IEOR 265 Project: Statistical Learning of the Impact of Self-fulfillment on an E-commerce Platform

Heyuan Liu, Pelagie Elimbi Moudio, Mengxin Wang

April 26, 2018

1 Introduction

1.1 Problem Background

Alibaba, the biggest e-commerce platform in China, has long been using third-party logistics (3PL) companies as their product fulfillment solution. This leads to a major difference comparing with the e-commerce platform Amazon, which is well known for its FBA (Fulfillment by Amazon) service.

As an e-commerce company, fast and cheap fulfillment service is a key to success. 3PL and self-fulfillment are two essentially different fulfillment solutions. 3PL provides the company with a flexible fulfillment option, eliminating the cost of setting up extensive warehousing and distribution infrastructure and often offers bulk discounted prices. However, when using 3PL, the platform loses information and control over a significant portion of the supply chain (from after payment till order fulfillment). This largely restricts the company's ability to improve its delivery service. Meanwhile, though incurring large setup cost of warehouses and delivery facilities, the self-fulfillment mode provides the platform with all the supply chain information from customer payment to receiving of the goods. self-fulfillment gives the company full control of the supply chain. The company thus can further improve its fulfillment service, brand reputation and in turn generate a great profit.

Alibaba also realizes this problem and has started to adopt the self-fulfillment mode and build its own logistics platform – Cainiao Network. Cainiao uses insights from data to improve efficiency along the logistics value chain. The Alibaba platform brings together merchants

selling a variety of products. Previously, merchants were responsible for managing their inventory and fulfilled orders by themselves or by various third-party logistics service providers before the Cainiao Network came along in 2013. Currently, these vendors have the added option to use Cainiaos Network of logistics services (warehouse and delivery services).

1.2 Problem Statement

As self-fulfillment incurs tremendous cost in infrastructure construction, it is important for Alibaba to know whether Cainiao Network would have a positive effect on its fulfillment service. This knowledge could inform further investment in logistics infrastructure. In addition, it is also worthwhile for merchants to know whether joining the Cainiao Network fulfillment option for a certain of its product types holds a certain value such as improved reputation towards their customers. Therefore, our project assesses whether using Cianiaos Network logistics services is correlated with higher customer review scores and if with the available data, we can qualify the merchants decision to use Cainiaos services over other default services. We will be focusing particularly on customer scores rating their experience with the ordered item's logistic service.

We carry out a preliminary statistical exploration to verify if there is a significant statistical difference between the average review scores from products that are handled through Cianiaos Network logistics services versus those using externally organized logistics services. We established that there is a significant difference between the 2 distributions (merchants using Cainiaos Network of logistic services vs not) using a t-test. We obtained a p-value of 1.43e-12 which indicated statistical significance. Based on this result we proceed to the next part of our project which involves investigating and isolating different confounding variables that could affect the logistics rating scores. We decide based on elimination and intuition that the most confounding factors available in the data provided are the item category, subcategory, brand and promised delivery speed. We then solve a matrix completion problem using the alternating least squares approach, This method allows us to predict the logistic rate for a specific item when given subcategory, promise_speed and if_Cainiao. Knowledge of their predicted logistic scores can serve as a decision support tool for the merchants. The tuning parameter in out matrix completion problem was the rank of the matrix. We used three values $1, 2, 3$ and found better performance when the rank was set to 3.

The organization of the following part is: Section 1.3 gives the data description; Statistical analysis of the logistics scores are summarized in Section 2; Section 3 describes the prediction model and analyzes the results. Section 4 is the conclusion and possible future works.

1.3 Data description

The data set [1] contains information on customer orders, inventory history and detailed order fulfillment logistics of merchants from 2017/01/01 to 2017/07/31. We identified over 200 merchants using Cainiaos integrated warehousing and fulfillment service, and more than 200 merchants who managed their own inventory and fulfillment.

For the statistical analysis of the overall impact of the Cainiao Network, we used the merchant data as obtained from the original data set with structure shown in Table 2. The data provided by table 2 contains detailed information about the daily operational performance of each merchant. For each merchant, the subcategory_id and whether it uses Cainiao Network is recorded in the data-set. Also, it provides us with daily page view, user view and average rating of product quality, service and logistics of each merchant.

