Report Assignment 1

Group 28

2.1 Development

2.1.1. Question 1

When the algorithm has not learned anything yet, it is at exploration and at this stage the policy typically involves random action selection. Based on the epsilon, the algorithm will either choose a random action or the best action for the current state. The best action is the one with the highest q in the qtable. If there are two or more actions which have the same highest q value, then the algorithm chooses an action randomly from those actions. This way there is no bias towards a certain action as they are chosen randomly.

2.1.2. Question 2

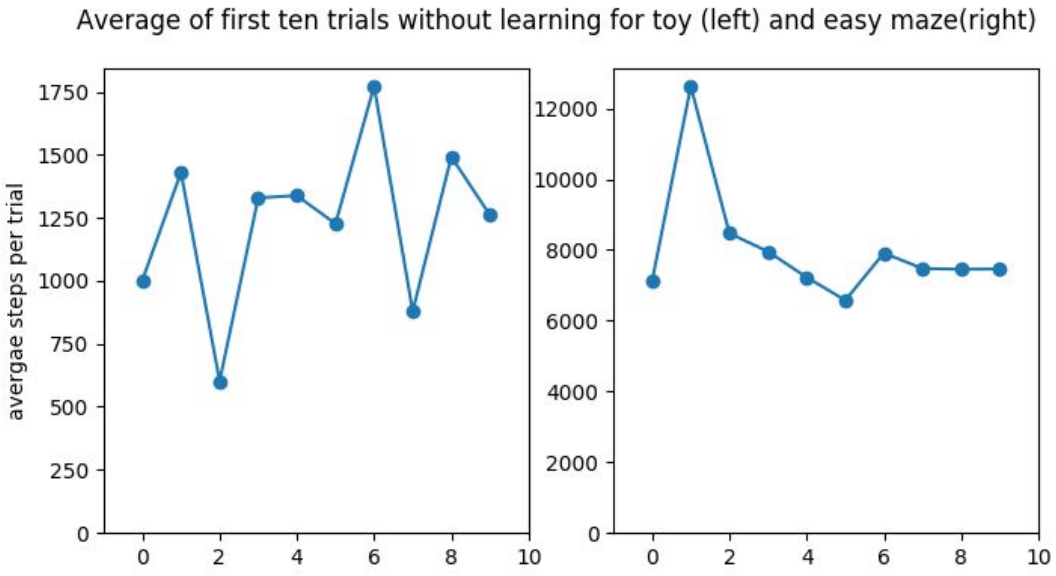
We declared the following variables:
Steps : to keep track of the steps taken by the robot in total;
Steps_trial : to keep track of the steps for each trial;
trials : to keep track of the total number trials taken and
Steps_trials : a list, to add all the steps taken in a trial (steps_trial) to.
Since the agent will not be able to learn yet, we choose a random action for the agent to do. After that we let the agent do the random action. For the state the agent ends up in after the action, we get the reward and save that in a variable.
For each run of the while loop, we check whether the reward is equal to 10, then we have reached the goal, or if the agent has taken 30000 steps.
If the agent has reached the goal, then we add the steps_trial to steps_trials, reset the steps_trial to 0 and reset the position of the agent to the starting position.
If the agent has reached 30000 steps, then we set stop to true, which will end the while loop. This will be a run.

2.1.3 Question 3

Done

2.1.4 Question 4

These graphs show the average of the number of steps taken per trial for 10 trials over 10 runs. The actions chosen by the robot at this stage are completely random so the average steps taken per trial is a similar number for most trials with some trials having more and some less. There is no learning so the number of steps does not decrease over the trials and is completely random but when averaged out over 10 runs, the number of steps per trial are somewhat similar.



