Collaborative Filtering Algorithms Implementation and Top K Recommendation Optimization

COMP9417, Assignment 2 Report
Peng Yi [redacted] Chengyu Jia [redacted]
, Ahmed Hannan [redacted]

Abstract

These report highlights the various kinds of recommendation algorithms known as “Collaborative Filtering” algorithms which have been divided into three parts: User based CF, Item Based CF and Model based CF. We have implemented all of these three algorithms. Based on our analysis, we proposed optimization algorithm based on KNN method to improve the recommendation precision. The experiments on the MovieLens 100K dataset [extracted from here https://grouplens.org/datasets/movielens/100k/] have shown that our proposed method significantly improve the Top K recommendation result.

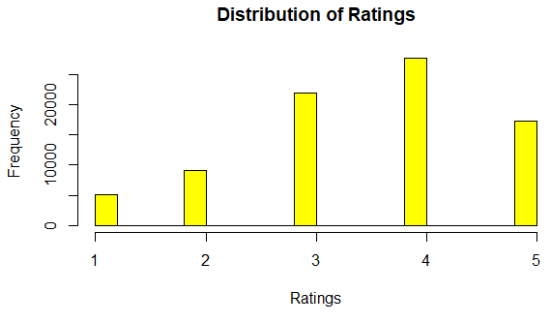
Additional files, such as codes, train dataset, test dataset can be found here https://drive.google.com/open?id=1kaA3maWUMuwAcZTrMbTpVw-Ap-QPzkoY

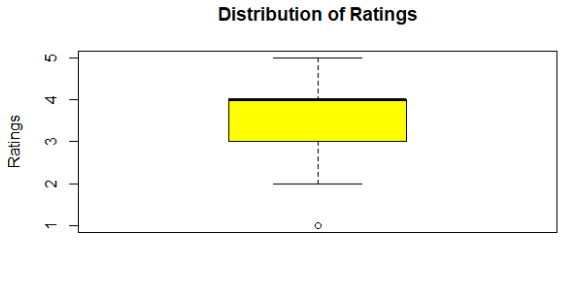
1 Introduction

Recommended systems or recommender systems are information filtering systems, which suggest products based on the user preferences. They process historical datasets using machine learning algorithms with the goal that these products will be purchased by the user in the near future. In the recent years, recommended systems have become increasingly popular, and are extensively utilised by a multitude of retailers/businesses to boost their revenues. Common examples of utilisation of recommended systems include recommended movies displayed on Netflix or Youtube based on the user’s search history, or suggested clothing ads on Facebook or instagram based on the user’s online shopping history. All in all, the widespread use of recommender systems have enabled various retailers and businesses to increase their sales by gaining a better understanding of what the customer may prefer to purchase. This is not only beneficial for businesses, but also customers who can access personalised products more easily and save time.

2. Data Exploration

We are provided with a 100,000 in our raw dataset discussing about 943 users and 1682 movies. Each user has given ratings to some of the movies between 1 to 5. At first, we plot a histogram and boxplot to see the overall trend of out our dataset:

