Million Song Dataset Challenge

Fengxuan Niu, Ming Yin, Cathy Tianjiao Zhang

1 Introduction

Million Song Dataset (MSD) is a freely available collection of data for one million of contemporary songs (http://labrosa.ee.columbia.edu/millionsong/). The Million Song Dataset Challenge (MSDC) is a large scale, music recommendation challenge posted in Kaggle, where the task is to predict which songs a user will listen to and make a recommendation list of 500 songs to each user, given the user's listening history. We *1)* implement both user-based and item-based collaborative filtering algorithm to learn the historical data and make recommendations, and *2)* improve and mix these two models to get a better result.

2 Data Set

The dataset for MSDC is the Taste Profile Subset of MSD (http://labrosa.ee.columbia.edu/millionsong/tasteprofile), which consists of more than 48 million triplets (user, song, play count) gathered from 1M users.

The dataset is then split in two: 1) the train set contains a little over a million users, full history released (available on the MSD website), 2) the validation and test sets combined contain 110k users, half of their history released. Our recommender system will need to predict unreleased history.

The data looks like this:

User ID	Song ID	count
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOAKIMP12A8C130995	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOAPDEY12A81C210A9	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBBMDR12A8C13253B	2
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBFNSP12AF72A0E22	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBFOVM12A58A7D494	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBNZDC12A6D4FC103	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBSUJE12A6D4F8CF5	2
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBVFZR12A6D4F8AE3	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBXALG12A8C13C108	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBXHDL12A81C204C0	1
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBYHAJ12A6701BF1D	1

The users-by-songs matrix is very sparse as the fraction of observed non-zero entries in the matrix is only 0:01. The full MSD dataset contains richer information about songs, such as lyrics and sound tracks.

3 Method

In general, there are two types of collaborative filtering: Memory-based algorithm and Model-based algorithm. Memory-based algorithms recommend items based on similarity between users or items (here it's songs). User-based model make prediction for a certain user based on its similar users. The intuition is that if user A has similar taste to another user B then A will be likely to enjoy the songs that B listened. Alternatively, item-based model make recommendation by finding similar songs to the songs in certain user's listening history. Here we implement

memory-based algorithm to this MSDC.

3.1 User-based Model

Common similarity measures between two users are Pearson correlation coefficient and cosine-based similarity. Since the data is very sparse, we prefer to use cosine-base similarity for this problem.

$$w(u_x, u_y) = \frac{\sum_{i \in S} X_i Y_i}{\sqrt{\sum_{i \in S} X_i} \sqrt{\sum_{i \in S} Y_i}}$$

where S is the set of songs, X and Y are the listening history(whether the user listened song i before or not) of user x and user y. The score of song s_k for user u_a is calculated as: $r_{u_a,s_k} = \sum_{i \neq a} I(u_i, s_k) w(u_a, u_i)$where $I(u_i, s_k)$ indicates whether user i listened to song k. Memory-based algorithms are simple and has little training phrase, but predicting a user history is very costly.

3.2 Item-based Model

Similarly to the user-based model, the item-based model has expression as follows,

$$w(s_x, s_y) = \frac{\sum_{i \in U} X_i Y_i}{\sqrt{\sum_{i \in U} X_i} \sqrt{\sum_{i \in U} Y_i}}$$

where U is the set of users, X and Y are whether the song X and song Y is listened by user i before or not, respectively. The score of song s_k for user u_a is calculated as:

$$r_{u_a,s_k} = \sum_{i \neq k} I(u_a, s_i) w(s_i, s_k)$$

where $I(u_a, s_i)$ indicates whether user u_a listened to song s_i.

3.3 Local Scoring Rule

We found that sometimes a popular song can get a higher score for a jazz music fan than a nice jazz song, which is perfect choice for the user. This kind of situation results from that the final recommendation for the pop song may be computed by aggregating a large number of low similarity scores (many people who this pop song don't have the same taste with jazz fan). But the final recommendation for the jazz song may be calculated by only several large similarity scores (people who share similar listening preference with the jazz fan).

To avoid the above situation, it is very important to determine how much each individual score component influences the overall score. We want to drop the very small similarity scores, while relatively emphasizing the high ones.

Therefore, we use a f(w) to measure the importance of the individual similarity score:

$$f(w) = w^q$$

3.4 Generalized Cosine Similarity

Instead of using regular cosine similarity, we consider using a general form,

$$w(u_y) \; for \; u_x = \frac{\sum_{i \in S} X_i Y_i}{(\sum_{i \in S} X_i)^{\rho} (\sum_{i \in S} Y_i)^{1-\rho}}$$

When we calculate recommendation for u_x, we are actually using all the similarity scores of u_x with respect to all other users. Therefore, $\sum_{i \in S} X_i$ term is the same for all the similarity scores. To emphasize the influence by u_y, we try to decrease the value of parameter ρ.

