Project Report: Train a smartcab to drive

Task 1: Implement a Basic Driving Agent

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?*

If the deadline is enforced, the random agent could reach the destination 22 out of 100 trials. If the deadline is not enforced, the random agent could reach the destination 67 out of 100 trials. Although the success rate increases when the deadline is not enforced, the random agent takes much more steps (7855 compared with 2743) and violate the traffic rules significantly, considering the negative total rewards (-273.5).

Task 2: Inform the Driving Agent

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?*

The variables I identified to model the state of smartcab:
light: the traffic light (red, green)
oncoming: the oncoming traffic (None, left, right, forward)
left: the traffic coming from left (None, left, right, forward)
next_waypoint: the route the agent takes (None, left, right, forward)

When thinking about the state, the primary concern is to figure out what variables determines the traffic rules. All these variables are really useful in performance. Following traffic rules, if the traffic light is green, the agent could move forward, or make left turn considering oncoming traffic, or make right turn. If the traffic light is red, the only legal movement for the agent is to make right turn considering no traffic coming from left. In either case, we do not need the information about traffic coming from the right, thus I do not include “right” in the state. And the next_waypoint serves as an important input to model the car. “Deadline” could also be an useful information, but it could dramatically increase the size of state and number of samples needed for the convergency to the optimal policy.

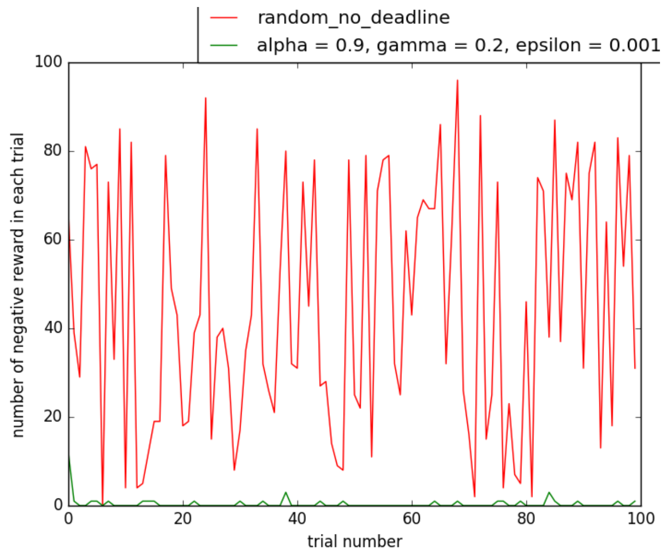
There are 2*4*4*4 = 128 states in total.

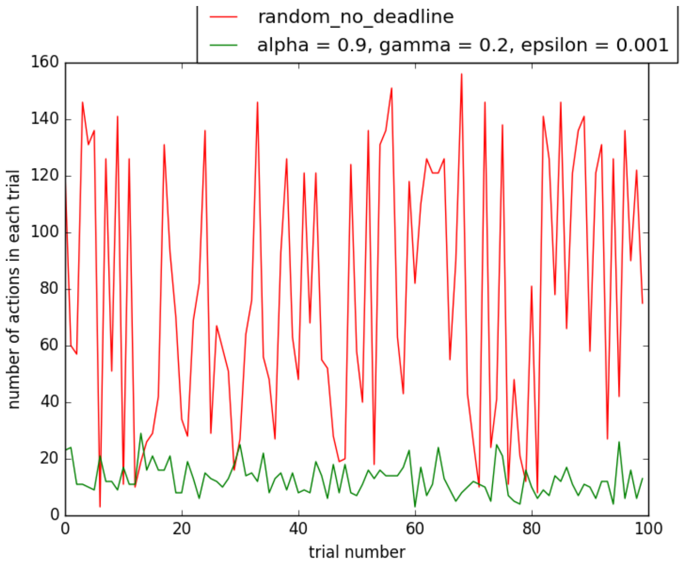
Task 3: Implement a Q-Learning Driving Agent

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?*

After implementing Q-learning on driving agent, I could see that the agent is able to make intelligent decision at each intersection. In the example shown below (qlearning_output_1), the agent could 100% successfully reach the destination with a continuously increasing total reward over 100 trials. From the left graph below, we could see that the random agent’s learning ability does not change over trials since it keeps violating traffic rule and receiving negative rewards, whereas the Q-learning agent tends to make the right decision after violating traffic rules for the first several trials. On the other hand, Q-learning agent still makes 1 more 2 mistakes in the following trials. And from the right graph below, we could see q-learning agent is much more efficient than the random agent, considering it only takes around 20 steps to reach the destination.

To sum, the random agent does not learn anything from previous trials. But the Q-learning agent learns an optimal policy as the program updates the Q-Table over the trials.





Task 4: Improve the Q-Learning Driving Agent

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?*
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?*

file name	alpha	gamma	epsilon	total actions	total rewards	successful trials
qlearning_output_1	0.9	0.2	0.001	1282	2235.5	100
qlearning_output_2	0.9	0.5	0.001	1357	2182.5	98
qlearning_output_3	0.5	0.2	0.001	1337	2240	99
qlearning_output_4	0.9	0.2	0.01	1385	2252	98

qlearning_output_5	0.5	0.5	0.01	1420	2430	96
random_output_enforce_deadline	NA	NA	NA	2743	12	22
random_output_no_enforce_deadline	NA	NA	NA	7855	-273.5	67

I performed 5 set of parameters to improve the Q-learning agent. The optimal parameters are:
alpha = 0.9 (higher alpha)
gamma = 0.2 (lower gamma)
epsilon = 0.001 (lower epsilon).
With the optimal parameters, the Q-learning agent could 100% reach the destination with a continuously increasing total reward over 100 trials. It shows least total actions (1282) and a high total rewards (2235.5).
I think the Q-learning agent is not close to finding an optimal policy although it did a very good job of reaching destination. But from the two graphs shown in previous question (green line), we could see that it still incurs penalties and takes lots of steps to reach the destination. As an optimal policy, it should never violate the traffic rules.