Recommender system using collaborative filtering

COMP9417, Assignment

Team members:
Hao Zhang, z5143832
Guangfei Liu, z5155853
Yiwen Xu, z5224340
Maowen Zhou, z5166834

Link of Google drive containing our programs and all files:
https://drive.google.com/drive/folders/1rkEVOzpF1PE5i9pMlhSjPQUAh3LqvpWS?usp=sharing

1. Introduction

Recommender system is a very popular and practical algorithm, which helps people find the items or products, for instance movies, that potentially suit their demands. A recommender system or recommendation system is a subclass of information filtering system that is able to predict the preference and rating that a user would give to an item [1]. In this assignment, we will focus on collaborative filtering that produces recommendations based on the knowledge of users' attitude to items by using "the wisdom of the crowd".
In this assignment, we will use part of MovieLens Datasets of size 100k to train and test the algorithms of collaborative filtering, including model-based and memory-based. For memory-based, we will use traditional methods such as item-based collaborative filtering by using KNN algorithm, and for model-based, we will focus on matrix factorization which projects users and items into a shared latent space, using a vector of latent features to represent a user or an item, such as SVD, this method would lead to low-rank matrices. Then multiplying these low-rank matrices together would generate a new matrix which fills the missing rating value of original matrix. Additionally, we tried to apply deep learning techniques to construct inner product on the latent features of users and items such as Multilayer Perceptron and Neural Matrix Factorization to explore more accurate and effective solutions in order to generate better recommendations.

2. Related work

Before starting this project, we had several discussions over the topic and started with searching and reading related papers as well as other materials, concluding that we would build our movie recommender system using collaborative filtering by applying KNN, SVD and Naive Bayes as well as MLP techniques. We divided our project task into four parts. The first one includes data collection, data cleaning and data processing. Our datasets are obtained from Kaggle MovieLens and we decided to choose 100K datasets, due to the fact that 20M MovieLens is too large. Therefore we have to try different ways to reduce the size of data as a result of limited memory size, we will talk about the problem of large datasets processing later. In the next part we will mainly focus on dealing with missing data and data sparsity problem. After finishing the previous part, we will select features such as movieId, userId and other tags in order to generate a pivot table as well as matrix factorization. Additionally, we will use datasets to train and test our model. Finally, we will generate recommendations based on users' inputs and movie names, then we will check and compare the accuracy of different collaborative filtering methods with conclusion in terms of our project tasks, problems and further study.

3. Solution

3.1 Memory-based collaborative filtering

3.1.1 KNN collaborative filtering algorithm

In this part, we are going to apply KNN algorithm to build a recommender system. KNN algorithm is called K nearest neighbor classification algorithm. The core idea of the KNN algorithm is that if the majority of K most neighbors of sample in the feature space belongs to a category, then the sample will be considered to belong to that category[2]. The basic idea of collaborative filtering recommendation algorithm is to introduce similar-rating or similar-interest movies to users, and we can see the processing figure 1 shown below. The basic tasks of the algorithm are data preprocessing and feature, item similarity calculation, KNN nearest neighbor selection, predict movie recommendation and conclusion and optimization.

