QUESTION 1: Observe what you see with the agent's behavior as it takes random actions. Does the smartcab eventually make it to the destination? Are there any other interesting observations to note?

ANSWER 1: When the agent is taking random actions, I notice that the rewards are mostly negative. It looks like the agent runs a lot of red lights, cuts other drivers off, ect. While I was watching the agent never arrived at a target destination, but taking random actions could theoretically get the agent there. There would just be a lot of luck involved. I also notice that when the agent goes off the side of the map, it appears on the opposite side - so this grid is actually a flattened sphere.

QUESTION 2: What states have you identified that are appropriate for modeling the smartcab and environment? Why do you believe each of these states to be appropriate for this problem?

ANSWER 2: I've defined the state as a list: the light color, if there is any traffic surrounding the car, which direction the planner is directing the agent, and a binned representation of the deadline. The the deadline bins should be 0-3: state 1, 4-8: state 2, 9-16: state 3, 17+: state 4.

My logic for including all of these components in state are:

- Light color - important for yielding
- oncoming front - also important for yielding
- oncoming left - yielding again
- oncoming right - yielding
- planner direction - important for getting to (or avoiding) the target
- deadline bins - provide tiers for how rushed the agent is

OPTIONAL: How many states in total exist for the smartcab in this environment? Does this number seem reasonable given that the goal of Q-Learning is to learn and make informed decisions about each state? Why or why not?

OPTIONAL ANSWER: There are 192 unique states possible given my definition of state. I feel like this is probably on the high end of what's a reasonable number of possible states for this relatively simple simulation, but I'm going to go for it and see how it goes. My hope is that the more complex representation of the world will pay off in the end, despite probably taking a longer time for the agent to learn.

QUESTION 3: What changes do you notice in the agent's behavior when compared to the basic driving agent when random actions were always taken? Why is this behavior occurring?

ANSWER 3: The agent does start to seek out the target destinations. It also stops running red lights, and cutting off other drivers. This is occurring because that each action and reward are now being used to update the agent's policy concerning that state, so when the agent finds itself in the same or similar state again it can use the updated policies to pick the next action, rather than a random choice. The overall effect is that there's a lot less negative rewards getting incurred constantly, and the smartcab does start seeking out the destination.

QUESTION 4: Report the different values for the parameters tuned in your basic implementation of Q-Learning. For which set of parameters does the agent perform best? How well does the final driving agent perform?

ANSWER 4: I tried setting epsilon to a couple different values between 0.001 and 0.1. 0.015 ended up working the best for optimizing my total destinations reached.

For my assumed Q estimation for unknown states, I tried -1, 0, 1, 5, and 10. I definitely had the most success with 1 - some of the more extreme values had strange consequencies, like my smartcab not moving at all.

For gamma, I actually found that setting it extremely low resulted in a better Average Destinations Reached value, but at the expense of Average Rewards per Turn. So while gamma=0.001 was great for getting the cab to the destination, I think the cab was driving terrible/inefficiently to get there. I ended up with a value of gamma=0.1 as a middle ground.

For alpha, I tried a pretty wide range of values again, from 0.0001 to 0.9999. I ended up going for 0.28, for the reason that somewhere between 0.25 and 0.3 I tended to get perfect destination totals for the second half of each experiment.

QUESTION 5: Does your agent get close to finding an optimal policy, i.e. reach the destination in the minimum possible time, and not incur any penalties? How would you describe an optimal policy for this problem?

ANSWER 5: Running the experiment 100 consecutive times and averaging together the trials yields the following graph.

My agent is approaching 1.0 for its average destinations reached, but it doesn't seem to improve much on average rewards per turn after about 20 trials. So I think I've currently got a reasonably dependable agent but which isn't driving safe or efficient… big problems once outside of a simulation.

An optimal policy for this problem would have a perfect destination count of 1 at the end of the trials, and it would never incur any negative rewards. My learner most certainly isn't there yet. I think having epsilon and the learning rate scale down as the trials progress, as mentioned in the lectures, would be a big step in the right direction. I'm also wondering if there's some way to forecast further than one expected state ahead, and if that could improve the learner in this exercise.

I also may need to remove any awareness of deadlines from my agent's state - 192 different states is probably too many for such a small number of trials. I currently have my choose_actions function set up with a crude way of picking a similar stage to use as a basis for newly encountered stages, but I probably need to evaulate whether setting that up was a good move.

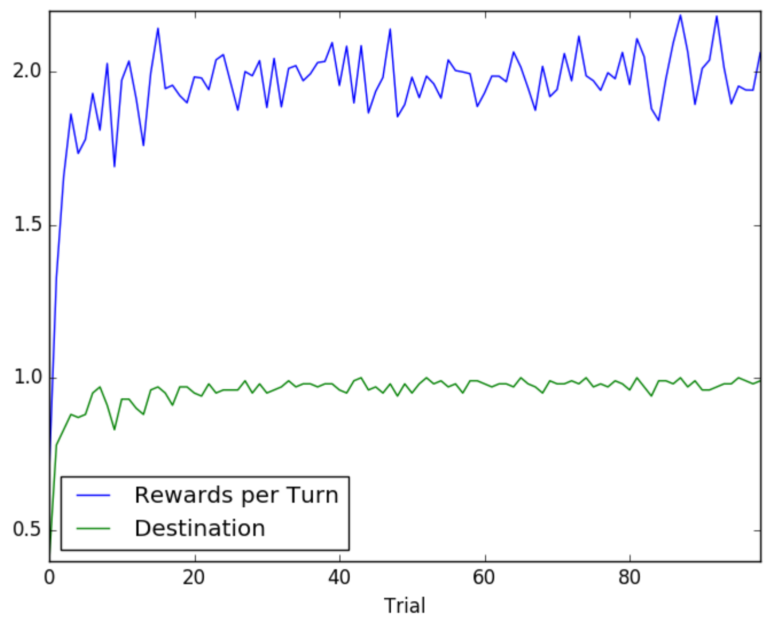

Figure 1: Summary Plot

All in all, though, this project was really rewarding! This is the first time I've coded any sort of AI, and I was really thrilled about how this first attempt went and seeing my agent moving around figuring things out!