Kevin Hou
CS 440
9/29/2014

MP 1 – Reinforcement Learning

1) a. With epsilon at 0.0, the robot figures out which squares are dangerous and gives negative rewards really quickly (the slippery squares), and learns to avoid them after the very first episode. However, because of how the map is structured, the robot then never attempts to cross into the slippery zone and stays inside the 'box' created by the walls and the slippery squares, and after the first episode, the robot's reward ends up always being 0.0, since it will never try attempting to move onto the slippery squares.

   b. With epsilon as 0.1, the robot will decide to randomly explore sometimes instead of staying in its 'comfort zone'. After the first couple episodes, the robot discovers a route that leads it to the delivery spot, and it will quickly learn to follow that route so it can get a higher reward instead of never attempting to step onto the slippery squares out of fear of a negative punishment. Even after the robot has discovered a good route, it will still decide to randomly explore every so often, which results in the robot occasionally losing rewards as it wanders onto the slippery squares more than necessary.

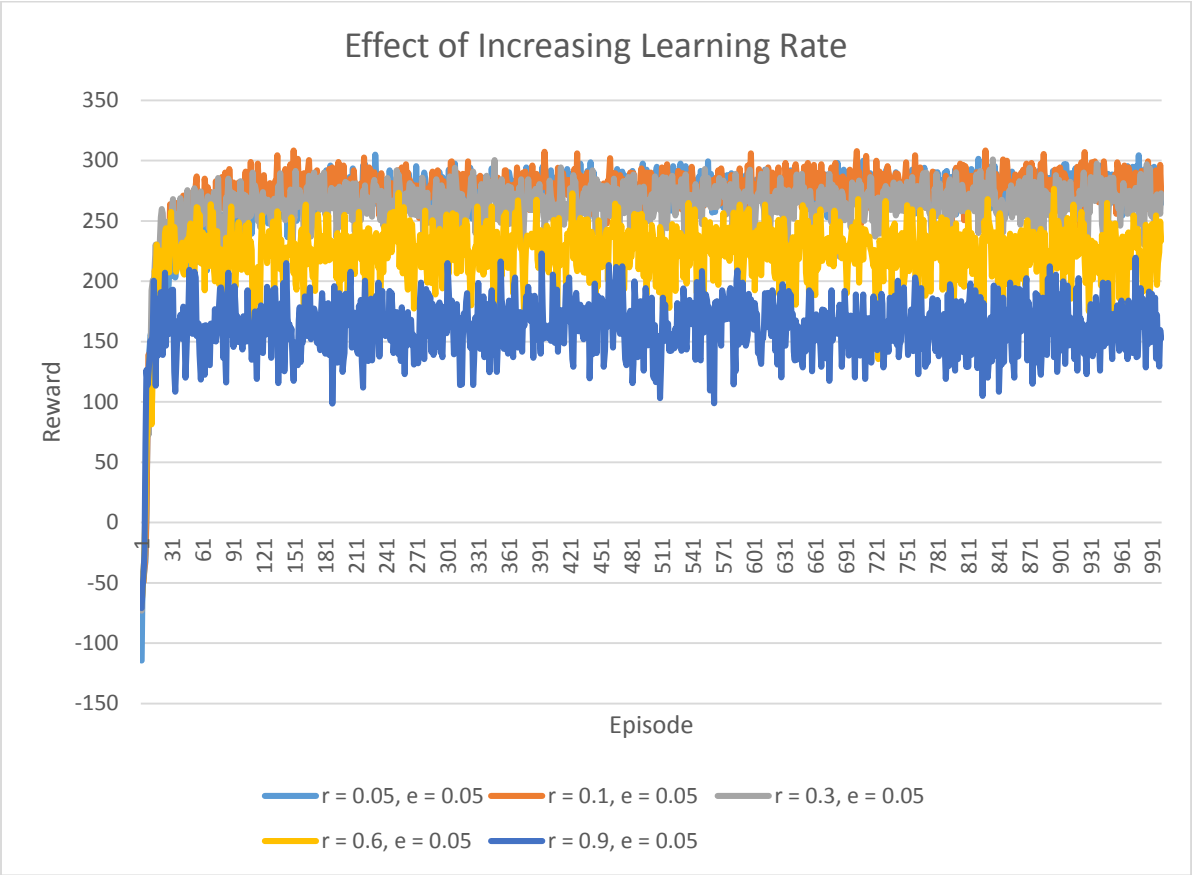
   c. With a high epsilon (0.5), the robot starts to act like the RandomAgent, and its rewards drop dramatically as it decides to explore for alternate routes rather than exploit the best path. The robot occasionally gets to the delivery spot and grabs the reward, but too often it wanders off course and loses points for wandering onto the slippery squares.

