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 sometimes reaches the destination, but the time it takes to get there is very random. When the deadline is enforced, it only 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 next waypoint of 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 next direction the planner decides, what best to do when there is oncoming traffic, and what to do a green vs. right light. These are the actions normal cab drivers take. I have excluded deadline as I don't want the smartcab to take "riskier" moves as the deadline approaches.

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 3 next waypoint values, 4 traffic states, and 2 light states. Therefore, there are a total of 24 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 is more than 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: {'forward': 2.5843722783431673, 'right': 0, None: 0, 'left': 0}
action: forward
reward: 2.0
new_state: {'forward': 3.6225601593751238, 'right': 0, None: 0, 'left': 0}
```

Indeed, based on the reward from reaching the new state, the model updates the q-value on the 'forward' 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, 0.9], [0.0, 0.9], [0.0, .05] respectively. I ran through each combination of the ranges in 0.1 increments (0.01 for epsilon). With alpha, gamma, epsilon values of 0.1, 0.1, 0.0, I was consistently getting an average of 96% success rate. Below is a heat map of average success rates per gamma and alpha combination (5 epsilon state results were rounded). What seemed to work best (consistently around 99-100%) was a gamma and alpha of 0.1 and 0.7 respectively.

		Gamma									
		0.0	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
Alpha	**0.0**	0.21	0.22	0.19	0.2	0.21	0.2	0.22	0.20	0.20	0.22
	0.1	0.98	0.98	0.98	0.988	0.98	0.97	0.96	0.96	0.94	0.95
	0.2	0.98	0.97	0.99	0.95	0.96	0.95	0.95	0.82	0.90	0.82
	0.3	0.98	0.97	0.98	0.991	0.97	0.88	0.95	0.82	0.84	0.82
	0.4	0.97	0.98	0.98	0.98	0.98	0.93	0.89	0.93	0.84	0.88
	0.5	0.97	0.99	0.98	0.98	0.98	0.96	0.91	0.91	0.95	0.88
	0.6	0.98	0.98	0.99	0.97	0.98	0.97	0.97	0.94	0.93	0.94
	0.7	0.98	**0.995**	0.97	0.97	0.98	0.97	0.97	0.86	0.91	0.89
	0.8	0.993	0.99	0.98	0.96	0.97	0.96	0.96	0.96	0.92	0.93
	0.9	0.97	0.97	0.97	0.98	0.96	0.96	0.98	0.96	0.94	0.94

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: I would say that my agent finds the optimal policy for the most part. While it learns all the traffic rules correctly (not doing anything would cause a collision), it at times gets stuck in a local minima and waits at a stop light for multiple turns. Toward the end it will consistently reach the destination, although it will take a few rounds to do some close-to-random exploration.