Delft University of Technology

Artificial Intelligence Techniques

IN4010-12

Assignment: Reinforcement Learning

Authors:
Yiting Li(5281873)
Jiaxuan Zhang(5258162)
Yi Yao(5250161)
Yu shen(5249260)

1 Q Learning

1.1 Code Exercise 1

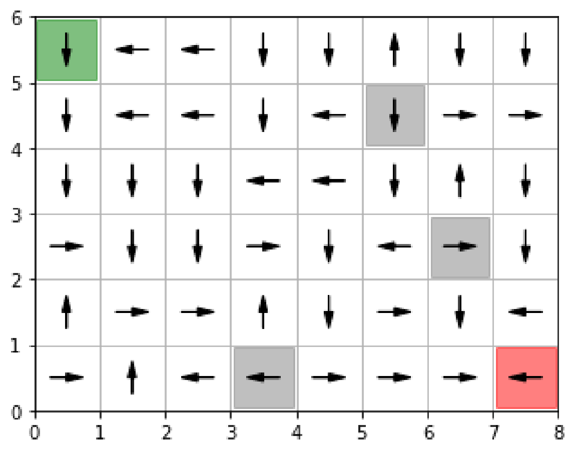
In this part, we finished the Q-learning agent with ϵ-greedy action selection. Next, we implement the value iteration with dynamic programming. We compare the result of value iteration and Q-learning to find the optimal extent of the result of Q-learning.

1.2 Question 1

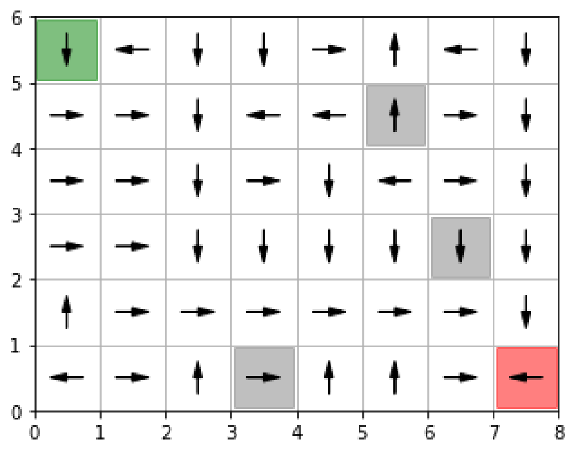
"wallkInThePark" is more difficult to learn in, because it has more states, so the state space and the combination of state-action pair is larger than "theAlley". That also means in this map, Q-learning has a higher probability to stuck in a local optimal point.

1.3 Question 2

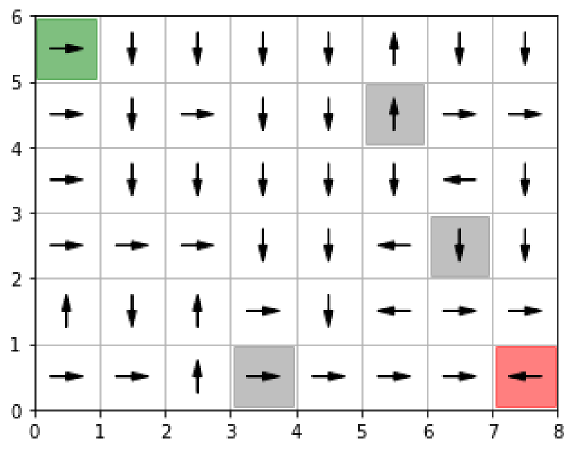
Four results of Q-Learning agent is shown in Figure 1. The policy of Value Iteration after 1000 episodes is shown in Figure 4.