2) a. The performance of the robot is extremely poor (though not as bad as the RandomAgent), with most episodes ending with negative rewards. This is probably due to the robot not being able to find the best path as the thief will occasionally block its way and cause the robot to lose a lot of points, which confuses its learning algorithm as certain squares can fluctuate wildly in rewards granted.

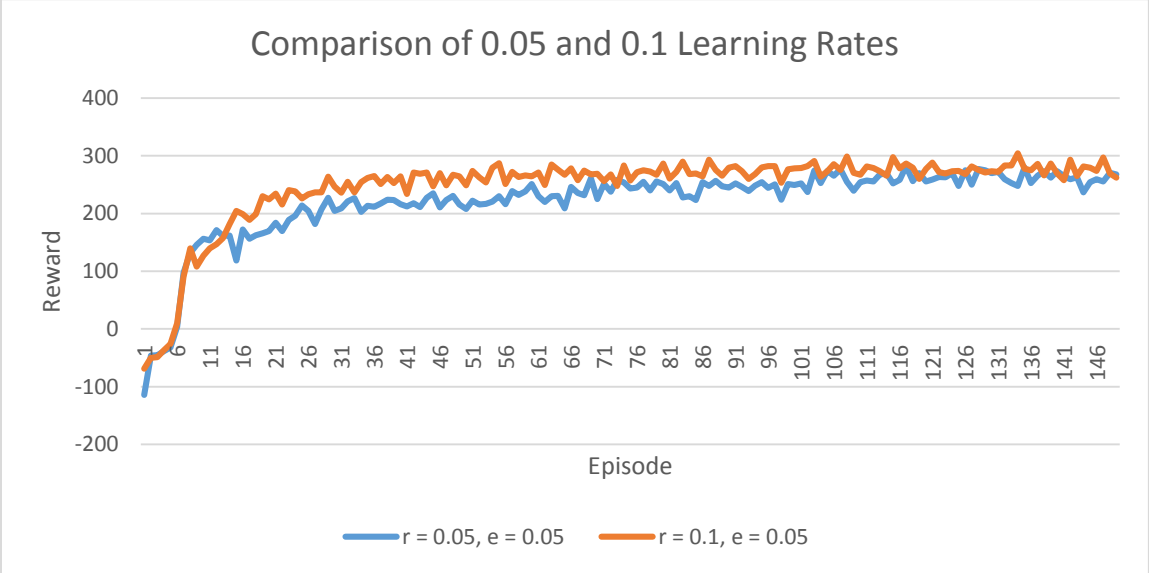
   b. The robot does poorly for the first couple episodes, but then very quickly learns which squares to avoid and will consistently score over 200 points in rewards after a couple more episodes after it finds the delivery point. This is due to the fact that it now knows where the thief is, so it can avoid the square the thief is in currently.

   c. First I tried increasing learning rate with the epsilon set to 0.05 to see what effect it would have on overall performance, shown in the chart below. The best results seems to occur when the learning rate was set to either 0.05 or 0.1. At higher learning rates (0.3, 0.6, and 0.9), the robot's performance fell off fairly dramatically.