2.1.5. Question 5

When the robot moves from one state to another by doing an action, the update_q method updates the q value for that action for the state from which the robot moved. This is done using the update rule:

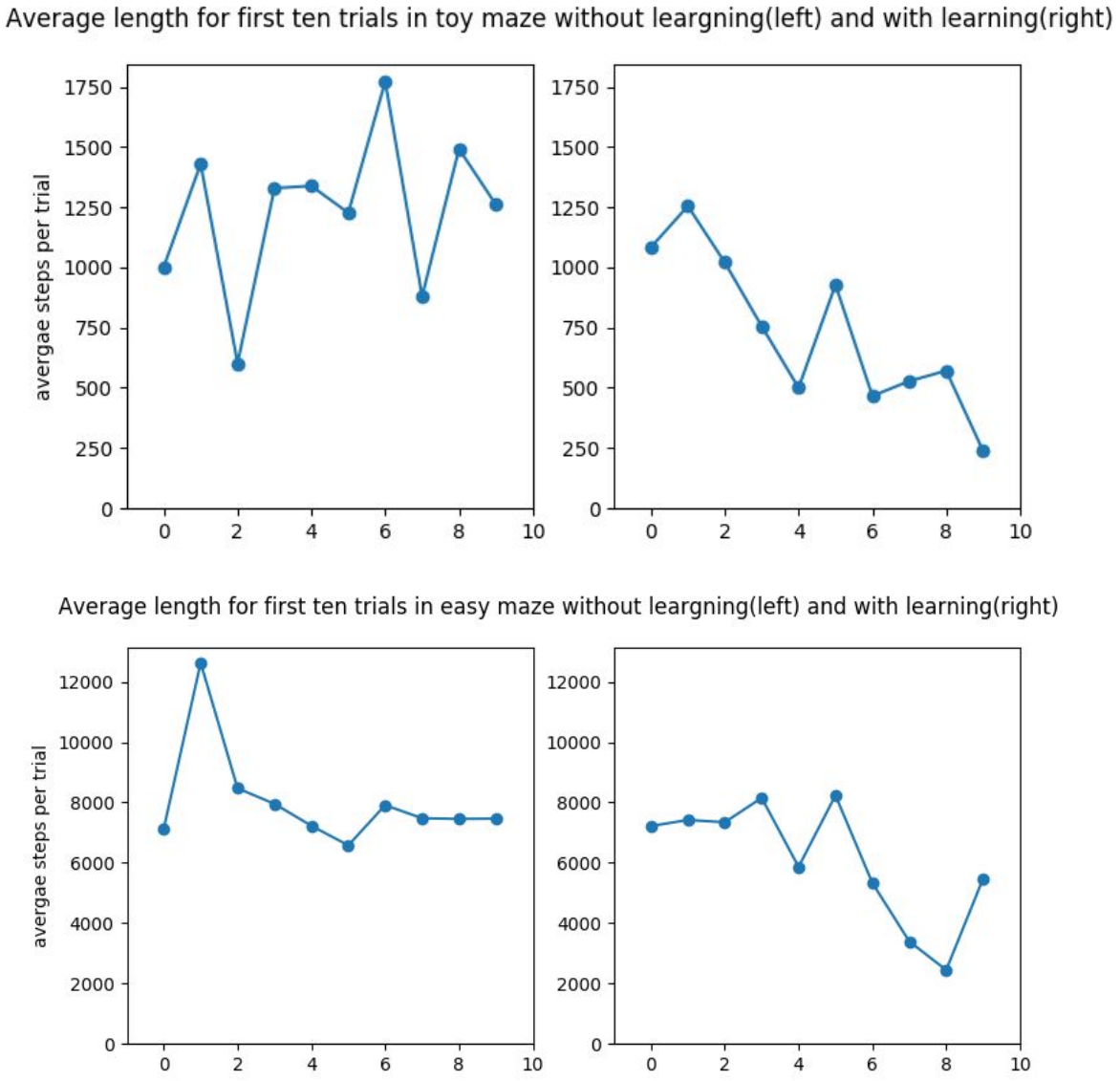
$$Q(s,a)_{new} = Q(s,a)_{old} + \alpha(r + \gamma Q_{max}(s', a_{max}) - Q(s,a)_{old})$$

The robot learns by taking actions and after each action updating Q(s,a).
We adjust our q-values based on the difference between the discounted new values and the old values. We discount the new values using gamma($\boldsymbol{\gamma}$) and we adjust our step size using alpha($\boldsymbol{\alpha}$). Alpha defines how much the new q value is accepted vs the old q value. In the equation above, we are taking the difference between new and old and then multiplying that value by the alpha. This value then gets added to our previous q-value which moves it in the direction of our latest update. The discount factor gamma is used to balance immediate and future reward and in the update rule this factor is applied to the maximum q value for the next state. R is the reward received after completing a certain action at a given state. A reward of 10 is awarded for landing at the target state.

