# Improvements to Collaborative Filtering Algorithm
# Project Report

Yingnan Han yhan@wpi.edu
Chenjie Jiang cjiang3@wpi.edu
Chen Liang cliang@wpi.edu

November 30th, 2017

**Abstract**

This project is mainly focus on improving the performance of the collaborative filtering algorithm.We used normalization method to increase the average similarity of neighbors founded by the algorithm, this method indeed lower the RMSE of CF. Also we try to use a weighted model to combine the SVD with CF, unfortunately, the improvement is not significant.
**Keywords:** Collaborative Filtering, Normalization, SVD

# Chapter 1

# Introduction

Recommendation system has been widely used in different kinds of websites,, in order to provide more accurate products or services to them. Collaborative Filtering algorithm (CF) [2] is considered to be the most common algorithm to construct the system. However, constrained by the sparsity of dataset and consider different real-world problems, CF algorithms still suffers from some shortcomings that may induce to giving a wrong recommendation.

In some cases,users or movies that may have exact same rating patterns can not be judged as neighbors in the CF algorithm. Thus, the low similarity of neighbors will have a bad effect on the performance of the recommendation, [3] [1]raised a method to help the algorithm find a better neighbors, thus contribute to the reduce of RMSE, we also tried some other statistical methods for the same goals.

# Chapter 2

# Methods

## 2.1 Dataset Introduction

In this project, we are using only the rating data file in the Movielens-latest dataset to finish all our work. From this dataset, we can get the userID, movieID, the rate and timestamp for each rating records. Since there is a small number of variables for us to analyze, we also tried to obtain some statistical measurements for measuring, including the average rating, the variance of rating and the count number of rating, correspond to each movie or user. All the variables we try to analyze is in the following table. Table 2.1 shows all the measurements we use for analyze.

| Variable | Type | Source of var | range |
|---|---|---|---|
| userID | numerical | dataset | [1,671] |
| movieID | numerical | dataset | [1,9066] |
| rating | numerical | dataset | [0.5,5] |
| timestamp | numerical | dataset | – |
| average of rating | numerical | program | [0,5] |
| variance of rating | numerical | program | – |
| number of rating | numerical | program | – |
| prediction error | numerical | program | [0,5] |
| average similarity | numerical | program | [0,1] |

Table 2.1: variables and measurements

## 2.2 Collaborative Filtering

Recommendation system that applies CF algorithm has two types, the item-based CF and the user-based CF, user-based CF uses the information of neighbor user to make predictions, item-based make infers with the help of the information from neighbor movies.

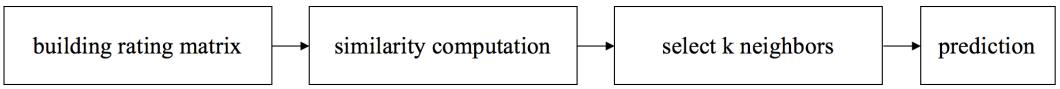



Figure 2.1: process of CF algorithm

Figure 2.1 shows the process of the CF algorithm. In general, CF algorithm first build a train rating matrix based on the training set we have, The rating matrix should be sparse since users are not likely have watched and rated all the movie they have seen.

Then, we calculated the similarities between users or movies, depending on the type of CF, and build up a similarity matrix. In this project, we use the following equation, which is the cosine similarity as the similarity measurement.

$$sim_{cos} = \frac{\sum_{p \in I_{ij}} r_{ip} r_{jp}}{\sqrt{(\sum_{p \in I_{ij}} r_{ip}^2)(\sum_{p \in I_{ij}} r_{jp}^2)}} \tag{2.1}$$

K-neighbor movies or users with the highest similarity will be selected for predicting ratings in the test set. Once we have the test movie ID and test user ID, we may have their k-nearest neighbors based on the similarity matrix, all the neighbor user must have rated the specific test movie, and vice versa.

$$\hat{r}_{ip} = \frac{\sum_{j=1}^{k} s_{ii_j} r_{i_j}}{\sum_{j=1}^{k} |s_{ii_j}|} \tag{2.2}$$

After retrieved the k-neighbor user or movies, we use the ratings from neighbors to calculate the rating prediction in the test set. For measuring if the CF algorithms works well in the test set, we calculate the RMSE for the test set with equation 3, all our improvements are suppose to reduce the RMSE of the CF algorithm.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\hat{r}_i - r_i)^2}{n}} \tag{2.3}$$

