Order Dispatch Improvement and Tendency Prediction via Visualization by Clustering

Yuzhuo Jing, Yimin Tang, Qifan Zhang, Zhijie Yang, Jia Du

Abstract: In this project, we used clustering to visualize the data given by DiDi to figure out an improved way of dispatching order to the drivers for less expected waiting time for the passengers and less distance traveled for the drivers to pick up the passengers.

Keywords: DiDi; clustering; supervised learning; dispatch; overall planning; visualization

I. Introduction

DiDi Chuxing (abbreviate DiDi) is the largest ride-hailing application in China. With its rising number of users, sharing a ride or hiring a taxi on DiDi has been generally accepted as a brand new mean of traveling. However, DiDi still has some unresolved problems such as efficiency-insufficient order dispatching mechanism and lacking of overall planning of scheduling the cabs. Since a pattern of traveling in a city is relatively fixed, it is possible to study on the distribution of the departures and destinations of orders to predict the directions of travel and schedule the cabs before the enormous orders had been posed.

II. Clustering and Basic Analysis

1. Clustering

Clustering here is the task of grouping the set of starting or ending points of the DiDi orders in such a way that points in the same group are more similar (in some sense) to each other than to those in other groups (clusters). It is the main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics.

We firstly use the algorithm named K-means. K-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells.

```
procedure KMEANS(DATA,K)
    SELECT K POINTS AS THE INITIAL STATE
    for ALL POINTS $\in$ THE DATA do
        FIND THE NEAREST POINT IN THE INITIAL STATE
    while POINTS IN THE INITIAL STATE ARE STEADY do
        for ALL POINTS $\in$ THE INITIAL STATE do
            POINT $\Leftarrow$ MEAN(ALL POINTS WHOSE NEAREST POINT IS THE POINT)
```

Then we found out that the initial points is very important for our final result. Different initial points make different final result in many cases within the data from DiDi. After we apply the K-means algorithm to the DiDi database, we also tried another algorithm named EM algorithm whose result is better than the K-means algorithm.

We finally use the model named Gaussian Mixed Model which is a special case from the EM algorithm. The idea of GMM is very simple.

```
1: procedure GMM(DATA,K)
2:     P(Y): Distribution over k components (clusters)
3:     P(X—Y): Each component generates data from a multivariate Gaussian
4:     Goal: maximize the likelihood
5:     $\Pi_j P(y_i = i, x_j) = \Pi_j N(\mu_i, \Sigma_i) P(y_j = i)$
6:     while POINTS IN THE INITIAL STATE ARE STEADY do
7:         Choose component i with probability P(y=i)
8:         Generate data point from N( $\mu_i$, Σi )
```

2. Basic Analysis

By comparing the heat map of origin and destination (Figure 1 and Figure 2), it is clear that most destinations lies inside the second ring elevated road, while there is a considerable part of the origins lies outside the second ring elevated road. A preliminary conclusion can be made is that DiDi can dispatch more orders of destinations outside of the second ring elevated road to distribute its capacity right in these areas that are short of cabs. If this pattern is

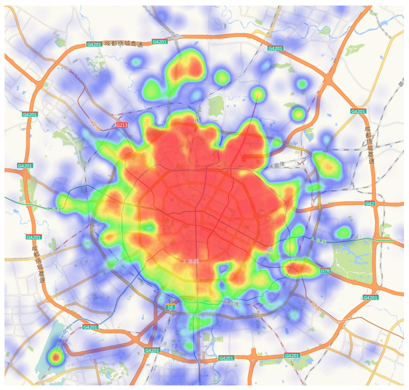
Figure 1: Heat Map of Origins

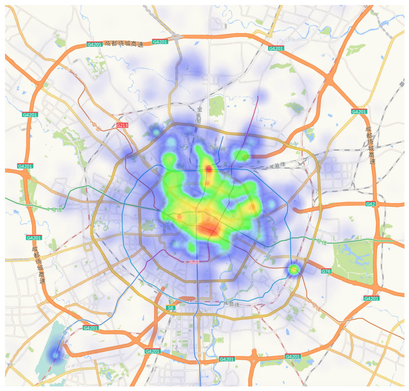
Figure 2: Heat Map of Destinations

III. Distribution Optimization

After we use *K-means* to get clusters of customers and cabs, we just see them as *hubs*. A *hub* is a site where customers and cabs around around could go and wait there for picking up (*or waiting for next customer*).

