Building an Ecommerce Recommendation System

Kun Qian, Xingzi Xu, Emily You

June 2020

1 Introduction

Recommendation systems allow businesses to interact with large and complex spaces in prioritizing items that may be of interest to the users. Personalized recommendation systems play a crucial role in many web services, such as Amazon, Netflix, and Youtube. In the world of e-commerce, especially, such recommendation systems are becoming important business tools by influencing the customers' decision-making processes.

Instacart is a grocery delivery service that also utilizes recommendation systems. The app allows consumers to easily order groceries when they need them. After the user selects desired items on the Instacart app, shoppers go into the store, shop, and deliver the products to the user. In order to make the shopping experience more efficient and enjoyable for the user, we developed models to predict which items that have not been bought by the user are likely to be purchased next. This could deliver business benefits to Instacart by increasing the average order value. We also compared our models to the popularity-based model (top 10) to see if they are more indicative of what the user actually purchases.

The input to our algorithm is the user, product, and order information from Instacart. The data set includes information of each product, where they belong at the grocery store, order hour of the day, days since prior order, user id, etc. The data is further described in detail in Section 3 of this paper. We then use three different collaborative filtering methods, including neighbor-based method with TF-IDF, matrix factorization with singular value decomposition (SVD), and matrix factorization with alternating least squares (ALS) to output 10 recommended items for each user.

2 Related work

Due to the growing popularity of e-commerce, the development of the automatic recommendation system, especially for implicit feedback, has attracted increasing attention. The existing recommendation systems are mainly based on two strategies, content-based approach and collaborative filtering(CF).(1) While content-based approach requires external information of customers, such as product reviews and personal information, CF can generate recommendations just based on past user behavior. However, CF suffers the cold start problem and thus cannot recommend new items until a minimum number of customers have used or rated those items.(2)(3) Due to their different nature, both approaches are discussed and evaluated in literature.

TF-IDF, a sub-area of Natural Language Processing (NLP), is a popular content-based method for feature extraction in information retrieval and have also applied in both explicit and implicit recommendation. It is also often used in modeling recommendation systems with CF. Chunliang Lu, Wai Lam and Yingxiao Zhang developed a model to extract a Twitter user interest profile with tweets and rank each tweet by affinity between users with TF-IDF.(4) TF-IDF term extraction method with cosine similarity measure is also applied to food recommendation based on healthy heuristics and standard food database.(5)

Considering the lack of explicit user feedback, CF has gained more popularity in product recommendation. The term collaborative filtering was first coined by Goldberg etal and CF was first used to build an e-mail filtering system that allowed users to annotate messages.(6) CF approaches applied in recommendation system can be separated into two kinds of models: neighborhood models and latent factor models. (8) The first automated CF system using a neighborhood-based method was introduced in GroupLens. (7) Improved accuracy, better scalability and good explanation power make item-oriented methods more popular than user-oriented methods among neighborhood models. (9) (10) Item-oriented models all lack flexibility in distinguishing user preferences and confidence in those preference, and thus are not applicable in cases with implicit feedback. Latent factor models have better performance in making recommendations with implicit feedback by uncovering latent features behind observed rating or buying frequencies. Latent factor models induced by SVD and ALS are commonly used in e-commerce due to their attractive accuracy and scalability.(11) SVD and ALS, both matrix factorization (MF) algorithm, decompose the interaction matrix to fix the sparsity problem with more condensed matrix.

3 Dataset and Features

'The Instacart Online Grocery Shopping Dataset 2017" comes from the Instacart website and covers information of 3.4 million purchases by more than 200,000 customers. There are six csv files in total: aisles, departments, products, orders, order_products__prior, and order_products__train.