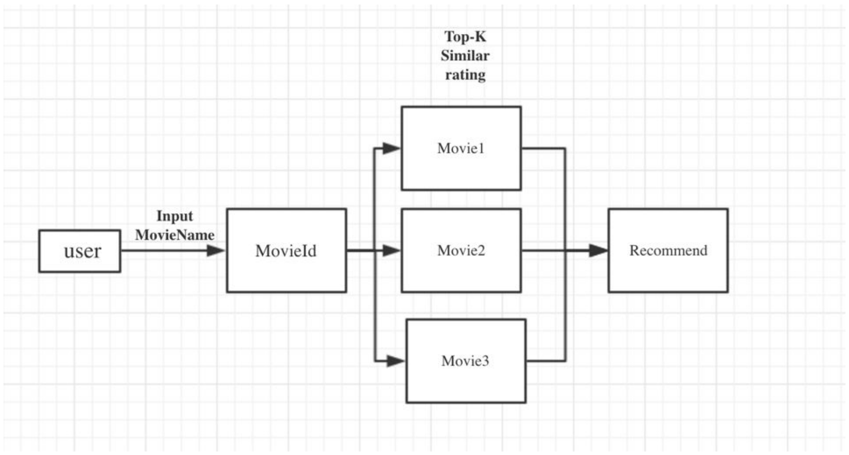


Fig.1. process of Item-based CF algorithm

3.1.1.1 Data preprocessing and feature

Before calculating similarity between different movies, we need to do data processing firstly, we will load movie and rating useful data, such as movieId, title, userId and rating score. And then, we can get the total number of rating score for different movies by different users, and now, we can process that missing value and filled with zero, and then pivot and reshape the datasets and then we need to take data sparse into consideration, so we can use csr matrix to get all valid userId, movieId and rating pairs.

3.1.1.2 Item similarity calculation

We can name the similarity of movie1 and movie2 as $sim(movie1, movie2)$, and then calculate the similarity by using Cosine Similarity and Euclidean distance.
Firstly, we can use Cosine Similarity to calculate the similarity between two movies by calculating the cosine of the angle between the two vectors:

$$sim(m1,\ m2) = \cos\left(\vec{m1}, \vec{m2}\right) = \frac{\vec{m1} * \vec{m2}}{|\vec{m1}| * |\vec{m2}|} \quad (1)$$

Secondly, we can use Euclidean distance to get the similarity between the two movies.

$$d(x,y) = \sqrt{\left(\sum (x-y)^2\right)} \qquad sim(x,y) = \frac{1}{1+d(x,y)} \quad (2)$$

But when we use this method, it has low accuracy.
Besides, we will use fuzzywuzzy method to match similar string or movie name, then return the highest similarity movieId and then use the id as the test tag for recommendation and also, we can return some similar movies based on names directly.

3.1.1.3 KNN nearest neighbor selection

After calculating the similarity $sim(m1,\ m2)$ between movie1 and movie2, we can select the highest top-k movies as the target movie1's neighbors, denoted as movie2, movie3 and movieK. And we can see the K nearest neighbors figure 2 shown below.