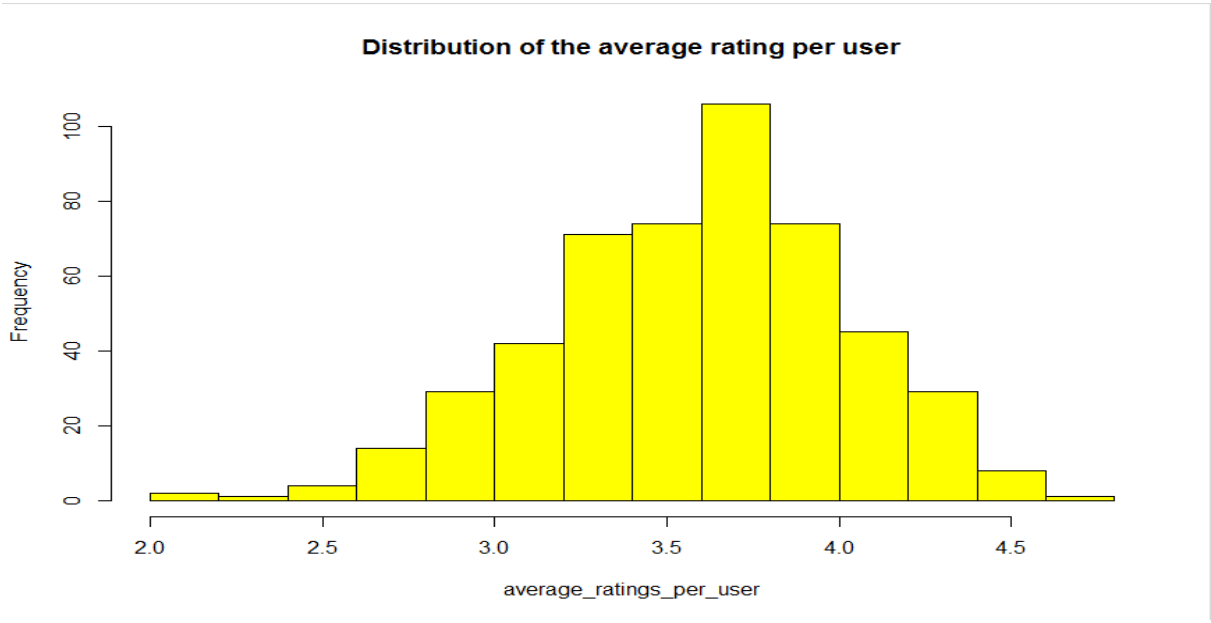




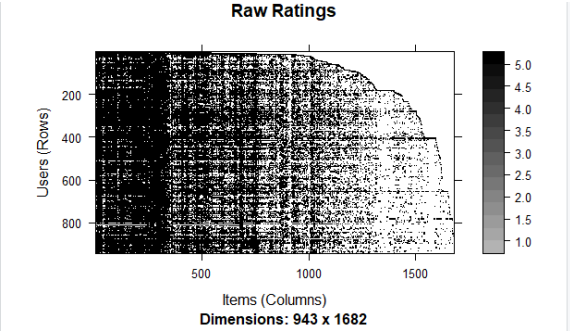
A greater userstanding of the descriptive statistics is as below:

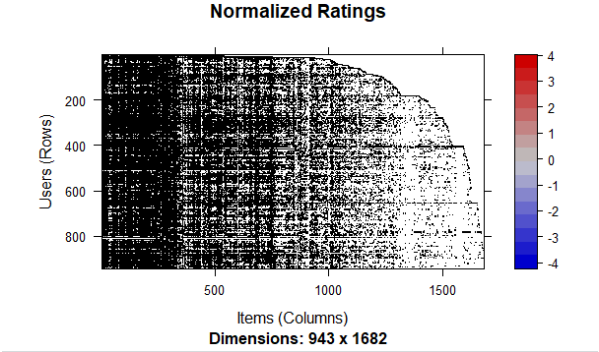
```
> summary(as.vector(as.matrix(jester)))
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
    1.0     3.0     4.0     3.5     4.0     5.0 1187781
> |
```

We have around 1187781 having NULL values of some users rating a particular movie. As from above we can see that the mean is lower than the median, confirming that the data is positively skewed. We therefore plot the average rating per user removing the missing values and the data is approximately normal though not zero centred:



A raw matrix and a normalise matrix diagram of 943 × 1682 is as follows. A column represents one specific movie and ratings by users, which are shaded region. Note that some items are always rated ‘black’ by many users. On the other hands a few users always give high ratings as in some cases a series of black dots cut across items:





3. Collaborative Filtering Algorithms

There are three algorithms we discuss in this report which are User Based Collaborative filtering (UBCF), Item Based Collaborative Filtering (IBCF) and Model based CF(PMF). In this algorithm we make the assumption that users with a similar taste at past will have similar taste in future. Suppose a person who watches a certain genre of movie would always like to watch the similar genre of movie, or a person who rates a certain cuisine would probably rate the similar value to other cuisine. Both of these algorithms are widely used in practice.

In order to do analysis, we have randomly selected 80% data as have use that for training purposes, and call that train dataset. The rest 20% are used for testing purposes and are called test data.

