Local Weighted Matrix Factorization for Implicit Feedback Datasets

Keqiang Wang[1], Xiaoyi Duan[1], Jiansong Ma[1], Chaofeng Sha[2(✉)], Xiaoling Wang[1], and Aoying Zhou[1]

[1] Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China
sei.wkq2008@gmail.com, duanxiaoyi@ecnu.cn, ecnumjs@163.com, {xlwang,ayzhou}@sei.ecnu.edu.cn

[2] Shanghai Key Laboratory of Intelligent Information Processing, Fudan University, Shanghai 200433, China
cfsha@fudan.edu.cn

Abstract. Item recommendation helps people to discover their potentially interested items among large numbers of items. One most common application is to recommend items on implicit feedback datasets (e.g., listening history, watching history or visiting history). In this paper, we assume that the implicit feedback matrix has *local* property, where the original matrix is not globally low-rank but some sub-matrices are low-rank. In this paper, we propose Local Weighted Matrix Factorization for implicit feedback (*LWMF*) by employing the kernel function to intensify *local* property and the weight function to model user preferences. The problem of sparsity can also be relieved by sub-matrix factorization in *LWMF*, since the density of sub-matrices is much higher than the original matrix. We propose a heuristic method DCGASC to select sub-matrices which approximate the original matrix well. The greedy algorithm has approximation guarantee of factor $1 - \frac{1}{e}$ to get a near-optimal solution. The experimental results on two real datasets show that the recommendation precision and recall of LWMF are both improved more than 30 % comparing with the best case of WMF.

Keywords: Recommendation systems · Local matrix factorization · Implicit feedback · Weighted matrix factorization

1 Introduction

In daily life, more and more people are going shopping online, enjoying music online and searching for restaurants' reviews before eating out. But online data are too large for people to find items that they want. So personalization recommendation can help people discover potentially interested items by analyzing user behaviors. Since they do not explicitly express user preferences. For example, ratings are explicit feedbacks which indicate users' preference, while visiting and buying history do not show users' preference directly so they are implicit

S.B. Navathe et al. (Eds.): DASFAA 2016, Part I, LNCS 9642, pp. 381–395, 2016.
DOI: 10.1007/978-3-319-32025-0_24

feedbacks. User *discovers* the item if her behaviors are implicit feedbacks, such as listened, watched or visited the item. Otherwise, user is unaware of the item. Different from the explicit feedback, the numerical value to describe implicit feedback is non-negative and very likely to be noisy [10].

Therefore, we assume that the implicit feedback matrix is not globally low-rank but some sub-matrices are low-rank. We call this important characteristic *local* property. Instead of decomposing the original matrix, we decompose the sub-matrix intuitively. We propose *Local Weighted Matrix Factorization (LWMF)*, integrating *LLORMA* [7] with *WMF* [10] in recommending by employing the kernel function to intensify *local* property and the weight function to intensify modeling user preference. The problem of sparsity can also be relieved by sub-matrix factorization in *LWMF*, since the density of sub-matrices is much higher than the original matrix.

The main contributions can be summarized as follows:

- We propose *LWMF* which integrates *LLORMA* with *WMF* to recommend items on implicit feedback datasets. *LWMF* utilizes the *local* property to model the matrix by dividing the original matrix into sub-matrices and relieves the sparsity problem.
- Based on kernel function, we propose *DCGASC (Discounted Cumulative Gain Anchor Point Set Cover)* to select the sub-matrices in order to approximate the original matrix better. At the same time, we conduct the theoretical submodularity analysis of the *DCGASC* objective function.
- Extensive experiments on real datasets are conducted to compare *LWMF* with state-of-the-art *WMF* algorithm. The experimental results demonstrate the effectiveness of our proposed solutions.

The rest of the paper is organized as follows. Section 2 reviews related work and Sect. 3 presents some preliminaries about *MF (Matrix factorization)*, *WMF* and *LLORMA*. Then we describe *LWMF* in Sect. 4. Section 5 illustrates the heuristic method *DCGASC* to select sub-matrices. Experimental evaluations using real datasets are given in Sect. 6. Conclusion and future work are followed in Sect. 7.

2 Related Work

In this section, we review some previous work on recommendation systems, including *K-Nearest Neighbor (KNN)*, *Matrix Factorization (MF)*, local ensemble methods, personalized ranking method and some other recommendation systems for special scenarios recommender.

