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?*

Even with enforce_deadline set to False, there is a hard time limit of 125 time steps to reach the destination. Generally, the agent failed to reach the destination – on those few occasions that it did, it was because the agent started close to the destination and managed to stumble onto the right set of actions.

Some other observations about the agent and the environment in general:

- Unlike most cities, this one is toroidal; when an agent moves off one boundary, it reappears on the other side of the map.
- The environment simulator enforces the traffic rules; rather than allowing accidents to occur and ending the simulation, the simulator simply forbids the action from occurring and the agent incurs a negative reward.
- All other agents communicate perfectly about their intent – the inputs list doesn't just indicate the presence of other agents at the intersection, but also the intended action of each agent.
- The reward function has an ideal route built into it based on a negative reward for not following the recommendation of the planner, and imposes no negative reward for idling (though when enforce_deadline is True, there would be an implicit penalty for idling.)

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?*

- A state is needed for the color of a light, since that is used by the traffic rules.
- States are needed for the upcoming actions of other agents at the intersection.
  - There are three possible locations for agents, and four possible actions for each of those agents.
  - We could reduce the dimensionality here by noticing that, based on the traffic rules, not all those states are relevant – the only three actions of significance are oncoming moving forward, oncoming turning right and left moving forward. There's a philosophical issue here – should we embed our domain knowledge about the traffic rules into the possible states, or should we be a little more tabula rasa and let the agent learn what is and isn't significant? After looking to the forums for some guidance from the coaches, it seems the intent is the latter – we are demonstrating the ability of the smartcab to learn traffic rules.
  - Another way to reduce the dimensionality is simply to mark the presence of other vehicles at the intersection with an intent to move, rather than their specific intents. This would cut the number of states from 64 down to eight, a factor of eight improvement, at the expense of making the agent overly cautious. I'll discuss this further in the "number of states" question below.
- Since we have a planner making recommendations, and since there is a reward or penalty associated with following or ignoring the recommendations of the planner, the planner's recommendation should be incorporated as a state (three possible values, since it never recommends None).
- While we're used to getting location as part of state in the gridworlds problems, location is explicitly disallowed here – it's not part of the inputs.
- The last consideration is whether or not time to deadline should be considered as a state. My original analysis went as follows:
  - Since the simulator does not allow us to violate traffic rules, the agent cannot gain by choosing such an action with time running out.
  - The agent *could* potentially gain by choosing a different action from the planner – but without information about the relative location of the destination, it could just as easily make matters worse, and the lessons learned would not be applicable to future trials.

- Including it in the state means that the agent will have to learn separately for each time step; in essence, we reduce the number of learning trials for general behavior. This could be resolved by converting deadline into a limited number of states – think about a football team having normal strategies vs. "two-minute" strategies when the clock is short.
- Finally, the agent will have a limited number of "time running out" trials in which to learn, even if we did provide it with relative position information. This could be solved by running a number of faster trials with time running out (akin to a football team practicing two-minute drills), but it's not the way our trials are currently setup.

Based on these considerations, I chose to omit deadline from the states.

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?*

There are two possible values for the state of the light, three possible values for the planner recommendation, and either eight or 64 total states for traffic interaction. In total, there are either 48 or 384 possible states and either 192 or 1536 possible tuples of state and action. With 100 trials and 20 to 60 time steps per trial, we will average around 4000 transitions. There are a number of states we will only see a limited number of times; in particular, we will have limited opportunities to learn about traffic interactions. As a result, I'd argue for trying states representing the presence of traffic from the various directions to limit the dimensionality and seeing how that works. (See discussion in the next question for how that worked out).

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?*

By the end, the agent is in general finding its way to the destination point within the allotted time. In particular, when the light is green and there is no interacting traffic, the agent usually (but not always, due to the non-zero exploration rate) follows the planner recommendation. This makes sense given that we are rewarding the agent for following the planner recommendation.

At a red light, with no traffic interaction, the behavior is more interesting. When the planner recommendation is to move forward, the agent will generally wait at the light. When the planner recommendation is to turn right, the agent learns the right turn. But when the planner recommendation is to go left, the agent will also sometimes learn a right turn if the light is red, and then loop back around to the intersection.

To understand this loop behavior, consider that while we get a penalty of -0.5 points for moving against the planner recommendation, we get a reward of 2.0 points per step for following the planner recommendation – so a loop has a net reward of 5.5 points. It suggests that the alpha value might be high, since the immediate penalty for violating the planner is being overridden by the future value of the loop, or that the reward function needs to be tweaked… but the latter is outside the defined scope of the project. The reason we don't get left-handed loops is due to the ability to turn right on red – the planner has learned that that is acceptable. It's interesting that the planner has learned *not* to loop when the planner is calling for 'forward' – one wonders if it might unlearn the loop with more trials.

