Recommender System using Collaborative Filtering

COMP9417, Assignment 2

Group name: PSG.LGD

Group member: z5147182 Xinyuan Qian z5137595 HengMing Wang z5129023 Zhihao Ye

1.Introduction

Recommender Systems apply Information Retrieval techniques to select the online information relevant to a given user. Collaborative Filtering (CF) is currently most widely used approach to build Recommendation System. CF techniques uses the user' behaviour in form of user-item ratings as their information source for prediction. There are major challenges like sparsity of rating matrix and growing nature of data which is faced by CF algorithms. These challenges are been well taken care by multiple algorithms.
In this project we attempt to build three Recommender Systems based on three algorithms, namely User-user Collaborative Filtering, Item-item Collaborative Filtering and Matrix Factorization (MF), Then evaluate each of them under different coefficients.

2.Implementation

2.1 User or Item based Collaborative Filtering

2.1.1 Correlation-based Similarity

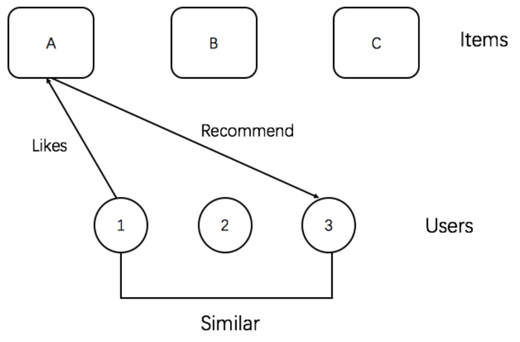
There are several different measures to determine the similarity of two users/items. One of the most popular measures which is widely used in Collaborative Filtering is Pearson Correlation Coefficient.

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

This coefficient is the covariance of the two vectors divided by the product of their standard deviations. It shows the tendency of two series of numbers, paired up one-to-one and then move at the same time. The coefficient will be close to -1 if the two vectors have opposite tendency and it will be close to 1 if the two vectors have the same tendency. So Pearson Correlation Coefficient is applied in both user-user CF and item-item CF to compute the similarity between the user-user and item-item.

2.1.2 User-user CF

User-User Collaborative filtering tends to predict the items to the target user. These items are interested by other users who are similar to the target user. This method is based on KNN algorithm with Pearson Correlation Coefficient as the distance. Searching for the neighbors of a user is an O(|U|) operation. And there is an assumption in user-user CF method is that we assume the similar users have the similar favours. If one user does not have the same favour with his similar users, then the result of this method will be bad.



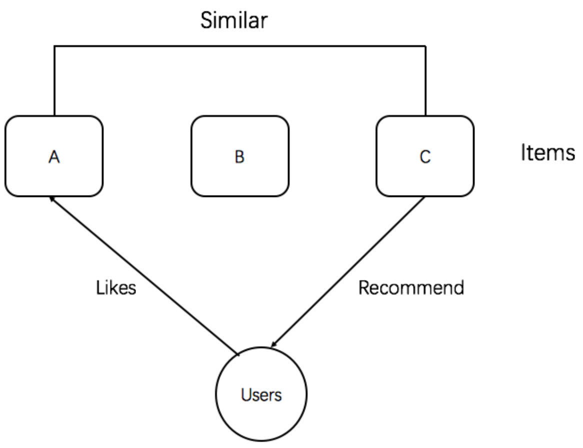
Here is the pseudo code of the implementation:

```
input target user u, the number of the similar users k
for each item i that user u has no rated yet:
    for every other user v that has rated for i:
        compute the similarity s between u and v
        if the similarity s is in top-k:
            add v's rating for i, weighted by s, to a running average
return the top items,ranked by weighted average
```

After tested by different NIL values (the number of most similar users taken into account), it is found that performance is better when this coefficient is 25 for the smaller dataset and 50 for the larger dataset. Additionally, in the experimentation part, NIL is taken as 25 for 100k dataset and 50 for 1M dataset as well.

2.1.3 Item-item CF

Item-Item Collaborative filtering first finds target user’s favorite items as target items and then uses the items which are similar to items to predict target user’s favour. Similar to user-user CF, item-item CF is also based on KNN algorithm with Pearson Correlation Coefficient as the distance. And find out the k-top similar items costs O(|U|) complexity. There is an assumption in item-item CF as well. That is each user will not change their flavours compared with what they liked before. If one user always changes his flavours, then the result from item-item CF will not very good.