Kevin Hou
CS 440
9/29/2014

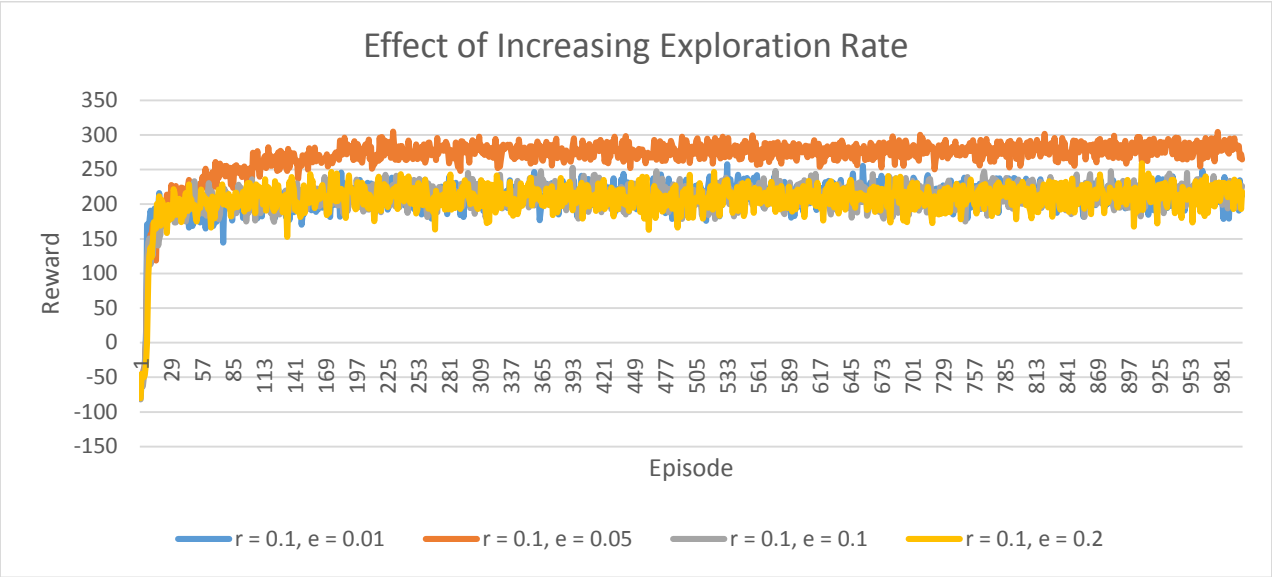


Investigating further, I found that the robot learned faster when the learning rate was set to 0.1, and its total reward gained was slightly higher.



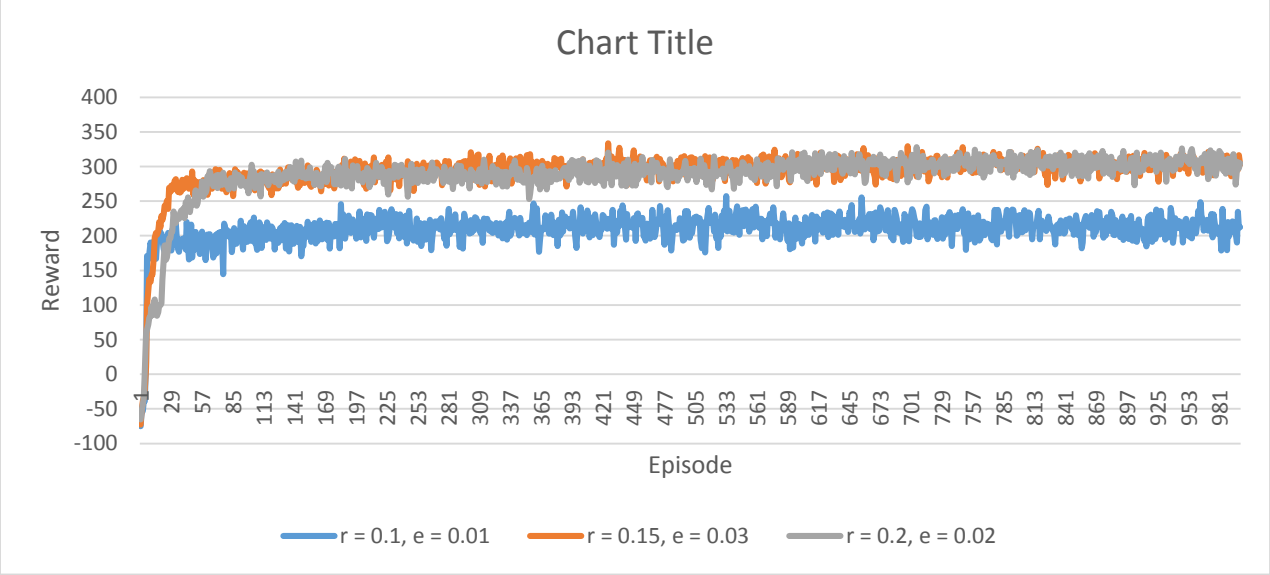
Next, I looked at the impact of increasing or decreasing epsilon (the exploration rate) on the robot. The results are shown in the graph below.

