Machine Learning Nano Degree Project 4

July 2016

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

The agent chooses random actions (None, 'forward', 'left', 'right'). But it obeys the traffic rules. The random move agent is able to reach the target with small reward if given more time. However, it rarely makes it to the target if deadline presents. Obviously, the route chosen by the agent is not optimal and the agent does not take into account past information when it makes a new move.

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

To model the agent and its environment, we need to take into account the factors that affect the agents to reach the destination. There are in total four directions of traffics. Excluding our smart cab itself, there are three directions left. We need to know at least two of them to make a decision which way to move to. First, we need to know the color of the traffic light. Second, we need to know if there are oncoming traffic that will lead to collision with the agent. Also, we need *next_waypoint* to show the direction of the agent. Last, the left or right traffic information are needed. We choose to include the left traffic in our states. So the state can be a vector, consisting traffic light, oncoming traffic, *next_waypoint* and left traffic.
We does not include "deadline" as one state because there are 100 values of "deadline" which will enlarge significantly the state space. The agent would not be able to explore all states in 100 trials. So "deadline" is not included in the state space.

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

The cab moves in a more organized way towards the destination. The rate reaching to the destination has been increased. This is due to the Q learning algorithm does both exploring the environment randomly as well as choose the best route that maximize the rewards. Combining those two strategies, the smart cab is able to reach the destination with higher confidence.

4 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?

I perform 6 sets of parameters.

epsilon	alpha	gamma	number of trials	success runs	success rate
0.05	0.1	0.9	100	99	99%
0.1	0.2	0.8	100	93	93%
0.1	0.5	0.8	100	85	85%
0.2	0.2	0.7	100	94	94%
0.2	0.2	0.8	100	85	85%
0.1	0.2	0.9	100	94	94%

Table 1: Experimental results

The last set of parameters with epsilon = 0.1, alpha = 0.2 and gamma = 0.9 performs the best.

5 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 agent is improved by using a decay factor for epsilon parameter. The longer it travels, the less likely it would need to do new explorations. Thus it minimizes the likelihood that the agent chooses a random action which may violate the rules in the later stage of this delivery process. As a result, the success rate is quite high, which is around 94%.

Below are a few logs from the agent. We observe that there is not traffic rule violation in later stage. All moves are valid. Only a few of them gives a different direction than the planner gives.

```
LearningAgent.update(): deadline = 22, inputs = 'light': 'green', 'oncoming': None, 'right':
None, 'left': None, action = None, reward = 0.0
LearningAgent.update(): deadline = 20, inputs = 'light': 'red', 'oncoming': None, 'right':
None, 'left': None, action = right, reward = -0.5
LearningAgent.update(): deadline = 19, inputs = 'light': 'red', 'oncoming': None, 'right':
'forward', 'left': None, action = right, reward = -0.5
LearningAgent.update(): deadline = 16, inputs = 'light': 'red', 'oncoming': None, 'right':
None, 'left': None, action = right, reward = -0.5
LearningAgent.update(): deadline = 14, inputs = 'light': 'green', 'oncoming': None, 'right':
None, 'left': None, action = forward, reward = 2.0 Environment.act(): Primary agent has
reached destination!
```

The optimal policy for the agent is that it follows the shortest route between source and destination while obeying traffic rules. If one intersection is blocked, the agent should stay idle until the traffics are cleared. In this way, the agent reaches destination with minimum time, receives no penalties while having maximum rewards. Our Q learning agent has a high rate reaching the destination, which is fairly close to the best case, 100% arrival rate.