# Recommender System（topic 9）

**z5002356 Yuxiang Qiu**
**z5085805 Yongxin Sun**
**z5077824 Hui Jia**

## 1.Introduction

Living in the time of information explosion, almost all of us would suffered how to make make good choices based on tons of information. The recommender system would help us especially when we do the shopping, watching, listening on line. In general, there are three designs for recommender system, content-based, knowledge-based and collaborative. The goal of this project is build the movie recommender system by using collaborative filtering algorithm. Collaborative filtering(item) is a method which can make the prediction for a user based on the historical preferences of this user. The specific steps are shown below, I implement it by using the item-based collaborative filtering based on knn model, and evaluate this system with MAE and RMSE

(a) Implement a user-friendly text-based UI to visualize the result.
(b) Predict the rate of unseen movie for the user.
(c) Recommend a movie list for the user
(d) Evaluate the system with MAE and RMSE

Besides, in order to make this system to be more convinced, persuaded and scientific. We add some extra features:
1. We implement two programs(one is based Cosine similarity, the other is pearson) to compare the evaluation.
2. In order to make user easy to use, we implement a delicated UI, show all the result in the table form.
3. We use the cross-validation(80%,20%) to evaluate MAE and RMSE

## 2.Method

Generally, there are two approaches using collaborative filtering, one is user-based, the other is item-based. For this project, I will use item-based. In order to do the project clearly, I set some simple step below:

### Step 1: Initialize to numpy matrix

In order to make this system more efficiently, I loading the training data(MovieLens 100k) into numpy matrix[movieId * userId]. Because numpy can help us perform all kinds of vector and matrix computation efficiently.

### Step 2: Get similarity measures

After preprocessing at step1, each move is a vector in the form like:
Movie_1:[0,1,2,3,4,5,1,2,3….], the value in the vector represent rate rated by the user as the index specified, for example: user_1 rate 0, user_2 rate 2…. In order to make it clear, I draw table below with small data.

| | User_1 | User_2 | User_3 | User_4 | User_5 |
|---|---|---|---|---|---|
| Movie_1 | 0 | 1 | 2 | 0 | 4 |
| Movie_2 | 1 | 0 | 3 | 4 | 0 |
| Movie_3 | 4 | 5 | 0 | 1 | 5 |
| Movie_4 | 4 | 2 | 0 | 0 | 0 |

Next, we should consider how to get the similarity between the movies. I implement two similarity function, one is **Cosine** and the other is **Pearson correlation**. And I will also compare the performance by using these two measures later.

## Cosine Similarity:

$$\mathbf{sim(i,j)} = \boldsymbol{cos(i,j)} = \frac{\boldsymbol{i \bullet j}}{|\boldsymbol{i}| * |\boldsymbol{j}|}$$

where "•" denote the product of two vector, "|i|" denote the length of vector I, and I, j are the movie vector.
For example the similarity of movie1 and movie2:

$$\mathrm{sim}(1,2) = \frac{0*1+1*0+2*3+0*4+4*0}{sqrt(1*1+2*2+4*4)*sqrt(1*1+3*3+4*4)} = \frac{6}{sqrt(546)} = 0.2567$$

the large similarity is, the more similarity they would be

## Pearson correlation:

$$\boldsymbol{sim(x,y)} = \frac{\sum_{s \in S_{xy}} (\boldsymbol{r_{xs}} - \overline{\boldsymbol{r_x}})(\boldsymbol{r_{ys}} - \overline{\boldsymbol{r_y}})}{\sqrt{\sum_{s \in S_{xy}} (\boldsymbol{r_{xs}} - \overline{\boldsymbol{r_x}})^2} \sqrt{\sum_{s \in S_{xy}} (\boldsymbol{r_{ys}} - \overline{\boldsymbol{r_y}})^2}}$$

where $\boldsymbol{r_x, r_y}$ is the average rate of vector x and vector y, $\boldsymbol{Sxy}$ is the items both rated by the user[1].
In order to compute Pearson correlation efficiently, I build a averageMatrix associated with the rate matrix above

| | User_1 | User_2 | User_3 | User_4 | User_5 |
|---|---|---|---|---|---|
| Movie_1 | 0 | 1-7/3 | 2-7/3 | 0 | 4 – 7/3 |
| Movie_2 | 1 – 8/3 | 0 | 3 – 8/3 | 4 – 8/3 | 0 |
| Movie_3 | 4 – 15/4 | 5 – 15/4 | 0 | 1 – 15/4 | 5 – 15/4 |
| Movie_4 | 4 – 6/2 | 2 – 6/2 | 0 | 0 | 0 |

