Kamal Kamalaldin

10/29/2016

Udacity MLND program

Smartcab project

Project Report

This report is provided to answer all questions that are posed by the smartcab implementation. The questions are listed in order of implementation, and the answer is provided after each question.

Question 1

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?

With the random choice implementation, the smartcab takes a very long time to reach the destination, and has no particular strategy as to how to reach it. The smartcab seems to go opposite to the direction of the waypoint at times, which is not an optimal behavior.

Question 2

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?

I identified the following the be important aspects of the state, and added reasons for why so: 1. Oncoming: This part of the state tells the cab if there are oncoming cars in traffic. This state variabel could inform the car's decision when facing a trafic light that has an oncoming car. If a car is oncoming, then the car should make a different decision from when there is no oncoming car.

1. light: This part of the state informs the cab that it is or isn't moving in the next time step. If it is not moving, then the smartcab might think it is actually advantageous to simply turn right at the red light, if permitted.
2. right and left: These state variables tell the smartcab whether there is an incoming right or left car.
3. waypoint: This state variable tells the smartcab where it should be heading to next in order to get to the waypoint. This is perhaps the most important state variable.

Question 3

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?

Since the first 4 states can either be true or false, the space for the state is 2^4= 16, and the waypoint can hold 3 variables, so that space is 2^4 * 3 = 48, which is a manageable state space for a Q learning algorithm of this size and computational capacity.

Question 4

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?

The agent, after 50 iterations, starts to head straight to the target. This behavior is occurring because of the Q learning algorithm I implemented. This algorithm ensures that the smartcab learns from the rewards it receives from the environment. In essence, the Q learning algorithm records and updates what the smartcab took at each state and the reward it received for that action. The policy of the smartcab is to take the best action given by the Q algorithm 75% of the time, and make a random choice 25% of the time. The 25% random choice ensures that the smartcab does not get stuck in a local minimum due to improper initialization or an early mistake.

Question 5

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?

Note that the files Environment.py and Simulation.py were modified in order to allow the collection of simulation results.

I implemented a graphing function taken from the different simulations with different ranges of values for alpha, gamma, and epsilon. This function plots the success rate of the last 10 trials, beginning from trial 10.

The following are some of the good parameters, and one is labeled as BEST.

- alpha: 0.9, gamma: 0.0, epsilon:0.1
- alpha: 0.3, gamma: 0.9, epsilon:0.1
- alpha: 0.3, gamma: 0.3, epsilon:0.1
- alpha: 0.6, gamma: 0.3, epsilon:0.1 BEST
- alpha: 0.6, gamma: 0.3, epsilon:0.3

Question 6

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?

Yes! The agent gets very close to finding an optimal policy, given a good set of initial hyper parameters. The policy rate (epsilon) might sometimes influence the success of the smartcab since it results in a random move which may cause a long detour from the destination. An optimal policy for this problem would have a somewhat high learning rate (0.6-0.9) and a somewhat medium to low utility rate (gamma), and a small exploration rate (epsilon). Although the small exploration might be harmful in the beginning of the training (because the smartcab might get stuck in a local minimum of optimality), in the long run the smartcab will make smaller random "mistakes" when implementing what it learned.