Machine Learning Engineer Nanodegree

Project 4: Train a Smartcab to Drive

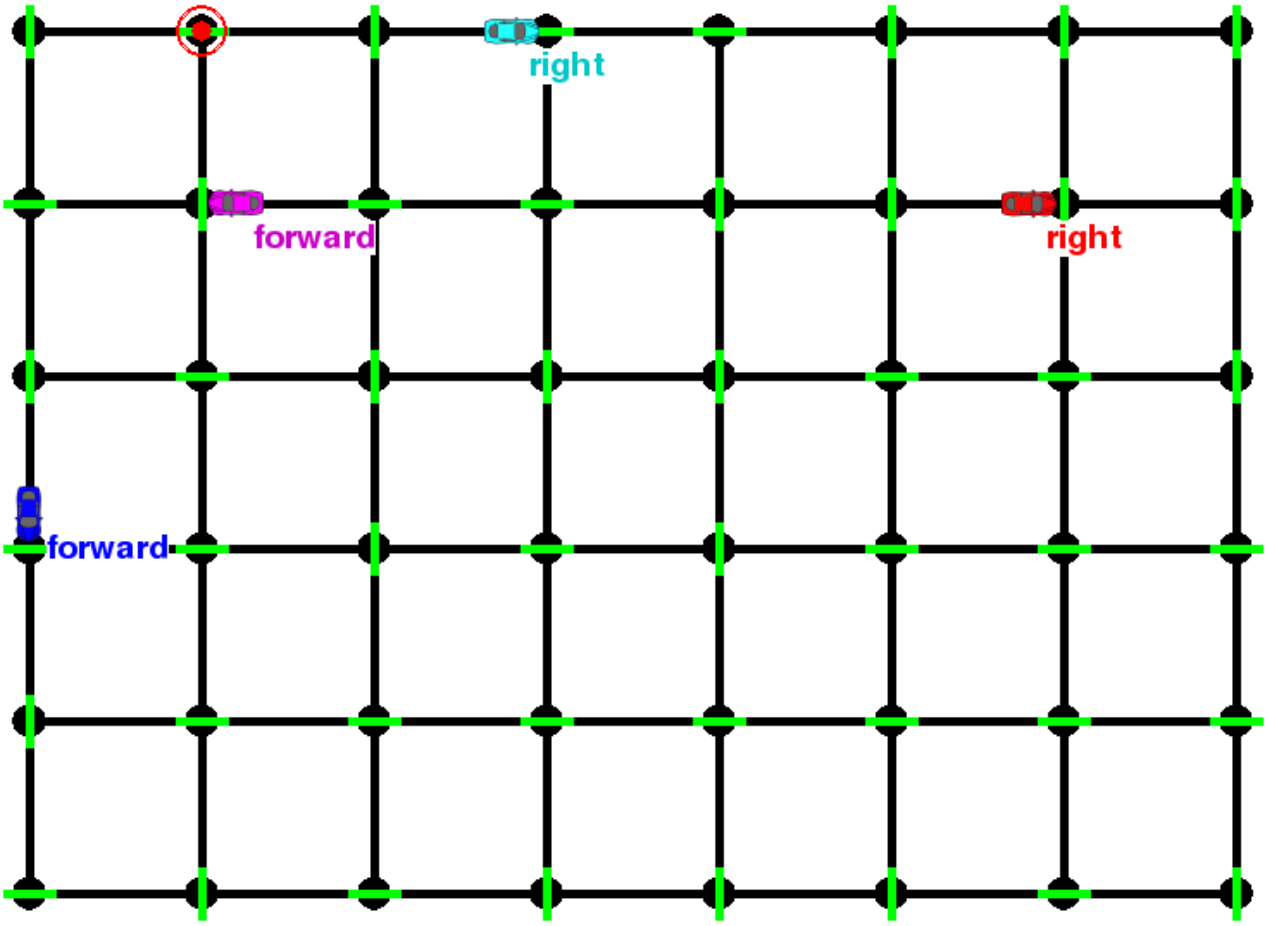
Introduction

This document presents results for the forth project within the Machine Learning Engineer Nanodegree program. This assessment required the student to design an Artificial Intelligence (AI) driving agent for a Smartcab, so that it may successfully complete a trip between two waypoints.

Environment

The Smartcab (agent) is randomly placed on a 5x7 grid which represents possible travel paths (see below). A destination marker is placed on the grid, which the agent must arrive to.

Figure 1: Possible travel paths



In order to 'successfully' arrive at the destination, the agent must make a number of considerations. First, at each intersection, there is a traffic light. Second, there are a number of other cars also travelling along the possible grid paths. And finally, there is a count of the amount of steps taken for the agent to arrive at the destination.

Implement a basic driving agent

The first part of this assessment involved implementing a basic driving agent. The basic agent is to be able to accept a number of specified inputs and produce a valid output in response to those inputs.

Inputs passed to the basic agent include:

- Position of waypoint in relation to agent: forward, left, or right
- Traffic light color: red or green

Outputs which can be taken by the agent include:

- Move forward: forward
- Move left: left
- Move right: right
- Do noting: none

For the basic agent, output selection was random. That is, although the above listed inputs were passed to the agent, the agent would make no consideration of these inputs and would instead randomly select between the above listed outputs.

