MP1 Part 2

Thomas Liu

2014-09-27 Sat

1 In WorldWithoutThief

(a) Set the discount factor to 0.9, the learning rate to 0.1, and the ϵ to 0.0 (which means your agent does not perform random actions except perhaps for ties). Simulate your agent for 1000 episodes of 10000 steps. Explain what you observe.

Setting ϵ to 0 caused the agent gain a total reward of 0 almost every episode past episode 1. This is because ϵ controls the rate of exploration; setting it to 0 would cause the Q Learner to be completely greedy, only acting only on the information it has already obtained rather than exploring different avenues for higher rewards. Therefore, in a world without thief, the agent learns how to avoid the negative reward of slipping and losing the packages in episode 1, but never tries to risk walking over the slippery spots to learn how to deliver the packages for a positive reward, since it is content with 0 reward.

(b) Set ϵ as 0.1, and keep all the other settings. What did you observe? Explain it.

This time, the results vastly differ. It takes longer for the Q Learner to avoid negative rewards; it is not until around episode 15 that the Q Learner figures out how to deftly navigate around all the slippery spots completely. Although this is bad for the Q Learner in the short time, by taking these risks, the Q Learner was able to figure out how to accrue positive rewards, resulting in massive reward gains later on. The 10% chance that it does not follow the policy allows it to discover things that may seem bad in the short term, but very rewarding in the long term. For example, there is a wall of slipperiness in between the robot's starting position and the package

destinations. If the robot did not occasionally try stepping on the slippery space (which, at first glance, would seem to produce negative utility), it would never find out that there were rewards on the other side. Setting the exploration rate to 0.1 allows this to eventually happen, meaning that the robot understands that while the immediate reward of stepping on the slippery spot is probably negative, the expected utility of stepping on that spot is much higher.

(c) Try larger ϵ and keep all other settings. What did you observe? Explain it.

In the first problem, the Q Learner was not doing enough exploration. Now, it's at the opposite end: exploring a bit too much. At $\epsilon = 0.5$, the agent is exploring half the time, so half the time it is not using what it learned. This causes large amounts of negative reward to accrue because of how far it is deviating from the policy.

2 In WorldWithThief

(a) Set $\epsilon = 0.05$ and keep all other settings. In the simulation command, set `knows_thief` as n. How is the performance of your agent? Explain it.

The agent is still able to learn a little bit, but since it is unaware of the thief, it cannot get an accurate grasp of the utility of taking certain actions in certain states. The thief throws a wrench into Q Learning since it throws off the mapping of states and actions to their corresponding utilities by randomly moving along the column and causing the agent to gain negative rewards seemingly out of nowhere. In fact, the policy that was generated seemed to send the robot into the corner to hide; the negative reward was probably from exploration.

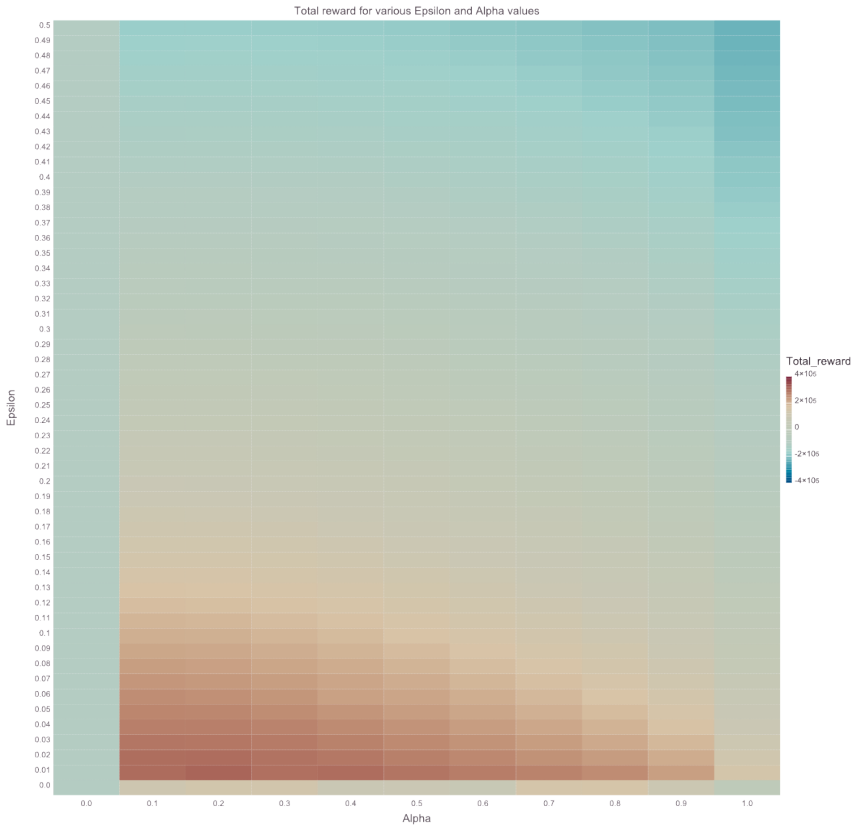
(b) Set `knows_thief` as y. Explain any performance difference.

This time, the agent is able to return to receiving large rewards after about 5 episodes. This is because it now has different states for different thief positions. By doing this, the agent is able to properly map states and actions to utility, removing the seeming randomness of the thief from before. It is able to time its actions carefully, adeptly dodging the patrolling thief. This

change makes a huge difference, since there are five times as many states as before, allowing the agent to evaluate the utility of states and actions much more effectively.