The loop behavior made me go back to my original decision to not include the deadline, since a strategy to maximize reward could be to pick up points for looping and then dash for the line as time is running out. But I believe this to be an artifact of the reward function, rather than a desired behavior, as the project requirements state our goal is that "the self-driving agent implementation should learn an optimal policy for driving on the city roads while obeying traffic rules, avoiding accidents, and reaching passengers' destinations in the allotted time." If the goal were to also maximize the length of the trip (and thus the fare) within those parameters, then maybe the looping behavior would be desirable!

The final observation I made was that the agent was doing a poor job of learning how to interact with traffic. The agent was still receiving traffic penalties even in the last trial, and by looking at the Q values, it was clear that it wasn't learning the rules.

I first explored whether this could be based on my initialization of the Q values. I at one time had initialized the values near 0, and found that that led to the agent sometimes choosing inaction when at a turn; I initially solved that by introducing a "bias for action" whereby the Q values for 'None' actions had a lower initial value that the other actions. In my current experiments, I had chosen a high value of 15.0, which cleared the inaction behavior, while keeping the looping behavior I had observed in the earliest experiments. Thinking about this further, I went back and chose a value of 4.0 – the idea was to make the penalty for a traffic violation and the discounted Q values (with gamma = 0.2) end up about the same, so the penalty would have more effect. While this did not change the traffic penalty behavior substantially, it did clear the looping behavior noted above.

I then took a look at the coverage of the various states by the trials. Only 6% of the transitions involved traffic interaction, and 25 of the states with traffic interactions were never experienced at all. With only 100 trials to work with, I decided to consider going back to the drawing board.

Instead of looking at the presence of a vehicle at each of the three directions, I decided to look at consolidating those down to one state. The danger here was that the agent might not learn enough caution, since in some transitions it would be able to move through the intersection safely. After the consolidation, I found that the agent was still not learning rules that would allow it to eliminate traffic penalties, but since each state had coverage, there is hope that with tuning or if additional trials were made available. Interestingly enough, the looping behavior returned with these changes.

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?*

I decided to look at tuning for each parameter, to see how they changed the behavior and to see if they helped with the limited transitions with interacting traffic issue. In the initial experiments, the values for the parameters were alpha = 0.2, gamma = 0.2 and epsilon = 0.1.

Before tuning, we need to address the appropriate metrics to tune against. While total reward would normally be an attractive metric, we've already identified an undesirable looping behavior that will improve reward at the cost of increasing the length of the trip. Instead, we'll look at metrics more closely related to the actual business goals.

- Success rate – how often does the agent reach the destination in time?
- Traffic penalties – how often does the agent violate traffic rules?
- Planner penalties – how often does the agent ignore the planner recommendation and potentially lengthen the trip? This metric should have less strength than the other two, since, in some cases, going against the recommendation might speed up the trip.

Since we want to judge how well the agent has learned, we don't want to consider the early, random training runs, and we'd like to reduce or eliminate exploratory behavior in the late trials. For this purpose, we'll break the trials into a training set (first 80) and a testing set (last 20), and turn the exploratory rate down to 0 for the testing set. Note that, unlike the use of this breakdown in supervised learning, the system will continue to be trained even when working on the testing data.

When running simulations, I noticed some variance in the behaviors learned. To tune parameters, we need to not just go through a number of trials with each parameter, but also go through a number of different training paths over different runs of 100 trials. Therefore, a group of 100 trials will itself be simulated 100 times, with the agent being reset back to an initial state after each set of 100 trials is

complete, and we'll look at the testing statistics over the results of all 100 simulations (10000 trials, with 2000 trials being used for testing).

I ran a first experiment with alpha ranging from 0.1 to 0.5, epsilon ranging from 0.1 to 0.5 and gamma ranging from 0 to 0.5. In the resulting data, I performed a linear regression between the number of trial successes and the total of planner penalties and calculated an R^2 coefficient of .9768. With that degree of correlation, it was sufficient to look just at the planner penalties for further analysis. Ignoring the trial successes. That further suggested that looking at the total of traffic penalties and planner penalties might be a reasonable simplification.

