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?

The SmartCab appears to eventually make it to the destination, although sometimes it takes quite a while. Interesting to see that if the cab goes off the boundary it appears on the other side. The route planner does not appear to recommend going off the boundary though. Also the state is always reported as none.

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?

I chose to use all the input values, plus the waypoint value. The input value includes traffic light, and the flow of traffic in each direction of the intersection. There is likely an argument to be made that maybe the traffic to the right does not matter because if our car has the green light then it has the right of way and should not consider where the car to the right is going. Also, if our car has the red light and wants to turn right, the only other direction of interest is the left traffic and if someone is coming straight through there. However for completeness I chose not to omit right traffic from the states. Also, the waypoint is clearly an important factor to include in the current state because that is where we want to go. I chose not to include the deadline in the state. This would have increased the number of possible states into the thousands which would be too many for our likely number of training instances. Also the reward structure should already be taking the deadline's effect into account. We get a large reward for finding the target and the agent should be finding the target as quick as possible regardless of the deadline. Maybe if we were simulating an ambulance or some other emergency vehicle and we would want our vehicle to break a few traffic laws if it meant reaching the deadline in time, then having the deadline would be very important. But in our application it should not really change the behavior of the vehicle.

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?

Each state is defined by the state of the traffic light, traffic to left, traffic to right, traffic straight ahead, and waypoint. The traffic light has 2 unique values, the traffic in each direction has 4 values, and the waypoint has 3. So the total number of states is 2x4x4x4x3 = 384 states. This seems like a fairly high number of states, but as I mentioned before the traffic to the right really makes no difference so this number could be 384/4 or 96 states. With the simulation set to run 100 times and if there are maybe an average of 30 actions per simulation, that will be 3,000 actions. I think this is probably enough to sort out which actions for a given state yield a high reward and which do not. I am guessing the most frequented state is going to be at an empty intersection with the light red

Implement a Q-Learning Driving Agent

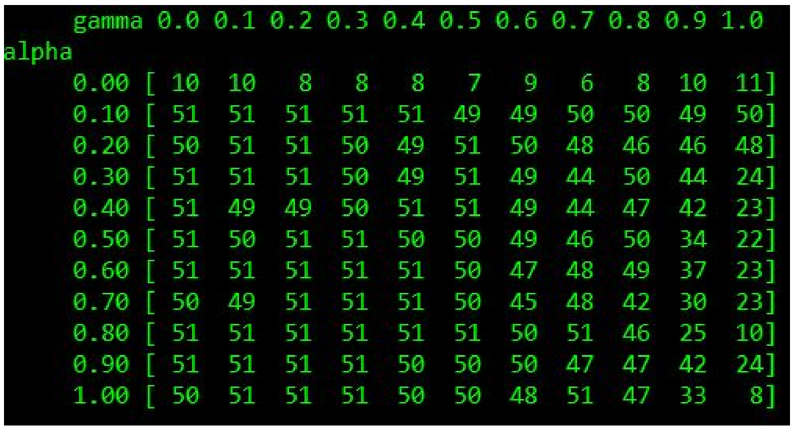
QUESTION: What changes do you notice in the agent's behavior when compared to the basic driving agent when random actions were always taken? Why is this behavior occurring?

With the Q learning implemented epsilon is set to 1 then reduces by 0.02 each trial, alpha and gamma are both set to 0.5. The agent now mostly randomly wanders around for a bunch of trials, usually does not reach the target but does sometimes. After a while the agent starts to follow the waypoints almost all the time and very consistently reaches the destination. The most noticeable behavior change is the agent following the waypoints. It does this now because in the past when it happened to take the action which matched the next waypoint it received significant positive reward for doing so.

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 modified the code so it completes one full trial of 100 rounds for each unique combination of alpha and gamma for the values I provided. I am also now tracking whether or not the destination is reached but I only track it if epsilon is equal to zero, which means we are not exploring randomly at all anymore. With epsilon starting at 1.0, and being reduced by 0.02 for each of the 100 trials, the maximum number of successfully reached destinations possible is 51 once epsilon is 0. Below is a table for gamma and alpha combinations where gamma and alpha are between 0 and 1 in 0.1 increments:

