Movie Recommendation System based on Collaborative Filtering algorithm

Project Report

Jianan Liu

226004489

Movie Recommendation System based on Collaborative Filtering Algorithm

Jianan Liu
226004489

Abstract

Recommendation system can help people to find their interesting things and they are widely used with the development of electronic commerce. And collaborative filtering is the most successful technology for building recommendation system. In this project, I implement movie recommendation system based on three different collaborative filtering algorithms. They are traditional user based collaborative filtering algorithm, traditional item based collaborative filtering algorithm, slope one algorithm. Then I will compare these three algorithms based on their recommended results and implementation time complexity. Final, combined the results and other reference, some improvements are given.

key words *recommendation system, collaborative filtering, slope one*

Ⅱ. Introduction and Motivation

As the development of internet, electronic commerce, we live in a Big-data age when there are huge amounts of information arriving we can hardly deal with. Sometimes We are overwhelmed by information. Thus, the personalized recommendation system appears and develops, it can help us to deal with the problem 'information explosion' through filter technologies. And collaborative filtering is the most classic technology for building recommendation system. It has proved to be one of the most effective for its simplicity in both theory and implementation. There are two methods in collaborative filtering algorithm: User based collaborative filtering and Item based collaborative filtering. User based collaborative filtering is a way to find the target user's interesting items based on other users who have similar backgrounds like age, gender, interests, etc. so this method tries to find the user's neighbors and then combine the neighbor users' rating scores and similarity weights. Item based collaborative computes similarity between every two items, then it tries to find recommended items to the target user based on user's historical record like items/movies this user used/watched combined with this user's rating scores.

In this project, I will implement these two classic collaborative filtering algorithms to build a movie recommendation system. As a movie enthusiast, I am very excited to see what kind of movies will be recommended to me according to these two methods.

Besides, I will also implement another algorithm called slope one to build the movie recommendation, slope one is introduced in a 2005 paper by Daniel Lemire and Anna Maclachlan. It is also an item based collaborative filtering algorithm. But unlike the traditional one, this algorithm does not calculate the similarity between items. It uses linear regression model.

When we talk about recommendation system, we need to consider two problems: data's sparsity and scalability. Based on these two problems and recommended results from different algorithms, there are several improvements being provided.

Ⅲ.Methodology

1. User based collaborative filtering

   For user based collaborative filtering, the first task is to find the similar users of targeted user. Specifically, for this movie recommendation system, we need to find similar users of targeted user, then this system will recommend movies that similar users watched/like.

   A. user-item rating matrix

   Firstly, we need to input data, which is usual can be represented as a user-item rating matrix. The table is as following.

	Movie 1	Movie 2	…	Movie m
User1	R_{11}	R_{12}		0
User2	R_{21}	0		R_{2m}
…			R_{ij}	0
User n	R_{n1}	R_{n2}		R_{nm}

Where R_{ij} denotes the score of item j rated by an active user i. If user i has not rated item j, then R_{ij} =0

B. Measure similarity

Since we have data, we need to measure similarity between users, which is called user based similarity. There are several ways to calculate similarity. In this project, I use Cosine similarity.

$$S_{user\ u\ and\ v} = \frac{|num(U)\ \hat{}\ num(V)|}{\sqrt{num(U) * num(V)}}$$

in this formula, the numerator is the number of movies both user u and v watched and rated. Then we normalize it. The user based similarity in this project is defined by that the larger the number of films they both watched, the more similar they are.

Since user - movie rating matrix is a sparse matrix, which means that in many cases we have $num(U)\ \hat{}\ num(V) = 0$ for decreasing timing complexity of implementation, we can construct movie - user inverted rating table.

C. Selecting neighbor

With user based similarity, we can select the neighbors who will serve as recommenders. We use k-nearest neighbors, k is given first.

D. Producing recommendation

When the system finds the k similar users, the movies they watched will be our recommendation movies candidates, then according to ranks of these movies, we choose the first N movies as final recommendation movies.

$$Rank_{movie\ i \in candidates} = \sum_{v \in V} S_{user\ u\ and\ v} * R_{vi}$$

Where movie i is from movie candidates, user u is our target user, set V is all users who are target user's similar users and already watched and rated movie i. From this notation, it's easy to find that the rank of the movie is proportional to the value of similarity of the target user and similar user who rates this movie, is also proportional to the number of similar users who rates this movie and value of R_{vi}.

