Taxi in New York City: Generating Lucrative Passenger Pick-up Strategy

Julian Gao (julianyg)[1] and Qiujiang Jin (qiujiang)[2]

[1]Stanford University
julianyg@stanford.edu
[2]Stanford University
qiujiang@stanford.edu

I. Introduction

New York City, one of the most famous metropolis in the world, has an enormous collection of human activities at all time. Among those activities, one of the most significant symbols, the Yellow Cab, is representative of population and capital flow in the city. According to the NYC taxi cab fact book[], there are over 485,000 taxi trips and 600,000 passengers per day during year 2014. These numbers, however, are going down due to the impact from other car-hailing platforms such as Uber and Lyft. In order to maximize the profit for traditional Yellow Cab drivers and to compete with car-hailing companies, we propose a software framework to generate a set of optimized passenger pick-up strategies. The framework combines methods of machine learning, data mining, and Markov Decision Process (MDP). We show that our method, when comparing with average hourly income from real data, performs much better. The framework we propose also applies to other user cases, which brings more social impacts other than helping NYC taxi drivers.

II. Related Works

There are many previous works on NYC taxi data set. However, they are mainly focused on analysis and prediction aspects of data. Chriswhong visualizes the movement and earnings of a taxi in 24 hours[]. Daulton *et al.* uses machine learning methods to predict pick-up density and drop-off locations[]. Tipsy Taxi reveals the fraud behavior, and performs traffic analysis, tip analysis for drivers[]. There are studies on taxi strategies, with binary behavior recommendation such as "hunting" or "waiting". Li *et al.* and Tang *et al.* uses data with taxi occupancy label[], which is not available on NYC data set. Yuan *et al.* proposes and relies on parking places detection method, but targets on saving cruise time and maximizing trip distance[].

III. Framework

A. Data Set

We use data directly from NYC Taxi & Limousine Commission (TLC) official website[]. Starting from 2009, the TLC publishes the Yellow, Green, and FHV (For-Hire Vehicle) trip sheet data in CSV format. Each sheet contains trip records in a specific month, where each record is composed by following information: VendorID, TPEP pick-up time, TPEP drop-off time, passenger count, trip distance, pick-up location, rate code ID, store and forward flag, drop-off location, payment type, fare amount, extra, MTA tax, improvement surcharge, tip amount, tolls amount, and total amount[]. The TLC cannot guarantee or confirm the accuracy or completeness of data.

B. Software Pipeline

The framework is composed by two parts: policy (pick-up strategy) generation (1) and trip simulation (2). The policy generation part can be further divided into two stages: feature extraction (1.1), and MDP value iteration (1.2). Each stage is an independent module, where generated parameter files can be stored and distributed, for convenience of reuse and computation, also serving sources for other data-related analysis.

IV. Methods

A. Assumptions & Simplifications

To simplify the problem, we make the following set of assumptions:

1) Locations are represented by grids in the form of tuples $((\text{lon}_W, \text{lon}_E), (\text{lat}_S, \text{lat}_N))$. Instead of using accurate GPS coordinates for locations, we choose to gridify the map region to reduce discreteness in location data. The grid size is controlled by the grid factor g, which is a trade-off between accuracy and complexity. The point coordinate of a grid is represented by that of its center point. One downside of using grids is that in baseline

policy, some non-reachable locations such as water area are included as states.

2) Region boundary $\mathcal{B}$ is reasonable. To reduce complexity and number of states, we use a boundary to exclude locations. If $\mathcal{B}$ is set too small, the generated policy may have not enough choices for driver to choose from.
3) Taxi drivers are motivated by rewards, and will not pick-up passengers on the way to next target location. We later show in section C that the latter part, while seemingly counter-intuitive, is reasonable and has minor effects on output.

B. Data Mining & Feature Extraction

1) Data Pre-Processing: We deploy the Spark Python API (PySpark) for fast data manipulation, where huge files are stored in the form of Resilient Distributed Dataset (RDD). As mentioned in section III.A, the trip records in TLC data set contain corrupted rows. Specifically, we find some trip distance, total amount, and location entries erroneous. Certain trip distances are a few thousand miles, and some payments are negative. The first step is to filter out the corrupted rows from input files. Next, we discard irrelevant entries, and reorganize the key-value pairs. In order to adapt patterns and avoid over-generalizing the data, we decide to train on data from specific week day, and generate the policy for the same week day. This takes account for different passenger behavior on different week days. The final data are stored as key-value pairs, where the key is a grid-hour tuple (pick-up location, time), and the value contains trip distance, trip duration, average speed, drop-off location, and total pay amount.

2) Parameters: To evaluate the best pick-up location, we take account of the following grid parameters: pick-up probability p, payment amount distribution μ_p, σ_p, trip distance distribution starting from the grid μ_s, σ_s, and trip time distribution starting from the grid μ_t, σ_t. We now explain each parameter in details.

1) Pick-up probability p. The passenger pick-up process within a grid g can be modeled by a standard Poisson process, where λ is the average waiting time for a pick-up in g. The average waiting time is calculated by the formula $\lambda_{(g,t)} = \frac{1}{D}\sum_d \frac{60}{\mathcal{P}_{(g,d,t)}}$, where D is the total number of week day d in the data set, and $\mathcal{P}_{(g,d,t)}$ is the total number of pick-up's in grid g during hour t of day d. Now that we have $\lambda_{(g,t)}$, we need to know the driver's waiting time in grid g. We assume the driver cruises in g until he meets any passenger. The cruising time t_c can be calculated by the size of g and cruising speed v_c. While the size l_g is easily calculated by using haversine formulas with the GPS coordinates, the cruising speed v_c is not included in data. Hence we introduce the concept of **congestion factor** α to infer v_c from the average trip speed v_t. The congestion factor α is calculated by the workload ω_g of the grid, which maps an exponential decay of ω_g to a more smooth scaling of taxi speed v_g. The workload ω_g is defined as $\frac{\mathcal{P}_{(g,t)}+\mathcal{D}_{(g,t)}}{\sum_{g,t}(\mathcal{P}_{(g,t)}+\mathcal{D}_{(g,t)})}$, the number of pick-up and drop-off activities at certain hour t in grid g divided by that of the entire map. Here we assume the taxi activity has a strong positive correlation with the traffic congestion, since more activities indicates a hotspot, thus more traffic.

C. Markov Decision Process

D. Policy Generation

1) Learned Policy:

2) Baseline Policy:

3) Oracle Policy:

V. Results

A. Policy Analysis

B. Visualization of Simulated Path

VI. Future Works