Implement a Basic Driving Agent

Q: 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?

A:

The smartcab is not smart enough and does not make it to the destination almost all the time. Sometimes it can arrive at the destination luckily with its random actions.

The smartcab doesn't wait patiently at intersections and often goes away from the destination. That is natural because the smartcab chooses his action randomly from the 4 choices i.e. None, forward, left and right, and the chance of choosing None and doing nothing and waiting is less than that of choosing the others and making moves.

Inform the Driving Agent

Q: 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?

A:

I identified that the states below are appropriate for modeling the smartcab.

Next waypoint information from the route planner:

My smartcab can learn the direction it should go from this information. Without this information, it would not know the right direction to make it to its destination.

Traffic light:

With the information of the next waypoint, the smartcab can approach the destination. But it must obey the traffic rules. So it also needs the information of traffic light (red or green).

Oncoming traffic, **Left traffic**:

To avoid accidents with other cabs, it should care oncoming traffics and left traffics. When it wants to make left turn at intersections, it must be sure that there is no oncoming traffic. Similary, when it wants to make right turn, it must be sure that oncoming traffic is approaching from its left through the intersection. I think it doesn’t need to care the oncoming traffic approaching from its right. That’s because there is no chance of my smartcab crashing into its right traffic unless it violates the traffic rules and goes through the intersections with the traffic light red.

I didn’t choose the **deadline variable** as an appropriate state. The raw deadline variable has many state like from 20 to 1. I thought including it would cause the agent not to learn well because it would have too many states to explore and handle and therefore it wouldn’t be able to learn each situation well in a given time. Actually, the agent succeeded less when it considered the deadline.

Implement a Q-Learning Driving Agent

Q: 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?

A:

Compared to the basic driving agent that always takes random actions, the agent with Q-Learning gradually learns how to make it to the destination. At first several trials, it doesn't know which way to go and takes takes almost random actions. It can't even approach the destination appropriately, much less arrive there. But as the trial goes on, it learns how to approach the destination and finally becomes to be able to arrive there.

Q-Learning makes this change in the agent's behavior. The agent learns the state it is in currently and the reward it takes from its action, and becomes to be able to select the appropriate action to get the maximum reward.

Improve the Q-Learning Driving Agent

Q: 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?

A:

I tried different values for the parameters alpha, gamma and epsilon and created the table below. The columns from 1st to 10th represent the number of simulations (, each of which consist of 100 trials) and the each value represents how many time the agent was able to get to the destination. For example, with alpha = 0.1, gamma = 0.9 and epsilon = 0.1, the agent was able to reach the destination 52 times in the 1st simulation. Summing up the values from 1st to 10th, I got the average number of success 69.6 for alpha = 0.1, gamma = 0.9 and epsilon = 0.1.

alpha	gamma	epsilon	avg. success	simulation:	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
0.1	0.9	0.1	69.6		52	80	89	86	25	79	83	85	84	33
0.2	0.1	0.1	85.6		78	89	88	87	81	91	85	92	79	86
0.2	0.2	0.1	85.7		90	74	89	90	87	88	94	66	85	94
0.2	0.3	0.1	81.2		89	93	81	55	85	90	79	70	81	89
0.2	0.4	0.1	82.5		92	79	80	75	86	85	86	82	86	74
0.2	0.5	0.1	78.8		85	84	84	75	83	56	78	78	86	79
0.2	0.8	0.1	63		51	67	76	60	57	45	40	78	77	79
0.2	0.9	0.1	76.1		77	80	74	84	73	72	85	79	54	83
0.3	0.9	0.1	74.2		79	71	85	48	78	80	71	79	76	75
0.4	0.9	0.1	62.4		66	63	65	47	72	64	48	65	64	70
0.5	0.9	0.1	66.2		65	74	58	53	66	73	69	56	70	78
0.5	0.9	0.2	58.7		61	63	56	61	56	62	47	61	63	57
0.5	0.9	0.3	51.5		61	47	44	55	47	39	58	49	55	60
0.5	0.9	0.5	41.5		45	30	42	47	36	40	46	38	48	43
0.6	0.9	0.1	63.6		64	64	59	67	62	76	65	54	63	62
0.9	0.9	0.5	36.7		40	40	37	31	34	42	31	39	25	48

The set of (Alpha = 0.2, gamma = 0.2, epsilon = 0.1) gave the agent the best result (the average number of success is 85.7).

I'm particularly convinced that the low epsilon values brought the best result. With high epsilons, the agent takes random actions too much and cannot exploit what it learned. Not able to learn from its experience, the agent has little chance of reaching the destination.

Q: 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?

A:

My agent gets close to finding an optimal policy pretty much, but strictly speaking it doesn't. Sometimes it can't find an optimal policy and ends up not reaching the destination, especially in long journey trials. It sometimes doesn't patiently wait at the intersection and takes an action which is valid and doesn't violate the traffic rules but isn't the best one. Or sometimes it takes valid but not optimal action and moves away the destination, and then takes an optimal action and moves towards the destinaion. These kinds of behaviors can happen because rewards for actions which are valid but not optimal have relatively small absolute values compared to optimal actions. The agent can have a positive cumulative reward by repeating a bevavior such as moving a little away from the destination and then moving towards there.

I think an optimal policy for this problem is appropriately waiting at intersections not moving around wastefully and only taking an action which brings the agent directly towards the destination. To be able to find an optimal policy, the agent might need less rewards for actions which are valid but not optimal and appropriate punishments. It can avoid the wastefull actions and only choose actions directly towards the destinaion.