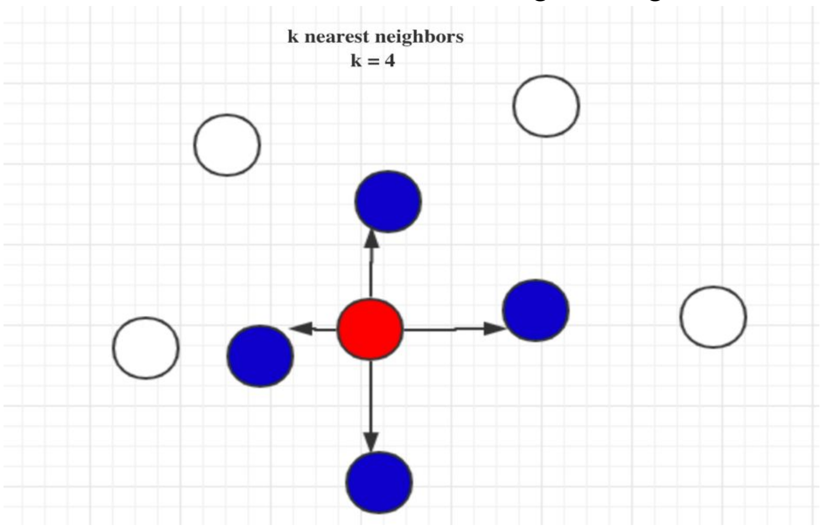


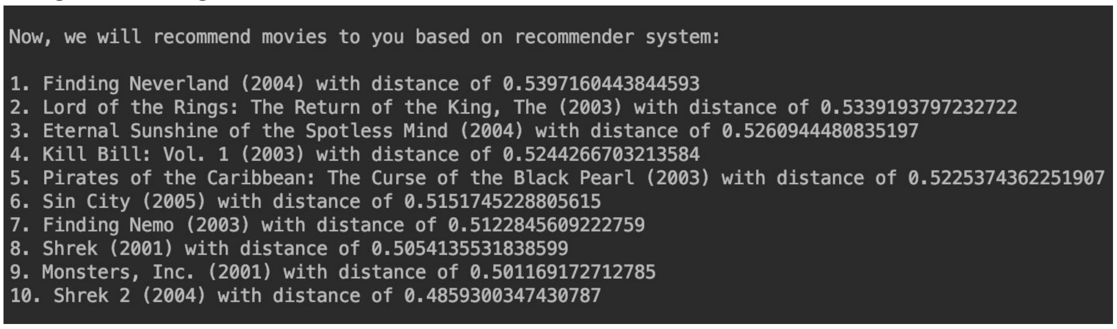
Fig.2. example for K nearest neighbors

In this part, we will import NearestNeighbors from sklearn to fit and predict datasets for movieId and rating. And we will set K value to be 20 or 25 and use brute algorithm and metric will be cosine and euclidean.

3.1.1.4 Predict movie recommendation

In this part, we are going to predict movies to users after getting input movie name and then get the most similar movie's Id so that using KNN algorithm to recommend top-K movies.
For example, when we input movie name such as big fish, and then we need to return top-10 similar movies by using recommend_movie() function, and we can get the

following results, figure 3:



```
Now, we will recommend movies to you based on recommender system:

1. Finding Neverland (2004) with distance of 0.5397160443844593
2. Lord of the Rings: The Return of the King, The (2003) with distance of 0.5339193797232722
3. Eternal Sunshine of the Spotless Mind (2004) with distance of 0.5260944480835197
4. Kill Bill: Vol. 1 (2003) with distance of 0.5244266703213584
5. Pirates of the Caribbean: The Curse of the Black Pearl (2003) with distance of 0.5225374362251907
6. Sin City (2005) with distance of 0.5151745228805615
7. Finding Nemo (2003) with distance of 0.5122845609222759
8. Shrek (2001) with distance of 0.5054135531838599
9. Monsters, Inc. (2001) with distance of 0.501169172712785
10. Shrek 2 (2004) with distance of 0.4859300347430787
```

Fig.3. example for movie recommender system

When we input favorite movie name, the best way is that the name of movie should belong to the movies dataset so that we can get the better recommendations.
And also, we can see that the distance between favorite movie and recommended movies, which is quite small for cosine similarity calculation. This is probably because there are too many zero values in the movie-user matrix. We can know that with too many zero values in our data, the data sparsity becomes a real issue for KNN model and the distance in KNN model starts to fall apart.
However, when we use 100K, there are fewer zero values in the dataset, which can relieve data sparsity problem significantly compared to 20M dataset. This is probably because the number of users for rating one movie and the number of movies rated by one user can be ensured, namely, we can ignore the situation that one user is new or just give one or two ratings to movies, or that one movie just get one or two ratings from users.
So, we finish the recommender system model based on KNN algorithm basically.

3.1.1.5 Conclusion and optimization

In this part, we are going to conclude the simple KNN collaborative filtering-based recommender system and the problem we met when we design the model and the ways we can use to optimize.
In this work, we find that using cosine similarity calculation would be better than Euclidean distance. And computing similarity using basic cosine measure in item-based case has one important drawback, the difference in rating scale between different users are not taken into account, so we can use adjusted cosine similarity to improve the performance of recommender system in the further study, which offsets this drawback by subtracting the corresponding user average from each co-rated pair. The adjusted cosine similarity formula[3] we can see figure4 below:

$$sim(i,j) = \frac{\sum_{u\in U}(R_{u,i}-\bar{R_u})(R_{u,j}-\bar{R_u})}{\sqrt{\sum_{u\in U}(R_{u,i}-\bar{R_u})^2}\sqrt{\sum_{u\in U}(R_{u,j}-\bar{R_u})^2}}.$$

