Project 4 - self-driving agent

Reinforcement learning

17 September 2016

QUESTION: Observe what you see with the agent's behaviour as it takes random actions. Does the **smartcab** eventually make it to the destination? Are there any other interesting observations to note?

No the smart cabs doesn't make it to destination most of the time (there are some lucky exceptions). As it takes a random action, the cab doesn't take into account the "next waypoint" information.

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?

The states I used to represent the smartcab and its environment are a combination of the light, the incoming/left traffic and the next waypoint : **(light, oncoming, left, next_waypoint)**
Considering the **light** variable is primordial, the agent needs to learn to stop at red lights. It is the main variable of the environment.
As for the traffic, I chose to take into account only the **incoming** and the **left** traffic. Incoming traffic is important : when the agent needs to turn left, it must yield the way for incoming cabs going forward and turning left. The left traffic is also important for the right turns on red. The way needs to be clear, this is checked by looking at the traffic coming from the left.
Finally we need to include the **next waypoint**, this variable has two purposes. It gives the smart cab the action that brings it closer to the destination. The next waypoint is also necessary in case of oncoming traffic and the agent needs to turn left. The agent must learn to stop and let the oncoming traffic cross the road.
I didn't include the deadline in the states because it would create way to much states. For each deadline value we would have 94 possible combinations. The Q Learning would not work in this case (or it will take way more than 100 trials). The agent needs to visit the states multiple times to compute the q values.

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?

The enumeration of the states goes as follow :

- The traffic light has 2 values : red or green
- At each intersection there is 2 variables that take a value among 4 : None, right, left, forward
- The next waypoint is a single variable and take a value among 3 : right, left, forward

Therefore we have $2 \times 4^2 \times 3 = 94$ possible combinations : the total number of possible states.

This number seems reasonable. The goal of Q Learning is to compute the q values of each states. Those values needs to be precise enough and different from each other for the agent to make a decision.
Hence in the learning phase, the agent needs to visit several time the states. If the space of the states is too big, the training will either take a lot of time or the agent will make random decision because it didn’t visit enough states in the training phase.

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?

Selecting the best action makes the agent reach its destination and this without committing any infringement. When the agent was taking random actions it didn’t reach its destination (except few lucky times) and it didn't care about the rules.
However this is not an immediate result, the agent first go through a learning phase.
In the first trials the cab needs many steps to reach the destination, even fail to do so sometimes. It also makes a lot of infringements as represented by the negative rewards.
Indeed, at first the agent doesn’t know any states, its Q matrix is empty (initialized with ones). At each state, the agent takes the best action i.e. the action whose q-value is the highest. If there are several action with the max q-value, it takes a random action within this subset.
Once it has taken an action, the reward for this very action is computed and the q-value of the matrix is updated. At first, most of the action taken are chosen randomly, this is the training phase where the agent explore a lot (since all the q value are equal to 1). After several trials, the q-values are more precise because the agent visited the state several times. In the end of the session (arround trial 80), the agent has already visited most of the state several times. For each state, one q-value is a lot higher than the other.
Therefore it can move from point A to point B without any infringement.

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 ran an exhaustive search over the parameters **ε** the exploration probability,**α** the learning rate coefficient and **γ** the discount coefficient. The corresponding function is called **run_exhaustive**. It basically runs the simulation five times for each set of parameters then averages on both the overall number of successful trials and the number of successful trials without -1.0 rewards in the last twenty trials. The output is written in the **gridsearch_results.txt** file. Finally after some text parsing we are able to retrieve the best parameters :

ε = 0, α = 0.9, γ = 0.2

Gridsearch results sample

	epsilon	alpha	gamma	avgresult	avgresult_p
0	0.0	0.1	0.1	99.2	18.4
1	0.0	0.1	0.2	98.4	16.8
2	0.0	0.1	0.4	96.2	18.4
…					
29	0.0	0.8	0.9	87.6	16.2
30	0.0	0.9	0.1	99.6	19.4
31	0.0	0.9	0.2	100.0	19.2
32	0.0	0.9	0.4	99.2	19.4
33	0.0	0.9	0.6	99.4	19.0

The full table is available at the end of the report.