One of the most traditional and popular way for recommender systems is KNN [1]. Item-based *KNN* uses the similarity techniques (e.g., cosine similarity, Jaccard similarity and Pearson correlation) between items to recommend the similar items. Then, *MF* [2–4] methods play an important role in model-based CF methods, which aim to learn latent factors on user-item matrix. *MF* usually gets better performance than *KNN*-based methods especially on rating

prediction. Recently some work focuses on ensembles of sub-matrix for better approximation, such as *DFC* [5], *RBBDF* [6], *LLORMA* [7,8] and *ACCAMS* [9]. However, such methods focus on explicit feedback datasets while most of the feedbacks are implicit, such as listening times, click times and check-ins. The implicit datasets do not need users to rate the items. So Hu et al. [10] and Pan et al. [11,12] propose *Weighted Matrix Factorization (WMF)* to model implicit feedback with *Alternative Least Square (ALS)*. Our work in this paper integrates *LLORMA* with *WMF* and proposes a heuristic method to select sub-matrices. Other related work on implicit feedback datasets are ranking methods, such as *BPR* [13] and Pairwise Learning [14]. With the explosion of size of the training data, the ranking methods need use some efficient sampling techniques to reduce complexity. Finally, there are several special scenarios, such as recommending music [15], News [16], TV show [17] and POI [18,19], utilizing the additional information (e.g., POI recommender considers the geographical information) to improve prediction performance.

3 Preliminary

In this section, we present some preliminaries about *MF*, *WMF* and *LLORMA*.

3.1 Matrix Factorization

MF is a dimensionality reduction technique, which has been widely used in recommendation system especially for the rating prediction [3,4]. Due to their attractive accuracy and scalability, *MF* plays a vital role in recent recommendation system competitions, such as *Netflix Prize*[1], *KDD Cup 2011 Recommending Music Items*[2], *Alibaba Big Data Competitions*[3] and so on. Given a sparse matrix $R \in \mathbb{R}^{N \times M}$ with indicator matrix I, and latent factor number $F \ll \min\{N, M\}$. The aim of *MF* is:

$$\min_{P,Q} \sum_{u=1}^{N} \sum_{m=1}^{M} I_{um}(R_{um} - \hat{R}_{um})^2 \tag{1}$$

In order to avoid over-fitting, regularization terms are usually added to the objective function to modify the squared error. So the task is to minimize $\sum_{u=1}^{N} \sum_{m=1}^{M} I_{um}(R_{um} - \hat{R}_{um})^2 + \lambda \|P\|_F^2 + \lambda \|Q\|_F^2$. The parameter λ is used to control the magnitudes of the latent feature matrices, P and Q. Stochastic gradient descent is often used to learn the parameters [4].

3.2 Weighted Matrix Factorization

[10] argues that original *MF* is always used on explicit feedback datasets, especially for rating prediction. So Hu et al. [10] and Pan et al. [11,12] propose *Weighted Matrix Factorization* (*WMF*) to handle the cases with implicit

[1] http://www.netflixprize.com/.
[2] http://www.kdd.org/kdd2011/kddcup.shtml.
[3] https://102.alibaba.com/competition/addDiscovery/index.htm.

feedback. Recently, *WMF* has been widely used in TV show, music and POI (Point-of-Interests) recommendation. To utilize the undiscovered items and to distinguish between discovered and undiscovered items, a weight is added to the *MF*:

$$W_{um} = 1 + log(1 + R_{um} \times 10^{\varepsilon}) \tag{2}$$

where the constant ε is used to control the rate of increment. Considering the weights of implicit feedback, the optimization function is reformulated as follows:

$$\min_{P,Q} \sum_{u=1}^{N} \sum_{m=1}^{M} W_{um}(C_{um} - P_u Q_m^T)^2 + \lambda \|P\|_F^2 + \lambda \|Q\|_F^2 \tag{3}$$

where each entry C_{um} in the 0/1 matrix C indicates whether the user u has discovered the item m, which can be defined as $C_{um} = \begin{cases} 1 & R_{um} > 0 \\ 0 & R_{um} = 0. \end{cases}$

3.3 Low-Rank Matrix Approximation

LLORMA [7,8] is under the assumption of locally low rank instead of globally low rank. That is, limited to certain types of similar users and items, the entire rating matrix R is not low-rank but a sub-matrix R_s is low-rank. It is to say that the entire matrix R is composed by a set of low-rank sub-matrices $\mathcal{R}_s = \{R^1, R^2, ..., R^H\}$ with weight matrix set $\mathcal{T} = \{T^1, T^2, ..., T^H\}$ of sub-matrices, where T_{ij}^h indicates the sub-matrix weight of R_{ij}^h in R_h:

$$R_{um} = \frac{1}{Z_{um}} \sum_{h=1}^{H} T_{u_h m_h}^h R_{u_h m_h}^h \tag{4}$$

where $Z_{um} = \sum_{h=1}^{H} T_{u_h m_h}^h$. *LLORMA* uses the *MF* introduced in Sect. 3.1 to approximate the sub-matrix R^h. If the matrix has *local* property, we can achieve good accuracy in predicting ratings.