## 2.3 Improvements

### 2.3.1 Normalization Model

Since we want to improve the method of calculating similarities, we analyze the users behavioral features. We consider the users rating preference to be one of the reasons making similarities calculating not accurate. Because of different rating preference, the average ratings of each user and movie cant be regarded as the same. In order to extract the influence of rating preference, we normalize the rating matrix by removing the effect of users and movies. So in the first step, we remove the average of each users

$$\begin{aligned} norm(r_{ip})_1 &= r_{ip} - \frac{1}{M} \sum_{k \in I_i} r_{ik} \\ &= r_{ip} - \bar{r_i} \end{aligned} \tag{2.4}$$

Then based on it, we remove the average rating of each movie.

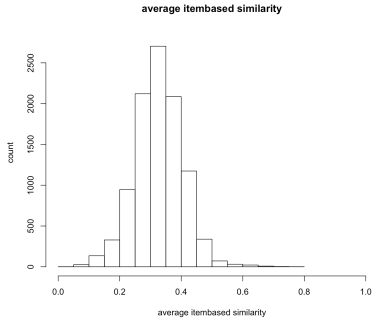
$$\begin{aligned} norm(r_{ip})_2 &= norm(r_{ip})_1 - \frac{1}{N} \sum_{k \in I_p} norm(r_{kp})_1 \\ &= r_{ip} - \bar{r_i} - \frac{1}{N} \sum_{k \in I_p} (r_{kp} - \bar{r_k}) \end{aligned} \tag{2.5}$$

Finally, we can get the normalized rating shown as follows:

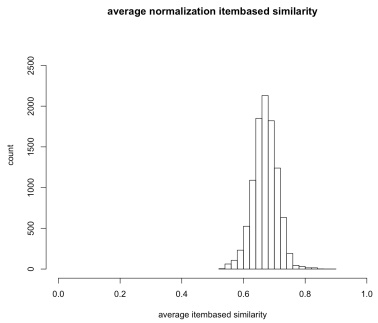
$$norm(r_{ip}) = r_{ip} - \bar{r_i} - \bar{r_p} + \bar{r} \tag{2.6}$$

Then we can use normalized rating matrix to calculate the brand new similarities.And also use KNN to select k nearest neighbors for prediction.

When we check the value of average similarities of each item or user, we can figure out the similarities increase overall:

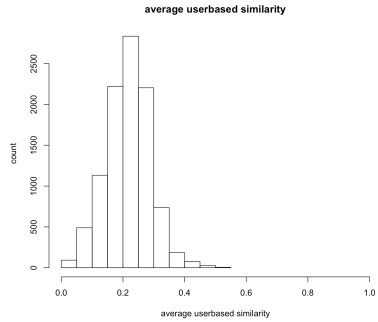



(a) before normalized

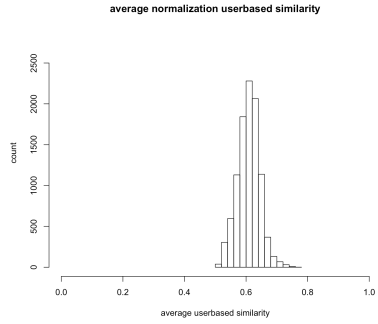



(b) after normalized

Figure 2.2: itembased




(a) before normalized




(b) after normalized

Figure 2.3: userbased

Since we change the value of the ratings, when we compute the predicted results, we have to put these factors back. The new equation for predicted rating should be:

$$\tilde{r}_{ip} = \frac{\sum_{j=1}^{k} s_{ii_j} r_{i_j}}{\sum_{j=1}^{k} |s_{ii_j}|} + \bar{r}_i + \bar{r}_p - \bar{r} \tag{2.7}$$

### 2.3.2 Weighted Normalization Model

For calculating the similarities accurately furthermore, we tend to give different weights to different movie or user. We believe that movie which has not much rating will reflect users preference better. That is to say, a commonly popular movie isnt a good standard to judge the similarities between users. And the same goes for the user. So the weight of each movie or user can be shown like this:

$$\omega_p = \frac{1}{\sqrt{(length(r_{,p})}} \tag{2.8}$$

So the new similarity calculating formula with weights on can be changed to be:

$$\hat{sim}_{cos} = \frac{\sum_{p \in Iij} \omega_p r_{ip} r_{jp}}{\sqrt{(\sum_{p \in I_{ij}} \omega_p r_{ip}^2)(\sum_{p \in Iij} \omega_p r_{jp}^2)}} \tag{2.9}$$

