Yongchao Tang

Implement a basic driving agent

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

Answer:

It did run randomly and aimlessly and neglected all the environmental information such as traffic lights and approaching vehicles at each intersection. However, it arrived its destinations eventually after long time. The reason is that random choices will enumerate all possible states which surely includes the target location for each trip.

Identify and update state

Justify why you picked these set of states, and how they model the agent and its environment.

Answer:

I express the state as:

state = (next waypoint, inputs['light'], inputs['left'], inputs['oncoming'])

These four parameters determine all road states a car would meet. The allowed actions of a cab are then grouped into four categories, which are:

Only to stay.

Can turn right and stay.

Can go forward, turn right and stay, but not to turn left.

Can do all four actions.

Extracting these states from the inputs information can not only reduce the state space dramatically and effectively, but also accelerate the learning rates.

Implement Q-Learning

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

Answer:

I pick up the action corresponding to the maximal Q value given a state. However, it did not perform well. It might end up with a sub-optimal action and it failed to reach targets within the deadlines of many trials.

The Q table is initialized with all zeros and updated as the car moves. Since Q value of the state updated at each time step is determined by the chosen action. Once a Q value of the four actions is the biggest among a state, the car is prone to always choose the action, and the learning progress stops to update the Q elements of other actions in that state. For example, in the cases where ‘forward’ is given and the car is allowed to turn right and stay. If it chose to turn right at first, then the Q value is increased from 0 to 0.5*learning rate. Without making use of the exploration, the Q value corresponding to the state is increased at the most of the time because only the violations of the traffic law will result in penalties and turning right is allowed under the most of situations. Thus, it will still choose to turn right in the same situation next time because the corresponding Q value has become the maximum value in that row. With the matrix element increasing, it will never escape the local minimal by only exploiting the Q matrix from the beginning of the training.

Enhance the driving agent

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?

Answer:

There are three changes to the original algorithm:

First, the agent will choose actions randomly within the first 300 steps of the training. In this way, the agent can explore the whole state space sufficiently instead of being trapped by certain states of the Q matrix.

Second, I introduce another parameter epsilon, in addition to the learning rate alpha, and the discount gamma. With the possibility epsilon, the car is able to choose accessible actions randomly instead of choosing based on the Q values. In this method, the RL algorithm may escape some local minimums and let the car explorer all possible states as many times as possible. Also, the epsilon is decreased as the training goes. The epsilon is set to be the inverse of the training counts.

Second, to find the optimal values for alpha and gamma, I wrote the assisting loops to search the alpha and gamma in the range of [0.1, 1.0]. The metric is the number of the optimal actions shown up in the final Q matrix after 100 trials.

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?

Answer:

With this method, the agent firstly finds an optimal policy, and alpha = 0.6, gamma = 0.2. The parameters characterizing the model are as follows:

1. The total penalty times for the whole training process is: 113
2. The total moving steps of the car during the whole training process is: 1366
3. Choices frequency table:

	Stay	Right	Forward	Left
‘next point’ == 'right', allowed action == 'Stay'	0	0	0	0
‘next point’ == 'forward', allowed action == 'Stay'	2	0	0	0
‘next point’ == 'left', allowed action == 'Stay'	2	0	0	0
‘next point’ == 'right', allowed action == 'Stay', 'right'	23	74	16	16
‘next point’ == 'forward', allowed action == 'Stay', 'right'	437	15	18	25
‘next point’ == 'left', allowed action == 'Stay', 'right'	124	12	17	21
‘next point’ == 'right', allowed action == 'Stay', 'right', 'forward'	0	1	0	0
‘next point’ == 'forward', allowed action == 'Stay', 'right', 'forward'	0	0	4	0
‘next point’ == 'left', allowed action == 'Stay', 'right', 'forward'	3	0	0	0
‘next point’ == 'right', allowed action == all	14	70	18	22
‘next point’ == 'forward', allowed action == all	18	12	364	16
‘next point’ == 'left', allowed action == all	8	12	13	77

The above table has the choice frequencies in the columns. It shows the total number of choices of the corresponding actions given a state during the whole training process. The red numbers are the max frequencies in each row. Given these states, the car will move according to the next way point instructions without the violations of the traffic law. On the contrary, the 1st row is full of zeros, which means that the car would choose randomly from all four possible actions with the given instructions of next way point ‘right’ under the state which allows the car to only stay. These all zeros are obviously caused by the limited number of explorations the agent has experienced. This drawback may be overcome by increasing the randomly-choose steps.

4. Q table:

	Stay	Right	Forward	Left
‘next point’ == 'right', allowed action == 'Stay'	0	0	0	0
‘next point’ == 'forward', allowed action == 'Stay'	0.912	0	0	0
‘next point’ == 'left', allowed action == 'Stay'	0.912	0	0	0
‘next point’ == 'right', allowed action == 'Stay', 'right'	1.489474	2.406159	-0.50095	-0.51225
‘next point’ == 'forward', allowed action == 'Stay', 'right'	1.25	0.908134	-0.75	-0.75
‘next point’ == 'left', allowed action == 'Stay', 'right'	1.25	0.992663	-0.75004	-0.75002
‘next point’ == 'right', allowed action == 'Stay', 'right', 'forward'	0	1.272	0	0
‘next point’ == 'forward', allowed action == 'Stay', 'right', 'forward'	0	0	2.402044	0
‘next point’ == 'left', allowed action == 'Stay', 'right', 'forward'	1.07424	0	0	0
‘next point’ == 'right', allowed action == all	1.495584	2.408933	0.979218	0.844346
‘next point’ == 'forward', allowed action == all	1.481048	0.839035	2.396889	0.985886
‘next point’ == 'left', allowed action == all	1.460783	0.990237	0.803201	2.405121

From the Q matrix above, it can be seen that there are 11 out of 12 states show the optimal actions which have been learned by the agent.