MAFS 5370 Assignment 1
Discrete Asset Allocation

Zihua SHE 20921494
Peng GAO 20931528

In this file, we would briefly describe the methods we use and the functions may be used in the program.

All the results are shown in the notebook `MAFS_5370_Assignment_1.ipynb`. Also we have a file to show the mathematical derivation.

1 Math derivation

Please refer to the file `The math derivation for Reinforcement Learning Assignment 1.pdf`.

2 TD Method

$$V\left(S_t\right) \leftarrow V\left(S_t\right)+\alpha\left[R_{t+1}+\gamma V\left(S_{t+1}\right)-V\left(S_t\right)\right]$$

2.1 STEP 1

we first make the following assumptions:

1. Investor has initial investment7 wealth $W_t = 1$.
2. There is only one risky asset.
3. For each step, investor only has two choices:
   (1) stay in risky asset.
   (2) draw 10% from the risky asset for consuming
4. The return of risky asset follows bernoulli distribution
5. The reward is calculated by CRRA utility function

2.2 STEP 2

we constrcut the DiscreteAssetAllocation class with following functions:

1. `returnOnRisky()`: this function gives the return on risky asset
2. `chooseAction()`: this function gives the action we take for the next stage
3. `takeAction(action)`: taking the action, which changes the state to the new state
4. `utility_Func(wealth)`: this function gives the utility of different wealth
5. `giveReward(action)`: given the action we will take, this function gives the reward after the action

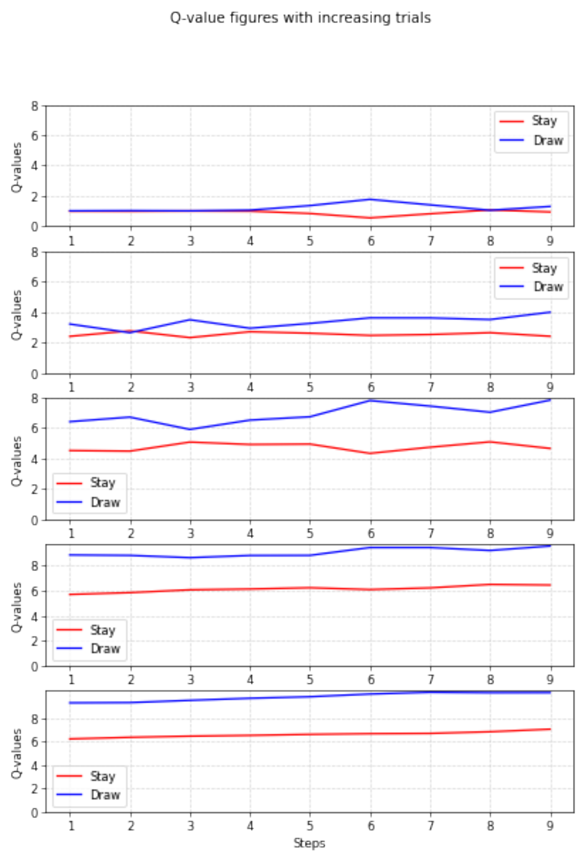
6. `play(rounds)`: simulate the whole process for updating Q-table

2.3 STEP3

we draw the plots with increasing number of trials:

- red line for Q-values of taking action "Stay" at each steps
- blue line for Q-values of taking action "Draw" at each steps

Following is the result of the experiement:



This plot shows that after many iterations, the result converges.It tells us that, under the current assumptions, we should always choose the action "Draw" at each step.

3 Q Learning

$$Q\left(s_t, a_t\right) \leftarrow Q\left(s_t, a_t\right) + \alpha\left[r_{t+1} + \gamma \max_a Q\left(s_{t+1}, a\right) - Q\left(s_t, a_t\right)\right]$$

Since we want to find the optimal strategy, we can also try Q Learning and SARSA to update the Q table. Q learning and SARSA are two of the most used algorithms.

3.1 Method description

We use time t to represent the state. Since we have $T = 10$, hence we have 10 states.

We suppose that the distribution of risky return Y_t is that,

$$P(Y_t = a) = p, \quad P(Y_t = b) = 1 - p$$

where $a = 0.5$, $b = -0.3$, $p = 0.4$.

To simplify the prpblem, we first set that there are 2 choices for the action. One is in risky asset, another one is in riskfree.

3.2 Function description

We firstly define the relative parameters in the code.