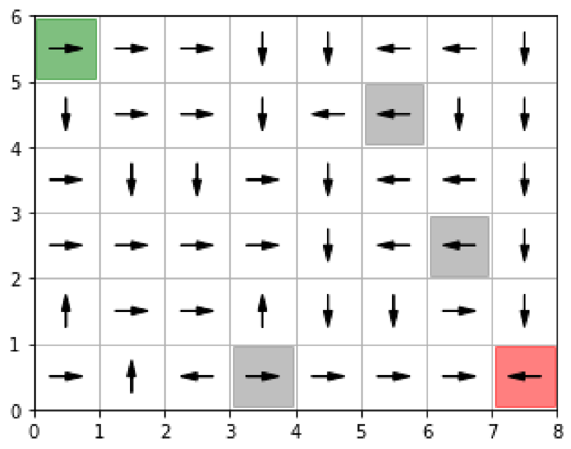
(a) wallkInThePark: case 1



(b) wallkInThePark: case 2



(c) wallkInThePark: case 3



(d) wallkInThePark: case 4

Figure 1: walkInThePark: Q-learning

In each case, Q-learning has a different final policy, and they are all not the same as the final policy of value iteration. However, the result of Q-learning does have some similar property with the final policy of value iteration. For example, they all try to avoid step into the hole in the last row. Considering the value iteration has excellent capability to find the global optimal, Q-learning results are always not the global optimal solution. But it is still can be a (local) optimal policy. There are some possible reasons why the Q-learning result is not the global optimal solution (the final policy of value iteration):

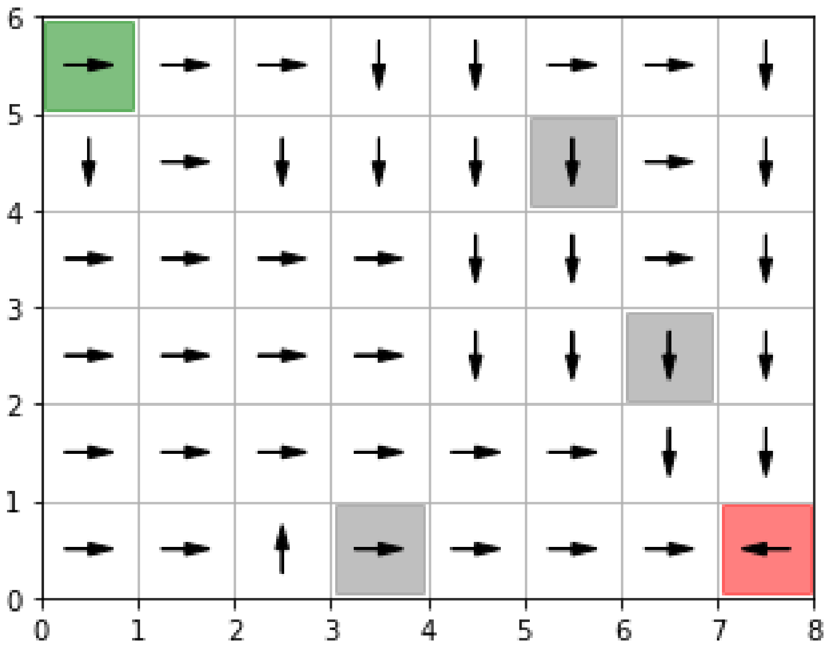


Figure 2: walkInThePark: Value Iteration

1. The episode is not enough, so the Q-learning model is not able to learn all the effect of the hidden model;
2. The episode is not enough, so the effect of the hole may not accurately present the possibility of 0.2;
3. Although we use ϵ greedy to help us explore more of the state space, Q-learning is still probably to step into a locally optimal solution but not explore the optimal global area;
4. A Q-learning agent learns the Q-table based on the response of environment, because there are random part(transition, penalty, etc) in the environment, so the response may not accurately implies the actual environment model.

1.4 Question 3

The optimal Q values calculated by value iteration (1000 iteration episodes)are:

state	Left	Down	Right	Up
state1	0.	0.	0.	0.
state2	0.	0.	0.	0.
state3	0.	0.	0.	0.
state4	0.	-0.03201494	-0.2561195	-0.03201494
state5	-2.06402987	-2.04258095	-0.35572152	-2.04258095
state6	0.17095759	1.91949009	2.37265049	1.91949009
state7	2.1947017	2.43608594	2.70218528	2.43608594
state8	2.49952138	2.77443121	3.07748879	2.77443121
state9	0.84667713	1.40976887	3.50491779	1.40976887
state10	3.74204896	5.88334788	6.76948971	5.88334788
state11	6.26177798	6.95047953	7.70969661	6.95047953
state12	7.13146936	7.91582391	8.7804878	7.91582391
state13	10.	10.	10.	10.

