Different Collaborative Filtering Algorithms in Recommendation System

Liyi Cao Yimeng Wang Yu Zhao
Boston University ECE Department
Boston, MA, United States
caoliyi@bu.edu wym613@bu.edu yuzhao95@bu.edu

Abstract

The recommendation system is an information filtering system for predicting a user's "score" or "preference" for an item. There are two recommended algorithms that are commonly used today: content-based recommendations, and collaborative filtering. Our project will use both memory-based and model-based algorithms in collaborative filtering algorithms. The model will be based on the user's collaborative filtering, and will evaluate the recommendations by calculating the correlation similarity to achieve the goal of predicting user ratings of book.

1. Introduction

With the continuous development of the Internet, the recommendation system came into being. The recommendation system's work-flow is to analyze the User's preferences through thehistorical behavior of a single user, then search the user-wide behavior of the entire platform and sort the contentaccording to the similarity of the user's preferences. There-fore, the recommendation system is to link the behavior of individual users and all- Platform users, and to find the fa-vorite content of people with the same interests. This is the most different place between the recommendation system and other machine learning, and it is also the meaning of "collaborative" in collaborative filtering. The recommended system is used in a wide range of fields, including e-commerce, social friends, search engines, information content, etc., especially in the advertising field. Therefore, research recommendation system is very necessary. By analyzing the similarity between the user and other users, this project uses a collaborative filtering algorithm to predict the user's rating of the book, so that the system can recommend books to the user.

2. Literature review

In "Empirical analysis of predictive algorithms for collaborative filtering" [1], J. Breese *et al*. explained and examined two different types of collaborative filtering: memory-based collaborative filtering and model-based collaborative filtering. For memory-based collaborative filtering, they mainly examined the Pearson correlation coefficient algorithm and the vector similarity algorithm. For model-based collaborative filtering, they mainly examined the cluster model and the Bayesian network model. In the process, they came up with the method of default voting and the method of inversing user frequency to improve performance. After the comparison, their result indicates that for a wide range of conditions, Bayesian networks with decision trees at each node and correlation methods outperform Bayesian clustering and vector similarity methods [1]. Between correlation and Bayesian networks, the preferred method depends on the nature of the dataset, nature of the application(ranked or one-by-one presentation), and the availability of votes with which to make predictions [1].

In "Item-based collaborative filtering recommendation algorithms" [5], B. Sarwar *et al*. extended the topic of item-based collaborative filtering, which is another type of memory-based collaborative filtering. They used more algorithms to compute the similarity, such as pure cosine-based similarity and the adjusted cosine similarity. in their conclusion, they showed that item-based techniques hold the promise of allowing CF-based algorithms to scale to large data sets and at the same time produce high-quality recommendations [5], which has a better performance than traditional user-based techniques.

In "A Survey of Collaborative Filtering Techniques" [8], X.Su *et al*. made a summary of most popular methods of collaborative filtering techniques. In their paper, they also referred and summarized the aboved-mentioned paper by J. Breese *et al*. and B. Sarwar *et al*. Besides the supplement algorithms of memory-based CF and model-based CF, they also explained the hybrid collaborative filtering techniques, which combine CF with other recommendation techniques (typically with content-based systems) [8].

In "The YouTube video recommendation system" [3], J. Davidson *et al*. discussed a practical example of the video recommendation system in use at YouTube. In their process, they used both of the content data and the user activity data. Therefore, they implemented a hybrid collaborative filtering technique.

Besides, we also find a useful book "The Adaptive Web" written by P. Brusilovsky *et al.* In chapter 9, 10, 11 and 12, it detailedly explained the "Collaborative Filtering Recommender Systems" [6]. It would be a useful reference in our work.

3. Datasets and Evaluating Metric

3.1. Datasets

Consider a recommendation system consisting of M users and N items. The task of collaborative filtering is to predict the preference of one user based on the opinions of a set of similar users [1]. Each user u_j has given opinions on a set of books and its opinion on book n is given an numeric rating x_{jn}. Note that x_{jn} could be empty. To predict the preference of the user, we need to estimate its rating on item n .Let A be a user-books matrix, where the value of $i-th$ row and $j-th$ column is x_{ij} , just like:

$$A = \begin{Bmatrix} x_{11} & \cdots & ? & x_{1n} \\ ? & x_{22} & \cdots & x_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ x_{m1} & ? & \cdots & x_{mn} \end{Bmatrix}$$