Here is the pseudo code of the implementation.

```
input target user u, the number of the similar items k
for each item i that u has no rated yet:
    for every other item j that u has rated for:
        compute the similarity s between i and j
        if the similarity s is in top-k:
            add u's rating for j, weighted by s, to a running average
return the top items,ranked by weighted average
```

After tested by different NIL values (the number of most similar items taken into account), it is found that performance is better when this coefficient is 25 for the smaller dataset and 50 for the larger dataset. Additionally, in the experimentation part, NIL is taken as 25 for 100k dataset and 50 for 1M dataset as well.

2.2 Matrix Factorization

Matrix factorization is a low rank approximation method to find two (or more) matrices whose product best approximate the original matrix. In our project, we implement our own matrix factorization model to decompose the sparse user-movie matrix into two matrices of user and movie and use inner product of these two matrices to estimate all missing ratings.

2.2.1 Basic Matrix Factorization

Assuming our rating matrix is R, and we need to decompose it into the user matrix P and the movie matrix Q:

$$R_{M\times N} = P_{M\times K}{Q_{N\times K}}^T\text{, where R << M, N}$$

For a given user, P_u is high factors of user's ratings of movies and for a given movie, Q_u is high factors of users giving ratings on it. $P_u{Q_u}^T$ approximates all users' ratings in all movies.

Let r_{ui} be the estimate rating:

$$r_{ui} = P_u{Q_i}^T$$

Once we get the proper value of P and Q, we can estimate all missing ratings easily by the function above. So, the key problem is how to get P and Q.

The squared error of training model:

$$\text{error} = \min \sum (t_{ui} - P_u{Q_i}^T)^2 + \beta(||P_u||^2 + ||Q_i||^2)$$

where t_{ui} is the training data and β is a constant to avoid overfitting. Our model tries to minimize the error by stochastic gradient descent.

The prediction error between real value and predicted value:

$$e_{ui} = r_{ui} - P_u{Q_i}^T$$

Then we can modify the P and Q by gradient descent:

$$P_u = P_u + \alpha * (e_{ui} * Q_i - \beta * P_u)$$
$$Q_i = Q_i + \alpha * (e_{ui} * P_u - \beta * Q_i)$$

By iteration we will get our model for basic matrix factorization.

2.3.2 Matrix Factorization with Bias

Matrix factorization is a good method in collaborative filtering to deal with various data aspects, however, the basic matrix factorization only cares about the collaborative effect of user and movies, it may cause some users give high ratings than others or some movies receive much higher ratings.

The bias is a combination of other effects that should be considered, like the average ratings, the effect of individual person and individual movie. The bias of r_{ui}:

$$b_{ui} = mean + b_u + b_i$$

the mean represents the overall average ratings, b_u and b_i are the observed deviations of user u and movie i, now, the estimate rating with bias is:

$$r_{ui} = b_{ui} + P_u{Q_i}^T$$

and the squared error of training model with bias:

$$\text{error} = \min \sum (t_{ui} - b_{ui} - P_u{Q_i}^T)^2 + \beta(\left||P_u|\right|^2 + \left||Q_i|\right|^2 + {b_u}^2 + {b_i}^2)$$

By using stochastic gradient descent to minimize the error, the prediction error:

$$e_{ui} = r_{ui} - b_{ui} - P_u{Q_i}^T$$

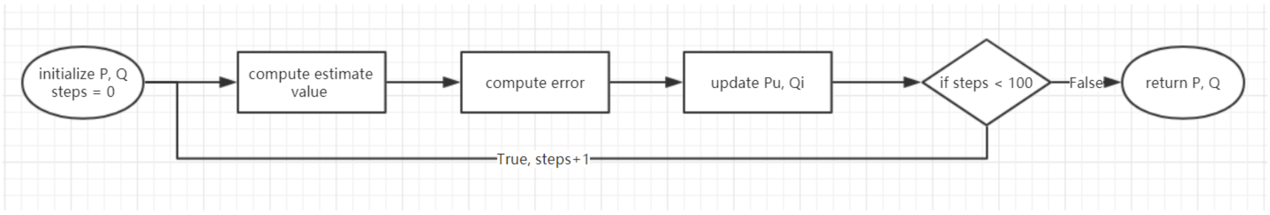
We should modify P, Q, b_u and b_i by gradient descent:

$$P_u = P_u + \alpha * (e_{ui} * Q_i - \beta * P_u)$$
$$Q_i = Q_i + \alpha * (e_{ui} * P_u - \beta * Q_i)$$
$$b_u = b_u + \alpha * (e_{ui} - \beta * b_u)$$
$$b_i = b_i + \alpha * (e_{ui} - \beta * b_i)$$

