SmartCab Report

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: Agent moves randomly through the grid. It stops at the intersection where light is red. Smartcab does stumbles upon the destination sometimes. However the rate is really low. It keeps moving agnostic to other cars and get negative rewards quite often for not sensing the other cars and taking decision.

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: The states I used to model the smartcab are-

1. Next waypoint - This state is necessary to make sense of the change in starting point and destination in each episode. As the destination and starting point are chosen randomly at start of a simulation, if we do not use this state, agent will not be able to learn to drive towards destination.
2. Light - This state is helpful in learning strategies when we see green/red light.
3. Oncoming traffic - This state should also be helpful for learning strategies while we sense oncoming traffic.
4. Left Traffic - This state helps to learn strategies to avoid traffic violations in case of Red light

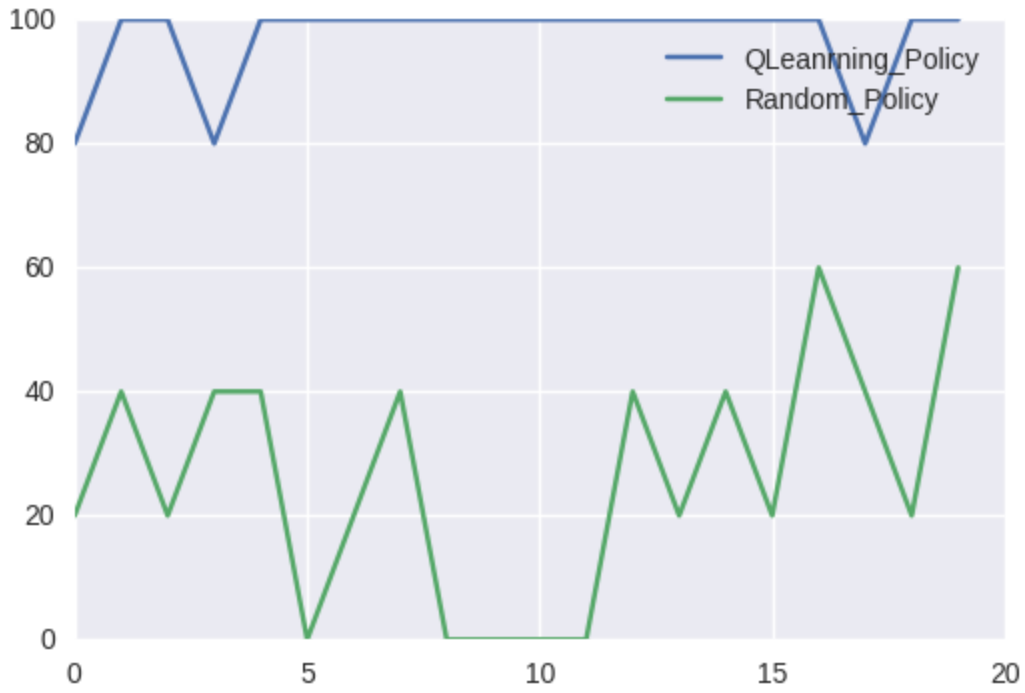
Above states and transitions between these state should be able to capture the policy for reaching destination in time. A few other states which were not included are -

1. Right Traffic - This state does not give any extra information and can be inferred from other states. Including this state would not help in learning any better policy but may make learning to take longer as the agent will have to learn the hidden correlation between this and other (left traffic, light) states.
2. Deadline - This is another input which was not included. The deadline can take arbitrary large value and including this variable as state may make state space large and hence hard to learn optimal policy.

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?

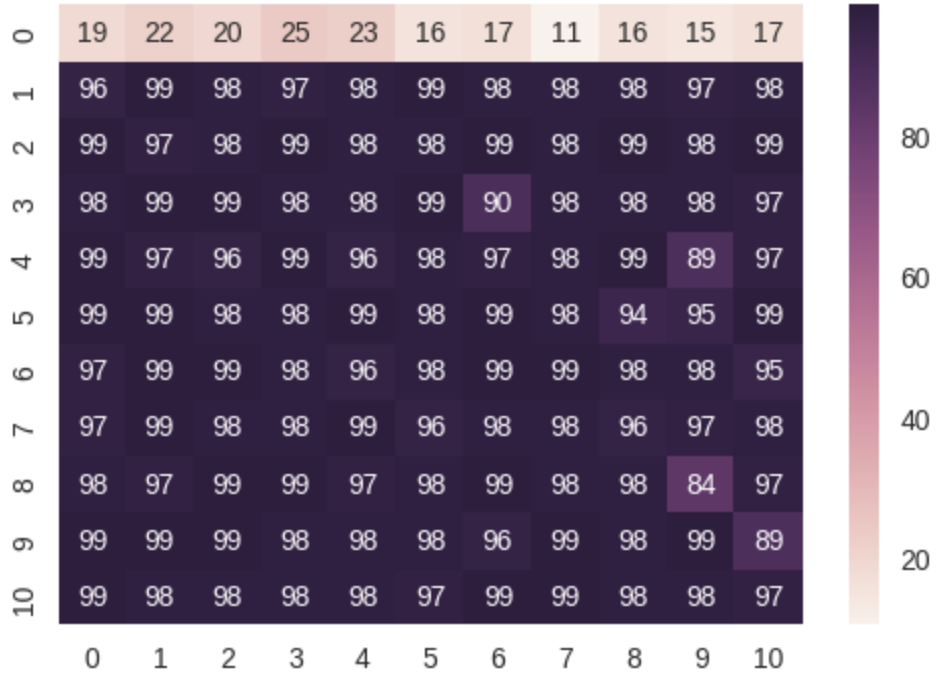
Answer: After implementing the q learning, agent is able to reach the destination in more trials and avoids getting a negative rewards for violating traffic rules. The q learning agent learns quickly and only after a few trials, it starts reaching destination almost perfectly all the time.

