Implement a basic driving agent

Implement the basic driving agent, which processes the following inputs at each time step:

- Next waypoint location, relative to its current location and heading,
- Intersection state (traffic light and presence of cars), and,
- Current deadline value (time steps remaining),

And produces some random move/action (None, 'forward', 'left', 'right'). Don't try to implement the correct strategy! That's exactly what your agent is supposed to learn.

Run this agent within the simulation environment with enforce_deadline set to False (see run function in agent.py), and observe how it performs. In this mode, the agent is given unlimited time to reach the destination. The current state, action taken by your agent and reward/penalty earned are shown in the simulator.

In your report, mention what you see in the agent's behavior. Does it eventually make it to the target location?

You can make the car drive randomly by setting epsilon to 1. The smart cab just drives randomly, but it does eventually make it to its destination. The smart will only complete 20 successful trips out of 100 trials when the deadline is enforced. It earns a total reward of 1506. The total number of moves the smart cab executed was 2663 which making the average move per trip 26.63.

At each time step, process the inputs and update the current state. Run it again (and as often as you need) to observe how the reported state changes through the run.
Justify why you picked these set of states, and how they model the agent and its environment.

My state space consisted of the waypoint and the inputs. It needs the waypoint to navigate to the final destination. It needs the inputs to learn traffic laws. I don't think the deadline is needed since the car cannot drive faster or slower. It will only increase the state space and slows down its learning.

Implement Q-Learning

Implement the Q-Learning algorithm by initializing and updating a table/mapping of Q-values at each time step. Now, instead of randomly selecting an action, pick the best action available from the current state based on Q-values, and return that.
Each action generates a corresponding numeric reward or penalty (which may be zero). Your agent should take this into account when updating Q-values. Run it again, and observe the behavior.
What changes do you notice in the agent's behavior?

You can make the smart cab always pick the best action available by setting epsilon to 0. The smart cab completed 38 successful trips compared to only 20 when it was driving randomly. It earned a total reward of 3006 which was roughly double compared to driving randomly. The total number of moves was 2552 vs 2663. This strategy is much more successful.

Enhance the driving agent

Apply the reinforcement learning techniques you have learnt, and tweak the parameters (e.g. learning rate, discount factor, action selection method, etc.), to improve the performance of your agent. Your goal is to get it to a point so that within 100 trials, the agent is able to learn a feasible policy - i.e. reach the destination within the allotted time, with net reward remaining positive.
Report what changes you made to your basic implementation of Q-Learning to achieve the final version of the agent. How well does it perform?

The smart cab will uses a combination of driving and the best action available according to the q-move. I used an epsilon of 0.10 and an alpha of 0.20. The smart cab was able to complete 80 successful trips and a positive total rewards of 3010.5. It also had the lowest number of moves 1792. This strategy is better than the first two approaches.

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?

The agent is unable to learn an optimal policy of reaching the destination in the minimum possible time without incurring any penalties. There is still room for the agent to learn. It could not explore every possible state action combination within 100 trials. Traffic violations were still appearing at trial #97. The fact that more did not occur after that was most likely due to luck.

The current reward structure still needs to be changed, because it encourage undesirable behaviors. It rewards the smart cab to disobey traffic laws when the time is low and to drive around in circles when the amount of time left is high. The system penalizes rather reward the agent for reaching the destination in least number of moves. The agent only needs to reach the destination with the deadline.

However, the agent does perform fairly well for getting to the destination on time with a positive cumulative reward. This agent outperformed both the random and deterministic agent in having the least number of moves, highest cumulative, and most successfully completed trips.