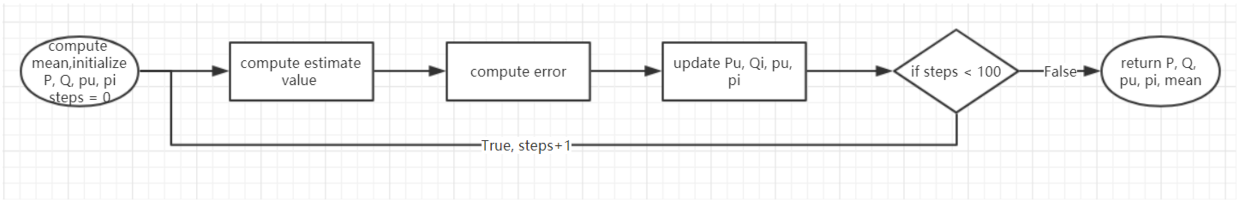
By iteration we will get our model for matrix factorization with bias.

2.3.4 Implement of Matrix Factorization

Basic matrix factorization:



Matrix factorization with bias:



Elements in P, Q are initialed with random value between 0 and 1, the size of P is M * K and the size of Q is N * K, where M is the number of users, N is the number of movies and K is number of the most K influential features, in our project, K = 100. P_u, P_i are two arrays with the length M and N, elements in P_u, and P_i are also initialed with random value between 0 and 1.

The value of α and β:

We use fixed training data and test data to find out the proper value of parameters, and we measure the effect of different parameters by computing the RMSE between test data and estimate data.

α	β	RMSE of basic MF	RMSE of MF with bias
0.01	0.15	0.9306	0.9263
0.02	0.15	0.9443	0.9303
0.03	0.15	0.9605	0.9339
0.02	0.10	0.9324	0.9207
0.01	0.10	0.9221	0.9178

In our project, the learning rate $\beta = 0.10$ and $\alpha = 0.01$

3.Experimentation

3.1 Evaluation criterion

In the experiments, we evaluate our recommender algorithms' accuracy by Root Mean Squared Error (RMSE) as it is the most common performance measure in the evaluation of regression and classification algorithms in the machine learning and statistic literature (Duda and Hart, 1973; Bengio and Grandvalet, 2004). If $p_{i,j}$ is the predicted rating for user i over movie j, and $v_{i,j}$ is the true rating, and K = $\{(i,j)\}$ is the set of hidden user-item ratings then the RMSE is defined as:

$$\sqrt{\frac{\sum_{(i,j)\epsilon K}\left(p_{i,j} - v_{i,j}\right)^2}{n}}$$

RMSE is suitable for our task because it measures inaccuracies on all ratings.

3.2 Dataset information

We test our Recommender System under two real-world dataset from https://grouplens.org/datasets/movielens/. Detailed dataset information is showed below:

Dataset	Number of Users	Number of Movies	Rating Scale	Number of Ratings	Density
MovieLens(100k)	1682	943	0-5	100000	6.30%
MovieLens(1M)	3705	6040	0-5	1000209	4.46%

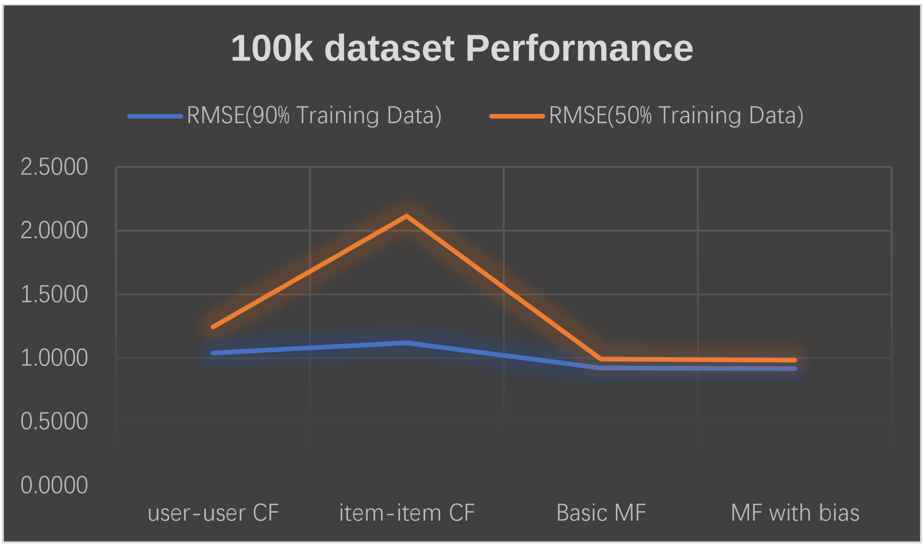
3.3 Results

We test basic MF and MF with bias on both datasets. While considering the efficiency of User-user CF and Item-item CF, we only test them on the 100k dataset.

