E-Commerce Recommender Modeling

A case study of *Ponpare* Coupon Purchase prediction

Hao Liu[1], Yuhao Zhao[2]
[12] Center for Data Science, New York University
hl2514@nyu.edu, yz3085@nyu.edu

Abstract

Recommender systems are changing the way of E-commerce marketing from massive advertising to more customized strategies. In this project, we implement and evaluate several machine learning methods in the context of developing a recommender system based on Ponpare Coupon data. In particular, we look at a Collaborative Filtering algorithms, a user Clustering algorithm, a Modified Cosine Similarity method and a Conditional model, and compare their performance in terms of MAP@10 measure.

I. Introduction

For a long time, Product recommendation has been challenging for e-commerce platforms like Amazon and eBay. However, in the last few decades, behavioral data mining has matured and gained universal acceptance and usage in the electronic commerce. In this project, we studied and modeled the user purchase behavior from *Ponpare.jp*, which is one of the leading joint coupon e-commerce websites in Japan. Based on the past user coupon purchase and browsing information, we build models to predict a list of coupons a customer will buy in the testing period. This problem was first announced in *Kaggle* as a competition on Thu 16 Jul 2015, and was ended on Wed 30 Sep 2015.

We start with a brief summary of the data set used in the development of the models and the metric that was used to measure performance, followed by each of the various approaches and algorithms we implemented. Under each approach, we discussed possible improvements, and the results are summarized in later sections.

II. Data Summary

The data set we used is Ponpare's coupon data provided by kaggle. [1] This data set contains the following information:

1. User Master: A description of 22,873 users of Ponpare. Features include age, gender, user location and etc.
2. Coupon Master: A description of 19,413 coupons of Ponpare which were available during some certain search period. Features include the price, discount rate, category expiration date, and etc.
3. Purchase log: Purchase history for each user from 2011-07-01 to 2012-06-23. 168,996 purchases in total. For each record, time and area of were recorded.
4. View log: Website Viewing history for each user from 2011-07-01 to 2012-06-23. 2,833,180 records in total. For each visit, the data recorded which coupon was viewed and whether the user made purchase.
5. Coupon Listing Area: Extra coupon location information for training data

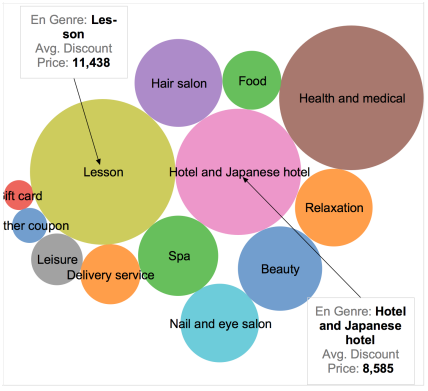

Figure 1: *Discount Price vs Category*

[1] *https://www.kaggle.com/c/coupon-purchase-prediction*

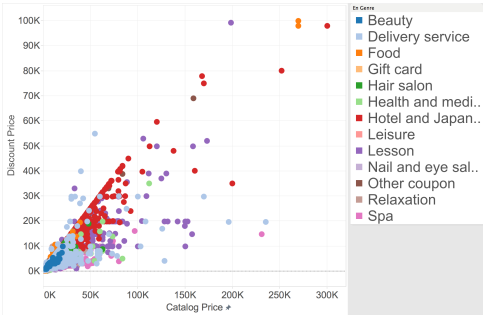

Figure 2: *Catalog vs Discount Rate*

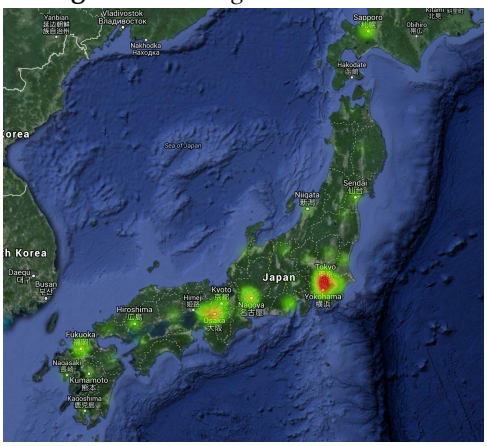

Figure 3: *User Distribution*

In the training set, we have 52 weeks of purchasing and website visiting data. During the modeling stage, we separated the last ten weeks (from April 19 to June 23) as the validation set. To produce the final model, we used the whole 52 weeks data.