The final agent performs quite well, it always reaches the destination before the deadline. We can also notice in the last twenty trials that there aren't many steps with a **-1.0** reward. In fact for the optimal combination of hyper parameters, out of the twenty last trials the agent reaches its destination 19 times without any driving violation.

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?

Regarding the smartcab problem, the optimal policy is found when the agent reaches the destination without any penalties and with the minimum number of steps. There are two types of penalties : **-0.5** penalties occurs when the agent takes an non-optimal action (for instance when the agent turn right instead of going forward). The **-1.0** penalty is the result of driving infraction.
Looking at the command prompt output, in the last trials there are only few to no steps with **-0.5** rewards. Therefore we could argue that the agent comes close to finding the optimal policy.

Below is a sample of the command prompt output where I printed the Q row of the state, the action taken and the reward for that action.

Q[state]	action	reward
{'forward': -0.6200000000000001, 'right': 1.0, None: 1.0, 'left': 1.0}	**right**	**-0.5**
{'forward': -0.17000000000000004, 'right': 2.4320970793835492, None: 1.0, 'left': 1.0}	**right**	**2.0**
{'forward': -0.6200000000000001, 'right': 2.040376671665892, None: 1.0, 'left': 1.0}	**right**	**2.0**

We can notice that in the case of the step with a -0.5 reward, the row of the state is not complete, there still are 3 values set to 1.0 (the initial value). The -0.5 steps highlight the fact that the agent hasn't visited each state enough times, it still requires further exploration to be fully optimal.