1.5 Question 4

As shown in figures below, the first one is the policy learned by q learning and the second one is the optimal policy obtained from value iteration. Both policies are the same from the last “Hole” to the goal and from the start point to the first “Hole”. But there are some differences in between. The state 9’s policy in q leaning is left while the optimal policy is right. This can be caused by the ε-greedy strategy. The q learning with a 0.05

ε makes it prone to take the action with maximum current reward and rarely explore new strategies. So there are only a few agents go through the holes at state 5 and 9. Thus, the update of q values will be very slow because of the small ratio of agents who can reach the goal. As we set the episodes to 1000, it is too small for our algorithm to learn the Q table completely. So the policy of q learning tries to get rid of state 9 rather than go through it and pick up the reward at state 13.

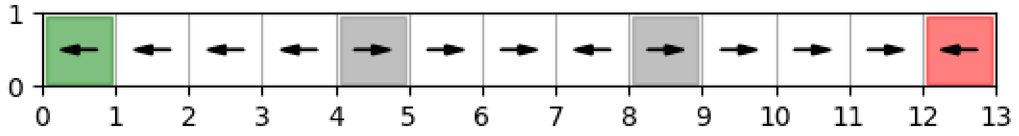


Figure 3: theAlley: policy of q learning

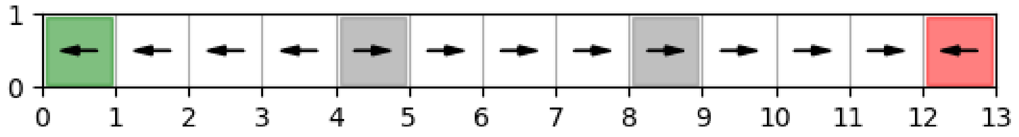


Figure 4: theAlley: policy of value iteration

1.6 Question 5

The optimal Q values calculated by value iteration are:

state	Left	Down	Right	Up
state1	0.84986218	0.86166582	0.94429131	0.86166582
state2	0.87346946	0.96953799	1.07544288	0.96953799
state3	0.99478467	1.10419604	1.22480995	1.10419604
state4	1.1329492	1.25755661	1.39492244	1.25755661
state5	0.29030326	0.55721724	1.58866167	0.55721724
state6	1.71951205	2.77349742	3.19819802	2.77349742
state7	2.95833316	3.28370539	3.64239218	3.28370539
state8	3.36921277	3.73977559	4.14827999	3.73977559
state9	2.83715899	3.38418886	4.72442999	3.38418886
state10	4.62009774	5.99310398	6.76948971	5.99310398
state11	6.26177798	6.95047953	7.70969661	6.95047953
state12	7.13146936	7.91582391	8.7804878	7.91582391
state13	10.	10.	10.	10.

1.7 Question 6

As shown in figure 7, the optimal policy obtained by the value iteration choose to go right all the time. This is because the expected reward (approximately, $0.8^x \cdot 10$ where x is the length of shortest path to the goal) is larger than the penalty of go through two holes(approximately, $0.8^a \cdot 10 + 0.8^b \cdot 10$ where a,b is the length of shortest path to different holes)now. So, the agent can go to the goal from the start point without a negative reward.

Figure 5 and 6 show that the policy of q learning varies in different rounds. Some policies choose to go through two holes which is similar to the optimal policy, some policies choose to go through only the last hole and some policies don't go through any holes. Compared to the result of question 4, the q learning with a -5 penalty is more prone to go through the holes. This is because a smaller penalty makes the reward at the goal more attractive.Thus, more agents can go through the holes and update the q table at the same time. There is only a small chance that the q learning algorithm can obtain an optimal policy. This is caused by the same reason as the question 4. There are only a few agents who can go through all two holes. So, the q values are not accurate and the policy is also not optimal. However, there is still a small chance that the q learning algorithm can get the optimal policy, because some agents can go through the holes by chance.