To train the prediction model, we used the more detailed data set created by extracting the desired information for each order and each item, from Table 3 and 4. These tables include a detailed inventory information and meta-data for each item, including merchant, brand, and category information. The order data set records payment time, items included, participating buyer, promised delivery speed, merchant, logistics score and whether using Cainiao Network for each order. And the item data set gives the brand, category, sub category and selling merchant of each item. Also each item has a feature of using Cainiao Network or not. The inventory information was only available for merchants using Cainiaos warehousing and fulfillment service. We further elaborate on our data preparation and processing in the subsequent sections.

2 Statistical analysis: Impact of Self-fulfillment towards the platform

In this section we analyze whether the Cainiao Network have a positive impact to the merchants in the platform. The basic idea is to do statistical analysis of whether there is a significant difference of the logistics score between Cainiao and non-Cainiao users. Though Cainiao Netork may also affect the service or product quantity score

2.1 Data Pre-processing

In the first step we searched through the data tables and eliminated duplicate information and rows with incomplete or unusual data types. The we proceeded to extract useful information from the data set shown in Table 2. In the original merchant data set, there are in total 247,470 records of Cainiao users and 235,651 records of non-Cainiao users. The records are daily-based. In other words, the average logistics score is not cumulative but for that single day. Therefore, we aggregate the logistics score per day for each merchant by

$$\frac{\sum(pc\_uv + app\_uv) * avg\_logistics\_review\_score}{\sum(pc\_uv + app\_uv)}$$

in which pc_uv+app_uv is an estimate of the number of order for a certain merchant each day. In this way we got an estimate of the total average logistics score for each merchant. After aggregation, we obtained 259 records of Cainiao Logistics services users and 266 records of non-CN users. The daily aggregation of the logistics scores was is done so as to uniformize the aggregation time periods. This becomes essential at later stages because we collated data from multiple raw data tables each with different levels of logistic score aggregation.

2.2 Statistical Analysis

Table 1: Mean and Standard Deviation of Logistics Scores

	Cainiao	non-Cainiao
mean	4.815507	4.780681
std	0.040773	0.066051

With the aggregated data, we performed basic statistical analysis on the logistics score of Cainiao and non-Cainiao users. Table 1 shows the mean and standard deviation of lo-

gistics score for Cainiao and non-Cainiao users. Figure 1 and 2 shows the histograms with respect to the two groups of users. The two distributions have relatively close mean values as shown in table 1. However , the distribution of logistics scores of Cainiao users differs from non-Cainiao users in the way that it's more concentrated around the mean and takes on higher values.
To derive more rigorous results, we performed a t-test with unequal sample sizes and unknown variance to these two groups. The p-value of the t-test is 1.43e-12, which indicates significant difference.

We hypothesized from the data and personal experience that customer ratings tend to lie in the range of 4 to 5. As such even a small difference in the data of the order of 0.1 could be represent a difference we cannot ignore. The results from the t-test based on the aggregated metric of logistics scores corroborated our initial postulation and implied a positive impact of doing self-fulfillment towards merchants on the e-commerce platform. This finding encouraged us to further explore the potential value of the the Cainiao Network.

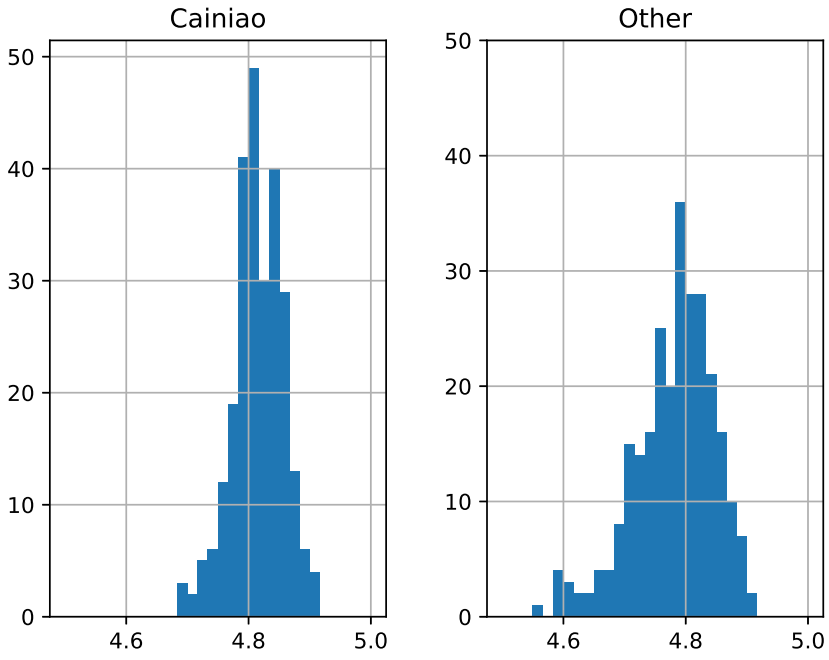

Figure 1: Histogram of the aggregated logistics score for Cainiao and non-Cainiao users

