Running the agent with random choices

The agent rarely got to its destination, yet it incurred of negative rewards along the way.

Choosing the state of the agent

The variables I chose to represent the state of the smart cab are: incoming, lights, left, right and next_waypoint.
Incoming traffic represents traffic oncoming.
Lights represent whether the lights are on or off
Left and Right represent traffic coming in on the left and the right
The next waypoint represents the next next action as received from the route planner

How QLearning was implemented

To understand how Q Learning was implemented in this exercise, we need to explain some concepts

The lookup table(Q-Table).

We used a dictionary whose keys are the states and whose value is another dictionary. This dictionary contains actions and the long term reward Q(s, a) when taking a given you are in state s.
A dictionary was used to model the Q table because it allows lookup to be done in constant time.
A dictionary was used to store the long term results of taking actions because we could easily compute the best action using python's operator keyword. We can also get the maximum Q by taking a max of the long term reward values.

Getting Best action

As explained earlier, to choose the best action that can be taken given that you are in a particular state, you get the action dictionary key with the maximum value for Q[key], where key is the state.
Once a state has not been explored, we give a default long term reward of 1 to all actions giving them equal probability to be chosen

Improvement gotten Q Learning.

The first approach before QLearning used random selection of actions. In this mode, the agent rarely got to its destination, let alone reach its destination in time.
With QLearning however, at the beginning, the agent incurred some negative scores, but it quickly improved this number. The agent learnt to reach its destination with net positive rewards and no negative rewards.

Tuning Qlearning parameters.

Two parameters that I tuned to get better results was alpha and gamma. Alpha represents the learning rate, while gamma represents the discount factor. I noticed that by setting the value of alpha at 0.1 and gamma at 0.9, the model was able to converge quickly at an optimal policy. From the 10^{th} trial, the model almost consistently experienced only positive rewards and reached its destination in time.