First optimization we will make is on the distribution system. Currently the distribution system is just fetching the nearest car around where the customer places the order. It is good for both the customer and the cab driver because it will make them both very efficient. However, in a larger scale, it is not so efficient. That the distance between the customer and the cab is whether 2 or 3 kilometers does not matter so much compared with the cab speed. Based on this, we may distribute a little farther but more vacant cabs to the customer ,and distribute nearer ones to where cabs are more urgently needed. It will cost both customers and drivers a little more time (*on average 1 or 2 minutes for 1 or 2 extra minutes*), but for the whole sector *supply-demand* balance will be neutralized more.

From this point of view, I form a new distribution system, which takes two factors into consideration:

1. Supply-Demand in the sector of the cab

2. Distance to the customer

we will finally calculate a *"score"* for each cab. Sorted by the *score*, we will pick up the most suitable one for the customer.

Based on this theory, we work out two methods to work out appropriate formula to calculate a score.

1. Manually Set Formula

Based on observation of the whole dataset, we calculate the *Supply-Demand* ratio and the distance between each cab and the customer. We think the *Supply-Demand* ratio has higher prior than distance, since this will make only a couple of minutes longer when the difference is 1 or 2 kilometers.

After comparison, we think this formula works best:

$$Score = 10^{Supply-Demand\ ratio^2-1} + 100^{threshold\ distance-distance}$$

In the formula, *threshold distance* is the maximum appropriate distance between the cab to be dispatched and the customer. If distance between the customer and the cab goes beyond this threshold value, it is not very suitable, but it is acceptable if there is no nearer cabs around.

We also set two special occasions for distance factor. If the distance is smaller than *minimum distance*, which could be set manually(*by default it is 0.5 kilometer*), the cab will directly get dispatched since it is so near that it costs almost nothing for this cad to pick up the customer. In contrast, if the distance is larger than *maximum distance*, which could also be set manually(*by default it is 4 kilometers*), the cab will be excluded from the cab waiting list directly, since the cost for pickup is far from satisfaction.

To test this formula, we assume that we call a cab on 1191 random sites in city center, the distance distribution is:

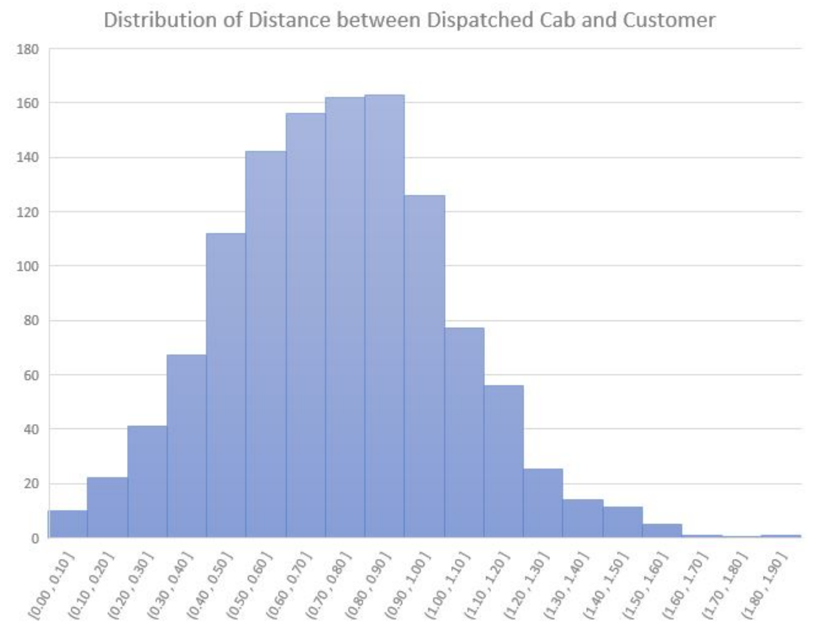

Figure 3: Distribution of Distance between Dispatched Cab and Customer Using Manually Set Formula

The distribution is very normalized. On average, the distance between the dispatched cab and the customer is 0.786 kilometer, which is acceptable.

2. Formula from Supervised Learning

Formula gotten from the first method works, but after all it is manually decided, which is not very reasonable. To make it more reliable and scientific, we use supervised learning to get a formula more reliable and scientific way.

First, we randomly take a coordinate as the customer location ($104.0545°N$, $30.7116°E$). Then we filter out suitable cabs, which are from *minimum distance* to *maximum distance* around. In total we have in total 2279 hubs meeting our requirement. Sorted by *Supply-Demand* ratio, we give each of the hub a score manually, which show our preference for each hub.

After giving each cab hub a score, we start to get a general formula using supervised learning.

For supervised learning, we follow:

$$\begin{aligned} h_\theta(x) &= \theta_0 + \theta_1 x_1 + ... + \theta_n x_n \\ &= x^T \theta \end{aligned}$$

where each x_i is a feature deciding the value of function $h(x)$. $h_\theta(x)$ means the final function we get is only from features of Θ, which may have some bias with real $h(x)$.

In this project, we have two features deciding the final score: *Supply-Demand* ratio and distance from the customer. Therefore, let function $h(x)$ be final score:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

where x_1 represents the *Supply-Demand* ratio factor and x_2 represents distance factor. $\theta_0, \theta_1, \theta_2$ are all constants making the final score reliable.

Just like the first method, *Supply-Demand* ratio factor is more important than distance factor, so we choose more *'dominant'* function for x_1 than x_2. For simplicity, define $x_1 = r^2$ and $x_2 = \ln(d)$, where r represents *Supply-Demand* ratio and d represents distance between the cab and the customer.

Having gotten scores, distances and demand for each cab, we just do a little Linear Algebraic calculation. This method is also known as *linear regression*:

$$\begin{aligned} X^T \theta &= H_\theta \\ X^{TT} X^T \hat{\theta} &= X^{TT} H_\theta \\ X X^T \hat{\theta} &= X H_\theta \\ \hat{\theta} &= (X X^T)^{-1} X H_\theta \end{aligned}$$

where the last raw of matrix X will be 1 since θ_0 is *constant parameter*.

Directly calculate inverse of XX^T using determinants is rather slow. Suppose we have $n \times n$ matrix, using $Crammer's\ Rule$, time complexity is $\Theta(n^3)$ since we need to calculate $det(B_i)$ each time, which needs to solve two $n \times n$ matrices' multiplication. We use SVD to optimize the procedure. Though time complexity is still $O(n^3)$, runtime has an improvement. This is because *SVD* saves some steps during calculation. We plan to use *Coppersmith-Winograd algorithm* to boost it more, which has $O(n^{2.38})$ time complexity. However, till the hand-in deadline we have not implemented it correctly.