Looking at the results grouped by the various parameters, I found the following:

alpha	Mean of traffic penalties	Mean of planner penalties	Mean of total penalties
0.1	-241.9666667	-1812.533333	-2054.5
0.2	-198.1333333	-1527.183333	-1725.316667
0.3	-163.8666667	-1331.016667	-1494.883333
0.4	-130.7666667	-1146.15	-1276.916667
0.5	-99.66666667	-967.4833333	-1067.15
Grand Total	**-166.88**	**-1356.873333**	**-1523.753333**

epsilon	Mean of traffic penalties	Mean of planner penalties	Mean of total penalties
0.1	-188.6	-1336.333333	-1524.933333
0.2	-176.1333333	-1370.066667	-1546.2
0.3	-159.8666667	-1362.183333	-1522.05
0.4	-160.1666667	-1365	-1525.166667
0.5	-149.6333333	-1350.783333	-1500.416667
Grand Total	**-166.88**	**-1356.873333**	**-1523.753333**

gamma	Mean of traffic penalties	Mean of planner penalties	Mean of total penalties
0	-161	-74.3	-235.3
0.1	-165.16	-81.48	-246.64
0.2	-163.96	-246.78	-410.74
0.3	-169.04	-1285.7	-1454.74
0.4	-175.8	-2840.82	-3016.62
0.5	-166.32	-3612.16	-3778.48
Grand Total	**-166.88**	**-1356.873333**	**-1523.753333**

- As the gamma value grows, the number of planner penalties grows – there is in particular a jump from 0.1 to 0.2. This makes sense in light of the looping behavior noted earlier – as we decrease gamma, we increase the focus on immediate reward over future utility, and following the planner recommendation in the current step overrides potential reward for entering a future loop. On the other hand, gamma appears to have little impact on traffic penalties, which again makes sense given that the entire impact is immediate; this might be different if a traffic violation had a chance of ending the trial.
- As the epsilon value grows, the traffic penalties trend downwards, but planner penalties don't change too much. However, if we restrict ourselves to looking at gamma = 0 or 0.1, then a trend downwards in planner penalties also occurs. Since the penalties are still trending downwards at the maximum epsilon considered for small gamma, we need to see what happens with higher epsilon values.

- Finally, both traffic and planner penalties trend downwards as alpha grows. Again, since the trend is downwards at the maximum alpha, we need to see what happens with higher alphas.

Based on these results, I ran a second set of experiments. This time, alpha was allowed to range from 0.4 to 0.8, epsilon from 0.3 to 0.8 and gamme from 0 to 0.1, again in increments of 0.1. Again looking at the results grouped by parameters, I found the following:

alpha	Mean of traffic penalties	Mean of planner penalties	Mean of total penalties
0.4	-108	-54.08333333	-162.0833333
0.5	-77.08333333	-35.5	-112.5833333
0.6	-41.58333333	-22.45833333	-64.04166667
0.7	-39.41666667	-17.33333333	-56.75
0.8	-27.5	-12.66666667	-40.16666667
Grand Total	**-58.71666667**	**-28.40833333**	**-87.125**

epsilon	Mean of traffic penalties	Mean of planner penalties	Mean of total penalties
0.3	-63	-27.95	-90.95
0.4	-58.7	-27.35	-86.05
0.5	-55.6	-26.65	-82.25
0.6	-57.3	-25.7	-83
0.7	-55.6	-30.85	-86.45
0.8	-62.1	-31.95	-94.05
Grand Total	**-58.71666667**	**-28.40833333**	**-87.125**

gamma	Mean of traffic penalties	Mean of planner penalties	Mean of total penalties
0	-55.13333333	-24.15	-79.28333333
0.1	-62.3	-32.66666667	-94.96666667
Grand Total	**-58.71666667**	**-28.40833333**	**-87.125**

- For gamma, this provides further support for choosing a low gamma – whether gamma should be 0 could depend on the final choice of the other parameters. If we look at the list of parameters sorted by total penalties, the top 10 points in parameter space all have gamma = 0.
- For epsilon, it seems as if we've captured the best values in our interval, since the penalties appear to reach a minimum near the midpoint of the range. The top 10 points in parameter space, though, run the full range of epsilon values.
- For alpha, the number of penalties is still decreasing even for the maximum alpha. This is a surprising result – it means that our last lesson appears to be the best lesson.

The epsilon and alpha results made me a little suspicious of the training vs. testing methodology. In particular, the epsilon value makes it clear that, during the training trials where the exploratory rate is non-zero, there is an advantage to letting the agent explore a variety of alternatives. But this comes at a cost; during the training trials, the agent is very random. If the objective is to build confidence in the agent's ability to learn quickly, an observer would be quite concerned at the agent's behavior.