Fig.4. formula of adjusted cosine similarity

In addition, there are some problems or bottleneck for KNN-based recommender system, such as cold start, data sparsity problem, and “Harry-Potter” problem (popular bias).
What is cold start? cold start concerns the personalized recommendations for users with no or few past history (new users). Providing recommendations to users with small past history becomes a difficult problem for CF models because their learning

and predictive ability is limited.
And in our work, we need to care about item-based cold start problem, which is that a new item is added to the system, it might have some content information but no interactions are present.
What can we do to solve or relieve cold start problem? When one new user or new item appears in our application, we can retrieve or read their interest from their choose or their our sign up account such as facebook, instagram, or we can require users to input some favorite movies firstly, or some interesting topics.
When it comes to “Harry-Potter” problem, namely, popular item bias problem, which refers to any item that is very popular. With a item-based collaborative filtering approach, such as Amazon's recommender system for other books, "People who bought this also bought", the system runs the risk of recommending Harry Potter to everyone just because most people have bought a Harry Potter book. But how can we handle this problem? when we calculate the similarity between the two movies, we can apply one punishing argument a (a∈[0.5, 1]) to restrict the popular item proportion or accounting, and then return one new similarity calculation formula based on cosine similarity calculation, we can see figure 5 below:

$$sim(m1, m2) = \cos(\overrightarrow{m1}, \overrightarrow{m2}) = \frac{\overrightarrow{m1} * \overrightarrow{m2}}{|\overrightarrow{m1}|^{\alpha} * |\overrightarrow{m2}|^{1-\alpha}}$$

Fig.5. formula of popular item punishing similarity calculation

Finally, data sparsity problem,there are a lot of missing data in our dataset, and what we can do is that we just fill zero into the missing position. Because the vast majority of entries in our data is zero, which explains why the distance between similar items or opposite items are both pretty large or similarity arguments for similar items are quite small. we plot the different rating scores count below, as figure 6 shown.

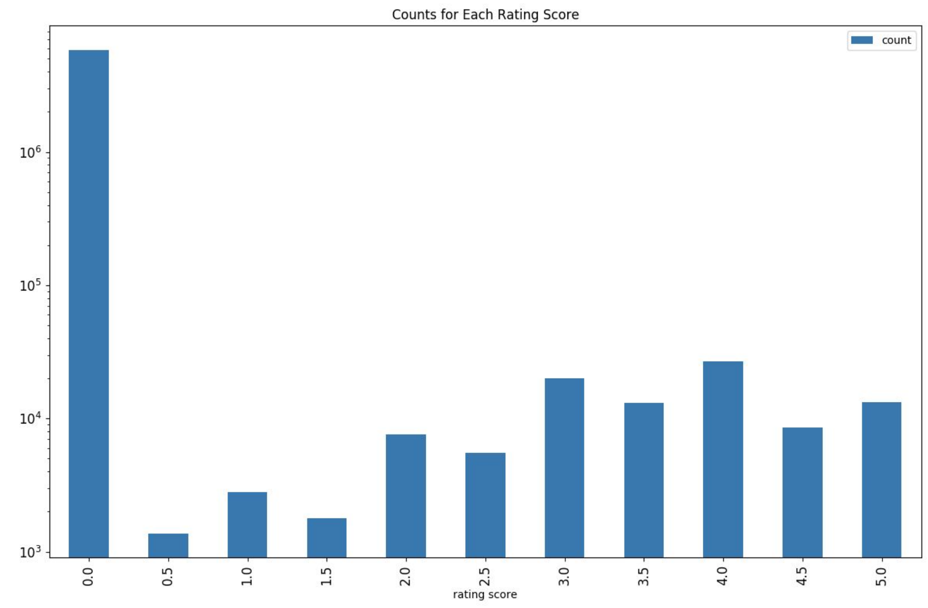


Fig.6.counts for each rating score

So, how can we solve the problem, we know that KNN algorithm is sensitive to missing data in our dataset, because it is involved in the distance between two items or two vectors, when one vector is missing, it will result in serious influence to our prediction and recommender system. So we can use the methods that do not need to calculate distance or distance (similarity) based algorithms to fit and predict data,

therefore, matrix factorization (MF) can be used to cope with data sparsity and possibly work even with such a sparse dataset.
we will talk about MF collaborative filtering method, such as SVD, in the following part.