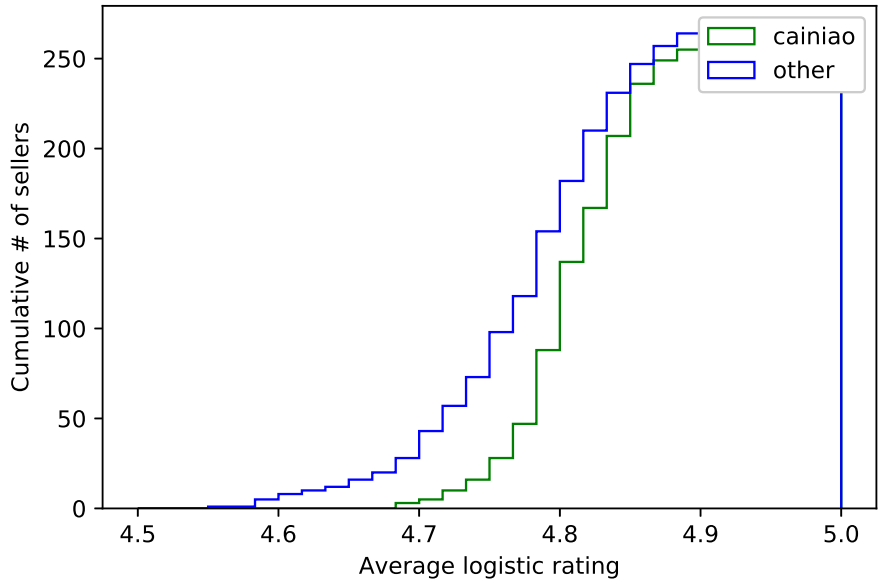

Figure 2: Cumulative histogram of the aggregated logistics score for Cainiao and non-Cainiao users

3 Prediction Model: Decision Support Tool for Merchants

3.1 Data Selection and Processing

From the order data (Table 3), we separated each order into distinct items with aggregated logistics scores and 'if_Cainiao' feature. For each item, we got its brand, category, subcategory and promised delivery speed from the item data (Table 4). Extracting the data from the three different tables and collating them as desired was one of the most time consuming parts of the project. This is because we had to match item id's across the various data set to find the corresponding information. In addition, the logistics ratings were aggregated at different levels in the different data tables. Some tables reported the value per item or group or items purchased while others aggregated them daily. We decided to aggregated all the score daily as shown in the section 2.2. In addition, in the case where the score was specified for a group of items, we uncoupled the items and gave all items in the same group the same score. Then we merged items with the same brand, category, subcategory and promised delivery speed together. The records in final data set has 5 features: brand, category, subcategory, promised delivery speed and if_Cainiao; and a response variable average logistics rating.

The variable selected for a model were chosen by elimination factors that we felt were less related to customer ratings. This elimination was mostly decided based on our experience using the products on the Alibaba website. Additionally, there were also some variables which we thought were relevant for analysis, however we were unable to reconcile or extract this information from the presented data set. This was usually due to the fact that incompatible aggregation levels for customer ratings which we could not correct with the available information or missing connecting item id's. This project highlight's the importance for companies to plan the structure of their collected data around potential decisions or questions that they might be interested in answering in the future. More so having a global identification indices for products can simplify the process of reconciliation of information across different data tables.

3.2 Model Description

Following the data processing described in section 3.1 we design a method to help individual merchant to decide whether to use the Cainiao network or not. There were several elements that informed our model selection. One predominant factor is that while we want to empower the merchants to make decisions about whether or not to use the Cainiao Logistics service, the optimal decision for each merchant may not necessarily be optimal for a system. Thus we decided against using a model like random forest that would allow us to predict the merchant's choice. This is because the random forest model would make these predictions for each merchant based on the choices of the other merchants which in many cases may neither be optimal for the merchant nor the system at large.

