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[1], 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[2]. Daulton *et al.* uses machine learning methods to predict pick-up density and drop-off locations[3]. Tipsy Taxi reveals the fraud behavior, and performs traffic analysis, tip analysis for drivers[4]. There are studies on taxi strategies, with binary behavior recommendation such as "hunting" or "waiting". Li *et al.*[5] and Tang *et al.* uses data with taxi occupancy label[6], 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[7].

III. Framework

A. Data Set

We use data directly from NYC Taxi & Limousine Commission (TLC) official website[8]. 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[9]. 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, trip time distribution starting from the grid μ_t, σ_t, and a drop-off location probability map. 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

$$\frac{\mathcal{P}_{(g,t)} + \mathcal{D}_{(g,t)}}{\sum_g (\mathcal{P}_{(g,t)} + \mathcal{D}_{(g,t)})} \tag{1}$$

, 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. The congestion factor α is defined as a sigmoid:

$$\alpha_g = 2(1 - \frac{1}{1 + e^{-K(\omega_g - \mu_\omega)}}) \tag{2}$$

$$v_g = \alpha_g v_t \tag{3}$$

The congestion factor is scaled by 2, because we assume the average speed at any grid is within amplitude of 2 of the average trip speed. The sigmoid function is intuitive to use, due to the transition points on its curve: on an empty street, the speed is harder to increase; and inside a jam, the speed is unlikely to decrease. The steepness factor K is tuned to accommodate another important feature: across the set of grids $\mathcal{G}$ passed in a trip, the average of α calibrated speed v_g should still average to trip average v_t. This can be generalized into the continuous case such that

$$\int \alpha v_t \cdot p(\alpha, s) d\alpha = v_t \tag{4}$$

where $p(\alpha, s)$ is the portion of grids with congestion factor α in a trip with distance s. Equation (4) can be further simplified to

$$\int \alpha p(\alpha, s) d\alpha = 1 \tag{5}$$

To prove this case, first we need to find the distribution function $p(\alpha, s)$. We plot the histogram of Number of Occurrence *vs.* ω at Tuesday 12am on a small but representative data set in figure 1. This is representative in all (day, hour) combinations. Apparently $p(\alpha, s)$ follows an exponential distribution, where

$$p(\alpha, s) = \kappa \frac{L}{s} e^{-\kappa \frac{L}{s} \alpha} \tag{6}$$

Here $\frac{L}{s}$ is the scale factor for κ, where L is the constant number of map size. This

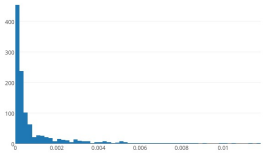

Fig. 1. Tuesday 12am Workload Distribution

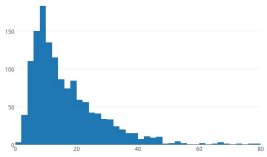

Fig. 2. 5pm Little Italy SOHO Trip Time Distribution

can be interpreted as the longer distance the driver traveled, the more he/she is likely to see the real distribution. With small value of s (short trip distance), by common sense the driver is more likely to see grids with smaller ω, equivalent to a steeper exponential probability curve after normalization. Now if we substitute $\kappa\frac{L}{s}$ as γ into (5), we get

$$\int_0^1 (1 - \frac{1}{1+e^{-K(\omega-\mu_\omega)}})\gamma e^{-\gamma\omega} d\omega = \frac{1}{2} \tag{7}$$

For this equation to hold, we tuned the steepness factor K empirically as $\frac{\mu_\omega}{2\sigma_\omega^2}$. This setup is later proved as work out well. With the cruise time calculated from $t_c = \frac{v_t}{\alpha v_g}$ and average waiting time λ, we sample from the process and calculate the pick-up probability:

$$p = \sum_{i=0}^{t_c} \frac{\lambda^i e^{-\lambda}}{i!} \tag{8}$$

2) Distribution mean, variance. We plot the histogram of trip distance, trip duration, and total pay amount starting from several POI's at different time on Tuesday in figure 2-4. Apparently, from the figures, we notice all three distributions are skewed normal. The

