Basic Driving Agent

When actions are chosen purely by random and the agent is given unlimited time to reach the destination, it does eventually reach it. However, this is purely a matter of chance.

Indentifying and Updating State

Relevant state variables:

- Direction of next waypoint: `[None, 'forward', 'left', 'right']`
- Does the agent have the right of way to proceed forward? `[True, False]`
- Does the agent have the right of way to turn right? `[True, False]`
- Does the agent have the right of way to turn left? `[True, False]`

Valid actions: `[None, 'forward', 'left', 'right']`

In this simplistic model, the smartcab cares about two things: first, the location of the next waypoint; and second, its ability (right of way) to execute its valid actions.

At first I considered simply using the inputs as returned by `Environment::sense()`, but that state space seemed too large. There are four possible values for the next waypoint, two for the light status, and four for each of the oncoming, left, and right traffic state. That's 512 possible states which, with four possible actions, makes 2048 possible state/action pairs that need to be visited for Q-learning to converge.

I therefore decided to "compress" the information regarding light and traffic status into three binary "right of way" inputs. This results in 32 possible states and 128 possible state/action pairs, a reduction of more than an order of magnitude.

Implementing Q-Learning

My Q-learning model is initialized uniformly with a value of 1 for all state/action pairs. The learning rate is `1/t` and the discount factor is 0.5. It has a 10% chance to return a random action other than the one with the highest Q-value.

The agent now consistently learns to reach the destination in time. For the first several trials it often fails to do so, but by trial 25 this trend shifts. By trial 50 it is almost always reaching the destination in time, and by trial 100 it is rarely receiving any penalties.

Optimizing the Agent

I was seeing some variance in the performance of the agent over several 100-trial runs, so I reduced the random action chance from 10% to 5%, which seemed to help with consistency.

Overall, I believe the agent performs very well. It consistently reaches the destination in time, for both short and long trips, and it usually receives either no or only one penalty during the trip.

Though I have seen several instances where the agent reaches the destination in the minimum possible time (assuming perfect luck with the lights), I believe its directness could be improved further.