# CS 578 - Final Report

**He Huang, PUID:0027401415**
**Ying-Chun Lin, PUID:0030028008**

**Dataset:** MovieLens 100K Dataset `https://grouplens.org/datasets/movielens/100k/`

## Definition of Problem

The goal of this project is to recommend top-n movies to users according to their rating history. The first problem in this project is how to model user preferences according to their behavior in the dataset. The user preference is represented as a user feature vector, which may be ratings toward movies, the relation between the tags and ratings, etc. The second problem is that, after we know user preference, how can we recommend movies based on their preferences. The second problem has two fundamental tasks: (1) modeling movie features, and (2) building an accurate recommending model to generate top-n list for each user.

## Dataset Description

In this part, we do some descriptive analysis to the dataset. We briefly overview this dataset by looking at some basic statistic facts.

### Rating Analysis

We have a total of 99,999 ratings in range 1 to 5, involving only integers. 4 is most occurred in the ratings, and 3 is the second most, so the distribution is a little bit left-skewed. Over a half of ratings are 3 or 4. The mean value of ratings is 3.5, while standard deviation is around 1.1.

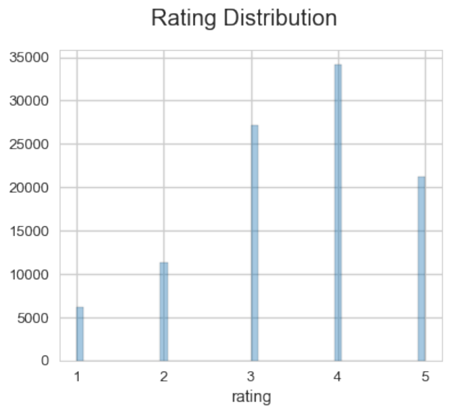


Figure 1: Rating Statistics

| | |
|---|---|
| count | 99999.000000 |
| mean | 3.529865 |
| std | 1.125678 |
| min | 1.000000 |
| 25% | 3.000000 |
| 50% | 4.000000 |
| 75% | 4.000000 |
| max | 5.000000 |

## User Analysis

We have a total of 943 users. Each of them rated at least 20 movies and at most 737 movies. The mean number of rated movie for users is 106 and standard deviation is around 100. It is a long-tailed and right-skewed distribution, which means most people rated 100 or less movies, and only few people rated a lot.

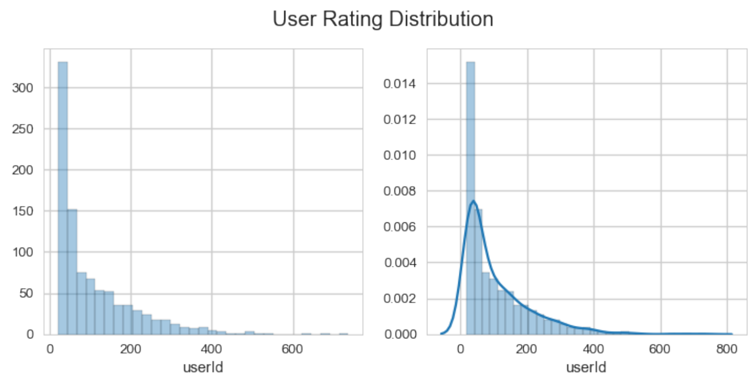


Figure 2: User Rating Statistics

| | |
|---|---|
| count | 943.000000 |
| mean | 106.043478 |
| std | 100.932453 |
| min | 20.000000 |
| 25% | 33.000000 |
| 50% | 65.000000 |
| 75% | 148.000000 |
| max | 737.000000 |

## Movie Analysis

We have a total of 1682 movies. They have been rated at least 1 time and at most 583 times. Mean value of number of ratings is around 60 but standard deviation is very huge, which is around 80. Most movies get 10 ratings or less, so the distribution is very right-skewed.

