EE5904/ME5404
Neural Network

Reinforcement Learning

Wang Jiangyi, A0236307J, Mathematics Department
National University of Singapore
E0732857@U.NUS.EDU
April 15, 2022

1 Details of Implementation (Framework)

1.1 Q-learning algorithm

For task 1, we design Q-learning algorithm (pseudo code) as follows:

For each run,
  Initialize Q-table and all parameters.
  For each trial,
    Select next action by ϵ-greedy algorithm and generate next state,
    Update Q-table from current state, action and next state,
    Set current state equal to next state and repeat until reach goal or learning rate sufficently small,
    Record all required information for analysis.
  End
End

1.2 Record and analyze

Here, we introduce (record) several additional pieces of information from our algorithm:

1. In the first run, number of visits to each state.
This is to check whether our choice of exploration probability is able to visit all states. In the convergence result, we need to guarantee that, our agent explore all states frequently in the asymptotic sense. Therefore, we must guarantee the randomness for the choice of action.

2. In the first run, number of steps (reach goal or learning rate sufficiently small) in each trial.
This is to check whether the choice of hyper-parameters is able to reach goal in one run.

3. In the last trial of the first run, the flag of exploration or exploitation for each step.
Based on this, we can analyze why we may not reach our goal for some hyper-parameters setting.

1.3 Find the optimal choice of hyper-parameters

Based on the result of task 1, we fix learning rate and exploration probability as $\alpha_k = \epsilon_k = \frac{100}{100+k}$.

What we still need to determine is the value of discount rate. With larger value of discount rate, Q-learning algorithm will pay more attention to the future reward, which may help our robot reach our goal (state 100). Here, all our candidate of discount rate is $[0.5, 0.6, 0.7, 0.8, 0.9, 0.95]$. Our aim is to select the optimal one based on 2 things:

1. Whether reach goal;
2. The average execution time for each run.

Based on our robot can reach the goal state, we choose the optimal hyper-parameters to minimize the execution time.

2 Task 1

2.1 Performance of Q-learning Table

Firstly, we give our Q-learning Table as follows:

ε_k, α_k	No. of goal-reached runs		Execution time (sec.)	
	$\gamma = 0.5$	$\gamma = 0.9$	$\gamma = 0.5$	$\gamma = 0.9$
$\frac{1}{k}$	0	0	NA	NA
$\frac{100}{100+k}$	10	10	2.56	4.32
$\frac{1+log(k)}{k}$	5	0	18.98	NA
$\frac{1+5log(k)}{k}$	10	10	3.33	9.45

Figure 1: Parameter values and performance of Q-Learning

2.2 Curves of 4 choices of learning rate

Then, before our discussion, we give the curves of different learning rate:

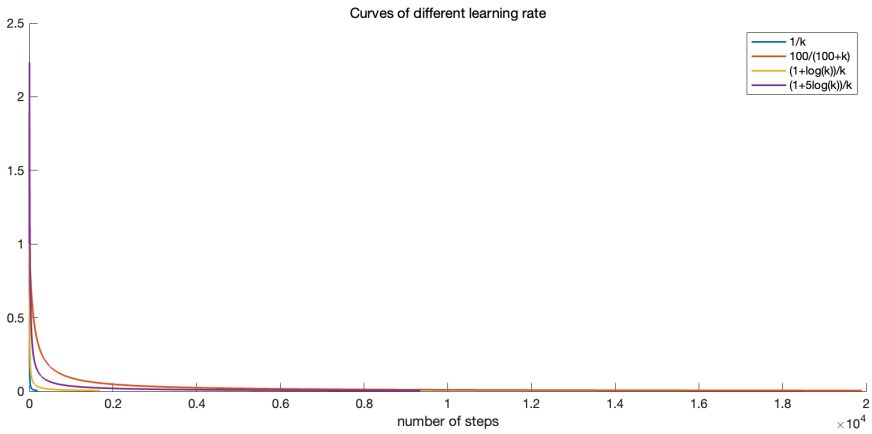

Figure 2: Curves of different learning rate

From Figure 2, we can find that, $\alpha_k = \frac{1}{k}$ decreases the most quickly, $\alpha_k = \frac{1+log(k)}{k}$ is the second quickly one, $\alpha_k = \frac{1+5log(k)}{k}$ is the third quickly one and $\alpha_k = \frac{100}{100+k}$ decreases the most slowly.

