<u>Machine Learning Nanodegree Project-04:Train a Smart cab to drive.</u>

Sanjiv Lobo,2016

Task 1:Implement a basic driving agent

Mention what you see in the agent's behavior. Does it eventually make it to the target location?

Implementing the basic driving agent

The initial driving agent was created with a greedy approach. It does better than choosing random values. It uses a greedy strategy and picks a random action to execute. After execution, it checks the reward if it's better than the earlier reward or not. If a bad reward comes, a random action is executed otherwise it performs an initiated action.
Yes it does make it to the destination since the grid is a closed loop.
Since enforce deadline is false, we do get 100% accuracy but most of the time the model overshoots the deadline parameter giving the model very bad efficiency.

Identifying the states

Here are some of the possible percepts that were candidates for selection:

- Deadline
- Next waypoint by the planner
- Traffic light
- Traffic states (oncoming, left, right etc.)

From all the inputs, the following state variables were chosen for this model:

- Next waypoint
- Traffic light
- Traffic states(oncoming)

Some of the ideal candidates in this case would be the destination or the location. But, here are some of the issues that could occur:

- Destination : It changes all the time, therefore making it a state would be useless as the agent would not learn properly.
- Location : The sheer size of the grid would cause problems, since for the q values to converge, it would need more trials.
- Deadline : It provides an unnecessary restriction on the model that is needed in the learning part of the process, if the deadline is used the model would not be accurate enough for a new environment.

Also, in order to model the destination as a part of the state, next waypoint was used.

Choosing traffic lights helps in modelling the traffic rule and also helps in training the cab to perform legal moves.

Choosing traffic state like oncoming helps in following traffic rules and making the model more realistic.

Task 2: Working of the Q-learning agent
States used are traffic light, oncoming and next waypoint.
The best action function determines the best action by checking the best possible q value,
The policy function gets the best q-value for any given state. Once the action is performed, we get the reward which updates the q-values. After few runs, the true q-values for each state are found, which helps in training the model and after every run the q table is updated,
The agent's behaviour is better than the basic agent implemented earlier as it learns from the mistakes made and behaves in an efficient manner. It breaks the traffic rules a lesser number of times now than with the basic agent. It also achieves a greater reward more often than in the basic agent. However, the route taken and the action done is not necessarily the best which indicates that the agent can be improved further by varying the parameters depending on it(alpha, gamma)

Test Results for the Q-learning agent
This is the initial configuration for the Q-learning agent(parameters alpha, gamma and beta)

1. Initial Q-value was set to a hypothetical value of 15 which is greater than any possible reward thus ensuring that exploitation and exploration don't occur.
2. Alpha value(learning rate) was set to 0.7, this value was found by trial and error over many values for alpha. Learning rate means the rate of updates that will happen to the q table. Having a value of 0 will make the model not learn at all and 1 will make it learn

everything. I chose a value of 0.7, so that the agent shouldn't learn everything at the same time we want it to learn quite fast so to achieve a balance the value was chosen.

3. Gamma value(discount value) was set to 0.5, set similarly by trial and error. It determines how important the rewards are to the agent. If set to 0 it regards only very recent rewards while 1 will make it consider long term rewards. We want an agent that will equally consider both and hence the value of 0.5 was chosen

Alpha	Gamma	Avg no. of fails(10 runs*100 trials)(less is better.)
0.2	0.3	12
0.4	0,6	10
0.6	0.4	6
0.8	0.6	9
0.3	0.3	11
0.7	0.5	5
0.5	0.7	9
1	0.5	7
0	0.5	85

About the agent's behaviour, as more runs are completed, the model makes less mistakes and is more efficient in its' functioning. The rewards gained are more than the earlier agent and the path found is way more efficient and quick. However, improvements can be made as it often takes a less efficient path which affects the rewards and efficiency.
The results improved a lot and an accuracy of more than 90% was achieved with enforce deadline is True, This is a great improvement compared to the basic agent and its accuracy within a deadline.
Also, the cab reaches the destination with a positive large cumulative reward.
The optimal path would be described as having the largest positive sum_reward and not breaking any traffic laws. It would also take the most effective route to the destination in the least amount of steps and reach there before the deadline is reached..

Thus our agent is fulfilling most of those requirements. To make it more optimal or ideal, further parameters can be examined by doing a grid search on the parameters alpha and gamma. It can also be made to optimize the routing and path of the car.