Implement a Basic Driving Agent

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: The agent does eventually make it to the destination, after a very long time. In my first simulation it took 89 turns AFTER the deadline to reach the destination.

I'm not sure these are actually interesting observations, but the agent is an awful driver when random. It regularly violates the rules of the road, and has no idea where it is going at any given time. These are to be expected with completely random actions.

Inform the Driving Agent

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: Our smartcab will need the following state information

- Light: red/green – Needed to learn if it is safe to go or not
- Waypoint: forward/left/right – Needed to learn the direction to go
- Oncoming: none/forward/left/right – Needed to learn to avoid accidents
- Left: none/forward/left/right – Needed to learn to avoid accidents
- Right: none/forward/left/right – Needed to learn to avoid accidents if we are in a country where people drive on the left side of the road

Note: "deadline" is being omitted, as it provides no valuable information for our agent. Since we want it to drive as quickly as possible, regardless of the deadline, the deadline shouldn't change its behavior.

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 384 (aka 2 lights x 3 waypoints x 4 oncoming x 4 left x 4 right) possible states that exist for the smartcab in this environment. This seems like a reasonable number, given the Q-Learning Goal. That is, we can probably use a simple table to store this data.

Source: https://en.wikipedia.org/wiki/Q-learning#Implementation

Implement a Q-Learning Driving Agent

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: Compared to the random agent, the Q-Learning agent learned very quickly how to get to the destination within the deadline. This behavior is happening because it very frequently sees common states, such as intersections with no other traffic, and its Q_table gains positive values for driving in the correct direction.

The learning agent is also pretty bad at figuring out what to do when there are other cars at the intersection. This is the result of states involving other cars being relatively rare – giving few instances to learn from – and also the state space involving other cars is unfortunately larger than I'd like. I am pretty sure it would eventually stop getting confused and/or causing accidents if it ran enough simulations, but it will take significantly more than learning to drive to the destination.

Improve the Q-Learning Driving Agent

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: There are three parameters I'm working with here.

Alpha: The learning rate.

Gamma: The discount on rewards for *a'* in *s'*.

Epsilon: The exploration rate. I set this to 0% for safety reasons. I don't want the agent to re-try a previous dangerous action very often, once it's learned that it is a dangerous action. Otherwise, it might randomly try hitting cars to see if that's really a bad idea.

For Alpha and Gamma, I experimented with the following values before settling on Alpha 0.9, Gamma 0.1:

Alpha (Gamma@0.1)	Average Performance		Gamma (Alpha@0.9)	Average Performance
0.1	74.6		0.1	78.6
0.2	72		0.2	75.4
0.3	78		0.3	73
0.4	78.4		0.4	74
0.5	75.6		0.5	73.8
0.6	76.2		0.6	69
0.7	78.4		0.7	74
0.8	75.2		0.8	69.8
0.9	78.6		0.9	75.2
1	76		1	72.6

'Performance' here is defined as how frequently our smartcab managed to reach its destination, without incurring any penalties, by the deadline, out of 100 simulations. This is averaged over 5 runs. By including all simulations, and not just the later ones, we can get a rough estimate of how quickly the algorithm is learning, in addition to how well it is learning.

As we can see in this table, the different values of Alpha generally did not create huge swings in our results (except 0.2, which was likely an outlier). By contrast, increasing Gamma above 0.1 did create a noticeable drop in performance. This makes sense, as high-gamma would imply that the Q algorithm is valuing subsequent actions very highly, and that is likely leading to our agent taking more frequent risky current actions on the hopes of getting higher payoffs on the next action. This is not particularly smart given the reward structure we're working with.

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: My agent looks like it is very close to the optimal policy. In the later trials, it is almost always good at reaching the destination in the minimum possible time, with few penalties. Unfortunately, it still occasionally has penalties when there are other cars in the intersection. I believe this is just because the state space for intersections is both large and infrequently seen. Reducing the state space, or running more trials, could both help with this issue. Alternatively, adding additional cars to the simulation could also help on this front.

I would describe the optimal policy as such:

If 'light' is red, and 'waypoint' is not 'right', take no action.
If 'light' is red, and 'waypoint' is 'right', and 'left' is not 'forward', move right
If 'light' is green, and 'waypoint' is 'left', and 'oncoming' is 'forward' or 'right', take no action
Else if 'light' is green, move to waypoint

This is assuming US driving laws, where right-on-red is legal.