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

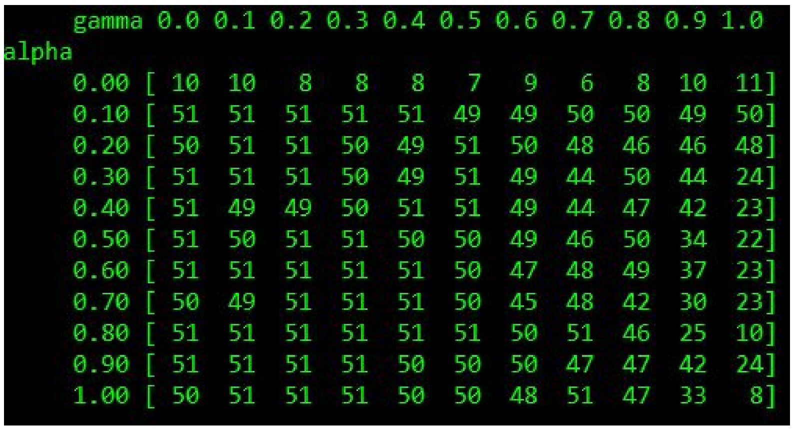
QUESTION: What changes do you notice in the agent's behavior when compared to the basic driving agent when random actions were always taken? Why is this behavior occurring?

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:



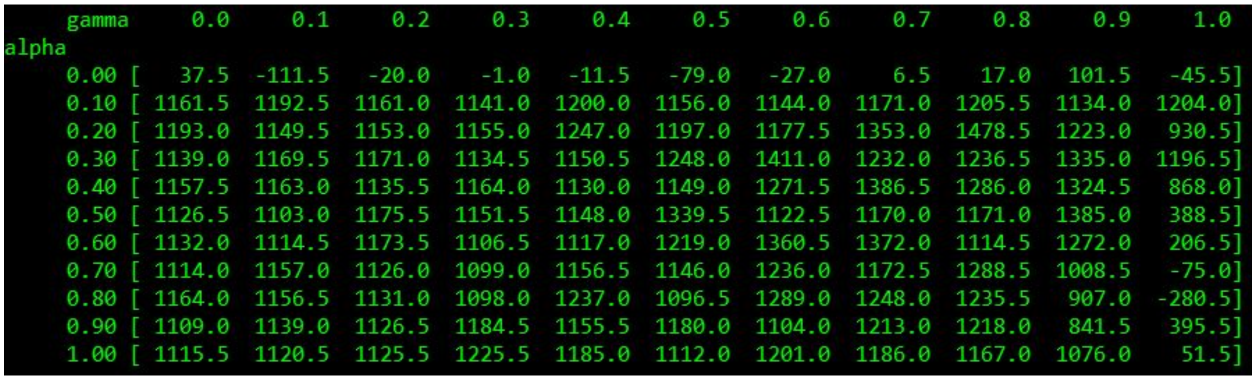
```
     gamma 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
alpha
     0.00 [ 10  10   8   8   8   7   9   6   8  10  11]
     0.10 [ 51  51  51  51  51  49  49  50  50  49  50]
     0.20 [ 50  51  51  50  49  51  50  48  46  46  48]
     0.30 [ 51  51  51  50  49  51  49  44  50  44  24]
     0.40 [ 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?

I would describe the optimal policy as one which gets to the destination successfully within the time limit as often as possible, or more precisely one which always follows the waypoints, unless there is a conflict with another vehicle, then Idle. However there is obviously an argument to be made where the optimal policy is one which simply maximises the reward.

The table below represents the total sum reward for all trials once epsilon reaches zero, as with the previous table:



```
     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 [   37.5  -111.5   -20.0    -1.0   -11.5   -79.0   -27.0     6.5    17.0   101.5   -45.5]
     0.10 [ 1161.5  1192.5  1161.0  1141.0  1200.0  1156.0  1144.0  1171.0  1205.5  1134.0  1204.0]
     0.20 [ 1193.0  1149.5  1153.0  1155.0  1247.0  1197.0  1177.5  1353.0  1478.5  1223.0   930.5]
     0.30 [ 1139.0  1169.5  1171.0  1134.5  1150.5  1248.0  1411.0  1232.0  1236.5  1335.0  1196.5]
     0.40 [ 1157.5  1163.0  1135.5  1164.0  1130.0  1149.0  1271.5  1386.5  1286.0  1324.5   868.0]
     0.50 [ 1126.5  1103.0  1175.5  1151.5  1148.0  1339.5  1122.5  1170.0  1171.0  1385.0   388.5]
     0.60 [ 1132.0  1114.5  1173.5  1106.5  1117.0  1219.0  1360.5  1372.0  1114.5  1272.0   206.5]
     0.70 [ 1114.0  1157.0  1126.0  1099.0  1156.5  1146.0  1236.0  1172.5  1288.5  1008.5   -75.0]
     0.80 [ 1164.0  1156.5  1131.0  1098.0  1237.0  1096.5  1289.0  1248.0  1235.5   907.0  -280.5]
     0.90 [ 1109.0  1139.0  1126.5  1184.5  1155.5  1180.0  1104.0  1213.0  1218.0   841.5   395.5]
     1.00 [ 1115.5  1120.5  1125.5  1225.5  1185.0  1112.0  1201.0  1186.0  1167.0  1076.0    51.5]
```

In this table the area I chose before (alpha = 0.8, gamma = 0.2) is not necessarily in an area with the highest reward. Although when re-run there is a far amount of variability in this table, the area where alpha is 0.4 and gamma varies from 0.6 to 0.9 is consistently higher than the rest of the combinations. Looking at the previous table, we can see this same area has a slightly lower rate of reaching the target than other more optimal combinations.

I think what might be happening is the area where alpha is 0.4 and gamma varies from 0.6 to 0.9 almost always reaches the target so it gets most of the final reward, but in addition it might also be making extra legal moves on its way to the target which get it a little extra reward but ends up taking longer. I would not call that behavior optimal.

Using my original definition of the optimal policy, I believe my adjusted model does reach an optimal policy.