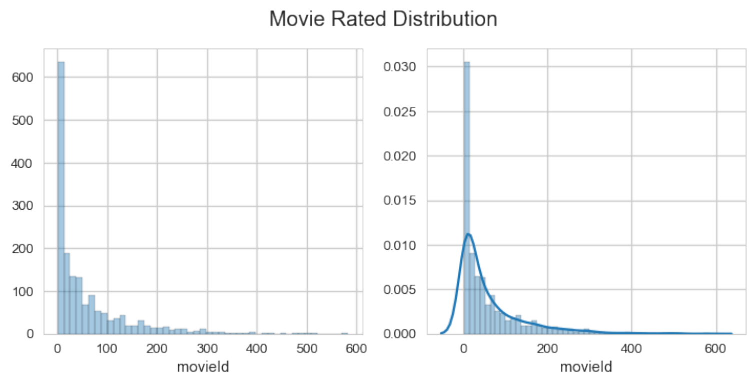


Figure 3: User Rating Statistics

| | |
|---|---|
| count | 1682.000000 |
| mean | 59.452438 |
| std | 80.383423 |
| min | 1.000000 |
| 25% | 6.000000 |
| 50% | 27.000000 |
| 75% | 80.000000 |
| max | 583.000000 |

# Preprocessing

Our goal is to recommend top-$n$ personalized movies to users. In this case, we split the original data into training and testing datasets with respect to each user. Specifically, given a user who rated several movies, 80% of the movie ratings will become our training data, and 20% of that is the testing data. For all users, we split their ratings in the same fashion. Therefore, we have 80% of the data for our model training and 20% for testing.

To prove the robustness of the comparing models, we will leave the testing data for the comparison between models. We can ensure that testing data is unseen during the model training phase. For model and parameter tuning, we only use our training data. For model tuning phase, we user 5-fold cross validation process. We split the training data, which is 80% of the original data, into 5 folds disjointly. Four folds are used for model training, and one fold is left for parameter tuning.

## Performance Evaluation Metrics

For parameter tuning, we would like to choose RMSE, which is rooted mean square error. We would like to calculate RMSE between predicted rating $\hat{r}_{u,i}$ and actual rating $r_{u,i}$ in the validation set. Then, we can choose the parameters which make more accurate prediction into our models to recommend items. It is defined as follows:

$$RMSE = \sqrt{\frac{\sum\limits_{r_{u,i} \in R, \hat{r}_{u,i} \in \hat{R}} (r_{u,i} - \hat{r}_{u,i})^2}{|R|}},$$

where $\hat{R}$ is the predicted rating matrix for all users and movies. Smaller RMSE means a better model.

For model comparison, we use average precision [1], average recall [2, 1], and accuracy to compare different recommending models thoroughly. Since our goal is to recommend top-$k$ items to user, we need to know how well a model can recommend right items according to learned user preference. Specifically, we calculate the precision, recall and accuracy for each person in our testing dataset. Then, we find the average for each metric to compare the performance between the recommending models.

The following equation is the definition of average precision [**ref**]. $L_u^{rec}$ denotes the recommendation list for a user $u$, and $L_u^{test}$ denotes the ground truth of in the testing dataset for user $u$. For each user, the precision is the number of items in both $L_u^{rec}$ and $L_u^{test}$ divided by the total recommended items, which is $|L_u^{rec}|$ . The average precision is simply the mean of precisions of all testing users.

$$Avg\_Pricesion = \frac{1}{n} \sum_{u=1}^{n} \frac{|\{i | i \in L_u^{rec} \wedge i \in L_u^{test}\}|}{|L_u^{rec}|}$$

We also compare the recall for our recommending models. The definition of recall is the number of items in both $L_u^{rec}$ and $L_u^{test}$ divided by the number of all ground truth items in the testing dataset for user $u$. We use the average recall for comparison.

