Train a Smartcab to Drive

This report is addressing the questions associated with an Udacity's project, in which we formulate reinforcement learning algorithms to train an agent (Smartcab) to drive around while learning the fastest way to reach its destination with minimum violation of traffic rules. First, we will implement a basic driving agent which randomly drives around. Then, we will inform the agent about the rules to update states. Next, we implement Q-learning. Finally, we optimize the behavior of the agent to reach the destination. Detailed answers to each question are *provided in blue*.

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?*

The agent which takes a random action from the set of actions (None, 'forward', 'left', 'right') at each intersection eventually makes it to the destination. It does not sense the environment (traffic light or car directions at the intersection). However, the random agent's approach to reach the destination is poor in performance. It sometimes takes no action even when it is not necessary (no oncoming traffic or red light). It sometimes hits the other cars or disobeys the traffic signals. The random agent is always trying new action without any knowledge of the past rewarding (or punishing) actions. Most of the time, it does not reach its destination in time. The mean successful rate for 3 tests is 0.23 with a standard deviation of 0.036.

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?*

The states should include information about the intersection (environment) that is necessary for choosing the correct action.

1. *The state of the traffic light ('red' or 'green')*
2. *The direction of the car in the oncoming lane (None, 'forward', 'left', 'right')*
3. *The direction of the car in the right lane (None, 'forward', 'left', 'right')*
4. *The direction of the car in the left lane (None, 'forward', 'left', 'right')*
5. *The direction suggested by the route planner-next_waypoint ('forward', 'left', 'right').*

In order to enhance the performance of the agent by performing legal moves of the agent, traffic light and the directions of the car in the oncoming lane and left lane are very important. The penalty will be imposed in the following conditions: (1) the next_waypoint is 'forward' or 'left' when the traffic light is 'red', (2) the next_waypoint is 'right' when the traffic light is 'red' and the direction of the car in the left lane is 'forward', (3) the next_waypoint is 'left' when the traffic light is 'green' and the direction of the car in the oncoming lane is 'forward'. Here the direction of the car in the right lane does not matter because of the USA right of way rules.

Although the deadline seems to affect the successful rate (whether the agent arrives at the destination within deadline), it is not included in the model for the following reasons. If we were to include the deadline into our current state, our state space would blow up, we would suffer from the curse of dimensionality and it would take a long time for the Q matrix to converge. Including the deadline could possibly influence the agent in making illegal moves when the deadline is near.

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?*

*If there are no rules, there are 2*4*4*4*3=768 states. However, some states never matter. In Q-Learning, we do not need to create the entire matrix, instead, we can fill the values iteratively as the agent drives around. The total number of states in this environment is 3 (next_waypoint) * 2 (traffic light) * 4*

*(oncoming) * 4 (oncoming) = 96.*

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?*

After a few iterations the agent started to obey the traffic signals and follow the directions provided by the route planner (decrease the distance between the destination and current position). Since the agent takes the learned Q-values into account during its exploration, it tends to stop at red light instead of other option such as right turns. It is because the intersection information is not included in the state modelling. Stopping at red never goes wrong.

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?*

With an initial Q-value of 100, I tuned the following parameters and calculated the successful rate (the probability that the agent arrives at the destination in time). Notice that I did not concern total penalties incurred, which would be a useful metric to characterize how well it's performing with regards to the agent following traffic rules. This can be implement like this:

```
If reward <0:
        self.penalty+=1
```

Learning rate (alpha)	Discount factor (gamma)	Exploration rate (epsilon)	Successful rate 1	Successful rate 2	Successful rate 3	Successful rate mean	Successful rate std
0.1	0.8	0.3	0.21	0.21	0.27	0.23	0.035
0.5	0.8	0.3	0.62	0.57	0.62	0.60	0.029
0.9	0.8	0.3	0.71	0.70	0.75	0.72	0.026
0.9	0.8	0.9	0.60	0.68	0.69	0.66	0.049
0.9	0.8	0.6	0.68	0.70	0.75	0.71	0.036
0.9	0.8	0.1	0.70	0.69	0.74	0.71	0.026
0.9	0.5	0.1	0.91	0.84	0.86	0.87	0.036
0.9	0.1	0.1	0.97	0.98	0.97	0.97	0.0058

The agent performs the best in terms of successful rate when we have a high learning rate (alpha = 0.9), low discount factor (gamma = 0.1), and low exploration rate (epsilon = 0.1). The final driving agent has a successful rate of 0.97 (standard deviation = 0.0058).

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?*

I think the optimal policy should be reaching the destination with minimum possible time and an optimal positive reward. My agent gets close to finding an optimal policy. Although my agent does not incur penalties, the time span can be further minimized if we consider intersection information (car directions of oncoming and left lanes). This is more obvious when number of dummy cars increase so that my agent has more opportunities to learn.