First, combining aisles, departments, and products csv, we can get a joint table with information of each product and where they belong in the supermarket. Second, orders file gives us an overview of all the orders including order id, user id, if the data point belongs to train set or validation set, order number, day of the week, order hour of the day, and days since prior order. This file helps us to connect the orders information with the user id in order to build the user-products matrix in the further analysis. Finally, order_products

__prior and order_products__train have the same data frame structure and they serve as the train set and validation set for model training and evaluation respectively. The training set has 206,209 orders and the validation set has 131,209 orders. Each file gives us information about which product (product_id) was ordered in which order (order_id), the order (add_to_cart_order) in which the products were put into the cart and if the product was reordered (1) or not (0). For the purpose of our analysis, we are only considering the feature of reordered, which could be transformed to an approximation of user's rating of the products. It is noted that, however, more complex predictive models can take time components into account, including features such as (order_hour _of_day) and (days_since_prior_order).

From preliminary data exploratory, we find that the distributions of train and validation set are comparable and people most often order around 5 items. The top ten popular products on the platform are: Bananas, Organic Bananas, Organic Strawberries, Organic Baby Spinach, Lemon, Organic Avocado, Organic Hass Avocado, Strawberries, Limes, and Organic Raspberries. This list of products would be used to build the baseline model, in which the recommendation system recommends the top ten popular goods to all the customers without catering to each individual's choice.

Since we do not have explicit data of customers' preference of each products, we are choosing number of reorders of a certain products to approximate the customers' ratings. Therefore, we created the user_product utility matrix that has user as rows, products as columns, and reordered quantity as the entries.

4 Methods and Evaluation Metrics

4.1 Baseline Model: Popularity-based Model

In popularity-based model, we ranked all items by their total sales as a measure of their popularity. We recommend the top 10 popular items to all users, except items that they have already purchased.

4.2 Neighbor-based Method with TF-IDF Weighted Matrix

TF-IDF (term frequency - inverse document frequency) weighting is mainly used to negate the effect of high frequency terms in determining their importance.

TF-IDF weight is defined as:

$$W_{td} = tf * log(N/df)$$

where
tf = number of occurrences of t in document d
df = number of documents that contain the term t
N = total number of documents in the dataset

In this project, each user is treated as a "document" and each product is regarded as a "term." To generate recommendations for a certain user (target user), we find products a similar user purchased that the target user did not buy.

Cosine similarity was used as a measure to define similarity among users.

$$cosine\ similarity(i,j) = \cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2}$$

The i and j vectors represent two user histories.

In this neigbor-based method, top K users similar to the target user defined by cosine similarity and were iterated from the most to the least similar. In this example, K was arbitrarily set to 20. Then a list of recommendations was generated by taking products that the similar user purchased but the target user did not purchase. The size of the list was capped to N products and N was arbitrarily chosen to be 10 in this project. Then, these N items were recommended according to the frequency of purchase in overall sales.

4.3 Matrix Factorization

CF generates recommendations by learning a user's preference from his/her engagement with certain items and his/her engagement with users who have engagement with the same items. The engagement between user and user, and the engagement between item and item are both represented by the similarity matrix. To implement the user-based CF, we built a user-based matrix from a user-item matrix, which is a co-clustering problem. In our Instacart case with the implicit dataset, entries (r_{ui}) of a user-item matrix (R) indicate observations for user actions, indicating the number of times a user u purchase a item i on Instacart.

Matrix Factorization (MF), one of the most popular algorithm to solve co-clustering problem, factorizing R into a user-factors vector $X \in R^{m\times k}$ and an item-factors vector $Y \in R^{n\times k}$, i.e. $R \approx X \times Y$, where m is the number of users, n is the number of items, and k is the number of features. MF solves the sparsity problem of R because X and Y are more condensed. Once R is built with the training data, we can predict whether to recommend an item i to a user u by taking an inner product, i.e. $\hat{r}_u = x_u \times y_i$. R is built based on the following assumptions:

1) Each user can be described by k features.

2) Each item can be described by the same set of k features.

