Jay Bensal
bensal2
CS 440

MP1 Part 2 - Experimentation

1. **World Without Thief**

epsilon = 0.0
When I set the epsilon value to 0.0 and tested the QLearning Agent, I saw two different outcomes. The outcome that happened most of the time was that the agent goes straight up the map and once it reaches the top, stays there and does not move. The other outcome I saw was that the agent did not move from the starting state. The agent does not want to cross the slippery path. Since there is no randomness, the agent doesn't falter from what it has been trained.

epsilon = 0.1
After testing the agent multiple times, I saw a variety of different outcomes. A few times, the agent got stuck at the beginning. A few more times than that, the agent trained a good policy where the reward consistently increased. However, in this policy, the reward would decrease by up to 2 points at a time and then continue increasing. Most of the time, I saw that the agent would make a few moves and then get stuck going back and forth between two squares that were right next to each other.

epsilon = 0.5
Based on the outcomes I saw, I think that this was the best value of epsilon for the agent. The agent had a good policy the vast majority of the time and only got stuck a few times. When the agent did get stuck, it was either at the beginning or in between two states in the middle of the map. I think the high chance of picking a random action prevented the agent from being stuck.

2. World with Thief

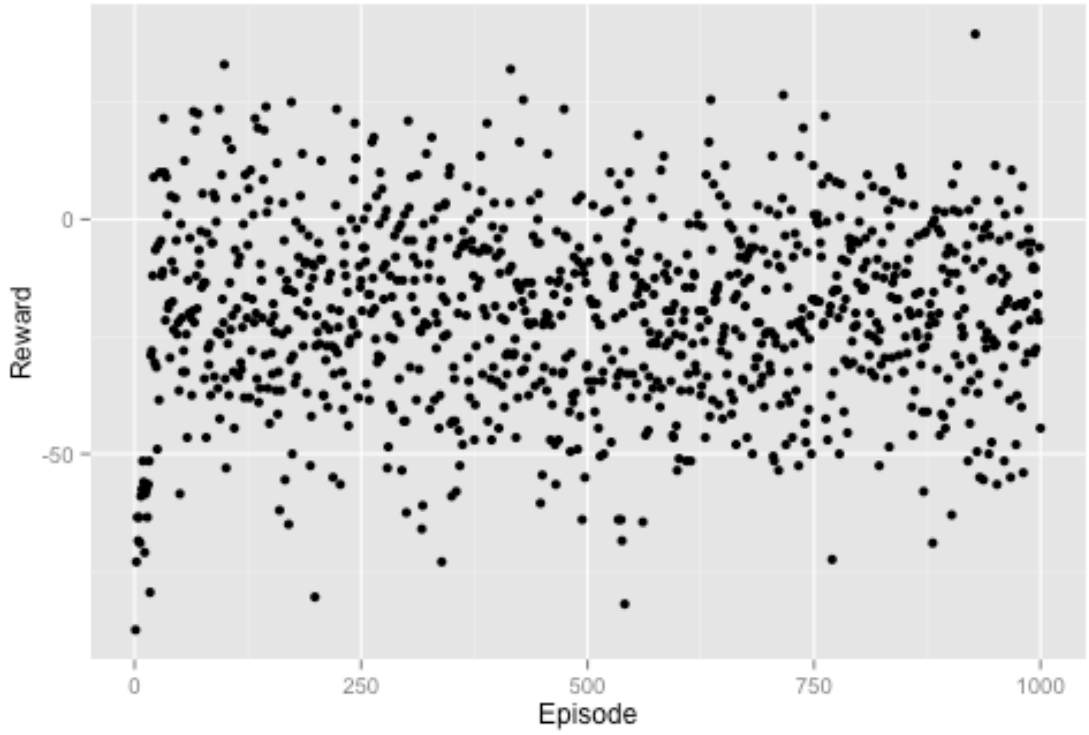
knows_thief = n
Every time I ran the agent it would go straight up on the map and not move. It got stuck on the top of the map. The thief was moving back and forth in the middle of the map.

knows_thief = y
The agent worked really well when it knows about the thief! Each policy that was trained was good, and the reward kept increasing. Having the agent know about the thief made a huge difference in the performance of the agent.

