Report Smartcab Project

1. Question

What I observe:

- Deadline goes negative
- The times the car achieves the destination was in average 15 times, from 30 trials I tried
- Very often the hard limit of -100 occurred
- The majority of inputs are almost always None, which indicates low traffic, rewards are randomly between -1 and 2

2. Question

I chose the inputs and waypoints to define the different states of the problem as follows:

waypoints = ['forward', 'left', 'right']
light = ['red','green']
oncoming = [None, 'left', 'right', 'forward']
left = [None, 'left', 'right', 'forward']

As other cars can also perform actions at a signal, the values from oncoming, right and left are basically the same as the valid actions from our agent. The reason for not using 'right' as to define my states, are the american rules for driving. If you have a red light you can still turn right considering there is no car coming from the 'left', the 'right' on the other hand is irrelevant for both red and green lights.

In total there are 96 combinations from the values above. This seems to be a good number, because we will have only 100 attempts, exploring all the states and at the same time updating all (or at least some) 4 times for each action seems plausible. Moreover, looking at the states and inputs, it seems that there is very low traffic (other vehicles inputs are very often all None), which means that many of the states might occur very rarely and might not be even visited when using the q_table later.

I decided to leave the deadline out because then the states would be too many. For states with a starting deadline of 50, this makes the number of states explode to 19.200 which for 100 trials is just way too much. On top of that, I wouldn't want my car to take risky actions when time is running down, as for my a good policy is the one that puts safety above all

3. Question

After implementing the Q-learning formula as following

(Q(s,a) = (1-alpha) * Q(s,a) + alpha * (reward + gamma * max(Q(s',a'))),

my car reached in average the destination in 48 out of 100 trials in 30 times I run the program. In comparison, when choosing a random action after running the programm 30 times I got an average of 15 times that the car reached its destination. This means, that indeed Q-learning did indeed work substantially better than random guessing (Note alpha=0.5 and gamma=0.5).

The reason for this improvement, is that the action is now informed by the Q value of the actions at a given state. These include rewards and maximum Q-value of all possible actions at that state. As I initialized Q-values at 0, every time a new reward is used to update the Q-value, the next time the car reaches that state, he will have a value for the previous action he had taken at that same state. Thus if the action was good it might take it again if it was negative it might take another one with value of 0 for instance. This rewards start propagating back and the more the q_table is updated the more we know about each long term reward for each state action pair.

An important observation was that in the trials, where the car achieved a high number of successes (made it to destination), the final trials were almost always successful and needed always less steps to get to the destination than trials in the middle and the beginning.

4. Question

See next page with results. Column painted red represents best combinations of parameters.

As a success metric I took the times the car reached its destination in the last 10 trials. In the most optimal combination of parameters the smartcab achieves its goal (the destination) in average 7.1 times in the last trials. A Bad move is counted as one with a reward of -1. In average the best results only got in average 0.63 bad moves per last 10 trials.

I think an optimal policy for this problem is one, that doesn't have invalid moves in it, like going left or forward on a red light and puts the safety of the driver first. A second factor would be a policy that gets fast to the destination, but always avoiding invalid moves above all. I think my smartcab achieved an optimal policy in the majority of the cases as it very rarely commits an invalid move.
This being said I would expect my smartcab to first have very low moves, which it did, by having only 0.63 per 10 trials at its best. Secondly, with those same parameter values it achieved 7.1 destinations reached out of 10, which with a threshold of 0.7 passes the test and achieves an optimal policy. Moreover if you see, the smartcab actually achieved a better 'destination reached' average with other parameters (see run 12 in table) but a much worse bad_moves average. As I prefer my car to be secured I think the policy of run number 11 fits best the one described above.