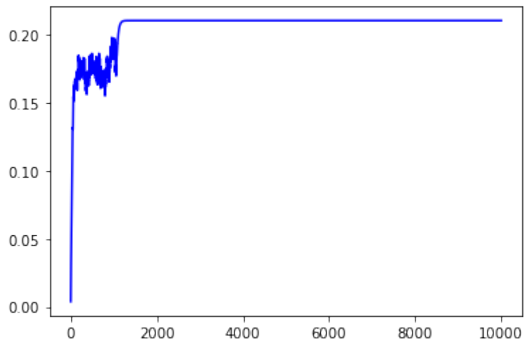
`take_action(current_state_f, action)`: It is used to simulate that we take an action and return the next state and reward. Also we would save the next state wealth in list `W`.

Then we run Q Learning method to update the Q table. Also we run `num_episodes` times to update.

3.3 Results

Consider the final Q table, we would find that the best choice is to stick on the asset whose expected return is higher. In our case, we set $r = 0.2$ so the expected wealth in next state in 1.2. While the expected wealth for risky asset is $1.5 \times 0.4 + 0.7 \times 0.6 = 1.02$. So in the end it would always invest on the riskfree asset. **This result is reasonable.**

We can also plot the last action's Q value in each iteration. The result is that,



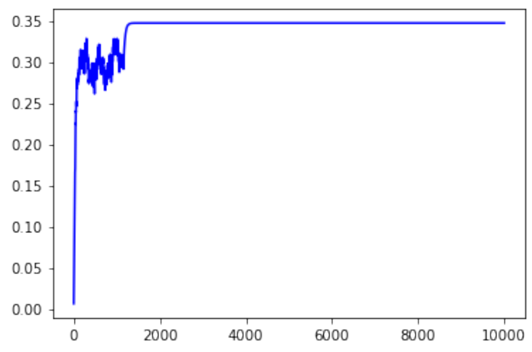
We can find that it can converge.

4 SARSA

$$Q\left(s_t, a_t\right) \leftarrow Q\left(s_t, a_t\right) + \alpha\left[r_{t+1} + \gamma Q\left(s_{t+1}, a_{t+1}\right) - Q\left(s_t, a_t\right)\right]$$

All the setting for SARSA method is the same as the Q Learning method.

We can also plot the last action's Q value in each iteration. The result is that,

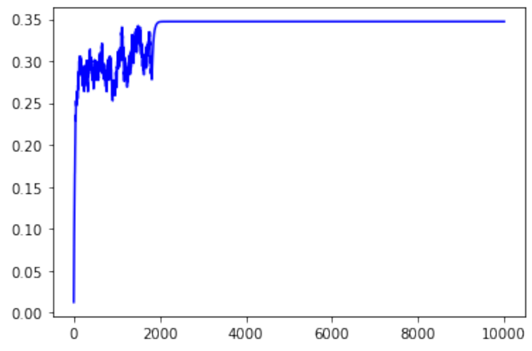


We can find that it can converge. Also we can also find that in the end it would always choose to invest on the riskfree asset (The higher expected return one).

5 Q Learning for 3 actions

Since in the above we consider only 2 actions. We can also change the action space to 3 actions, which contains "All in risky", "All in riskfree" and "half in risky half in riskfree".

The code is similar. We can also plot the last action's Q value in each iteration. The result is that,

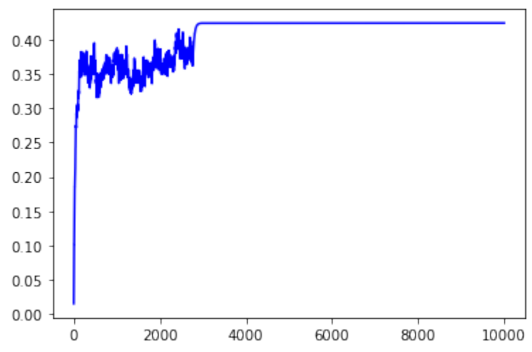


We can find that it can converge. Also we can also find that in the end it would always choose to invest all on the riskfree asset (The higher expected return one).

6 Q Learning for 4 actions

We can also change the action space to 4 actions, which contains "All in risky", "All in riskfree", "half in risky half in riskfree" and "80% in risky 20% in riskfree".

We can also plot the last action's Q value in each iteration. The result is that,



We can find that it can converge. Also we can also find that in the end it would always choose to invest all on the riskfree asset (The higher expected return one).