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?

Yes, the cab makes it to its destination eventually. It can take a really long time though, and it's actually not guaranteed to ever reach the destination (intuition - it is possible for the random sequence of moves to just all be left turns, all though this is ridiculously unlikely). This is actually limited to 100 steps past the deadline in `environment.py`. The car makes no attempt to minimize the length of its route, or to avoid other vehicles, it just eventually stumbles onto its destination.

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?*

The appropriate parts of the smartcab's state are 'light': the light at the current intersection, 'oncoming': the presence of oncoming traffic, 'left': the presence of traffic approaching from the left, 'next_waypoint': the current direction of travel. 'light' tells us whether or not we can proceed in the current direction or turn left, depending on the actions of other cars. 'oncoming' can restrict whether or not we can turn left given a green light. 'next_waypoint' differentiates the current direction of our movement. With this in the state, going 'left' at an intersection is not the same action as turning 'right' (all other things being equal).

It's worth noting that 'right' is not necessary. On a green light, there is no restriction on our movement due to traffic approaching from the right. On a red light, the only move we're allowed to make is a 'left' turn, but this again is unaffected by approaching traffic from the right.

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 2 states each for 'light', 'oncoming' and 'left'. There are 3 states for 'waypoint'. In total there are 24 possible states. This is a very small number of states, and should be very amenable to Q-learning. It is not inconceivable that the smartcab will pass through each of these states in a single trip.

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 agents movements are much more measured. They are clearly no longer random, as the agent now continues along the same path much more often, and seems to gradually move closer to the target. In general, the agent converges on the destination in much less time than it did with the random action selection.

This is due to the presence of an adaptive policy. The agent will generally act in a manner that has historically maximized utility, while occasionally taking a random action to explore different policies.

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 ran 1000 trials for each setting of parameters.

When I set discount factor (gamma) to 1, learning rate (alpha) to 1, and exploration probability (1 - epsilon) to 1, I get Average reward per trial: 0.083: Average number of steps per trial: 27.707 Missed 80.4% of deadlines

This is equivalent to making random moves

setting exploration rate to 0: Average reward per trial: 2.0065 Average number of steps per trial: 30.476 Missed 99.6% of deadlines

That's really terrible

With exploration rate at 0.5: Average reward per trial: 13.9015 Average number of steps per trial: 23.628 Missed 49.4% of deadlines

With exploration rate at 0.75: Average reward per trial: 6.1475 Average number of steps per trial: 26.672 Missed 72.2% of deadlines

that is pretty poor, so let's explore less. With 0.25: Average reward per trial: 15.449 Average number of steps per trial: 23.185 Missed 45.6% of deadlines

At 0.125: Average reward per trial: 16.1945 Average number of steps per trial: 21.247 Missed 38.8% of deadlines

At 0.0625: Average reward per trial: 16.9675 Average number of steps per trial: 19.008 Missed 29.3% of deadlines

At 0.03125: Average reward per trial: 14.8225 Average number of steps per trial: 20.958 Missed 40.5% of deadlines

At 0.46875: Average reward per trial: 16.9895 Average number of steps per trial: 18.954 Missed 30.8% of deadlines

At 0.05: Average reward per trial: 16.6205 Average number of steps per trial: 20.001 Missed 33.2% of deadlines

Another run at 0.625: Average reward per trial: 20.9805 Average number of steps per trial: 15.593 Missed 9.6% of deadlines

which seems flukishly good. Another run.

Average reward per trial: 17.6285 Average number of steps per trial: 18.676 Missed 28.5% of deadlines

I'll leave that there for now.

Let's fuzz with the discount factor now. At 0.5: Average reward per trial: 21.463 Average number of steps per trial: 14.921 Missed 4.3% of deadlines

Massive improvement!

At 0.25: Average reward per trial: 21.5995 Average number of steps per trial: 14.518 Missed 6.0% of deadlines

At 0.5 again: Average reward per trial: 21.4345 Average number of steps per trial: 14.45 Missed 5.8% of deadlines

At 0.75: Average reward per trial: 22.459 Average number of steps per trial: 14.324 Missed 3.1% of deadlines

is that reproducible? Average reward per trial: 22.7035 Average number of steps per trial: 13.985 Missed 2.4% of deadlines

sweet!

At 0.875: Average reward per trial: 22.6075 Average number of steps per trial: 15.072 Missed 4.6% of deadlines

At 0.625: Average reward per trial: 22.4495 Average number of steps per trial: 14.214 Missed 1.6% of deadlines

and again Average reward per trial: 22.353 Average number of steps per trial: 13.443 Missed 1.9% of deadlines

which is impressive.

And now adjusting the learning rate: At 0.5: Average reward per trial: 22.4925 Average number of steps per trial: 13.712 Missed 1.9% of deadlines

At 0.75: Average reward per trial: 22.407 Average number of steps per trial: 13.88 Missed 1.9% of deadlines

Which is not much different.

At 0.25: Average reward per trial: 22.3855 Average number of steps per trial: 14.398 Missed 2.4% of deadlines

Average reward per trial: 22.436 Average number of steps per trial: 13.343 Missed 1.5% of deadlines

Two divergent arrival rates. One more to confirm: Average reward per trial: 22.5855 Average number of steps per trial: 13.897 Missed 1.7% of deadlines

At 0.125: Average reward per trial: 22.4025 Average number of steps per trial: 13.057 Missed 1.4% of deadlines

At 0.0625: Average reward per trial: 22.293 Average number of steps per trial: 13.717 Missed 1.9% of deadlines

Average reward per trial: 22.4025 Average number of steps per trial: 13.786 Missed 2.4% of deadlines

Average reward per trial: 22.4745 Average number of steps per trial: 13.794 Missed 1.0% of deadlines

Average reward per trial: 22.4745 Average number of steps per trial: 13.794 Missed 1.0% of deadlines

Average reward per trial: 22.298 Average number of steps per trial: 13.391 Missed 1.7% of deadlines

Average reward per trial: 22.2995 Average number of steps per trial: 13.755 Missed 1.2% of deadlines

My final parameter selection is 0.625 for the discount_factor, 0.0625 explore_rate (i.e. 0.9375 epsilon) and 0.0625 learning rate.

The vehicle makes it to its destination before the deadline about 98.5% of the time.

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?*

On our grid, the average distance between 2 points will be 2 in the vertical direction and about 2 in the horizontal direction. That means that if the car going in the right direction, it can get there in 4 moves on average. If going in the wrong direction, the car can turn around by moving in a loop, which requires 4 moves, for a total of about 8 moves. That means that in the average case, the car should be able to reach its destination about 6 moves.

So with 6 moves established as a theoretical lower bound, how well does our smartcab do? With the parameters I ultimately chose, it makes about 22-23 moves ultimately. This is close to 4 times what the car could do if it was omniscient. Our smartcab cannot be omniscient however. Considering that the smartcab is learning its way as it goes along, I think that this performance is reasonable. Recalling that the number of possible states is 24, by the time the cab has made ~22 moves, it can't really have full information about the costs and rewards of its actions yet. The fact that it can on average make it to the destination before it even has had the opportunity to pass through all of these states means it is performing very well. admirable. The fact that the cab only misses it's deadline about 1.5% of the time is also remarkably good.

An optimal policy for this problem would always make the right move, the one maximizes utility or expected reward. In this case, the optimal policy is to always make a move one step closer to the destination, when such a move is legal.