AI6101 INTRODUCTION TO AI & AI ETHICS

Assignment 1 Reinforcement Learning

Zheng Weixiang

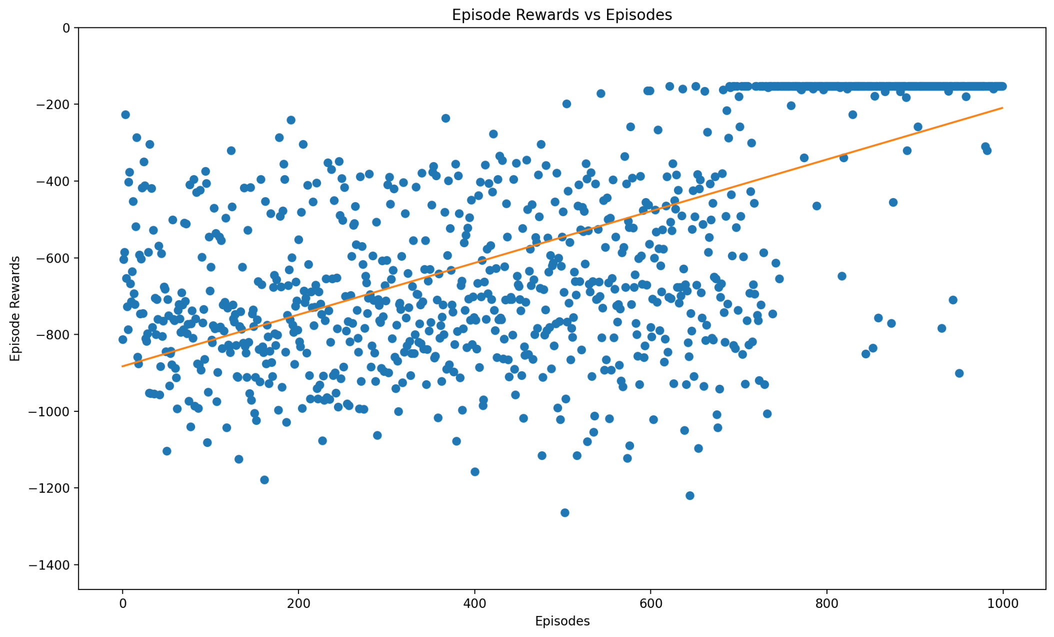
G2103278G

1. Marking Criteria:

Bug-free: correctly implement the code of your chosen RL algorithms	50%
Show or plot the learning progress: episode rewards vs. episodes and Q/V values of each state over time	25%
Show the **correct final Q/V table** and its corresponding policy.	25%

2. Deliveries according to Marking Criteria:
   - The bug-free code implementation (attached as a separate file: *Q_learing.py*)
   - Plot of the Learning process (attached as a separate file: *RewardEpisodePlot.png*)
   - Q Value Table at different times of the training process (Middle_Q_Table.pdf)
   - Final Q Table (attached as a separate file: *Final_Q_Table.pdf*)
   - PDF report: Analysis of the final Q-table and analyze its corresponding policy (this report)

Episode rewards vs. Episodes:

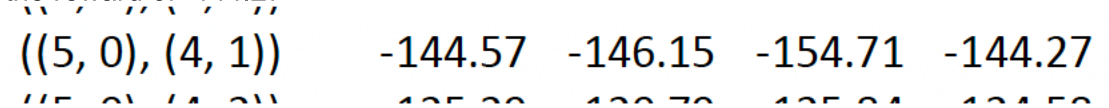


Analysis:
We can see clearly from the scatter plot that at the beginning, the learning process was quite slow, after a certain episode, i.e. around 750th episode, the learning algorithm reached the highest possible value, i.e. -153, for the first time, and since the Q Learning algorithm recorded that experience into the Q-Table, for the next episode, the algorithm will more likely just go ahead and choose the optimal routine, that's why we see a crowded points group around the highest possible value at the end of training. If we plot the best fit line of these scatter plot, we can see even more clearly that although there's a fluctuation when moving forward, every small success, i.e., higher score does contribute to the next episode, and that memory of success makes the overall best fit line upwards. And after a certain point, the algorithm will always

choose the optimal routine unless random.random() < self.epsilon and the algorithm decides to move randomly.

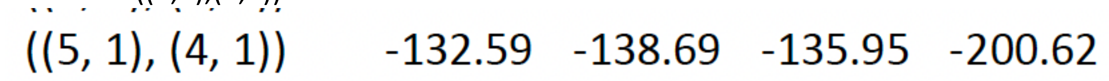
Points made from comparing start, middle and final Q table.
Also, there is an interesting point we can make about the training process. Since the beginning table of Q table values are all zeros as initialized by np.zeros, I didn't attached it as a separate file, but by comparing with the Middle_Q_Table.pdf and Final_Q_Table.pdf, we can see clearly that in the middle of training process, some state are not assigned a value yet, this is because our method use random algorithm to try out different situations, and when num_iteration is just 500, the random exploration might not be significant enough to find out at least one feasible path that could result in a positive reinforcement. However, from both the plot of reward and the Final_Q_Table.pdf, we can see that if at least one random try result in a highest score, i.e. randomly picked the optimal path, the reinforcement process will keep adding score to the existing path, and all following decision will pick that optimal path except the extremely small epsilon situation which was designed to try out some random move especially. This result in a very significant high score, for example ((3, 0), (3, 1)) -106.96 -107.09 -100.36 -92.78, the right action score is so much higher than the other one, since all other direction could no result in a goal situation not matter how, and that's why one winning situation adds a little score to the right direction, but then it keeps going that way, result in a much higher score, i.e. strong preference to go that direction.