Discussion:

The more quickly the learning rate decreases, the less randomness our Q-learning algorithm will attain, which is a terrible thing for Q-learning algorithm.

Supporting argument:

This is because, if the learning rate decreases quickly, since we set exploration probability equal to learning rate, then the exploration probability will go to 0 quickly. Therefore, after very few steps, the exploration probability will be very small, i.e., 0.1. Then, we will almost follow the Q-table to generate actions. This is **something terrible** because, in the convergence theorem of Q-learning algorithm, we are required to visit all states infinitely often. **If the exploration probability shrinks dramatically, we cannot satisfy this condition in numerical experiments**.

This also **corresponds to Figure 1**. When we choose $\alpha_k = \frac{1}{k}$ or $\alpha_k = \frac{1+log(k)}{k}$ (exploration probability shrink quickly), it is difficult for us to reach goal-state (100) in each run. When we choose $\alpha_k = \frac{1+5log(k)}{k}$ or $\alpha_k = \frac{100}{100+k}$ (exploration probability shrink slowly), we will definitely reach goal-state (100) in each run.

To conclude, exploration is extremley important for Q-learning algorithm, especially in the sense of convergence condition.

2.3 Optimal Policy for each combination of hyper-parameters

Here, we visualize our optimal policy for each case:

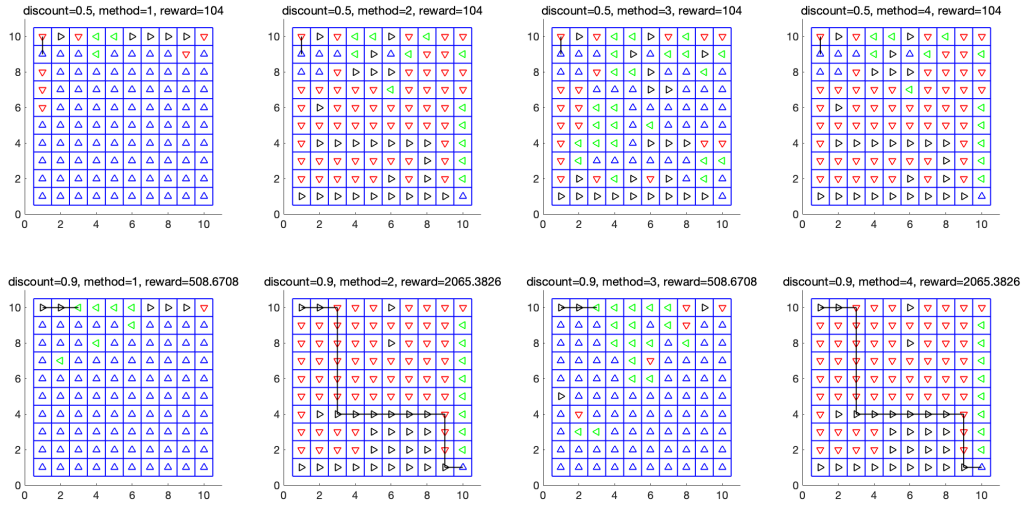

Figure 3: Optimal Policy for each case

1	2	3	4	5	6	7	8	9	10	11
2	3	3	3	3	3	3	3	3	2	2

Figure 4: Optimal Policy when discount = 0.9, method = 4 (Part)

Discussion:

From Figure 3, we can observe that,

1. When discount rate $\gamma = 0.5$, we cannot reach goal state for all 4 choices of exploration probability. For Sub-Figure 1 and 3, the reason is Q-learning algorithm hasn't converged yet. **For Sub-Figure 2 and 4, you can easily see these 2 optimal policies are exactly the same**. Actually, both of these 2 cases, Q-learning algorithms converges! Therefore, when $\gamma = 0.5$, **starting from state 1, the maximal reward is attained by wandering between state 1 and state 2**, which will be asymptotically **104**.