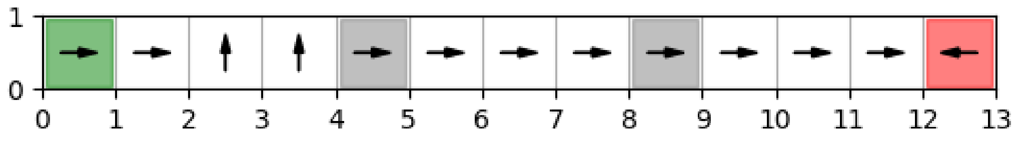


Figure 5: theAlley: policy1 of q learning

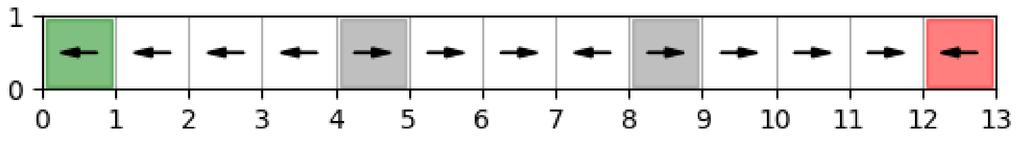


Figure 6: theAlley: policy2 of q learning

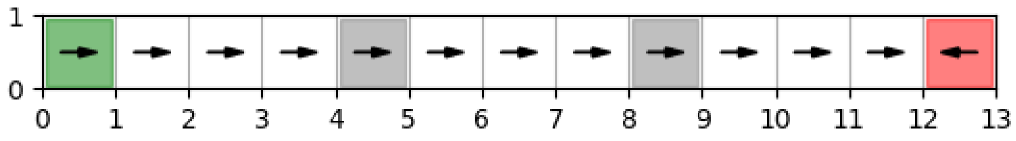


Figure 7: theAlley: policy of value iteration

1.8 Code Exercise 2

We modified the exploration strategy at selectAction function to make it find optimal policy quicker and more often.

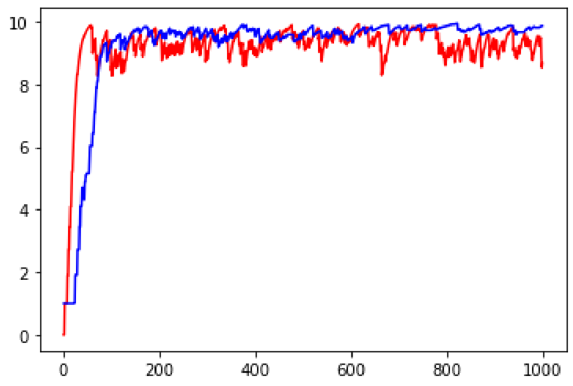
1.9 Question 7

In the ε-greedy exploration, we pick up the action with maximum reward or pick up an action randomly. This strategy should choose an ε carefully and don't make full use of the data in Q table at the same time. To overcome this shortcoming, we can use the boltzmann distribution [1] [2]to determine which action to be taken. The possibility of different actions can be calculated by following equation.

$$P(k) = \frac{\exp(Q(k)/C)}{\sum_{i=1}^{K} \exp(Q(i)/C)} \tag{1}$$

It is obvious that every action has a chance to be taken. However, actions with smaller Q value or negative Q value will have a smaller chance to be taken. This can be better than the ε-greedy strategy because it takes the differences of q values into account rather than just chooses the action with maximum Q value. With boltzmann distribution exploration, we found that the q learning algorithm always find the optimal policy when the penalty is -10 and find a roughly optimal policy when the penalty is -5 in "theAlley" map. For "wallkInThePark" map, q learning with boltzmann distribution also reaches the optimal policy more frequently and converges with a quicker speed at the same time.

Based on "walkInThePark" MAP, we set the maximum episodes number to 1000 and 5000, and plot the max Q value in each iteration of normal Q-learning (red line) and Q-learning with boltzmann distribution (blue line). The result is shown in Figure 8. It can be seen that the fluctuation of blue line (boltzmann) is slighter. That means the modified way do reach the optimal value more often. Which proof the availability of boltzamann method.


