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 have chosen to use: light, direction, oncoming, left. I combine these to a string for ease of use in python. For example: 'green-left-None-right'. I chose to omit the deadline because I think it would lead to unfavourable behaviour plus a lot more unnecessary states. It will lead to unfavourable because it might actually favour making it to the destination above following traffic rules. (Not unlike some human behaviour, but not really something we want our smartcab to learn).

It also introduces a lot of states we don't actually need, when you think about it: We're trying to learn the smartcab to obey the traffic rules and follow the planner. It does't really matter if we're at the beginning of the trip or at the end of the trip. It should still obey the rules, and follow the planner.

The state can also ignore the traffic from the right, because either the lights are green and it's irrelevant, or the light is red, and we use the right-of-way rule and they have to yield.

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 red/green option, the smartcab's direction, oncoming traffic's direction, left's direction.. The number of states are: $2 * 4 * 4 * 4 = 128$. I don't believe the planner actually uses the None direction. So in reality it's: $2 * 3 * 4 * 4 = 96$

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! (even more of an improvement if you were to add deadline)

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 used gamma = 1, learning rate = 1, epsilon = 0 (no exploration). Epsilon is defined as the chance to take a random action.

This 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. A lot of trials follow the optimal route. I tweaked the alpha and gamma values again using gridsearch, 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.

One thing 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 used gridsearch to generate this table for the exact results, in this table I used epsilon of 0.8 on the first 100 trials and 0 and the second 100 trials. And 10 other cars in the environment. The numbers mean (wrong actions, unsuccesful trials)

$\alpha \downarrow$ - $\gamma \rightarrow$	0.1	.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
0.0	(18, 0)	(27, 8)	(12, 1)	(19, 0)	(12, 1)	(23, 0)	(20, 0)	(26, 1)	(23, 3)	(25, 0)
0.1	(29, 1)	(13, 0)	(12, 0)	(32, 0)	(29, 0)	(25, 2)	(27, 9)	(21, 0)	(31, 0)	(33, 0)
0.2	(12, 0)	(10, 0)	(23, 0)	(10, 1)	(31, 1)	(13, 2)	(19, 0)	(23, 2)	(22, 0)	(32, 1)
0.3	(16, 0)	(17, 0)	(17, 0)	(31, 0)	(24, 1)	(7, 0)	(17, 1)	(25, 1)	(35, 0)	(25, 0)
0.4	(35, 1)	(25, 0)	(9, 0)	(24, 1)	(27, 0)	(21, 0)	(9, 0)	(17, 1)	(13, 0)	(12, 0)
0.5	(30, 1)	(3, 0)	(52, 1)	(17, 0)	(15, 0)	(15, 0)	(28, 0)	(11, 0)	(31, 1)	(24, 0)
0.6	(27, 1)	(8, 0)	(6, 1)	(29, 1)	(17, 0)	(21, 0)	(19, 0)	(24, 1)	(20, 1)	(27, 0)
0.7	(19, 1)	(25, 2)	(9, 0)	(14, 0)	(10, 0)	(20, 0)	(15, 0)	(16, 0)	(27, 1)	(12, 0)
0.8	(27, 0)	(21, 0)	(24, 0)	(20, 0)	(37, 1)	(27, 0)	(25, 0)	(23, 0)	(13, 1)	(21, 0)
0.9	(16, 0)	(8, 0)	(40, 0)	(15, 1)	(17, 1)	(19, 0)	(19, 1)	(9, 0)	(28, 12)	(18, 0)

I ended up with the following:

- epsilon at 0.8 for 'training and 0 for test

- alpha at 0.5
- gamma at 0.2

This gives me 3 invalid actions and 0 unsuccesful trials.

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 only 3 invalid actions, which means at least 97% 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!