$$Avg\_Recall = \frac{1}{n} \sum_{u=1}^{n} \frac{|\{i | i \in L_u^{rec} \wedge i \in L_u^{test}\}|}{|L_u^{test}|}$$

Finally, the accuracy shows how many times a model can accurately find the preferred items for a user. The definition is as follows:

$$Accuracy = \frac{\text{number of recommended lists containing true items}}{\text{number of recommended lists}}$$

## Collaborative Filtering Model and Parameter Tuning

Collaborative Filtering (CF) models are used in many recommending applications in practice. The basic idea behind CF models is that similar users would prefer similar items, or user preference can be inferred by the items he/she rated before.

We first observe how a Item-based CF (ICF) [3] for rating prediction. For the ICF model, the rating of an unseen movie is predicted by aggregating the ratings of similar movies for the movies in the user logs, denoted as $I_u = i_{u,1}, i_{u,2}, \cdots, i_{u,m}$. The rating is modeled as follows:

$$\hat{r}_{u,j} = \frac{\sum_{i \in I_u} w_{i,j} \cdot r_{u,i}}{\sum_{i \in I_u} w_{i,j}},$$

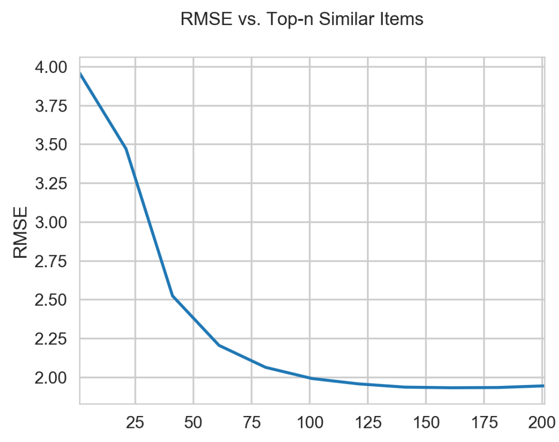
where $w_{i,j}$ can be viewed as similarity between movie $i$ and movie $j$. The similarity can be the cosine value of the feature vectors between $i$ and $j$, which is $w_{i,j} = \cos(\vec{x_i}, \vec{x_j})$. In our case, we simply use the user ratings towards the same movie $i$ in the training set as our feature vector.

Similarly, a User-based CF (UCF) [4] model predict item ratings based on how the similar users rates the items. Specifically, if user $u$ and user $v$ are similar to each other, and user $v$ has watched a movie $i$ which user $u$ has not watched yet, our best guess is that the rating of item $i$ for user $u$ is somehow close to the rating for user $v$. The rating is predicted as follows:

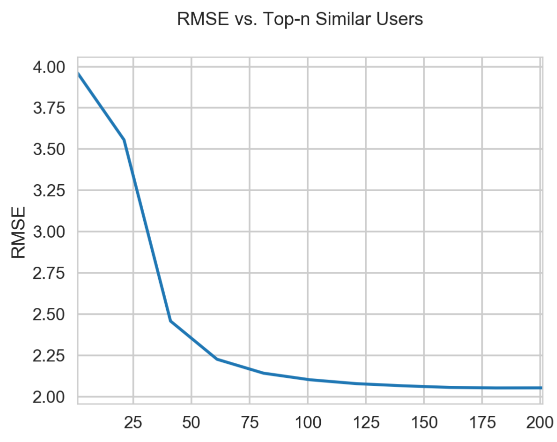
$$\hat{r}_{u,j} = \frac{\sum_{v \in U} w_{u,v} \cdot r_{v,i}}{\sum_{v \in U} w_{u,v}},$$

where $w_{u,v}$ is the similarity between user $u$ and user $v$, and $U$ is the set of users. Similar to item similarity, this can be modeled by the cosine value the feature vectors between $u$ and $v$.