3.1 User Based Collaborative Filtering

The core idea of User Based Collaborative Filtering(UBCF) is user would prefer the item which his friends prefered[1]. More, specifically, the UBCF can be divided into two step: The first step is finding the same-preferences friends for target user by his(her) historical ratings. The second step is utilizing the frends' rating history to make prediction for target user.

As for UBCF algorithm, Ahmed implemented it with R. During impletation we first calculates the similarity among user by looking at an indivicual user what he/she has rated to other items. It can be done via three different methods:

- Cosine Similarity
- Pearson Correlation Similarity
- Euclidean Distance

Three of these methods again uses three separate collaborative filter models for normalizing the data. One is just building the raw model without any normalisation, other is focusing the data's Centre and the other approach is via Z-score normilisation applied to the data.

After similarity calculation, the weighted is used to calculated the prediction for each user. A simple RMSE comparision of three similarity method are presented below:

Models	RMSE
UBCF_N_C	2.558167
UNCF_C_C	1.081696
UBCF_Z_Z	1.081621

From the above table we see the RMSE decreases as the data is more normalised. Our Z-Score normalisation gives an lowest RMSE of 1.081621.

3.2 Item Based Collaborative Filtering

Compared to User-based Collaborative Filtering, Item-based Collaborative Filtering is a more stable approach as movie ratings do not depend on people's varying tastes or moods[2]. Hence, a superhit movie would generally earn a good rating while a flop movie would earn a bad rating.

The Item-based Collaborating Filtering is also based on the ratings given by the target users. But the similarity is calculated between items instead of users.After the similarity is calculated, a weighted average can be achieved by similarity between target movie and the movied user already rated . The UBCF also can be divided into two steps: Similarity Computation and Prediction.

(1) **Similarity Computation.** For the Similarity Computation, we used the Pearson Correlation Similarity which can be formulated as follows:

$$sim(i,j) = \frac{\Sigma_{u \in U} \left(R_{u,i} - \bar{R}_i\right)\left(R_{u,j} - \bar{R}_j\right)}{\sqrt{\Sigma_{u \in U} \left(R_{u,i} - \bar{R}_i\right)^2} \sqrt{\Sigma_{u \in U} \left(R_{u,j} - \bar{R}_j\right)^2}}$$

The Pearson Correlation Similarity uses a particular user u rated i-th movie's rating score to minus the average of this i-th movie's rating score, same as the j-th movie. Hence, this means that when we generate the similarity of the i-th and j-th movies, the common users who rated both i and j movies should be selected. Then we acquire a similarity form, which can be used to predict the future.

(2) **Prediction.** For the prediction part, we used the previously calculated similarity form to compute the average weight for the movies which the target users didn't rate. Then, we selected the top 100 movies which have the highest average weights.

$$P(u,i) = \frac{\Sigma_{all\ similar\ item,\ N} \left(s_{i,N} * R_{u,N}\right)}{\Sigma_{all\ similar\ item,\ N} \left(|s_{i,N}|\right)}$$

As for IBCF algorihm, Jack implemented it in python. First, we randomly split the data into train data and test data with a ratio of 8:2. Then, we use this train set to do the similarity computation as well as prediction. And the RMSE between the predicion and the test data is 0.8075.

3.3 Model based CF (Matrix Factorization)

Compared to the UBCF and IBCF, the model-based CF algorithm is much different. Instead of assuming, the user preference can be explained by the similarity between users and items. The model-based methods are trying to find the latent representation of users and items by matrix factorization method[3]. More specifically, if a user latent representation can be written as $\mathrm{Ui} \in \mathbb{R}^{\mathrm{N}}$ and an item latent representation can be written as $\mathrm{Vj} \in \mathbb{R}^{\mathrm{N}}$, then the rating that this user gives this movie can be formulated as the inner product of Ui and Vj, as $\mathrm{Rij} = Ui' \times Vj$.

In model-based CF algorithm, the process can be also divided into two steps: Firstly, utilizing the user rating matrix to learn the latent representations for users and items. Secondly, using the latent representations to calculate the prediction.

As for Model-based CF algorithm, we implemented the state of art algorithm – Probabilistic Matrix Factorization(PMF) in python. Considering the situation that it is computational expensive to

calculate the closed form solution, during the implementation we utilize the stochastic gradient descent(SGD) to calculate the latent representations for users and items. The loss function we use in the implementation is presented below:

$$Loss = \sum_i \sum_j (R_{ij} - U_i^T V_j)^2 + \alpha \sum_i \|U_i\|^2 + \beta \sum_j \|V_j\|^2$$

The train test data split is the same as in IBCF, and the final RMSE is 0.9832 on the test data.