We consider the possibility that each merchant may have diering business strategies for their various products. As such, we implement a matrix completion algorithm to help us predict the logistic rate for a specific item when given subcategory, promise_speed and if_Cainiao. The model is designed to allow more granular predictions such that we predict the merchant's logistic scores for each of its product subcategories. We rationalize that there is a possibility that some product types may benefit from being included in the Cainiaos services while the reverse may be true for other classes of products. The predicted scores obtained from this method will serve as a support tool when developing their supply chain and logistics strategies whether or not they are already using Cainiaos logistics services.

We use the alternating least square approach for the matrix completion algorithm algorithm(outlined in Appendix B).This approach enable us to predict the logistics rate based on the brand, (sub) category, promised speed, and whether the Cainiaos logistic services are

used or not. Using the table derived from the raw data after processing, we construct a matrix $A_{i,j}$ which can be decomposed as follows

$$A_{i,j} = \sum_{j=1}^{k} u^j v_{j'}$$

We thus can develop the following matrix completion problem.

$$\hat{\psi} = \arg\min_{\psi} \left\{ \sum_{j=1}^{m} \left(y(i) - \psi_{i,j}\right)^2 | \text{rank}(\psi) \leq k \right\} = uv,$$

where

$$(u, v) = \arg\min_{u,v} \sum_{j=1}^{k} \left(y(i) - \sum_{j=1}^{m} u^j_{x(i)_1} v_{x(j)^{j'}_2} \right)^2 .$$

where i belongs to all sub categories of item, j index is the combination of promise_speed and if_Cainiao. From the data, we recovered four different deliver_speed options and if_Cainiao is a Boolean, the we have $4 \times 2 = 8$ choices . This implies that we are going to complete a 404×8 matrix.

3.2.1 Data training and Experiments

We train the data using 998 measurements, while in test data the number of measurements is 1075. We implement the alternating least square method on the training data, and at the same time, we examine the performance of the iterates on the test data. The tuning parameter in this problem is the rank of the matrix. At each iteration, we evaluate the full gradient of the objective function. The numerical results are as followed.

3.2.2 Results and Analysis

Figure 3 summarizes the numerical results obtained from our experiments. The plots outline the performance of the matrix completion. In both charts, the objective value which in this case is the minimized average squared loss plotted against the number of iterations for the training data(left) and the test data(right). This error represents how far our logistic ratings predictions are from their real data values. There is a reduction is a reduction in value of the

average squared error as the number of iterations increases and as expected we are unable to recover the level of performance obtained using the trained data.

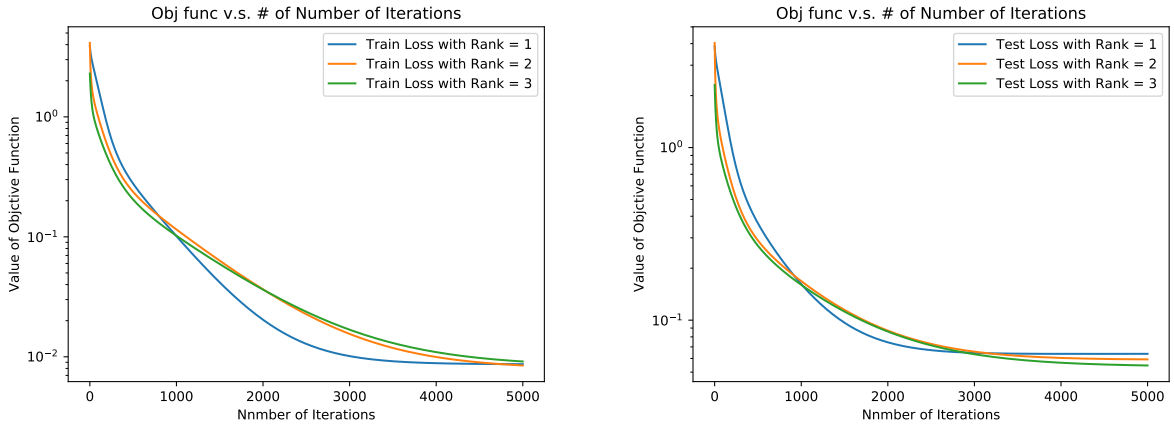

Figure 3: Mean square loss v.s. number of iterations on training data set(left) and test data(right)

We implement 3 different numerical experiments with different ranks $1, 2, 3$ in an attempt to tune the rank parameter. Each colored graph on Figure 3 represents the experiment with a specific rank. the best performance was observed when the rank was set to 3. This difference may not seem apparent from the plots because the objective value is plotted on the log scale. Table **??** shows the objective value after 5000 iterations.