### 2.3.3 SVD and Weighted SVD Model

SVD is another method widely used in the recommendation system. SVD is to decompose the rating matrix into two matrices:

$$R_{m \times n} = P_{m \times p} * Q_{p \times n} \tag{2.10}$$

So we can use these two decomposed matrices to measure predicted rating:

$$\acute{r}_{ip} = P[i,] * Q[,p] \tag{2.11}$$

Since SVD is algorithm with high computational complexity. We dont explore which number of feature is the best to fit. We decompose the matrix with 100 features.However, using SVD directly to predict, we have RMSE of the test data to be 0.97.Based on SVD, we combine the CF and normalized CF with it to explore to reduce RMSE of the test data. We use the weight $\alpha$ to define the new predicted rating formula:

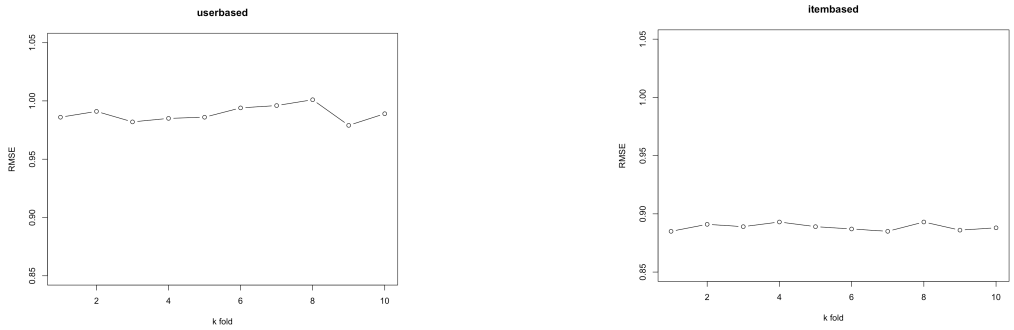
$$\check{r}_{ip} = \alpha \acute{r}_{ip} + (1 - \alpha)\hat{r}_{ip}, \alpha \in (0, 1) \tag{2.12}$$

And we can change the value of $\alpha$ to get a best result.

# Chapter 3

# Results

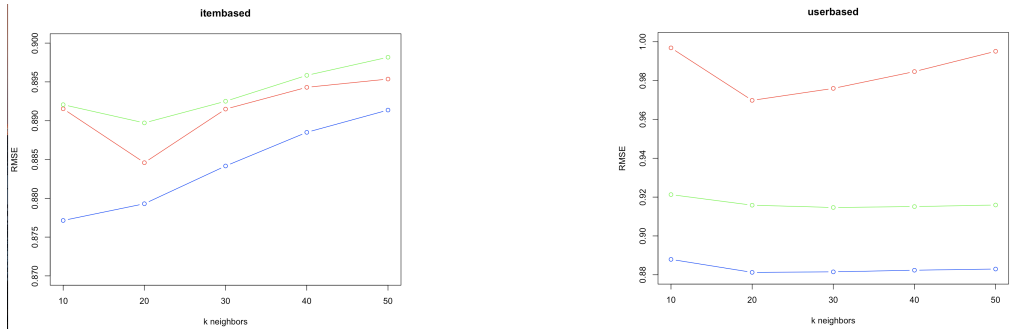
For the construction and verification of the CF algorithm, we applied 10-fold cross validation to measure the RMSE of it. Figure 3.1 shows the RMSE in different folds.




(a) userbased (b) itembased

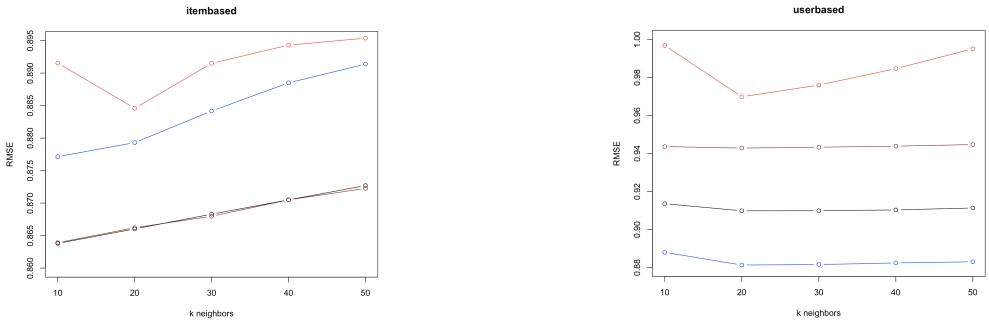
Figure 3.1: RMSE in different fold

For the improvements we have, we separated the data in 1:9, considering them as test set and train set, respectively. While not all of the improvements has a satisfactory result. Here are plots that show RMSE changes with the number of neighbors.




