Randomly-Acting Agent

The randomly-acting agent may or may not reach the goal in any length of time depending on which actions it takes. The directions on the screen next to each vehicle appear to indicate the next waypoints calculated by the planner. If the agent does follow some of the correct waypoints it may move closer to the goal and improve the probability that it will arrive there by chance.

The dummy agents' waypoints do not appear to be steering them towards the same goal as the primary agent, so they are not all converging on the same point. They do seem to be following the waypoints given to them and not taking actions that diverge from these waypoints.

Smartcab States

Each state is represented using five variables: the next waypoint, the traffic light input, and the input from three directions of traffic (oncoming, left, and right in that order). The state ('forward', 'red,' None, 'right', None) indicates that the next waypoint is 'forward,' the light is red, and there is a car approaching from the left and turning right.

The waypoint direction is the most important piece of information for the smartcab to learn, as it indicates the correct action for it to take in circumstances barring a red light or oncoming traffic. The traffic light and vehicle inputs allow it to learn when it should override the waypoint information and take a different action such as waiting.

The deadline information is not represented in the state data because the agent does not necessarily need to behave differently depending on the time it has left. It should be finding the optimal path to the goal regardless of the time remaining.

This method of representing states results in 512 (4 x 2 x 4 x 4 x 4) unique possible states for the smartcab to be in. This is likely excessive and some states could be combined with others in a more succinct representation. It is not obvious that all the state variables in this representation need to be given equal weight in the agent's decisions.

Q-Learning Agent

The learning agent initially takes random actions to explore its environment. Its behavior for the first few trials does not appear significantly different from the randomly-acting agent's behavior. As it tabulates rewards and updates Q-values for each state-action pair, correct actions become more highly weighted and the smartcab chooses these actions more often.

As the smartcab learns, its behavior becomes more predictable and it soon reliably finds and executes better paths to the destination. It begins to head in the direction of the destination faster and by taking fewer steps. It learns to wait at red lights instead of trying to run them or

make unnecessary turns to avoid them. It also learns to wait for traffic to leave an intersection instead of causing accidents or turning unnecessarily to avoid it.

When the smartcab chooses to take a correct action and receives the reward, the Q-value for that state-action pair is increased in the Q table and the smartcab is then more likely to take that action in the future when in that state. As it continues to update rewards and future utility values, the Q table converges to its final values. When this happens, it will allow the smartcab to behave optimally.

Improving the Q-Learning Agent

The learning rate alpha ranges from 0 to 1. A learning rate of zero will have the agent learn nothing, and a learning rate of 1 will have the agent learn policies entirely specific to a fully deterministic environment.

The discount rate gamma ranges from 0 to 1. A discount rate of zero will have the agent value immediate rewards over long-term rewards, and a discount rate of 1 will have it assign a greater value to long-term rewards and overall utility.

The exploration rate epsilon ranges from 0 to 1. When the agent acts, there will be a probability equal to epsilon that it will take a random (exploratory) action, and a probability equal to 1 - epsilon that it will take the current highest Q-valued action for its current state. Epsilon exists to prevent the agent from getting stuck on local maxima and should decay over time as the agent learns more about its environment and the Q table converges on its final optimal values.

The epsilon value decays over time depending on the number of trials the smartcab has run. The formula I used divides the value of epsilon by (1 + trials run / 100). This decays epsilon relatively quickly so the ideal epsilon value starts slightly high at about .3. Using an initial epsilon that was too low would cause it to decay too fast and would not allow for enough exploration. By the end of a large number of trials it will decay to near zero, at which point the agent does not need to make random moves but has come close to figuring out the optimal actions to take.

Another way to decay epsilon would not be based on the number of trials run but could be achieved by adding scaled values to the Q values that were not chosen so they are more likely to be chosen next time. This gives the advantage of having the agent's choices depend more on the Q values and its learning process and not on a totally random value.

The following parameter combinations were used to test the smartcab:

	alpha	gamma	epsilon	Mean no. of steps	Failed trials
Test 1	0.8	0.8	0.2	13	6

Test 2	0.5	0.8	0.3	20	11
Test 3	0.8	0.4	0.3	11	2
Test 4	0.5	0.4	0.3	17	8
Test 5	0.8	0.2	0.2	14	1

The chart above shows the approximate mean number of time steps it took the smartcab to attempt to reach the goal in 100 trials for each test, including aborted trials. The number of trials that were aborted when time ran out are reported in the last column.

The smartcab with an alpha value of 0.8, a gamma value of 0.4, and an epsilon value of 0.3 appears to be the fastest at learning an optimal policy. It made fewer mistakes than the other iterations after the first 20 trials and had come close to behaving optimally by the final trials. It arrived at the destination in approximately 10-12 time steps on average and failed to reach the destination in 2 trials out of 100. Its performance improved the longer it ran.

The smartcabs with alpha values of 0.5 performed much worse than the iterations with higher alpha values. Because the smartcab's environment is largely deterministic, a high alpha value allows it to retain more information in order to learn the environment's rules and develop optimal policies quickly.

While learning an optimal policy, the smartcab often makes the mistake of choosing to take a wrong turn, taking a penalty of -0.5 points, and gaining three "correct" turns worth 2.0 points each, resulting in a net gain of 5.5 points, as opposed to 2.0 points for taking the correct action in the first place. A way to prevent this behavior from occurring and having to be corrected might be to incorporate the deadline into the state information and use it in the decision process.

Another way to correct this unnecessary turning would be to lower the gamma value and have the agent consider current rewards more heavily than future rewards. A gamma value of 0.4 appears to perform better than a gamma value of 0.8 in these tests. It could also be possible to lower gamma to zero and not consider future rewards at all, since the agent only needs to follow the waypoint direction and not violate traffic rules to behave optimally, and could accomplish this even if it ignored all future utility.

A smartcab will have found an optimal driving policy when the Q table has converged to its final values. As the Q values are updated, it gets closer to this optimal policy. An optimal policy in the driving problem would result in arriving at the destination in the minimum time by making the minimum number of moves, avoiding other vehicles and obeying all traffic rules.

The best-performing agents in these tests were almost always following the rules of the road in the last 10-20 trials. They had learned to stop and wait at red lights instead of running them or

trying to turn to avoid them. They sometimes still had trouble when meeting other cars at the intersection and would try to turn to avoid them instead of waiting for them to leave the intersection. Since the smartcab rarely meets other cars at intersections, it has fewer chances to learn the correct policy for dealing with them. This is another problem that could be adjusted by lowering gamma, since there is a penalty for taking a wrong turn but not for choosing to wait at the intersection.