Getting $\hat{\theta}$, just like the first method, we randomly choose 1139 point to call a "cab". Distribution of distance between cabs and customers is below:

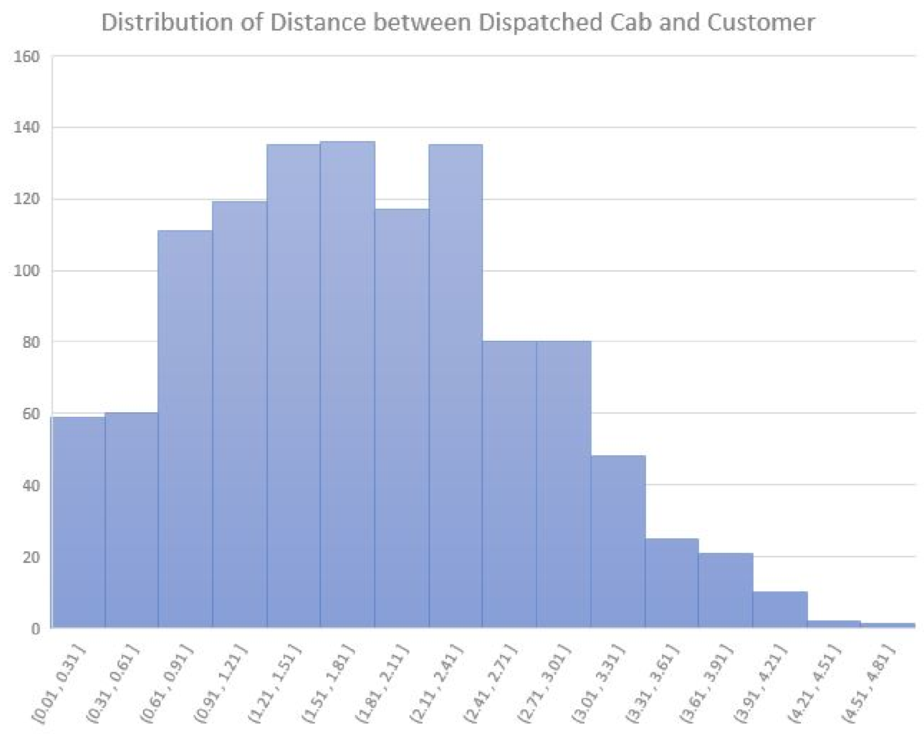

Figure 4: Distribution of Distance between Dispatched Cab and Customer Using Supervised Learning

Compared with Manual Method, average distance becomes larger and variance is larger too. But still it is acceptable. It is highly possible that functions we choose for x_1 and x_2 are too arbitrary, we need more trials to make it more reliable. Supervised Learning itself also causes inaccuracy. If permitted, we will use $NN - algorithms$ to make it more flexible and accurate.

IV. Real-Time Bonus

Nevertheless, distribution optimization just makes *Supply-Demand* ratio as balanced as possible at the last time. If there is a relatively large sector where cabs are in urgent short, however distribution is optimized, customers will not all be satisfied. To solve this, we need to motivate cab drivers to drive where cabs are urgently needed. Bonus is the best to motivate them.

In fact, *Didi* itself has pushed forward bonuses for orders from specified areas, such as *Zhangjiang* in *Pudong*. However, it updates every three days or even every week. It is complained by many drivers because of its lagging. We aim to design a real-time bonus model for *Didi* so that bonus will be much more efficient than the current one.

According to *Uber Mini Project*, they form a model of bonus:

$$Bonus = Bonus\ Base\ \times\ Variance^{1-Supply-Demand\ Ratio}$$

It provides a basic bonus and then relate it with urgency of supply and demand. To be more detailed, we form the formula to:

$$Bonus = k_b \times {k_v}^{\frac{c}{t}}$$

where k_b is Bonus base constant, k_v is variance constant, t is the number of available cabs and c is the number of customers.

First, we define*urgent hubs* as those who has fewer cabs than customers in the sector we set. And *lacking hubs* as those who has less than a certain number of cabs left for possible customers to take. The certain number is defined as $ThesholdCabLeftNumber$. Only this two kinds will need bonus to motivate vacant cabs to meet customers' needs. Due to difference on urgency, two kinds of bonus hubs will have different

Cab drivers we give bonus to is also a question. We take the following method: we set maximum distance for drivers to get bonus. Only those within maximum distance will be targets of bonus. Then, we will calculate the *supply-demand* ratio for each cab hub, only those who are neither *urgent hubs* nor *lacking hubs* will be valid for bonus.