3.1.1 Original Datasets

First of all, we chose to use the book recommender data [7]. We got a huge data set which contains lots of information. There are $10k$ books and 53424 active users in total. For each book, there are around 100 rating records and the standard deviation is 5.7593, which means that the number of rating records for each book doesn't vary much. And for each active user, the mean number of rating records is 18.3340 and the standard deviation is 26.2246, which varies a lot. The minimum number of rating record for a active user is 1 while the maximum number is 200. Overall, there are 981756 rating records, which range form 1 to 5. The mean rating for all the records is 3.8565. The ratings' distribution is as the following table shows: From the

Table 1. Book Rating Distribution

Rating	1	2	3	4	5
Number	19575	63231	248623	357366	292961
Percent	1.99%	6.44%	25.32%	36.40%	29.84%

table we could see that rating records with 4 has the maximum number. Half of the rating records are over 3. Only 8.43% records are with rating 1 or 2. Besides, our data is really sparse, which has a sparsity of 99.82%. We made a sparse matrix using *user ID*, *book ID* and the *rating records*. Part of the data distribution, whose user ID ranges from 1 to 10000, is as the following figure shows:

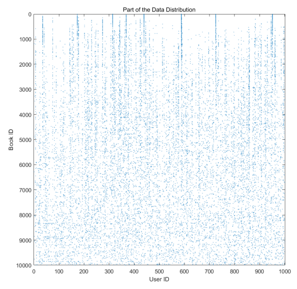


Figure 1. Part of the Data Distribution

3.1.2 Extracted Small Datasets

Because the sparseness of the original matrix is too high, we extracted a matrix of size 48*35, which has a sparsity of 31.07%. This small data set has 1153 rating numbers, which are distributed as shown in the table below, and which is almost identical to the original rating distribution. At the same time, we have increased its sparsity by 35% to 80% each time by 5%. So we have a total of 10 small data sets with different sparsity levels. Figure 2 shows the small data sets with different sparsity.

Table 2. Book Rating Distribution

Rating	1	2	3	4	5
Number	25	72	248	417	396
Percent	2.16%	6.22%	21.42%	36.01%	34.20%

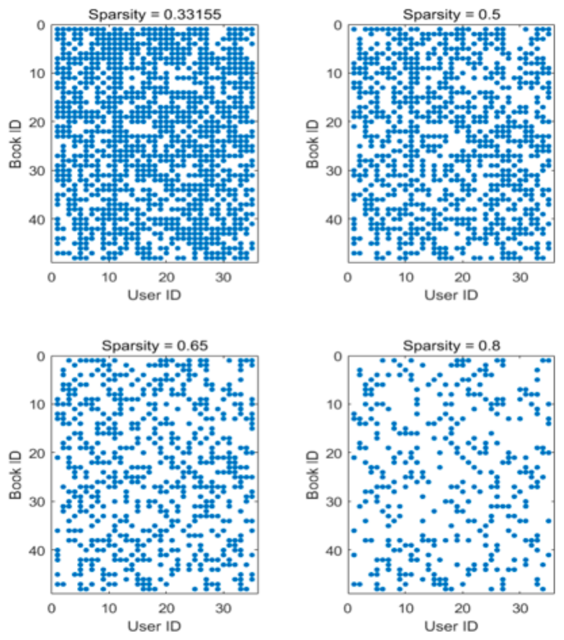


Figure 2. Small data set with different sparsity

3.2. Model Evaluating

To analyze the performance of our collaborative filter, we firstly divide the data set into a training data set a test data set. For each user ID in the test data set, we will randomly choose one of its rating records and mark it as the one that we want to predict. Those data will be treated as the ground truth data. We will use the rest test data set and the training data set to predict those marked data, which gives a rating between 1 and 5.

3.2.1 MAE

The Mean Absolute Error:

$$MAE = \frac{\sum_N p_{ij} - r_{ij}}{N}$$

where *N* is the total number of our marked data, p_{ij} is our predicted rating of *user* i on *book* j, and r_{ij} is the true rating record [2].

3.2.2 RMSD/RMSE

The Root Mean Square Error:

$$RMSD = \sqrt{MAE} = \sqrt{\frac{\sum_N p_{ij} - r_{ij}}{N}}$$

4. Baseline

In this project, we firstly set a baseline which could help us comparing the models. The baseline model is that to make every predicted rating as 4, which is the average rating of the whole dataset.

