to 0.2% transaction fee, from the beginning to around 700th episode, the portfolio value stays less than 1,000. It suddenly passes 10,000 at 747th episode, and keeps increasing and fluctuating continuously until about 2,500th episode. From 2,500th episode, it starts to surge dramatically, and at 2,921th episode the portfolio value peaks at 123,074.

Lower graph shows the portfolio value as day passes at the final episode, which is 3,000th. A benchmark is the strategy that we 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 widely as upper graph shows. We changed figures of hyperparameters to solve this problem, but nothing was effective.

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

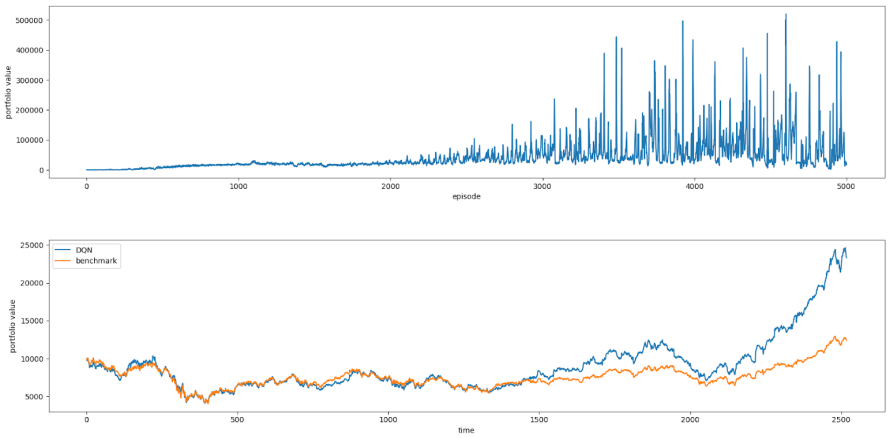


Figure 5.19: Training 2 : With 5,000 episodes

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

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

Thus, hoping that it could be solved if the number of episodes increases a lot, thus we increased the number of episodes to 50,000.

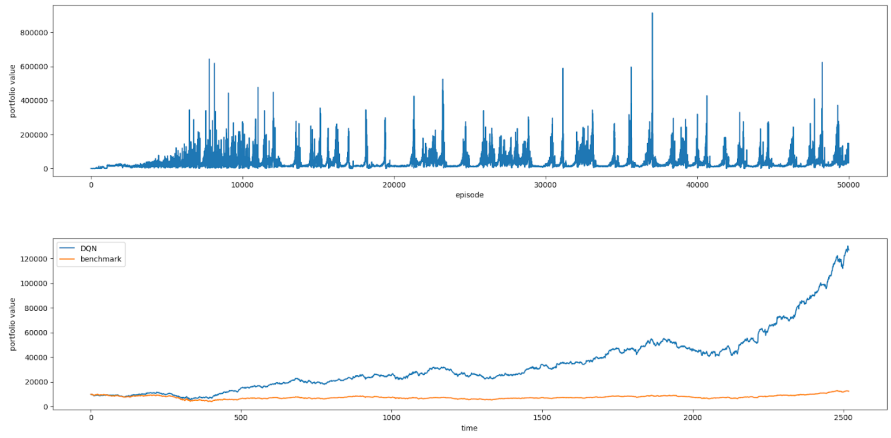


Figure 5.20: Training 3 : With 50,000 episodes

The portfolio value of the last day still varies too much for every episode, but maximum portfolio value increased again. The maximum portfolio value is 914,586 which happened at 37,042th episode. It seems that a trend that maximum portfolio value increases as an episode goes stops, so we decided to stop training.

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

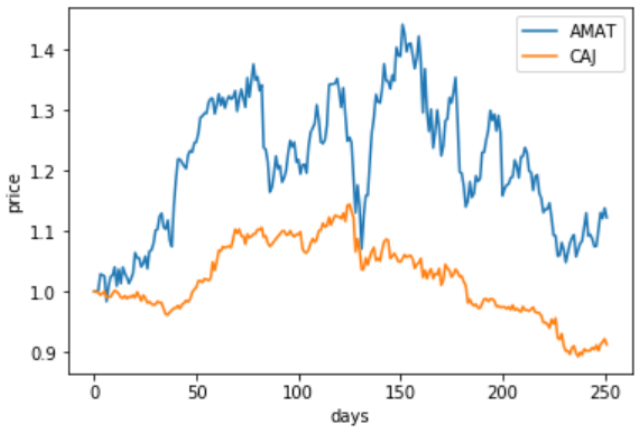


Figure 5.21: Data

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

We picked a model which has maximum portfolio value from each of three results. Thus, for 3,000 episodes, we picked 2,921th episode which has 123,074 portfolio value, for 5,000 episodes, we picked 4,602th episode which has 520,438 portfolio value, and for 50,000 episodes, we picked 37,042th episode which has 914,586 portfolio value.

The benchmark is same with training, each stock has half of total budget for every single day.

