Music Recommendation with Implicit Feedback and Side Information

Shengbo Guo
Yahoo! Labs
shengbo@yahoo-inc.com

Behrouz Behmardi
Criteo
b.behmardi@criteo.com

Gary Chen
Vobile
gary.chen@vobileinc.com

Abstract

Music streaming services (e.g., Spotify, Pandora, Douban.fm) provide personalized music recommendations given users' historical interactions such as labeling a track as their favorite. The main challenge for music personalization is serendipity, i.e., how one can learn users' preferences for recommending music that can lead to pleasant surprise to the users. In this paper, we cast music recommendations as a matrix factorization problem, and we extend the known matrix factorization model with item (e.g., music tracks, artists) side information under a principled framework. In particular, we enhance user preferences over artists by integrating artist biographies. To efficiently learn the model parameters, we propose to use distributed alternating least squares implemented in the MapReduce framework. Empirically, we demonstrate on real-world data sets that the proposed models can improve the performance evaluated in terms of commonly used metrics including precision and recall.

1 Introduction

Music personalization has been an active research topic for the past decade [4], leading to rich literature and widely used music streaming services such as Spotify, Pandora, Douban, etc. The active research towards this topic can further be reflected by the recent KDD-Cup'11 Challenge and some of the publicly available data sets including Yahoo! Music Dataset [2]. Despite of the recent advances for music personalization, there are two challenges for one to deliver (nearly) optimal music recommendations to users: (1) how can one personalize music recommendation, particularly for items which are not very popular (serendipity), and (2) how can one develop scalable algorithms in order to meet the need for real-world streaming services.

To put these challenges in the context, we first describe the motivation of our work. The motivation is generated from Samsung Music Hub[1], a personalized music service for smart phone users. This app allows one to interact with different types of entities including albums, artists, and songs. Data obtained from this App allows us to obtain the relation – (user, album, count), the number of times that a user shows interests in an album [2]. Note that here the interests mean that a user clicks an album either for listening to a song in the album or for purchasing the album. Given this type of observations, the problem here is to predict a ranked list of albums that a user will like to listen or purchase with high probability. Essentially this is a matrix factorization problem with the objective of predicting the missing values in the user-album count matrix.

[1]http://www.samsung.com/us/samsunghub/
[2]the characteristics of the dataset is available in the experimental part

One widely used approach [3] for such matrix factorization problem is to build latent factor models in order to find like-minded users and items, and further to consider user feedback as implicit rather than explicit. Despite of the promising performance as originally shown in applications including the TV program recommendations, one difficult associated with this approach is how one can make recommendations for non-popular items which is called *long-tail*.

In this paper, we propose new models based on [3] for music recommendation by integrating side information, particularly for making recommendation in the long-tail. Our new models aim at learning user-artist preference represented in terms of a tuple – (user, artist, count), i.e., the number of times that a user shows interests in an artist, but we can integrate side information from artist biography to enhance the quality of prediction. The idea behind integrating artist biographies is essentially to exploit the artist similarity reflecting by the semantics (e.g., culture background, famous albums) in the text descriptions of biographies. To efficiently learn the model parameters, we propose to use the alternating least squares (ALS) implemented in the MapReduce framework. To this end, our main contributions in this paper are in threefold:

1. Apply matrix factorization on predicting two levels of user preferences – (user, album, count) and (user, artist, count), which correspond with user preference over albums and artists, respectively. Further we show that models with *implicit* feedback achieve the best performance evaluated with precision and recall based metrics.
2. Propose to integrate the side information for making recommendation, particularly useful for recommendations in the long-tail.
3. Show that the (nearly) real-time model based on the co-occurrence can achieve comparable results with matrix factorization.

In the following sections, we first introduce some basline models for matrix factorization with implicit feedback based on [3], and then we formulate the proposed models for integrating the side information into the matrix factorization framework. After briefly describing the ALS algorithm, we present our empirical results in the real-world data set obtained from the Samsung Music Hub.

