Smartcab Project Report

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?*

Ans: The code I use for agent to take random action is:

```
action = random.choice([None, 'forward', 'left', 'right'])
```

With this code, the cab randomly chose its direction to move. Sometimes it hit to the destination, but most of the time it failed within the given deadline. The first interesting thing I notice is that the cab will stop to wait when the light is green no matter what action it supposed to have. After checking the code in environment.py, I found that in act() function, the cab will be enforced to obey the traffic rule no matter what action you give it. The second one is about the planner.py, it gives the guide action to the cab. After I set action = self. next_waypoint, the cab will reach the destination with almost 100% rate, very good planner!

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?*

Ans: At first, the states I identified is location and heading. I also realized that the start point and destination changed every trial, so, in my first perception, I reset my Q table every tiral. After finishing the code for Q_learning, I found the succeeded trial rate was awful, only 24% out of 100 trials. After re-checking the whole case situation, I found that, without knowing the guided next_waypoint and destination, my cab is like a silly robot with randomly moving in the map and trying to make a good decision out of 8*6*4 (grid col* grid row* heading) states within just 20 to 40 steps . And everytime it ran out of deadline, the Q table are reset (like losing memory).

After struggling for a while, I realized that the state that the cab stay should include the guided next_waypoint from planner since it is reasonable for a cab to know the destination and know how to plan its movement. The next thing the cab should know is the traffic light and if there are other cars in its location. After rechecking the traffic rule, I found the information of other car on its right never matter the action, then, the final state I chose is this:

```
    self.state = {  "light":inputs["light"],
                     "oncoming":inputs["oncoming"],
                      "left": inputs["left"],
                      "direction": self.next_waypoint}
```

And the Q_table should not reset for every trial.

I did not chose ‘deadline’ as a variable is because if I include it, there will be 20 or 30 folds more states which is too much. And in return, the deadline itself contain less information to ask my cab to turn left, going forward or turn right.

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?*

Ans: For the state feature I chose, there are 2 color of light (green and red), 4 possibility of oncoming cars (none, forward, right, left), 4 possibility of left cars, and 4 direction of next_waypoint. In total, there are 2*4*4*4=128 states. Considering, there are 4 possible actions for the cab, there are 128*4= 512 state/action pairs.

At first glance, I thought it was not reasonable given the total trial is only 100. Then I realized that, most of the above states will hardly shows up since it is very hard for my cab to meet the other cab, the most likely states are 2 color of light, none coming cars, none left cars, 3 next_waypoint (None will only be set when it reach to destination). With 4 possible actions, the most appearance states/action pairs number is only 24. In this way, it is reasonable for Q-learning in this scenario.

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?*

Ans: Compared to the basic driving agent, my cab's movement will quickly follow the guidance of next_waypoint after several steps in the first trial or first and two trials. And eventually hit the destination within deadline.

Considering my state is about light, oncoming, left and direction. Each time it chose the action= direction, it will get reward of 2 or 0 (depends on the light, green 2, red,0, and sometimes the other cars matters). If it chose the action != direction, it will get reward of -0.5 or 0 (depends on the light, green -0.5, red 0, and sometimes the other cars matters). In this way, when light is green, it will follow the instruction from planner, when light is red, it is better to stop there or it will get panelty. In this way, with a good planner, my cab will successfully reach its destination. Furthermore, it also obeys the traffic rules.

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?*

Ans: The parameters I tuned are alpha(learning rate) and gamma(discount rate). The performance are showed in the following table:

(performance: the rate of successfully reached destination of 100 trials.)

(row: alpha, col: gamma)

	0.2	0.5	0.8
0.2	99%	98%	99%
0.5	98%	100%	98%
0.8	99%	98%	99%

With alpha = 0.5 and gamma = 0.5, my cab can 100% reach the destination within dealine.

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?*

Ans: My agent did get close to finding an optimal policy base on the succeeded trail rate(98% to 100%) and for each trial it can get reward of 16 to 34 (depends on how many steps it needed to reach destination). The optimal policy for this problem is try to follow the guidance from the planner, but at the same time, it should also obey the traffic rules. If the traffic light is red, it will stop or turn on right if the planner told it to. If the traffic light is green, it will follow the planner’s suggestion expect when the planner asked to left turning. In left turning situation, it will try to find what the situation of the oncoming lane is. If there is a car in the oncoming lane turning right, it will stop, otherwise, it will left turn.