Michael (Zhaochuan) Qiu

zqiu5

ECE 448

MP1 Part 2

(1) WordWithoutThief

a. discount = 0.9, rate = 0.1, epsilon = 0.0

The robot moves north from the starting position and then goes back and forth between the tile above it and it current position. This is because when epsilon is 0, this means that the Q-Learning Algorithm is purely greedy, and it does not incorporate exploration because there is a probability of 0 that the agent will execute randomness (except for when there are multiple optimal actions). Since the algorithm is purely greedy, the agent does not have a chance to explore its surroundings to learn about the environment. It simply executes the action that gives it the most rewards at the next state. Since initially, the robot is surrounded by slippery tiles and doesn't know about the rewards gained from delivering the packages, it simply moves back and forth between two tiles.

b. discount = 0.9. rate = 0.1, epsilon = 0.1

In this scenario, the robot is able to deliver both packages while trying to minimize the penalties by stepping only on the less slippery tiles. Because epsilon is 0.1, there will be a small chance at every step that the robot will explore and execute a random action rather than be greedy. Because of this added exploration factor, the robot gets to learn about its environment, therefore allowing it to successfully deliver both packages. However, the path the robot took to deliver the packages wasn't the most optimal one. The robot passed through multiple slippery tiles on the way when it could have just avoided them and stepped on a non-slippery tile. This was because the epsilon may have been a tad too small.

c. discount = 0.9, rate = 0.1, epsilon = 0.5

In this scenario, the robot was able to deliver both packages, and the path that the robot took seemed to be a better one than when epsilon was 0.1. Due to the large epsilon, the robot was able to explore almost all of its surroundings and determine the optimal path through the right amount of both exploitation and exploration.

(2) WorldWithThief

a. discount = 0.9, rate=0.1, epsilon=0.05, doesn't know thief

The robot moves to the upper left corner of the grid and gets stuck there. The rewards in episodes.txt are largely negative, meaning that the agent runs into the thief constantly and accumulates negative rewards. Therefore, the expected discounted reward converges to a negative number, which is undesirable in this problem. For this reason, the robot cannot determine the optimal actions to deliver both packages without running into the thief.

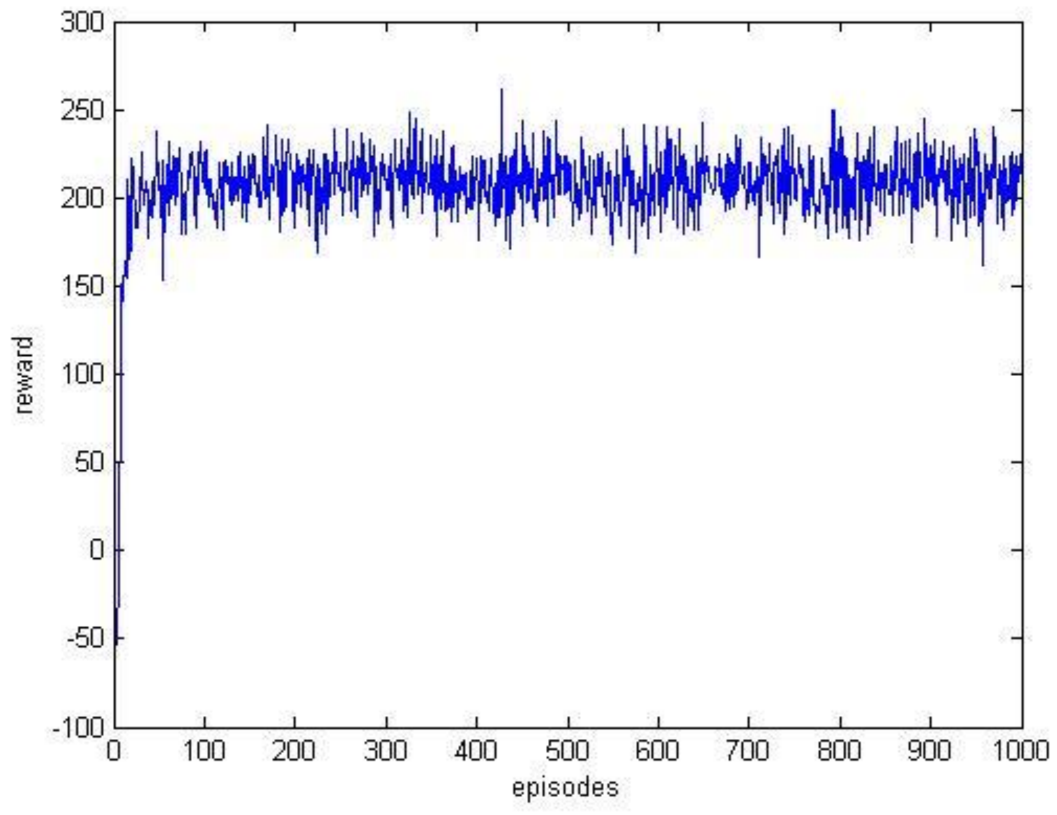
b. discount = 0.9, rate=0.1, epsilon=0.05, knows thief

The robot successfully delivers both packages without meeting the thief because this time, the agent knows that the thief is there. Because of this, the rewards shown in episodes.txt are largely positive, and the expected discounted reward converges to a positive value. However, the values for the rewards shown in episodes.txt are not optimal due to the values of the learning rate and epsilon not being ideal.