Exerpiment Nr.	1		2		3		4		5		6		7		8		9		10		11		12	
alpha	0.5		0.5		0.5		0.4		0.9		0.4		0.2		0.35		0.6		0.7		0.7		0.7	
gamma	0.5		0.75		0.6		0.6		0.9		0.9		0.7		0.65		0.6		0.6		0.6		0.55	
epsilon	0		0		0		0		0		0.001		0.001		0.001		0.001		0.001		0.001		0.001	
	Destination reached	Absolute Bad Actions	Times Destination reached	Bad Actions	Times Destination reached	Bad Actions	Times Destination reached	Bad Actions	Times Destination reached	Bad Actions	Times Destination reached	Bad Actions	Times Destination reached	Bad Actions	Times Destination reached	Bad Actions	Times Destination reached	Bad Actions	Times Destination reached	Bad Actions	Times Destination reached	Bad Actions	Times Destination reached	Bad Actions
1	4	3	1	2	9	0	5	1	9	0	6	64	2	1	1	3	8	1	7	23	9	0	9	1
2	7	1	3	2	9	0	5	1	9	0	6	54	2	1	8	1	5	1	6	1	10	3	9	0
3	6	0	8	0	3	0	9	1	6	39	6	1	4	9	3	2	9	0	1	19	10	0	6	0
4	3	1	6	2	7	0	8	0	9	1	6	1	2	0	5	0	6	1	5	1	4	2	10	0
5	6	0	7	1	3	0	5	2	3	0	8	1	4	1	5	1	5	0	6	2	3	1	7	3
6	10	0	3	3	1	0	6	1	10	2	6	30	3	0	8	3	9	0	5	0	6	3	3	22
7	4	2	6	2	2	20	8	0	9	0	8	0	7	0	2	0	8	0	9	0	6	0	1	5
8	6	0	0	0	7	2	9	2	9	0	3	3	0	4	3	0	6	0	5	1	6	0	10	0
9	5	2	7	12	6	0	9	1	7	0	6	37	3	0	5	3	10	0	5	1	2	2	4	2
10	9	0	7	57	3	0	7	0	8	0	4	0	7	1	4	1	6	1	8	1	9	0	10	1
11	9	1	8	0	9	2	2	1	6	29	6	0	6	2	0	4	4	3	6	0	7	1	9	2
12	7	1	1	0	8	0	9	0	6	0	1	2	3	2	8	4	4	0	4	1	10	0	8	0
13	7	0	8	1	4	1	2	0	5	0	4	37	2	1	1	3	5	1	2	0	7	0	3	0
14	9	1	10	1	7	1	4	0	9	1	5	0	2	1	6	0	8	0	2	4	8	1	9	0
15	0	0	1	1	5	0	6	0	7	27	2	1	6	45	4	0	8	0	8	1	2	1	8	0
16	9	0	7	1	4	3	3	1	5	94	2	1	9	42	9	1	10	3	5	3	9	0	7	1
17	7	1	9	1	7	2	9	9	5	0	4	2	7	0	3	1	10	1	8	1	10	0	9	0
18	0	3	2	1	7	0	9	0	9	0	4	0	8	1	2	2	0	0	5	0	7	0	8	3
19	9	0	10	0	10	0	9	0	0	1	4	0	5	0	5	3	9	0	3	1	7	0	8	2
20	5	1	2	1	4	2	10	1	6	2	2	2	10	0	10	0	8	0	5	0	10	0	5	1
21	10	0	7	0	6	0	10	0	4	0	6	49	7	0	7	2	5	2	8	0	9	0	9	4
22	9	0	4	0	4	0	4	0	10	0	10	14	2	2	10	0	9	0	6	0	4	1	5	0
23	6	0	7	0	10	0	7	0	3	0	1	2	9	0	7	1	2	2	9	1	6	0	9	1
24	7	0	3	1	8	20	6	0	7	1	4	0	9	0	5	0	7	0	8	2	10	0	9	12
25	3	0	7	3	9	0	6	1	6	1	10	77	1	3	5	0	6	1	10	1	9	1	9	3
26	7	2	6	0	3	1	3	1	7	48	6	1	3	0	6	2	8	2	9	0	9	1	0	5
27	10	2	7	0	7	1	0	3	2	1	6	4	3	1	8	0	5	2	8	1	10	0	8	0
28	7	0	3	0	4	0	8	0	6	34	9	81	5	1	5	2	8	1	10	0	4	0	10	3
29	4	1	5	2	9	0	8	0	6	0	6	0	3	0	5	1	7	0	10	19	4	1	7	0
30	8	1	9	0	9	0	5	2	2	1	8	0	1	2	6	1	8	0	9	0	5	1	7	1
Total	193	23	164	94	184	55	191	28	190	282	159	464	8	0	156	41	203	22	192	84	212	19	216	72
Median	7.00	0.5	6.5	1	7	0	6.5	0.5	6	0.5	6	1.5	3.5	1	5	1	7.5	0	6	1	7	0	8	1
Average	6.433	0.767	5.467	3.133	6.133	1.833	6.367	0.933	6.333	9.400	5.300	15.467	4.500	4.000	5.200	1.367	6.767	0.733	6.400	2.800	7.067	0.633	7.200	2.400