3.2 Model-based collaborative filtering

3.2.1 SVD collaborative filtering algorithm

3.2.1.1 Model implementation:

- Step 1: generating training sample

Firstly, use dataframe.sample() to select some small parts from rating.csv file of 20M data and save them as sample_*.csv file;

- Step 2: generating a pivot table

Read each sample_*.csv to get a dataframe in memory and only choose the columns of userId, movieId and rating as rating_matrix;
Use pivot_table function of pandas to convert the rating_matrix into a new matrix(pivot_matrix), which uses userId as index, movieId as column and rating as value. In order to increase the precision of training the pivot_matrix should be normalized by minus each userId's mean for their evaluation of movies to get a normalized pivot_matrix;

- Step 3: matrix factorization

For the normalized pivot_table it can be decomposed by a module called svds from scipy.sparse.linalg to generate three low-rank matrices, respectively user-feature matrix, weights matrix, and movie-feature matrix;
During the processing of decomposing the latent feature number can be set as 20 and weights matrix should be converted into the diagonal form, such as the format:

$$A_{m\times n} = U_{m\times x} \times sigma_{x\times x} \times V^{T}_{x\times n}$$

where x is the number of latent features, m is the total number of users and n is the total number of movies

- Step 4: generating a predicted matrix

Multiply the three low-rank matrices to get a new matrix and then add each user's mean of rating for movies to get the predicted matrix;
To make the predicted matrix same with the pivot_matrix, add the index and column name into the predicted matrix in pandas function;

- Step 5: testing the predicted matrix in RMSE

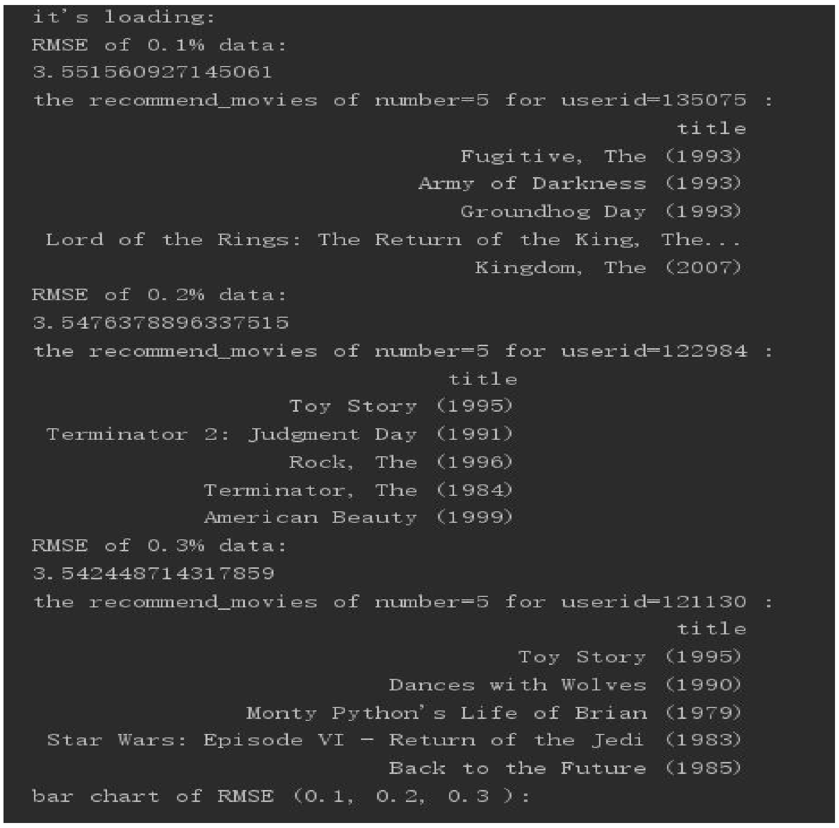
Now there are two matrices, one is original pivot_matrix losing some ratin value and another is predicted matrix filling the missing rating value;
Choose the users and movies which have corresponding rating value in original pivot_matrix;
Compare the actual rating value with predicted rating value between the two matrix,so we use RMSE as the evaluation standard, such as that:

$$RMSE = sqrt((\sum_{i}\sum_{j}(actual\_rating[i,j] - predicted\_rating[i,j])^2)/N)$$

where i and j are the index of user and movie, N is the total number of actual rating value.
Then each sample file has a value of RMSE and get an average error in all samples.