Kevin Hou
CS 440
9/29/2014



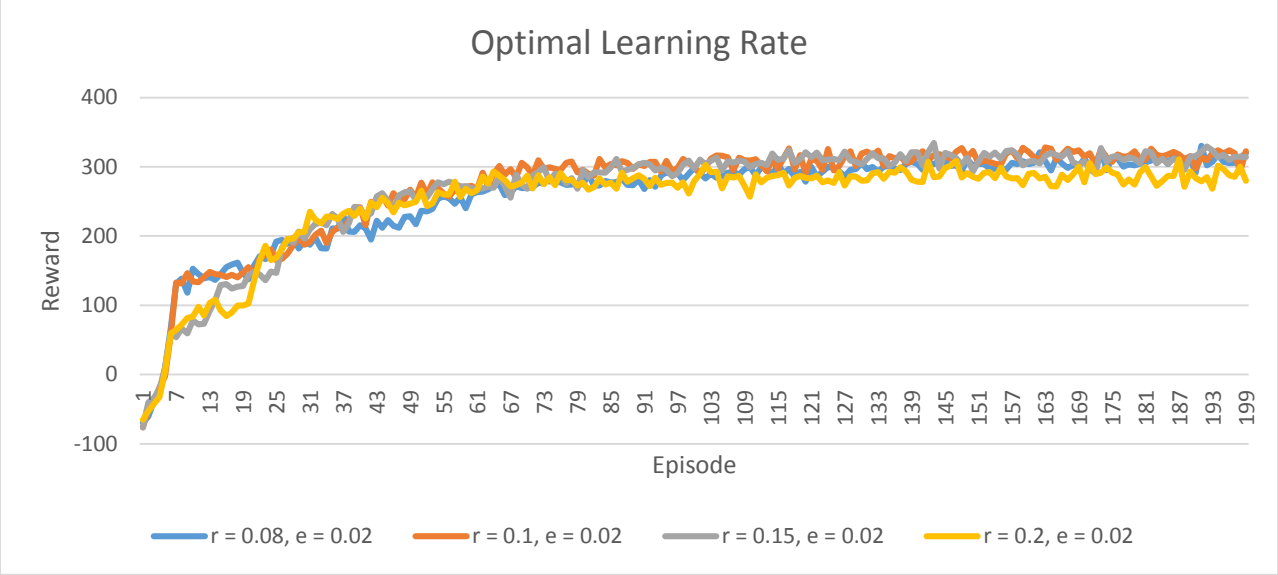
With the learning rate fixed at 0.1, we can clearly see that the robot performs most optimally when epsilon is set to 0.05. With too low an exploration rate, the robot is not able to explore enough to find the most optimal path, and with too high an exploration rate, the robot deviates from the most optimal path too much to try and find a better path that doesn't exist. My conclusions from these preliminary testing is that the learning rate and epsilon cannot deviate too far from the r = 0.1, e = 0.05 range or else there is too much performance loss.

