Smartcab Project Report

Ricardo Cortez

1) QUESTION: Observe what you see with the agent's behavior as it takes random actions. Does the smartcab eventually make it to the destination? Are there any other interesting observations to note?

The smartcab actions are randomly selected, therefore, sometimes it makes it to the destination but not in an efficient way. The majority of the times the smartcab does not make it to the destination.

2) 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?

Identified states: 'traffic light', 'oncoming', 'right', 'left', 'next_waypoint'

The first four states come from sensing the environment. They are necessary because before the car takes any action it needs to know whether the traffic light is red or green as well whether there are oncoming cars from any directions.

The fifth state, 'next_waypoint' was considered appropriate because the smartcab needs to factor in the action it needs to take to get to its destination. This does not necessarily mean that that is the most efficient move, it would depend on the sensing environment whether it moves in that direction or not. For instance, 'next_waypoint' could indicate to move forward but if 'traffic_light' is red, then smartcab action will be to idle at the intersection (None).

3) 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?

Initially, the Q-matrix was initialized to zero and the rate to convergence was low. Then, I consider increase the number of trials by an order of 10. With 1000 trials, the Q-matrix performed better but running time was too costly. Also, at this point in the implementation, my code was lacking of the simulated annealing algorithm and I was only considering taking actions from the Q-matrix in the final trials, all the other times it was taking random actions.

The next step was to initialized the Q-matrix randomly which made a considerable improvement. That, along with the added code to account for the learning rate (alpha), the discount factor (gamma) and the exploration rate (epsilon) made great improvement. The number of trials was reset to 100.

The smartcab makes errors and fails in the initial trials but as it learns from these trials it starts to improve its performance. The learning rate is high, 1.0, in the initial trials indicating that a large weight would be put into learning. As the learning rate decreases, at every step, a larger weight is put into the learned knowledge. Another way to see it is that at the beginning it learns and stores that information in its memory (Q-matrix) but as it evolves it starts using more its memory and less from its learning ability.

Similarly, the exploration rate (epsilon) allows to take a random actions every so often so that it allows

for exploration of the environment. Also, eliminates the problem of local minima (sub-optimal policy is achieved but it is unable to improve upon it). As the trials are exercised, the rate decays so that more emphasis goes to the exploitation and lesser to exploration. It is done so that at the final trials epsilon is close to zero and the exploitation occurs with probability 1- epsilon (the probability of exploitation is very high).

Once all these smaller pieces are put together it is noticeable the evolving behavior of the smartcab over the trails as it makes lesser errors and achieves its destination more often.

4) 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?

The discount factor, gamma, was tested with the following [0.2 0.4 0.6 0.8]. Similarly, the exploration rate wast tested with values [0.3 0.5 0.7 0.9].

The learning rate, (alpha), was computed as 1/t where t is initialized to 1 and increases by 1 at each step so that the sum(1/t) diverges but the sum(1/power(t,2)) converges.

The performance for each combination of alpha and epsilon is done by running 100 simulations and computing the average performance. Each time a new simulation is run, the performance over the last 10 trials is recorded. If the smartcab arrived to the destination within the alloted time, it succeeded, otherwise it failed. The performance per simulation is then the number of successes in the last 10 trials divided by 10. The overall performance is then the average of the individual performances throughout the 100 simulations.

The table below shows the individual performance for each combination of Gamma and Epsilon, the set of parameters under which the performance was the highest corresponds to Gamma = 0.2 and Epsilon = 0.5 with an overall Performance of 0.89.

Gamma	Epsilon	Performance
0.2	0.3	0.84
0.2	0.5	0.89
0.2	0.7	0.88
0.2	0.9	0.88
0.4	0.3	0.79
0.4	0.5	0.83
0.4	0.7	0.87
0.4	0.9	0.88
0.6	0.3	0.80
0.6	0.5	0.82
0.6	0.7	0.84
0.6	0.9	0.88
0.8	0.3	0.72
0.8	0.5	0.81
0.8	0.7	0.86
0.8	0.9	0.88

Table 1. Smartcab performance for different values of gamma and epsilon

5) 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?

After running the simulation 10 times, the smartcab makes in average 0.4 invalid moves (penalty = -1) during its last trail. It also makes an average of 0.3 valid but penalized (0.5) moves, these moves are legal but different to the next_waypoint. Therefore, the smartcab seems to learn from both, avoiding illegal turns as well as inefficient turns. It seems to be close to find an optimal policy.
An optimal policy would be such that the smartcab takes the same actions as the next_waypoint except when the traffic light is red for which case the action should be to idle at the intersection(None).