MLEND Capstone Project: Training a Smartcab Planner

Lucas Murtinho
lucas.murtinho@gmail.com

2016-Jul-20

Contents

1 Definition **2**
1.1 Project Overview . . . 2
1.2 Problem Statement . . . 2
1.3 Metrics . . . 2

2 Analysis **3**
2.1 Data Exploration . . . 3
2.2 Exploratory Visualization . . . 3
2.3 Algorithm and Techniques . . . 3
2.4 Benchmark . . . 3

3 Methodology **3**
3.1 Data Preprocessing . . . 3
3.2 Implementation . . . 3
3.3 Refinement . . . 3

4 Results **3**
4.1 Model Evaluation and Validation . . . 3
4.2 Justification . . . 3

5 Conclusion **3**
5.1 Free-Form Visualization . . . 3
5.2 Reflection . . . 3
5.3 Improvement . . . 3

1 Definition

1.1 Project Overview

In Project 4 of Udacity's Machine Learning Engineer Nanodegree, I had to teach a smartcab in grid-like world how to get to a destination on time by obeying traffic rules and following the directions passed by a planner. Without any kind of hard programming, the agent should learn right-of-way rules and to go straight when the planner sent a "forward" input, for instance.

A natural extension of this project is, instead of relying on an outside planner, coming up with an agent that, given the smartcab's location and destination, learns how to plan a route. This is the goal of my capstone project.

1.2 Problem Statement

The problem at hand is to program an agent that learns to identify what the next waypoint should be for a smartcab in a grid-like world to reach its destination as fast as possible.

In a grid-like world, each position is defined by an (i, j) tuple, in whic i is the longitude (the position across the East-West axis) and j is the latitude (the position across the North-South axis). In a $m \times n$ grid, $(1, 1)$ represents the northwesternmost position, while (m, n) represents the southeasternmost position.

The goal, then, is that, given a position tuple (i_{cab}, j_{cab}), a destination tuple (i_{dest}, j_{dest}), and a heading (described below), the planner should be able to come up with the best next action for the smartcab: `forward`, `right`, or `left`.

I'll use an 8×6 grid for the project, similar to the one used for the Nanodegree Project 4, but in principle the solution should apply to a grid of any size.

1.3 Metrics

The goal of the planner is to come up with the best possible action for the smartcab at all times. As we will see, it is not hard to write a planner that always comes up with the best next waypoint (not taking into account eventualities such as red lights or traffic jams). I'll present such a "perfect planner" in the Benchmark section below. The main metric I'll use to evaluate my learning agent, then, is the **rate of agreement** with the perfect planner.

However, it is possible that a learning agent will come up with a different action that is just as good, or even better, than a "perfect planner" would. Therefore, I'll also keep track of metrics such as the number of steps needed for the planner to reach the deadline (again disconsidering the possibility of red lights or traffic; i.e., the agent will always be able to move), and also, more generally, the number of destinations reached.

Finally, I'll also keep track of the sum of rewards received by the agent. However, as we'll see in the Algorithm and Techniques section below, the rewards are just a means to an end: they exist to get the agent to learn the desired behavior, and its accumulation only matters inasmuch as it helps with this goal.

2 Analysis

2.1 Data Exploration

2.2 Exploratory Visualization

2.3 Algorithm and Techniques

2.4 Benchmark

3 Methodology

3.1 Data Preprocessing

3.2 Implementation

3.3 Refinement

4 Results

4.1 Model Evaluation and Validation

4.2 Justification

5 Conclusion

5.1 Free-Form Visualization

5.2 Reflection

Does agent consider if it's easier to turn right or left (or go forward)?

5.3 Improvement

Improvement: put planner and smartcab together to learn how to play around traffic or red lights