Meera Parat
9/24/14
CS 440

MP 1 - Reinforcement Learning - Part 2

1. WorldWithoutThief
   A. discount factor = 0.9, learning rate = 0.1, epsilon = 0.0, 1000 episodes, and 10,000 steps: The reward started out very negative (-16.5 for the simulation that I did), and the reward never went that low ever again. The reward stayed a single digit negative number, 0.0, or a very small positive number. It never went very high because this agent was greedy. Since we set epsilon to 0, there were no random actions except for when there was a tie, and therefore the agent kept trying to find an optimal path. Once a q value for a state became negative, the agent never went back there event though it might have been the optimal path in the long run.

   B. epsilon = 0.1: I observed something different - where the negative phase of the rewards lasted much shorter. The reward was negative only for about the first 20 episodes, and then after that it was all positive. That too, instead of being in a range from 0 to 200, the positive rewards were mostly in the triple digits, somewhere in the 100s. Thus, the average reward was probably much higher in this scenario where epsilon was 0.1 than in the previous where epsilon was 0.0.

   C. epsilon = 0.5: The results of this scenario was between the results of the first two cases. The negative reward period lasted longer than when epsilon was 0.1, but shorter than when epsilon was 0.0. That too, the average of the positive rewards was not as high as when epsilon was 0.1. These rewards had more double digit values that the previous case, but still were in the 100s or 200s now and then. This shows how too little or too much (in such a small scale) randomness can drastically change the outcome, and how there seems to be an ideal value of epsilon such that the rewards are max.

2. WorldWithThief
   A. epsilon = 0.5, knows_thief = n: All the rewards are negative. It starts out very negative (-42.0 in my simulation), and then gets a little positive, but stays in between -10 and -30. There are only a few single digit negative number, but most of the rewards are negative teen numbers. Basically, not knowing that the thief is around and can steal the robot's packages is harmful to the reward. This shows that the robot is constantly running in to the thief and losing its packages.

   B. knows_thief = y: These results are actually even better than in a Word without the thief and the same parameters! Basically, the rewards for this situation were only negative for the first 6 episodes, and then they jumped to high positive numbers. Within 2 episodes, the reward went from negative to a number in the 100s, and then within around 6 more episodes, the reward was in the 200s and it remained in the high 200 for the rest of the rewards. When the robot knows that there is a thief and where it is, the robot picks the optimal solution based on that information, thus the robot has more input to learn from and can get higher rewards quicker.

   C. Best learning rate = 0.9 and best epsilon = 0.05: With a WorldWithThief where thief is known, I did not find an optimal value for epsilon. I tried very high, low, and medium values and the fluctuation of the reward did not alter significantly. I noticed that regardless of the epsilon, the rewards hovered in the 250-290 range. For the environment that we are dealing with, learning rate of 0.9 is optimal. Our environment is pretty deterministic because the same set or parameters will lead to similar results. The more stochastic an environment is, the closer to 0.1 the learning rate should be, and the more deterministic the environment is, the closer to 1 the learning rate should be. If this were a stochastic environment, the same parameters would lead to a variety of different reward results due to the inherent randomness. Our randomness value is

also set to a very low probability. Looking at our results, the reward is always in the same range of values when we run the simulation with the same parameters, therefore there is not a ton of randomness in our outcomes. Also, by a bit of trial and error with different learning rate and epsilon values, a learning rate of 0.9 results in a higher average value of rewards.

3. Plotting: It's hard to distinguish individual points or trials, but the overall pattern is obvious. The reward almost jumps from a small in magnitude negative number to a large positive number in the 200s within 10-20 episodes, and then plateaus and remains in the same range in the 200 until the last episode. There must be a maximum reword per episode that can be achieved and the robot is hitting this. It gets to the optimal reward very quickly.