5. Memory-based CF

5.1. Introduction of Memory-based CF

Memory-based collaborative filtering is the most common collaborative filtering. In memory-based collaborative filtering, it will use the entire dataset to make the final decision.

Memory-based collaborative filtering has 2 categories: user-based and item-based. In this project, we use the user-based collaborative filtering. The key idea behind this model is that similar users share the same interest on books.

Memory-based collaborative filtering assumes that for those users who have made similar decisions before will also made the similar decision in the future [3]. Therefore, the main problem in memory-based collaborative filtering is to calculate the similarity between users. There are various algorithms that can calculate the similarity, and one of the most popular algorithms is the correlation.

5.2. Similarity

- Pearson Correlation

In this project, we use Pearson correlation [1] coefficient to calculate the similarity between user i and user j . The correlation is:

$$w(i,j) = \frac{\sum_n (v_{i,n} - \overline{v}_n)(v_{j,n} - \overline{v}_n)}{\sqrt{\sum_n (v_{i,n} - \overline{v}_i)^2 \sum_j (v_{i,j} - \overline{v}_i)^2}}$$

where the summations over n are over the items for which both users i and j have recorded rates. To recommend, we would pick top N similar users and collect their high-rating books, which has not read by the user who is going to be recommended.

- Cosine Similarity

Another method we use is to calculate the cosine similarity between users.

$$cos(A,B) = \frac{A \cdot B}{\|A\| * \|B\|} = \frac{\sum_{i=1}^n A_i * B_i}{\sqrt{\sum_{i=1}^n A_i^2} * \sqrt{\sum_{i=1}^n B_i^2}}$$

Where A and B represent two users, respectively. A_i and B_i represent the components of the vector A and B, respectively. The summations over n are over the items for which both users i and j have recorded rates. The similarity ranges from -1 to 1. -1 means that the two vectors A and B are opposite direction. 1 means that their orientations are exactly the same, and 0 usually means that they are independent. [9]

5.3. Implement Memory-based CF

Firstly, we form our users-items matrix. Note that in this process, we assume that all mssing rateing is 0. Because of the previous data analysis, we know that the number of books read by each user is very different. A user has only evaluated at least one book, but has rated at most 200 books. Therefore, the problem we have is that if the user has read too few books, it is difficult to accurately find the user most similar to the predicted user. Therefore, we only predict users who have read more than 7 books, about 30k users. Secondly, we randomly select one of the books read by each user as a prediction. We look for users who have read the book among all users, and calculate the similarity between the predicted user and other users who have read the book by calculating the Pearson coefficient and the cosine similarity. Finally, we look for the top N users most similar to the predicted users. Look at their rating on the book and take their averages or mode, and round them to get our predictions. When processing small data, the entire data set has only 48 books. When the result is predicted, if the user reads less than two books, then directly select the most similar users. If the user has read more than 4 books, then the two closest users are averaged.

6. Model-based CF

In model-based CF we will firstly use the data to train a model, and then use the model to make prediction. In our practice, we try to implement the Naive Bayes CF model and the Probabilistic Matrix Factorization model on Matlab with our own code.

6.1. Naive Bayes Collaborative Filter

	I_1	I_2	I_3	I_4
U_1	4	?	5	5
U_2	4	2	1	
U_3	3		2	4
U_4	4	4		
U_5	2	1	3	3

Naive Bayes model is frequently used in multi-class classification. In the Naive Bayes model, it uses a Naive Bayes classifier, which assumes that the features are independent of the given class. With features $f_1, f_2 ... f_n$ and a class c_i, we will have the following equation [1]:

$$P(c_i, f_1, f_2, ..., f_n) = P(c_i) \prod_{j=1}^{n} P(f_j | c_i)$$

In our data set, suppose we are going to predict the rating result of user i on book j, which is the prediction rating $r_{i,j}$. The value of the prediction rating $r_{i,j}$ is our class, which ranges from 1 to 5. The prior probability of $P(r_{i,j})$ can be calculated through the other rating records of user i. The features are the rating records of other users', which is $r_{k \neq i,j}$, k is from 1 to n where n is the number of total user. Hence we have:

$$\begin{aligned} r_{i,j} &= \underset{r_{i,j}=1,2,3,4,5}{\arg\max}\ P(r_{i,j}, r_{1,j}, r_{2,j}, ...r_{i-1,j}, r_{i+1,j}, ..., r_{n,j}) \\ &= \underset{r_{i,j}=1,2,3,4,5}{\arg\max}\ P(r_{i,j}) \prod_{k=1, k \neq i}^{n} P(r_{k,j} | r_{i,j}) \end{aligned}$$

We build a Naive Bayes collaborative filter mainly based on this classifier.