4 Local Weighted Matrix Factorization

In this section, we introduce our method *LWMF* for implicit datasets. Following the *LLORMA*, we first select sub-matrices from the original matrix, then each sub-matrix is decomposed by *WMF* methods as shown in Fig. 1. We propose *LWMF* which integrates *LLORMA* with *WMF* to recommend top-N items on implicit datasets. We estimate each sub-matrix R_h by *WMF* in Sect. 3.2 as follows:

$$\min_{P_h,Q_h} \sum_{u_h=1}^{N_h} \sum_{m_h=1}^{M_h} T_{u_h m_h}^h W_{u_h m_h} (R_{u_h m_h}^h - {P_{u_h}^h}^T Q_{m_h}^h)^2 + \lambda_1 \|P^h\|_F^2 + \lambda_2 \|Q^h\|_F^2 \tag{5}$$

So the original Matrix R can be approximated by the set of approximated sub-matrices $\hat{\mathcal{R}}_s = \{\hat{R}^1, \hat{R}^2, ..., \hat{R}^H\}$:

$$R_{um} \approx \frac{1}{Z_{um}} \sum_{h=1}^{H} T^h_{u_h m_h} \hat{R}^h_{u_h m_h} \tag{6}$$

where $\hat{R}^h = {P^h}^T Q^h$. So there are mainly two problems: (1) How to calculate the weight matrix T^h? (2) How to select the sub-matrix set $\mathcal{R}_s$? *LLORMA* uses the kernel methods[4] to solve these two problems. *LLORMA* employs three popular smoothing kernels (i.e., the uniform kernel, the triangular kernel and the Epanechnikov kernel) to calculate the relationship between users or items. Then it randomly chooses some user-item pairs[5] from training data as the anchor point set. Each anchor point $a_h = (u_h, m_h)$ is related to other pairs (u_i, m_i) of which the kernel is greater than 0. So the related pairs and the anchor point a_h make up a sub-matrix R^h while the kernel matrix acts as the weight matrix T^h.

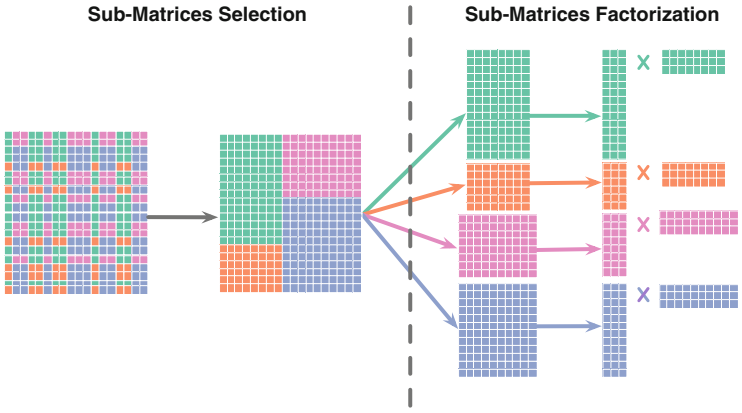


Fig. 1. Local matrix factorization

In this paper, we use the Epanechnikov kernel to calculate the relationship between two pairs (u_h, m_h) and (u_j, m_j). It is computed as the product of user Epanechnikov kernel $(E_{b_u}(u_h, u_h))$ and item Epanechnikov kernel $(E_{b_m}(m_j, m_j))$ as follows:

$$E(a_h, a_j) = E_{b_u}(u_h, u_j) \times E_{b_m}(m_h, m_j) \tag{7}$$

where

$$E_{b_u}(u_h, u_j) \propto (1 - d(u_h, u_j)^2)\, \mathbf{1}_{\{d(u_h, u_j) \leq b_u\}}$$
$$E_{b_m}(m_h, m_j) \propto (1 - d(m_h, m_j)^2)\, \mathbf{1}_{\{d(m_h, m_j) \leq b_m\}}$$

[4] https://en.wikipedia.org/wiki/Kernel_(statistics).
[5] Pair (u, m) means that the user u discovered the item m.

where b_u, b_m are the bandwidth parameters of kernel. Distance between two users or two items is the distance between two row vectors (for user kernel) or column vectors (for item kernel). The initial user latent factor and item latent factor are learned by *WMF*. Accordingly, the distance between users u_i and u_j is $d(u_i, u_j) = arccos(\frac{P_{u_i} \cdot P_{u_j}}{\|P_{u_i}\| \cdot \|P_{u_j}\|})$, where P_{u_i}, P_{u_j} are the ith and jth rows of P. The distance between items is computed in the same way.

