**Train a Smartcab to Drive**

Yiying Wang

July, 2016

**Task 1**

**Implement a basic driving agent**

**Question: what you see in the agent’s behavior. Does it eventually make it to the target location?**

The agent randomly chooses an action from “stay put”, “turn left”, “turn right” or “go forward” at each time point. The agent’s action most of the time is not the same with the one that the planner told it (as shown right next to the car). When the agent’s action is randomly chosen to be “**forward**”, it knows to wait for the traffic light to turn green before it makes a move (this new move changes, it is not necessarily the same as before). When the agent’s action is randomly chosen to be “**left**”, if the light is red, it knows to not make a move and choose again if there is another car coming straight from forward. If the light is green and there is either no cars coming from front or the oncoming car is about to take a left, agent takes a left. When the agent’s action is “**right**”, if the light is green, it will turn right right away. If the light is red, it will look at its left, if there is no car on its left or if the car on its left doesn’t want to go straight, it will turn right.

After a lot of driving around, wandering back and forth, the agent can eventually reach the destination after a lot of steps (much larger than the required time frame).

**Task 2**

**Identify and update sate**

**Identify a set of states that are appropriate for modeling the driving agent. Justify why you picked these set of states, and how they model the agent and its environment.**

I pick the set of states as 5 variables and put them in a namedtuple data structure.

1. Next\_waypoint: this is the basic instruction of which direction that the agent can take. Without this input, the agent cannot learn how to reach the destination most quickly.
2. Light: red or green. This is the traffic light for agent. The agent could not run the red.
3. Oncoming: this is to indicate if there is another car coming from the front at the same intersection with the agent. If the value is “None”, it means there is no car coming from front. Otherwise, its value indicates the direction the other car is going, e.g. “left”, “right”, “forward”.
4. Left: this is to indicate if there is another car coming from the left at the same intersection with the agent.
5. Right: this is to indicate if there is another car coming from the right at the same intersection with the agent.

I picked these variables because they are most relevant for agent to make the optimal decisions. Without the higher-level planner assignments (**next\_waypoint**), the agent is aimless in a sense.

**The rest variables** are the information for the surroundings. The light status is important because the agent needs to obey the traffic rules and not move or yield to traffic when the light is red. The information of other cars that are at the same intersection is important because our agent need to know if it should yield to other traffics. If it runs into other cars, it should get a big penalty so that it can learn that he should take none action in that case. **Deadline** is not included in state because the agent’s action is not very related to deadline. If the agent is at a position that needs to take a left to get to the destination, it’s action should be ‘left’ either the deadline is 20 or the deadline is 5. Also, the inclusion of deadline would inflate the size of Q-table and makes the algorithm takes much longer to converge.

**Task3**

**Implement Q-learning**

**What changes do you notice in the agent’s behavior?**

At the beginning of the run, the agent was aimless and making many mistakes. It was not able to reach the destination within in the required time. But as it learns, it made less and less wrong choices and was able to reach the target in time.

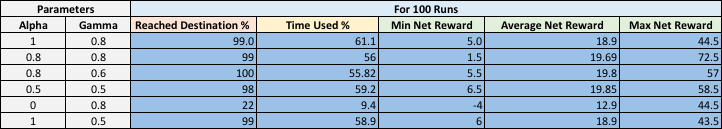
**Task4**

**Enhance the driving agent**

**Report what changes you made to your basic implementation of Q-learning to achieve the final version of the agent. How well does it perform?**

**Does your agent get close to finding an optimal policy, (reach the destination in the minimum possible time, and not incur any penalties)?**

Below is a summary table of performance for smartcab to run the first 100 times.



The column “Reached Destination %” shows the number of runs that reached destination within allotted time out of 100 runs. The column “time used %” is the percent of remaining time to the total allotted time when the smartcab reached the destination (mean of 100 runs). The next three columns are the minimum, mean or maximum net reward the smartcab got for 100 runs. According to this table, the driving agent is able to reach the destination consistently within allotted time except the case when alpha is 0. Of all parameters combinations tested, I think the one that performs best is when Alpha = 0.8 and Gamma = 0.6 because it reached destination 100 % times, it has the lowest time used % and its minimum net reward is greater than 0. An ideal policy would be the one that leads the agent to reach destination within allotted time in the smallest steps and not get any negative penalties. The best performance of this agent (namely, alpha = 0.8, gamma = 0.6) can meet the above requirements in the last ten trials because it can reach the destination in a few steps and there are no -1.0 rewards (representing disobey traffic laws) recorded.