6.1.1 Laplace Estimator

To avoid that some conditional probability could be 0, we also apply a Laplace estimator method as the following shows [8]:

$$P(X_i = x_i | Y = y) = \frac{\#(X_i = x_i, Y = y) + 1}{\#(Y = y) + |X_i|}$$

where $|X_i|$ is the size of the set $\{X_i\}$. In our case, the set of $\{r_{k,j}\}$ is 5. Hence for an example we have:

$$P(r_{k,j} | p_{i,j}) = \frac{0}{10} = \frac{0+1}{10+5} = \frac{1}{15}$$

Using the Laplace estimator we avoid the case that a conditional probability to be 0.

6.2. Probabilistic Matrix Factorization

Another model that we used in our practice is the Probabilistic Matrix Factorization, which is the PMF. PMF is one of most popular model-based collaborative filter, which is widely used in online predicting with excellent performance of prediction. In our practice, we try to learn how the PMF works and build up our own PMF model.

The motivation of PMF is simple. Suppose we have a complete rating matrix R with dimension m by n, we want to factorize it into two matrices: matrix U with dimension m by d and matrix V with dimension n by d, and we want that $R = UV^\top$.Here the parameter d stands for the dimension of latent features in a rating matrix. In our dataset, we have 10k books and 50k users. For those books they have different types, and for those users they also have different preferences, such as fiction or non-fiction etc. Those different types and preferences are just the latent features which are hidden behind the rating matrix. What we want to do is to extract these latent features out of the original rating matrix.

6.2.1 Model of PMF

According to the paper by [4], the model of PMF is as the following function shows:

$$\begin{aligned} f(\theta) = \sum_u \sum_i \Big\{ & W_{i,u} \Big(R^{o\&i}_{i,u} - (r_m + UV^\top)_{i,u} \Big)^2 \\ & + \lambda (\sum_d U^2_{i,d} + V^2_{u,d}) \Big\} \end{aligned}$$

where θ stands for all the parameters in the function. The function is just a MAP probabilistic framework. The first part is the square error between the true label, the observed and missing rating matrix $R^{o\&i}_{i,u}$, and our prediction $r_m + UV^\top$. Here r_m is a small rating offset for the missing data, which can help to improve the performance of the model. W is the weighting matrix, which equals to one for the observed data, and 0 or small value for the missing data. The second part is called the regularization term, which can be treated as the structure penalty related with the complexity of our model. By adding a appropriate regularization parameter λ, it can improve the performance of our model greatly.

Now our problem becomes the optimization of a least square problem, and for least square problem there should be a local minimum. To solve our main parameter U and V, we can just do the gradient descent and iteratively learn them. By taking the gradient we have:

$$\nabla_U f(\theta) = -2W.*\left(R^{o\&i} - (r_m + UV^\top)\right)V + 2\lambda U$$
$$\nabla_V f(\theta) = -2W.*\left(R^{o\&i} - (r_m + UV^\top)\right)V + 2\lambda V$$

Hence by adding a small learning rate ϵ, we can get the final solution of matrix U and V:

$$U = U - \epsilon\left\{-W.*\left(R^{o\&i} - (r_m + UV^\top)\right)V + \lambda U\right\}$$
$$V = V - \epsilon\left\{-W.*\left(R^{o\&i} - (r_m + UV^\top)\right)V + \lambda V\right\}$$

Although we have the equation for U and V now, we still need to know how to initialize them. We cannot initialize them with 0, or they will always be 0. We cannot equalize them either, or the row vector of U and V will always have the same value. Hence we can only randomize them. Since U and V is initialized randomly in high dimension, and we can only find the local minimum of out model, therefore the result of the PMF model may be slightly different each time you run it.

6.2.2 Optimization of Parameter

After solving the matrix U and V, we still have a lot of work to do, which is to optimize the other parameters. In our practice, we always set W_m and r_m to 0, which we haven't go deep in that. We try to optimize the learning rate ϵ, the regularization parameter λ and the dimension of latent features d.