Below is the graph showing the difference in performance of Random policy and Q-learned policy. Each point on X axis represents a bucket of 5 trials. Y axis represents the the percentage of times the cab was able to reach its destination. As apparent from the graph, the random policy did not perform too well. However, Q learning policy was almost perfect after a few trials.

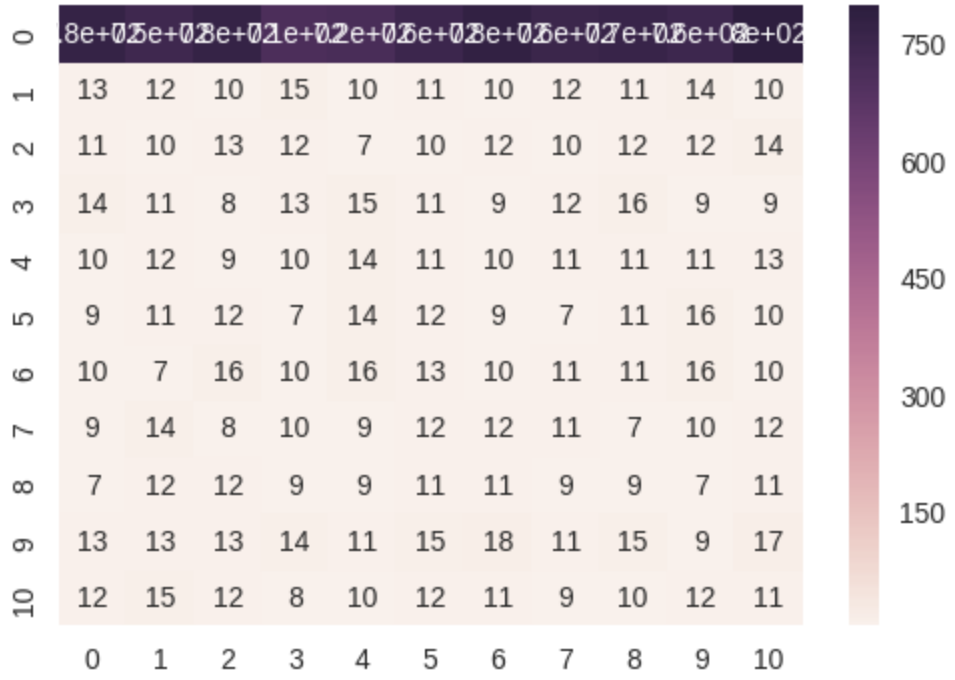


Question: Report the different values for the parameters tuned in your basic implementation of Q-Learning. For which set of parameters does the agent perform best? How well does the final driving agent perform?

Answer: The Q learning model was optimized for parameters alpha and gamma. The heatmap below shows the performance of Q learning algorithm for the parameter space. X axis and Y axis shows the variation of parameter in range (0.0 - 1.0) for alpha and gamma respectively. (Note the values shown on axis (0-10) are parameter*10 values.). Each grid value shows the number of times the cab was able to reach destination out of 100 trials. It shows that results of gamma=0.0 performed quite poorly which is expected as with gamma as zero, the Q value does not learn from earlier Q value (so the learning is myopic and based on only near future rewards).



Another grid below shows the the total number of times the negative reward was received by the agent for not following traffic rules. The results are similar, at gamma = 0.0, the number of violations are quite high. For other values, the number of violations is quite low.



Given the above result, I choose the following values for my model.

- alpha=0.1 - As the model has been able to learn its object relatively quickly for almost all values, a smaller value of alpha will make sure that the learning is stable in all cases (larger values of alpha may make the convergence of utility values hard).
- Gamma=0.8 - From the above heatmaps, with alpha = 0.1, this value of gamma presents high success rates and lowest number of penalty cases.

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:

Defining an optimal policy directly even for a simple task as smartcab is a hard task (and proving that the policy is optimal is even harder). For smart cab, optimal policy should avoid any penalty and should reach the goal for sure. With these constraint, path with minimum delay would be optimal.

It is not possible to directly calculate an optimal policy as the model as the rewards for each stage are not explicitly provided. Moreover the presence of other cabs will make it extremely difficult to calculate the optimal policy directly even if the rewards are given. The results however show that Q values do not change much after 90 trials and point to that it is quite near to optimal

policy. Also, on visual inspection of last 10 trips, cab always reached the destination, does not go in circles (as was doing with random policy). However, it violated the rules a few times. It is not perfect on the measure I mentioned earlier, however probably optimal based on the states it can sense.