Note that we does not prediction the ratings for all unseen movies. For each item $i_{u,m}$ in user log $I_u$, we find the top-$n$ similar items $i_j$ for that particular item $i_{u,m}$, and put $i_j$ into $\hat{I}_u$. For each $i_j \in \hat{I}_u$, we predict the ratings by above equation. We observe how the $n$ changes affects the prediction accuracy, which is RMSE in Figure 4a. We can observe that as the number of similar items increase, the predictions are more accurate. We use $n = 100$ for our model comparison. Likewise, we choose top-$n$ similar users for rating prediction of our UCF model, and the value is 100 as well.



(a) RMSE of Item-based CF.



(b) RMSE of User-based CF.

Figure 4: Parameter tuning for User-based CF and Item-based CF.

## Matrix Factorization Model and Parameter Tuning

Besides collaborative filtering, matrix factorization [5] is another basic approach of recommendation. Basically, matrix factorization would deal with users that rated disjoint items better. For example, if one rated 5 super hero movies, and another user rated 5 different super hero movies, the collaborative filtering might have a very low similarity, but matrix factorization does better when facing this kind of problems in practice.

In our approach, we used SVD to find a low rank approximation of the original rating matrix where every row represent a user while every column represent a movie. SVD can be represented mathematically by

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

where $R$ represent rating matrix, $U$ represent user features, $V^T$ represent movie features and $\Sigma$ represents weights (singular values). In this format, $U$ and $V^T$ are orthogonal matrices, while $\Sigma$ is a diagonal matrix.

A possible parameter in this model is number of features we keep, and it will be denoted as $m$. This is the most important feature in this model, and we cannot get this number from observation. Thus we perform a experiment, and tuning this parameter using the metrics of RMSE on on a 5-fold validation. Using the training and validation sets, we did 5 experiments for each $m$ we choose.

First we did a test at a step of 50, so that we can have an overview on the dataset. We find that in this case RMSE is lowest at around 50. We can treat the model as overfitting over 100 features. In the figure below, the line represents mean RMSE in 5 experiments, and the light blue part represents standard deviation.

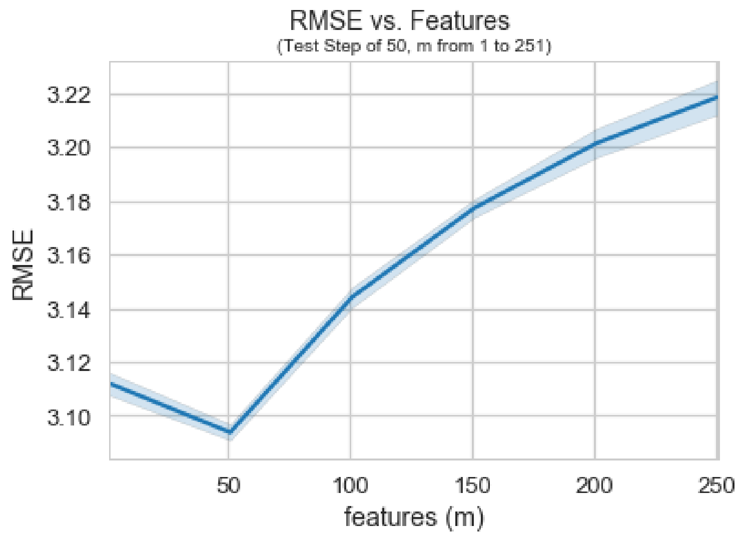


Figure 5: RMSE for MF models with time step of 50.