Gridsearch results

	epsilon	alpha	gamma	avgresult	avgresult_p
0	0.0	0.1	0.1	99.2	18.4
1	0.0	0.1	0.2	98.4	16.8
2	0.0	0.1	0.4	96.2	18.4
3	0.0	0.1	0.6	93.6	18.0
4	0.0	0.1	0.8	88.6	16.8
5	0.0	0.1	0.9	96.6	18.8
6	0.0	0.2	0.1	99.2	19.6
7	0.0	0.2	0.2	98.4	19.2
8	0.0	0.2	0.4	97.8	19.0
9	0.0	0.2	0.6	96.2	17.8
10	0.0	0.2	0.8	94.4	19.0
11	0.0	0.2	0.9	79.0	14.2
12	0.0	0.4	0.1	99.2	18.8
13	0.0	0.4	0.2	99.2	18.4
14	0.0	0.4	0.4	98.8	18.2
15	0.0	0.4	0.6	94.4	18.0
16	0.0	0.4	0.8	89.8	17.2
17	0.0	0.4	0.9	72.8	15.0
18	0.0	0.6	0.1	99.4	19.8
19	0.0	0.6	0.2	99.4	19.0
20	0.0	0.6	0.4	99.0	18.4
21	0.0	0.6	0.6	95.2	18.8
22	0.0	0.6	0.8	81.4	15.8
23	0.0	0.6	0.9	76.2	15.8
24	0.0	0.8	0.1	99.2	19.4
25	0.0	0.8	0.2	99.8	19.2
26	0.0	0.8	0.4	98.8	19.4
27	0.0	0.8	0.6	98.8	19.0
28	0.0	0.8	0.8	90.8	19.0
29	0.0	0.8	0.9	87.6	16.2
30	0.0	0.9	0.1	99.6	19.4
31	0.0	0.9	0.2	100.0	19.2
32	0.0	0.9	0.4	99.2	19.4
33	0.0	0.9	0.6	99.4	19.0
34	0.0	0.9	0.8	86.2	16.6
35	0.0	0.9	0.9	73.6	15.2
36	0.1	0.1	0.1	96.6	12.2
37	0.1	0.1	0.2	97.0	13.2
38	0.1	0.1	0.4	87.2	11.2
39	0.1	0.1	0.6	88.2	11.0
40	0.1	0.1	0.8	88.8	12.0
41	0.1	0.1	0.9	91.8	14.6
42	0.1	0.2	0.1	98.0	12.4
43	0.1	0.2	0.2	96.8	11.4
44	0.1	0.2	0.4	91.2	14.2
45	0.1	0.2	0.6	87.4	12.0
46	0.1	0.2	0.8	74.8	11.6
47	0.1	0.2	0.9	75.6	10.8
48	0.1	0.4	0.1	98.0	12.6
49	0.1	0.4	0.2	97.4	12.6
50	0.1	0.4	0.4	95.8	13.4
51	0.1	0.4	0.6	89.8	14.4
52	0.1	0.4	0.8	81.2	12.2
53	0.1	0.4	0.9	78.6	10.0
54	0.1	0.6	0.1	97.6	13.6
55	0.1	0.6	0.2	97.2	12.6
56	0.1	0.6	0.4	95.4	13.0
57	0.1	0.6	0.6	87.2	14.4
58	0.1	0.6	0.8	76.2	11.8
59	0.1	0.6	0.9	72.2	11.0
60	0.1	0.8	0.1	97.0	12.8
61	0.1	0.8	0.2	98.4	12.4
62	0.1	0.8	0.4	95.8	14.0
63	0.1	0.8	0.6	88.2	14.0
64	0.1	0.8	0.8	85.2	12.0
65	0.1	0.8	0.9	72.4	10.0
66	0.1	0.9	0.1	97.0	12.8
67	0.1	0.9	0.2	98.4	12.2
68	0.1	0.9	0.4	95.6	12.4
69	0.1	0.9	0.6	89.4	11.4
70	0.1	0.9	0.8	80.8	10.2
71	0.1	0.9	0.9	66.2	9.2
72	0.2	0.1	0.1	93.0	8.0
73	0.2	0.1	0.2	93.6	8.6
74	0.2	0.1	0.4	83.2	7.2
75	0.2	0.1	0.6	82.6	9.6
76	0.2	0.1	0.8	82.4	10.2
77	0.2	0.1	0.9	75.2	9.0
78	0.2	0.2	0.1	93.2	8.4
79	0.2	0.2	0.2	93.8	8.2
80	0.2	0.2	0.4	84.8	11.4
81	0.2	0.2	0.6	81.4	8.6
82	0.2	0.2	0.8	81.4	7.0
83	0.2	0.2	0.9	64.4	7.2
84	0.2	0.4	0.1	94.6	9.8
85	0.2	0.4	0.2	95.2	8.8
86	0.2	0.4	0.4	88.2	7.6
87	0.2	0.4	0.6	84.0	9.4
88	0.2	0.4	0.8	78.8	9.0
89	0.2	0.4	0.9	69.2	8.0
90	0.2	0.6	0.1	95.0	7.4
91	0.2	0.6	0.2	92.8	10.6
92	0.2	0.6	0.4	92.0	9.2
93	0.2	0.6	0.6	82.2	7.2
94	0.2	0.6	0.8	79.6	8.2
95	0.2	0.6	0.9	67.0	6.6
96	0.2	0.8	0.1	95.2	8.6
97	0.2	0.8	0.2	93.0	7.8
98	0.2	0.8	0.4	90.6	9.6
99	0.2	0.8	0.6	82.0	10.2
100	0.2	0.8	0.8	72.6	6.2
101	0.2	0.8	0.9	66.2	5.8
102	0.2	0.9	0.1	93.6	10.0
103	0.2	0.9	0.2	91.6	7.4
104	0.2	0.9	0.4	91.8	10.0
105	0.2	0.9	0.6	83.2	9.2
106	0.2	0.9	0.8	70.6	6.6
107	0.2	0.9	0.9	63.6	5.6