Then we can compute cosine similarities between rows
For example: the Pearson similarity between movie_1 and movie_2 is:

$$\mathrm{Sim}(1,2) = \frac{\left(2-\frac{7}{3}\right) \times \left(3-\frac{8}{3}\right)}{\sqrt{\left(1-\frac{7}{3}\right)^2 + \left(2-\frac{7}{3}\right)^2 + \left(4-\frac{7}{3}\right)^2} \times \sqrt{\left(1-\frac{8}{3}\right)^2 + \left(3-\frac{8}{3}\right)^2 + \left(4-\frac{8}{3}\right)^2}} = -0.024$$

Compared with cosine similarity between movie1 and movie2, the Pearson <0 and cosine similarity > 0. I implemented both of them, and demonstrate which one would be better later.

### Step 3: Predict rate

Having get the similarity between items, then we would use these measures to predict rate, based on formula below[2]:

$$P_{u,i} = \frac{\sum_{similar\ movies,N}(S_{i,n} * R_{u,n})}{\sum_{similar\ movies,N}(|S_{i,n}|)}$$

Specifically, to get the predication on movie I for a user u, we get the sum of rating given by the user on the movies similar to i. Each rating is weighted by the corresponding similarity $S_{i,j}$ between movie I and j.

Besides, we also use **K-Nearest Neighborhood** to predict ratings which only look for n neighbors among movies that the user rated which have the largest similarity with the target movie. After the experiments, we found the accuracy of prediction perform best when N ≈ 15 by using **Cosine** similarity.

### Step 4: Recommend List

This is mainly to recommend the user with list of movies. We predict all the movies that the user unseen, and select top-N of the movies with highest rate to the user.

## 3.Result

### 3.1 Experiment result

We implemented two versions of recommender system. One is our core code, **cf_item_knn.py** which is based on Cosine similarity and the other is **cf_pearson.py** which is based on pearson similarity. The result below is based on **cf_item_knn.py**

Firstly, this system received the target userId and target movieId(unseen) as inputs, and then output the predict rate of the movie for this user, ranging from [0.0, 5.0].
Secondly, take userId and number of recommend movies as input, and output the recommend movies sorted by the predict rate. Each of them contains movieId, title, release date and predict rate.

Last, we also provide the interface to allow user to access the MAE and RMSE to the training data, at this time we should input datasetId, and neighbors. Then the system would compute MAE and RMSE after 1minute. All the sample result are shown in Appendix.

## 3.2 Evaluation

Recommender system are usually evaluated by the predication accuracy. Among them, the MAE and RMSE are the popular measurement received by public.

$$\text{MAE} = \frac{1}{n} \sum_1^n |p_i - a_i|$$

The pair($p_i$ , $a_i$) refer to the prediction rate and actual rate based i-th item. We use MAE to measure average absolute deviation between predict rate and actual rate. Also, we use Root Mean Squared Error(RMSE), the variation of MAE to measure the performance.

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_1^n (p_i - a_i)^2}$$

Based on KNN, we use these two measurements to select the best n as the number of neighbors. We use cross-validation to train $U_i$.base, (i = 1,2,3,4,5) and use $U_i$.test, (i = 1,2,3,4,5) to measure MAE and RMSE with different K respectively.
And I find that for the Cosine similarity, it perform best at K ≈ 15,and also find that for the Pearson similarity, it perform best at K ≈ 50. The graph can be seen on Appendix. The more specific data can be seen at result.txt in this folder.

Cosine similarity, K = 15

| **Cosine** | **U1** | **U2** | **U3** | **U4** | **U5** |
|---|---|---|---|---|---|
| **MAE** | 0.7727 | 0.7565 | 0.7518 | 0.7607 | 0.7603 |
| **RMSE** | 0.9925 | 0.9718 | 0.9657 | 0.9743 | 0.9720 |

Pearson similarity, K = 50

| **Pearson** | **U1** | **U2** | **U3** | **U4** | **U5** |
|---|---|---|---|---|---|
| **MAE** | 0.8395 | 0.8280 | 0.8146 | 0.8222 | 0.8223 |
| **RMSE** | 1.1057 | 1.0755 | 1.0394 | 1.0542 | 1.0508 |

Avg_MAE(Cosine) = 0.7604 Avg_RMSE(Cosine) = 0.9753
Avg_MAE(Pearson) = 0.8253 Avg_RMSE(Pearson) =1.0651

From the results above, we can find that the Cosine_similarity with K = 15 smaller than Pearson_similarity with K = 50 in this case. Besides, we know that all individual residuals in MAE are equally weighted while in RMSE large errors get penalized more than small errors since the errors are squared before averaged. Therefore the item-based collaborative filtering algorithm using Cosine similarity based on KNN would be the best choice.

