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']
right = [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.

In total there are 384 combinations from the values above. This seems to be a lot at first, because we will have only 100 attempts, exploring all the states and at the same time updating all 4 times for each action seems difficult. Nevertheless, 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.

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.

In the most optimal combination of parameters the smartcab achieves its goal (the destination) in 55% of the time. Very rarely it got a reward of -1 for an invalid move and more often it got -0.5 for not going into the direction of the next way_point.

I think an optimal policy for this problem is one, that doesn't have invalid moves in it, like going left or forward ona 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.

Exerpiment Nr.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
alpha	0.5	0.75	0.5	0.5	0.25	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.25	0.4	0.45
gamma	0.5	0.5	0.75	0.5	0.5	0.25	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.9	0.75	0.6	0.6	0.6	0.6
epsilon	0.5	0.5	0.5	0.75	0.5	0.5	0.25	0	0.1	0.05	0.025	0.01	0.001	0.001	0.001	0.001	0.001	0.001	0.001
1	21	26	46	6	50	18	28	92	35	48	60	35	27	56	53	63	20	42	26
2	51	25	51	24	53	52	33	88	43	71	23	57	75	25	71	76	82	57	33
3	23	21	39	31	50	35	54	83	21	74	72	57	18	25	56	31	12	57	85
4	24	26	28	43	49	29	49	35	51	41	57	22	65	37	58	51	10	60	66
5	36	38	59	20	16	5	45	31	35	61	90	34	40	65	52	36	28	59	87
6	18	28	27	39	25	37	49	48	29	18	70	29	66	42	39	47	51	26	48
7	53	33	30	29	41	23	31	25	31	34	73	41	84	16	62	54	19	56	34
8	28	42	34	21	43	29	52	60	47	39	67	53	47	65	64	78	64	20	34
9	24	30	39	39	34	25	30	83	44	57	44	24	48	44	29	25	59	59	48
10	43	20	42	23	52	24	63	40	64	26	63	84	67	24	68	75	7	37	46
11	54	17	42	19	67	53	39	76	63	57	13	24	33	90	36	41	80	26	73
12	13	33	48	29	38	24	21	37	26	54	70	35	61	12	55	39	84	55	26
13	13	34	30	36	37	32	33	10	33	48	2	31	67	42	63	51	49	45	57
14	15	21	40	28	35	27	29	92	23	62	71	85	80	37	52	30	51	2	27
15	37	35	32	31	24	35	44	56	79	34	46	31	82	88	88	19	17	62	63
16	24	16	24	31	30	14	21	47	65	26	36	31	64	15	58	85	23	37	82
17	38	26	13	36	54	11	46	43	59	52	42	23	22	62	18	29	27	56	50
18	27	36	26	30	7	29	36	27	42	43	49	37	82	49	79	93	33	29	68
19	19	22	22	21	43	38	28	84	11	32	62	44	49	25	60	88	9	35	10
20	39	29	41	22	37	26	43	81	65	45	41	24	53	32	8	57	52	42	83
21	45	34	46	28	45	38	24	64	25	28	66	24	28	9	59	81	10	59	87
22	25	24	48	23	22	25	48	64	38	56	36	85	79	52	62	68	0	32	65
23	48	31	20	14	31	25	10	22	52	74	47	52	75	16	7	37	55	62	34
24	51	40	18	25	44	29	63	8	27	63	36	78	33	73	40	76	76	15	22
25	35	18	31	32	24	6	36	32	49	33	73	67	66	56	87	56	62	62	58
26	43	19	51	55	40	29	39	15	43	66	39	44	48	46	54	13	58	50	73
27	36	24	27	22	11	25	36	11	43	25	11	44	57	18	70	65	52	25	82
28	39	28	24	17	60	20	32	49	53	60	42	23	9	84	15	64	17	40	66
29	53	40	32	23	45	37	23	57	28	14	41	76	18	62	59	61	62	27	53
30	36	38	38	21	46	10	28	67	57	62	32	49	90	45	52	57	78	56	52
Total	1011	854	1048	818	1153	810	1113	1527	1281	1403	1474	1343	1633	1312	1574	1646	1247	1290	1638
Median	36	28	33	26.5	40.5	26.5	36	48.5	43	48	46.5	39	59	43	57	56.5	50	43.5	55
Average	33.700	28.467	34.933	27.267	38.433	27.000	37.100	50.900	42.700	46.767	49.133	44.767	54.433	43.733	52.467	54.867	41.567	43.000	54.600