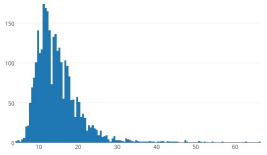

Fig. 3. 6pm Lincoln Tunnel Pay Distribution

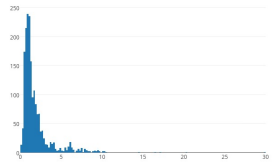

Fig. 4. 11am Sutton Place Trip Distance Distribution

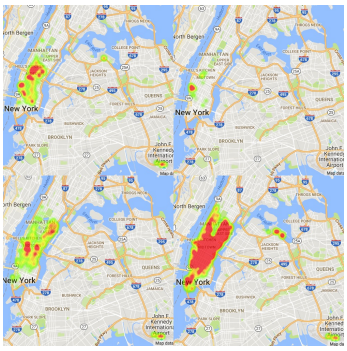

Fig. 5. From Top-Left to Bottom-Right: 12am, 3am, 6am, 9am

means and variances of theses distributions are thus effective in evaluating the reward of a grid. Given the numbers, for the specific week day, we define the following reward function for a (grid, hour) key:

$$R(g,t) = p\frac{(\mu_p - \frac{\sigma_p}{2})^2}{(\mu_t - \frac{\sigma_t}{2})(\mu_s - \frac{\sigma_s}{2})} \tag{9}$$

We will discuss later their usage in path simulation.

3) Drop-off location probability map. This is crucial data for the MDP process and path simulation. Given location-time pair (g,t), we store the mapping

$$\mathcal{M} : (g,t,g') \to \{\rho_{g'}\} \tag{10}$$

for all g' in the drop-off location set $\mathcal{D}_{(g,t)}$, where ρ is the probability of drop-off at g'. Note that $\sum_{g'} \rho_{g'} = 1$.

With above parameters described, we visualize a few pick-up probability maps of NYC, generated from a 15-month data set. We also plot the drop-off location map from certain pick-up locations. See figure 5-9.

C. Markov Decision Process

Now we formally define the MDP for the passenger pick-up problem.

States: $\mathcal{S} = \{(g,t) : g \in \mathcal{B} \wedge t \in (T_s, T_e)\}$, where T_s, T_e are start and end time (driver shift).

$\mathbf{s_{start}} = (g_s, T_s)$ where g_s is starting location.

Actions$((g,t))$ =

$$\{(g',t') : t' = t + t_s \wedge g' \in \mathcal{N}(g) \cup \mathcal{H}\} \tag{11}$$

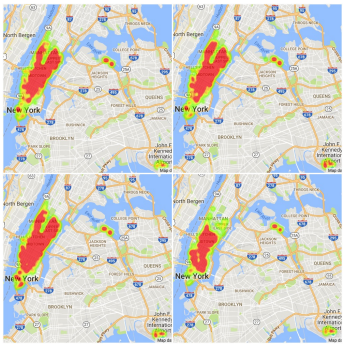

Fig. 6. From Top-Left to Bottom-Right: 12pm, 15pm, 18pm, 23pm

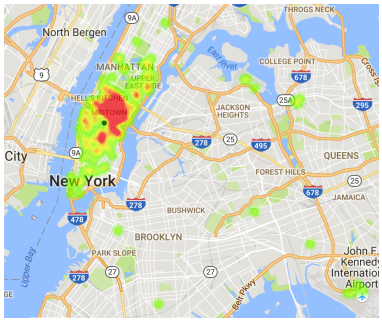

Fig. 7. 5pm Drop-off From Empire State Building

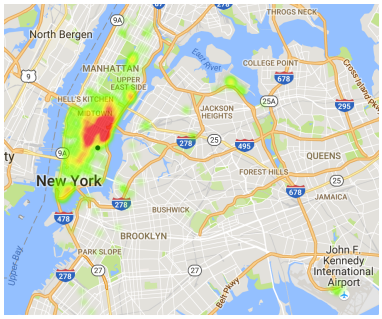

Fig. 8. 9pm Drop-off From Mt. Sinai Hospital

Fig. 9. 11am Drop-off From Chanel Shop

where t_s is the trip duration, $\mathcal{N}(\})$ is the neighbors of g (including itself), and $\mathcal{H}$ is the set of hotspots on the map.
$\mathbf{T}((g,t),a,(g',t'))$=