In the user data set, we transformed the user Residential Prefecture using one hot encoding. In the coupon detail data, we first conducted one hot encoding to the coupon area and categories, and then transform the discount price using inverse log discount price. For some of our proposed models, we normalized the training data by features, for other models such as random forest (scale is irrelevant) we did not normalize.

III. Evaluation Metrics

$$MAP@10 = \frac{1}{|U|}\sum_{u=1}^{|U|}\frac{1}{min(m,10)}\sum_{k=1}^{min(n,10)}P(k)$$

where $|U|$ is the number of users, $P(k)$ is the precision at cutoff k, n is the number of predicted coupons, and m is the number of purchased coupons for the given user. If m = 0, the precision is defined to be 0.

IV. Approach

I. Baseline mode (User based Naive Bayes):

Our first approach is to build an empirical model based on user visiting history. For each user, we computed the historical visiting counts under each coupon categories. In order to smooth the distribution, we add 0.5 to all the categories. Meanwhile, we constructed Naive Bayes models which simply calculate the percentage of purchase under each category. To predict the purchase, we first sample 50 coupons from the testing set with replacement followed by the smoothed user empirical visiting distributions. Among the sampled testing coupons, the likelihood of purchasing is given by the Naive Bayes Model. Finally, the recommendations was done by selecting the 10 sampled coupons with highest likelihood.

II. User-Based Collaborative Filtering

As our first implement, we tried to measure the user-wised and coupon-wised similarities. We regard the purchase history as users' additional features, since we believe the purchase history can indicate their patterns.

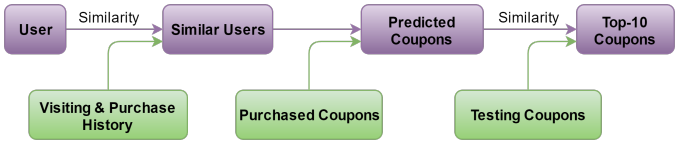

Figure 4: *Model 1 Flow Chart*

The first step is to build the user-coupon matrix. Then for each user, we attempted to find the 100 most similar users based on cosine similarity. Due to the reason that the user-coupon matrix has high dimension and sparsity, we need to reduce the dimensions in order to increase clustering efficiency. Therefore, we applied singular value decomposition, and reduced the dimension of user space into 10. In this lower dimension space, we found 100 most similar candidates for each user. Afterward, in the full matrix, using the normalized similarity as weight, for each user, we predict the score of each coupon by weighted average the purchase counts of the

100 candidates, in particular:

$$u^*_{ij} = \sum_{u_{kj}=1} u_{kj} v_{ik}$$

$$v_{ij} = cos(u_i, u_j) = \frac{\sum_{k=1}^{m} u_{ik} u_{jk}}{\sqrt{(\sum_{k=1}^{m} u^2_{ik} u^2_{jk})}}$$

In the above equation, $u_{i,j}$ is the purchase of users i for coupon j. It is 1 if purchased, and 0 otherwise.

After calculating and selecting the highest 10 scores, we got the 10 coupons in the training set that a user would buy in the future. However, these coupons were already expired. The last step is to find the most similar coupons to the predicted training coupons in the testing set by cosine similarity.

Improvements:

1. *When calculating user similarity, we only considered the original feature and purchase history. One improvement in this approach would be adding more user related features such as visiting record and user-coupon joint features.*

2. *When searching similar users, we reduce the dimension of user space to optimize the computing efficiency. This process may hurt the prediction due to the lost of information. One way to alleviate this would be using Locality-Sensitive Hashing(LSH).*

III. User Clustering

In this approach, we first consider the similarity of users. For users having similar features or visiting histories, their purchase pattern tends to be very similar. In order to classify users, we conducted user clustering based on cosine similarity. In particular, we first added several generated features to each user, including total purchasing counts, website visiting counts, active days counts and favorite category. Then we represent each user by his/her top 10 principal components, and run clustering.

