Udacity MLND P4: Train a Smartcab

Implementing 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?

What I notice in agent's behavior is that its very random. There are lot's of -1 rewards which suggest that it is certainly not performing the way its intended to. In some of the cases it does make up to the location but its just pure luck. In most of the trials the end result is "Primary agent hit hard time limit (-100)! Trial aborted."

Identify and update state

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

I think the state of the agent is described by the next_waypoint as well as the inputs which our environment is sensing.
Thus the state comprises of following key value pairs

```
{
        'Left' : ['left', 'right', 'forward', 'none'],
        'Right' : ['left', 'right', 'forward', 'none'],
        'Oncoming' : ['left', 'right', 'forward', 'none'],
        'Light': ['Red' , 'green' ],
        'Next_waypoint' :  ['left', 'right', 'forward', 'none']
}
```

So the state space comprises of 4x4x4x4x2 = 512 states which is finite but too large. However, by making some smart guess we can cut down the state space. If you notice the significance of 'Right' input, there's nothing much it conveys we can ignore it. Which reduces our state space to 4x4x4x2 = 128 states. So our final states consist of following 4 keys

('Left', 'Oncoming', 'Light', 'Next_waypoint')

Out of these next_waypoint plays an important role in helping the agent reach his destination whereas the other three are consequences of rules and good driving habits.

Implement Q-Learning

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

After implementing Q-learning and making choices based on q_values, the performance of agent improved significantly. Without deadline, the agent reached destination 100% of the time.
Also with deadline, after 100 trials, the agent was able to reach destination in time without taking any wrong step most of the time.

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?

So I made a comprehensive_search function (similar to grid search) that selects the parameters that would result best best reward/time ratio. After getting the optimal parameters the agent was performing faster and the errors (-ve rewards) also reduced slightly.

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?

Yes I think that the agent was indeed able to find the optimal policy by reaching as soon as possible and taking care of the penalties. It comes out that after training with 100 trials and then setting epsilon to 0. The next 100 trials were absolutely free of penalties. So yes, our agent did a good job.