Introduction to Machine Learning - Project: Black Friday - Recommender System

Azza Abdalghani, Giulia M. Milano Oliveira, Keerthana Chandrasekar Jeyanthi

1 Problem statement

The goal of this project is to design, implement and evaluate a recommender system (RS) using the Black Friday data-set to recommend a product that will be most liked by a given user. The idea is that consumers are most likely to purchase items that are more relevant to them, therefore recommender systems are crucial to increase revenue in online services, such as e-commerce and entertainment on demand.

2 Proposed solution

The basic idea of a RS is to recommend the customers their most preferred products through the ratings given by the customers for their previously purchased products. Here, we do not explicitly have the ratings given by the customers. To handle the above problem, we propose two ideas:

- Using only the purchase history of customers with the assumption that the customers have bought the products that they liked. This gives same ratings to all the products that the customers bought.
- Formulate a rating that the customer might have given. For this we assume a pattern of customer behaviour. We assume that the customers who tend to spend more money might tend to buy costly products while the customer who tend to spend less might tend to buy cheap products. Hence, if the Purchase amount of the Product_ID is near to the average amount spent by the User_ID, then the customer might prefer that particular product more than the other products.

With the above two ideas, we first try to implement a very simple model, the "popularity model", which recommends the most popular items to the user, as a baseline.

For a content-based RS it is required that the products have features, so that it can recommend a product to a consumer similar to one that it has liked before. With the data we have, we face two problems:

- We need product features to build a content-based RS. We can consider Product_Category_1, Product_Category_2, Product_Category_3 as product features, but Product_Category_2 & Product_Category_3 have a lot of missing values which cannot be imputed by 0's or mean values.

- Further, the implementation would consist of performing one regression model for each user using product features and there are 5,891 users, hence we need to build 5,891 regression models. This would require considerable computational effort.

Thereby, we decided to eliminate this approach.

Next, we evaluated if we could then use a Collaborative Filtering(CF) RS, which in turn make recommendations based on historic users' preference for items, such as items purchased. As the data provides User_ID and Product_ID together with the Purchase amount, we decided to apply the two main types of CF, the user-based CF and the item-based CF.

Finally, we tried to see if a Singular Value Decomposition(SVD) can be done so as to improve the computational efficiency of the CF algorithms.

3 Experimental evaluation

3.1 Data Preparation

The data set contains 5891 unique User_ID's and 3623 unique Product_ID's. Each entry of the User_ID represents a product bought by the customer with the Purchase amount in the Purchase column, hence each row in the data can be considered as a transaction.

To incorporate our desired data into a RS algorithm,we build a matrix with User_ID as rows and Product_ID as columns and call this as the user-item matrix. The user-item matrix is filled with the ratings given by the customers.

As previously stated, we do not know the ratings of the user explicitly. Hence, we try to implement the two ideas as proposed before.

- The first idea was to use only the customer's previous purchase behaviour. This lead to the use of **"0-1 Data"** [1]We fill the user-item matrix with 0's and 1's, 1's representing that the customer has bought that product, 0 otherwise. This matrix is known as the $binaryRatingMatrix$

- The second proposal was to formulate a rating. The goal of the rating is to be such that the RS recommends cheaper products to customers who prefer to spend less and costlier products to customers who prefer to spend more. The formulation of the rating is discussed in detail below. We fill the user-item matrix with ratings obtained and this is known as $realRatingMatrix$

As per the second proposal, we formulate the rating as follows:

- Assume the buying capacity of the customer is the average amount of money spent by him in the previous purchases. (i.e. the average Purchase amount for each User_ID in the data-set)
- Rate the product higher if the Purchase amount of the product is closer to the buying capacity of the customer.
- For this, divide the Purchase amount(P) of the product by the buying capacity of the customer(B) if P < B, or divide B by P if P > B.
- Thus we obtain ratings in the interval $[0, 1]$. If the rating is closer 1, then the Purchase amount of the product is closer to the buying capacity of the customer.

3.2 Assessment

For assessment of our RS algorithms, we use a 5-fold cross validation. We choose a 5-fold CV as a CV takes incredible computational effort, hence sticking to the minimum common value of k = 5.

Moreover, how to analyze if the recommender system has correctly recommended the customer's preferred product is a bit of challenge, as there is no explicit way of checking the same. So, when predicting the customer's preferred products, we randomly withhold some product ratings from the test data in each cross validation fold. These withheld products are assumed to be the customer's preferred products as they have actually purchased them and might have rated them also. For performing such assessment, we resort to $Given10$ protocol [1]. As per the protocol, 10 randomly chosen products from the test data are given to the RS to predict and the remaining products are used for evaluation. To use this protocol we require that each customer must have bought at least more than 10 products. Hence,for the sake of evaluation, we filter the data so that the customers have bought more than 50 products.

3.3 Evaluation metrics

For the comparison of our RS algorithms, we first tried to use the *recommenderLab* inbuilt metrics of precision and recall but there were some conflicting properties, where a low precision means high recall and vice-versa [1]. Hence, we tend to calculate $Accuracy@k$ with k = 10(a realistic value) and use it as our main metric.

3.4 Implementation

Following from Data Preparation, we convert the user-item matrix filled with ratings into the $binaryRatingMatrix$ and $realRatingMatrix$. We gave the rating matrices to the RS algorithms namely the Popularity model, User Based CF, Item Based CF and SVD (only for $realRatingMatrix$) and assessed with a 5 fold CV and $Given10$ protocol.

While using a binaryRatingMatrix,we consider all 0's to be missing ratings. In such cases, the Cosine or Pearson similarity might not perform well, hence we use the Jaccard Index for calculating similarities in the UBCF and IBCF algorithms [1]. The evaluation metrics are calculated and compared for all the algorithms with 2 different input matrices.

4 Results and discussion

System	Accuracy@10	Precision	Recall	Avg Time for each CV fold
Popular	0.82	0.34	0.031	1.5s
UBCF	3.91	0.39	0.038	7s
IBCF	3.32	0.32	0.031	165s

Table 1: Comparison of recommender systems with input as a binaryRatingMatrix

Table 1 shows the results of using a binaryRatingMatrix. We can see that the UBCF system performs better than the baseline and all other algorithms in terms of accuracy@10, precision and recall. It is also computationally competent.

System	Accuracy@10	Precision	Recall	Avg Time for each CV fold
Popular	0	0.24	0.026	5.5s
UBCF	4.50	0.22	0.024	30s
IBCF	0.03	0.004	1.6×10^{-4}	275s
SVD	3.48	0.23	0.026	3s

Table 2: Comparison of recommender systems with input as a realRatingMatrix

Table 2 shows the results of using a realRatingMatrix, where the Popular system has 0 accuracy@10, we can tell that with such a formulation of ratings, the most popular products were not the preferable products to our domain users. In conclusion UBCF has higher accuracy@10 than all other RS.

Regarding the accuracy@10, in general,all the RS implemented above doesn't seem to have a good performance, but considering the fact that our data set is large with high number of products and the users have bought only a handful of products among them, we can compare the results only with respect to a baseline.

Considering the trade-off between computational effort and accuracy@10, we can conclude that the UBCF RS with $binaryRatingMatrix$ as input is better than the baseline and any other RS that has been implemented over here.

References

[1] Michael Hahsler. “recommenderlab: A Framework for Developing and Testing Recommendation Algorithms”. In: ().