SmartCab project

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?*

The smartcab eventually reached its destination once but that was completely luck. Most of the times the smartcab doesn't reach its destination and isn't smart… yet. I notice that the smartcab is performing completely random actions, thus it receives rewards randomly.

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 environment entails various variables that influence the state the smartcab is in:

- deadline, which is the number of actions left before the game is over
- the 'inputs' features indicating if there is another car either oncoming, on the left or on the right and its direction but also whether the light is green or red for the smartcab.
- Next waypoint which is the direction the smartcab should go to reach the next intersection towards the destination.

Deadline may not be appropriate in our case for modeling the environment because it makes the space significantly bigger and in 100 trials it is unlikely that the agent encounters the exact same state.

With all the variables left we have:

- Light: 2 possible states, red or green
- Oncoming, left, right have 4 options each which are left, right, forward or None.
- Next waypoint: 3 options: forward, left or right.

So 2*4^3*3 = 384 possible states.

I believe each of these states are appropriate for this problem because they each correspond to a particular situation that the smartcab can encounter and for which it has to learn how to respond to. But after having thought about it, information about cars coming on the right might not be relevant because they don't cross the lane we're in unless there's no car approaching on their left or the light is green for them. So this piece of information doesn't seem to matter to learn the rules of the road. By removing this input we now have 96 states.

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 the Q-learning algorithm, the smartcab now often manages to reach its destination and that is because it gets a small reward for going to the next waypoint. The smartcab takes random actions at first, modifying the Q-values with the reward it gets, until the right action for a given state becomes having a Q-value higher than the others. This action with the highest Q-value is being chosen consistently, thus reinforcing the right behavior.

However, I noticed that the smartcab sometimes circles around one block and then resume the shortest path. Here's my hypothesis. We know that turning right at a red light is okay when there is no car coming on the left so the smartcab received a small reward each time it does that. At some point the Q-value for turning right becomes larger than the Q-value for holding still ('None') so the smartcab is taking a right instead of waiting at the light and then going forward. Also, since we took away 'deadline' as an input, there's no incentive to hurry so the algorithm might be tempted to take detours to collect more rewards on the way.

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?*

Here are different parameters used to test the Q-learning algorithm. Success is defined as "the smartcab reached its destination before the deadline".

With 'right' input taken into account:

My first basic implementation of Q-learning took into account 384 states and had the following values:

- Epsilon = 0.3
- Gamma = 0
- Learning rate = 1

Average reward for all trials = 18

Successes: 75/100

After tuning the learning rate:

- Epsilon = 0.3
- Gamma = 0
- Learning rate = 1/ log(t+2)

The smartcab reached its destination 80 times out of 100 with an average net reward of 19.

Without right input taken into account:

After reducing the space to 96 states.

First run:

- epsilon = 0.3
- gamma = 0
- learning_rate = 1

Successes: 83/100. Average reward for all trials: 17

Second run:

- epsilon = 0.3
- gamma = 0
- learning_rate = 0.9

Successes: 81/100. Average reward: 19. Note: 2 failures in the last 10.

Third run:

- epsilon = 0.3
- gamma = 0.3
- learning_rate = 1/math.log(t+2)

Average reward for all trials = 17

Successes: 47/100

Fourth run:

- epsilon = 0.3
- gamma = 0.9
- learning_rate = 1/math.log(t+2)

Average reward for all trials = 1

Successes: 27/100

Fifth run:

- epsilon = 0.1

- gamma = 0
- learning_rate = 1/math.log(t+2)

Average reward for all trials = 22

Successes: 93/100

Sixth run:

- epsilon = 0
- gamma = 0
- learning_rate = 1/math.log(t+2)

Average reward for all trials = 21

Successes vs Failures: 95/100

The agent seems to perform best with the parameters of the sixth run:

- epsilon = 0
- gamma = 0
- learning_rate = 1/math.log(t+2)

This seems odd because with these parameters one part of the equation is discarded because gamma equals 0 and the agent is not doing any exploration because epsilon equals 0.

However, the agent seems to perform well with Average net reward around 20 and successes around 90+/100.

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?*

The agent is close to an optimal policy because it reaches its destination in a fairly short amount of time but it still gets some penalties from time to time. An optimal policy for this problem would be such that the agent never gets a penalty and always get to its destination in the minimum possible time. The final driving agent is not following this optimal policy since it still receives penalties.