Reinforcement Learning Project: Smartcab

Implement a Basic Driving Agent

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?*

ANSWER: The smartcab might eventually make it to the destination, but purely by chance. Observations I found interesting revolved around how this simulation works, mainly that the smartcab can't run a red light and that the other cars don't seem to be in a hurry to get anywhere specific. Also despite the project prompt including a discussion of the legality of taking a left turn after yielding to oncoming traffic the simulation had no intersection to enter, so it's not possible. So much for city driving.

Inform the Driving Agent

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?*

ANSWER: The states are made up of the next waypoint [left, forward, right], the traffic light state [red, green], and two of the other-agent spaces [oncoming, left] each with four possible states [None, left, forward, right]. Since these are mostly received as strings I label states by concatenating each, e.g. leftredNoneright would be one state. This takes into account all but two pieces of information that the smartcab has access to: right coming traffic and the deadline value. Right coming traffic never has right of way at a traffic light under US traffic law - either it is waiting at a red light or a right turn on red never crosses it's lane. The deadline also provides no useful information because the cab is entirely dependent on the planner for route finding and never knows the remaining distance to the goal, meaning even if the smartcab knew it was short on time there would be no way for it to assess the remaining route for short-term-penalty/long-term-reward changes. Hence including the deadline in the state would do nothing but increase the number of states by a factor of 60.

OPTIONAL: *How many states in total exist for the* ***smartcab*** *in this environment? Does this number seem reasonable given that the goal of Q-Learning is to learn and make informed decisions about each state? Why or why not?*

ANSWER: 96 states in total. This seems a reasonable number of states to learn in 100 trials but the majority of these states are traffic encounters and with the way the simulation is set the chance of encountering any of the traffic states is very small. Each state needs to be encountered at least four times before having any confidence in choosing an action so with each trial taking roughly 25 actions (assuming progressive improvement) then a traffic-learning state, without repeats beyond the desired four, would have to be encountered more than three times per trial. This estimate is a bit contrived but it shows how unreasonable an expectation this is with only a few other agents being rendered. Initially I reduced the number of states by considering even more equivalencies like how a car to the right of the agent never has any influence on the appropriate decision, and taken to its extreme by effectively hard-coding in traffic laws this reduces the number of states to eight. However posts by reviewers and staff on the Udacity forums make it clear this is against the spirit of the project. Personally I think letting a smart car learn the law through trial and error seems a bit unlikely but I understand that the problem lies in the infrequency of other traffic rather than the learning algorithm itself. Increasing the number of other agents would be an effective way of increasing traffic encounters and thus converging on a more optimal policy.

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?*

ANSWER: Compared to the random actions the smartcab is clearly getting better at reaching the destination in successive trials. This is because it's method of choosing which action to take is now based on a learning method that takes the rewards into account, over time reinforcing the choice to follow the planner while avoiding accidents. However, looking directly at the Q values shows that even doing a reasonable action lowers the score a bit since taking no action gives 0.0 reward which will always lower a Q score even if it is the only non-negative reward. Also, as discussed above, the sheer number of states involving other traffic means the smartcab acts mostly randomly when encountering other cars.

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?*

ANSWER: The Q-Learning parameters used were α, γ and ε with a decay rate. First I tried three parameter sets and measured the success ratio (trials/successful trials), the penalty ratio (number of negative rewards/total number of rewards), and the sum of all rewards. The three parameter sets I tested were:

A: {α:0.8, γ:0.8, ε :0.1 with -0.05/action}

B: {α:0.8, γ:0.0, ε :1.0 with -0.005/action}

C: {α:0.1, γ:0.01, ε :0.5 with -0.005/action}

I've constructed a table below showing how the three parameter sets performed on average over 10 sets of 100 trials.

Averaged Success Metrics for Varied Parameter Sets

	Success Rate (%)	Penalty Rate (%)	Cumulative Reward
A	91.3	9.99	2379.3
B	96.2	6.25	2171.3
C	97.1	8.78	2233.75

After observing these results I tried one more set, D:{α:0.8, γ:0.01, ε :0.5 with -0.005/action}.

Averaged Success Metrics for Final Parameter Set

	Success Rate (%)	Penalty Rate (%)	Cumulative Reward
D	98.2	3.77	2214.8

This shows a marked improvement over the other sets in success rate and penalty rate. The results also suggest cumulative reward is a poor judge of success, as for A B and C it has a superficial correlation with penalty rate. I suspect that this is showing that taking wrong actions results in taking more actions overall, which inflates the cumulative reward. With parameter set D the final driving agent performs very well. One additional note is that I initialized Q values to 2.0, which promotes exploration slightly.

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?*

ANSWER: I believe it gets close, as the averaging above demonstrate a 98.2% success rate. This suggests it's forming a good policy within a few trials. It's not perfect, however, as it does incur some penalties – the infrequency of states including other traffic mean that it is likely to act somewhat randomly when encountering traffic – though this doesn't seem to affect reaching the destination and could be solved with a larger number of trials or traffic agents. Measuring the minimum possible time would be very challenging without determining how the traffic lights switch and whether there's any way to go around red lights, which I don't think is information the car is supposed to have access to anyway. Knowing the traffic laws an optimal policy would be: check if the next waypoint is blocked by a red light or other traffic having the right of way, if so do None, if not go towards the next waypoint.