In each user clusters, we fitted logistic regression based on all the coupons that the users in this clusters has visited. The label here is purchased or not. In particular, each model predicts whether the users in this cluster will buy the coupon given its coupon features. Since logistic regression gives the likelihood of purchasing, recommendations for users was the top 10 likely coupons. We did several experiments with different cluster numbers, and base on which we tuned parameters.

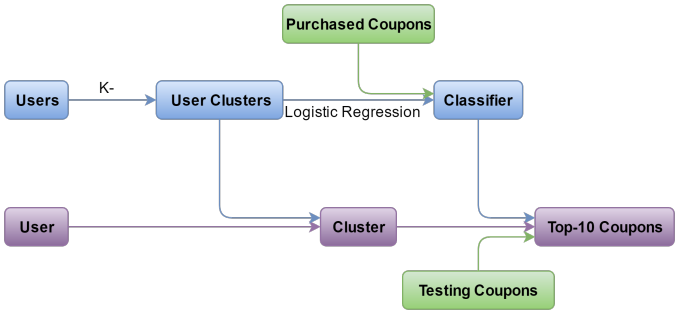

Figure 5: *Model 2 Flow Chart*

We noticed that with more user clusters, the overall recommendation gets better, while the error for each model gets larger. The reason is that since the recommendations for users in the same clusters are identical, more clusters will generate more user purchase patterns. However, more clusters means fewer samples in each cluster, training models in each cluster became harder.

One issue with this approach is that for users in the same cluster, the model would predict exact the same list of coupons. Thus, we could expect that with more user clusters, the model would perform better. And it was indeed consistent with our experiment results. In the extreme case, we need to eliminate the clusters and built models for everyone individually. However, in that case, the training data would be very unbalanced.

Although the testing result is not very good for this approach, we think this method would be useful for users who is completely new or has very few visiting history.For those users we can still find the corresponding user cluster and make recommendations.

Improvement:
We can analyze the importance of coupon features in each different user clusters. As we can see in the next approach, applying different feature weights in different user groups can indeed improve the performance

IV. Modified Cosine Similarity

In the previous model, we noticed that users can have dramatic change in purchase patterns and clustering based on cosine similarity won't capture those change. What's more, one big problem is that most of the coupons in the testing data set are brand new.

In this approach, we defined and trained modified cosine similarity which basically first transform(scale) the features and then measure similarities. The idea is that for each user, we weighted average his/her purchased coupons and use this as user representation. Since not every coupon features are equally important, our target is to scale the features before looking for similar coupons in the testing(validating) coupons. In particular we want the transformed user features to be similar to the coupons that he has purchased and not similar to those that he viewed but not purchased. We defined the loss function as :

$$L(w) = \sum_{U_i}\Big(\sum_{C_i \in Si,1} \frac{(w \odot U_i) \cdot C_j}{||w \odot U_i|| \times ||C_j||} - \tag{1}$$

$$\sum_{C_i \in Si,2} \frac{(w \odot U_i \cdot C_j}{||w \odot U_i|| \times ||C_j||}\Big) \tag{2}$$

Where, $S_{i,1}$ are the coupons that user i has visited but not purchased, $S_{i,2}$ are the coupons the user i has purchased. Since the loss function is not convex, we minimized the function by greedy algorithm. After initializing the weights, in each step, we randomly selected one dimension i and search the best w_i in that dimension from 0 to 1.

After training the weights, the recommendations for each user is selected by the top 10 most similar coupons in the testing set with the scaled user feature representations.

We also tried training different weights for Male and Female, the performance gets better. The interesting fact is that weights for coupon category and discount price are significantly larger for male, while the usable date, coupon larger area name and display day count are larger for female.

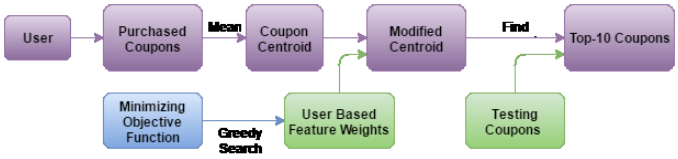

Figure 6: *Model 3 Flow Chart*

This approach achieved the best testing result in our experiments. We believe the reason is that we use the mean of purchased coupons to represent the user. In this representation, it embeds the user's preference for coupons. And it shows that the behaviour pattern of user may be less important in our project, which means the order of visiting and purchase need not to be considered.