Different from *LLORMA*, all kernel weights are set to the same value (i.e., $T^h = I^h$) in this paper. That is to say, the weights of all user-item pairs to the anchor point in the sub-matrix are identical. The intuition behind the setting is that:

1. In *LWMF*, we aim to find sub-matrices. So we just use the Epanechnikov kernel to get one sub-matrix by selecting an anchor point and do not care much about the weight;
2. Due to the weight matrix W and preference matrix C settings, the preference matrix C is not sparse so that stochastic gradient descent is not applied. *WMF* employs *Alternative Least Square (ALS)* to optimize this objective function. The iterative formulas are $P_u = (QW^uQ^T + \lambda I)^{-1}QW^uC^u$ and $Q_m = (PW^mP^T + \lambda I)^{-1}PW^mC^m$ where for user u, W^u is a diagonal $M \times M$ matrix and $W^u_{mm} = W_{um}$, and C^u is the vector of the preferences C_{um} by user u. So the meaning of W^m and C^m for item m is as same. The total running time of naive calculation for all users is $O(F^2NM)$. [10] devises a nice trick by using the fact that $QW^uQ^T = QQ^T + Q(W^u - I)Q^T$. So the running time for each iteration is reduced to $O(F^2\mathcal{N} + F^3N)$, where $\mathcal{N} = \sum_{u=1}^{N}\sum_{m=1} I_{um}$. It is to say that its time complexity is in proportion to the total number of non-zero entries in the matrix R. So if we treat all weights of sub-matrices all the same, the iterative of *WMF* for sub-matrices is $O(F_s^2\mathcal{N}_\int + F_s^3N_s)$. Otherwise, the running time is $O(F_s^2N_sM_s)$. Considering the training data reduction of each sub-matrix, the running time of each one can be much faster than the original matrix.
3. We do some extensive experiments and find that the results of *LWMF* are almost the same whether considering kernel weight or not.

Each anchor point stands for a sub-matrix. Selecting the sub-matrix set $\mathcal{R}_s$ is in fact to select a set of anchor points. The details of selecting anchor point set is discussed in the next section.

5 Anchor Point Set Selection

Intuitively, the sub-matrix set $\mathcal{R}_s = \{R^1, R^2, ..., R^H\}$ should cover the original matrix R, that is $R = \cup_{R^h \in \mathcal{R}_s} R^h$, so these sub-matrix sets $\mathcal{R}_s$ can approximate the original matrix R better than the set that does not cover. So the anchor points selection problem can be reduced to the set cover problem. Given the candidate anchor point set $A = \{a_1, a_2, ..., a_n\}$ while every candidate point a_i can cover itself several other candidate points denoted by $A_i = \{a_i, a_{i1}, a_{i2}, ..., a_{ih}\} \subset A$, we propose the naive anchor points cover method:

Anchor Point Set Cover (ASC): returns an anchor point set $S \subset A$ such that

$$\max f(S) = |\cup_{i \in S} A_i| \\ s.t. |S| = K \tag{8}$$

Here, we use all pairs (u, m) in training data as the candidate anchor points. Obviously, the *ASC* problem is submodular and monotone. So the greedy algorithm has $1 - \frac{1}{e}$ approximation of optimization.

However, covering all training data only once in ASC is not enough. Although performance is improved by increasing cover times, the gain is discounted, which is similar to the situation in ranking quality measures *NDCG* (Normalized Discounted Cumulative Gain) [23] and *ERR* (Expected Reciprocal Rank) [22] in *IR*(Information Retrial). So we propose a heuristic method to model this situation as follows.