First of all we try the different value of learning rate ϵ and the result is as the following figure shows: We can see that

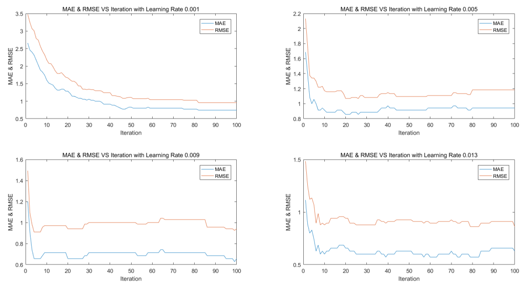


Figure 3. Different Learning Rate in PMF Model

with a small learning rate the curve converge slowly, and with a large learning rate it will begin to oscillate. With a even bigger one, it could just diverge to infinity. We choose $\epsilon = 0.001$ later.

Next we try to optimize the regularization parameter λ. We choose 10 λ for -0.04 to 0.05 and the result is as the following figure shows: The best one and the worst one have a 50% difference. Hence choosing a appropriate regularization is important to the PMF model. We choose $\lambda = 0.01$ with our dataset.

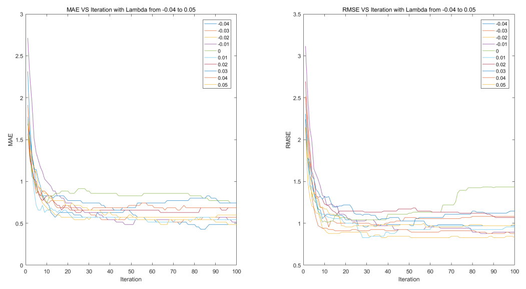


Figure 4. Different Regularization Parameter in PMF Model

The last one is the dimension of latent features d. Here we use our small dataset which is about 50 by 40 and try the value from 1 to 20, which is as the following figure shows:

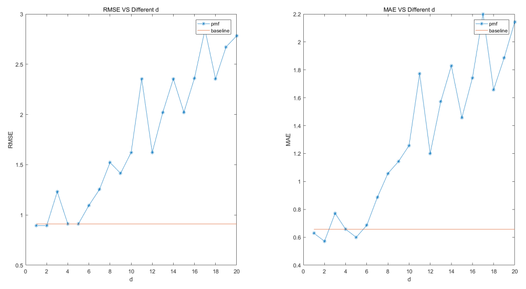


Figure 5. Different Dimension of Latent Features in PMF Model

Since the dataset is small, we cannot expect that there are too many lantent features in that. If we take $d = 20$, we could see that the model will just stop work. From the figure we can see that it is better to choose a d which is smaller than 5. Now we understand how these parameters wok. Finally we choose some appropriate parameter to train the entire dataset, and get the result as the following figure shows. It shows that out PMF model work correctly. Remember

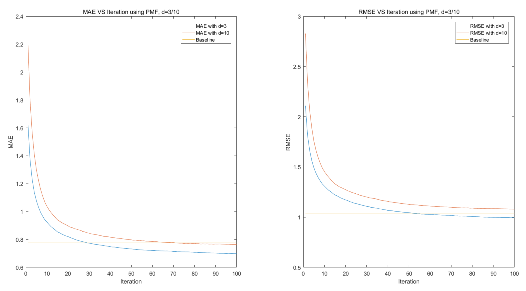


Figure 6. Training of PMF Model over Entire Dataset

that collaborative filter is always try to find the similarity between users or items, therefore we expect that the similar users and items should be in the similar position in the d dimension space after the factorization. Hence we have some interesting results to show in the appendix.

7. Experimental Result

Describe the experiments you have conducted on synthetic and real data and the results you have obtained (datasets, performance measures used, sensitivity to initialization or other algorithm parameters, impact of real-world degradations like missing data, etc.). Discuss the appropriateness of the performance metrics that you have chosen. Discuss if your results are consistent with your understanding, i.e., are they expected or unexpected? do they make sense?