2.1.6 Question 6

Both plots show that the average number of steps taken per trial for the first run is very large, but after running RunMe 10 times, you can see that the average number of steps taken per trial is reduced. After all 30000 steps, the average number of steps per trial are reduced to 24 which is the optimum solution for the toy maze.
For the easy maze, however, it is not the case. The easy maze is so big that a stopping criterion of 30000 steps is not enough to complete 10 trials, thus we increased the stopping criterion to 75000 steps, this way we will have 10 trials. As we can see at the beginning the number of steps taken is very large, but as we run RunMe more, the agent starts to learn and the average number of steps taken per trial is reduced . However, 10 trials is not enough to reach the optimum solution for this maze. If the stopping criterion is set to a very high number like 150000 steps, we see that after enough trials, the optimum solution of 38 steps

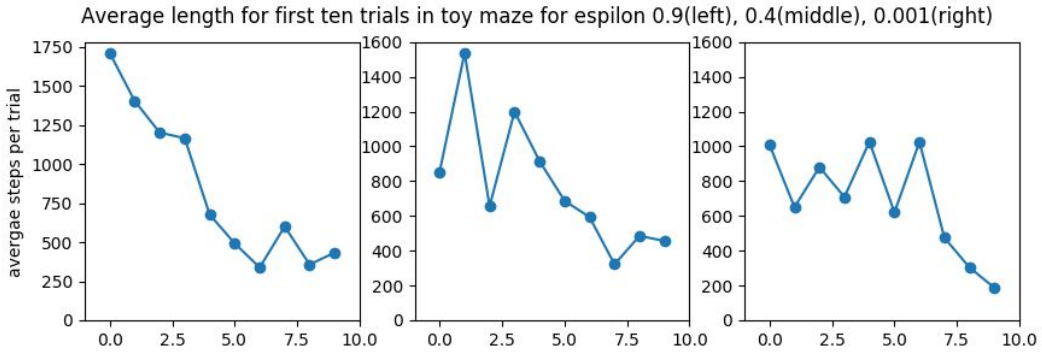
is reached for the easy maze. On the left are the plots without Q-Learning and on the right are the plots with Q-Learning:



2.2 Training

2.2.1 Question 7

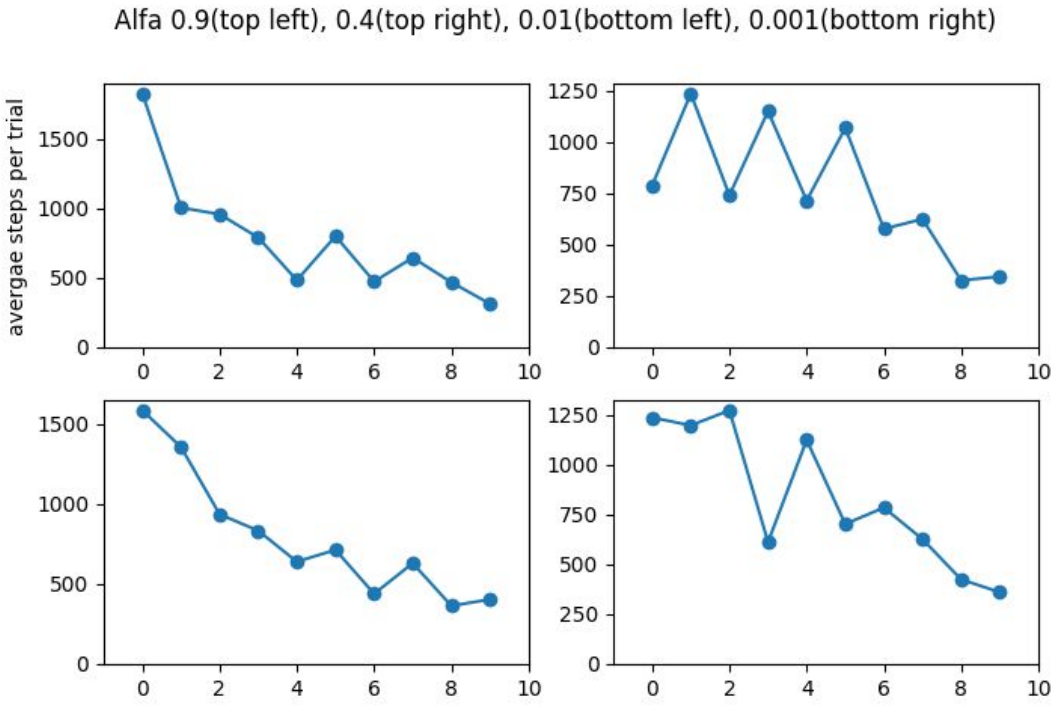
If the epsilon is higher, the agent is more likely to explore, so take a random action. If the epsilon is lower, the agent is more likely to take the best action. Since we only have one goal, we do not have the issue of future reward. The agent just has to reach the one goal. As we can see, with a high epsilon, the agent is more likely to take more steps per trial because of the random steps taken. However with a lower epsilon, the agent is less likely to take a random action, thus the average steps taken per trial is also less. We see that in trial 10, when the epsilon is low, the algorithm needs fewer steps to reach the goal than when the epsilon is high. This is because when the epsilon becomes low, over the trials as the robot learns, the number of steps required will converge to the optimum solution.



For the leftmost graph, epsilon of 0.9: the steps taken in the last trials was more than 100.
For the middle graph, epsilon of 0.4: the steps taken in the last trials was between 40-60.
For the rightmost graph, epsilon of 0.001: the steps taken in the last trials was between 24-26.

2.2.2 Question 8

Alpha is the learning rate of the q learning algorithm and is set between 0 and 1. An alpha of 0 would mean that the algorithm does not learn at all and the q values are never updated but an alpha of 0.9 would mean that the algorithm learns quickly.
For all alfa the final trials took between 24-34 steps.



2.2.3 Question 9

If you have a high epsilon, the agent is more likely to explore, so there is a higher chance to find delayed reward, but if there is no delayed reward, you wasted steps.
If you have a low epsilon, the agent will most likely exploit, so take the best action. This way

you maximize the reward every action, but the accumulated reward may be less than when you explored more, since there may have been a delayed reward.

2.3 Optimization

2.3.1 Question 10

After adding the second reward of 5 at the top right and running the program a couple of times, the number of steps that are taken are reduced from 24 to 21.
This probably happens because the second goal is closer to the starting point of the agent and will thus be reached earlier, leading to the agent learning how to reach a goal faster, even if this goal has a lower reward.

2.3.2 Question 11

At first we start with an epsilon of 1 because we have not learned anything yet and want to explore rather than exploit. Everytime the robot reaches the state (9,0) which has reward 5, the epsilon decreases by 0.1% and when the robot reaches the state (9,9) which has the reward of 10, the epsilon decreases by 5%. Although state (9,0) is closer to the starting point of the robot, we want the robot to aim for state (9,9) as it has a higher reward. This is the reason why the epsilon decreases more when state (9,9) is reached and less when state (9,0) is reached.

2.3.3 Question 12

We set the gamma lower, thus the agent will now prefer near-term award over long-term award. This means that the agent will value the current reward more than future rewards that may come. Thus our agent will reach the closer goal with a reward of 5. Thus, in the toy maze, the optimum value is 21 when the gamma is set lower and the optimum value is 24 when the gamma is higher.