$$1\{g' \in \mathcal{N}(g)\}(1-p_g) \quad (12)$$
$$+1\{g' \notin \mathcal{N}(g)\}p_g \cdot \mathcal{M}_{(g,t,g')} \quad (13)$$

which can be viewed as two parts: if the taxi does not pick up a passenger in current location, then it can arrive in the neighbors; otherwise it can only arrive at location g' if the passenger also goes there.
Reward$((g,t),a,(g',t')) = R(g,t) \cdot \frac{t'-t}{|g'-g|}$
This is simply dividing reward function (9) by the travel time towards g'.
IsEnd$((g,t)) = 1\{t > T_e\}$.
γ = 1.

Now the logic behind the scene. There are many assumptions to make to solve the problem: if we assume the driver to pick up whenever he/she sees passengers on the way, then we can only allow Actions$((g,t))$ to be $\mathcal{N}$, otherwise there is no guarantee that the driver will ever arrive at any chosen g'. However, if we only allow Actions$((g,t))$ to be $\mathcal{N}$, then if the driver is in the middle of nowhere (unseen states in data set), then he/she will get bogged.

On the other hand, if we assume the driver only goes to chosen target g' without taking any passengers midway, then we can allow Actions$((g,t))$ to be arbitrary $g' \in \mathcal{B}$. This setting explores the entire map with every possible situation, which finds the global optimum of the problem. However this is computationally unacceptable. Instead we decide to take a step back, and choose to search for local optimum: let Action$((g,t))$ to be $\mathcal{N} \cup \mathcal{H}$, a combination of neighbors and hotspots. This prevents the driver from wandering in nowhere, and also stops him/her from over-exploiting the hotspots. This is an analogy of the Exploration-Exploitation problem, where ϵ stands for the size of $\mathcal{H}$. Our assumption of not picking up midway also works perfectly here: if the driver is heading to the hotspots, the g's on the path will not likely have high pick-up probabilities, otherwise they are hotspots themselves.

D. Policy Generation

1) Learned Policy: We use value iteration method to generate the learned policy for MDP states. Different from what we stated in the formulation of MDP above, in actual implementation we choose states as the set of states appeared in data set keys. This is to reduce unnecessary computation and expedite the iteration. If the successor (g',t') from (g,t) is unseen (new t' from time elapse, or new g' from wandering into neighbors), then we assume average t_c, v, and $p = 0.0$, since not

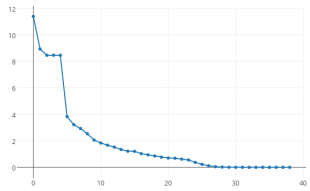

Fig. 10. Value Iteration Learning Curve

recorded means there is no previous pick-up event. The probability sums to 1 as follows:

$$p + (1-p)\sum_{g'} \rho_{g'} = 1 \tag{14}$$

Figure 10 is the learning curve for value iteration.

2) Baseline Policy: An obvious baseline policy is to run randomly on $\mathcal{N}$. One disadvantage mentioned before is that the driver may run into unreachable spaces such as the middle of Central Park, or Hudson Bay. Using coordinates of POI's can solve this problem.

3) Oracle Policy: The main obstacle for achieving an oracle policy is that the TLC data set is highly anonymous, with all license and Medallion ID erased from records, thus we cannot follow on individual taxis to directly pick out the route with maximum net profit. We come up with a dynamic programming get-around. Different from the baseline and learned policies, the oracle takes in the initial location g_i to generate an optimal route. We first process the original data RDD to reverse link all states, so that a key-value pair has form of $s' \to (s, m(s, s'))$, where $m(s, s')$ is the utility earned by going from $s = (g, t)$ to $s' = (g', t')$, evaluated by the Reward function. The base case is initializing all utilities to be zero: $u_s = 0$ for all $s \in \mathcal{S}$. Next we define the transition function

$$u_s = \max(u_{s'} + m(s, s'), m(s_i, s)) \tag{15}$$

where s_i is the initial state. Note that all unseen states are assigned with $u = 0$. Backtracking the utility map, we get the oracle policy.