2 Models and Algorithm

We consider two collaborative filtering (CF) algorithms: $1)$ the neighborhood approach (CF-Neighborhood) and $2)$ the latent factor model (CF-MF). We also consider a non-personalized algorithm based on top-popular items as a baseline (TopPop).

2.1 Top-Popular Baseline (TopPop)

As its name indicates, Top-Popular recommends to each user a predefined, fixed list of most popular items, i.e., items received the largest number of actions (e.g., likes, thumbs-up, favorite). Note that in this approach all users are presented with the same set of recommended items. Top popular baseline commonly used as a baseline in most recommended system, partially due to the user cold-start problem.

2.2 Neighborhood CF (CF-Neighborhood)

Neighborhood based CF relates similar users or similar items based on historical user behavioral data. For example, two items are considered similar if the same set of users have acted (e.g., rating, liking, clicking) similarly on them. We use normalized item similarity CF as follows:

$$\mathrm{sim}(i, j) = \frac{\text{num. of users click on both items } i \text{ and } j}{\text{num. of users click on item } i} \tag{1}$$

Item similarity represented in 1 is preferred to user similarity due to the scalability (usually more user than item). Moreover, the normalization term of the co-occurrence similarity by the total number of actions for each item removes the bias toward highly rated items.

2.3 MF with Implicit Feedback (CF-MF)

The latent factor model that we use in the paper is based on the one proposed in [3]. Denoted by $\boldsymbol{Y} \in \mathbb{R}^{M\times N}$ the partially observed counting matrix where M is the number of users and N is the number of items, and $\boldsymbol{Y}_{ij}$ is the number of counts that user i listens to item j. We propose to introduce latent factors to represent users and items, and denoted by $\boldsymbol{U} \in \mathbb{R}^{M\times K}$ and $\boldsymbol{V} \in \mathbb{R}^{N\times K}$ the latent factor matrix corresponding to users and items, respectively.

Given the above notations, we can define the optimization problem as follows:

$$\min_{U,V} l(\boldsymbol{Y}, \boldsymbol{U}\boldsymbol{V}^T) + \lambda(\|\boldsymbol{U}\|^2_{\text{Fro}} + \|\boldsymbol{V}\|^2_{\text{Fro}}), \tag{2}$$

where $\|\cdot\|_{\text{Fro}}$ is the Frobenius norm, λ is the regularization parameter, and $l(\boldsymbol{Y}, \boldsymbol{U}\boldsymbol{V}^T)$ is the squared loss function defined below

$$l(\boldsymbol{Y}, \boldsymbol{U}\boldsymbol{V}^T) = \sum_{(i,j)\in\Omega} c_{ij}(\boldsymbol{Z}_{ij} - \boldsymbol{U}_i\boldsymbol{V}_j^T)^2, \tag{3}$$

where c_{ij} is the confidence level regarding user i's utility over item j and $\boldsymbol{Z}_{ij}$ is a binary matrix with $\boldsymbol{Z}_{ij} = 1$ if $\boldsymbol{Y}_{ij}$ is observed, and $\boldsymbol{Z}_{ij} = 0$ otherwise.

We pause at this point to note that c_{ij} can be set with the value of $\boldsymbol{Y}_{ij}$ – the number of time user i showing interests in item j, contrasting with explicit feedback where $c_{ij} = 1, \forall i, j$ and $\boldsymbol{Z} = \boldsymbol{Y}$.

2.4 MF with Implicit Feedback and Side Information

We propose to incorporate the side information in terms of artist biographies in order to improve the predictive performance for the relation (user,artist,count) by adapting the models proposed in [3]. To distinguish the two matrices: (user, artist, count) and (artist, words, count), we denote them by $\boldsymbol{Y}^1 \in^{M\times N}$ and $\boldsymbol{Y}^2 \in \mathbb{R}^{N\times|V|}$, respectively, where $|V|$ is the carnality of the set of terms V built from the artist biographies. To this end, we propose the optimization problem below:

$$\begin{aligned}\min_{\boldsymbol{U}^1,\boldsymbol{U}^2,\boldsymbol{V}} & \sum_{(i,j)\in\Omega} c_{ij}(\boldsymbol{Y}^1_{ij} - \boldsymbol{U}^1_i\boldsymbol{V}_j^T)^2 + \sum_{(i,j)\in\Omega} c_{ij}(\boldsymbol{Y}^2_{ij} - \boldsymbol{U}^2_i\boldsymbol{V}_j^T)^2 \\ & + \lambda(\|\boldsymbol{U}^1\|^2_{\text{Fro}} + \|\boldsymbol{U}^2\|^2_{\text{Fro}} + \|\boldsymbol{V}\|^2_{\text{Fro}}),\end{aligned} \tag{4}$$

We pause at this point to note that one can concatenate these two matrices $\boldsymbol{Y}^1$ and $\boldsymbol{Y}^2$, and thus obtain the augmented matrix $\boldsymbol{Y} \in \mathbb{R}^{(M+|V|)\times K}$, and derive the following optimization problem for optimizing $\boldsymbol{U} \in \mathbb{R}^{(M+|V|)\times N}$ and $\boldsymbol{V} \in \mathbb{R}^{(N)\times K}$

$$\min_{U,V} \sum_{(i,j)\in\Omega} c_{ij}(\boldsymbol{Z}_{ij} - \boldsymbol{U}_i\boldsymbol{V}_j^T)^2 + \lambda(\|\boldsymbol{U}\|^2_{\text{Fro}} + \|\boldsymbol{V}\|^2_{\text{Fro}}). \tag{5}$$

which is equivalent to (4) despite of the difference in the dimension of $\boldsymbol{U}$.

2.5 Alternating Least Squares

We propose to use alternating lease squares (ALS) for optimizing $\boldsymbol{U}, \boldsymbol{V}$ as in (2) and (4). The idea behind ALS is below: initialize $\boldsymbol{U}$, and solve a linear algebra problem for $\boldsymbol{V}$, holding $\boldsymbol{U}$ fixed. Next hold $\boldsymbol{V}$ fixed and solve for U [3]. The liner algebra problem needs to be solved is formulated below

$$\boldsymbol{U} = (V^TCV + \lambda I)^{-1}U^TCZ, \tag{6}$$

$$\boldsymbol{V} = (U^TCU + \lambda I)^{-1}V^TCZ. \tag{7}$$

3 Experiments

In this section, we first introduce the data set – Samsung Music Hub (SMU) data set, and then report empirical results on this data set comparing our proposed models with 1) TopPop and 2) CF-Neighborhood.

3.1 Dataset

We collected the user logs for SMU from $9/1/2013$ to $2/10/2014$. Each line of the log file includes: user id, item id, and one of the three actions including button click, screen view, and search besides other information. There are three different types of items that user can interact with in SMU namely, albums, artists, and songs. The platform where the data is gathered only generates one item id and it depends to the tab with which the user is interacting. After processing the log files, we found the majority of users ($> 85\%$) have acted with the album tab. We processed each log line and selected users who have button click actions towards an album. We also have the mapping between album id and artist id from the ROVI[3] meta data, based on which we generate the observations (user, artist, count).

Table 1: Statistical properties of Samsung Music Hub data set.

	Users	Item	Count	Sparsity
Album	$225,433$	$40,371$	$611,053$	0.007%
Artist	$225,433$	$52,487$	$1,061,040$	0.009%

Table 1 shows the statistical properties of the data set[4]. Before we ran the experiment, we filtered out users and items (albums, artists) that has less than 5 button-click actions. The statistical property of the data set after filtering which is used in the experiment is given in Table 2. Therefore, we increased the observation density from 0.007% to 0.11% for (user, album, count), and from 0.009% to 0.13% for (user, artist, album).

Table 2: Statistical properties of Samsung Music Hub data set used in experiment.