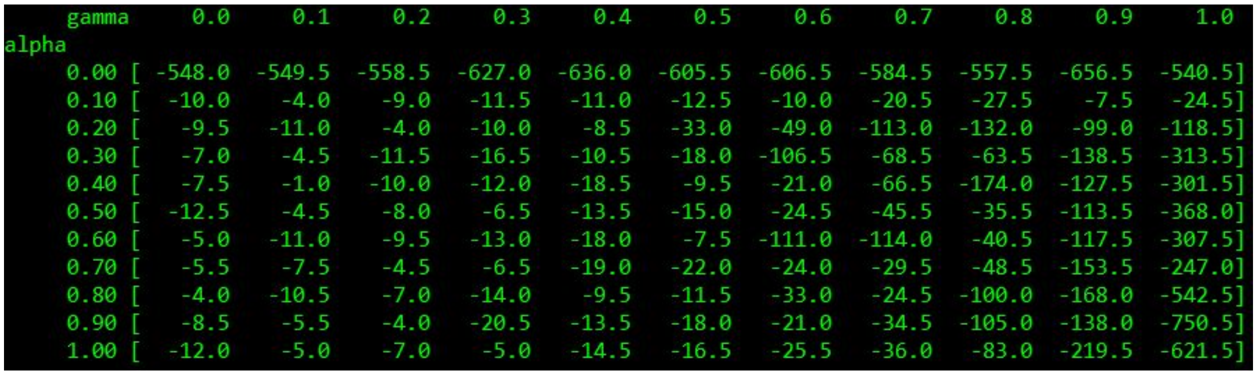


```
     gamma 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
alpha
     0.00 [ 10  10   8   8   8   7   9   6   8  10  11]
     0.10 [ 51  51  51  51  51  49  49  50  50  49  50]
     0.20 [ 50  51  51  50  49  51  50  48  46  46  48]
     0.30 [ 51  51  51  50  49  51  49  44  50  44  24]
     0.40 [ 51  49  49  50  51  51  49  44  47  42  23]
     0.50 [ 51  50  51  51  50  50  49  46  50  34  22]
     0.60 [ 51  51  51  51  51  50  47  48  49  37  23]
     0.70 [ 50  49  51  51  51  50  45  48  42  30  23]
     0.80 [ 51  51  51  51  51  51  50  51  46  25  10]
     0.90 [ 51  51  51  51  50  50  50  47  47  42  24]
     1.00 [ 50  51  51  51  50  50  48  51  47  33   8]
```

The values in this table are successful completions only once epsilon has reached 0, with the maximum being 51 successes. Looking at this table I chose gamma to be 0.3 and alpha to be 0.7 for my final implementation. There are lots of choices where the learner basically always succeeded after it finished all exploring but this combination seems good because it is generally near the center of a cluster of optimal values. With alpha and gamma set to these values the agent usually gets the maximum number of successful trials.

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?

Looking at the instances where there is still negative reward after all the exploration is over (epsilon is equal to zero) is interesting. Below is a similar table as before except in this case the values are the total sum of all negative rewards for all the sets of trials once epsilon reaches zero:



```
     gamma     0.0     0.1     0.2     0.3     0.4     0.5     0.6     0.7     0.8     0.9     1.0
alpha
     0.00 [ -548.0  -549.5  -558.5  -627.0  -636.0  -605.5  -606.5  -584.5  -557.5  -656.5  -540.5]
     0.10 [  -10.0    -4.0    -9.0   -11.5   -11.0   -12.5   -10.0   -20.5   -27.5    -7.5   -24.5]
     0.20 [   -9.5   -11.0    -4.0   -10.0    -8.5   -33.0   -49.0  -113.0  -132.0   -99.0  -118.5]
     0.30 [   -7.0    -4.5   -11.5   -16.5   -10.5   -18.0  -106.5   -68.5   -63.5  -138.5  -313.5]
     0.40 [   -7.5    -1.0   -10.0   -12.0   -18.5    -9.5   -21.0   -66.5  -174.0  -127.5  -301.5]
     0.50 [  -12.5    -4.5    -8.0    -6.5   -13.5   -15.0   -24.5   -45.5   -35.5  -113.5  -368.0]
     0.60 [   -5.0   -11.0    -9.5   -13.0   -18.0    -7.5  -111.0  -114.0   -40.5  -117.5  -307.5]
     0.70 [   -5.5    -7.5    -4.5    -6.5   -19.0   -22.0   -24.0   -29.5   -48.5  -153.5  -247.0]
     0.80 [   -4.0   -10.5    -7.0   -14.0    -9.5   -11.5   -33.0   -24.5  -100.0  -168.0  -542.5]
     0.90 [   -8.5    -5.5    -4.0   -20.5   -13.5   -18.0   -21.0   -34.5  -105.0  -138.0  -750.5]
     1.00 [  -12.0    -5.0    -7.0    -5.0   -14.5   -16.5   -25.5   -36.0   -83.0  -219.5  -621.5]
```

The first interesting thing is that there is still some negative reward for all combinations of alpha and gamma. I also added into the code to print the state information, the reward, and the q table for the current state for instances where epsilon is zero and negative reward was returned. Looking at this information I can see that the q table has not fully converged because there are still values that are 0. For most of the cases where negative reward is returned after exploration is complete there was another vehicle in the intersection, which is a fairly unique case. If you look at the agent operating in the environment, the vast majority of the time there is not another vehicle in the intersection. So I think what is happening is not every unique state/action set is explored since some of them are fairly rare.

Using the definition in the question for optimal policy, no, my agent does not find an optimal policy because it still receives some penalty. If I wanted to get the agent to find the optimal policy without changing the number of trials then I think the best approach would be to try to reduce some of the possible states. Removing the traffic to the right option would be one way to reduce it. Another possibility would be to pre-process the state of an intersection based on my knowledge of traffic laws and create states that are simpler, such as an instance where the light is green but a car is coming from forward and passing straight through the intersection could be encoded as forward = clear, left = yield, right = clear. A scheme like this would dramatically reduce the number of states and increase the likelihood of the q table converging once the exploration phase is complete. But this would be more work and not fully utilizing the power of the learner.

The agent does however almost always reach the destination for all remaining trials once epsilon reaches 0. I would describe the optimal policy as one which gets to the destination successfully within the time limit as often as possible while incurring very little penalty. So using this definition the agent does achieve an optimal policy, or close to it. But as mentioned before the model would probably benefit greatly from a dimensionality reduction for the states.