Machine Learning Nanodegree - Smartcab

Richard Deurwaarder

1 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?*

The smartcab usually makes it to the destination, but not always. There is an 100 step hard-limit on trials so every once in a while it will go up to that limit and a new trial is started.

I tried to see what would happen if I'd let it go only *forward*. I notice it does stop for red lights (with negative reward). This might be useful to know when creating/debugging the other tasks .

2 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?*

For the states I used the following as input: State of the light, traffic from all three directions and their direction, the direction I want to go (based on the planner).

I combine the direction with the traffic from all three directions to determine if I am going to 'cross' any traffic. For instance: if I want to go *left*, and there is traffic on my *left* which is going *forward* or *left* they will cross me. This will give me the boolean value *true* in this case.

Using the traffic crosses variable with the state of the light and the direction I want to go, I create the state. I put it in a string so ease of use in python dictionaries. An example state where the light is green, no traffic that crosses me and I want to go forward would be: false-green-forward.

I use these states to limit the total number of states in an attempt to give Q-Learning a better chance to get good results. The way I basically limit the amount of states is to ignore certain data. For instance when I am going *right* it really does not matter if there is any traffic on the intersection going forward because they will not cross me.

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?*

Because my state is combined from a boolean, and red/green option and a direction. The number of states are: $2 * 2 * 4 = 16$. I don't believe the planner actually uses the None direction. So in reality it's: $2 * 2 * 3 = 12$

Given that Q-learning optimally needs infinite trips to each state we would like to limit the amount of states. The original input + direction would have given us: $2 * 4 * 4 * 4 * 4 = 512$ states (lights * left * right * oncoming * my-direction) so this is a good improvement!

3 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?*

I was not entirely sure how this question was different than the next question, so I interpreted as gamma = 1, learning rate = 1, epsilon = 0 (no exploration). Epsilon is defined as the chance to take a random action.

This obviously gave pretty terrible results, the car was moving in circles (to the left to be precise). This happens because initially every Q value is set to *None*. This means the code to choose an action will default to the first action (left). Now two things might happen: either we end up in a state which has all *None* values so the same happens as does initially. Or we end up in a state which has a Q value for $Q(state, left)$. Now because we only have a value for left, the code will always pick that action.

In short it does not do any exploration, and will only go to left-actions or fall back to a default action (left). This is just as bad as random actions, although the random actions will eventually reach the destination, both will violate many traffic rules.

4 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?*

I played around with the gamma and alpha values first, with an epsilon of 0.5. This gave me some marginal improvements but nothing serious. Next I split up the runs: A simulation of 100 trials with an epsilon of 0.8 and a simulation of 100 trials with an epsilon of 0. This gives good results. Nearly all trials follow the optimal route. I tweaked the alpha and gamma values again, this time I noticed you want gamma to be low for this problem. I assume this is because you don't really need to look ahead, as long as you follow the planner and don't make any traffic violations the smartcab does really well. Alpha did not really make much of a difference as long as it's not too close to 1.

What I did notice is that when the smartcab does make a mistake it is on an intersection with other traffic. I figured this might be because during training it doesn't see a lot of other traffice because there's only 3 cars. When I changed the environment to 10 cars. It can handle traffic a lot better.

I ended up with the following:

- epsilon at 0.8 for 'training and 0 for test
- alpha at 0.2
- gamma at 0.01

This gives me between 0 and 4 negative rewards in all 100 trials (100 * 15 = 1500 rewards received)

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 it does get close to an optimal policy. It had between 0 and 4 negative rewards, which means at least 95 of the trials go optimally. I say optimally because an optimal policy would mean: Don't make any traffic violations, always go the direction the planner indicates you should go. So with almost all rewards being zero or positive this means no traffic violations and following the planner. (or waiting for traffic light). So that's pretty good!