3.5 Mixed model

3.5.1 Mix Popular Song to the Recommendation

For each user, we find that in the recommendation list of 500 songs the least x%(x=5,10,15) scores are so small that these recommendation may not be meaningful. The user have a very low probability to listen these low score songs. So it could be better to just recommend the popular songs to replace the x% songs with the least scores.

3.5.2 Linear Combination of User-based and Item-based Models

Using User-based and Item-based models, we get two different scores of song s_k for user u_a, r_{u_a,s_k}^{user} and r_{u_a,s_k}^{item}. And then get two different recommendation lists of all songs for each user, which is ordered by the value of scores.

Combining the results of two models, we can probably get a better prediction. Our first try is to combine the two models linearly with weight α and $1-\alpha$, respectively, as follows,

$r_{u_a,s_k}^{new} = \alpha * normalized\ r_{u_a,s_k}^{user} + (1-\alpha) * normalized\ r_{u_a,s_k}^{item}$Here, due to different scales of r_{u_a,s_k}^{user} and r_{u_a,s_k}^{item}, we first normalize the scores and then combine them linearly to get the new scores. The 500 songs with the largest new scores will be chosen to form the new recommendation list.

3.5.3 Aggregation of User-based and Item-based Model

Except combining the scores, we try to merge the two different recommendation lists of 500 songs from two models.

We first pick up the songs that exist in both user-based and item-based recommendation lists. Then use x% recommendations from Item-based list, and (1-x)% from User-based list for the rest of the final recommendation list.

3.5.4 Stochastic Combination of User-based and Item-based Model

This model is a variation of the previous one. After picking up the songs that exist in both user-based and item-based recommendation lists, we randomly choose the rest of final recommendations from the item-based list with probability p and from the user-based list wit probability 1-p.

4 Evaluation

In the Kaggle competition, the truncated mAP (mean average precision) is used as the evaluation metric. The mAP metric emphasizes the top recommendations, and is commonly used throughout the information retrieval literature.

$M_{u,i}$ is 1 if user u listened to item i, and 0 otherwise. Let y denote a ranking over items, where y(j) = i means that item i is ranked at position j. For any $k < \tau$, the precision-at-$k(P_k)$ is the proportion of correct recommendations within the top-k of the predicted ranking:

$$P_k(u,y) = \frac{1}{k}\sum_{j=1}^{k} M_{u,y(j)}$$

For each user, we now take the average precision at each recall point:

$$AP(u,y) = \frac{1}{n_u}\sum_{k=1}^{\tau} P_k(u,y) * M_{u,y(k)}$$

where n_u is the smaller between τ and the number of user u's positively associated songs. Finally, averaging over all m users, we have the mean average precision:

$$mAP = \frac{1}{m}\sum_{u} AP(u,y_u)$$

5 Result

Due to the large scale of the data and the high time complexity of the algorithm, we have adopted multiple approaches to promote the efficiency of the experiments with limited computation resources. First, we used MapReduce to distribute our works onto multiple machines. Each mapper worker keeps a copy of training data in memory with multiple threads taking testing data and producing recommendations. The reducers compute the AP and then aggregate them to generate the mAP evaluation.

Benchmark	mAP
Recommendation by Popularity	0.02262

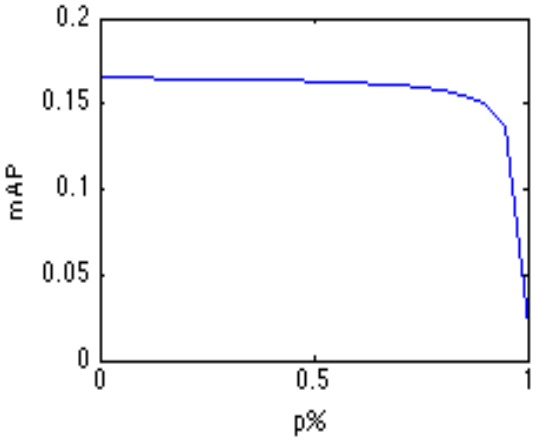
5.1 User-based and Item-based Model

Model	Best mAP	Parameter for the best result
User-based	0.126100586965	q=5, ρ =0.3
Item-based	0.165176611821	q=3, ρ =0.15

5.2 Mix Popular Song to the Recommendation

Cutting last p% and substitute with the most popular songs:

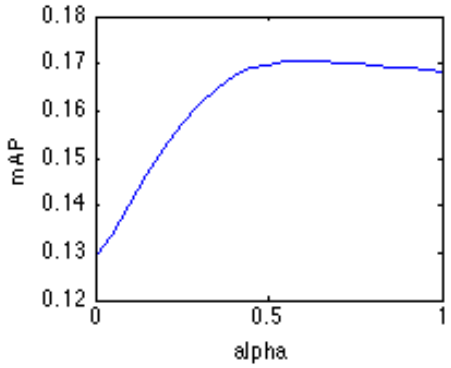
P%	mAP	P%	mAP
0	**0.165176574**	0.55	0.162789096
0.05	0.165054094	0.6	0.162276481
0.1	0.164939915	0.65	0.161633366
0.15	0.164825657	0.7	0.160753245
0.2	0.16468442	0.75	0.159503314
0.25	0.164534334	0.8	0.157723347
0.3	0.16436705	0.85	0.154964397
0.35	0.164160106	0.9	0.149925018
0.4	0.163919627	0.95	0.136753472
0.45	0.163590651	1	0.019490785
0.5	0.163209619		