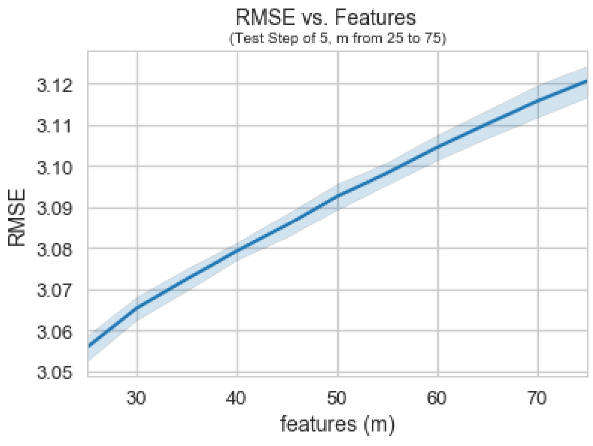
Based on result above, we tried to do smaller steps in smaller intervals. When we do the experiment in the interval of 25 to 75, RMSE is strictly increasing, which means the model is overfitting in this interval. Thus we did another experiment in range 1 to 24. We find the model is underfitting when there are less than 5 features, and it is overfitting when there are over 10 features. We should do some further experiment on other metrics to see which number is the best for this dataset.

In conclusion, matrix factorization performs best when we keep top 5 to 10 features. We would keep this result for further testing.

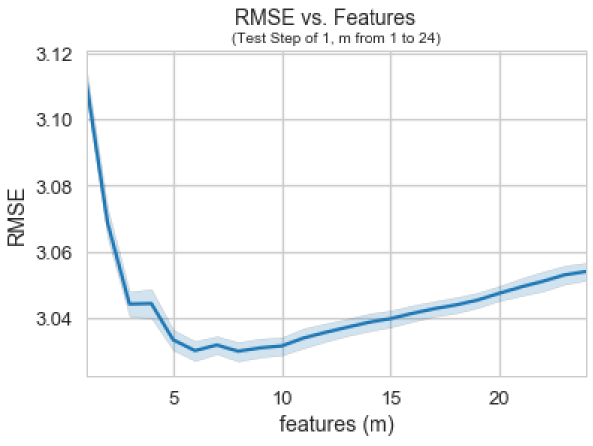
## Comparison Between Models

In this section, we compare the three models: (1) User-based CF, (2) Item-based CF, and (3) Matrix Factorization for the top-$k$ recommending tasks. If a model has better performance, it means that the model can identify user preference accurately, and can recommend the correct items to users. We evaluate the performance of each model in terms of average precision, average recall, and accuracy.

In Figure 7, we can observe that the similar trends between different models. We show how the change of $k$ value can affect precision in Figure 7a. When the number of recommended items in the list increases, the value of precision decreases. Although the recommended items grows, these models usually recommend items which users does not prefer to

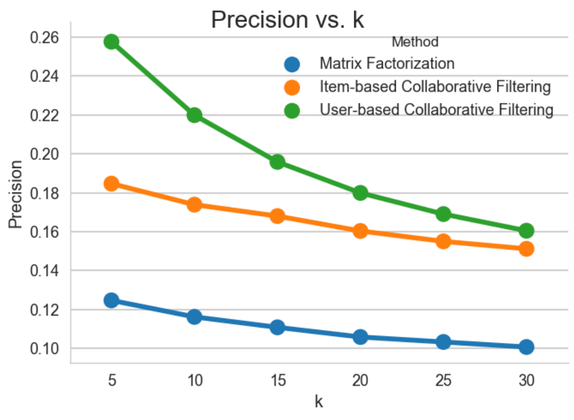


(a) Time step of 5.

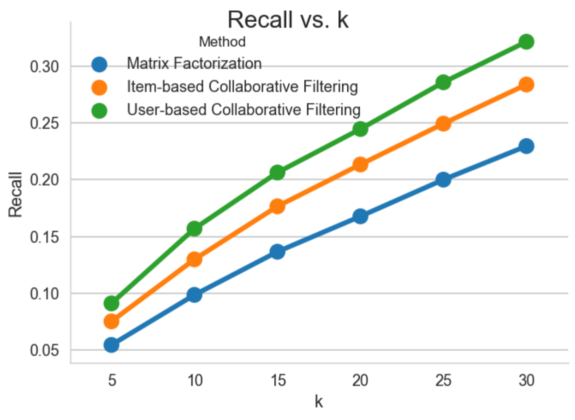


(b) Time step of 1.