4 Optimization

4.1 Recommendation Result Analysis

(1) **RMSE**: After discussing from above, we'll analyses all the three algorithms and it shows IBCF and PMF is much better compared to UBCF as the RMSE for both IBCF and PMF is lower than UBCF.

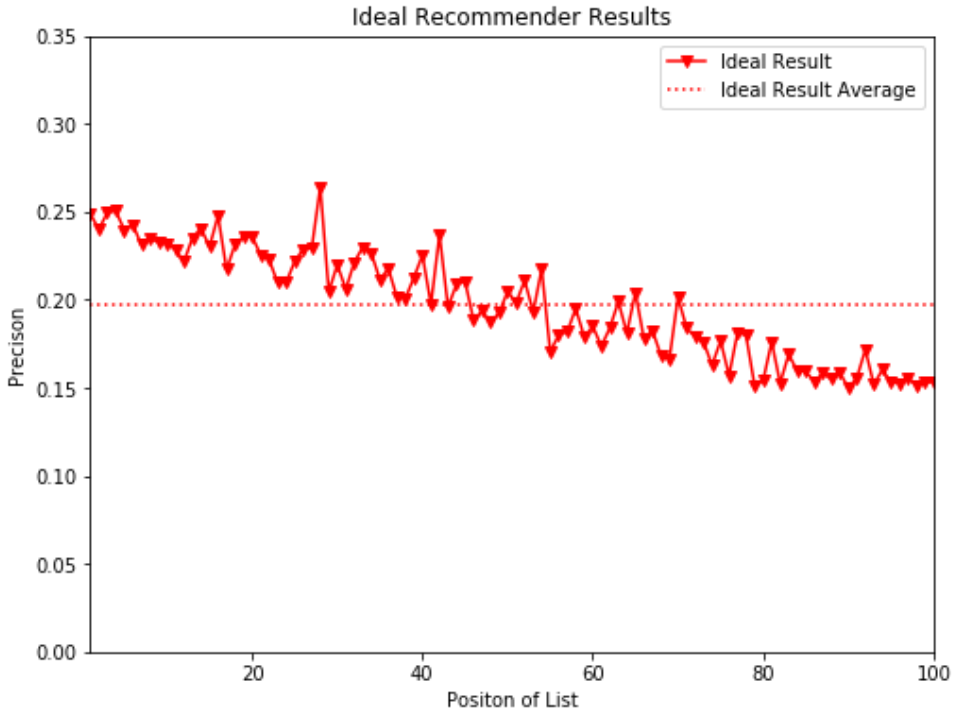
Recommender System Category	RMSE
User Base Collaborative Filtering	1.0816
Item Base Collaborative Filtering	0.8075
Model based CF (PMF)	0.9832

From the IBCF's RMSE, it shows the Item Recommender System can always make relatively accurate prediction. And the benefit of the high accurate predict is from Item Base just according to the users historical rating to do the prediction and would not impacted by the user's current or future preferences. And the RMSE shows IBCF has relative stability rate, so the applicable period is relatively long than UBCF. So it can help company to save lots of money, without update it in a short period.

(2)**Precision**: In real world recommender system, users want to see their preferred movies in the top of the list rather than the bottom. In other words, if the length of the prediction list is quite high (e.g.100), we want to put the item of higher preference on the top of the list for faster user response. Based on the analysis above, in this section, instead of calculate the average precision of the whole recommendation list, we choose to calculate the precision of each position in the recommendation list which can help us to anaylsis the prefermance of the recommendation list. More specifically, If we use i to indicate the i-th position of the list, precision on i-th position can be formulated as:

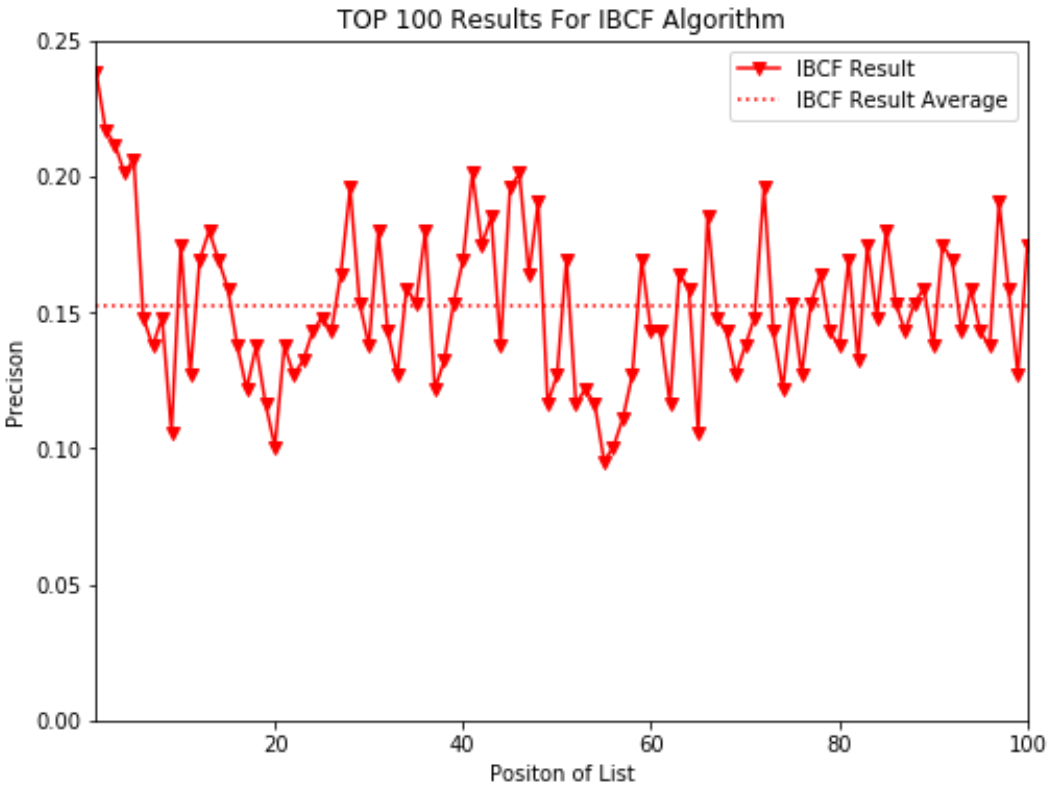
$$p(i) = \frac{number\ of\ users\ who\ get\ right\ prediction\ in\ the\ ith\ position}{number\ of\ all\ test\ users}$$

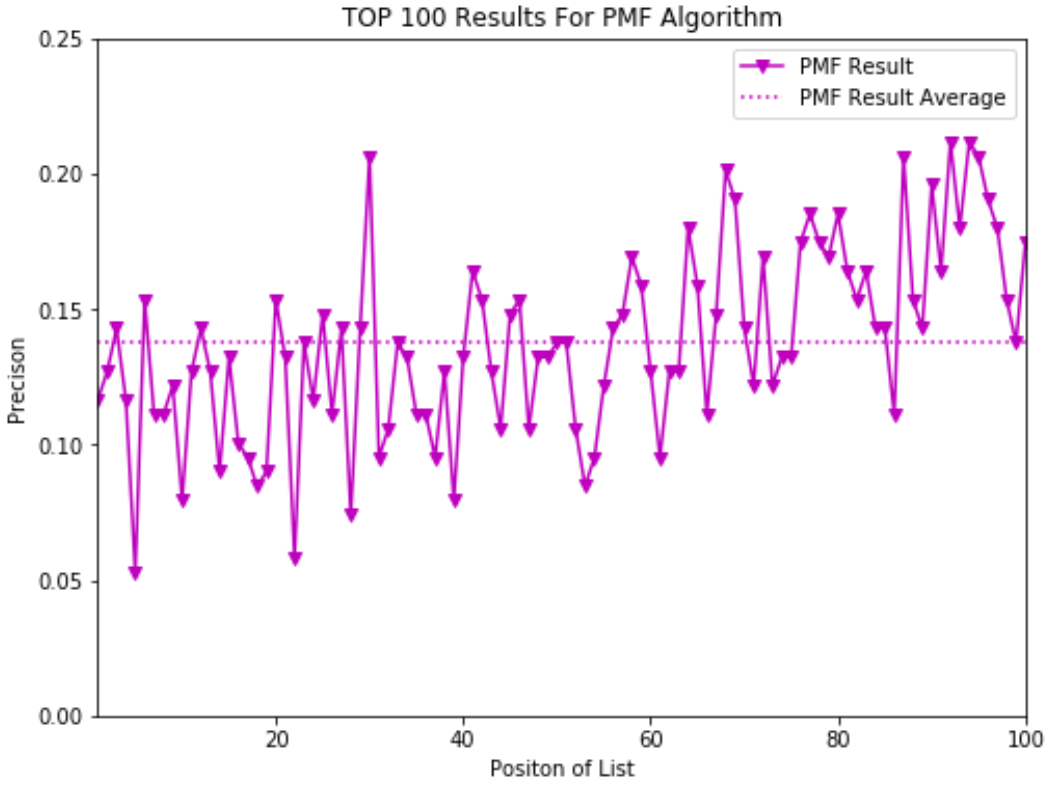
In an ideal recommendation algorithm, we want to put the item with higher possibility in the top of the list, which also means we want the results similar to the following image:



As shown in the picture, the precision of the front of the list (where i<=30) is higher than the precision of the bottom of the list (where i>=70).

But the result of IBCF and PMF algorithm in our paper is as follows:





As shown in the pictures, the IBCF and Probabilistic factorization Matrix (PFM) algorithm doesn't give higher position precision in the top of list. Furthermore, the PFM algorithm gives higher prediction in the bottom of the list. Thus, we come up with an optimization method to use the re-ranking to re-arrange the recommendation list and yield a more accurate result for the top K recommendation.

4.2 Optimization Method

4.2.1 Methodology

The main idea for optimization is to re-rank the items in the original list. More specifically, we try to find out the items which user would more preferred in the original, and reranking them into the top of the recommendation list. Thus, a two-step optimization technique was used to find out the items with higher possibility and re-ranking the list.

(1) **How to find the more preferred items?** We generalize the KNN method in our situation. The main idea of KNN classification algorithm is to find the K nearest neighbour for target item and using the neighbours' classification to do the prediction for the target user.

In this paper, we use similar idea of KNN. Take a single User-x as an example. We assign the items that User-x has rated as positive class, the rest of other items as negative class. The original recommendation result for User-x can be written as *[Rx1, Rx2, Rx3…. Rx100].* For each *Rxi* in the list, find the K-nearest neighbour *of Rxi*. If one of those K nearest neighbours is in the positive class, we assign *Rxi* in positive class "+1" (with the higher possibility that user will like, so we should put it in the top of the recommendation list). Otherwise, we assign R_{xi} in negative class "-1" (with the lower possibility that user will like it, so we should put it in the bottom of the recommendation list). The process can also be explained as follows:

User-X: POSITIVE Class { Item-A-NN1, Item-E-NN3, Item-F, Item-H}
NEGATIVE Class { Items-M, Item-N, Item-P, ….. }
Original list: { Item-A, Item-B, Item-C, Item-D, Item-E}
Find KNN for each item in original list, classify the item:
Item-A: {Item-A-NN1, Item-A-NN2, Item-A-NN3} Classify as POSITIVE Class +1
Item-B: {Item-B-NN1, Item-B-NN2, Item-B-NN3} Classify as NEGATIVE Class -1
Item-C: {Item-C-NN1, Item-C-NN2, Item-C-NN3} Classify as NEGATIVE Class -1
Item-D: {Item-D-NN1, Item-D-NN2, Item-D-NN3} Classify as NEGATIVE Class -1
Item-E: {Item-E-NN1, Item-E-NN2, Item-E-NN3} Classify as POSITIVE Class +1

However, the important constraint in this approach is the calculation of similarity between the two items. Thus, the similarity calculation allows to find the K nearest neighbour of the items. Instead of utilizing the user-movie rating, we use the genre tags for each movie provided by Movielens dataset. In the dataset, there are totally 18 tags for movies, and each movie may have 1 to 8 different tags. Then, for each movie, its tag feature can be represented as an 18 length binary list. 1 in i-th position of the tag list means this movie has i-th tags, 0 otherwise. Then, it will be easy to calculate the ***Jaccard*** Similarity between two movies by its binary tag feature list.

