Udacity Reinforcement Learning Project

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

1. The agent sometimes reaches its destination.
2. There had been instances where the agent hit the hard time limit of -100. Thus, acting randomly is not the way to go.
3. Agent receives a reward of -1.0 when it violates traffic rules, 2.0 when it completes an action without violating any rule, 0.0 when no action is taken. One thing to add is to impose a negative reward to agent as time elapses. This will encourage agent to reach the destination sooner, rather than loitering around and picking up a reward of 2.0 for every action completion.

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

1. Minimal state representation: agent's next_waypoint, traffic light's state
2. Don't include anything that can be computed based on the the available states
    a. Agent's next_waypoint: because agent needs to know the next positions it can go to in order to successfully reach the destination.
    b. Traffic light's state: because agent needs to know the state of the traffic light to learn that it should stop when the light is red and proceed when the light is green.
    c. Left: This is important because of the US right of way traffic.
    d. Oncoming: This can help the agent learn to not make a left turn if there's oncoming traffic even when the light is green.
3. Deadline should not be included because if the deadline is 50 timesteps, then the size is 50^(6*8) to fully represent the state spaces.
4. I did not include 'right' because of the US right of way traffic.

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?

1. Compared to the random agent, the Q-Learning agent tries to maximize its expected utility by selecting the best possible action from its current state. Thus, its net_reward is larger and it reaches the destination within the alloted time way more frequently.

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?

Experiment 1:

alpha: 0.8
gamma: 0.2
epsilon: 0.05
abortion rate: 0.07
success rate: 0.93

Experiment 2:

alpha: 0.2

gamma: 0.8

epsilon: 0.05

abortion rate: 0.17

success rate: 0.83

Experiment 3:

alpha: 1

gamma: 1

epsilon: 0.05

abortion rate: 0.85

success rate: 0.15

Experiment 4:

alpha: 0.2

gamma: 0.2

epsilon: 0.05

abortion rate: 0.03

success rate: 0.97

Experiment 5:

alpha: 0.1

gamma: 0.1

epsilon: 0.05

abortion rate: 0.00

success rate: 1.00

1. From the qvalue update equation, which is

   $$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + (\alpha)\left[r + \gamma \max_{a'} Q(s',a')\right]$$

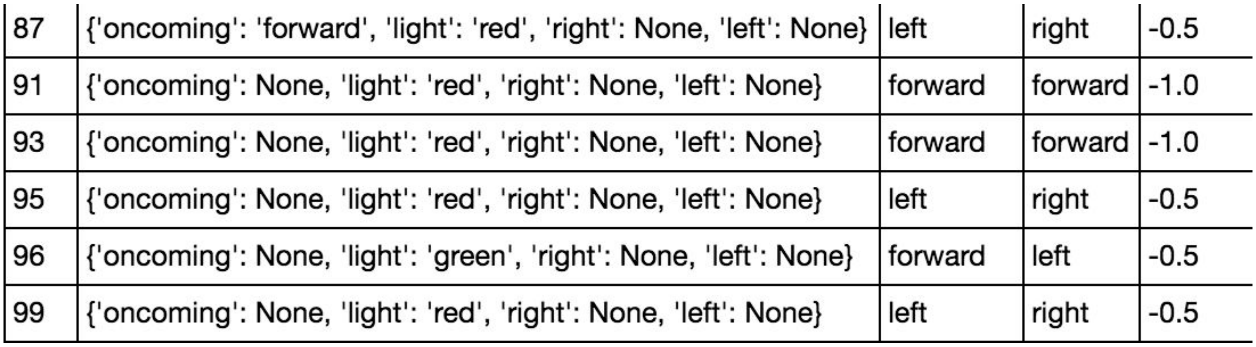
   a. We know that when alpha is big, the agent tends to give little weight to the oldValue/past experiences and pays more attention to the recent experiences.

   b. A large discount factor (like =1.0) means that agent will receive a large reward even when taking a longer time to reach the destination. On the contrary, having a small discount factor (like = 0.1) will encourage the agent to reach the destination sooner so as to maximize the reward; loitering will exponentially decrease its final reward (the one it gets when it reaches the destination successfully).
2. So, the chosen values for both alpha and gamma are 0.1, which help yield a success rate of 1.0.

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?

1. It does incur a few penalties.
2. It reaches the destination in the alloted amount of time.

The optimal policy is one where the agent does not incur any penalty and reaches the destination in minimal amount of time. However, my agent receives negative rewards in such cases because it violates the traffic rules, such as moving forward even when the light is red.



87	{'oncoming': 'forward', 'light': 'red', 'right': None, 'left': None}	left	right	-0.5
91	{'oncoming': None, 'light': 'red', 'right': None, 'left': None}	forward	forward	-1.0
93	{'oncoming': None, 'light': 'red', 'right': None, 'left': None}	forward	forward	-1.0
95	{'oncoming': None, 'light': 'red', 'right': None, 'left': None}	left	right	-0.5
96	{'oncoming': None, 'light': 'green', 'right': None, 'left': None}	forward	left	-0.5
99	{'oncoming': None, 'light': 'red', 'right': None, 'left': None}	left	right	-0.5

Table legend from left to right: (trial, inputs, waypoint, action, reward)