(a) comparison 1: 1000 episodes

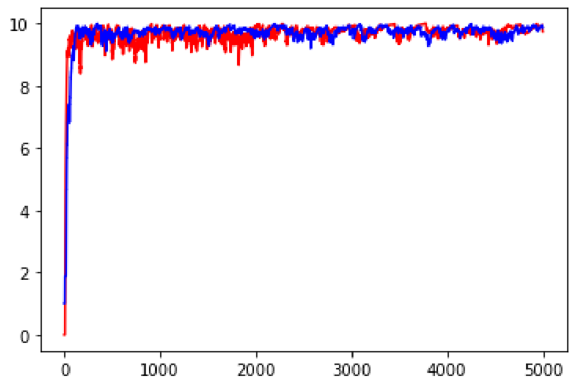
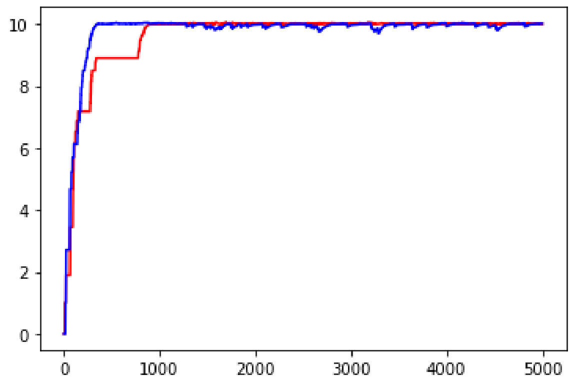

(b) comparison 2: 5000 episodes

Figure 8: wallkInThePark Comparison, red for normal method, blue for modified method

Based on "theAlley" MAP, we set the maximum episodes number to 5000, and plot the max Q value in each iteration of normal Q-learning (red line) and Q-learning with boltzmann distribution (blue line). We plot the result of penalty equal to -5 and penalty equal to -10. It seems that in both case, the new exploration strategy can help the agent to find the optimal Q value faster.


(a) comparison 1: penalty -5

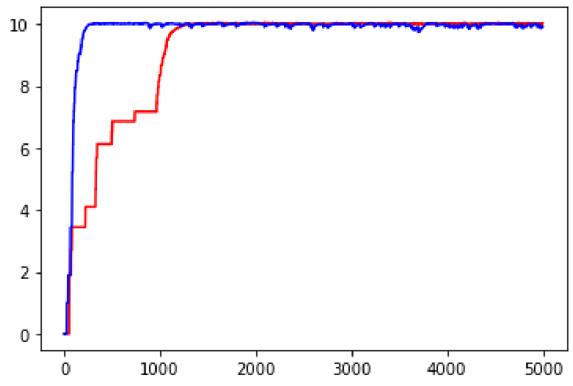

(b) comparison 2: penalty -10

Figure 9: theAlley Comparison, red for normal method, blue for modified method

2 Deep Q learning

2.1 Question 8

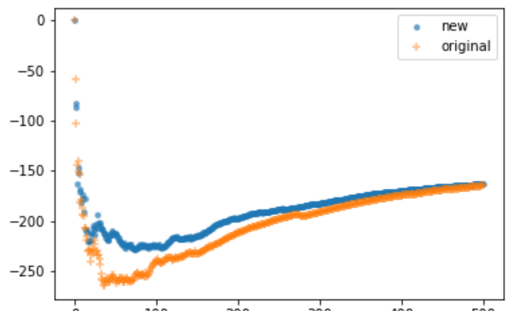
From the behavioral observation, the behavior of the small spacecraft changed from a direct crash at the beginning to a smooth crash at the end. It can basically fall within the range of the flag, but the spaceship does not fall slowly, and there are also cases where the agent has been hovering, resulting in overtime.

Observing from the final reward score, the same conclusion can be obtained, that is: the craft scores very low at the beginning, and then gradually learns to fall steadily, which manifests as an increase in the score. But the agent is not fully learned, so the score is not very good.