Overall, we use tags information to calculate the K-nearest-neighbour for each predicted item and using the calculated K-nearest-neighbour to decide whether it is the item that user will have higher preference.

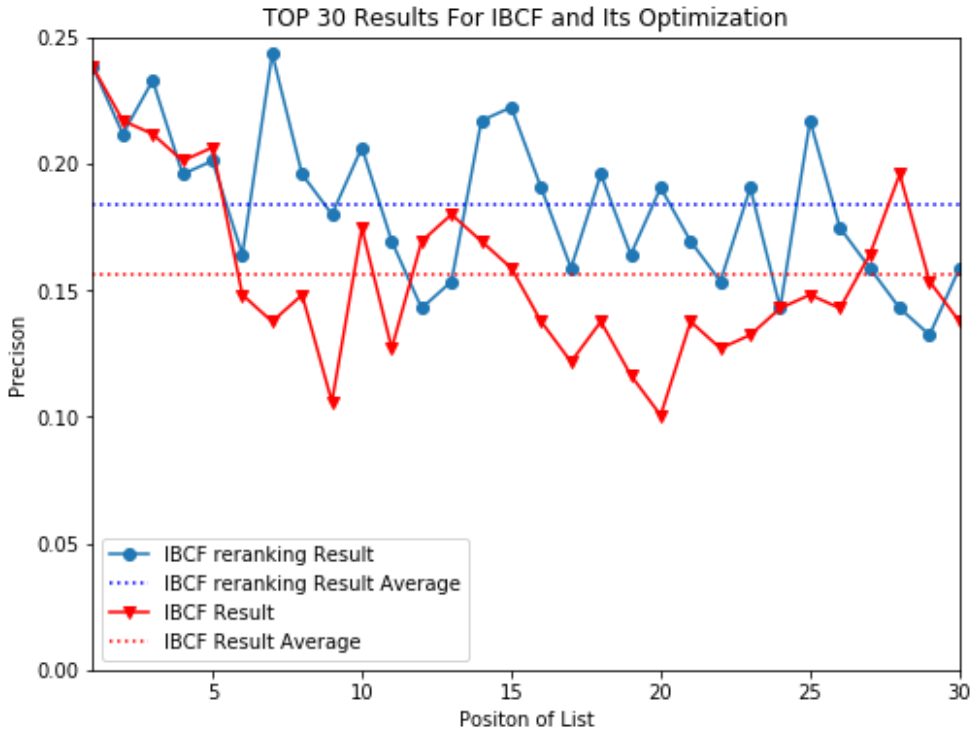
(2) **How to do re-rank?** The core idea is to putting the positive item (we get from the above step) at the front of the final list. The detailed method can be presented as follow: Given a User-x's original recommendation result *[Rx1, Rx2, Rx3…. Rx100]. For* each item in the list assigned a ranking weight that is equal to its position eg. $W\text{-}R_{x1}=1$, $W\text{-}R_{x2}=2$, $W\text{-}R_{xi}=i$. It is obvious that the lower $W\text{-}R_{xi}$ means the higher importance of the R_{xi}. As in last step we already assign the items in the original list into positive class and negative class, in this step, we update $W\text{-}R_{xi}$ for items in