As expected, when the choice of action is random, the agent will regularly ignore traffic signals and ignore other vehicles. And while the basic agent does occasionally reach the destination by chance, the agent often requires a large amount of trials to do so. However, the lack of learning for the basic agent does have the advantage of allowing full (random) exploration of the entire range of input combinations.

Identify and update state

The second part of this assessment required the student to identify states that model the driving agent and environment, and have the agent update its state when running, based on the current input.

For this part of the assessment, I expanded the set of inputs used to represent the current 'state space'. These final set of inputs passed to the agent include:

- Position of waypoint in relation to agent: forward, left, or right
- Traffic light color: red or green
- Direction of oncoming traffic: no traffic, forward, left, or right
- Direction of right traffic: no traffic, forward, left, or right
- Direction of left traffic: no traffic, forward, left, or right

There are advantages to keeping the state space small as it reduces the processing and memory requirements. However I came to the conclusion that it was necessary to include the above inputs in order to provide the agent with the necessary 'state space' to successfully arrive at the destination.

The position waypoint is passed by the RoutePlanner as a direction to the final waypoint in relation to the agent. Passing this input will allow the agent to understand if its actions are reducing the distance to the final waypoint. In order to avoid running red lights, the traffic light color is also passed to the agent. And finally, the direction of traffic which is oncoming, or to the right or left of the agent was also accounted for in order to avoid traffic collisions.

Implement Q-Learning

The third part of this assessment required the student to implement a Q-Learning algorithm/ matrix which the agent is able to update according to a reward metric which is provided for the agents selected action.

Q-Learning is a model-free reinforcement learning technique. Specifically, Q-Learning can be used to find an optimal action-selection policy for any given (finite) Markov Decision Process (MDP). The reward metric is a critical element for the Q-Learning algorithm. The agent gets a reward for each correct move executed at an intersection. The reward is not always positive however, with the agent receiving a small penalty for an incorrect move, and a larger penalty for violating traffic rules and/or causing an accident.

In summary, an action made by the agent:

- that obeys the rules and moves towards the destination receives a reward of 2
- that obeys the rules but fails to move toward the destination receives a reward of 0.5
- to remain stationary generally leads to a reward of 1
- that does not obey the rules results in a negative reward

For this part of the assessment, a Q-matrix was implemented with each possible state space combination as row and each possible action as column. As such, the Q-matrix represents all possible state-action combinations.

Table 1: Q-matrix (first five records)

	Forward	Left	Right	None
('forward', 'green', None, None, None)	1	1	1	1
('left', 'green', None, None, None)	1	1	1	1
('right', 'green', None, None, None)	1	1	1	1
(None, 'green', None, None, None)	1	1	1	1
('forward', 'red', None, None, None)	1	1	1	1

Each action taken by the agent updates the Q-matrix according to the reward metric provided to the agent. To select an action, the best (highest Q-value) action for the given state is returned according to the Q-matrix.

Results from each iteration after implementing the Q-Learning process suggest learning. The agent can be observed to learn optimal actions for each state. For example, learning to ‘do nothing’ when the traffic light color is ‘red’. And while the agent is rarely able to reach the destination by the deadline during early trials, the agent is able to regularly reach the destination by the deadline within later trials.

Enhance the driving agent

The final part of this assessment required the student to make improvements to the basic Q-Learning implementation, in order to have the agent learn a feasible policy within 100 trials.

The Q-Learning process includes three influence variables:

- Learning rate: alpha
- Discount factor: gamma
- Initial condition: q-values

The learning rate determines to what extent the newly acquired information will override the old information. A factor of 0 will make the agent not learn anything, while a factor of 1 would make the agent consider only the most recent information. The discount factor on the other hand, determines the importance of future rewards. A factor of 0 will make the agent ‘myopic’, only

considering current rewards, while a factor closer to 1 will make the agent strive for a long-term high reward. The initial condition of the Q-matrix is another consideration. High initial values, also known as ‘optimistic initial conditions’, can encourage exploration, that is, no matter what action is selected, the update rule will cause it to have lower values than the other alternative, thus increasing their choice probability.

The final agent

A number of trials were run with various learning rate and discount factor settings. From the experiments, it became obvious that a low discount factor would lead to improved choices, while a high learning rate, would lead to rather fast learning in early trials but seem to ignore learning from past trials.

The number of successful trials from varying alpha, gamma and the q-value are shown in the tables below.

Table 2: Successful trials (when varying alpha)

alpha	gamma	q-value	successful trials
0.2	0.5	10	61
0.4	0.5	10	67
0.6	0.5	10	72
0.8*	0.5	10	76
1.0	0.5	10	71

Table 3: Successful trials (when varying gamma)

alpha	gamma	q-value	successful trials
0.5	0.2	10	73
0.5	0.4*	10	77
0.5	0.6	10	72
0.5	0.8	10	69
0.5	1.0	10	68

Table 4: Successful trials (when varying q-value)

alpha	gamma	q-value	successful trials
0.5	0.5	0	53
0.5	0.5	5	61
0.5	0.5	10	71
0.5	0.5	15*	83
0.5	0.5	30	79

Ultimately, I settled on a learning rate (alpha) of 0.8, a discount factor (gamma) of 0.4, and a q-value of 15. After a number of trials, I found the agent will often find a superior policy, resulting in low travel times and low penalties. However, I am unsure as to whether the ‘optimal’ policy has been found, as the agent will still, on rare occasions, incur a penalty or travel in an incorrect direction.