MLND - P4 - Vishakh Rayapeta

1) 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?

Using a random action gets the agent to the destination eventually. In most cases, the agent reaches the target location after the deadline value has expired. The agent's performance over consecutive trips is random - in some cases the target location is reached early and in other cases much later. The agent does not learn to avoid traffic violations (negative rewards).

2) Identify and update state

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

The Q-state selected comprises of {next_waypoint, light, oncoming, left}.

The first variable 'next_waypoint' must be learnt by the agent to successfully reach the assigned destination by the planner.

The next three variables {light, oncoming, left} must also be learnt to obey traffic rules. The traffic on the right does not impact any traffic rule and can be disregarded.

*I am using one binary bit (0 or 1) to represent the state of the light (red or green). Two binary bits (00, 01, 10, 11) are used to represent the state of next_waypoint, oncoming & left (None, right, left, forward). This translates to a total of (2 * 4 * 4 * 4 = 128) states in the Q-matrix .*

3) Implement Q-learning

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

The Q-matrix elements are each initialized to a very large positive value (10). This encourages the learning agent to explore as many states as possible at least once. The first time a state is explored the assigned reward is used to overwrite the initial value. Subsequent rewards are derated by an 'alpha value' and added to the existing reward.

The initial trial(s) provide the most learning opportunity for the agent. During the initial trials the agent explores many states and begins to learn to follow the planner and the traffic rules.

After implementing Q-learning, the agent manages to reach the target location before the deadline value has expired. The agent's performance over consecutive trips shows improvement that can be measured both in terms of the ability to avoid traffic violations (negative rewards) and to reach the destination before the deadline.

NOTE: Since the simulator uses randomization in generating the environment, training curve of the agent will vary a little bit between consecutive runs.

4) 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?

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?

The reward implementation provides a significantly large reward (12) if the agent reaches the destination before the expiration of the deadline. This reward is problematic because it has many random elements to it (traffic profile, distance of the destination from the starting point). Since this reward will randomly skew the Q-matrix, I do not use it in the Q-matrix updates.

I selected an alpha value of 0.4 to encourage a gradual learning rate. Did not experiment much with trying to optimize alpha since the results seem to be good.

The agent seems to be able to reach the destination within the time provided at least 95% of the time. The rate of penalty incurrence is reduced in later trials and is an indication of the success of Q-learning.