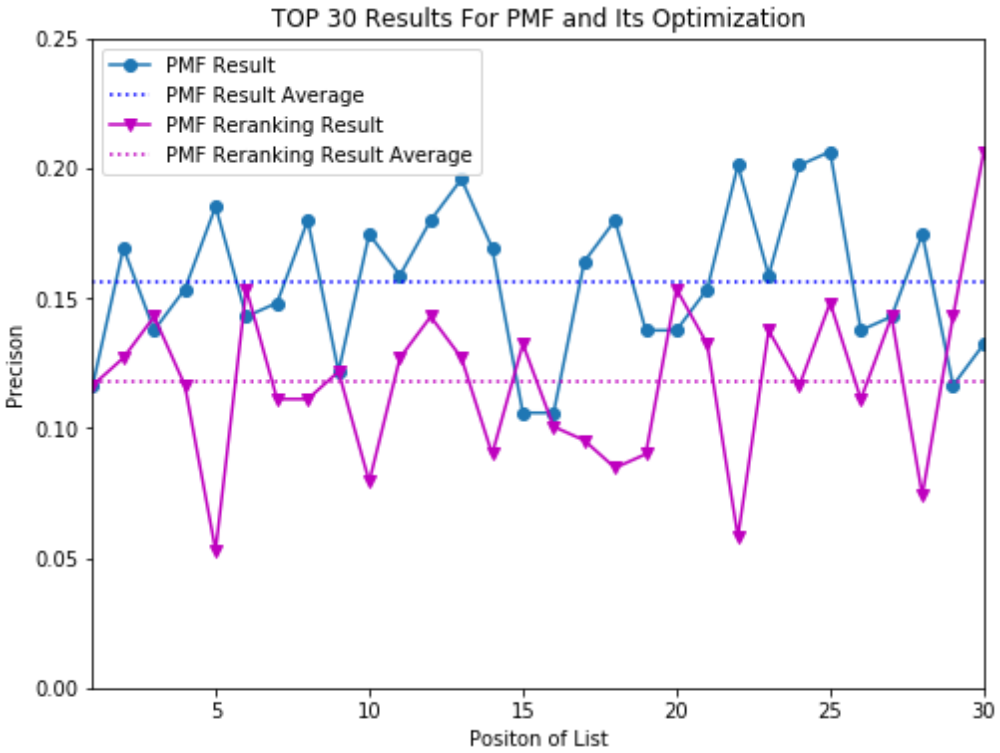
positive class by multiplying with an important weight Wi (0<Wi<1). Then, the final list will be the re-ranked list by the updated W-R_{xi} of the items. The process can also be explained as follows：

Original list: *{Item-A, Item-B, Item-C, Item-D, Item-E}*
KNN result: *{ 1 , -1 , -1 , -1 , 1 }*
Original W-Rxi: *{ 1 , 2 , 3 , 4 , 5 }*
Updated W-Rxi: *{ 0.5 , 2 , 3 , 4 , 2.5 }, Where Set Wi=0.5*
Reranking list: *{Item-A, Item-B, Item-E, Item-C, Item-D}*

2) The Optimization result:

To test the proposed method, we compared the Top 30 result of the original list and the reranking list. The evaluation method is still the precision of each position in the list. The original recommendation lists are generated by Item Based CF and PMF. The results are shown in the following image.





As shown in the above pictures, the average precision of Top 30 recommendation is significantly improved by the reranking method for both IBCF and PMF. As for IBCF, although the original recommendation results (the red line in the first image) get really good performance in position of 1 to 5, the precision on position 6 to 10 is not acceptable. After our modification, the reranking list (the blue line in the first image) maintain the high precision on the top 5 position, but we also largely improve the precision on position 6 to 10. It is also being proved in the PMF algorithm with the top position 1 to 15. As the precision of the top positions in the list are significantly improved, the effectiveness of our proposed method has been proved.

5 Summarize

In this paper, we focus on the Collaborative filtering recommendation algorithm. We implement three CF algorithms including User-based Collaborative Filtering, Item-based Collaborative Filtering and Model based Collaborative filtering(PMF). By analysis the RMSE for three algorithm, it is shown that the IBCF and PMF has shown better result on Movielens100K dataset. Moreover, by analysis the precision of each position of the list, we propose a novel KNN-based reranking algorithm to improve the Top K recommendation. The experiments on IBCF and PMF also prove the effectiveness of our proposed method.

6 References

[1] Herlocker J L, Konstan J A, Riedl J. Explaining collaborative filtering recommendations[C] // Proceedings of the 2000 ACM conference on Computer supported cooperative work. ACM, 2000: 241-250.
[2] Sarwar B, Karypis G, Konstan J, et al. Item-based collaborative filtering recommendation algorithms[C]//Proceedings of the 10th international conference on World Wide Web. ACM, 2001: 285-295.
[3] Mnih A, Salakhutdinov R R. Probabilistic matrix factorization[C]//Advances in neural information processing systems. 2008: 1257-1264.