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 smart cab did eventually make it to the point after a lot of moves. One thing I found interesting is that the cab seems to snake around a lot it doesn't really remember states its been in and may return to that same spot. Other cars seem to go father(not distance-wise but displacement-wise) over the same time.

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 have identified 128 states to be a combination of traffic at the light to the left actions, the oncoming traffic action, the direction of the waypoint and the color of the light. defined in code as follows:

```
        self.state = (inputs['left'], inputs['light'], inputs['oncoming'], self.next_waypoint)
```

States are appropriate to our problem because they encompass the norms of driving for our world(traffic rules i.e. lights and right of way) while also always taking into account which direction we ultimately want to go(waypoint).

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 exist 7,776 unique states in this environment. This number does not seem reasonable in this environment. We only have a finite number of moves to reach the goal state. Some of the information is either irreverent or its relevant information can be summed up by other given information. this model would take along time to train.

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?

The agent no longer acts purposeless. It makes mistakes but actively is seeking its target. I only seems to not reach its target on the first trial but is very successful afterwards. The agent starts to repeat correct steps and does not repeat its mistakes. In the driving agent’s first time in a state it is making some mistake and getting some things right. It remembers these things. If the agent find himself in a similar state again he will know what the wrong thing not to do is or the right thing to do.

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?

Let n = trial number.

epsilon	alpha	gamma	performance
1/n	0.9	0.9	91%
1/n	0.9	0.7	96%
1/n	0.9	0.5	84%
1/n	0.9	0.3	87%
1/n	0.9	0.1	56%
1/n	0.7	0.9	92%
1/n	0.7	0.7	98%
1/n	0.7	0.5	71%
1/n	0.7	0.3	45%
1/n	0.7	0.1	57%
1/n	0.5	0.9	97%
1/n	0.5	0.7	91%
1/n	0.5	0.5	91%
1/n	0.5	0.3	81%
1/n	0.3	0.1	55%
1/n	0.3	0.9	96%
1/n	0.3	0.7	98%
1/n	0.3	0.5	94%
1/n	0.3	0.3	79%
1/n	0.3	0.1	40%
0.5/n	0.9	0.9	80%
0.5/n	0.9	0.7	94%
0.5/n	0.9	0.5	92%
0.5/n	0.9	0.3	73%
0.5/n	0.9	0.1	53%
0.5/n	0.7	0.9	92%
0.5/n	0.7	0.7	88%

epsilon	alpha	gamma	performance
0.5/n	0.7	0.5	85%
0.5/n	0.7	0.3	78%
0.5/n	0.7	0.1	91%
0.5/n	0.5	0.9	87%
0.5/n	0.5	0.7	89%
0.5/n	0.5	0.5	83%
0.5/n	0.5	0.3	81%
0.5/n	0.5	0.1	28%
0.5/n	0.3	0.9	82%
0.5/n	0.3	0.7	92%
0.5/n	0.3	0.5	84%
0.5/n	0.3	0.3	90%
0.5/n	0.3	0.1	73%
0	0.9	0.9	93%
0	0.9	0.7	81%
0	0.9	0.5	98%
0	0.9	0.3	87%
0	0.9	0.1	71%
0	0.7	0.9	96%
0	0.7	0.7	99%
0	0.7	0.5	88%
0	0.7	0.3	31%
0	0.7	0.1	70%
0	0.5	0.9	89%
0	0.5	0.7	89%
0	0.5	0.5	92%
0	0.5	0.3	88%
0	0.5	0.1	49%

epsilon	alpha	gamma	performance
0	0.3	0.9	75%
0	0.3	0.7	84%
0	0.3	0.5	93%
0	0.3	0.3	93%
0	0.3	0.1	87%

The set of parameters the agent performs best with is (alpha, epsilon, gamma) = (1, 0, 1)

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?