Walkthrough of the last episode by reading the final Q-Table:
So next, we are going to walk through the last episode game states and analyze why the agent decided to move like that. At the beginning, the state was ((5,0), (4,1)), i.e. the agent was at the position of (5,0) and the box was at the position of (4,1) and by reading the Q-table, agent saw following: going up has the reward of -144.57; down: -146.15; left: -154.71 and going right has the reward of -144.27



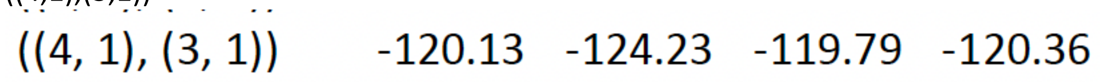
((5, 0), (4, 1))	-144.57	-146.15	-154.71	-144.27

The agent feels going up (-144.57) or going right (-144.27) might be the good option and going right is slightly better than going up, this might due to one arbitrary episode, the agent tried the sequence of actions starting with going right and reached the goal eventually, but for the other option (going up), it would never reach the goal, so there is a difference. For the next step, the state was ((5,1),(4,1))



((5, 1), (4, 1))	-132.59	-138.69	-135.95	-200.62

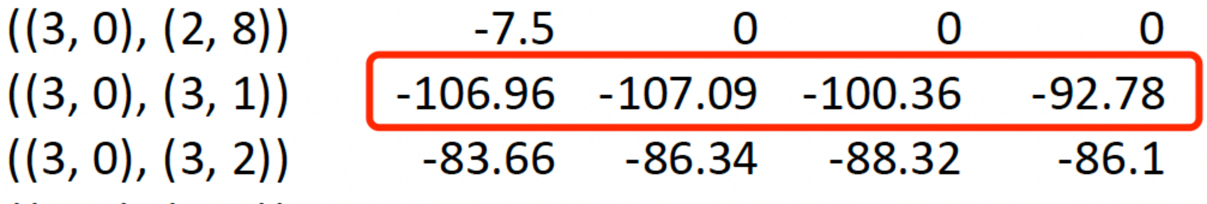
We can see going up is definitely the correct move to make. After pushing the box out, the agent needs to circumvent to its left in order to push it to the left side. This is when state is ((4,1),(3,1))



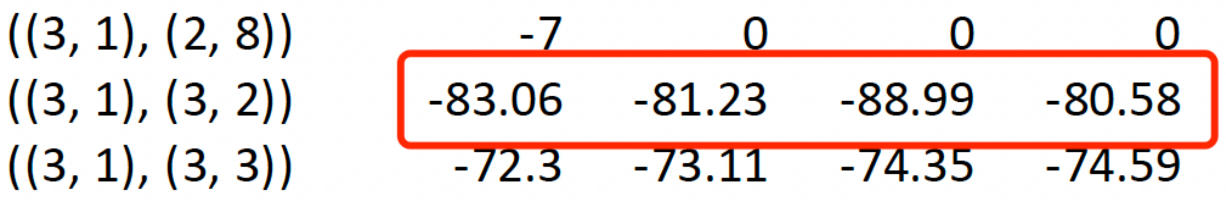
((4, 1), (3, 1))	-120.13	-124.23	-119.79	-120.36

We can see the algorithm indeed recognize the situation and circumvent. I think this is related to the design of reward. The reward was design to be the summation of three values: movement, dist(box, goal), and the dist(agent, box). During the training, I notice during some episodes, the agent continued to push the box even further because it does not want to circumvent to add additional cost, but after it pushed the box to (2,1), it still needs to circumvent and push the box to the right, follow that path, the agent also reached the final

state, it’s just that the path is not the optimal therefore the reward is smaller(more negative).
The next several states are ((3,0),(3,1)), ((3,1),(3,2)) all the way to ((3,12),(3,13))

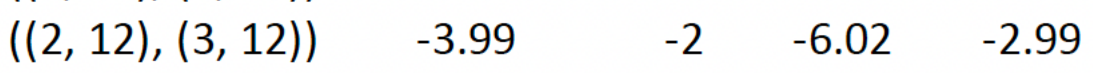


((3, 0), (2, 8))	-7.5	0	0	0
((3, 0), (3, 1))	-106.96	-107.09	-100.36	-92.78
((3, 0), (3, 2))	-83.66	-86.34	-88.32	-86.1



((3, 1), (2, 8))	-7	0	0	0
((3, 1), (3, 2))	-83.06	-81.23	-88.99	-80.58
((3, 1), (3, 3))	-72.3	-73.11	-74.35	-74.59

I will skip these next few state screenshots since they are the same and the policy is just keep moving towards the right.
After we reached the ((3,11),(3,12)), the agent also circumvent and push it downwards according to the best move on the Q-table. Then we got the best rewards -2.



((2, 12), (3, 12))	-3.99	-2	-6.02	-2.99