Figure 6: RMSE for MF models with different time steps.



(a) Precision.



(b) Recall.

Figure 7: Recommending performance in terms of precision and recall.

so much. It indicates that the number of preferred items in the list does not grow as fast as the list grows. Therefore, the precision value decreases. This indicates that all three models often suggest the items which users do not have any positive feedbacks.

On the other hand, we observe the opposite trend in Figure 7b against the trend in Figure 7a. When the $k$ value increases, the value of recall increases as well. Because the number of recommended items grows, a model has higher chance to include the preferred items into the list. As a result, the recall value increases accordingly.

Finally, Figure 8 demonstrates that if we raise the value $k$, the accuracy of all three models can reach greater than 90%. The reason of the increasing trend is similar to what happens when we observe the trend in Figure 7b. This suggests that most of the recommended item in the lists have at least one items which a user like when a model recommend more items for the user.

In conclusion, all of the above figures have shown that User-based CF has best performance in terms of precision, recall and accuracy among all three models. Since our goal is to recommend movies to users, movie recommended by similar users may accurately identify the movie preference of our target users. Item-based CF models does not perform as well as User-based CF because users seldom repeatedly watch the same type of movies. They may prefer comedy movies at this time, and romance movie for the next time. Therefore, the similar-item approach may not work well in this scenario. For a matrix factorization model, we can observe that the rating prediction does not performance as well as both CF models in Figure 5 and Figure 6. This may be due to the sparsity of the rating matrix, and need more sophisticated techniques to overcome the challenge. Due to the higher rating prediction error, it is difficult for a matrix factorization model to recommend the right items in our top-$k$ recommendation tasks. We finally conclude that User-based CF can capture user preference accurately from our experiments.

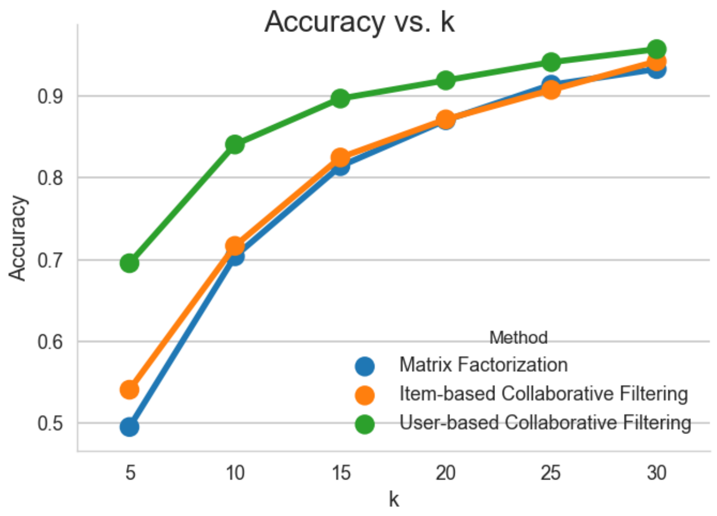


Figure 8: Recommending performance in terms of accuracy.

## References

[1] A. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization: scalable online collaborative filtering," in *Proceedings of the 16th International Conference on World Wide Web, WWW 2007*, 2007, pp. 271–280.

[2] C. Yang, X. Yu, and Y. Liu, "Continuous KNN join processing for real-time recommendation," in *2014 IEEE International Conference on Data Mining, ICDM 2014*, 2014, pp. 640–649.

[3] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, 2003.

[4] Z. Zhao and M. Shang, "User-based collaborative-filtering recommendation algorithms on hadoop," in *Third International Conference on Knowledge Discovery and Data Mining, WKDD 2010*, 2010, pp. 478–481.

[5] Y. Koren, R. M. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Computer*, vol. 42, no. 8, pp. 30–37, 2009.