	Train	Test
Rank $= 1$	0.0087	0.1036
Rank $= 2$	0.0090	0.0528
Rank $= 3$	0.0093	0.0344

Figure 4: Error in different rank constrains

4 Conclusion

4.1 Summary

The focus of this project was to provide a decision support model based on customer logistics ratings to assist merchants that utilize Alibaba's online platform in deciding whether or not to use Cainiao Network's logistic service. We used inventory and logistics data provided by Alibaba during the period spanning 2017/01/01 to 2017/07/31. Preliminary statistical

analysis indicated a difference between average logistics scores of Cainiao versus non-Cainiao users. we obtained a p-value on the t-test of 1.43e-12 and this suggested a significant enough statistical difference between scores from Cainiao versus non Cianiao users. We opted for a matrix completion model using the alternating least squares approach that would allow us to predict logistics score for each merchants products by subcategory.

We briefly explored implementing methods like random forest to predict the merchant's decision on whether or not to use Cainiao's services. However such a prediction model relies on the decision of other merchants but it is not always the case that optimality for one merchant may mean optimality for another, or the whole system. After training the data using 998 measurements, we use a data set of 1075 measurements to test the data. the tuning parameter in the matrix completion model is the rank of the matrix A_{ij}. We obtained the best objective value after 5000 iterations with the rank equal to 3.

4.2 Future Work and Recommendations

Throughout our analysis, we assumed that we were trying to complete a low rank matrix with no errors on observed data. Some interesting future investigations could involve solving a low rank matrix problem with noise. In addition it might be valuable to carry out an exploration that involves solving a high rank matrix completion problem. One of the drawbacks of the low rank matrix completion problem is that it does not fully capture the all the information from the data set. Further exploration into potential categorical methods could provide more insight about the data and allow more extensive decision making support for both the merchants and Alibaba.

Probably the biggest challenge on this project was collating the data from different tables to create the data set that we wanted. While there was a lot of redundant information across tables, it was sometimes impossible to recover the connection between same items across data sets. The data records were kept inefficiently as each item had multiple id's on the different data tables and even sometimes on the same table. This probably has a lot to do with how the internal logistics of the company works. In the future, it will be interesting if Alibaba had a more targeted data collection plan where the collected data is structured to answer specific questions. In addition, the insertion on global item identification across data sets can facilitate the combination of information across multiple data sets.

Appendices

A

Data Tables Structure

Data obtained from Alibaba ranges from 2017/01/01 to 2017/07/31. It includes more than 200 merchants using Cainiaos integrated warehousing and fulfillment service, and more than 200 merchants who managed their own inventory and fulfillment.

A.1

Merchant Data

Table 2: Merchant Data

Field	Data type	description	sample
day	string	Date of the snapshot	2E+07
merchant_id	string	ID of the merchant	5
subcategory_id	int	ID of the sub category	1
pc_pv	int	Total page view on pc	100
pc_uv	int	Total unique visitor count on pc	20
app_pv	int	Total page view on app	150
app_uv	int	Total unique visitor count on app	30
avg_logistic_review_score	double	Average review score on logistic service	4.5 (range in 1-5)
avg_order_quality_score	double	Average review score on order quality. Here, order quality means the quality of products in the order	4.4 (range in 1-5)
avg_service_quality_score	double	Average review score on service quality. Here service quality means the quality of on-line purchase service	4.6 (range in 1-5)
if_cainiao	boolean	Whether the merchant uses Cainiaos warehousing and fulfillment service	1: Use Cainiaos warehouse, 0: not use Cainiaos warehouse

A.2 Order Data

Table contains detailed information about customer orders.