I then tested out arbitrary values for learning rate and exploration rate, keeping in mind not to deviate their values too far.



With this, I found that the optimal epsilon rate is around the 0.02 to 0.03 range. With further testing, I found that increasing the epsilon to 0.04 caused a falloff in performance, while having epsilon in the 0.02 – 0.03 range kept the rewards high. With epsilon as 0.02, there was less deviation in rewards while with 0.03 epsilon, there was slightly more deviation as the robot went off track too much. Because of

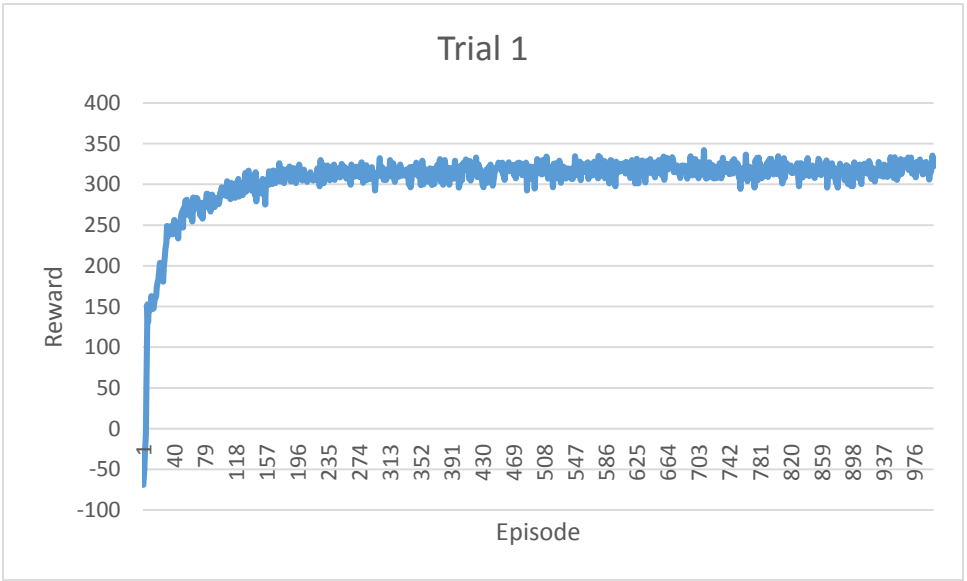
this, I kept 0.02 as the optimal epsilon value.
For the optimal learning rate, I found the range to be rather high. I tried values ranging from 0.08 to 0.2 and found little variation in performance.



The robot learns fairly quickly in all cases, but it seems that the most optimal values in the very first couple episodes are 0.08 and 0.1. Since 0.08 learns too slowly in the 35 – 90 trials, I decided to stick with 0.1 as the most optimal learning rate.