Improvements:

1. *As we applying different feature weights for Female and Male users, the performance gets better. Thus we can combine approach 2 and 3 together to improve the performance, which means first clustering the users, and then training modified feature weights within each cluster.*

2. *Another issue for this approach is that it completely ignores the visiting history. We believe that the browsing history is also informative, if we could include visiting as features in our prediction, the performance may improve.*

V. Model 4: Conditional Model

As our final approach, we constructed a two-step model. The first model calculates the probability of the user clicking/browsing the coupon, and then given the coupon features the second model calculates the probability of purchasing it. The likelihood of purchasing is, therefore, the product of these two models.

To model the clicking probability, we used the smoothed empirical user category visiting distribution. For example, we selected two users, their visiting distributions are shown as below. We can see that the first user obviously prefers Food and delivery coupons, and the second user prefers Nail and eye salon coupons. What's more, we used this distribution as well as the purchase history distribution as features for user. Since there

are 14 categories in total (including category 'NA' for missing value), we added 28 additional features.

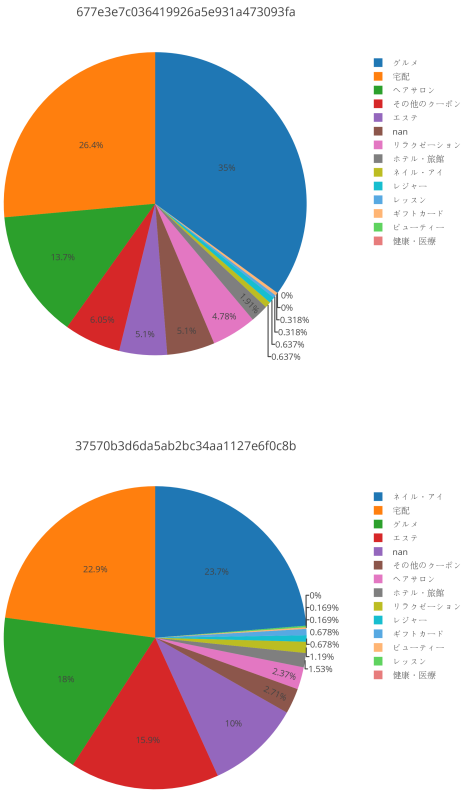

Figure 7: *User Example*

To model the coupon purchasing, we also generated a set of additional features for coupons including the count, average discount price, and average discount rate for coupons of same category within the same period, and the count, average discount price, and average discount rate for coupons of same large area within the same period. Afterward, based on the coupon and users features we generated user-coupon pairs for all users and their visited coupons.

In each pairs we also added some user-coupon interaction features, e.g. whether this coupon is in the top-3 favorite visiting categories, whether the area for this coupon is the same as the user's registration area. The target for each pair is set to 1 if the user purchased the coupon and 0 otherwise.

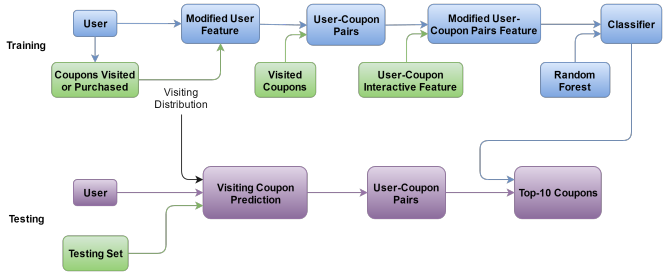

Figure 8: *ROC for Model 4*

For each user-coupon pair, it's a binary classification problem. According to the prior statement, our classifier would base on the condition that the user has visited this coupon website. To train our classifier, we tried several machine learning algorithm and in this stage, we applied AUC on the validation set to evaluate the performance the corresponding performance. We tested the follow models:

1. Logistic Regression: We applied l2 penalty for regularization and tuned the regularization constant.
2. Adaboost: We applied decision stumps as the base estimator and used a total of 200 estimators.
3. Random Forest: We used 100 estimators and have tuned the 'max_depth', 'min_sample_split' to get the best result.
4. Gradient Boosting: We used 100 estimators and have tuned the 'max_depth' and "min_sample_split"

In Figure 9, we compared the ROC for different models. And we found out that Random Forest with 100 estimators, max_depth = 20 and min_split_sample = 5 achieved the best AUC. And we used this as our final model for prediction.