Finally, for those who are valid for bonus, we will also give compensation based on distance from the initial cab hub to *urgent hubs* or *lacking hubs*.

Taking all factors into consideration, we work the final formula:

$$Bonus = u \times k_b \times {k_v}^{\frac{c}{t}} + cpk \times d$$

where u means urgency of the bonus hub. The more urgency it is, the larger u is. cpk means *Compensation per kilometer*. In fact, it has already existed to compensate cab drivers from current location to pickup location. For example, in *Shanghai* it is 0.8 *yuan* per kilometer *(told by a Didi driver)*.

V. Hub Deployment Improvement

From Fig.1, it is very obvious that pickup locations have a very regular distribution. In future, we could only allow customers to get a cab in regulated hubs.

Take *Jinke Road Subway Station* around as an example:

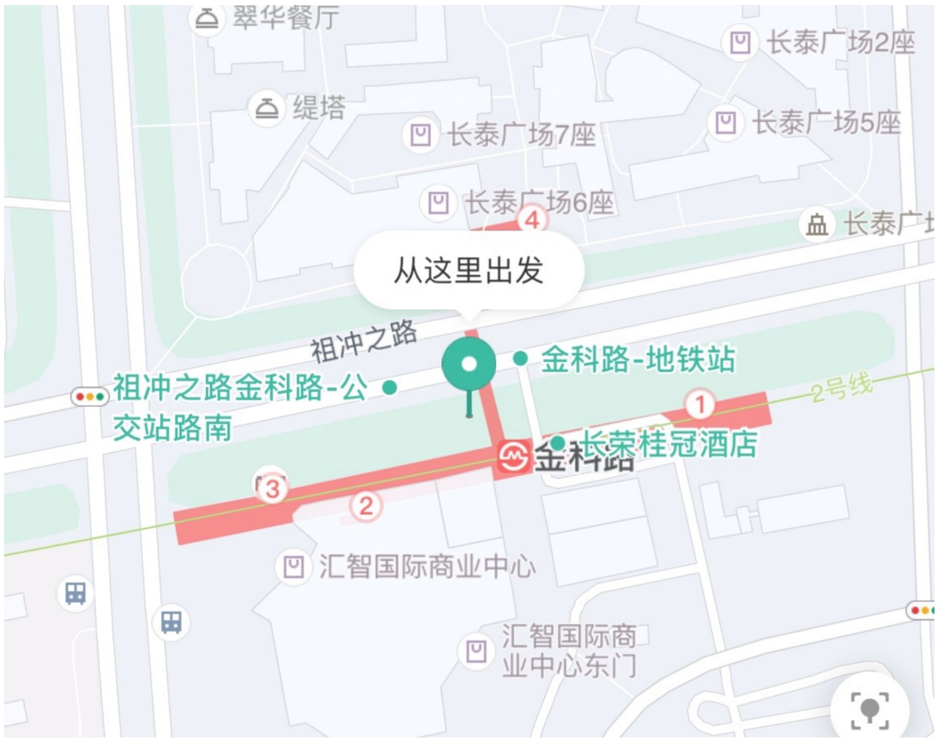

Figure 5: Jinke Road Subway Station *(from Didi Chuxing App)*

From figure 5, there are 3 pick up hubs for customers, but from our using experience, the pickup location in the middle and on the south side is very inconvenient for both cab drivers and customers. A traffic accident occurred once near *Jinke Road-Subway Station* pickup hub just because of picking up a customer on that hub. In the future, *Didi* should consider making surveys around the busy sectors and allow only safe hubs to pick up and drop off customers. Not only will traffic be safe, it will be convenient for both customers and cab drivers.