5.3 Linear Combination of User-based and Item-based Models

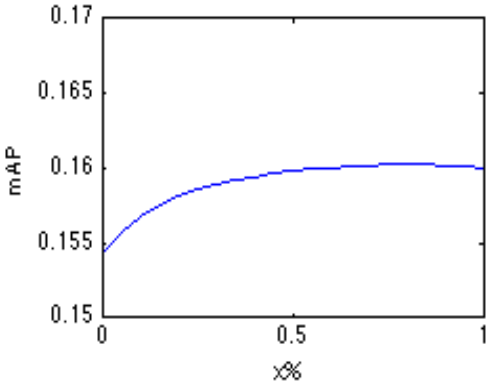
$$r_{u_a,s_k}^{new} = \alpha * normalized\ r_{u_a,s_k}^{user} + (1-\alpha) * normalized\ r_{u_a,s_k}^{item}$$

alpha	mAP	alpha	mAP
0	0.129300587	0.55	0.170479949
0.05	0.133443227	**0.6**	**0.170506752**
0.1	0.140024626	0.65	0.170357485
0.15	0.146444031	0.7	0.170151429
0.2	0.152188236	0.75	0.169863574
0.25	0.157173406	0.8	0.169474359
0.3	0.161272195	0.85	0.169223911
0.35	0.164629404	0.9	0.168975251
0.4	0.167224766	0.95	0.16872603
0.45	0.1689843	1	0.168376612
0.5	0.169838796		



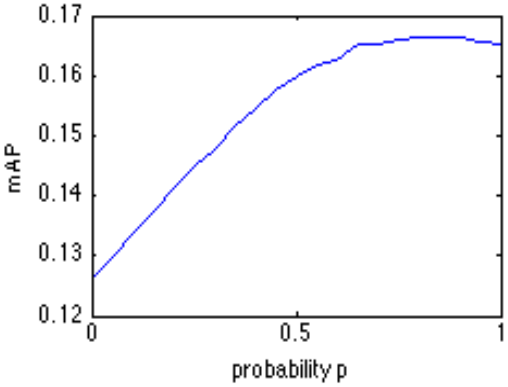
5.4 Aggregation of User-based and Item-based Model

X%	mAP	X%	mAP
0	0.154307682	0.55	0.1599339
0.05	0.155749975	0.6	0.160044142
0.1	0.15682858	0.65	0.160133312
0.15	0.157570064	0.7	0.160207329
0.2	0.158128412	0.75	0.160258716
0.25	0.158578223	**0.8**	**0.160290188**
0.3	0.158923803	0.85	0.160263322
0.35	0.15921611	0.9	0.160219189
0.4	0.159444759	0.95	0.160126948
0.45	0.15963592	1	0.160071507
0.5	0.159802653		



5.5 Stochastic Combination of User-based and Item-based Model

Probability	mAP	Probability	mAP
0	0.126100957	0.55	0.161815041
0.05	0.129782884	0.6	0.162541884
0.1	0.133422744	0.65	0.165150909
0.15	0.137437142	0.7	0.165041273
0.2	0.141183793	0.75	0.165842722
0.25	0.144885668	0.8	0.166350983
0.3	0.147523216	0.85	0.166336533
0.35	0.151396975	**0.9**	**0.166409741**
0.4	0.154710693	0.95	0.165588379
0.45	0.157734663	1	0.165176574
0.5	0.159750357		



5.6 Conclusion

Item-based model generally generates better results than user-based model. Mixing popular songs into recommendation is not helpful. One possible explanation is that the model itself already takes account into the popularity. Aggregation does not perform well possibly because the existence in recommendation lists from both models takes too much weight without considering the scores. Stochastic method takes certain portion from each list choosing more from the preferred item-based model also taking top scored ones from user-based model. The linear combination generates the best result (0.170506752) of all. In the Kaggle leaderboard, our mAP result is in the third place among 153 teams.

References

[1] http://www.kaggle.com/c/msdchallenge

[2] http://labrosa.ee.columbia.edu/millionsong/

[3] http://en.wikipedia.org/wiki/Collaborative_filtering

[3] T. Bertin-Mahieux, D.P.W. Ellis, B. Whitman, and P. Lamere. The million song dataset. In ISMIR 2011: *Proceedings of the 12th International Society for Music Information Retrieval Conference, October 24-28, 2011, Miami, Florida*, pages 591–596. University of Miami, 2011.

[4] F. Aiolli, A Preliminary Study on a Recommender System for the Million Songs Dataset Challenge Preference Learning: Problems and Applications in AI (PL-12), ECAI-12 Workshop, Montpellier