Assignment 4 – Smart Cab

Implement a Basic Driving Agent

*In your report, mention what you see in the agent's behavior. Does it eventually make it to the target location?**

The agent always makes a random move. The agent eventually reaches its destination—Although the length of time for the agent to reach its destination varies widely.

*Please set self.decrement_epsilon to False for the code to reflect this behavior.

Identify and Update State

Justify why you picked these set of states, and how they model the agent and its environment.

I first looked at what are the possible actions that the agent can make and decided that what states the agent needs to be in to be able to perform those actions.

The information needed for these states was the state of the traffic light (Red or Green), whether there was oncoming traffic, and whether the traffic on the left was going forward. I combined this information with the next_waypoint variable to arrive at all of the states I chose. From any of the states not including the starting state, the agent is able to make an action and arrive at the other states.

It is no use that the agent knows what the car on the right is doing; therefore, I did not include that as part of the state. It is important for the state to contain information in the direction that it's supposed to go in order for it to learn from the directions it has taken in the past under the same conditions (eg took a left on red with no traffic around). The deadline was disregarded because it would create too many states, and would take more trials for the agent to land in the same space again. The idea is to reduce the number of states as much as possible while maintaining maximum information. If there are too few features, the agent will not be able to properly assess special cases. For example, if we only kept track of the light and its direction, it would never know why it is being penalized if it makes a left on green with oncoming traffic.

Implement Q Learning

*What changes do you notice in the agent's behavior? **

After implementing Q-Learning and removing randomness, the agent figures out that if it stays in the same spot, it will keep getting positive rewards, so it will never move.

*Please set self.decrement_epsilon to False and self.epsilon to 0 for the code to reflect this behavior.

Enhance the Driving Agent

*Report what changes you made to your basic implementation of Q-Learning to achieve the final version of the agent. How well does it perform? **

As per the Q-Learning videos, I implemented a learning rate (alpha), and a discount rate (gamma). I tested different values for alpha (0.9, 0.8, 0.5) and different values for gamma (0.05,0.1, 0.3,0.5). Based on those values, it seems that a an alpha of 0.8 and gamma of 0.1 worked better in general when taking into account mean and media deadline scores.

I also implemented a randomness variable (epsilon), which decreases over time and eventually becomes zero in which case the algorithm always picks the best action instead of sometimes picking a random action. This is done to encourage exploration at the beginning and try different actions from a given state. If it takes the wrong action, it will eventually not try those out anymore. However, without some randomness, the agent may make an "okay" action and keep picking it because it has yet to explore the best action. With enough randomness, it will try all the actions out. However, the randomness needs to decrease over time or else the agent would not really be learning and applying what it learned.

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?

On the first few trials, the agent rarely reaches its destination before the deadline. This is partly explained because of the randomness involved. As the randomness decreases over the trials, and the agent learns, the agent improves its performance, and the Q-learning reaches the expected Q table ie go right if you should go right, go left if you should go left, etc. It is also interesting to note that, for example, if you could go forward, you should technically still be able to perform a right turn or yield and not cause an accident or break the law. As such, the other actions from this state still show positive values.

* Please set self.decrement_epsilon to True and self.epsilon to 100 for the code to reflect this behavior.