3.2.1.2 Running result

Model-based collaborative filtering:
sample_0.csv has 0.1% of the rating.csv and RMSE is 3.551560927145061;
sample_1.csv has 0.2% of the rating.csv and RMSE is 3.5476378896337515;
sample_2.csv has 0.3% of the rating.csv and RMSE is 3.542448714317859;



```
it's loading:
RMSE of 0.1% data:
3.551560927145061
the recommend_movies of number=5 for userid=135075 :
                                              title
                              Fugitive, The (1993)
                         Army of Darkness (1993)
                              Groundhog Day (1993)
 Lord of the Rings: The Return of the King, The...
                               Kingdom, The (2007)
RMSE of 0.2% data:
3.5476378896337515
the recommend_movies of number=5 for userid=122984 :
                              title
                  Toy Story (1995)
 Terminator 2: Judgment Day (1991)
                  Rock, The (1996)
            Terminator, The (1984)
            American Beauty (1999)
RMSE of 0.3% data:
3.542448714317859
the recommend_movies of number=5 for userid=121130 :
                                              title
                                  Toy Story (1995)
                         Dances with Wolves (1990)
               Monty Python's Life of Brian (1979)
 Star Wars: Episode VI - Return of the Jedi (1983)
                         Back to the Future (1985)
bar chart of RMSE (0.1, 0.2, 0.3 ):
```

Fig.7. running_result of SVD method

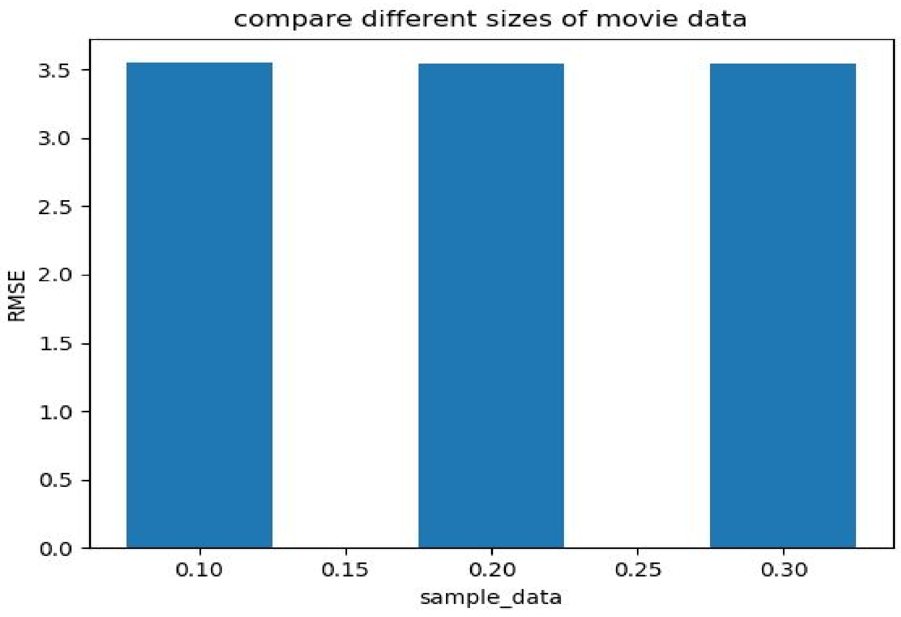


Fig.8. RMSE of SVD method

3.2.1.3 analysis

SVD algorithm can have lower training error, but this method needs high computation and memory for training, because of matrix transformation. Therefore, for the 20M of movie data we can only select a part of data to implement the SVD of latent features and compare how the training error is changing, which can prove if this method is more useful and efficient. Finally, because we only extract a part of data, we can only predict userid from sample data. Consequently, SVD method can improve training error but would user more property of computer.

3.2.2 Naive Bayes collaborative filtering

In addition, we can implement a collaborative filter recommendation system by Naive Bayes, as we know Naive bayes model is a generative model that is commonly used for classification. Also, use Naive bayes model, we can treat each movie as a feature and users like instances in order to infer the missing entries with a classification model. For our sparse data set that have challenge with hugh features process which means we need to use our model for collaborative filtering is that any feature (movie) can be the target class in collaborative filtering, and one also has to work with incomplete feature variables. These differences can be handled with minor modifications to the basic methodology of the naive Bayes model.