Discounted Cumulative Gain Anchor Point Set Cover (DCGASC): returns an anchor point set $S \subset A$ such that

$$\max f(S) = \sum_{a_l \in \cup_{a_i \in S} A_i} \sum_{o_l=1}^{O_l} \alpha^{o_l - 1} = \sum_{z=1}^{|S|} \sum_{a_l \in A_{iz}} \alpha^{o_{lz} - 1} \\ s.t. |S| = K \tag{9}$$

where O_l denotes the covered time of a_l by itself or other selected anchor points. Below we prove that $f(\cdot)$ is also submodular and monotone.

Theorem 1. *DCGASC function is submodular and also monotone non-decreasing.*

Proof. Let $S \subseteq V \subseteq A$ and $A_h \in A \backslash V$. We have that

$$\begin{aligned} f(S \cup \{A_h\}) - f(S) &= \sum_{z=1}^{|S \cup \{A_h\}|} \sum_{a_l \in A_{iz}} \alpha^{o_{lz} - 1} - \sum_{z=1}^{|S|} \sum_{a_l \in A_{iz}} \alpha^{o_{lz} - 1} \\ &= \sum_{z=1}^{|S|} \sum_{a_l \in A_{iz}} \alpha^{o_{lz} - 1} + \sum_{a_l \in A_h} \alpha^{o_{lz}^S - 1} - \sum_{z=1}^{|S|} \sum_{a_l \in A_{lz}} \alpha^{o_{lz} - 1} \\ &= \sum_{a_l \in A_h} \alpha^{o_{lz}^S - 1} \geq 0 \end{aligned} \tag{10}$$

and

$$\begin{aligned} f(S \cup \{A_h\}) - f(S) - (f(V \cup \{A_h\}) - f(V)) &= \sum_{a_l \in A_h} \alpha^{o_{lz}^S - 1} - \sum_{a_l \in A_h} \alpha^{o_{lz}^V - 1} \\ = \sum_{a_l \in A_h} (\alpha^{o_{lz}^S - 1} - \alpha^{o_{lz}^V - 1}) &= \sum_{a_l \in A_h} \alpha^{o_{lz}^S - 1} (1 - \alpha^{o_{lz}^S - o_{lz}^V}). \end{aligned} \tag{11}$$

Algorithm 1. *DCGASC* Greedy Algorithm

Input : Set of anchors A, anchor number K, DCGASC function f and sets A_i covered by each anchor a_i
Output: $S \subseteq A$ with $|S| = k$

$S \leftarrow \{\arg\max_{a_l \in A} |A_l|\}$;
while $|S| < K$ **do**
  $l \leftarrow \arg\max_{a'_l \in A \setminus S} f(S \cup \{a'_l\}) - f(S)$
  $S \leftarrow S \cup \{a_l\}$
end
return S

Because the number of anchor points covered satisfies that $o_{lz}^S \leqslant o_{lz}^V$ and discount parameter $\alpha \in [0, 1)$, we know that $f(S \cup \{A_h\}) - f(S) - (f(V \cup \{A_h\}) - f(V)) \geq 0$. Therefore, it is proved that the *DCGASC* function is monotone and submodular.

Due to the monotonicity and submodularity of *DCGASC* function, the greedy Algorithm 1 can provide a theoretical approximation guarantee of factor $1 - \frac{1}{e}$ as described in [24].

6 Experiments

In this section, we evaluate the method proposed in this paper using real datasets. We first introduce the datasets and experimental settings. Then we compare our method with *WMF* under specific parameter settings. We also compare results with different anchor numbers and three anchor points selection methods.

6.1 Dataset

We choose two datasets. One is the *Gowalla* from [20], one of the most popular online LBSNs datasets. Another is *YES* [21], which is a playlist dataset crawled by using the web based API[6].

The experimental dataset is chosen from Gowalla dataset within the range of latitude from 32.4892 to 41.7695 and longitude from −124.3685 to −114.5028, where locate in California and Nevada. We consider users who check in more than 10 distinct POIs and the POIs which are visited by more than 10 users. So it contains 205,509 check-ins made by 5,086 users at 7,030 locations. Finally the density of is 4.68×10^{-3} and very sparse.

YES [21] consists of radio playlists.[7] There are 431,367 playlists and 3,168 songs. A playlist corresponds to a user. Similarly, we consider users who listen in more than 10 distinct songs and the songs which are listened by more than

[6] http://api.yes.com.
[7] www.cs.cornell.edu/~shuochen/lme/data_page.html.