	Users	Items	Action	Sparsity
Album	$22,209$	$8,318$	$219,723$	0.11%
Artist	$33,559$	$13,830$	$616,253$	0.13%

3.2 Evaluation

The evaluation methodology employed in this paper is based on the performance of the recommendation model in recommending the top-n favorite items to a user following [5]. We split the known actions (button clicks) for each user to the training and test set. We assume that all the known action items in the test set are favorite to a user. Beside the item that are favorite in the test set, we randomly choose 1000 items which are not favorite to users. Our goal is to learn the recommendation model based on the training set and predict the ranking for all the items in the test set. Then for a specific integer N, we calculate the recall and precision-recall as follows:

$$\text{recall}(N) = \frac{\#hit}{|T|}, \tag{8}$$

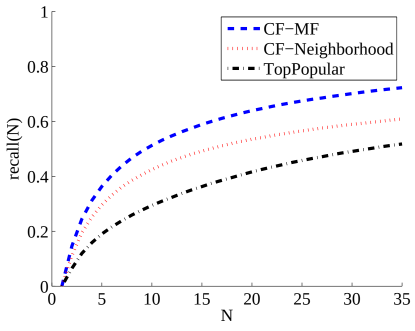
where *hit* occurs if the known action items in test set are among the recommended items to a user and $|T|$ is the total number of the known action items in the test set. We also calculate the precision at N as (please see [1] for further discussion):

$$\text{precision}(N) = \frac{\text{recall}(N)}{N}, \tag{9}$$

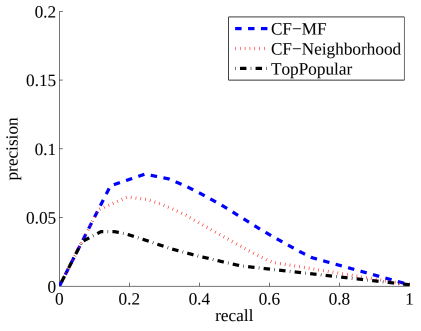
For evaluation of the algorithms on the long tail, we removed the portion of items which has 30% of the actions. This portion for (user,album,count) relation is 8% and for (user,artist,count) is 6%.

[3]`http://www.rovicorp.com/products-and-solutions/products/data/rovi-music.html`

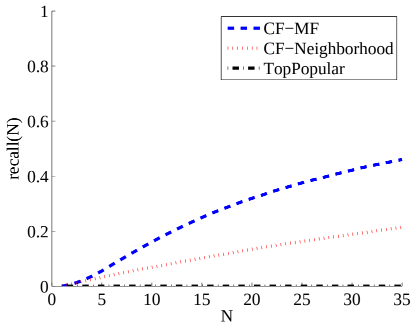
[4]Note that since each album can have more than one artist; thus the number of artist is higher than number of albums



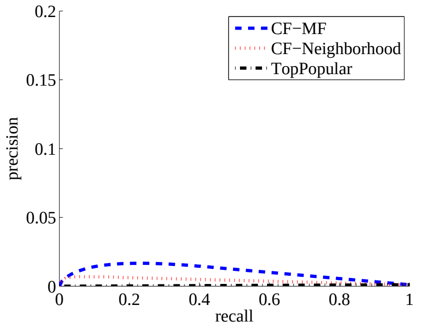
(a) recall on all items



(b) precision-recall on all items



(c) recall on long tail



(d) precision-recall on long tail

Figure 1: Results for (user,album,count) relation: a) recall at N b) precision-recall for all items c) recall at N on long tail (92% of items) d) precision-recall on long tail (92% of items)

3.2.1 Experiment Setting

We used efficient and scalable algorithms implemented in the Hadoop [5] environment based on the open source Mahout machine learning library[6]. All experiments were run with 6 Amazon Web Services (AWS) machine where the specification for each machine is: m1.xlarge (instance type), 64-bit (processor), 4 (vCPU), and 15GB (memory).