## 4.Discussion

**4.1** I think one drawback in my code is that I initialise all the movies the which user unseen as 0, which can be misunderstanding with rate 0. For example, if userA rate movieA at 0, I would regard it as unseen for userA. This should be optimized.

**4.2** It is about the implementation and optimization of how to get similarity which I stuck for several days. One thing is that(I think it's very important), we only need to compute the similarity between the unseen movies and seen movies. For example, for the user_movie table shown at step2, I just select user_4 as the target user, when we want to predict the movie1 for user_4, we only need to compute the similarity with (movie1, movie2), (movie1, movie3), rather than need to get similarity with (movie1, movie4). Because the as item_based algorithm, we only need to consider the movies the user rated as our preferences. In other words, the movies seen are the only evidences we can use to predict the other movie.

By doing so, I decrease from 20min to 1 min to get the MAE and RMSE for one base file and test file(u1.base, u1.test). It's very efficient. Otherwise, you will be stuck not only the run time but also the result, because it's sparse matrix.

**4.3** Compared with user_based algorithm, I think item_based can be more reasonable, convinced persuade, and more acceptable for the user. Because the recommend movie is based on the movies the user seen. However the user_based would perform a little bad. Because we do not know the others who are similar to us.

## 5.Conclusion

From the result we can conclude that item-item collaborative filtering algorithm by using Cosine similarity based on KNN(k = 15) model perform better(mae, rmse) compared with by using pearson similarity in this case.

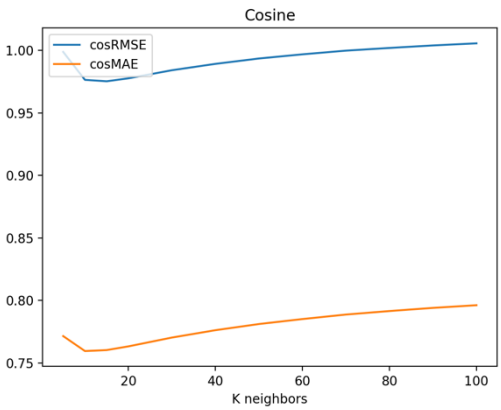
## 6.References

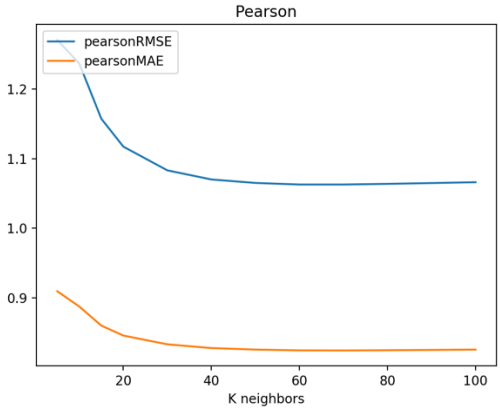
[1] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl: Item-based Collaborative Filtering Recommendation Algorithms, GroupLens Research Group/Army HPC Research Center (2001)

[2] P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. AAAI, 2002.

