MLND P4: Train a Smartcab to Drive

Implement a basic driver agent

Question: The agent moves randomly across the city, while it's possible it may reach the destination by chance it did not occur while I was watching. The rewards oscillate around 0, the average seems to be positive as the range is [-1,2]

Inform the driving agent

Question: At every intersection, the useful information the agent has access to is the state of the traffic light (green/red), traffic conditions (other cars in the intersections) and the next way point hint. These information are useful because they all affect the reward of an action.
So the state is {light,car_right,car_left,car_oncoming,next_waypoint}

Which is 2x2x2x2x4 states = 64 states

I did not include the deadline for 2 reasons. First, it would multiply the number of states by a big number, making it difficult to visit all the states and reach a convergence within 100 trials. Second, including the deadline could bias decisions and incite illegal moves: if the agents only have a couple of moves left and the only way to get to the destination on time is to ignore a red light, the agent could decide to do that

Implement a Q-Learning driving agent

Question: At the beginning actions still look random (as they are), but we can notice the agent learns to listen to the waypoint hint and not to cross intersections where traffic light is red or when there's a car in the direction it wants to go. With time it goes faster and can reach it's destination very fast. However there are situations where the agent is blocked in a loop where it keeps on turning right when the light is red and the waypoint is in another direction. The matrix gets biased by the fact that staying put is less rewarding than an immediate legal move in the wrong direction.
I observed two other blocking situations but for which I cannot rest provide an intuitive explanation:

- sometimes the agent remains blocked at an intersection until the games end, even if the light is green and free from other cars
- sometimes the agent keeps on going straight and crossing the board until the game ends while it could turn in the direction of the destination when it crosses the lane where the destination is.

However those situations stop occurring at some point. The agent becomes efficient and usually takes the shortest path to the destination while stopping when the light is red and avoiding crashes. That being said, the agent has occasionally a bad streak at the beginning and seems to stay stuck: it keeps failing to get to the destination even after 100 trials.

Enhance driving agent

My approach is to run a kind of grid search over γ, ε. The learning rate is 1/t, I have explored using powers of t but it clearly degrades results and is not worth including in the simulation.

ε decreases over time to reflect that the more the agent learns, the more it can exploit and the less it needs to explore. There's a simple rule here: ε decreases linearly at the rate of 1/1000 per action.

I'll test all values of
γ between 0 and 0.5 with 0.1 steps
ε between 0 and 0.5 with 0.05 steps

To do that I slightly modified environment.py so it stores a **result** variable in the form of a tuple (outcome [0,1], score, total penalties). I access this **result** from the run() method of agent.py .

For each combination I run the simulator 10 times, average the results and look at the 10 last instances of the averaged results: If a policy is found, a convergence should be found at least after the 90th trial.

The table below summerizes the best results. With γ = 0.4 and ε = 0.05, the destination is reached 99% of the time (every combination is tried 10 times and the results are then averaged. With γ=0 and ε = 0, the destination is only(!) reached 98% of the time but average penalty is also slightly lower.

The full table can be found here:
https://docs.google.com/spreadsheets/d/1uhY4MCMy8WGwTSLVOfRponRN_dF88SsT1ZcAkvj0TEs/edit?usp=sharing

γ	ε	avg_penalty_last_10	avg_score_last_10	avg_outcome_last_10
0.4	0.05	-1.50	9.55	0.99
0.25	0.1	-1.09	10.50	0.98
0.35	0.05	-0.64	10.38	0.98
0	0	-0.37	10.94	0.98

The policy tends to be “Follow next waypoint indication but stay put if traffic light is red or pick another direction if another vehicule is coming in the intersection”

Here is the transcript of the 100th trial with γ=0.4 and ε=0.05. (See how this file is generated in agent.py)
The agent follows the policy except when another car enters the intersection. It generally avoids collision but still takes an incorrect action. This is probably because since the agent doesnt encounter that many other vehicles in its journey, it didn’t visit states with incoming traffic enough to form good Q values. I suspect it faces a decision where all Q values are 0 and moves randomly.

	inputs	waypoint	action
0	{'light': 'green', 'oncoming': None, 'right': None, 'left': None}	right	right
1	{'light': 'red', 'oncoming': None, 'right': None, 'left': None}	right	right
2	{'light': 'red', 'oncoming': 'forward', 'right': None, 'left': None}	forward	forward
3	{'light': 'green', 'oncoming': None, 'right': None, 'left': None}	forward	forward
4	{'light': 'green', 'oncoming': None, 'right': None, 'left': 'left'}	forward	right
5	{'light': 'green', 'oncoming': None, 'right': None, 'left': None}	left	left
6	{'light': 'red', 'oncoming': None, 'right': None, 'left': None}	forward	
7	{'light': 'red', 'oncoming': None, 'right': None, 'left': None}	forward	
8	{'light': 'red', 'oncoming': None, 'right': None, 'left': None}	forward	
9	{'light': 'green', 'oncoming': None, 'right': None, 'left': None}	forward	forward
10	{'light': 'red', 'oncoming': None, 'right': None, 'left': None}	forward	
11	{'light': 'red', 'oncoming': None, 'right': None, 'left': None}	forward	
12	{'light': 'red', 'oncoming': None, 'right': None, 'left': None}	forward	
13	{'light': 'red', 'oncoming': None, 'right': None, 'left': None}	forward	
14	{'light': 'red', 'oncoming': None, 'right': None, 'left': None}	forward	
15	{'light': 'green', 'oncoming': None, 'right': None, 'left': None}	forward	forward
16	{'light': 'green', 'oncoming': None, 'right': None, 'left': None}	forward	forward
17	{'light': 'red', 'oncoming': None, 'right': None, 'left': None}	forward	
18	{'light': 'red', 'oncoming': None, 'right': None, 'left': None}	forward	
19	{'light': 'red', 'oncoming': None, 'right': None, 'left': None}	forward	
20	{'light': 'green', 'oncoming': None, 'right': None, 'left': None}	forward	forward
21	{'light': 'green', 'oncoming': None, 'right': None, 'left': None}	right	right