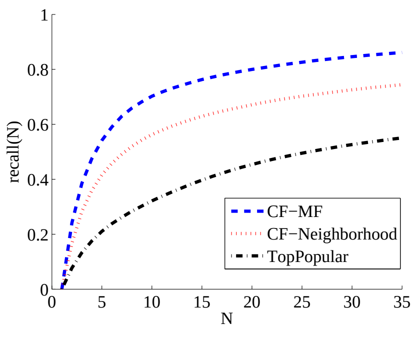
Neighborhood CF is free of regularization parameters. Matrix factorization has two parameters: 1) the number of latent factors r and 2) the regularization parameter λ. We do the cross validation for tuning r and λ over the range of $\{10, 20, 30, 50, 100\}$ and $\{0.01, 0.05, 0.1\ , 0.5\}$, respectively.

3.2.2 Results for Predicting (user, album, count)

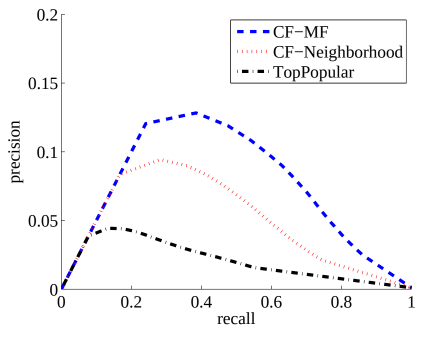
We evaluated the performance of all three algorithms on the (user, album, count) relations. After completing cross-validation over the regularization parameters for CF-MF, the best results obtained are for $r = 20$ and $\lambda = 0.01$. As shown in the Figure 1(a), we observed that CF-MF with implicit feedback consistently outperformed CF-Neighborhood and TopPop recommendations in terms of recall. It is worth to note that the recall at $N = 20$ for CF-MF is around 0.64 where for CF-Neighborhood it is around 0.53 (0.11 lower) and for TopPop it is 0.42. Figure 1(b) further showed the precision of each algorithms at a given recall. The results confirm that CF-MF outperforms CF-Neighborhood and TopPop in terms of precision-recall.

To evaluate the performance of recommendation in the long tail, Figure 1(c) showed the results on the long tail (8% of items have 30% of all actions which considered as short head). Interestingly, the

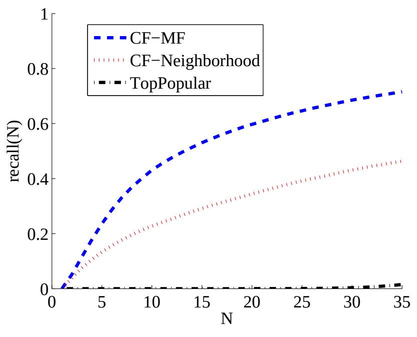
[5]`http://hadoop.apache.org/`
[6]`https://mahout.apache.org/`



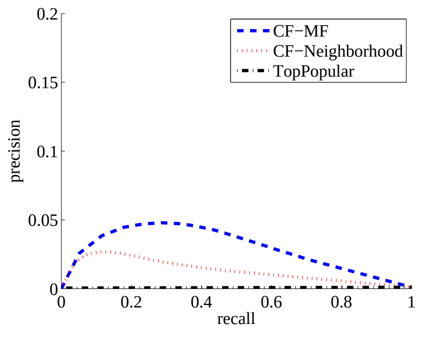
(a) recall on all items



(b) precision on all items



(c) recall on long tail



(d) precision-recall on long tail

Figure 2: Results for (user,artist,count) relation: a) recall at N b) recall at N on long tail (94% of items) c) precision-recall for all items d) precision-recall on long tail (94% of items)

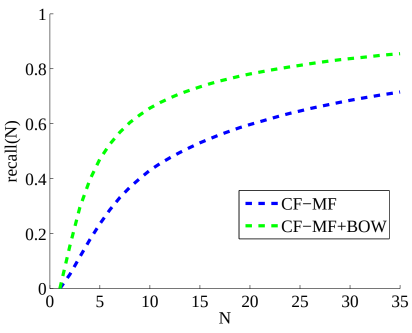
TopPop recommendation algorithm shows a very poor performance (recall at $N = 20$ is 0) where CF-MF and CF- Neighborhood performs much better than TopPop.

