Train a Smartcab How to Drive

Michael Recachinas
Udacity Machine Learning Nanodegree Program
HW 4

Basic Driving Agent

The agent's behavior is simply random. Over 10 trials, the agent eventually made it to the destination in 4 out of those 10 trials albeit coincidentally. No learning is performed across trials because the agent is simply choosing randomly among 4 possible actions without any knowledge of state.

Informed Driving Agent

The states I've identified that are appropriate for modeling the smartcab and environment are

- `next_waypoint` is important because we are primarily concerned about where we should move next (with respect to the end destination). For a real smart car, this would be akin to a GPS system.
- `light` is important primarily because of safety, but also, we need to determine what we should do in the presence of a red vs green light (e.g., is it more advantageous to turn right on red, than it might be to wait to turn left, etc.).
- `oncoming` is important because it more accurately models the environment with regard to safety. Because it's relatively rare that the agent meets other cars at an intersection and the smaller state space could enable the agent to learn faster, it does not contribute towards a higher cumulative reward. That said, a driving agent should always prioritize safety, so adding `oncoming` to learn the U.S. Right of Way rules (e.g., wait for someone before making a right on red, wait to ensure no incoming traffic before making a left on green, etc.) is important. Specifically, I chose to use `oncoming` and `left` as the inputs for this state because if the light is green and oncoming traffic is moving forward, then we should give right of way to the oncoming traffic before making a left. Likewise, if the light is red, and the left side is moving forward, we should give right of way before making a right turn. I did not include the right side because there is no "left on red" and if the agent has a green light, that implies the right has a red light, thereby yielding to the agent's right of way.

`deadline` is another state that I considered, but realized isn't as important as the `next_waypoint`, and the discount factor already penalizes late arrivals/missed deadlines and rewards early arrivals. Adding `deadline` would greatly increase the size of the state space by a factor of the max deadline (set to 50) for each current state. Therefore, in total, the state space would increase from 96 to 4800. This would greatly reduce the rate at which the agent learns for trivial gains for which are already accounted, and in some cases, would likely encourage the agent to break traffic rules as the deadline is approaching.

Q-Learning Driving Agent

The agent's behavior when compared to the basic driving agent when random actions were always taken became less unpredictable or *random*. Each successive trial saw the agent start to move toward the destination and being positively rewarded as opposed to moving away and being penalized. This behavior is occurring because of Q-Learning, where the agent determines an optimal action-selection policy based on state-action pairs.

The different values for the parameters tuned in my basic implementation of Q-Learning are the discount factor and learning rate. The following illustrates various values tested with the number of penalties and successful arrivals.

Discount Factor	Learning Rate	Arrival/Total	Num Actions/Actions Avail.	Total Penalties
0.6	0.25	86/100	1604/6416	488
0.4	0.25	88/100	1442/5768	259
0.5	0.35	87/100	1593/6372	474
0.25	0.4	94/100	1451/5804	96

The set of parameters that optimizes the performance of the agent are therefore a learning rate of 0.4 and discount factor of 0.25. That being said, it's important to note that it does have a non-trivial amount of penalties across its 100 trials (96) – if this is simply taking a turn away from the destination, this is acceptable at the beginning; however, if this is running a red light or thereabouts, then this is unacceptable. That said, it is important to note that this learning would likely take place with a human supervisor, and therefore mistakes are acceptable during the early trials. The final driving agent performs at a rate of 94/100 successful arrivals at 1451 actions taken out of 5804 total actions.

The agent gets close to finding an optimal policy, i.e., reaching the destination in the minimum possible time, and not incurring any penalties, but it does not achieve it. For example, because the agent rarely meets other cars at an intersection, it has a hard time learning when to yield. An optimal policy for this problem would be discovered with more trials.