2. Item based collaborative filtering

A. Measure similarity

the data input is still user-item rating matrix. For item based collaborative filtering, we need to calculate item based similarity

$$S_{movie\ i\ and\ movie\ j} = \frac{|num(M_i)\ \hat{}\ num(M_j)|}{\sqrt{num(M_i) * num(M_j)}}$$

Where the numerator is the number of users who both watched M_i and M_j. Then we normalize it. In the denominator, $num(M_i)$ is the number of user who watch M_i, which measures the popularity of M_i.

B. Selecting neighbor

With item based similarity, according to target user's watched/rating movies, we use k-nearest neighbors, k is given first. Then for every movie in the target user's watched list, we choose its k similar movies as recommendation candidates. The

total number of recommendation candidates will be $k * num(U)$ where $num(U)$ is the number of movies target user watched.

C. Producing recommendation

When the system finds the recommendation candidates, then according to ranks of these movies, we choose first N movies as recommendation movies.

$$Rank_{movie\ i \in candidates} = \sum_{a \in A} S_{movie\ a\ and\ i} * R_{ua}$$

Where u is our target user, A is a movie set where all movies are from movie list of target user u and all movies in A includes movie i as their own k-nearest neighbors.

3. Slope one

slope one is introduced in a 2005 paper by Daniel Lemire and Anna Maclachlan. unlike the traditional collaborative filtering algorithm, this algorithm does not calculate the similarity between items. It uses linear regression model.

$$w = f(v) = v + b$$

if we know a set of training data $(v_i, w_i)_{i=1}^{n}$, we can use this linear model to minimize the sum of squares of prediction errors.

$$b = \frac{\sum_i (w_i - v_i)}{n}$$

based on this idea, define the deviation of movie i and movie j:

$$dev_{ij} = \frac{\sum_{u \in U_{ij}} (R_{ui} - R_{uj})}{num_{ij}}$$

Where $U_{ij} = U_i \cap U_j$, U_i is a set that represents all users who rate movie i, thus, U_{ij} is a set where all users rate movie i and movie j. num_{ij} represents the number of users in U_{ij}.

When we have dev_{ij}, we can get formula Weighted Slope One, which is the prediction score of target user to movie i. the higher score, the more chance this movie will be recommended to target user.

$$P(u)_i = \frac{\sum_{j \in I_u} (num_{ij} * (dev_{ij} + R_{uj}))}{\sum_{j \in I_u} num_{ij}}$$

Where I_u is a movie set where all movies are from target user's watched movie list. Once getting $P(u)_i$, soring it and choosing k movies with highest score as recommended movies.

Ⅳ. Results

now we can implement these three different algorithms on a movie data set, which is MovieLens dataset. MovieLens dataset is collected by GroupLens Research from University of Minnesota, Twin Cities. It includes scores that users rate movies. There are different sizes dataset. The one I use includes around 700 users, 9000 movies, 100000 rating records.

I use python to write codes to implement these three different algorithms. In order to directly see the quality of movies recommendation, I take myself as a target user. Then I rate around 30 movies I like from this dataset and input these new data. Below is the movie I rate.

Cinema Paradiso	5
legends of the fall	5
Forrest Gump	5
L.A. Confidential	5
No Country for old men	2
Big fish	5
Mission Impossible 3'	4
Mission Impossible 4'	5
Mission Impossible 5'	3
edge of tommorrow	4
Mission Impossible 1'	3.5
Mission Impossible 2'	4
Primal Fear	5
Space Jam	3.5
12 monkeys	3
Surviving the game	2.5
Top gun	4
Phone Booth	5
Lucy	3
X-man	3.5
Begin again	3.5
Kill bill 1'	4.5
Kill bill 2'	4
Boyhood	5
Once upon a time in america 12	5
lost in translation	4.5
Black hawk down	4
Trainspotting	4.5
Godfather2'	4
Godfather1'	4.5
Godfather3'	3.5