In other word, if our aim is to reach goal-state 100, this choice of discount rate ($\gamma = 0.5$) is unreasonable, which cannot yields an optimal path from state 1 to state 100 by maximizing Total Return.

2. When discount rate $\gamma = 0.9$, in Sub-Figure 1 and 3, our Q-learning algorithm still cannot converge. **In Sub-Figure 2 and 4, similarly, these 2 optimal policies are the same**. Moreover, in the following discussion, we actually find that, **these 2 optimal Q-tables are almost the same**! Starting from this optimal policy, we generate an optimal path from state 1 to state 100 and maximize the Total Return, which is around **2065.4**.

Therefore, compared with $\gamma = 0.5$, $\gamma = 0.9$ is a **more reansonable choice if our goal state is 100**. In this case, by maximizing Total Return, we are able to generate an optimal path from state 1 to state 100. This is what our ultimate goal (start from state 1, end at state 100 and maximize the total return) requires.

Supporting argument:

By checking the reward table, we can find that, **the biggest reward occurs when we are in state 90 and take action a_3 to state 100**, which is **9999**. This gives a reasonable explanation for why large discount rate will tend to give us an optimal policy from state 1 to state 100, but small discount rate not:

With small discount rate, when we arrive at state 90 and take action to state 100, the reward (**9999**) after discounting becomes extremely small. Therefore, we tend to focus on the biggest immediate reward and wander around between some certain states.

With large discount rate, the reward from state 90 to state 100 will be relatively big after discounting. Therefore, this will lead our robot from state 1 to the goal-state 100 since our robot can know there exists some big rewards from state 90 to state 100 by exploration.

In the following discussion, we will give more details about the Q-learning Table and Optimal Policy.

2.4 Detailed discussion

In the following discussion, we will focus on some quantities for the iteration of Q-learning algorithm. The aim is to discover (or explain) why Figure 3 looks like this.

We mainly concentrate on 3 quantites:

1. In the first run, number of visits to each state;

2. In each trial of the first run, number of steps we need to reach goal-state 100 (or exploration probability becomes sufficiently small, i.e., 0.005);

3. In the last trial of first run, whether our robot explores or exploits within each step.

2.4.1 Number of Visits to Each State

Firstly, we discuss about in the first run, number of visits to each state. This part can somehow explain the convergence of algorithm. The visualization result is given in Figure 5:

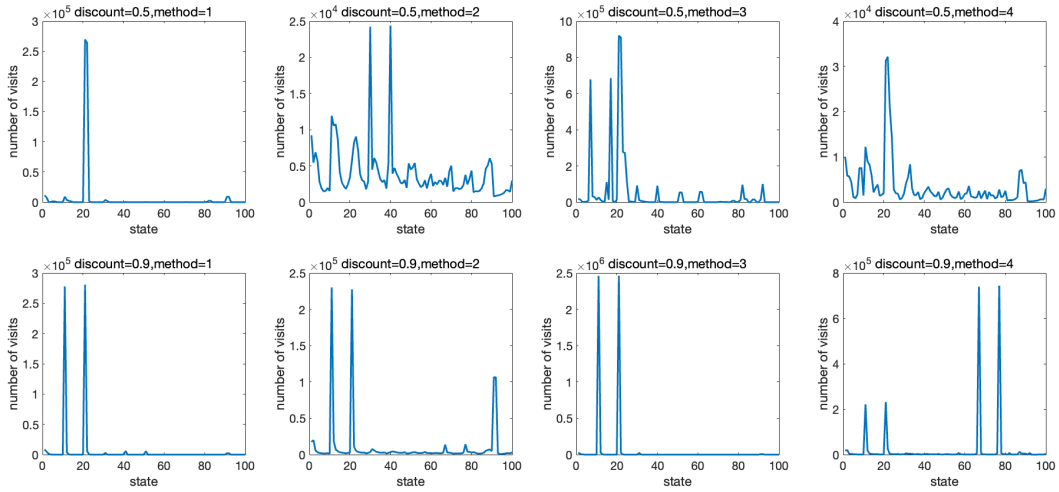

Figure 5: Number of Visits to Each State for Different Hyper-parameters

Discussion and Supporting argument:

In Figure 5, when discount rate $\gamma = 0.9$, for method 1 and 3 (1-st and 3-rd choice of learning rate), our robot will not visit most of the states within one run. This violates the condition that we require our robot to visit all states. **Therefore, for this two case, the Q-table attained by Q-learning algorithm will not converge to optimal Q-table**. For method 2 and 4, it can be observed that our robot will visit almost all the states, although the number of will be small for some states. Therefore, in principle, the Q-table should converge to optimal one for these 2 cases and actually it does.

When discount rate $\gamma = 0.5$, for method 1, 2 and 4, the situation is almost the same as $\gamma = 0.9$. However, for method 3, these 2 cases (visit all states or not) both exist. Sometimes we can visit all states, as the case that shown in Figure 5, while sometimes not. This result corresponds to Figure 1, where the number of goal-reached runs is 5 (of 10 runs) for method 3 and $\gamma = 0.5$.

Moreover, it is worthwhile to notice that, **the convergence of Q-table is not equivalent to the existence of an optimal policy from state 1 to goal-state 100**. That is, the optimal policy corresponds to our hyper-paremeters setting. If we choose small γ, we may attain the largest return by wandering between some states forever instead of reaching state 100.

This part can be one evidence of the convergence for Q-learning algorithm. The randomness of robot (tend to explore) will be of great help for our algorithm to achieve convergence.

2.4.2 Number of Steps to Terminate Each Trial

Then, we discuss for the first run, number of steps we need to terminate each trial. This part can explain the average execution time of each combination of hyper-parameters. The result is shown in Figure 6:

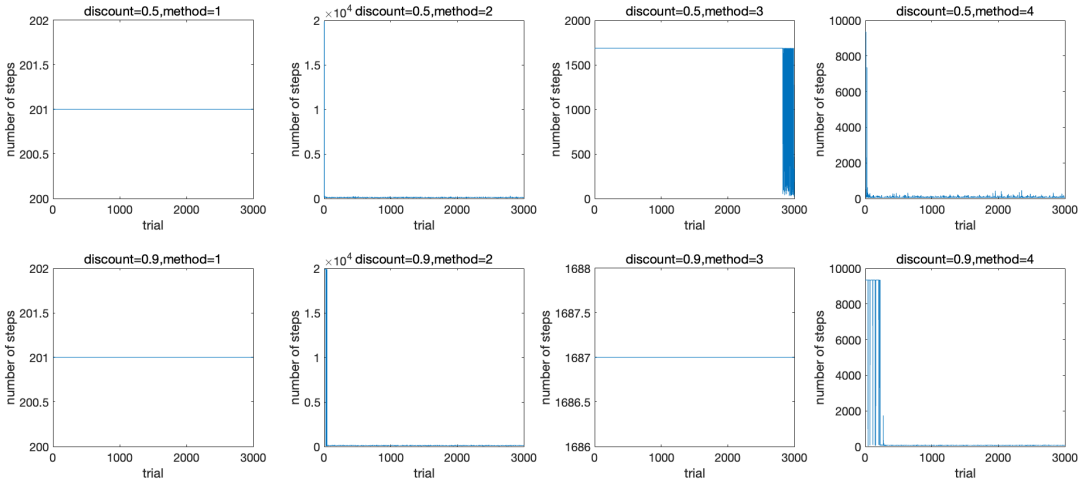

Figure 6: Number of Steps Required to Terminate Each Trial

Discussion and Supporting argument:

In Figure 1, it can be observed that, when $\gamma = 0.5$, execution time for method 1, 2, 3, 4 is NA, 2.56, 18.98, 3.33 seconds respectively. When $\gamma = 0.9$, their execution time is NA, 4.32, NA, 9.45 seconds respectively.