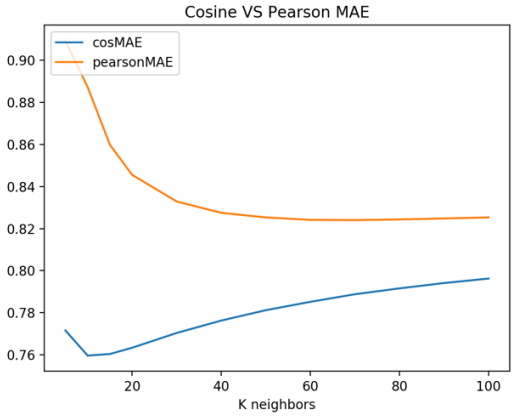
## 7.Appendix

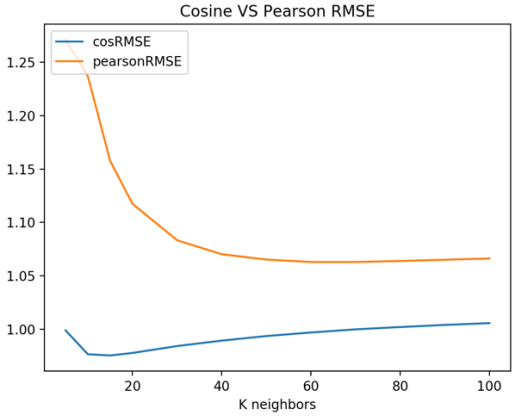
1. System Evaluation Result(specific data in result.txt):




Cosine, Pearson Evaluation




MAE, RMSE Evaluation

2. Recommender Demo:

Display user rated list:

```
jiahuis-MacBook-Pro:project Jason$ python cf_item_knn.py
Initialise Success...
~~~~~~~~~~~~~~~~Welcome to Recommender System based on MovieLen 100k~~~~~~~~~~~~~~~~
Action Menu:
display                 Display user rates
predict                 Predict user's rating on movie
recommend               Recommend List for user
evaluate                Evaluate Recommender System
quit                    Quit
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Please input action key:   (? for menu)
display
Please input userId:
100
+----------------+-----------------------------------------+-------------+
|     MovieId    |                Movie Title              |     Rate    |
+----------------+-----------------------------------------+-------------+
|     258        |               Contact (1997)           |     4.0     |
|     266        |          Kull the Conqueror (1997)      |     2.0     |
|     268        |             Chasing Amy (1997)          |     3.0     |
|     269        |            Full Monty, The (1997)       |     4.0     |
|     270        |                Gattaca (1997)           |     3.0     |
|     271        |           Starship Troopers (1997)      |     3.0     |
|     272        |           Good Will Hunting (1997)      |     4.0     |
|     286        |         English Patient, The (1996)     |     3.0     |
|     288        |                Scream (1996)            |     2.0     |
|     289        |                Evita (1996)             |     3.0     |
|     292        |              Rosewood (1997)            |     2.0     |
|     294        |              Liar Liar (1997)           |     4.0     |
|     300        |            Air Force One (1997)         |     4.0     |
|     302        |           L.A. Confidential (1997)      |     4.0     |
|     310        |           Rainmaker, The (1997)         |     3.0     |
|     313        |                Titanic (1997)           |     5.0     |
|     315        |              Apt Pupil (1998)           |     5.0     |
|     316        |          As Good As It Gets (1997)      |     5.0     |
```

Predict rate, recommend movies and evaluate system

```
~~~~~~~~~~~~~~~~Welcome to Recommender System based on MovieLen 100k~~~~~~~~~~~~~~~~
Action Menu:
display                 Display user rates
predict                 Predict user's rating on movie
recommend               Recommend List for user
evaluate                Evaluate Recommender System
quit                    Quit
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Please input action key:   (? for menu)
predict
Please input userId and predict MovieId separated by space:
100 100
Predict Rate:  3.52444575976
Please input action key:   (? for menu)
recommend
Please input userId and topN of recommend list separated by space:
100 5
+----------------+-----------------------------------------------------+--------------------+----------------------+
|     MovieId    |                     Movie Title                     |       Release      |      Predict Rate    |
+----------------+-----------------------------------------------------+--------------------+----------------------+
|     1619       |                All Things Fair (1996)               |     08-Mar-1996    |     4.26218716383    |
|     1556       |                 Condition Red (1995)                |     01-Jan-1995    |     4.21736009494    |
|     486        |                    Sabrina (1954)                   |     01-Jan-1954    |     4.03117866119    |
|     1674       |                  Mamma Roma (1962)                  |     01-Jan-1962    |          4.0         |
|     312        |    Midnight in the Garden of Good and Evil (1997)   |     01-Jan-1997    |     3.98028324785    |
+----------------+-----------------------------------------------------+--------------------+----------------------+
Please input action key:   (? for menu)
evaluate
training SetId: [1|2|3|4|5]      with Cosine similarity
Please input training setId, neigbours
1 20
Runing... Please wait about 1m...
Err:  15513.9340377  count:  20000
MAE:  0.775696701884   RMSE:  0.9955925210000118
Please input action key:   (? for menu)
```