Tasks

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?

Answer: The Driving Agent's behavior as it takes random action within the simulation is that it does not care whether the light is red or green, whether or not there is an oncoming vehicle. It randomly goes "right, left, forward or none" without regards to the target location. It does not arrive at the target location. Every reward is equal to float(zero). The agent does not seem to reach the destination. Here is a sample after over hundred iterations as follows:

LearningAgent.update(): deadline = -100, inputs = {'light': 'red', 'oncoming': None, 'right': None, 'left': None}, action = None, reward = 0.0

Environment.step(): Primary agent hit hard time limit (-100)! Trial aborted.

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?

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?

Answer:

The states to consider are the following:

1. Light: Red or Green, (2)
2. Oncoming: forward, left, right, or None, (4)
3. Left: forward, left, right, or None, (4)
4. Next waypoint: forward, left, or right, (3)
   With the above five folded states, the state-space in the q_learning is the following size: 2*4*4*3 = 96

Yes, this number is reasonable because if you follow the U.S right-of-way rules, it does not matter much whether there is a vehicle at your right unless it is a stop and go intersection or you are in the middle of the intersection trying to turn left while light has turned red after yielding for oncoming vehicle. But with this environment, the light turns either red on green, no other stop signs or blinking red light for stop and go situation. Therefore, omitting the state “right “reduces the state size and the process speed without considerable impact.

If I were to add the state Right: forward, left, or right, or None, the state size would have been:

2*4*4*3*4 = 386 instead of 96.

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?

Answer: The agent follows the U.S right-of-way and traffic rules for 90 percent out of 100 trials. It also reaches its destination within the deadline. It receives more positive rewards and very few negative rewards. This is because I implemented the agent's Q-Learning to choose between staying in same intersection or choosing the next waypoint depending on the traffic light and the next destination not by random. Staying, in another word doing nothing if the light for example is red may delay the time to reach the destination but can prevent accident. Choosing the right nex_waypoint is necessary for reaching the destination on time.

Improve the Q-Learning Driving Agent

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

After trying many different learning rate and discount factor values, I calculate the average reward by the total moves and I had the alpha values = [0.1, 0.3, 0.5, 0.7, 0.9] and the same values for the learning rate [0.9, 0.7, 0.5, 0.3, 0.1] which gives 5*5 = 25 combinations. For alpha=0.9 and gamma=0.5 yield the highest reward. The agent did not receive any punishment or negative reward from trial 87 to trial 99. Therefore, it learnt from its mistakes and performed well in the final.

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?

I think the action selection method is optimal in my implementation of Q-Learning. I made the learning agent choose between two actions: either choose the next waypoint as the action, or stay in the same intersection. The benefit of this is that there is less space for the learning agent to explore. Taking the action of going to the next waypoint brings us closer to the destination, but can be risky of violating traffic laws (such as red light, or other vehicles). On the other hand, doing nothing does not get us closer to reaching the destination on time, but may prevent violation of traffic laws (and be less of a safety risk). Thus in optimizing my implementation, I instead chose to fiddle with the learning rate and the discount factor. After 100 iterations, the agent seems to consistently take correct actions. I also think that by increasing the number of dummy agents, the car would probably interact more with them, and would accumulate penalties to avoid earlier than later. This would then translate to less training time or less iterations with penalties, and would be a nice enhancement.