Observation 1:

Firstly, focus on method 2 and 4. When $\gamma = 0.5$, their execution time is almost the same, but when $\gamma = 0.9$, their execution time is quite different.

Supporting argument:

We can explain this observation via Figure 6. Notice that, when $\gamma = 0.5$, the curve looks the same for these 2 methods. The curve will quickly shrink, which means we collect enough information about environment via exploration. Therefore, when $\gamma = 0.5$, both of these 2 methods are very efficient. However, when $\gamma = 0.9$, it is not the case. **It can be observed that, for method 4, it will require around 9000 steps for 400 trials, while for method 2, it only requires 20000 steps for 30 trials**. In this case, method 2 is far more efficient than method 4, and the execution time actually can prove this.

Observation 2:

Secondly, notice that, for method 3 with $\gamma = 0.5$, for almost all trials, it will terminate until the learning rate is sufficiently small, instead of reahing the goal-state 100. Only when number of trials is close to 3000, it just started to reach the goal-state.

Supporting argument:

Intuitively, for cases like this, our robot collects very few information about environment around the goal-state**. The key reason is, the randomness is not enough for our choice of exploration probability (method 3)**. After few steps for each trial, robot will tend to exploit with large probabilty. **This greatly hinders us from reaching some good states (in the sense that, if we arrive at these states, then we can follow greedy policy and reach the goal)**. Therefore, this explain why the number of goal-reached runs is unstable (since we rely on **small exploration probability** to reach some good states), sometimes we can reach but sometime we cannot. Also, that is why we need **18.98** seconds for each run.

2.4.3 Exploration and Exploitation for Each Step

Lastly, we discuss in the last trial of first run, robot chooses to exploit or explore for each step. The result is given in Figure 7:

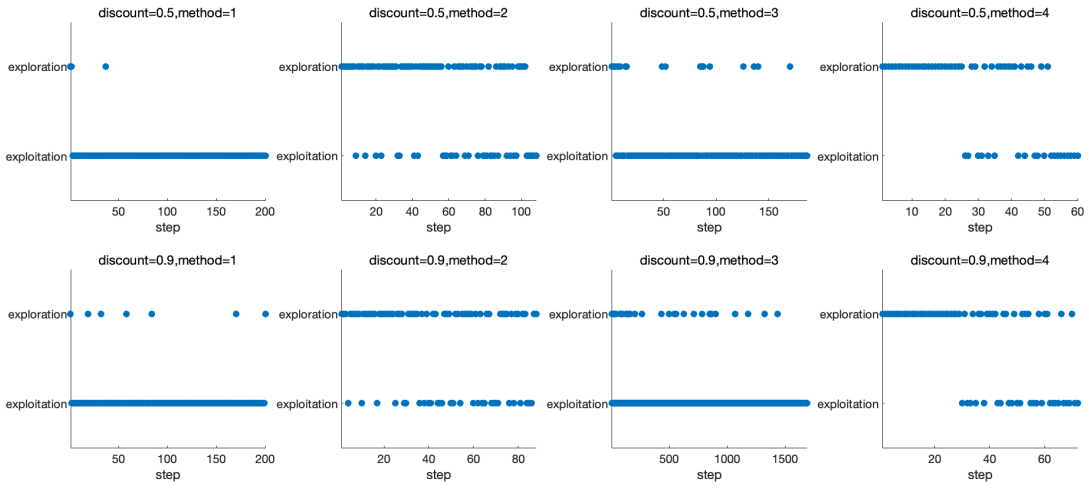

Figure 7: Exploration and Exploitation for Each Step

Discussion and Supporting argument:

Before discussing, recall that in Figure 2, we have Method 2 > Method 4 > Method 3 > Method 1 in the sense of exploration probability.