(c) Search for the best learning rate and ϵ

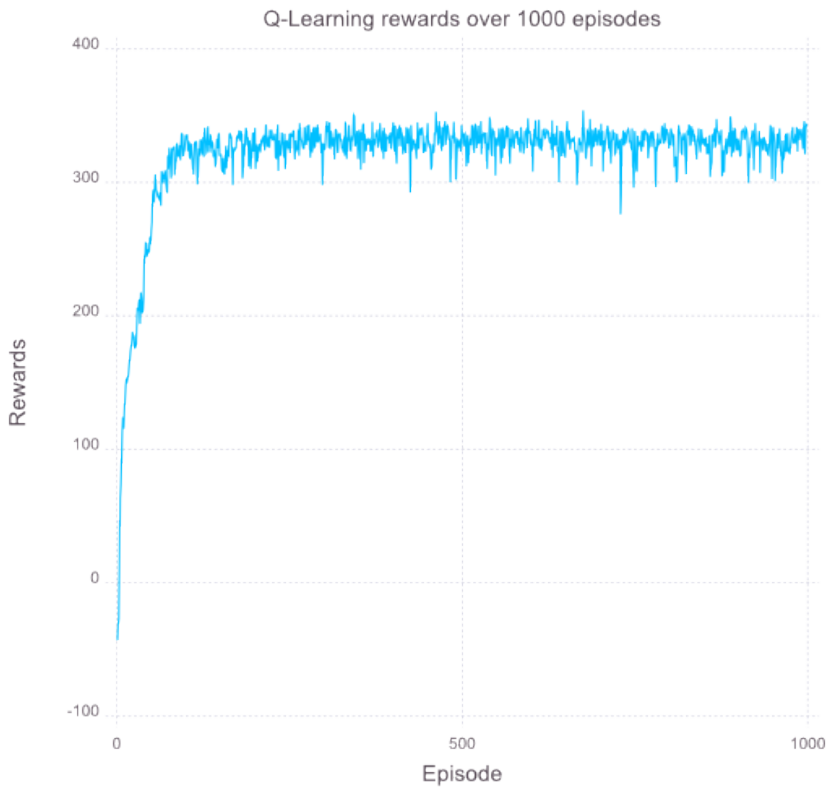
I wrote a script that compiled the learning agent with various values of epsilon and alpha, and plotted the total rewards (summing together the reward from all the 1000 episodes) garnered from each setting with a heatmap. I tested alpha values from 0.1 to 1.0 using increments of 0.1, and epsilon values from 0.0 to 0.5 using increments of 0.01.



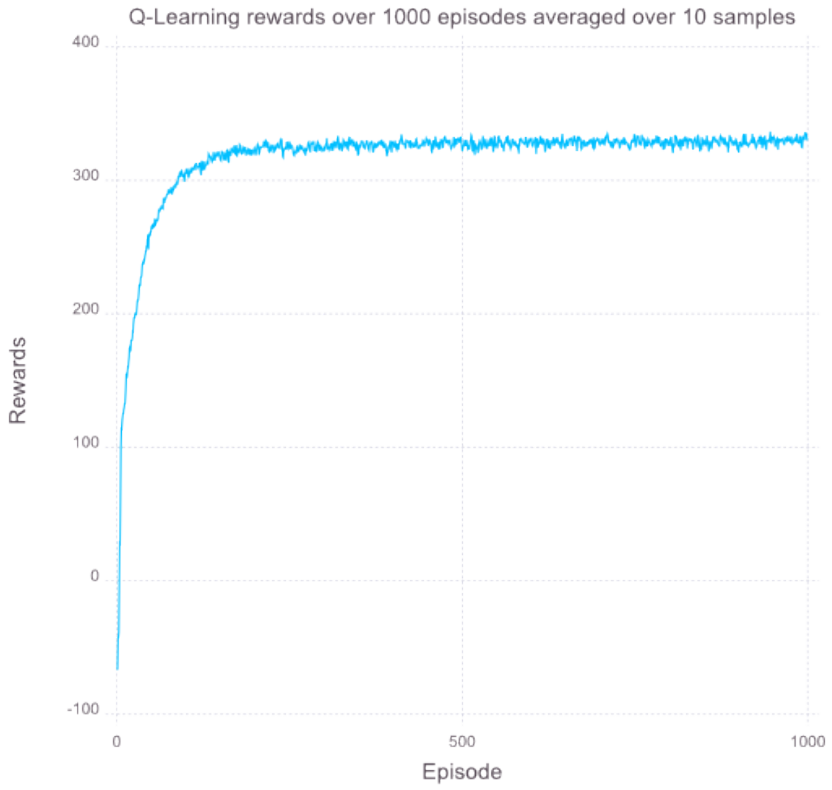
Not surprisingly, having epsilon values that are too high is detrimental to gaining rewards. But having an epsilon of 0 also severely damages reward gain. In terms of alpha, the learning rate, it looks like most values at least somewhat work, as long as epsilon is an okay value. However, just like with epsilon, the extremes, 0.0 and 1.0, don't work out very well. Looking at the

plot, it seems like the optimal value for alpha is 0.2, and the optimal epsilon value is 0.01.

3 Using your best parameters from 2c, simulate your agent for 1000 episodes of 10000 steps. Plot the expected discounted reward achieved at the end of each episode showing how the agent improves. What do you notice? Repeat the procedure nine more times and again plot the expected discounted reward at the end of each episode but now averaged over all ten simulations. Explain what you observe.



The plot shows that the agent learns very rapidly, quickly moving from negative to positive range, then flattening out slightly above 300, with quite a bit of random variation, but staying inside a pretty constant range.



Averaged over ten samples, the randomness of the previous plot has been lessened. It is now easy to tell that the rewards from each episode hovers around 330 after a while. It looks like the Q Learning agent approaches convergence on the optimal policy at some point close to the 200 episode mark.