movies I rate

Below is original coding run result

```
Build user co-rated movies matrix ...
Build user co-rated movies matrix success!
Calculating user similarity matrix ...
Calculate user similarity matrix success!
user 672 :recommendation movies list based on UserCF
[('296', 10.030162073714 85), ('2571', 9.620786034021672), ('318', 8.524690597401
01), ('260', 7.944544036396677), ('5952', 7.8171018135722), ('2959', 7.641325014
868923), ('1089', 7.639779728114562), ('47', 7.558651544346793), ('1210', 7.4899
96342284136), ('50', 7.162939626856237)]

>>>
============= RESTART: D:\tamu\ecen765\project\ItemCF_NOtest.py =============
Similar movie number = 20
Recommneded movie number = 10
Load D:\tamu\ecen765\project\ml-latest-small\ml-latest-small\new.csv success!
Split trainingSet and testSet success!
TrainSet = 100035
TestSet = 0
Total movie number = 9066
Build co-rated users matrix success!
Calculating movie similarity matrix ...
Calculate movie similarity matrix success!
user 672 :recommendation movies list based on itemCF
[('4963', 13.446301090877803), ('608', 12.160524424601064), ('2571', 12.07005935
8893568), ('2959', 11.600478251047186), ('50', 10.21712908811277), ('260', 9.546
069311891033), ('5418', 8.974285367478288), ('4011', 8.926094432073928), ('6365'
, 8.911099063325327), ('377', 8.876850914339524)]
>>>
================ RESTART: D:\tamu\ecen765\project\slopeone.py ================
Recommneded movie number = 10
Load D:\tamu\ecen765\project\ml-latest-small\ml-latest-small\new.csv success!
Split trainingSet and testSet success!
TrainSet = 100035
TestSet = 0
Calculate movie deviation success!
user 672 :recommendation movies list based on SlopeOne
[('107559', 7.5), ('122888', 7.5), ('91690', 7.5), ('5301', 7.5), ('3879', 7.5),
 ('48791', 7.25), ('80839', 7.25), ('32289', 7.25), ('41573', 7.25), ('56633', 7
.0)]
>>>
```

In the movie recommendation result list, I get ten recommended movies. The first number is the movie id, the second number is this movie rank. In order to show the result more directly, I use two tables and write the recommended movies name down according to the movie id.

Below are the recommended results from item based collaborative filtering and user based collaborative filtering.

Ocean's Eleven (2001)
Fargo (1996)
The Matrix (1999)
Fight Club (1999)
The Usual Suspects (1995)
Star Wars: Episode IV - A New Hope (1977)
The Bourne Identity (2002)
Snatch (2000)

The Matrix Reloaded (2003)
Speed (1994)

results of item based collaborative filtering

Pulp Fiction (1994)
The Matrix (1999)
The Shawshank Redemption (1994)
Star Wars: Episode IV - A New Hope (1977)
The Lord of the Rings: The Two Towers (2002)
Fight Club (1999)
Reservoir Dogs (1992)
Seven (1995)
Star Wars: Episode VI - Return of the Jedi (1983)
The Usual Suspects (1995)

results of user based collaborative filtering

In these two tables, the order of movie shows the movie's rank. Thus, 'Ocean's Eleven' and 'Pulp Fiction' are No.1 movies that recommend to me based on item based collaborative filtering and user based collaborative filtering respectively. Personally, as a movie lover, I really like these two movies, especially 'Pulp Fiction' whose film editing is real amazing! For these two tables of 10 films, I already watched most of them, they are real great and perfectly suit my taste!

Also, in these two tables, the four movies by red mark shows that they are both recommended to me from these two algorithms, which to some extent shows the accuracy of these two algorithms.

Now below is the result from Slope one algorithm

Secretariat (2010)
Death of a superhero (2011)
Ben-hur (2016)
Friends with kids (2011)

Bite the Bullet (1975)
Art of war (2000)
Flicla (2006)
Ice princess (2005)
Family Stone (2005)
Shattered (2008)

Result of slope one

The result is quite different from the two algorithms. For movie lovers it is very straightforward to get this point. The recommended movies are less popular compared above. The reason I think is because of sparsity. Even users are very active, there are still large number of items without co-rated in a user-item rating matrix. Large levels of sparsity can lead to less accuracy. Thus, there exists a lot of less popular but not bad movies which only a few users watched and gave high rates. So according to the Slope one algorithm, these movies are very likely to recommended to users.

By analyzing this problem, we can improve slope one algorithm by combining user-based collaborative filtering. The process is that firstly we can use user-based collaborative filtering algorithm to find k similar users and then we use weighted slope one algorithm to predict the targeted user's rating to movies which only from K similar users. According to materials from other people and my understanding, this idea can work well, but my coding of this part still has some bugs when I run this improved slope one algorithm, so in this project there are no results showing about this slop one algorithm combining user-based collaborative filtering.