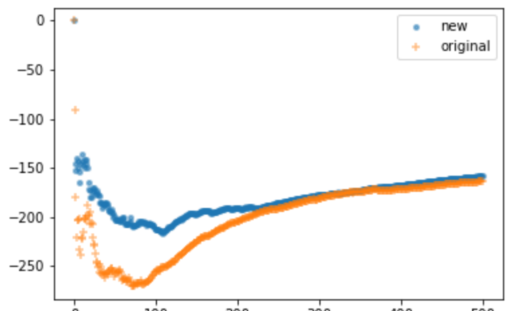
In conclusion, a learning process can be observed, but the final result is not so good.

2.2 Question 9

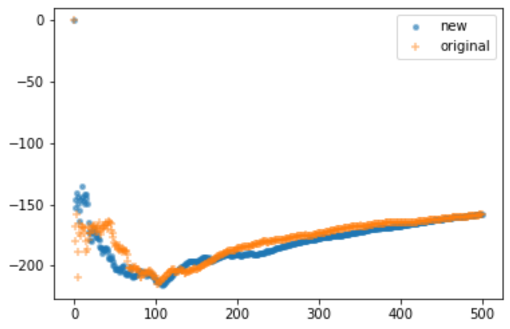
The agent's performance of the aircraft is the same as before. From the beginning, the performance was poor and gradually improved.



(a) case 1: New method can improve the performance



(b) case 2: New method can improve the performance



(c) case 3: New method can't improve the performance

Figure 10: comparison between memory method and non-memory method

Compared with non-memory learning, the learning method that adds memory can improve the performance of the aircraft with a high probability as shown in Figure10, that is, the overall score of the aircraft is improved.

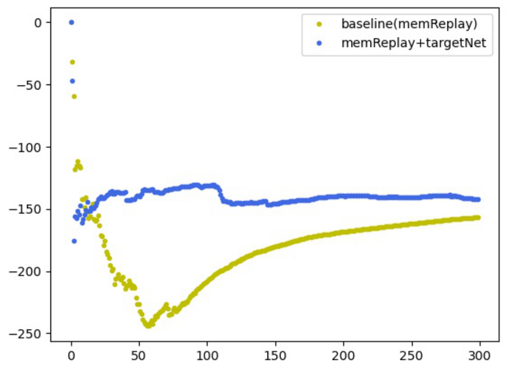
2.3 Question 10

After employing the target network, the aircraft's behavior and the returned reward is similar to the baseline. The returned reward first drops drastically as the aircraft explores the environment and bounces back after a number of episodes as the agent is approaching the optimal policy. While compared to the baseline, by adopting the target network, the agent shows faster and more stable convergence to the optimal policy , as is shown in case 1 in Figure11(a). Such improvement should be attributed to the target network which helps alleviating the bootstrapping of DQN. Occasionally such improvement can be less evident to none as is shown in case 2 in Figure11(b). In general the target network is effective for the learning of DQN.

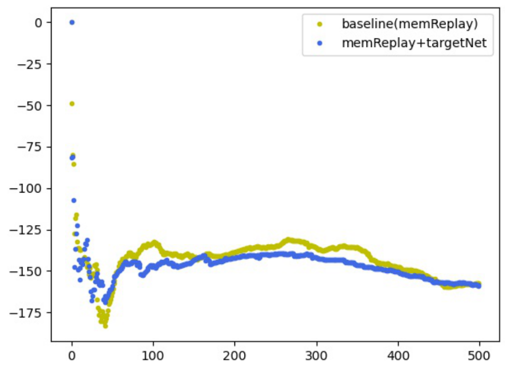
References

[1] Brian Sallans and Geoffrey E Hinton. "Reinforcement learning with factored states and actions". In: *Journal of Machine Learning Research* 5.Aug (2004), pp. 1063–1088.

[2] Zhihua Zhou. *Machine Learning*. 2016.



(a) case 1



(b) case 2

Figure 11: comparison between baseline(memory-replay) and baseline+target network