Table 3. Result

Method	MAE	RMSD
Baseline (with Mean Value 4)	0.7757	1.0324
Cosine Similarity	0.8260	1.1911
Pearson Correlation	0.8575	1.1876
Naive Bayes	1.4744	1.8572
PMF	0.6989	0.9940

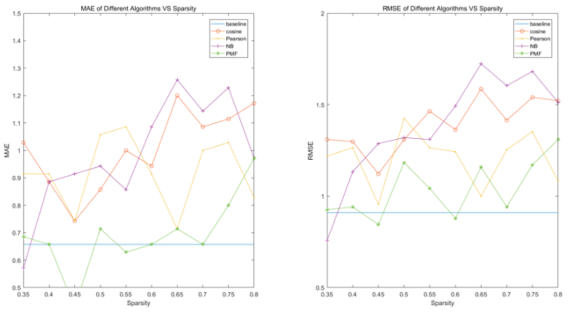


Figure 7. Different Algorith VS Sparsity

The results of each algorithm on the original data are analyzed through a table. In contrast, we can see that only the PMF method is better than our baseline and naive Bayesian method is the worst. The performance of the two user-based methods is basically the same. On the MAE, the cosine similarity is slightly better than the Pearson coefficient, and the two are almost identical in the performance of the RMSE. The Figure Different Algorithm VS Sparsity shows the relationship between the sparsity of the matrix and the experimental results. Due to the large amount of rating data lost and the matrix being too sparse, some of our experimental results are lower than our baseline. From the picture we can see in detail that the PMF method usually gives us the best solution. In general, the algorithmic error of Naive Bayes is the largest. But the wonderful thing is that if our matrix sparseness is small enough, Naive Bayes has the best effect. Compared with the two methods of calculating the similarity between Pearson coefficient and cosine similarity, the similarity of cosine is greatly affected by the degree of sparseness of the matrix, while the result calculated by Pearson's coefficient is relatively more stable.

8. Conclusion

Through the study of the entire project, we have a deeper understanding of the collaborative filtering algorithm. Especially for the two algorithms based on memory and model-based collaborative filtering. At the time of data processing, we realized a very important phenomenon. Rating missing is not random, and it often has certain user preferences. Therefore it cannot be ignored. In collaborative filtering, the sparseness of the matrix often plays a large role. The degree of matrix sparsity affects different algorithms differently. For example, the influence on PMF and Pearson coefficient is relatively small and stable, and the influence on cosine similarity and naive Bayes is great.

In the future, we will continue to study collaborative filtering algorithms, including more model-based algorithms, such as latent semantic analysis, Bayesian networks, and so on. In addition, we will continue to learn the hybrid collaborative filtering algorithm. Based on user collaborative filtering, nested using item collaborative filtering.

9. Description of Individual Effort

The contibutions of all team members to the project are as following:

Table 4. Individual Effort

Task	Lead
Data collection	Yu Zhao
Memory-based model using cosine similarity	Liyi Cao
Memory-based model using Pearson correlation	Yimeng Wang
Model-based model using Naive Bayesian	Yu Zhao
PMF tuning parameters	ALL

The contributions to the final report are as follows:

- Yu Zhao wrote: Model-based Literature, Naive Bayes Collabroative Filter, Probabilistic Matrix Factorization
- Liyi Cao wrote: Memory-based Literature, Abstract, Introduction, Dataset, Implement Memory-based CF, Experimental Result, Conclusion
- Yimeng Wang wrote: Memory-based Literature, Model Evaluating, Baseline, Introduction of Memory-based CF, Similarity, Optimization of Parameter

References

[1] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, pages 43–52, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[2] L. Cao, Y. Wang, and Y. Zhao. Github link: https://github.com/yuz1225/bu-ec503-project-fall-2018.git.

[3] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, and D. Sampath. The youtube video recommendation system. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 293–296, New York, NY, USA, 2010. ACM.

[4] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, pages 1257–1264, USA, 2007. Curran Associates Inc.

[5] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.

[6] J. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. In *Brusilovsky P., Kobsa A., Nejdl W. (eds) The Adaptive Web*, Lecture Notes in Computer Science, vol 4321, Berlin, Heidelberg, 2007. Springer.

[7] P. Spachtholz. Book recommender: Collaborative filtering, shiny : https://www.kaggle.com/philippsp/book-recommender-collaborative-filtering-shiny/data.

[8] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. In *Advances in Artificial Intelligence*, Vol. 2009, Article ID 421425, 2009.

[9] Wikipedia-contributors. Cosine similarity in wikipedia https://en.wikipedia.org/w/index.php?title=cosine$_{s}imilarity\&oldi$
$872928175, 2018, December 10$.