4.3.1 Matrix Factorization with Singular Value Decomposition (SVD)

SVD is a matrix factorization technique that is usually used to reduce the number of features of a data set by reducing space dimensions from N to K where $K > N$. SVD reduces the dimension of the utility matrix by extracting its latent factors. In the context of collaborative filtering, SVD is used in order to recover the very sparse and huge utility matrix with the two simpler matrices and a sigma matrix.

$$R = U\Sigma V^T$$

Here, U represents the product factors and V represents the user factors and σ is a diagonal matrix. The columns of U can build back all of the columns of R and the columns of V can build back all of the rows of R.

It is noted that SVD is simply an algorithm for matrix factorization. It is efficient at most times but it also has several drawbacks. First, since a data point in SVD is represented as a linear combination of vectors in the basis, if the data is strongly non-linear then SVD might not work well. Second, the SVD matrix is very hard to interpret.

4.3.2 Matrix Factorization with Alternating Least Square(ALS)

To approximate original matrix R with two matrices X and Y, MF is a optimization process minimizing the following cost function:

$$J = \|R - X \times Y^T\|_2 + \lambda \cdot (\|X\|_2 + \|Y\|_2)$$

The first term is the Mean Square Error (MSE), measuring the distance between R and its approximation $X \times Y^T$. The second term is the regularization term to prevent over-fitting problem from irreducible noises. This cost function is non-convex because x_{uk} and y_{ik} are all unknown.

Alternating Least Square(ALS) fixes this non-convex problem through a two-step iterative optimization process. In each iteration, it first fixed X and solves for Y, and then fixed Y and solves for X. In other words, the problem is simplified to a linear regression problem, which can be easily solved with the Ordinary Least Squares(OLS). ALS is guaranteed to converge to a local minima due to the nature of OLS. Since OLS is unique and ensures a minimum MSE, the cost function either decreases or remains unchanged in each iteration.

In our Instacart case, the algorithm developed by Yifan Hu, Yehuda Koren and Chris Volinsky(9) is applied with *AlternatingLeastSquares* in library called *implicit.als*. Compared to the basic ALS, this algorithm has better performance in analyzing implicit feedback. Implicit user feedback r_{ui} is transformed into two paired magnitudes: a binary variable p_{ui} and confidence level c_{ui}.(8)
p_{ui} is derived from r_{ui} such that:

$$p_{ui} = \begin{cases} 0 & p_{ui} > 0 \\ 1 & p_{ui} = 0 \end{cases} \tag{1}$$

Confidence level c_{ui} is introduced to measure our confidence of seeing p_{ui}. As r_{ui} grows, we have higher confidence that the user will purchase the item. The proper choice for c_{ui} is that

$$c_{ui} = 1 + \alpha \cdot r_{ui}$$

The cost function therefore becomes:

$$\sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda(\sum_u \|x_u\|^2 + \sum_i \|Y\|^2$$

The solutions for these two cost functions are:

$$x_u = (Y^T C^u Y + \lambda I)^{-1} Y^T C^u p(u)$$

$$y_i = (X^T C^i X + \lambda I)^{-1} X^T C^i p(u)$$

5 Experiments/Results/Discussion

Due to the limitation of computation time, we choose to run our recommendation models on line 1 to 20,000 (validation set 1) and line 20,001 to 40,000 (validation set 2) of the validation data for model evaluation and comparison.

5.1 Evaluation Metrics: F1 score

The performance of a recommendation model is mainly measured by two metrics: precision and recall.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Precision is a better measure of accuracy when the costs of False Positive is high, while recall is a better measure of accuracy when the costs of False Negative is high. Since our objective for the recommendation model is to obtain higher True Positive recommendation, the cost of False Positive and False Negative is the same. To balance between precision and recall, we choose to use F1 score, a function of precision and recall.

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

5.2 Results

We found that among all the methods we tried, TF-IDF performs outstandingly better than matrix factorization methods. All methods give a better F-1 score than the baseline model, which does not account personalization into account. Also, all models have relatively stable performance and no obvious overfitting problem due to their similar performances on different validation sets.

