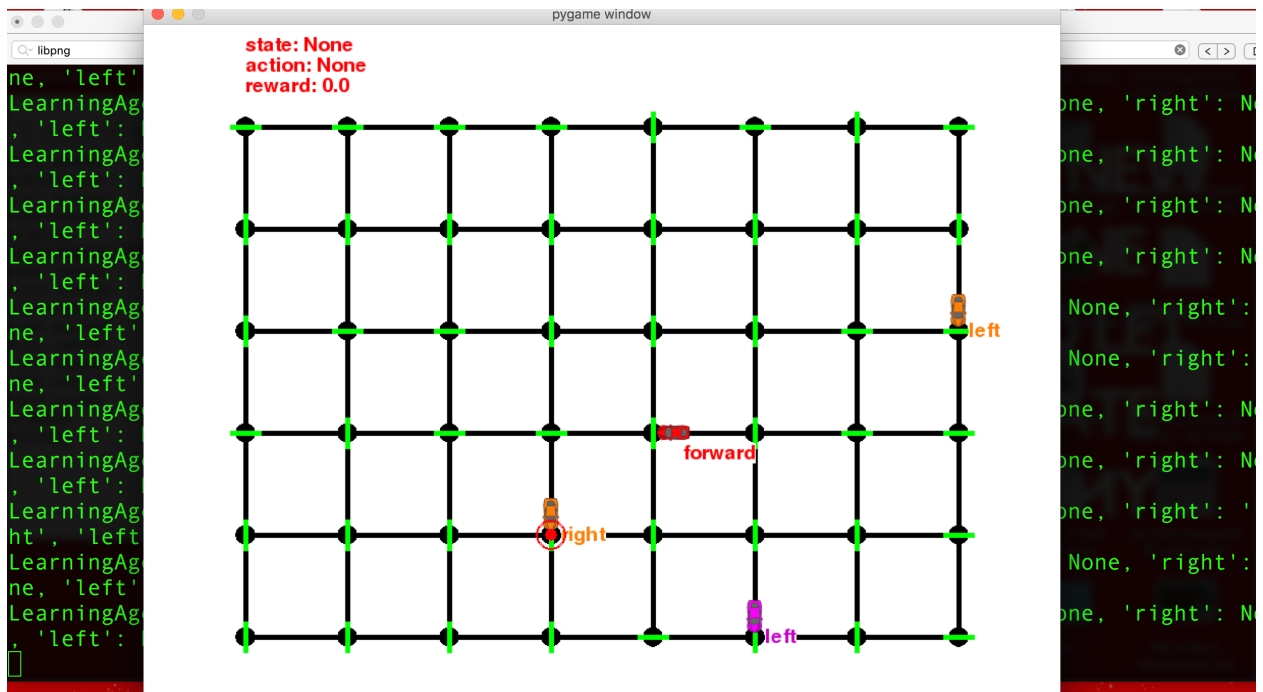


1. **QUESTION:** Observe what you see with the agent's behavior as it takes random actions. Does the **smartcab** eventually make it to the destination? Are there any other interesting observations to note?

ANSWER: The agent eventually makes it to the destination as shown in the image below. However, numerous times a car would be close to the destination and never get there. After a while with no agent getting to the destination the destination would just change, which might be a random restart. Also important to note, it looks like cars can go “off screen.” If the car is at the far left of the grid and takes a left it will wind up on the right side of the grid. That’s something that would need fixing because it doesn’t happen like that in the real world, unless we are assuming this is a 2d visualization of a 3d sphere.



Inform the Driving Agent

2. **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? Each of the grid locations, the light is red or green, if there is traffic coming from the left or right or forward directions or none. In total that is 3 state variables with 48 grid locations, 2 light options, and 4 traffic options. These are important for this problem because each of those variables affect what action is possible for the agent.
3. **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? The total number of states in the smartcab environment is $48 \times 2 \times 4 = 384$ total states are possible. This seems like a reasonable number of states. Because the addition of a new “feature”

or variable that impacts the state will possibly cause the number of possible states to grow exponentially, I would expect a learning algorithm to be able to cope with a very large number of possible states. If the algorithm could not account for multiple features impacting the state, then it seems the user would have to leave out a lot of information that would otherwise be important in order to simplify the problem for the algorithm.

Implement a Q-Learning Driving Agent

4. 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 seemed to move more frequently instead of staying at one position for multiple turns. This enabled the agent to experience more possible states in the given time frame. As it explores more states it begins to find the destination more efficiently.

Improve the Q-Learning Driving Agent

5. 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?

*to determine how many times the cab reached the destination I printed out the log to a text file by running the python file in terminal using the command 'python agent.py > log.txt, then using search to see how many times "reached destination!" appeared. To determine how many steps were taken overall I searched for "LearningAgent.update():"

epsilon = 0.15
alpha = 0.7
gamma = 0.2
success = 95/100
steps= 1,568
penalties = 208

epsilon = 0.3 ← changed from 0.15
alpha = 0.7
gamma = 0.2
success = 94/100 (decrease success rate)
steps= 1,640 (Increase # of steps)
penalties = 192

epsilon = 0.15

alpha = 0.9 ← changed from 0.7
gamma = 0.2
success = 94/100 (decrease success rate)
steps= 1,534 (decrease # of steps)
penalties = 220

epsilon = 0.15
alpha = 0.9
gamma = 0.8 ← changed from 0.2
success = 81/100 (decrease success rate)
steps= 1,842 (Increase # of steps)
penalties = 461

epsilon = 0.1 ← changed from 0.15
alpha = 0.8 ← changed from 0.7
gamma = 0.1 ← changed from 0.2
success = 97/100 (increase success rate)
steps= 1,572 (decrease # of steps)
penalties = 173

Best Performance

epsilon = 0.05 ← changed from 0.15
alpha = 0.9 ← changed from 0.7
gamma = 0.05 ← changed from 0.2
success = 99/100 (increase success rate)
steps= 1,470 (decrease # of steps)
penalties = 91

6. 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?

The agent does use less steps than the other agents and variables tested. It also incurs the least penalties. There are multiple ways to describe the optimal policy, one is the least amount of lefts and rights (turns), another is the least number of mistakes/penalties, another is the highest success rate, and another is the least number of steps taken. However, I personally would prefer the least number of turns.