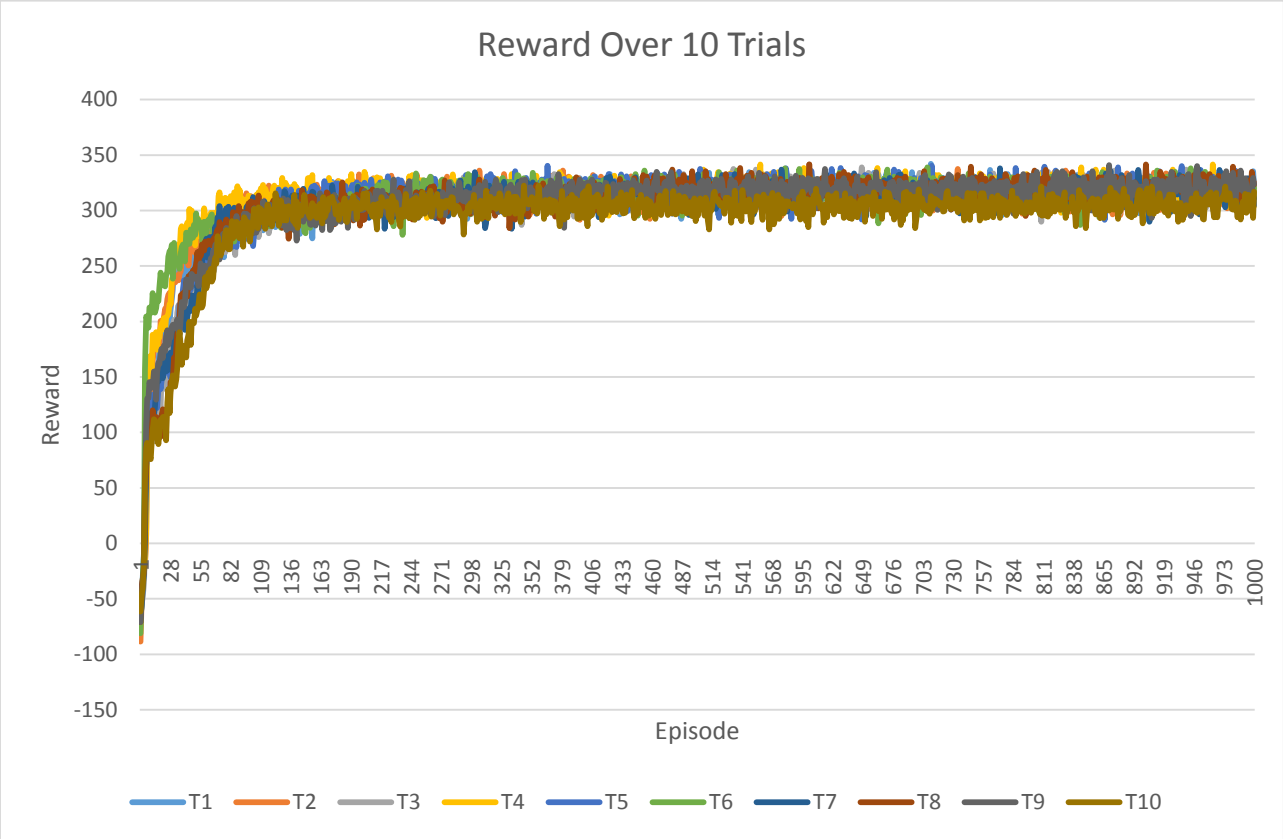
Therefore, the optimal values that I found are **Learning Rate = 0.1, Epsilon = 0.02.**

3) The graph below shows my results for the first trial. The robot finds the optimal route in about 200 episodes, and follows the route fairly consistently, improving only very slightly at the end of the 1000th trial.



Kevin Hou
CS 440
9/29/2014

The next chart shows the results of 10 trials and what the reward values were at the end of each episode.



While the end results were the same, we can see that in certain trials, the robot learns slightly faster or slower based on successful or unsuccessful exploration attempts. However, after about 170 episodes, all trials eventually end up averaging roughly the same reward.

The chart below shows the average reward collected across all 10 trials. While the robot does poorly for the first couple episodes, by the 6th or 7th episode, it is already starting to collect positive rewards, and this reward value increases extremely rapidly with each episode, topping off in the mid 300's. It takes the robot about 80-100 episodes before it finds a clear optimal route, and over the course of the next 900 episodes, its performance increases ever so slightly from the lower 300's to the mid 300's in terms of rewards received. With a low epsilon, the robot doesn't deviate too much from its most optimal route, and we don't see the wild fluctuations we saw in trials where the robot had epsilons of over 0.1. By balancing a lower epsilon and a higher learning rate, we minimize the amount of deviation we take from the optimal route while quickly learning from our few exploration attempts early on.

Kevin Hou
CS 440
9/29/2014