My agent does find the optimal policy; The algorithm seems to follow path that promises the most reward. I would say the optimal policy is to always go towards the waypoint only stopping at a red light or when going towards the waypoint would break the traffic rules. For example in my last ten trials listed below, my algorithm only makes 1 mistake. That is when my algorithm see the state (None, 'red', None, ‘left') it chooses the ‘right’(wrong) action. It should instead pick the ‘None’(right) action. This seems to be the only time the algorithm makes mistake. The mean sum of rewards for the last 10 trials is 24. This is higher than 0 meaning more right choice were made than false.

```
Simulator.run(): Trial 90
Environment.reset(): Trial set up with start = (4, 5), destination =
(8, 6), deadline = 25
RoutePlanner.route_to(): destination = (8, 6)
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'green', None, 'forward'),forward)->2.0
((None, 'green', None, 'forward'),forward)->2.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'green', None, 'forward'),forward)->2.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'green', None, 'forward'),forward)->2.0
Environment.act(): Primary agent has reached destination!
((None, 'red', None, 'right'),right)->12.0
Simulator.run(): Trial 91
Environment.reset(): Trial set up with start = (1, 6), destination =
(8, 5), deadline = 40
```

```
RoutePlanner.route_to(): destination = (8, 5)
((None, 'red', None, 'left'),right)->-0.5
Environment.act(): Primary agent has reached destination!
((None, 'red', None, 'right'),right)->12.0
Simulator.run(): Trial 92
Environment.reset(): Trial set up with start = (8, 3), destination =
(5, 5), deadline = 25
RoutePlanner.route_to(): destination = (5, 5)
((None, 'green', None, 'right'),right)->2.0
((None, 'green', None, 'right'),right)->2.0
((None, 'green', None, 'forward'),forward)->2.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'green', None, 'forward'),forward)->2.0
((None, 'red', None, 'left'),right)->-0.5
((None, 'red', None, 'right'),right)->2.0
((None, 'red', None, 'right'),right)->2.0
((None, 'red', None, 'right'),right)->2.0
Environment.act(): Primary agent has reached destination!
((None, 'green', None, 'left'),left)->12.0
Simulator.run(): Trial 93
Environment.reset(): Trial set up with start = (6, 4), destination =
(8, 2), deadline = 20
RoutePlanner.route_to(): destination = (8, 2)
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'green', None, 'forward'),forward)->2.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'green', None, 'forward'),forward)->2.0
((None, 'green', None, 'left'),left)->2.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
Environment.act(): Primary agent has reached destination!
((None, 'green', None, 'forward'),forward)->12.0
Simulator.run(): Trial 94
Environment.reset(): Trial set up with start = (2, 3), destination =
(6, 3), deadline = 20
RoutePlanner.route_to(): destination = (6, 3)
((None, 'red', None, 'right'),right)->2.0
((None, 'green', None, 'forward'),forward)->2.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'green', None, 'forward'),forward)->2.0
((None, 'red', None, 'forward'),None)->0.0
Environment.act(): Primary agent has reached destination!
((None, 'green', None, 'forward'),forward)->12.0
```

```
Simulator.run(): Trial 95
Environment.reset(): Trial set up with start = (5, 5), destination =
(8, 1), deadline = 35
RoutePlanner.route_to(): destination = (8, 1)
((None, 'red', None, 'right'),right)->2.0
((None, 'green', None, 'right'),right)->2.0
((None, 'green', None, 'forward'),forward)->2.0
((None, 'green', None, 'forward'),forward)->2.0
((None, 'green', None, 'left'),left)->2.0
((None, 'green', None, 'forward'),forward)->2.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
Environment.act(): Primary agent has reached destination!
((None, 'green', None, 'forward'),forward)->12.0
Simulator.run(): Trial 96
Environment.reset(): Trial set up with start = (6, 5), destination =
(4, 3), deadline = 20
RoutePlanner.route_to(): destination = (4, 3)
((None, 'red', None, 'left'),right)->-0.5
((None, 'green', None, 'right'),right)->2.0
((None, 'red', None, 'right'),right)->2.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'green', None, 'forward'),forward)->2.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'green', None, 'forward'),forward)->2.0
((None, 'green', None, 'right'),right)->2.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'green', None, 'forward'),forward)->2.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
Environment.act(): Primary agent has reached destination!
((None, 'green', None, 'forward'),forward)->12.0
Simulator.run(): Trial 97
Environment.reset(): Trial set up with start = (3, 1), destination =
(7, 3), deadline = 30
RoutePlanner.route_to(): destination = (7, 3)
((None, 'green', None, 'forward'),forward)->2.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'green', None, 'forward'),forward)->2.0
((None, 'green', None, 'forward'),forward)->2.0
((None, 'green', None, 'forward'),forward)->2.0
((None, 'green', None, 'right'),right)->2.0
```

```
Environment.act(): Primary agent has reached destination!
((None, 'green', None, 'forward'),forward)->12.0
Simulator.run(): Trial 98
Environment.reset(): Trial set up with start = (3, 3), destination =
(8, 1), deadline = 35
RoutePlanner.route_to(): destination = (8, 1)
((None, 'green', None, 'right'),right)->2.0
((None, 'green', None, 'forward'),forward)->2.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'green', None, 'forward'),forward)->2.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'green', None, 'forward'),forward)->2.0
((None, 'green', None, 'forward'),forward)->2.0
((None, 'green', None, 'left'),left)->2.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
Environment.act(): Primary agent has reached destination!
((None, 'green', None, ‘forward'),forward)->12.0

Simulator.run(): Trial 99
Environment.reset(): Trial set up with start = (5, 4), destination =
(8, 1), deadline = 30
RoutePlanner.route_to(): destination = (8, 1)
((None, 'green', None, 'right'),right)->2.0
((None, 'red', None, 'right'),right)->2.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'green', 'left', 'forward'),forward)->2.0
((None, 'green', None, 'forward'),forward)->2.0
((None, 'red', None, 'left'),right)->-0.5
((None, 'green', None, 'right'),right)->2.0
((None, 'green', None, 'right'),right)->2.0
((None, 'green', None, 'right'),right)->2.0
((None, 'green', None, 'left'),left)->2.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
((None, 'red', None, 'forward'),None)->0.0
Environment.act(): Primary agent has reached destination!
((None, 'green', None, 'forward'),forward)->12.0
```