1. **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?

When the agent takes random actions, it just moves anywhere randomly. Of course, there is a small possibility it will reach its destination at random. It did not make it to its destination during the simulation. The agent does not mind attempting to collide with cars nor does it pay attention to traffic lights.

1. **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 states I have identified as appropriate for modeling the smartcab are the inputs and the next\_waypoint. The inputs give traffic directions such as lights and oncoming. Oncoming tells whether a car is coming and what direction. This will allow the car to not crash into another car. The lights tell whether the car is at a red or green light. This enables the car to obey traffic laws. The ‘left’ input tells the agent if a car is approaching from the left. This is important for turning right at red lights. The ‘right’ input which told the agent whether a car was coming from the right was removed. No driving laws are there for cars coming from the right. The deadline is not included in the state because I do not want the agent to disobey traffic laws in order to get to its destination to earn 10 points, which is possible if given a deadline. Also, there are over 50 possible values for deadline, which would make the total states larger.

1. **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?

In total, there are the inputs by next\_waypoint = 2(lights)x4(oncoming)x4(left)x3(next\_waypoint)=96. This number seems reasonable. The agent must factor in its surroundings along with understanding the direction it needs to go in.

1. **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?

I notice that the agent is trying to determine the optimal path. It barely makes mistakes after the third trial. The main difference is that the agent is now finding an optimal path easily, whereas the random state would rarely ever get to its destination, let alone moving properly. This behavior is occurring because the agent is rapidly starting to learn the best algorithm.

1. **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?

I imported the agent’s state. When the location was equivalent to the destination, I marked it as a success. I also recorded the trials which it succeeded.

Count is the number of times it has visited the state. For the first set of parameters, the learning rate is , the gamma is 0.1, and the epsilon is . The number of successes is 100. For the second set of parameters, the learning rate is , the gamma is 0.5, and the epsilon is . The number of successes is 99. For the third set of parameters, the learning rate is , the gamma is 0.5, and the epsilon is . The number of successes is 100. The final driving agent performs great when given a deadline. There is minimal difference between the parameters, but the first and third one performed the best with 100% of their trials.

1. 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?

My agent finds the optimal policy after 100 simulations. An optimal policy would be for the agent to head towards the destination, decreasing the distance at every step. If the destination is straight ahead, keep going straight and stop for red lights. If the destination is to the right, watch for oncoming traffic, then turn right and go straight ahead. If the destination is to the left, watch for oncoming traffic and wait until the light is green, then turn left and go straight ahead. If the destination is straight ahead and to the right, keep going straight ahead, obeying lights, but at the first red light, make a right if no oncoming traffic. If the destination is straight and to the left, keep going straight ahead, obeying lights, but at the first green light with no oncoming traffic, make a left. If destination is directly behind, make two rights while obeying light and oncoming laws. If destination is directly behind and to the right, make a right while obeying light and oncoming laws (now it’s straight ahead to the right or to the right). If destination is directly behind and to the left, make a left while obeying oncoming and light laws (now it’s straight ahead to the left or to the left).

But the policy the agent will eventually get to is to obey traffic laws, and when permitted to move with respect to those laws, move according to the next\_waypoint.

1. In your report, mention what you see in the agent’s behavior. Does it eventually make it to the target location?

I see my agent learning in a very effective manner. It stops for red lights and all other traffic laws. My agent always makes it to the target destination.

1. Justify why you picked these set of states, and how they model the agent and its environment.

The agent need to know if there is oncoming traffic. ‘oncoming’ tells the agent whether a car is oncoming or not. The agent needs to know what the light’s color is. ‘light’ tells the agent what color the light is at its current location. The agent needs to know if a car is coming from his left in case the agent wants to turn right on a red light. The agent also needs to know the next\_waypoint it is expected to go to to get the biggest reward.

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

I notice the agent learning. It no longer appears to move in random directions. It reaches its destination every time. It also obeys traffic laws.

1. 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?

The final parameters are learning rate at , gamma at 0.1, and the epsilon at . The model makes it to the destination 100% of the time. The number of successes is 100. It pretty much knows an optimal policy by 10 trials.

1. 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?

The agent finds an optimal policy. The agent does not wander off or make mistakes after the fifth trial. If it does make mistakes, they are very seldom. The agent also incur any penalties after the fifth trial. It has found an optimal route. An optimal policy is mentioned in problem 6.