Train a smartcab to drive

Implement a basic driving agent

Run this agent some random move/action and observe how it performs.

If the random act is chosen, the agent will perform a “random walk on the grid. It can eventually reach the destination, but it can take many steps.

If "next_waypoint" is chosen, the agent will go directly to the target point but doesn’t obey the rules

Identify and update state

Identify a set of states that you think are appropriate for modeling the driving agent. Justify why you picked these set of states, and how they model the agent and its environment.

State include: light, oncoming, and next_waypoint

Light {‘red’, ‘green’}

oncoming{ None, 'forward', 'left', 'right'}

next_waypoint{ 'forward', 'left', 'right' }

1. Light: light determines whether the agent should stop or go forward at an intersection

2. Oncoming traffic: oncoming traffic is relevant because if the oncoming vehicle is going forward, the agent wants to go left. The agent should stop and wait for the oncoming vehicle.

3. Next_waypoint: the next_waypoint is important information because the planer instructs for the agent to reach the destination.

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.

What changes do you notice in the agent’s behavior?

In the beginning, the agent takes many steps to reach the destination and has low rewards. After several iterations, the agent reaches the destination in fewer steps and with higher rewards.

- The Q-table updating rules:

$$\hat{Q}(s,a) \overset{\alpha_t}{\leftarrow} r + \gamma \max_{a'} \hat{Q}(s',a')$$

- Decay learning rate α for convergence:
  Requirements for convergence:

$$\sum_{t=1}^{\infty} \alpha_t = \infty, \sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

  Learning rate update function.

$$\alpha_t = \frac{1}{c + t/b}$$

  Here c is a constant; b is a tuning parameter

- "Simulated annealing" liked approach for choosing action

$$\hat{\pi}(s) = \begin{cases} \text{argmax}\hat{Q}(s,a) & \text{w. p. } 1-\varepsilon \\ \text{random action} & \text{w. p. } \varepsilon \end{cases}$$

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.

Tuning parameter includes:

(1) Learning rate α: in our case, c and b are the tuning learning rate α.When α is close to 0, the agent tends to learn little from the interaction. When α is close to 1, the agent tends to have a "short memory". It reflects more on the recent interactions.
(2) Discount factor γ: the value ranges from 0 to 1. The value more close to 0 the influence of future states are reduced and vice versa.

By trial and error, we find by decreasing γ the agent has better change getting to the destination. The ideal γ is about 0.3 through experiments.

Results

The results of Q-table after 100 trials

light, oncoming, next_waypoint	forward','left' 'right' None
('red', None, 'right')	['5.58', '5.57', '8.45', '7.47']
('green', 'right', 'left')	['1.00', '2.91', '1.93', '1.00']
('green', None, 'forward')	['7.84', '6.61', '6.61', '7.21']
('red', None, 'forward')	['4.20', '4.22', '6.54', '6.21']
('green', 'left', 'left')	['3.36', '1.00', '1.00', '1.00']
('red', 'left', 'left')	['0.78', '0.13', '2.10', '1.00']
('red', 'forward', 'right')	['0.52', '0.52', '7.40', '1.00']
('green', 'forward','right')	['1.00', '1.00', '1.00', '1.32']
('red', 'right', 'left')	['0.66', '1.00', '1.00', '1.00']
('red', 'right', 'forward')	['0.52', '0.54', '1.66', '1.00']
('green', 'forward', 'left')	['1.00', '1.00', '1.00', '1.00']
('green', None, 'right')	['7.03', '6.53', '8.30', '7.59']
('green', None, 'left')	['6.56', '7.39', '6.82', '7.16']
('green', 'left', 'forward')	['7.25', '1.00', '1.00', '1.85']
('green', 'left', 'right')	['6.28', '1.00', '1.00', '2.30']

```
('red', 'forward', 'left')          ['0.20', '0.52', '2.71', '1.44']
('green', 'right', 'right')         ['1.00', '1.00', '1.00', '1.00']
('red', 'right', 'right')           ['1.00', '1.00', '1.00', '1.12']
('red', 'left', 'right')            ['0.66', '0.82', '3.10', '2.21']
('red', None, 'left')               ['4.41', '4.51', '6.89', '6.53']
('red', 'left', 'forward')          ['0.53', '0.73', '2.36', '1.00']
('green', 'right', 'forward')       ['1.00', '1.00', '1.00', '1.99']
('green','forward','forward')       ['3.80', '1.00', '1.00', '1.00']
('red', 'forward', 'forward')       ['0.20', '0.65', '5.28', '1.00']
```

The great majority of the policy make sense with several expectations (mark in red) it all involve an upcoming car, possible because the situation hasn't occurred too often for the agent to learn the correlate policy.