3.2.3 Results for Predicting (user, artist, count)

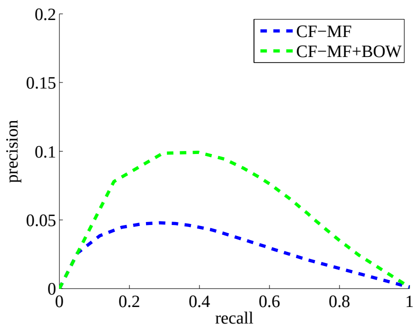
Further we evaluated the performance of all three algorithms on the matrix of (user, artist, count) relations. Cross validation on the model parameters identified optimal parameters for CF-MF with $r = 50$ and $\lambda = 0.05$. As it is shown in Figure 2, the CF-MF consistently outperformed the other two algorithms in terms of both recall and precision-recall. Note that the recall at $N = 20$ for artist recommendations is around 0.80 which is 0.18 is higher than CF-MF for (user, album, count) relation. Since (user, artist, count) relation is obtained from (user, album, count) relation, this result confirms that (user, artist, count) relation is the better representation for the music recommendation. Similar to (user, album, count) relation, the performance of CF-MF and CF-Neighborhood is better than the TopPop.

3.3 Results on Integrating Artist Biography

We used artist biography as side information to enhance the performance of music personalization. We used a bag-of-word representation and construct the term-frequency vector for each artist. Then, we concatenated term-by-artist matrix to user-artist action matrix and used the augmented matrix in the experiment. We only compared the results on CF-MF algorithm. The results of comparisons are shown in Figure 3. As indicated in the Figure, the MF with artist biography (MF-BOW) consistently outperformed the standard CF-MF. This result confirmed that integrating artist biography will improve the performance significantly, i.e., recall at $N = 20$ is 30% – from 0.60 to 0.78. The intuition is that the augmented matrix can help refine the representation of the latent factors of users and items, which is essential for one to achieve better performance. In other words, the correlation between the artist biography and artists is utilized in the concatenated matrix, leading to improved performance.



(a) Comparison of Recall with-without Side Information



(b) Comparison of Precision with-without Side Information

Figure 3: Comparison of the CF-MF performance on (user,artist,count) relation with and without artist biography $a)$ recall at N on long tail $b)$ precision-recall on long tail

3.4 Computational complexity analysis

All three algorithms are implemented in the distributed system and run in the Hadoop environment. Concerning computational complexity, TopPop is linear $\mathcal{O}(M)$ in terms of the number of items, CF-Neighborhood is quadratic $\mathcal{O}(M^2)$, and CF- MF is qubic $\mathcal{O}(M^3)$. We have to mention that the running time for TopPop, CF-Neighborhood, and CF-MF, took 3, 12, and 24 minutes, respectively. Note that 24 minutes running time is for each parameter setting of CF-MF.

4 Conclusion

In this paper, we augment the classic matrix factorization models for music recommendations by considering implicit feedback and integrating side information in a principal framework. We focused on recommendations for two types of user preferences including user preferences over albums and artists. We augmented the target matrices (user, artist, count) by side information (artist, biography) in order to recommend items in the long tail. We demonstrated on a real world mobile music App that the proposed model achieved significantly higher predictive accuracy when integrating side information. Further, we demonstrated that recommendations in the long-tail are enhanced by integrating the artist biographies.

References

[1] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *ACM RecSys*, pages 39–46. ACM, 2010.

[2] Gideon Dror, Noam Koenigstein, Yehuda Koren, and Markus Weimer. The yahoo! music dataset and kdd-cup'11. *Journal of Machine Learning Research-Proceedings Track*, 18:8–18, 2012.

[3] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of ICDM*, ICDM '08, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society.

[4] Noam Koenigstein, Gideon Dror, and Yehuda Koren. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. In *Proceedings of ACM RecSys*, pages 165–172. ACM, 2011.

[5] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of ACM SIGKDD*, pages 426–434. ACM, 2008.