Another user's original coding results are below:

```
========== RESTART: D:\tamu\ecen765\project\ItemCF_NOtest.py ==========
Similar movie number = 20
Recommended movie number = 10
Load D:\tamu\ecen765\project\ml-latest-small\ml-latest-small\ratings.csv success!
Split trainingSet and testSet success!
TrainSet = 100004
TestSet = 0
Total movie number = 9066
Build co-rated users matrix success!
Calculating movie similarity matrix ...
Calculate movie similarity matrix success!
user 2 :recommendation movies list based on itemCF
[('380', 73.68600041584372), ('597', 65.65002148391238), ('344', 51.07052001831181), ('316', 35.10655587209551), ('595', 31.35747513686826), ('231', 29.99839299266203), ('434', 25.312224038170317), ('11', 17.904422534124016), ('318', 17.663831834274696), ('440', 16.59600379209757)]
>>>
========== RESTART: D:\tamu\ecen765\project\UserCF_NOtest.py ==========
Similar user number = 20
Recommended movie number = 10
Load D:\tamu\ecen765\project\ml-latest-small\ml-latest-small\ratings.csv success!
Split trainingSet and testSet success!
TrainSet = 100004
TestSet = 0
Building movie-user table ...
Build movie-user table success!
movie_user:
Total movie number = 9066
Build user co-rated movies matrix ...
Build user co-rated movies matrix success!
Calculating user similarity matrix ...
Calculate user similarity matrix success!
user 2 :recommendation movies list based on UserCF
[('318', 33.720904224036424), ('380', 32.7489170225259), ('595', 32.72572672419682), ('344', 28.211184360744312), ('316', 27.13610982699496), ('597', 27.086418414169103), ('440', 26.639578248152805), ('329', 26.105211128719525), ('434', 25.741736549175 99), ('231', 24.353336591779275)]


Recommended movie number = 10
Load D:\tamu\ecen765\project\ml-latest-small\ml-latest-small\ratings.csv success!
Split trainingSet and testSet success!
TrainSet = 100004
TestSet = 0
Calculate movie deviation success!
user 2 :recommendation movies list based on SlopeOne
[('109249', 8.5), ('39416', 7.0), ('7773', 7.0), ('5422', 6.5), ('4796', 6.333333333333333), ('4076', 6.333333333333333), ('4591', 6.333333333333333), ('1563', 6.333333333333333), ('1819', 6.333333333333333), ('5427', 6.333333333333333)]
>>>
```

V. Conclusions

From some target user's recommendation results, in many cases for a certain user, there are 40%~60% movies that are recommended by both user-based collaborative filtering and item-based collaborative filtering. And according to the experimental results and theoretical analysis, user-based collaborative filtering recommends users' movies based on similar users' watched movies, usual if some movies are very popular, a lot of similar users may watch it, so it pays more attention to socialization. Item based collaborative filtering recommends similar movies based on user's historical behavior, and pays more attention personalization. And as a movie lover, I am satisfied with the recommended results from both methods. As for slope one algorithm, because of dataset's sparsity, recommended movies are not so accurate. But if targeted user wants to try some less popular movies with not bad evaluation from a few people, slope one can be a choice.

For coding implementation result, user-based collaborative filtering is faster than the other two, the reason is that for the imported dataset, there are around 700 users and 9000 movies. We need to go through all users twice to compare and find the user similarity, which is O(700^2), however for the other two, we need go through all movies twice to find either item similarity or deviation of every two movies, which is

O(9000^2).

As for the future research direction, firstly I will focus on how to optimize these algorithms based on results and time complexity. I will fix bugs about slope one combing user-based collaborative filtering. Secondly, we can use some statistic method to evaluate the results, for example Mean Absolute Error and Root Mean Square Error, because of limited time, I didn't do it in this project. So this is another direction I will focus on in the future.

Ⅵ. Reference

[1] Songjie Gong. *Journal of Software*, VOL. 5, NO. 7, JULY 2010

[2] Liang Zhao, Naijing Hu, Shouzhi Zhang. *Journal of Computer Research and Development*, Vo l. 39, No. 8 Aug. 2002

[3] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5, 4, Article 19 (December 2015)

[4] Junfeng Zhou, Xian Tang, JingFeng Guo. *Journal of Computer Research and Development*, Vo l. 41, No. 10 Oct. 2004