Reinforcement Learning Project Report
Nathan Mayer

Question 1: ***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 behaves very erratically, as all of its decisions are being made randomly. Because of this random process, the smartcab does eventually find its way to the destination, not because of any intelligence, but because it is randomly navigating through a relatively small grid. Eventually it just happens to run into the destination by accident.

The car is frequently penalized for running into other cars and for not following the directions, but it also occasionally receives rewards for moving in the right direction. The car never learns from these rewards or penalties, however.

Question 2: ***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 identified all of the input as appropriate for modeling the smartcab and its environment. In addition to the input provided, I also used the coordinates for the smartcab and the destination to produce two boolean values that described whether the smartcab was above or below the destination and whether the smartcab was to the left or right of the destination. I also included the car's heading.

I believe these states are appropriate for solving the problem because they all communicate useful information about the system in ways that add relatively little complexity to the state for predictions. I reduced the complexity of the state by converting location and destination data into two simple boolean values, making it more feasible for the program to learn over the course of a 100-trial experiment. I chose not to further simplify traffic values and direction values because in doing so I would be writing rules for the system to learn from, rather than allowing the system to learn from the data and infer the rules. While writing those rules would have likely resulted in better performance, the problem of inference (is it safe to turn? Am I going the right way?) seemed more interesting to solve with reinforcement.

Question 3: ***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?***

When the simulation begins, the agent's behavior is very similar to the random actions taken by the previous agent. This is because all decisions are equally weighted and there is no reason to choose one direction over another. There are many punishments given and the agent cannot find the destination before the deadline.

As the simulation progresses, there are fewer and fewer punishments given and more and more rewards given. The agent can find the destination sometimes, and its driving pattern is not as erratic. This is because the agent is learning weights for its various transitions and is figuring out what transitions result in the biggest rewards (reaching the destination).

By the end of the 100-trial experiment, the agent performs much better than it did at the start, but often gets stuck at intersections, choosing not to make any moves. This is because it has weighted all other directions lower than the value of just sitting still. It doesn't always find the destination, but does find it fairly regularly. This is because the agent has learned the value of finding the destination, but isn't optimistic enough to find better ways of getting there.

Question 4: ***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 discount the estimate of maximum value for the next state by a factor of 0.9.
I use a learning factor alpha of 0.5.
I initialize the system optimistically, with every transition having a value of 5.
This set of parameters was the most performant set that I could derive.

Near the end of the simulation, the agent has learned weights for most or all of the possible states of the system. Those weights are still not as well-tuned as they should be, so the agent often makes loops and hesitates at intersections. It makes it to the destination nearly every time, however, and makes very few invalid moves (less than 1 per round). At this point, the agent has a good model of the system, and can find its way to the destination safely and reliably (though still sub-optimally). In a 100-trial experiment, it reaches the destination 75 times and makes ~120 invalid moves.

Question 5: ***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?***

An optimal policy for this problem would be a policy which does not result in any traffic violations, takes the shortest route possible (according to the route planner), and always gets to the destination on time. Because of the large reward for getting to the destination on time, there would sometimes be a case where it is most optimal to incur a penalty, if

there is little time remaining and there is another car that is preventing you from taking the shortest path.

My agent does not come close to finding an optimal policy in the 100-trial runs. It often makes loops and wrong turns, although it very rarely incurs a traffic penalty and always reaches the destination in the last 25% or so of trials. On 500-trial and 1000-trial runs, the agent comes much closer to finding an optimal policy, very quickly reaching its destination and rarely making wrong turns.