3.2.2.1 Data preprocessing and feature

In the beginning, we will consider rating csv file, because we need to rebuild user and movie table, each row group by user id, each column present movie id. Then each entry have rating with corresponding movie. We can consider the *u*th user, who has specified ratings for the set of movies M_u , which means that if the *u*th row has specified ratings for the first, second and 100th columns, we will have $M_u = \{1,2,100\}$. Then there have missing data where Bayes classifier needs to predict, we can note with r_{uj} that means user u ratings for movie j. Also, the rating we can consider as classes, thus we can fix rating as set $S = \{0,1,2,3,4,5\}$. Then we prefer to determine the probability that r_{uj} takes on any of these values conditional on the observed rating in M_u .

3.2.2.2 Predict unobserved rating

For each value i from set S, we have the following:

$$P(r_{uj} = S_i \mid Observed\ rating\ in\ \ M_u) = \frac{P(r_{uj}=S_i)\cdot P(Observed\ rating\ in\ \ M_u \mid r_{uj}=S_i\ )}{P(Observed\ rating\ in\ \ M_u)} \quad (3.1)$$

This expression using the well-known Bayes rule in probability theory:

$$P(A|\ B) = \frac{P(A)\cdot P(B \mid A\ )}{P(B)} \quad (3.2)$$

For determine the value S_i in the aforementioned expression, we can ignore the denominator and express the aforementioned equation cause the denominator probability we can consider as a constant of proportionality. Therefore, we can present the numerator part as:

$$P(r_{uj} = S_i)\cdot P(Observed\ rating\ in\ \ M_u \mid r_{uj} = S_i\ ) \quad (3.3)$$

Also, because right hand side probability is independent we have the following:

$$P(Observed\ rating\ in\ M_u \mid r_{uj} = S_i) = \prod_{k \in M_u} P(r_{uk} \mid r_{uj} = S_i) \quad (3.4)$$

In the above expression, for each k, we also need to consider unobserved rating, that lead a problem is the probability will be zero, and also lead to the above expression result to zero. Therefore, we handle unobserved rating, we need to consider use smoothing method for calculate these probabilities.

3.2.2.3 Smoothing method

There are several widely used smoothing methods such as Absolute Discounting, Laplace Smoothing, Good-Turing Estimation and Shrinkage, we have tried to implement our model with different smoothing method, and finally we still choose Laplace smoothing method, because this method can get better complexity and performance for our model. There are another reason for use smoothing method is our rating matrix is spares and the number of observed ratings is small. Therefore, the data-driven estimations may not remain robust. When we calculate the prior probability is unlikely to be robust if a small number of users have specified ratings for the corresponding movie. In order to handle this problem, we can use Laplace Smoothing method, if the user rating q movie, we can present as set q be the number of users that have respectively specified the ratings $M_1 \ldots M_u$ for jth movie, thus we can instead of estimating $P(r_{uj} = S_i)$ as $q_s / \sum_{t=1}^{j} q_t$, then we can add the smoothing parameter n:

$$P(r_{uj} = S_i) = \frac{q_s + n}{\sum_{t=1}^{j} q_t + j \cdot n}$$

3.2.2.4 Illustration of the Bayes method with a simple rating matrix

	Movie ID 1	Movie ID 2	Movie ID 3	Movie ID 4	Movie ID 5	Movie ID 6
User 1	4	2	?	2	2	3
User 2	2	5	2	4	4	0
User 3	3	3	3	?	1	0
User 4	?	4	?	5	0	?
User 5	1	1	2	?	?	2
User 6	5	4	4	3	3	0

We can use above simple rating matrix to explain our model, this rating matrix have 5 user and 6 movie id and the rating are drawn from set $S = \{0, 1, 2, 3, 4, 5\}$. We can consider the case in which we want to predict the rating of the two unspecified rating of user 1. Therefore, we need to consider calculate the probabilities of the unobserved ratings r_{13} taking on each of the value from set S, by using equation 3.3 we can obtain the following probability for the rating of movie id 3 by user 1:

$$P(r_{13} = 0)\cdot P(r_{11} = 0 \mid r_{13} = 0)\cdot P(r_{12} = 0 \mid r_{13} = 0)$$
$$\cdot P(r_{14} = 0 \mid r_{13} = 0)\cdot P(r_{15} = 0 \mid r_{13} = 0)\cdot P(r_{16} = 0 \mid r_{13} = 0)$$

For each value i of from set S, we need to compute as the above equation for it, then we can compare the probabilities result can pick the maximum one as our target rating. We can calculate the values of the individual probability of the aforementioned equation and we can also consider using Laplace smoothing method.

