Dohn Arms - CS440 MP1 - Experimentation

1. WorldWithoutThief

(a) $\epsilon = 0$

When there was no randomness to actions taken by the learning robot, the first episode had a negative total reward, and every episode after that has a total reward of zero. When the resulting policy was used, the robot would move back and forth between two spaces, without going onto a slippery area ever.

In that first episode, the robot randomly explored the actions, which all had Q values of 0. It inevitably slipped multiple time and got negative rewards, never discovering how to deliver packages for a positive reward. As there was no randomness to its actions, it avoided future actions that had a negative Q value, staying on those that had a zero Q value.

The policy generated limited the robot to two squares next to the company, never crossing a slippery square or delivering any packages.

(b) $\epsilon = 0.1$

When a small amount of randomness was applied to the actions of the learning robot, things were very different than before. The first episode's result is negative, and for the next 20 episodes the results become less negative. After that, the results become positive and within a couple episodes goes to a mean of 140. The results at this point vary quite a bit between 0.5 and 250, but they are always positive.

Again the robot randomly explored the actions as all Q values were zero. The nonzero ϵ made the robot take actions that did not have the best Q values, possibly penalizing the robot. This however forced the robot to explore further areas. Once the robot delivered both packages, it finally received a positive reward. Over more runs, a path of positive Q-values formed from the company to both houses (in some order), and back to the company.

The policy would cause different actions depending on whether it as two, one, or no packages, as one would hope. With two packages, the policy leads the robot to house 2; with one package, it leads to house 1; and with no packages, it leads back to the company. The policy also avoids the more slippery spots; it leads the robot to cross the less slippery spots twice instead of crossing the more slippery spots once.

(c) $\epsilon = 0.5$

When a large amount of randomness was applied to the actions of the learning robot, the learning became less clear. Almost all of the episodes' results are negative, going between 20 and -80, with the mean at around -30.

While the robot explored actions, half of its actions were randomly determined. This led the learning to be very random as well. While values for Q were being updated for the optimal path, Q values were also being updated for the many bad paths, making learning the optimal path much slower. Also, the episodes generally ended in a negative result as a result of all the randomness, hiding the fact whether a good path had been learned or not.

The resultant policy was similar to that of when $\epsilon = 0.1$.

2. WorldWithThief

(a) No thief

When the robot can't see the thief, all the episodes ended with negative values, with a mean of around -13.

The robot effectively ignores the thief's position, and is therefore robbed quite often. An effective path for making deliveries is never found. The slippery spots do not help with the learning.

The resultant policy keeps the robot at the company, as its experience was such that any attempt would be futile.

(b) Thief

When the robot can see the thief, the episode results start out negative, then quickly turn positive, ending with a mean of around 270. It took around 85 episodes for the results to level off.

The robot now treats each of the thief's five positions separately, with different Q values for each position. This results in five times the states to use while learning.

