Implement a Basic Driving Agent

Q: 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?

A: As the question states, when the agent takes random action from the four options [None,'forward','left','right'], the results are... random.

If the agent reaches the destination before the deadline, it is by complete accident. Ususally, the time runs out before our wandering "smart" cab can stumble upon its destination.

Inform the Driving Agent

Q: 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?

A: The following inputs have been deemed appropriate for use in the smartcab's self.state variable:

input['light']: ('red','green') #Simplified traffic lights, red means stop, green means go

inputs['oncoming']: (None,'left','right','forward') #If None, no oncoming vehicle, otherwise, state describes the move being made by said vehicle

inputs['left']: (None,'left','right','forward') #If None, no vehicle to the left, otherwise, state describes the move being made by said vehicle to the left

self.next_waypoint: ('forward','left','right') #If the smartcab had GPS navigation, this variable represents turn-by-turn directions

I believe that all these variables are appropriate for depicting our smartcab's state, as it gives a fairly complete depiction of the different environmental states the cab will encounter on the road

I did not, however, include the **deadline** variable in my states, both because it would multiply the number of potential states that the smartcab would encounter, and because the smartcab is egocentric, and a bit myopic.

Deadline is not a meaningful measure if the cab is not aware of current location relative to the goal. Without knowing how far away the goal is, the deadline is just another number ticking down at each turn. Though, the deadline variable is useful for the program, at large, because it dictates the doling out of the bonus +10 reward for reaching the goal before deadline reaches 0.

But, assuming that the cab learns how to follow the proper waypoint, it should reach the goal in a reasonable amount of time. That, and the reward for reaching the goal in time is a fixed

(boolean) value, so as long as the cab makes it “on time”, it does not matter how far into the deadline the trip goes.

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?

A: The total number of possible states is 2*4*4*3=96

For a small number of trials (e.g. n=100), our program may not be able to explore every single combination of state and action, but as the number of trials grows, this issue becomes less relevant.

Though, for this project, it seems as though traffic in the smartcab's world is fairly sparse, so it is most likely that the cab will enounter 6 states most often: light*waypoint -> (red,green)*(forward,left,right)

Q: 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?

A: When a very basic Q-learning equation is implemented (alpha = 0.5, gamma = 0.0), and action is chosen based on highest reward from prior exploration, the smartcab does learn a policy in which it obeys traffic signals, shares the road, and copies the "GPS" navigation, often reaching the destination in proper time.

Implement a Q-Learning Driving Agent

Q: 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?

A: As compared to the random wanderer, the Q-learning smartcab works in a much more logical, calculated fashion. Though, in the first few steps that the cab takes, it behaves much like its primitive, random predecessor, because it has not received any feedback for taking a given action in a given state. When the cab gathers reward information for state-action pairs, it writes down these values to its memory, and refers back to them when it re-encounters a previously visited state. With the way that the environment is set up, there is only one ‘best’ action for a given state, and with sufficient exploration, the cab will learn reward values for every action taken in every state.

Improve the Q-Learning Driving Agent

Q: 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?

A: The parameter of importance to the smartcab is alpha, the learning rate. An extreme example that produces noteworthy results is when alpha is set to 0. In this scenario, the smartcab does not interpret any rewards, and reverts to virtually random actions.

Another noteworthy example is when alpha is set to 1, meaning that the smartcab only interprets the most recent rewards given. Even though the cab is essentially "stuck" in its present state, it does manage to drive to the goal within the allotted time.

When Gamma is set to 1 in addition to Alpha, the cab loses its accuracy, as the back-assigned rewards compound at an alarming rate (to the order of 10 E+138), which makes for poor decision-making by the cab's internal policy.

It seems as though the cab performs best with intermediate values of alpha and gamma (e.g. alpha=0.5, gamma=0.25), almost always reaching the goal in the later trials, and very rarely encountering a penalty instead of a reward (usually because the state is new to the cab).

Q: 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?

A: It does appear that our smartcab approaches an optimal policy in the later trials (one where it reaches the goal in time, and does not make any penalized moves), though it is not perfect, since some states may not be encountered until late, which gives rise to the possibility of making a mistake. In this problem, the optimal policy seems to be that in which the car follows the waypoint "navigation", obeys traffic signals, and follows the proper right-of-way rules when other cars are present.