For train and test data selection, we attempt two ways: 90% or 50 % random allocate for training and remaining allocate for testing.

Result of 100k dataset is showed in the table and corresponding figure below:

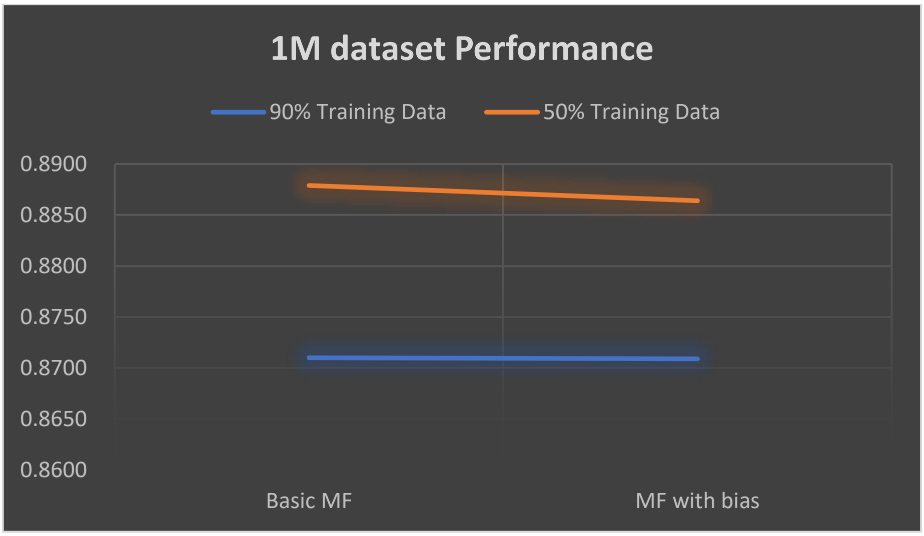
100k	user-user CF	item-item CF	Basic MF	MF with bias
RMSE(90% Training Data)	1.0389	1.1197	0.9221	0.9178
RMSE(50% Training Data)	1.2431	2.1143	0.993	0.9835



Based on the result, it can be found that performance of MF dominates both user-user and item-item KNN approachs but the later two's RMSE is still acceptable as being really close to 1. However, when the dataset is more sparser, MF algorithms still keep decent RMSE while user-user and item-item's performance is much worse.

Result of 1M dataset is showed in the table and corresponding figure below:

1M	Basic MF	MF with bias
90% Training Data	0.8710	0.8709
50% Training Data	0.8879	0.8864



MF with bias dominates Basic MF in both fraction 90% and 10% in terms of 1M dataset.

4.Conclusion

Collaborative Filtering(CF) algorithms are most commonly used in Recommender Systems(RS). CF algorithms nowadays face the problem with large dataset and sparseness in rating matrix. In this report, we have implemented our own user-user CF, item-item CF, basic MF and MF with bias to deal with the Collaborative Filtering challenges.

From the result, we can say that MF with bias is able to handle large dataset, sparseness of rating matrix and scalability problem of CF algorithm efficiently compared to the other algorithms we implemented. It dominates the basic MF and is much better than user-user CF and item-item CF when dealing with sparseness.

5.Extra function

To wrap our Recommender System up, the program can either output the top five movies for recommendation promoted by a specific user id or print the RMSE in terms of evaluation. Details can be found in "Readme".

6.Reference

1. Dheeraj kumar Bokde, Sheetal Girase. Role of Matrix Factorization Model in Collaborative Filtering Algorithm: A Survey, IJAFRC, 2014
2. Asela Gunawardana, A Survey of Accuracy Evaluation Metrics of Recommendation Tasks, Journal of Machine Learning Research 10, 2009
3. Michael D. Ekstrand, John T. Riedl and Joseph A. Konstan, Collaborative Filtering Recommender Systems, Foundations and Trends in Human–Computer Interaction,2010
4. Yehuda Koren, Robert Bell and Chris Volinsky, Matrix factorization techniques for recommender systems, Yahoo Research and AT&T Labs-Research, August 2009.
5. https://en.wikipedia.org/wiki/Matrix_decomposition
6. https://www.cse.unsw.edu.au/~mike/comp9417/ass2/RecSys.pdf
7. https://datascienceplus.com/building-a-book-recommender-system-the-basics-knn-and-matrix-factorization/
8. https://joyceho.github.io/cs584_s16/slides/mf-16.pdf
9. https://towardsdatascience.com/my-journey-to-building-book-recommendation-system-5ec959c41847