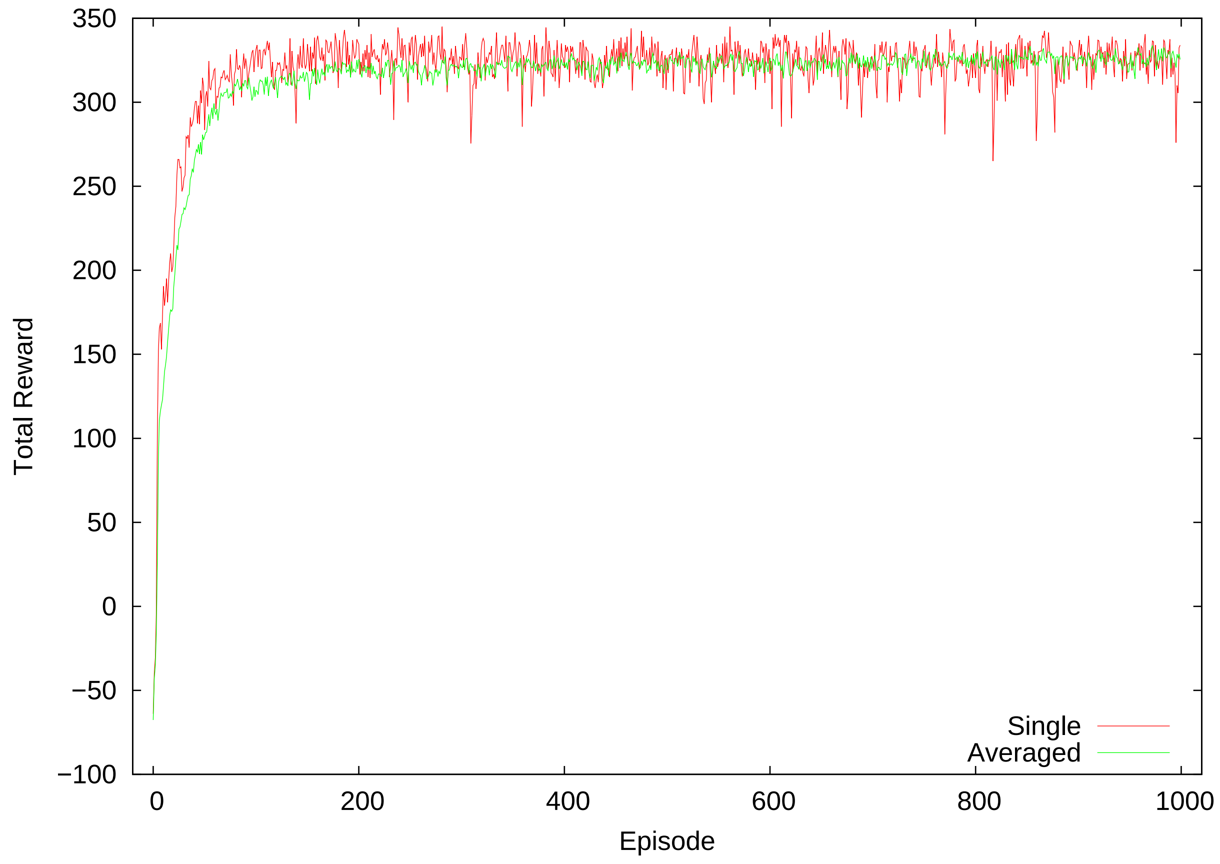
When using the resultant policy, the robot learned to avoid the slippery areas completely, as there is a clear path. When in the vicinity of the thief, it will move to avoid it. The robot's final result is now limited by how hard the robot has to act to avoid the thief.

(c) Best Learning Rate and ϵ

The process I used was that after varying the learning rate or ϵ, I would rerun the simulation and see the resulting rewards. At some point, the rewards for the episodes would fluctuate around some mean. I would vary the learning rate to maximize this mean, then vary the ϵ and maximize again. I did this sequence several times to make sure I was at the maximum.

The best learning rate I found was 0.25 and the best ϵ was 0.01, with a resultant reward mean around 330. If epsilon is big, the randomness negatively affects the total reward, so it makes sense that it is small. The large learning rate allows the robot to learn fairly quickly, so that the lower ϵ was okay.

3. Plots



For one run of the 1000 episodes ("Single" on the plot), the results start off negative, then quickly go positive up to around 320 before slowly drifting to 330. This shows that the robot quickly learned a good policy, and over the rest of the episodes, slowly refined the policy to be be slightly better. The reward jitter is due to the randomness from ϵ and the thief's movement, as ϵ can cause the robot to run into the thief or into the slippery areas (for a penalty), and there are times the robot has to spend more time avoiding the thief than other times (reducing the total reward).

For the averaged episodes over ten runs ("Averaged" on the plot), the basic shape is similar to that of the single plot. The plot is much smoother than the single case, and there is some difference in the mean values in the earlier episodes. The smoothness is from the averaging reducing the probability distribution around the expected value. The difference in the earlier episodes is that for each run, as the learning process is somewhat random, the learning may be faster or slower (and in this case, it was faster than average).