$P(r_{13} = 2) = (2+1 \ / 6+6) = 0.3$
$P(r_{11} = 2 \mid r_{13} = 2) = (1+1 / 6+6) = 0.16$
$P(r_{12} = 2 \mid r_{13} = 2) = (0+1 / 6+6) = 0.08$
$P(r_{14} = 2 \mid r_{13} = 2) = (0+1 / 6+6) = 0.08$
$P(r_{15} = 2 \mid r_{13} = 2) = (0+1 / 6+6) = 0.08$
$P(r_{16} = 2 \mid r_{13} = 2) = (1+1 / 6+6) = 0.16$

Upon substituting these values in the aforementioned equation, we obtain a probability of $P(r_{13} = 2 \mid \textit{Observed rating in } M_1)$, we still need to calculate other five probability and to find the top one result as following:

$$\begin{aligned} & argmax_{S_i}\ P(r_{uj} = S_i \mid \textit{Observed rating in } M_u) \\ = \ & argmax_{S_i}\ P(r_{uj} = S_i)\cdot \prod_{k\in M_u} P(r_{uk} \mid r_{uj} = S_i) \end{aligned}$$

3.2.2.4 Recommandation movie to user

After predict unobserved rating, we can generate a reliable result for movie recommendations, our model allows user input a favor movie name, then we can assume this user rating input movie in highest rating score which is 5, then we can find this movie in our integrity matrix, if there is another user rating this movie 5, we can output other high rating movies which are rating by this user. Otherwise, if there not have a match case, we can output top-k highest rating movie.

4. Conclusion

The study has referred to many machine learning methods, and our group select three main algorithms, namely KNN, SVD and Naive Bayes. Through specific implementation we find that each algorithm has its own advantages and disadvantages as written above in each section, no algorithm is perfect in classifying. Therefore different methods could be selected according to different requirements and situations. For example, if the requirements of users or clients demands more recommendation precision and data size is relatively small and data sparsity problem for KNN, SVD method may be a better alternative.

5. Further Study

To improve the performance of the algorithms, we have attempted to apply deep neural networks such as MLP techniques, while the remaining time is relatively short, therefore we have decided to continue with this in future learning, such as using Neural Matrix Factorization and other deep learning methods, such as Autoencoder, CNNs based, LSTM,.etc. we have read some papers about deep learning methods, but maybe because we are not familiar with the knowledge that we need to apply, so it is difficult to implement the whole algorithms for this assignment. In future studies, we can think more about some more advanced methods and also, we can design a complete recommender system website both from frontend and backend layers.

References

[1] Ricci, F., Rokach, L. and Shapira, B. (2019). Introduction to Recommender Systems Handbook.
[2]Cui, B. (2017). Design and Implementation of Movie Recommendation System Based on Knn Collaborative Filtering Algorithm. ITM Web of Conferences, 12, p.04008.
[3]YANG, X., YU, J., Turgun, I., QIAN, Y. and SUN, H. (2013). Collaborative filtering recommendation models considering item attributes. Journal of Computer Applications, 33(11), pp.3062-3066.
[4]James L 2018, The 4 Recommendation Engines That Can Predict Your Movie Tastes, accessed 17 July 2019, <https://medium.com/@james_aka_yale/the-4-recommendation-engines-that-can-predict-your-movie-tastes-bbec857b8223>
[5]Jae DS 2018, My Notes for Singular Value Decomposition with Interactive Code (feat Peter Mills), accessed 16 July 2019, <https://towardsdatascience.com/my-notes-for-singular-value-decomposition-with-interactive-code-feat-peter-mills-7584f4f2930a>