As a result, I decided to run a third and final experiment. For this experiment, I allowed alpha to range from 0.1 to 1, epsilon to range from 0.1 to 1, and gamma to range between 0 and .1. For each point in parameter space, I took the four totals of planner penalties and traffic penalties for all trials and the test trials, and scaled each total to the interval [0. 1], where 0 represented the worst total across parameter space and 1 the best total. I then considered the average of those scores to generate two scores, one for

the all trials case and one for the test trials space, and then took an average of those averages to arrive at a final score.

Looking at those scores, I found the following cross-section results:

alpha	Mean of total score	Mean of test score	Mean of final score
0.1	0.440310129	0.355787208	0.398048668
0.2	0.47112175	0.491009695	0.481065723
0.3	0.481295429	0.594560768	0.537928098
0.4	0.487541583	0.688532556	0.58803707
0.5	0.493394426	0.8144796	0.653937013
0.6	0.497649277	0.879864956	0.688757117
0.7	0.500067092	0.890470181	0.695268637
0.8	0.501483757	0.919566996	0.710525376
0.9	0.504292454	0.944168857	0.724230656
1	0.505510799	0.951750186	0.728630493
Grand Total	**0.48826667**	**0.7530191**	**0.620642885**

epsilon	Mean of total score	Mean of test score	Mean of final score
0.1	0.883004744	0.696488743	0.789746743
0.2	0.811539141	0.726240321	0.768889731
0.3	0.726826979	0.752706105	0.739766542
0.4	0.630005058	0.771455613	0.700730336
0.5	0.519367953	0.758505849	0.638936901
0.6	0.399112478	0.774327906	0.586720192
0.7	0.272110005	0.774490691	0.523300348
0.8	0.140030382	0.774982629	0.457506505
0.9	0.012403287	0.747974047	0.380188667
Grand Total	**0.48826667**	**0.7530191**	**0.620642885**

:

gamma	Mean of total score	Mean of test score	Mean of final score
0	0.489503136	0.764605554	0.627054345
0.1	0.487030203	0.741432647	0.614231425
Grand Total	**0.48826667**	**0.7530191**	**0.620642885**

Interpreting these results:

- Taking gamma = 0.0 yields, on the whole, slightly better results than gamma = 0.1 for all three scores
- All three scores also monotonically increased with alpha to the maximum value of 1.0; in other words, the algorithm learns best when past results are ignored.
- The best value for epsilon depends on whether we are only interested in the behavior of the agent in the final trials (a steady-state behavior) or the behavior across all trials. In the former case, a high epsilon is preferred to allow for a thorough explanation of parameter space, though not the highest possible values. In the latter case, a low value of epsilon is preferred. When we weight these two equally, a low value is still preferred.

When looking at the tuples of parameters:

- The highest value for the score restricted to the last 20 trials occurs for alpha = 0.9, epsilon = 0.8 and gamma = 0.0.
- The highest value for the score across all trials occurs for alpha = 0.9, epsilon = 0.1 and gamma = 0.0.
- The highest value for the mean of those two scores occurs when alpha = 1.0, epsilon = 0.1 and gamma = 0.0.

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?*

I looked at a sample simulation run for each of the three parameter sets above. In all three cases, the agent learned the same policy on green: follow the planner recommendation unless being directed to turn left with interacting traffic present, in which case it should idle. The first two agents learned the same policy on red: idle if directed to move forward or left, but turn right if directed right, regardless of interacting traffic. The third agent had the same policy, except that it was still trying to turn illegally left on a red when interacting traffic was present. It is worth noting that this state only occurred twice in the simulation run, so even with the limited states, 100 trials may not always be sufficient.

In all three cases, the behavior within the last 20 trials was nearly perfect; all three reached the destination on time in all 20 trials, and only the agent running on the third set of parameters incurred a traffic penalty within the last 20 trials. However, there was a marked difference when looking at the speed of convergence on good behavior – the agent running on the first set of parameters had a 45% success rate, compared to 95% or better for the other two agents, and incurred nearly ten times as many traffic penalties (490 vs. 46 and 54, respectively. Based on those anecdotal results combined with the analysis above, I would be inclined to choose alpha = 0.9, epsilon = 0.1 and gamma = 0.0 in general.

Given the simplified state, the policy learned by the first two agents is close to but not quite optimal. When interacting traffic is present on a red light, an agent with the optimal policy should decline to turn right when directed by the planner, since the intervening traffic might be traffic from the left moving forward. The failure to learn this piece of the optimal policy is likely due to the scenario failing to occur during the test trials.