From Figure 7, for method 2 and 4, its exploration probability is relatively high, which guarantees that it can explore almost all states. **In most time of one trial, it will keep exploring**. For these 2 cases, Q-learning algorithm performs quite well and always converges.

For mehtod 1 and 3, it can be observed that the robot seldom explores. **It will always follows the Q-table to make actions, which is terrible for the updation of Q-table**. By checking the element of the optimal Q-table (attained after 3000 trials) for these 2 cases, we find that most of them are 0. This also shows that the robot hasn't reached many states after 3000 trials. Therefore, for these 2 cases, it is difficult for the robot to reach goal within one run, and Q-learning algorithm will not converge either.

2.4.4 Difference between Q-table for different runs

We also check the difference of Q-table for fixed hyper-parameters. Here is the conclusion:

1. For Method 2 and Method 4, the Q-tables attained in 10 runs are all the same. I mean, these 20 Q-tables are all the same!

2. For Method 1 and Method 3, the Q-tables attained in 10 runs are quite different.

This also shows the convergence of Q-learning algorithm for different hyper-parameters.

2.4.5 Agreement between total reward and Q-table

Another observation about total reward is found, which can be of one evidence for the convergence of the algorithm for Method 2 and Method 4, that is:

When we focus on the Q-table with the starting state and corresponding optimal policy, we find that it is the same as the total reward if the algorithm converges (Method 2 and Method 4).

For example, if we take Method 2 and $\gamma = 0.9$, then we will have: Q-table(1,2) = 2065.3826, which is exactly the total reward. Here, the optimal policy for state 1 is just action 2.

3 Task 2

Based on our Task 1 experiments, we find 2 things:

1. Large choice of discount rate will lead to longer execution time;

2. With small choice of discount rate, we may not reach goal.

Therefore, we want to **achieve a trade-off between attainability and time-cost**. Next, we fix the exploration probability at $\alpha_k = \frac{100}{100+k}$. The candidates of discount rate is [0.5, 0.6, 0.7, 0.8, 0.9, 0.95]. We aim to select the optimal discount rate from them, and the result is given as follows:

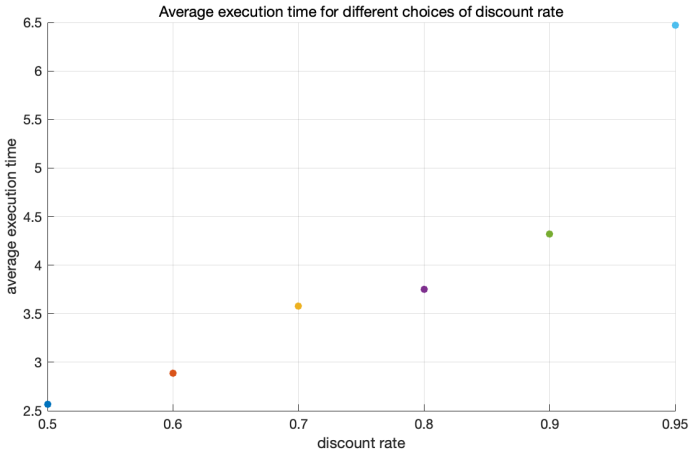

Figure 8: Execution Time for Different Discount Rate

By checking the corresponding optimal policy, we find that for discount rate $\gamma = 0.5, 0.6$, the robot cannot achieve the goal-state 100. For other choices of γ, we can always reach the goal-state.

Therefore, **we finally choose $\gamma = 0.8$, $\alpha_k = \frac{100}{100+k}$ for task 2 implementation**. We do not choose $\gamma = 0.7$ because the decrease of execution time is very small, and larger γ will focus more on the future reward.

4 Appendix

All matlab codes (*.m) will be attached in .zip.