Yahoo Music Recommendation

Kaggle Group: Sunset Old People Tour Group

1st Shiyao Li
Department of ECE
Stevens Institute of Technology
Hoboken, The United States
sli113@stevens.edu

2nd Yarong Liu
Department of ECE
Stevens Institute of Technology
Hoboken, The United States
yliu255@stevens.edu

3rd Wanying Cao
Department of ECE
Stevens Institute of Technology
Hoboken, The United States
wcao9@stevens.edu

I. Introduction

Nowadays, almost all the Music applications are trying to recommend different music to different users based on their listening records. In this project, we try to do the same thing like the companies of the applications. We use three methods to select three tracks from the given six tracks for 20,000 users and decide not to recommend the remaining three tracks.

II. Project Solution

Using what we've learned from the EE627 course, we try to define our own scoring rule and assign different weights to album scores, artist scores and genre scores so we can get our first group of scores. We also tried the matrix factorization and ensemble all the solutions we previously got to calculate the coefficients which we can use to get the true result.

A. Data Set Description

The whole data set is published by Yahoo on the website. The music data is stored in hierarchy structure. There's no user records for the tracks. The data set only contains the scores for the album, artist, type, etc. So we have to match the track which we'll consider to recommend and the album the artist it belongs to and we have to make our own rules to rate these tracks.

B. Data Integration

In the final project, we have 20,000 customers and six songs. From the given six tracks, we have to select three tracks labeled as "1" and recommend to each customer, and we label the remaining three songs as "0". Firstly, we try to integrate the data files, The first data file "testItem2.txt" shows the userID and the given six tracks, the second data file "trackData2.txt" shows the hierarchy structure of each track. From these data files, we designed Python file "Geth.py" to get hierarchy structure file "Ranking2.txt", the form of the content is shown as below:
UserID|trackID|Album|Artist|Genre1|Genre2|...
We also design the Python file "Rating.py" to add up each score and make the file links each user and a given track with the score.

C. Implementation Process

- Adjust the weight of each part's scores
  In our first method, among the score for albums, artists, genres, we try to figure out which part is more important for a user to decide if he or her will try a new track.
  - Preliminary Tuning
    Firstly, we assign the weight 0.9 for the album score, 0.7 for the artist and 0.4 for the genre score. We get the accuracy for 68% for the first solution so we must change the weight assignment. We continue to decrease the weight of the album scores and the artist scores. While we try the weight 1, 0.01, 0.00007 for each part, we get the accuracy for 0.865 and we make a huge progress. However, we can not make any progress after tuning the weights. Therefore.
  - Make the fair rule
    We reconsider the whole definition of the weights. We find that we can not give a track a better grade just because it has more genres than other songs and we should make the rule fair to the tracks which don't have any genre. So while we compute the genre scores, we take the average values and we calculate the average score again after getting the final scores. As for the tracks which don't have genre, we just take the average value of the album score and artist score. Therefore, the rule has become more resonabloe and we get the accuracy for 0.8741, which ranked 1st in the kaggle competition.
- Matrix Factorization
  Matrix factorization models map both users and items to a joint latent factor space of dimensionality F, such that user-item interactions are modeled as inner products in that space. This time we use the whole data set to recommend tracks for each user. Some of the albums or genres are never been graded, so using matrix factorization may help us get more scores and therefore we can sort the scores again to gain the new solution.

  - Split the Dataset
    In order to calculate such a huge data set, we decide to use pyspark to train the ALS model so we install

pyspark in the terminal. We split the whole data set into two parts: the tracks which are never graded and the tracks which have been graded. And then using "GetBlank.py" and "GetNoBlank" to get these two parts.

– Searching Best Paremeters
After importing pyspark and defining the RDD which contains the ID of tracks and users and the scores, we use the graded data set to train the ALS model and try to find the best rank value and iteration times by comparing the errors of each model.

```
Error_min = 1000
for rank in [2,5,10,15,20,25,30]:
    for itr in [2,5,10,15,20,22]:
        model = ALS.train(n,rank,itr)
        predictions = model.predictAll(n_test_data).map(lambda r: ((r[0], r[1]), r[2]))
        ratesAndPreds = n.map(lambda r: ((r[0], r[1]), r[2])).join(predictions)
        MSE = ratesAndPreds.map(lambda r: (r[1][0] - r[1][1])**2).mean()
        if MSE<Error_min:
            Error_min = MSE
            rank_min = rank
            itr_min = itr
        print("The rank is: ",rank)
        print("The iteration time is: ",itr)
        print("Mean Squared Error = " + str(MSE))
print("The best rank is: ",rank_min)
print("The best iteration time is: ",itr_min)
print("The minimum of the MSE is: ",Error_min)
```

Fig. 1. Search the best Parameters

```
Mean Squared Error = 0.00035900035020169067
The rank is:  30
The iteration time is:  5
Mean Squared Error = 0.00017552321476085956
The rank is:  30
The iteration time is:  10
Mean Squared Error = 9.731756609928635e-05
The rank is:  30
The iteration time is:  15
Mean Squared Error = 0.00010169450396150961
The rank is:  30
The iteration time is:  20
Mean Squared Error = 7.276481302868509e-05
The rank is:  30
The iteration time is:  22
Mean Squared Error = 7.911103895655716e-05
The best rank is:  20
The best iteration time is:  22
The minimum of the MSE is:  6.616675374040241e-05
```

Fig. 2. The result of the searching

– Make Predictions
After getting the best model, we try to use the function "PredictAll" to get the tracks which are never graded. Now using matrix factorization helps us grade some tracks that we can not grade before. It helps us rationally choose three songs for users. We integrate these results and the original results into a new solution by running "Recommend.ipynb".