Instacart Recommendation System Models		
Method	Validation Set 1	Validation Set 2
Baseline	1.53%	1.54%
TF-IDF	16.04%	16.11%
SVD with 3 factors	2.00%	2.00%
ALS with 25 factors & $\alpha = 10$	2.34%	2.29%

For the neighborhood-based method with TF-IDF weighted matrix, the hyperparameters, K (number of similar users to consider) and N (number of products to be recommended) were arbitrarily set to 20 and 10, respectively. With parallelization of the algorithm, it would be more feasible to perform cross validation to fit the hyperparameters. However, in this experiment, computation time was a limiting factor in determining the best hyperparameters to use. This algorithm performed very well compared to the other two methods due to the assumption that not all products are equally weighted in making recommendations. Also, it is possible that Instacart already has a similar recommendation system in place that has affected its users' purchases. The TF-IDF model would recommend products that a user has not purchased yet but is likely to purchase once they are given the recommendation.

In the tuning process for SVD method, we randomly chose small subsets of 200 samples and observed the average performance of a range of factors number. The process narrows down the potential range to be less than 30 so in the end we chose 3, 5, 10 and 25 as number of factors. Overall, the method of matrix factorization with SVD is not very successful. The F-1 results are only slightly higher than that of the baseline model. It is also noted that the increasing factors number might cause overfitting on the training set so that large factors numbers give lower F-1 score.

For the ALS method, we experimented on two hyper-parameters to build the best models with respect to the F-1 scores and the detailed results are included in the appendix:

1. Latent feature parameter (α): α controls the relation between the matrix entry r_{ui} and the confidence variable c_{ui} with $c_{ui} = 1 + \alpha \cdot r_{ui}$. We found that α gave the best F-1 score in out experiment.

2. The number of latent feature: We used 25 as the number of latent features of the best ALS model after trying 10, 25, 50, 75, and 100. The improvement from 10 to 25 is more significant from 25 to higher number of latent factors. Therefore, 25 is chosen to balance between F1-score and over-fitting problem.

6 Conclusion/Future Work

Out of the collaborative filtering methods that were tested, neighbor-based recommendation with TF-IDF weighted matrix had the highest F-1 score. This method was followed by ALS with 25 factors and $\alpha = 10$ and SVD with 3 factors in the order of decreasing F-1 score.

TF-IDF weighted matrix performed better because of the assumption that not all products are equally important in calculating the conditional probability.

In traditional recommendation system, in the feature space for users, each factor measures how much the user likes movies that score high on the corresponding movie factor. However, in our Instacart case, we do not have negative preference from the customers. The greater the reordered quantity, the more popular the product is. But 0 or null entries does not necessarily mean low ratings. Therefore, the performance of matrix factorization is not ideal in our modeling. This indicates that models built with matrix factorization would generate recommendations that are irrelevant, or items that the user is unlikely to buy. In ALS and SVD, missing entries are equally weighted, which may lower the model's accuracy and explanation power.

Further research is needed to improve the performance of the models, especially in the following aspects:

* Select better hyperparameters in each method. Because of the limitation of computational complexity, a thorough search of hyperparamters was not conducted for the TF-IDF method. Use parallelization to improve the running time.

* Find the balance between improved the accuracy Matrix Factorization with ALS algorithm and the increased computational complexity. For example, introducing additional parameters to treat the user-item interaction with zero value with different meanings.

* Include more available factors, such as purchase time, department, and repurchase habits, to make further improvement in F-1 score and explanation power of models.

* Use tree models and simple neural networks models to build the recommendation systems.