Table 3: Order Data

Field	Data Type	Description	Sample
day	string	Date of payment	2E+07
order_id	string	ID of the order	123456
item_det_info	string	Includes item id, quantity and payment amount for all items in the order	123:2:4.5;234:2:35.5.
pay_timestamp	string	Time of payment (rounded to 1 mins intervals)	
buyer_id	string	ID of the user who placed the order	111
promise_speed	int	Promised delivery speed	1: same day delivery, 2: next day delivery, 0 or null :no promise
if_cainiao	int	Whether the order is delivered from cainiaos warehouse, i.e., did this order use Cainiaos warehousing and delivery service	0: not delivered from cainiao warehouse, 1: delivered from cainiao warehouse
merchant_id	string	ID of the merchant	222
Logistics_review_score	int	User's review score on the logistic service	Scale from 1 to 5, and higher score represents better logistic service

A.3 Item Data

Table 4: Item Data

Field	Data type	Description	Sample
date	string	Date of item view	2E+07
item_id	string	ID of the item	123
front_page_item_id	string	ID of the item at the e-commerce website	456
merchant_id	int	ID of the merchant	12
brand_id	int	ID of the brand of the item	11
category_id	int	ID of the main category of the item	1
sub_category_id	int	ID of the sub category of the item	2
pc_pv	int	Total page view on pc	100
app_pv	int	Total unique visitor count on app	70
pc_uv	int	Total unique visitor count on pc	50
app_uv	int	Total page view on app	150
if_cainiao	boolean	Whether the item is delivered by	1: Use Cainiaos warehouse, 0: not use Cainiaos warehouse

