Train a Smart cab to Drive

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?

The agent chooses from a random set of valid action: **'forward', 'left', 'right'**. And through multiple attempts, the agent reaches its goal. However, the agent rarely achieves this goal due to the fact that the approach is random. It fails to reuse optimal paths that is effective in producing rewards. Given the absence of any sort of learning, the agent is always in exploration mode, each new trip is a roll of the dice. Another odd observation is that the agent is sometimes in the **'None'** action state, even when the traffic light is green and there is no oncoming traffic. The end result of always being in exploration mode is that the agent will not exploit what is has learned to increase its chances of getting rewards,

The success rate of the driving after **100** runs was an average of **19%**

Identify and update state

QUESTION: What states have you identified that are appropriate for modeling the *smartcab* and environment? Why do you believe each of these states to be appropriate for this problem?

There are four states that utilize for modeling the smart cab and the environment and these states include the traffic light (red/green) and the next waypoint are information that is available to the cab. I also use the oncoming traffic information input_1 is for 'oncoming' and input_3 is for 'left' One of the reason to use the traffic light is that helps in training the cab to carry out valid moves (this will come into play in Q learning). By using the traffic light and the next waypoint and oncoming traffic (left, right or None) from other cars in combination, they boost the agents learning which result in better success rate. Some other candidates in this case would have been deadline and destination. However, there are some issues that arise when using these variables; hence they had to be excluded.

Destination: The agent would be unable to properly learning if the destination was to be encoded as a state. Given that at each interval, the destination changes.

Location: Due to the sheer size of the grid, this is a variable that should be not be used to model the state as it leads to the problem of having to many states. And for the q values to converge, we will need more than 100 runs.

Deadline: Time is not included because at any given time a state is defined, time is already incorporated. Hence, time is not used as a state in Reinforcement Learning.

Implement a Q-Learning Driving Agent

QUESTION: What changes do you notice in the agent's behavior when compared to the basic driving agent when random actions were always taken? Why is this behavior occurring?

The easiest thing to notice after implementing q-learning is that the agent success rate shoots up, the agent has learned the rules of the traffic light and avoids penalizing actions. Also, the agent no longer remains in an inactive state when the light is green and there is no oncoming traffic.

This is happening because the agent is exploiting paths that maximize its reward rather than blindly exploring its environment.

Improve the Q-Learning Driving Agent

QUESTION: Report the different values for the parameters tuned in your basic implementation of Q-Learning. For which set of parameters does the agent perform best? How well does the final driving agent perform?

Q-value was not set to a specific value. By comparing the random value generated against the epsilon, we either of explore or exploit (select the max q value from the Q table)

Alpha value was set to 0.1 but this was not an optimal learning rate. Over multiple iterations, the value was set to 0.5.

Gamma value was set originally to 0.5. Through constant iteration and good performance output when testing over a set of ranges, the final value was 0.9.

Epsilon value was originally set to 1. so if the random generated value was less than or above the epsilon, this influenced the agent's decision to explore or exploit.

The agent on average scored 84%

Alpha	gamma	Epsilon	results
0.2	0.5	1	13% success
0.2	0.5	0.1	64% success
0.1	0.7	0.1	41% success
0.1	0.8	0.1	71% success
0.5	0.9	0.1	57% success
0.25	0.9	0.5	61% success
0.1	0.9	0.05	84% success

An epsilon value of 0.05 is used so that the agent does a lot more exploitation than exploration

QUESTION: 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? How would you describe an optimal policy for this problem?

The agent seemed able to find the optimal policy by being able to find the quickest path with very small error rate once Q values are converged. After iterating over multiple alpha, gamma and epsilon values and learning the rules of the environment, the cab, on average reached the destination 84 times out of a 100 runs.

We can describe an optimal policy for this problem as our agent haven learned to not only make the right(valid) moves, but to also take the shortest route without performing many illegal moves. However, there are a few trials (last 10) where the agent breaks the law. And some of these include

Iteration	Input	waypoint	action
90	{'light': 'red', 'oncoming': 'left', 'right': None, 'left': 'forward'}	Right	Right
94	{'light': 'red', 'oncoming': 'left', 'right': None, 'left': None}	Forward	Forward
94	{'light': 'red', 'oncoming': 'left', 'right': None, 'left': None}	Left	Forward
100	{'light': 'red', 'oncoming': 'forward', 'right': None, 'left': None}	None	Left

More data on the last 10 trials available in the qlearning_log.txt in the result folder

References

Studywolf – reinforcement learning