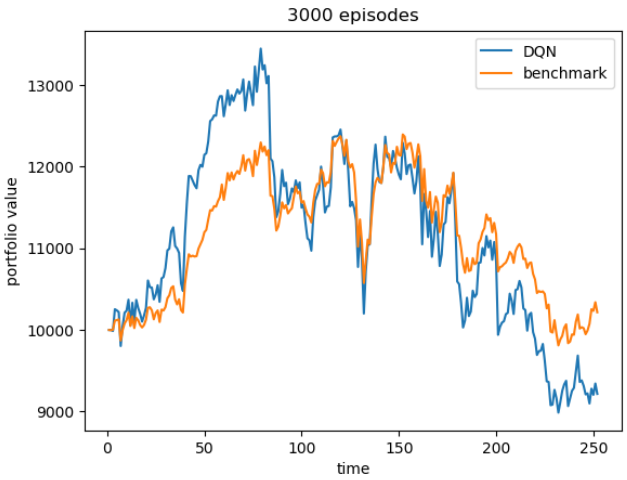


Figure 5.22: Testing (3,000 episodes)

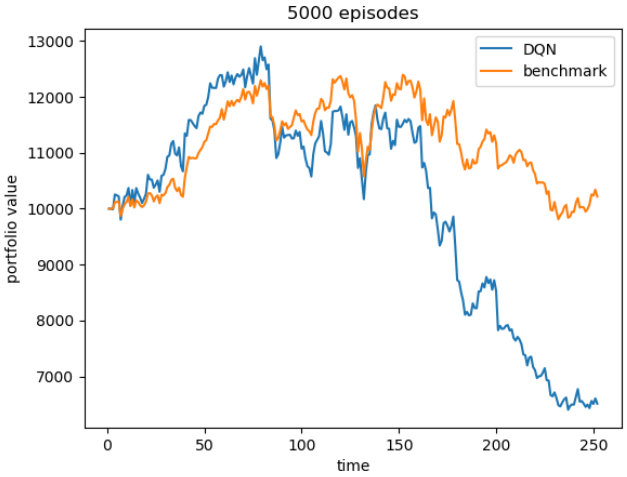


Figure 5.23: Testing (5,000 episodes)

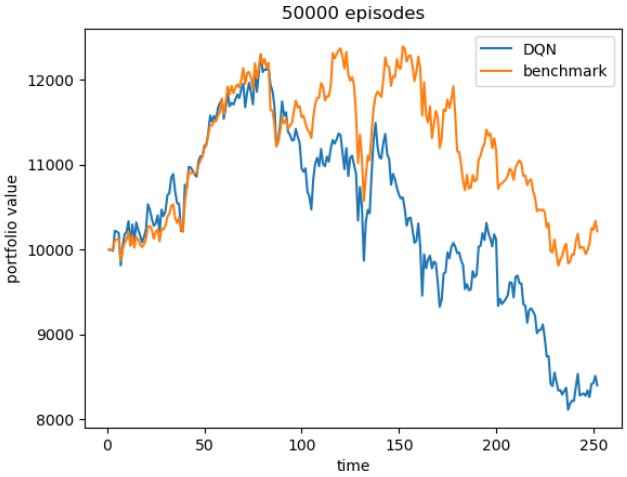


Figure 5.24: Testing (50,000 episodes)

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 how good the result for training is has no relationship with how good the result for testing. Moreover, all three results have less portfolio value than the benchmark which just keeps the ratio with 50:50 for every single day.

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 5.25: Comparasion

Chapter 6

Conclusion

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

For Deep Q Network, the reason why training results are good while testing results are bad is not clear, but one interpretation is that it is because the states, which are price changes for past 30 days, have no relationship with future price change. If this is a true reason, it can explain mostly with why training results are good but testing results are bad. From this view, training results are good because when the computer decides the action, it already knows the reward, which means it knows future price as well. However, when the computer applies the model from training to testing, it does not know the future price and it decides action by the model itself only, whose input is only the past price. And if the past price has no relationship with the future price, the model is useless and this can be the reason why training result is good whereas testing result is not good.

Therefore, our conclusion is that it's hard to predict future price with past price data. 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 test result was not good. We concluded that although how we choose the dataset affects the performance, it is not crucial, rather the model itself is not meaningful.

Chapter 7

Future Works

There are some limitations of our research. Firstly, we just used two stocks to make portfolio. It would be better to build portfolio with various industry sectors. Secondly, we assumed that we can buy and sell stocks in non-integer form, which makes our model unrealistic. Thirdly, we defined the state only using the stock price, and we showed that it is hard to predict the stock price tendency by this way. Hence, for better portfolio model using stocks as assets, state should be defined on environmental factors such as market sentiment, average stock price of each industry 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 conclusion.

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 date with one of the stocks' currency in the portfolio. Since sometime due to the difference of public holiday and time-zone in different countries, we may face the situation that one country's stock market is working while the other one is not. When we face such situation, we will just skip that date 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 you have 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).