B

Matrix Completion Source Code

```
import numpy as np
import csv
import pandas as pd
import time
import random
import pylab as pl
from numpy import linalg as la
from collections import namedtuple
import matplotlib.pyplot as plt
import timeit
```

```
# Calculate the value of objective function
def obj(sub_max, deli_max, rank_max, score_train, num_train, score_test,
    num_test, u_now, v_now):

   loss = 0
   loss_test = 0

   score = np.dot(u_now, v_now)
   score_loss = np.asmatrix(np.zeros((sub_max+1, deli_max)))

   for i in range(sub_max):
      for j in range(deli_max):
         if num_train[i, j] > 0:
            temp = score_train[i, j] / num_train[i, j] - score[i, j]
            score_loss[i, j] = temp
            loss = loss + temp * temp

         if num_test[i, j] > 0:
            temp = score_test[i, j] / num_test[i, j] - score[i, j]
            loss_test = loss_test + temp * temp

   return loss, loss_test, score_loss


# Alternating least square method for matrix completion
def alter_ls(sub_max, deli_max, rank_max, t_max, alpha, score_train, num_train,
    score_test, num_test, u_initial, v_initial):

   u_now = u_initial
   v_now = v_initial

   loss_train = [0 for t in range(t_max)]
   loss_test = [0 for t in range(t_max)]

   for t in range(t_max):

      obj_func, test_func, obj_matrix = obj(sub_max, deli_max, rank_max,
         score_train, num_train, score_test, num_test, u_now, v_now)
      loss_train[t] = obj_func
```

```
        loss_test[t] = test_func

        u_t = u_now + alpha * np.dot(obj_matrix, np.transpose(v_now))
        v_t = v_now + alpha * np.dot(np.transpose(u_now), obj_matrix)

        u_now = u_t
        v_now = v_t

    return u_now, v_now, loss_train, loss_test



dict = {}
k = 0
sub_max = 0
t0 = timeit.default_timer()


deli_max = 8
item_train = 0

score_train = np.zeros((405, deli_max))
num_train = np.zeros((405, deli_max))

# Read logistic rate for trainning
with open('order_to_item_score_1.csv', newline = '') as csvfile:
    f_csv = csv.reader(csvfile, delimiter=',')

    for row in f_csv:
        item_info = row[0]
        acm_score = int(row[1])
        acm_num = int(row[2])

        info_list = item_info.split(':')


        if info_list[2] != '':
            item_brand = int(info_list[0])
            item_cate = int(info_list[1])
            item_sub = int(info_list[2])
```

```
            item_speed = int(info_list[3])
            item_deli = int(info_list[4])

            item_train = item_train + 1

            if item_sub > sub_max:
               sub_max = item_sub

            deli_info = item_speed + 4 * item_deli

            score_train[item_sub, deli_info] = score_train[item_sub, deli_info] +
                acm_score
            num_train[item_sub, deli_info] = num_train[item_sub, deli_info] + acm_num

            if k == 0:
               print(item_brand, item_cate, item_sub, item_speed, item_deli)

            k = k + 1

      print('Sub_Max = ', sub_max)

      sum_deli = 0

      for i in range(sub_max):
         for j in range(deli_max):
            if num_train[i, j] > 0:
               sum_deli = sum_deli + 1

      print('Sum_Deil = ', sum_deli)



item_test = 0

score_test = np.zeros((405, deli_max))
num_test = np.zeros((405, deli_max))

# Read logistic rate for testing
with open('order_to_item_score_7.csv', newline = '') as csvfile:
```

```
f_csv = csv.reader(csvfile, delimiter=',')

for row in f_csv:
   item_info = row[0]
   acm_score = int(row[1])
   acm_num = int(row[2])

   info_list = item_info.split(':')


   if info_list[2] != '':
      item_brand = int(info_list[0])
      item_cate = int(info_list[1])
      item_sub = int(info_list[2])
      item_speed = int(info_list[3])
      item_deli = int(info_list[4])

      item_test = item_test + 1

      if item_sub > sub_max:
         sub_max = item_sub

      deli_info = item_speed + 4 * item_deli

      score_test[item_sub, deli_info] = score_test[item_sub, deli_info] +
          acm_score
      num_test[item_sub, deli_info] = num_test[item_sub, deli_info] + acm_num

      if k == 0:
         print(item_brand, item_cate, item_sub, item_speed, item_deli)

      k = k + 1

print('Sub_Max = ', sub_max)

sum_deli = 0

for i in range(sub_max):
   for j in range(deli_max):
      if num_test[i, j] > 0:
```

```
            sum_deli = sum_deli + 1

    print('Sum_Deil = ', sum_deli)


t1 = timeit.default_timer()
print(k, t1 - t0)

t_max = 5000
alpha = 1e-4
sigma = 1

# Initializing for matrix completion
rank_max_1 = 1
s_initial_1 = 1
u_initial_1 = np.asmatrix([[s_initial_1 + random.gauss(0, sigma) for j in
    range(rank_max_1)] for i in range(sub_max + 1)])
v_initial_1 = np.asmatrix([[s_initial_1 + random.gauss(0, sigma) for j in
    range(deli_max)] for i in range(rank_max_1)])

rank_max_2 = 2
s_initial_2 = 1
u_initial_2 = np.asmatrix([[s_initial_2 + random.gauss(0, sigma) for j in
    range(rank_max_2)] for i in range(sub_max + 1)])
v_initial_2 = np.asmatrix([[s_initial_2 + random.gauss(0, sigma) for j in
    range(deli_max)] for i in range(rank_max_2)])

rank_max_3 = 3
s_initial_3 = 1
u_initial_3 = np.asmatrix([[s_initial_3 + random.gauss(0, sigma) for j in
    range(rank_max_3)] for i in range(sub_max + 1)])
v_initial_3 = np.asmatrix([[s_initial_3 + random.gauss(0, sigma) for j in
    range(deli_max)] for i in range(rank_max_3)])

# Matrix completion
u_final, v_final, loss_train_1, loss_test_1 = alter_ls(sub_max, deli_max,
    rank_max_1, t_max, alpha, score_train, num_train, score_test, num_test,
    u_initial_1, v_initial_1)
u_final, v_final, loss_train_2, loss_test_2 = alter_ls(sub_max, deli_max,
    rank_max_2, t_max, alpha, score_train, num_train, score_test, num_test,
```

```
    u_initial_2, v_initial_2)
u_final, v_final, loss_train_3, loss_test_3 = alter_ls(sub_max, deli_max,
    rank_max_3, t_max, alpha, score_train, num_train, score_test, num_test,
    u_initial_3, v_initial_3)

iter = [t for t in range(t_max)]

plt.figure(1)
loss_train_1 = [loss_train_1[t] / item_train for t in range(t_max)]
loss_train_2 = [loss_train_2[t] / item_train for t in range(t_max)]
loss_train_3 = [loss_train_3[t] / item_train for t in range(t_max)]
f1, = plt.semilogy(iter, loss_train_1, label = 'Train Loss with Rank = 1')
f2, = plt.semilogy(iter, loss_train_2, label = 'Train Loss with Rank = 2')
f3, = plt.semilogy(iter, loss_train_3, label = 'Train Loss with Rank = 3')
plt.legend(handles=[f1, f2, f3])
plt.xlabel('Nnmber of Iterations')
plt.ylabel('Value of Objctive Function')
plt.title('Obj func v.s. # of Number of Iterations')
plt.savefig('score_2_1.pdf')

plt.figure(2)
loss_test_1 = [loss_test_1[t] / item_test for t in range(t_max)]
loss_test_2 = [loss_test_2[t] / item_test for t in range(t_max)]
loss_test_3 = [loss_test_3[t] / item_test for t in range(t_max)]
f1, = plt.semilogy(iter, loss_test_1, label = 'Test Loss with Rank = 1')
f2, = plt.semilogy(iter, loss_test_2, label = 'Test Loss with Rank = 2')
f3, = plt.semilogy(iter, loss_test_3, label = 'Test Loss with Rank = 3')
plt.legend(handles=[f1, f2, f3])
plt.xlabel('Nnmber of Iterations')
plt.ylabel('Value of Objctive Function')
plt.title('Obj func v.s. # of Number of Iterations')
plt.savefig('score_2_2.pdf')
```