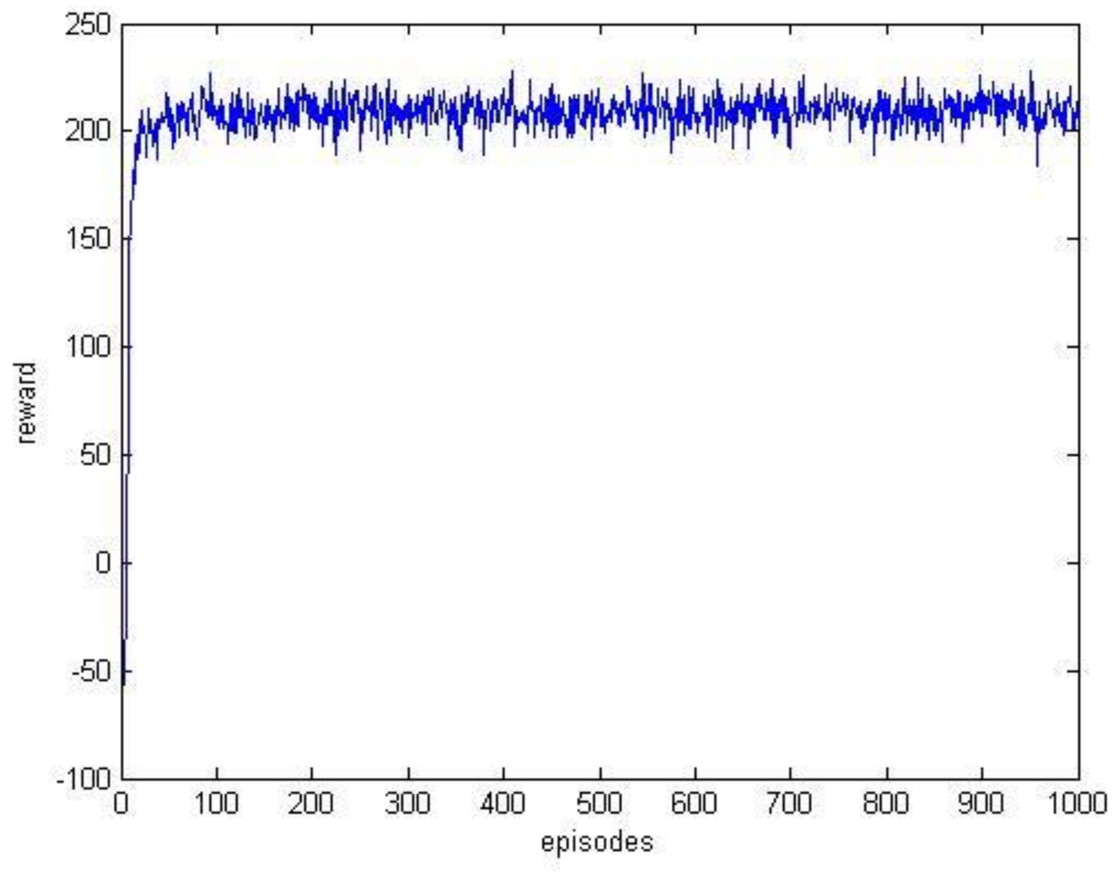
c. The optimal settings are when rate=0.22 and epsilon=0.1. In order to determine the optimal epsilon, I kept the rate epsilon constant at 0.1 and used the hill climbing method and tried increasing epsilon from 0.05 until the rewards in episodes.txt went from increasing to decreasing. Through this method, I determined that the optimal epsilon was 0.1. I applied the same procedure to find the learning rate, keeping epsilon constant at 0.1. By increasing the rate from 0.1, I searched for the learning rate that would yield the greatest values in episodes.txt, and I found that the optimal learning rate was 0.22.

(3)

A. Expected Discounted Reward (1 simulation)



B. Expected Discount Reward (10 Simulations)



After running the simulation once, I plotted the results in MATLAB. I noticed in the plot that the agent improves its rewards as the number of episodes increases from 0 to about 25 from a negative reward to about 200. After that, it appears as though the agent's rewards converges to about 200, but there is still a noticeable variation from 200. This variation is due to the randomness that occurs in the algorithm. However, after running the simulation 10 times and averaging the results, I noticed that the variation from 200 decreased substantially. By averaging over many simulations, the randomness of the simulations becomes less noticeable due to the law of large numbers, so as the number of simulations increases, the less noise there will be in the plot, and we will be able to see very clearly (almost) the exact value that the reward converges to.