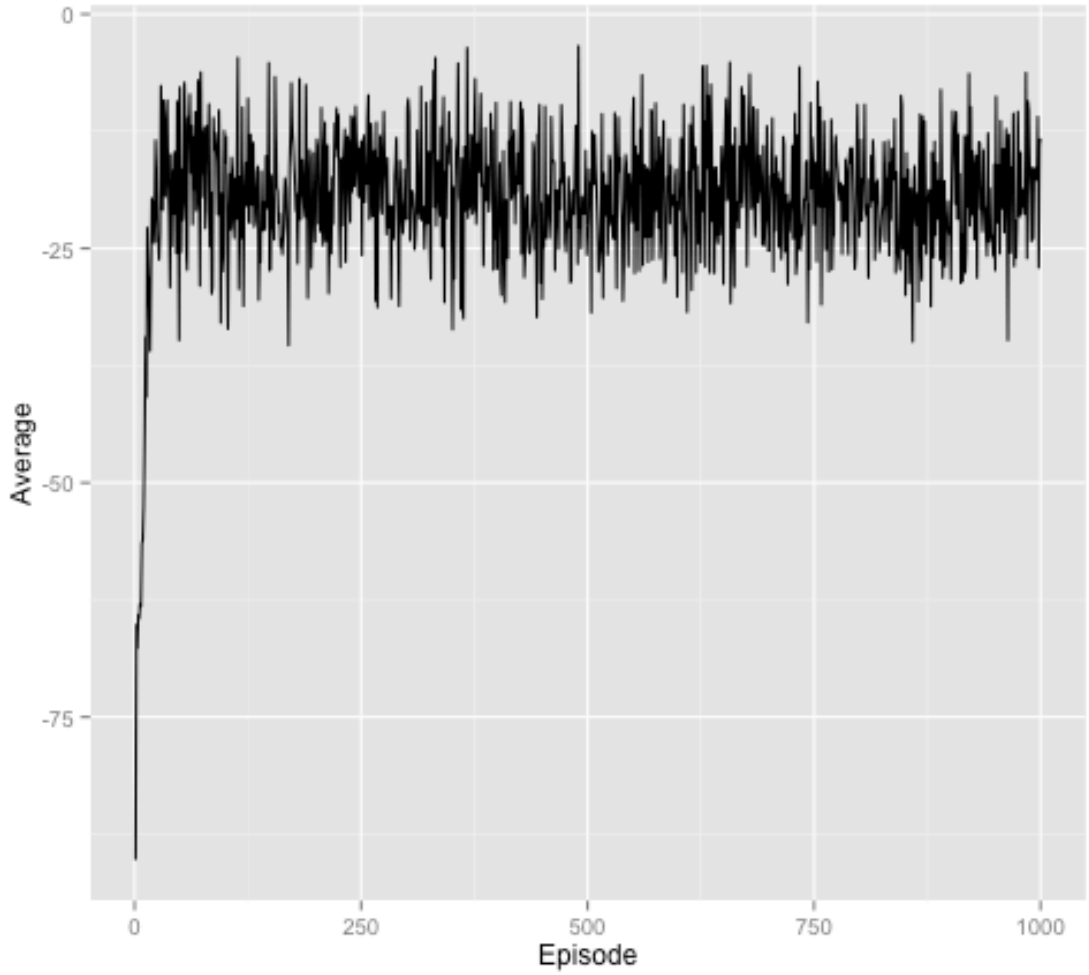
Finding the best rates
In order to find the best rates, I started by trying out the four possible combinations of high/low learning rate and epsilons. With a low learning rate and low epsilon, the

agent moved well without wasting any steps. With a high learning rate and high epsilon, the agent made little progress towards the goal. Because of the high randomness, the agent rarely picks the best state. Because of the high learning rate, this random action affects the states greatly and results in the agent moving around without a good model. A high learning rate and low randomness caused the agent to move really well. This agent was probably the best of the bunch. An agent with a low learning rate and high epsilon worked reasonably well, however high randomness is not always good as the agent may move towards a non-optimal state. The low learning rate accounted for that fact too. I'd recommend a moderately high learning rate and moderately low epsilon (0.7, 0.3) for best results.

Graph



This is the plot of one run. As we can see, the agent slowly gets better as the episodes progress, however there is not a very clear improvement.



This is the graph of ten simulations' expected reward averaged. As we can see, the reward starts sharply negative, but then very quickly rises. This mean that the agent learns how to maximize reward very quickly and then operates at a high level for an extended period of time.