In testing stage, for each user, we gave a weight for each coupon in the test set based on the category empirical distribution and sample 50 coupons with replacement. In this way, coupons in the user's preferable categories will have a greater chance to be sampled. Based on the samples, we created user-coupon pairs, and for each pair, we calculated the probability of purchasing using random forest classifier. The overall probability of those samples are the product of the probability of being visited and the probability of being purchased

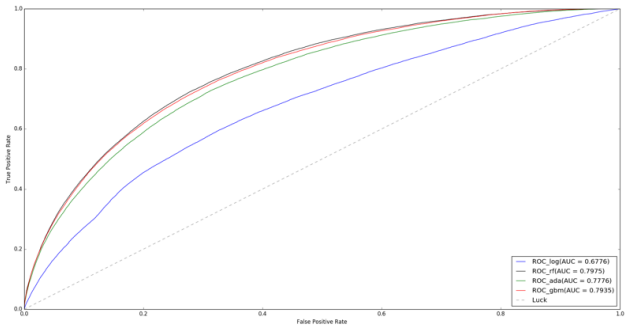

Figure 9: *ROC for Model 4*

Model	**Map@10**	**Description**
Baseline	9.25×10^{-4}	
UBCF	2.15×10^{-3}	
Modified Similarity	5.40×10^{-3}	Same weights for all users
	7.59×10^{-3}	Different weights for Male and Female
User Clustering	1.05×10^{-3}	100 Clusters
	1.95×10^{-3}	1000 Clusters
Conditional Model	$1.52 \times 10^{-}3$	Random Forest as classifier

Figure 10: *Model Summary*

given visited.

We also tried creating pairs for all testing coupons. However it performed worse. The reason is that there will be totally 6 million pairs in testing, however, we only have 1.6 million pairs in the training set and 0.4 million in validation set. What's more, for some users, their preference may focus on a very small number of categories. If we create all the pairs, it may give very random predictions.

One issue for this method is that the we can't directly use the MAP@10 Metrics in the training. In order to tune the parameters for each model, we used AUC as the criterion to compare the performance.Thus although we achieve a good AUC on validation set by Random Forest. The testing result is still not good enough.

Improvement:

In our approach, we simply used the empirical distribution to predict the visiting probability. One way to improve is to build a better model for visiting. In this case, the training data will be very large and unbalanced. Because for each user, most of the coupons have never been clicked.

V. Result and Discussion

The table of model performance on the testing data was summarized in figure 10. During the experiments, we found out that the visiting and purchasing history indicates little information about the user consumption behavior(the order of clicking and purchase has little information). Instead, the most important factor(find a good representation for the user preference and then) is to modify the product similarities since the coupon data contains more useful features. However, the contribution of each coupon features may vary for different user groups.

Although our best approach involves no information of visiting history, we still think it would be helpful to include it in some certain way during training process. A perfect visiting model would reduce the number of

likely candidates in the test set, and therefore improve the performance.

References

[Figueredo and Wolf, 2009] Figueredo, A. J. and Wolf, P. S. A. (2009). Assortative pairing and life history strategy - a cross-cultural study. *Human Nature*, 20:317–330.

[Cheung, 1999] Cheung, William K and Kwok, James T and Law, Martin H and Tsui, Kwok-Ching (1999) Mining customer preference ratings for product recommendation using the support vector machine and the latent class model. *Citeseer*

[Adomavicius, Gediminas, and Alexander Tuzhilin, 2005] Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on 17.6 (2005)* 734-749

[Yang, Shuang-Hong, et al, 2011] Collaborative competitive filtering: learning recommender using context of user choice. *ACM, 2011.*

[Zhang, Yuchen, et al, 2014] Taxonomy discovery for personalized recommendation. Proceedings of the 7th ACM international conference on Web search and data mining. *ACM, 2014*