7 Appendices

SVD Method		
Method	Validation Set 1	Validation Set 2
SVD with 3 factors	2.00%	2.00%
SVD with 5 factors	2.01%	2.00%
SVD with 10 factors	1.93%	1.89%
SVD with 25 factors	1.71%	1.66%

ALS (# factors = 10)		
α	Validation Set 1	Validation Set 2
10	2.16%	2.14%
15	2.08%	2.08%
20	2.02%	2.05%
25	1.99%	1.99%
30	1.92%	1.95%

ALS (# factors = 25)		
α	Validation Set 1	Validation Set 2
10	2.34%	2.29%
15	2.21%	2.19%
20	2.16%	2.21%
25	2.07%	2.01%
30	2.01%	1.96%

ALS (# factors = 50)		
α	Validation Set 1	Validation Set 2
10	2.30%	2.33%
15	2.23%	2.19%
20	2.15%	2.10%
25	2.05%	2.04%
30	2.00%	1.95%

ALS (# factors = 75)		
α	Validation Set 1	Validation Set 2
10	2.36%	2.34%
15	2.25%	2.25%
20	2.15%	2.12%
25	2.05%	2.04%
30	1.98%	1.97%

ALS (# factors = 100)		
α	Validation Set 1	Validation Set 2
10	2.39%	2.41%
15	2.25%	2.26%
20	2.19%	2.17%
25	2.05%	2.04%
30	2.01%	1.97%

8 Contributions

Each member contributed equally. Each member explored one algorithm, with Emily You working on TF-IDF, Xingzi Xu working on SVD and Kun Qian working on ALS. The rest, including debugging, composing report and drawing conclusions, was done jointly by all members.

The link to the project Github repository is: https://github.com/yemilyou/CS229-Rec-Project

References

[1] Y. Koren, R. Bell and C. Volinsky, *Matrix Factorization Techniques for Recommender Systems*. vol. 42, no. 8, pp. 30-37, Aug. 2009, doi: 10.1109/MC.2009.263.

[2] O'Mahony, Michael P., and Barry Smyth. *From Opinions to Recommendations*. Springer, Cham,2018

[3] Kanavos, A.; Iakovou, S.A.; Sioutas, S.; Tampakas, V. *Large Scale Product Recommendation of Supermarket Ware Based on Customer Behaviour Analysis*. Big Data Cogn. Comput. 11 Feb, 2018

[4] Chunliang Lu and Wai Lam and Yingxiao Zhang.
Twitter User Modeling and Tweets Recommendation Based on Wikipedia Concept Graph. AAAI Workshops, 2012, `https://www.aaai.org/ocs/index.php/WS/AAAIW12/paper/view/5262/5629`

[5] El-Dosuky M.A., Rashad M.Z., Hamza T.T., EL-Bassiouny A.H. *Food Recommendation Using Ontology and Heuristics. In: Hassanien A.E., Salem AB.M., Ramadan R., Kim T. (eds) Advanced Machine Learning Technologies and Applications.* AMLTA 2012. Communications in Computer and Information Science, vol 322. Springer, Berlin, Heidelberg

[6] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. *Using collaborative fillering to weave an information tapestry.* Communications of the ACM, 35(12): 61-70, 1992.

[7] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. *GroupLens: Applying collaborative filerting to Usenet news.* Communications of the ACM, 40(3):77-87, 1997

[8] Y.F. Hu, Y. Koren, and C. Volinsky, *Collaborative Filtering for Implicit Feedbck Datasets.* Proc. IEEE Int'l Conf. Data Mining (ICDM 08), IEEE CS Press, 2008, pp. 263-272

[9] B. Sarwar, G. Karypis, J. Konstan and J. Riedl, *Item-based Collaborative Filtering Recommendation Algorithms.* Proc. 10th International Conference on the World Wide Web, pp. 285-295, 2001

[10] R. Bell and Y. Koren, *Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights.* IEEE International Conference on Data Mining (ICDM'07), pp. 43–52, 2007.

[11] Z. Huang, D. Zeng and H. Chen, *A Comparison of Collaborative-Filtering Recommendation Algorithms for E-commerce.* IEEE Intelligent Systems 22 (2007), 68–78.

[12] https://github.com/verma-rahul/RecommendationSystem