After getting the new solution we submitted on Kaggle and we get the accuracy for 84.12%. It is obvious that matrix factorization can not help us get a better solution even though we re-grade some tracks for users.

- Ensemble Solutions
The goal in this method is to find a perfect matrix A(size:

```
In [50]: nb = model.predictAll(b_data)

In [52]: blk = nb.sortByKey().collect()
         blk

Out[52]: [Rating(user=199810, product=242681, rating=4.328283794694013),
 Rating(user=199814, product=52519, rating=-11.554977178834742),
 Rating(user=199815, product=118841, rating=40.370491944524242),
 Rating(user=199815, product=129866, rating=-19.487839627792447),
 Rating(user=199816, product=274758, rating=-10.332824411537674),
 Rating(user=199819, product=214280, rating=1.9298185203833036),
 Rating(user=199820, product=292341, rating=27.081081300111368),
 Rating(user=199821, product=51933, rating=8.3794897773368525),
 Rating(user=199821, product=42811, rating=1.5438507702141564),
 Rating(user=199822, product=185734, rating=-0.8255864726205624),
 Rating(user=199826, product=179298, rating=-0.0698098693636494),
 Rating(user=199827, product=92599, rating=44.641063575815835),
 Rating(user=199838, product=46697, rating=5.163624825133493),
 Rating(user=199839, product=266143, rating=-0.017225807150921417),
 Rating(user=199844, product=140229, rating=-2.7273598398648478),
 Rating(user=199844, product=77700, rating=-0.09972106953076021),
 Rating(user=199845, product=189985, rating=-31.345164178645 37),
 Rating(user=199847, product=63072, rating=-26.685313760700605),
 Rating(user=199853, product=256134, rating=-6.405145672738552),
```

Fig. 3. Grade the songs which are never graded

```
TrackID,Predictor
199810_208019,0
199810_74139,0
199810_9903,0
199810_18515,1
199810_242681,1
199810_105760,1
199812_276940,0
199812_29189,0
199812_211361,0
199812_142408,1
199812_130023,1
199812_223706,1
199813_20968,0
199813_111874,0
199813_184173,0
199813_188441,1
199813_79640,1
199813_21571,1
199814_52519,0
199814_122375,0
199814_122429,0
199814_232332,1
199814_189043,1
199814_262193,1
199815_129866,0
```

Fig. 4. The new solution by matrix factorization

$n \times 1$) which is equal to the totally correct matrix Y by multiplying matrix R. (ie : $S \times A = Y$)This linear regression can help us to get more close to the total correct result. The implementation steps are shown as follows.

– Data Processing
Combine n solutions to build matrix S(size:120000$\times$ n). We can replace all "0" in R to "-1" easily in Excel.

	A	B	C	D	E	F	G	H
1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1
3	-1	-1	-1	-1	-1	-1	-1	-1
4	-1	-1	1	-1	-1	-1	-1	-1
5	1	1	-1	1	1	1	1	1
6	-1	-1	-1	-1	-1	-1	-1	-1
7	1	1	1	1	1	1	1	1
8	1	-1	1	1	1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	-1
10	-1	1	1	1	1	1	1	1
11	-1	-1	-1	-1	-1	-1	-1	-1
12	1	1	-1	-1	-1	-1	-1	-1
13	1	1	-1	-1	-1	-1	-1	-1
14	1	-1	-1	-1	-1	-1	-1	-1
15	1	-1	1	1	1	1	1	1
16	-1	-1	-1	-1	-1	-1	-1	-1
17	-1	1	1	1	1	1	1	1
18	-1	1	1	1	1	1	1	1
19	-1	-1	-1	-1	-1	-1	-1	-1
20	-1	-1	-1	-1	-1	-1	-1	-1
21	1	1	1	1	1	1	1	1
22	1	1	1	-1	1	1	1	1
23	1	1	1	1	1	1	1	1
24	-1	-1	-1	1	-1	-1	-1	-1
25	-1	-1	-1	-1	-1	-1	-1	-1
26	-1	-1	-1	-1	-1	-1	-1	-1

Fig. 5. Replacement in Excel

- Matrix operation
  Using the formula: $A = (S^T \times S)^{-}1 \times S^T \times Y$, we can compute matrix A. Since we do not have the original answer, we must get Y by computing $Y = K \times (2 \times correctrate - 1)$, then get the answer A.
- Computing Y
  Using formula: $S \times A = Y$ to compute matrix Y.
- Data integrate
  Actually matrix Y contains many positive and negative values which range from -1 to 1, and the number of negative values are not equal to the number of positive values. So sort all the value in the Y and label three tracks with higher values among the given six tracks as "1" and recommend to users.
- Get the solution
  During this process, Firstly we choose 8 solutions to combine matrix R and compute Y. We get the accuracy for 0.87041. Thinking about the reason of this result, the matrix S may be not full rank and there is little difference between 8 solutions. Secondly, we choose 5 solutions to combine matrix R and follow the same steps before. Unfortunately, the result has no improvement, we still got the accuracy for 0.87041, which is the same result as what we get from tuning the weights.

III. Results and conclusion

After applying all the methods we learned from the course, we get different results as follows:

TABLE I
THE COMPARISON OF EACH METHOD

Method	Recommendation Accuracy
Tuning Weights	87.041%
Matrix Factorization	84.12%
Ensemble Solutions	87.041%

So based on our implementation, we find that which matters most is the album scores and the genres scores take only a little weight in the whole calculation. Matrix Factorization can help us get more scores, however, it can not help with getting better accuracy. In the last method, some of our solutions are really similar results so the accuracy just stays the same. In future, we'd like to get more new better solutions by tuning the weights and we'll ensemble it with our previous solutions to make better predictions.