Machine Learning Engineer Nanodegree

Reinforcement Learning

Project 4: Train a Smartcab How to Drive

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 smartcab does pretty much what you expect - it behaves randomly. It would occasionally stop for no reason, run lights and completely discregard traffic. Sometimes it would do the right thing and in very rare cases make it to the destination.

There are 3 other agents that exist simultaneously in the world. The observed rewards are as follows:

- Violating a traffic rule: -1.0
- No violation but going in a direction other than specified by the planner: -0.5
- No violation and going in the right direction: 2.0
- No violation, right direction and reaching the destination: 12.0

Inform the Driving Agent

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 identified state consists of all the inputs plus the next waypoint. The following are its parameters:

- light
- oncoming
- right
- left
- waypoint

Generally speaking, our problem is to get our smartcab to a specified location in our grid world. Normally, this would be much more complicated. However, pathing has already been solved for us by the planner module as it already tells us where the next waypoint is. This also effectively let's us bypass the problem of localization, so we don't have to worry about sensing and ascertaining where our smartcab is exactly. What we do have to worry about is the conditions of its immediate vicinity (the inputs) and where it must go next (next waypoint). So the inputs plus the waypoint is all we need really to take an action.

Each parameter can take on the following values:

- light ['green', 'red']
- oncoming, right, left [None, 'left', 'right']
- waypoint [None, 'left', 'right']

Therefore, altogether there are 2x3x3x3x3 = 162 total states. It's a small number compared to possible events that can occur in a real world environment. If we were to take account different vehicles, pedestrians, road signs, weather, road elevation, etc., it would certainly be a lot more than 162 states.

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 smartcab's behavior would start out random but would slowly correct itself once negative rewards were encountered. For example, after it runs a stop light and receives a negative reward, it would stop doing that and try something else the next time around. This is because the Q-value of that state-action pair is degraded and is seen as a less appealing option by the smartcab.

Interestingly, the cab encounters more common cases first (usually without traffic) and therefore has Q-values for those states-action pairs first. It can get confused when it encounters an entirely new state as the actions for that state will always start out with uniform Q-values. For example, even though it ran a stop with no traffic before and got a negative reward, if there was suddenly any traffic in any direction, it's possible that it will run the stop again so it can seemingly make the same mistakes.

Improve the Q-Learning Driving Agent

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?

ANSWER:

I have chosen to implement the 'optimistic' version of the Q-leaning algorithm where the agent begins with sufficiently high Q values but approaches correct values as it acquires penalties for wrong actions. This is an elegant solution to this particular problem because:

- of the way the rewards are structured
- each state is independent from each other
- there is only one correct action for each state

As such, we only need to tune 2 parameters - learning rate alpha and our initial Q-values. We run the simulation for a combination of learning rates and initial Q-values:

- Learning rate: 0.2 to 1.0 in increments of 0.2
- Q-values: 0, 2, 10

The results are as follows:

```
    Alpha  Ave Penalty  Ave Reward  Efficiency  Q Initial  Success Rate
0     0.2       -0.400      13.860   28.484841          0            48
1     0.4       -0.385      13.535   23.987843          0            43
2     0.6       -0.400      13.920   31.340238          0            52
3     0.8       -0.410      11.245   22.701508          0            36
4     1.0       -0.380      13.080   25.924206          0            48
5     0.2       -0.490      22.250   61.520794          2           100
6     0.4       -0.400      22.900   59.780916          2           100
7     0.6       -0.370      22.010   60.900072          2            99
8     0.8       -0.255      23.025   59.392063          2           100
9     1.0       -0.285      21.535   60.724286          2           100
10    0.2       -1.140      21.520   53.170736         10            95
11    0.4       -0.775      21.320   54.215447         10            96
12    0.6       -0.550      22.245   57.467482         10            99
13    0.8       -0.480      22.395   57.158153         10            99
14    1.0       -0.485      22.285   53.989149         10            98
```

We note that the inital Q-values are extremely important and the cab performs optimally at 2 with success rates close to 100. As for the learning rate, 0.6 to 1.0 are good values.

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?

ANSWER:

All in all, the cab has a very good success rate - around 99 out of 100. At best, the cab has an efficiency of 60%. That is, it only uses up 40% of the steps allotted to arrive at the destination. The average penalty per trip is around 0.3, which means it still makes a few mistakes along the way so the policy is not perfect. This is because the cab still has not encountered all possible states so it will behave naively when it is faced with a new state.

An optimal policy put simply is - proceed to the next waypoint when it's safe (no violations), stay in place otherwise. We can arrive at this optimal policy by either letting the cab 'experience' all combinations of 162x4 states and actions or apply some heuristics.