Smart Cab Project

QUESTION: 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**: As the agent takes random behaviors, the agent eventually reaches the destination, but the time it takes to get there is very random. When the deadline is enforced, it reaches the destination about 25% of the time.

QUESTION: 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**: Some states I have identified as the location of the destination vs. the smartcab, the left, right and oncoming sides of traffic, and the status of the lights. I believe these states are appropriate for this problem because at it needs to know which directions to head based on the x/y distance of the destination, what best to do when there is oncoming traffic, and what to do on a green vs. right light. These are the actions normal cab drivers take.

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?

> **ANSWER**: There are 6*8*2 = 96 location/destination states, 2*2*2 = 8 traffic states, and 2 light states. Therefore, there are a total of 96*8*2 = 1536 states. Since we are training the smartcab over 100 trials of at most 25 moves, we don’t want too many states such that the model will never train over, nor too few such that it doesn’t capture the complexity of the environment. We will have be able to reach a max of 25*100 = 2500 states, which should be enough to reach most of the states at least once.

QUESTION: 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:** Some changes are that the smartcab is learning to not take actions that resulted in a negative reward and taking actions that resulted in positive rewards when reaching an old state. An example is this:

old_state: {‘s': -0.20000000000, 'e': 2.862364791808, 'w': 0, 'n': 0}
heading: e
reward: 2.0
new_state: {‘s': -0.20000000000, 'e': 3.1471391292825599, 'w': 0, 'n': 0}

Indeed, based on the reward from reaching the new state, the model updates the q-value on the ‘e’ action from the old state.

QUESTION: 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: I tried alpha, gamma, and epsilon values in the range of [0.0, 0.95], [0.0, 0.95], [0.0, .05] respectively. I ran through each combination of the ranges in 0.05 increments. With alpha, gamma, epsilon values of 0.1, 0.2, 0.0, I was consistently getting around 70% success rate. What seemed to work best (consistently above 0.75%) was a gamma, alpha, and epsilon combination of 0.4, 0.45, and 0.0, resulting in a success rate of .84.

QUESTION: 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: Yes, I would say that my smartcab reaches the optimal policy. I would describe the optimal policy for the problem as the highest success rate with the least amount of time used and the least amount of traffic laws broken. One time over 100 trials, I had an average success rate of .84 and an average percent time used of around .2, almost always had positive reward at the end.