C

Some Data Processing Source Code

C.1

```
import numpy as np
import csv
import pandas as pd
import time
import random
import pylab as pl
from numpy import linalg as la
from collections import namedtuple
import matplotlib.pyplot as plt
import timeit

dict = {}
k = 0
t0 = timeit.default_timer()

# Read info of item
with open('item_to_detail.csv', newline='') as csvfile:
   f_csv = csv.reader(csvfile,delimiter=',')

   for row in f_csv:
      item_id = row[0]
      brand = row[1]
      category_id = row[2]
      sub_category_id = row[3]
      dict[item_id] = [brand, category_id, sub_category_id]
      k = k + 1

t1 = timeit.default_timer()
print(k, t1 - t0)
print(dict['57829'])
print(dict.__contains__(57829))
print(dict['86889'])
print(dict.__contains__('86889'))
```

```
dict_detail = {}

k = 0

with open('msom_order_data_7.csv', newline='') as csvfile:
   f_csv = csv.reader(csvfile,delimiter=',')

   for row in f_csv: # Get info of order
      item_det_info = row[2]
      promise_speed = row[5]
      if promise_speed == '':
         promise_speed = '0'
      if_cainiao = row[6]
      if row[8] != '':
         logistics_score = int(row[8])

         item_list = item_det_info.split(',')
         for item_detail in item_list: # Split different item in one order
            item_info = item_detail.split(':')
            item_id = item_info[0]

            detail = dict[item_id]
            brand = detail[0]
            category_id = detail[1]
            sub_category_id = detail[2]

            item_type =
                brand+':'+category_id+':'+sub_category_id+':'+promise_speed+':'+if_cainiao
            if k == 0:
               print(item_type)

            if dict_detail.__contains__(item_type): # Add logistic rate
               type_detail = dict_detail[item_type]
               score = type_detail[0]
               number = type_detail[1]
               score = score + logistics_score
               number = number + 1
               dict_detail[item_type] = [score, number]
            else:
```

```
            dict_detail[item_type] = [logistics_score, 1]

      k = k + 1

t2 = timeit.default_timer()
print(k, t2 - t1)

# Output accumlated rate and quantity
with open('order_to_item_score_7.csv', 'w', newline = '') as csv_output:
   writer = csv.writer(csv_output)
   for item_type in dict_detail:
      type_detail = dict_detail[item_type]
      writer.writerow([item_type,type_detail[0],type_detail[1]])

t3 = timeit.default_timer()
print(k, t3 - t2)
```

C.2

```
import numpy as np
import csv
import pandas as pd
import time
import random
import pylab as pl
from numpy import linalg as la
from collections import namedtuple
import matplotlib.pyplot as plt
import timeit

k = 0

t0 = timeit.default_timer()

with open('msom_item_data.csv', newline='') as csvfile:

   for row in csvfile:
      if k < 10:
```

```
            print(row)
        k = k + 1

t1 = timeit.default_timer()
print(k, t1 - t0)

k = 0

dict = {}

# Get id, brand, category, sub-category of all item
with open('msom_item_data.csv', newline='') as csvfile:
    f_csv = csv.reader(csvfile,delimiter=',')

    for row in f_csv:
        item_id = row[1]
        brand = row[4]
        category_id = row[5]
        sub_category_id = row[6]
        dict[item_id] = [brand, category_id, sub_category_id]
        k = k + 1

# Output info of item
with open('item_to_detail.csv', 'w', newline = '') as csv_output:
    writer = csv.writer(csv_output)
    for item_id in dict:
        writer.writerow([item_id,(dict[item_id])[0],(dict[item_id])[1],(dict[item_id])[2]])

t2 = timeit.default_timer()
print(k, t2 - t1)
```

References

[1] "Cainiao msom data-driven research competition." https://tianchi.aliyun.com/competition/introduction.htm?spm=5176.100066.0.0.cab63d74jciN6&raceId=231623.