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 the three other-agent spaces [oncoming, left, right] each with four possible states [None, left, forward, right]. Since these are mostly received as strings I label states by concatenating each, e.g. leftredNonerightNone would be one state. States made this way represent the full extent of information the smartcab has access to, describing every situation it might encounter, making them the appropriate selection for this problem.

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: 384 states in total. Frankly, this is too many to cover in 100 trials. Each state needs to be encountered at least four times before having any confidence in choosing an action, and the chance of encountering any of the traffic states is very small. If each trial takes roughly 25 actions (assuming progressive improvement) then a traffic-learning state, without repeats beyond the desired four, would have to occur over 60% of the time. This estimate is a bit contrived but it shows the absurdity of expecting the car to learn in the time given. Initially I reduced the number of states by considering equivalencies, e.g. 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. I might try increasing the number of NPCs to test this.

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. Some different sets I tried were:

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

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

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

They all seemed to perform well, reaching the destination every time from the fiftieth or so trial on. Interestingly a high γ seemed to cause the car to prefer turning right at a red light over waiting for it to change, despite the penalty, and a high α balloons the final Q scores quite a bit. Neither seemed to affect the overall success rate. I chose the last set, {α:0.1, γ:0.01, ε :0.5 with -0.005/action}, and set the state initialization to start each Q score at 2.0.

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; in over ten sets of 100 trials it successfully reached the destination on at least the last 60 trials. But not perfect, 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. 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.