V. Results

A. Policy Analysis

We run the framework on the same small representative data set, run generated policies on same location and time settings for 30 iterations, and use the average reward to evaluate the correctness of our methods. Below is the table comparing results from three policies.

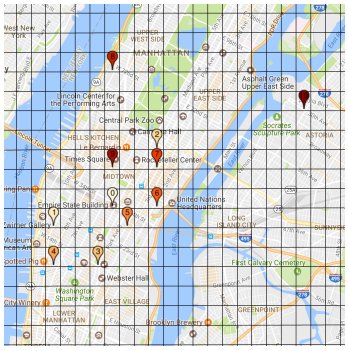

Fig. 11. Oracle Route. Earned 104.01$

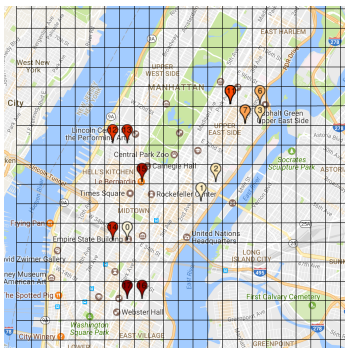

Fig. 12. Learned Route. Earned 83.80$

Policy	Profit	Depreciation	Performance
Baseline	51.93	14.94	3.48
Learned	74.99	19.55	3.84
Oracle	79.97	20.66	3.87

B. Visualization of Simulated Path

In order to better extrapolate the feasibility and correctness of our framework, we run the generated policies on our simulator and visualize some representative routes in figure 11-13, starting from Tuesday 12:01 AM from Empire State Building, driving for 80 minutes. The full visualization part is included in code repository, with drop pins marking earned money, and time. In simulation phase, we sample the multinomial drop-off locations from $\mathcal{M}$. Note that the profit values are higher than the actual case of average 50$, due to the slightly large fare rate in simulation.

VI. Future Works

As mentioned previously, the gridify process is a bold assumption we make, and migrate to POI GPS coordinates can make the framework more accurate. Also in evaluating cruise time from current location to target location, we simply divide the straight line distance by average speed, which is not ideal. Introducing Brehensam's Algorithm can generate a location list, from where we can make use of congestion factors along the path and get a more accurate estimation of cruise time between locations. As a side mark, we will

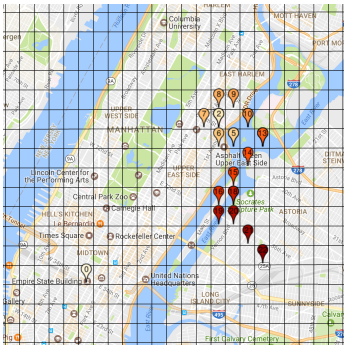

Fig. 13. Baseline Route. Earned 25.09$

run on larger data set for more comprehensive policies, with extra computational resources such as AWS. Code available on GitHub repository https://github.com/JulianYG/221finalproject/.

REFERENCES

[1] N. T. . L. Commission, "2014 taxicab fact book," 2014.
[2] C. Wong, "A day in the life, a data visualization of a single nyc taxi over 24 hours," 2015.
[3] S. R. Samuel Daulton and T. Kindt, "Nyc taxi data prediction," 2015.
[4] lucianoiscool, "Tipsy taxi," 2014.
[5] B. Li, D. Zhang, L. Sun, C. Chen, S. Li, G. Qi, and Q. Yang, "Hunting or waiting? discovering passenger-finding strategies from a large-scale real-world taxi dataset," *IEEE Xplore 10.1109/PERCOMW.2011.5766967*, 2011.
[6] H. Tang, M. Kerber, Q. Huang, and L. Guibas, "Locating lucrative passengers for taxicab drivers," *ACM SIGSPATIAL 10.1145/2525314.2525471*, 2013.
[7] J. Yuan, Y. Zheng, L. Zhang, X. Xie, and G. Sun, "Where to find my next passenger?," *UbiComp 10.1145/2030112.2030128*, 2011.
[8] N. T. . L. Commission
[9] N. T. . L. Commission, "Data dictionary - yellow taxi trip records," 2015.