(a) itembased (b) userbased

Figure 3.2: CF,normalized CF, weighted normalized CF




(a) itembased (b) userbased

Figure 3.3: CF,normalized CF,SVD&CF,SVD&normalized CF

Red line denotes original CF algorithm; Blue line denotes normalized CF algorithm; Green line denotes weight normalized CF algorithm; Dark red line denotes SVD&CF algorithm; Black line denotes SVD&normalized CF algorithm.

In the figure3.2, we can figure out that weight normalized CF doesn't perform well. Since we don't have much time to make more experiments, we should consider more factors to adjust the weight for a better result.

In the figure 3.3, we can find that combination of SVD and CF work really well in the itembased while normalized CF algorithm work best in the userbased.

Besides, we used paired t-test method between improved algorithms and the original CF algorithm [4] to verify that our improvements all indeed make significant changes to lower the RMSE of the original CF algorithm.

| improvement | t | df | p-value |
|---|---|---|---|
| normalized CF | 4.0087 | 4 | 0.008007 |
| weight normalized CF | -2.6751 | 4 | 0.9722 |
| SVD&CF | 15.834 | 4 | $4.649 * 10^{-05}$ |
| SVD&normalized CF | 15.867 | 4 | $4.61 * 10^{-05}$ |

Table 3.1: itembased

| improvement | t | df | p-value |
|---|---|---|---|
| normalized CF | 23.091 | 4 | $1.042 * 10^{-05}$ |
| weight normalized CF | 14.736 | 4 | $6.172 * 10^{-05}$ |
| SVD&CF | 8.1382 | 4 | 0.0006202 |
| SVD&normalized CF | 15.711 | 4 | $4.794 * 10^{-05}$ |

Table 3.2: userbased

According to the t-test results, in the userbased method, all of these four improvements reduce the RMSE statistically significantly compared with the original CF algorithm. And in the itembased method, except the weight normalized CF, all those three improvements make RMSE deceased statistically significantly.

# Chapter 4

# Discussion and Conclusion

In this project, we've tried several methods, attempting to avoid the shortcoming in finding neighbors of CF algorithm, infer more information based on the similarity to make a better prediction.

Among all the improvements we have tried, the normalization method indeed increased the similarities and significantly lower the RMSE, as we expected. The quantity of rating for a user or movie may have impact on the similarity or prediction computation, since we believe popular movies or movies that few people see is an important indicator of the user's expectation or preferences, but the relationship between the preferences and the quantity is not the way what we thought to be. Thus we have to do some further work to explore the weighted model.

As for the SVD, the weighted model that combining initial CF algorithm and SVD also did a great work in decreasing the RMSE, which indicates that, to some extent, SVD makes a valid contribution to the prediction. However, the combination of SVD and the CF, no matter if normalization is used, performs not stable and worse than CF only using normalization. This situation make us thinking of if there is a conflict between SVD and normalization. SVD is also a process of feature extraction and the reconstruction of the rating matrix, rating patterns supposed to be an feature, The bad performance of SVD is probably because normalization removed these feature, inducing to a worse reconstruction of rating matrix. For further research, we have to find a better way to combine these two techniques.

# References

## Literature


[1] Robert M Bell and Yehuda Koren. "Improved neighborhood-based collaborative filtering". In: *KDD cup and workshop at the 13th ACM SIGKDD international conference on knowledge discovery and data mining*. sn. 2007, pp. 7–14 (cit. on p. 1).

[2] *Implementing your own recommender systems in Python*. `https://cambridgespark.com/content/tutorials/implementing-your-own-recommender-systems-in-Python/index.html` (cit. on p. 1).

[3] Jun Wang, Arjen P De Vries, and Marcel JT Reinders. "Unifying user-based and item-based collaborative filtering approaches by similarity fusion". In: *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2006, pp. 501–508 (cit. on p. 1).

[4] Tianqing Zhu et al. "Privacy preserving collaborative filtering for KNN attack resisting". In: *Social network analysis and mining* 4.1 (2014), p. 196 (cit. on p. 7).