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?

Smartcab keep moving randomly and is not able to reach destination most of times. It is getting lots of negative rewards irrespective of trial number hence it is not learning anything from its history.

In our run, we were able to reach destination only 14 times by chance. It seems slightly surprising that a random agent could reach its goal more than 10% of trials.

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?

State models the agent's environment in abstract terms on basis of which it can take optimal action as per its policy.

Following variables as sensed by smartcab is identified to be part of state

- Traffic light
    - It has boolean True/False as per red/green state of traffic light
    - Crossing a red light is a violation of traffic rules and moreover it could cause accident with oncoming vehicles which makes it important parameter of state.
- Direction of oncoming vehicles in each of following directions
    - Oncoming
    - Left
    - Right
        - Each of above oncoming vehicle, if present can have one of directions (None, Forward, Left, Right)
        - These are part of state as compliance of traffic rules by agent's next action depends upon directions of oncoming car.
- next_waypoint
    - This is direction calculated by planner as best action to reach destination disregarding traffic rules. This parameter as part of our state ensures that we are reaching destination in optimal time and before deadline finishes along with abiding of traffic rules ensured by other parameters of state.

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?

We have 2*4*4*4*3 = 384 states

[Traffic Light - 2 (red/green)
Direction - 4('None', "Forward', 'Left', 'Right')
Next_waypoint - 3 ('Forward', 'Left', 'Right')]

Yes, this number seems reasonable. We have large number of trials making sure that (state,action) pairs with traffic rules violations are learnt in Q-table to avoid them in final trials.

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?

We see following differences in rationalist agent's behavior compared with random driving agent:-

- ❖ Agent is able to continuously reach destination except few misses.
- ❖ Most of rewards are positive and negative rewards decrease with each successive trial

This change in agent's behavior is due to learning as it takes different actions. Rewards received by an agent is stored in corresponding (state,action) pair entry of Q-table so that next time this state re-occurs, it would take action with best rewards and avoid traffic rules violation.

When we try to use epsilon-greedy strategy, agent would also take random-action sometimes to not get caught in local minima and find unexplored actions. We would also use historical rewards, while updating along with futuristic rewards of next state with a discount factor.

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?

We have following parameters to tune for Q-learning

- epsilon

This parameter decides how much random exploration is done to learn new rules as compared to exploitation of learnt Q-values with positive rewards.

- alpha

This parameter decides learning rate. How much learning to be done for current reward as compared to past rewards.

- gamma

This parameter decides discount factor i.e. discount on futuristic reward as per current (state, action) pair.

To measure performance of our smartcab with respect to above parameters, we used average of following quantities in 10 trials as metrics:-

- ★ Number of trials out of 100 in which smartcab is able to reach destination
- ★ Last number id(starting with 1) of trial in which smartcab failed to reach destination
- ★ Last number id of trial with a negative review

We tried following experiments to fine tune parameters based on above metrics.

1) First we used only simple rewards i.e. (state,action) entry is filled with reward of current action. Here (epsilon = 0, alpha = 1, gamma = 0)

RUN NO	SUCCESSFUL TRIALS	ID OF LAST FAILED TRIAL	ID OF TRIAL WITH NEGATIVE REWARD
1	99	1	100
2	100	0	98
3	99	1	97
4	99	20	98
5	99	1	99
6	99	60	100
7	99	24	99
8	99	1	92
9	100	0	95
10	99	1	88
AVERAGE	99.2	10.7	96.6

2) Now we use learning rate to use past Q-value along with epsilon greedy approach to explore random new actions.
(epsilon = 1/self.time, alpha = 1/self.time, gamma = 0)

RUN NO	SUCCESSFUL TRIALS	ID OF LAST FAILED TRIAL	ID OF TRIAL WITH NEGATIVE REWARD
1	98	9	98
2	99	1	100
3	98	73	97
4	99	21	100
5	100	0	99
6	99	1	93

7	99	2	84
8	99	1	90
9	99	5	98
10	98	69	88
AVERAGE	98.8	18.1	94.7

4) Now we add gamma(a discount factor) for taking into account futuristic rewards (epsilon = 1/self.time, alpha = 1/self.time, gamma = 0.1)

RUN NO	SUCCESSFUL TRIALS	ID OF LAST FAILED TRIAL	ID OF TRIAL WITH NEGATIVE REWARD
1	96	100	100
2	99	1	99
3	98	68	76
4	97	32	95
5	98	3	91
6	99	75	98
7	100	0	85
8	100	0	92
9	99	1	93
10	93	82	95
AVERAGE	97.9	36	92.4

We see that all of experiments yields high number of successful trials. As in first experiment with just using current reward to update Q-table, we see that chances of getting failed in reaching destination is lowest at last trials, so we would just use (epsilon = 0, alpha = 1, gamma = 0). It seems that we these parameters might be useful to more complex systems where uncertainty is high.

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?

A smartcab with an optimal policy would always be able to reach its destination without incurring any penalties (negative reward) in minimum possible time. Our smartcab agent is able to reach destination most of time failing only 1 time out of 100 trials to reach the destination. Also trial in which our agent failed to reach destination is far off from last trial validating that our agent is continuously learning and following an optimal policy.