10. Appendix

10.1. Interesting Results about Matrix U and V After Factorization

Using the PMF methods, we can factorize the original rating matrix R into a item matrix U and a user matrix V. Since collaborative filter is always try to find the similarity between users or items, we expect that the similar users and items should be in the similar position in the d dimension space after the factorization. If we choose $d = 2$ or $d = 3$, we can visualize the user and item in a 2-d coordinate or a 3-d coordinate. First let's take a look on the result of our small dataset, which is as the following figures show: Since

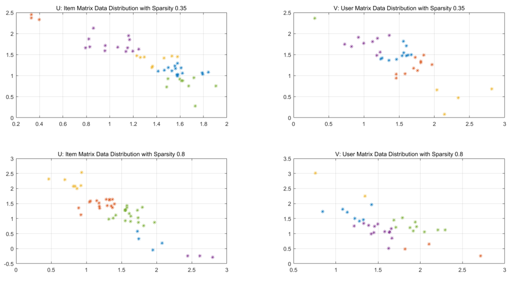


Figure 8. U and V of Small Dataset in 2-D

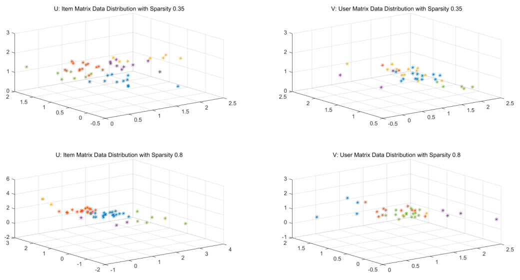
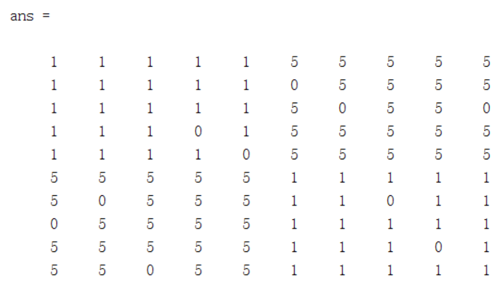


Figure 9. U and V of Small Dataset in 2-D

the number data is small it's hard to find very typical result. However we can still find that some points is close to each other, which means that they have the same type. To see a typical result, we create some very typical dataset which is as the following shows:



```
ans =

     1     1     1     1     1     5     5     5     5     5
     1     1     1     1     1     0     5     5     5     5
     1     1     1     1     1     5     0     5     5     0
     1     1     1     0     1     5     5     5     5     5
     1     1     1     1     0     5     5     5     5     5
     5     5     5     5     5     1     1     1     1     1
     5     0     5     5     5     1     1     0     1     1
     0     5     5     5     5     1     1     1     1     1
     5     5     5     5     5     1     1     1     0     1
     5     5     0     5     5     1     1     1     1     1
```

Figure 10. Typical Dataset

Here we can see that there should be two latent features. Then we use a PMF model with $d = 2$, and we can see the result of U and V is as the following figure shows:

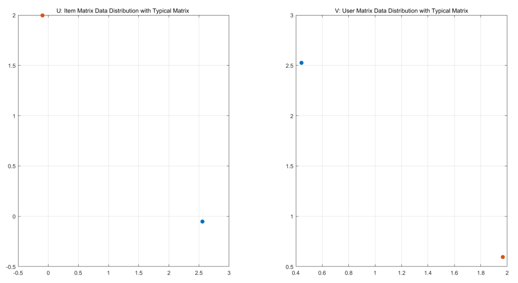


Figure 11. U and V Typical Dataset

In this case we can only see two points, which means all the same points is located at the same or very close position in the 2-d coordinate. Hence we verify what we expect. Finally, let's take a look on the result of our entire dataset using PMF model with $d = 3$: All the data look like a ball

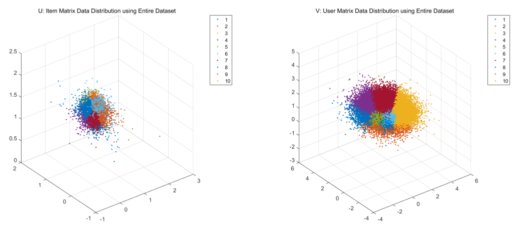


Figure 12. U and V Typical Dataset

in the 3-d coordinate, which make a lot of sense. From the figure we can see that similar users and similar items will locate at the similar location in the space, while different users and items will locate in different positions after the Probabilistic Matrix Factorization.