

QUESTION: *Observe what you see with the agent's behavior as it takes random actions. Does the **smartcab** eventually make it to the destination? Are there any other interesting observations to note?*

Answer: The agent behavior appears to be completely random in this case. The agent is roaming in the environment without any sense of direction. Agent is accumulating negative reward because of breaking traffic rules and for causing accidents. The agent is also missing deadlines because of that.

QUESTION: *What states have you identified that are appropriate for modeling the **smartcab** and environment? Why do you believe each of these states to be appropriate for this problem?*

Answer: I explored various possibilities of state spaces, and tried simulating the agent behavior with that state space. Here are some of the state space strategies I have tested:

- i) Manhattan Distance based strategy: In this strategy, I tried to use the Manhattan distance between the current location of the agent with the destination for its state space. This strategy didn't work out quite well.
- ii) Manhattan Distance based with a sense of direction: In this strategy, I have identified different states in the system depending on its direction with respect to the goal state. This strategy didn't also capture the essence of the environment and didn't do well.
- iii) Final Strategy: The final strategy I choose is very similar to the strategy I used in the last submission with some new changes. Here's the state space of my final strategy.
 - a. Light = green and next waypoint = left and inputs["left"] = None and inputs["oncoming"] - This is identified as state 0.
 - b. Light = green and next waypoint = forward and inputs["left"] = None and inputs["right"] = None - This is identified as state 1.
 - c. Light = green and next waypoint = right - This is identified as state 2.
 - d. Light = red and next waypoint = right - This is identified as state 3.
 - e. Otherwise, it will be state 4.

OPTIONAL: *How many states in total exist for the **smartcab** in this environment? Does this number seem reasonable given that the goal of Q-Learning is to learn and make informed decisions about each state? Why or why not?*

Answer: If we take the rough count values, the total number of states, which we will need to model this world, will be 48. This number doesn't seem reasonable since in order to learn a valid Q function by Q-learning we need to visit every state action pair almost infinite number of times, to model the true value of the Q function. In addition, in the real world we won't model all the intersection in a town as a state of the smart car.

QUESTION: *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?*

Answer: After many iterations of the Q-learning algorithm, the agent seems to be learning the approximation of the optimal policy. The agent seem to reaching the goal state, more rapidly compared to way it was roaming in the environment at random. Also, the agent is not missing the deadlines, as it was doing before. Here's the Q matrix from the algorithm, after 7 trails of the algorithm.

```
[[0.      3.636736  9.717031  1.38804]
 [5.65433182 12.45019291  6.16948495  0.9381]
 [0.      -0.375   -0.375    0.      ]
 [0.      -0.5     5.33622278 10.41058417]
 [0.      -0.494375 -0.99804688 -0.484375]
 [0.      0.       0.       0.      ]]
```

This behavior is occurring because the agent is learning the approximation of the true $Q(s, a)$ function (or the policy) in this environment. Agent is converging towards a optimal policy after 6th iteration itself.

QUESTION: *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?*

Answer: I trained the smartcab with the two choices of the learning rate parameter in the q_learning algorithm.

- a) learning_rate = 0.1 – While using learning rate of 0.1, the smartcab took more time to reach to a good approximation of the optimal policy. After around 51 or 52 iterations, the smartcab was able to traverse optimally in the environment.
- b) Learning_rate = 0.5 – While using the learning rate of 0.5, the smartcan took less time to reach a good approximation of the optimal policy. It started performing better actions, after only 15 or 16 trails in the experiment.

Along with the learning rate, I have also introduced other parameters like the epsilon and gamma. Epsilon is used to trade off between the eploration and exploitation phase of the system. Initially, there is a large probability value assigned for choosing random action. Since, intially we want the system to explore the environment and avoid getting stuck in a particular state. As the algorithm iterates, this value keeps on decreasing. This will allow the agent to gradually shift its behaviour from exploration state to exploitation state.

QUESTION: *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? How would you describe an optimal policy for this problem?*

Answer : Yes, the agent is able to reach the goal in most cases. After, having a good approximation of the q_matrix . The agent is able to traverse in the environment in a systematic manner. After the initial, training the agent no longer misses the deadline. The optimal policy for this environment will be take the route in the environment which gives the maximum reward in long term.