Table 1. Detail information of *Gowalla* and *YES*

Gowalla	
#user	5,086
#locations	7,030
#check-ins	167,404
avg.#users per loc.	23.81
avg. #loc. per user	32.91
max #users per loc.	1,644
max #loc. per user	762

YES	
#user	40,465
#songs	2,563
#listens	735,367
avg. #user per song	286.92
avg. #songs per user	18.17
max #users per song	4,217
max #songs per user	134

10 users. So the dataset contains 40,465 users and 2,563 songs. The density is about 7.09×10^{-3}.

More details about two datasets are showed in the Table 1. We randomly select 80 % of each user's visiting locations as the training set and the 20 % as the testing set.

6.2 Setting

Next, we show the parameter values. The regularization λ is set to 0.01 and the performance of recommendation is not sensitive to this parameter. The weight parameter ε is set to 4. We set the bandwidth parameter in Epanechanikov kernel as $b_u = b_m = 0.8$. The discount α of *DCGASC* is set to 0.7. We select 200 anchor points for Gowalla dataset and 100 anchor points for YES dataset. In the experiments, we observe that if the number of anchor points is larger, the performance is better. But the training time increases accordingly.

We employ the Precision@N and Recall@N to measure the performance. For a user u, we set $\mathcal{I}^P(u)$ as the predicted item list and $\mathcal{I}^T(u)$ as the true list in the testing dataset. So the Precision@N and Recall@N are:

$$Precision@N = \frac{1}{|U|} \sum_{u \in U} \frac{|\mathcal{I}^P(u) \bigcap \mathcal{I}^T(u)|}{N}, \; Recall@N = \frac{1}{|U|} \sum_{u \in U} \frac{|\mathcal{I}^P(u) \bigcap \mathcal{I}^T(u)|}{|\mathcal{I}^T(u)|}$$

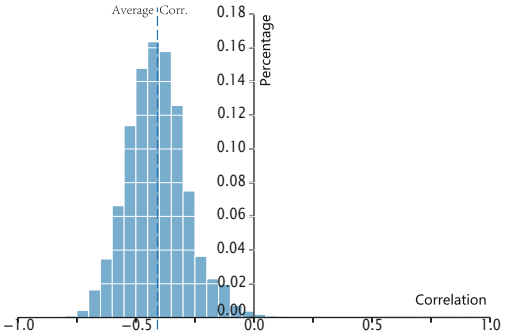
where $|\mathcal{I}^P(u)| = N$. In our base experiments, we choose top 5, 10, 20 and 30 as evaluation metrics.

We compare two methods for implicit feedback datasets:

- *WMF:* This is the state-of-the-art method which is designed for implicit feedback [10].
- *LWMF:* This is our proposed method that takes account of two ideas of *WMF* [10] and *LLORMA* [7].

Then we compare three anchor points selection methods to study the performance of *LWMF*:

- *Random:* Sampling anchor points uniformly from training dataset as paper [7] does.



$sim(m_1, m_2)$ is calculated by $cos(m_1, m_2) = \frac{m_1 \cdot m_2}{\|m_1\| \cdot \|m_2\|}$; $distance(m_1, m_2)$ is calculated by Euclidean distance.

Fig. 2. Statistics of correlation between sim(m_1, m_2) and distance(m_1, m_2) on *Gowalla*

- *Anchor Set Cover (ASC):* Set cover method, which is submodular and monotone, and the greedy algorithm can provide a theoretical approximation guarantee of factor $1 - \frac{1}{e}$.
- *Discounted Cumulative Gain Anchor Set Cover (DCGASC):* Discounting cumulative gain of covering the points which is also submodular and monotone.

So *LWMF* can be expanded into three sub-methods *LWMF_Random*, *LWMF_ASC* and *LWMF_DCGASC*. By defualt, *LWMF* means *LWMF_DCGASC*. Each method is conducted 5 times independently. Therefore, the average score indicates the performance of the recommendation methods.

6.3 Results

In this section, we discuss the experimental results on *Gowalla* and *YES* datasets.

6.3.1 Recommendation Methods Comparison

Tables 2 and 3 list the precision and recall of methods *WMF* and *LWMF* with *DCGASC*. It shows the same result as [7] that *LORMA* outperforms *SVD*, and *LWMF* always outperforms *WMF*. With the rank r increases, both *LWMF* and *WMF* get better, but the improvements get less when r is 20 of *WMF* and 15 of *LWMF*. Noted that the results of LWMF with rank 3 are almost the same as WMF with rank $r = 20$ on both datasets. For *Gowalla* dataset, *LWMF* with rank $r = 15$ even improves the precision@5 by 35.37 % and the recall@5 by 31.37 %. For *YES* dataset, *LWMF* improves the precision@5 by 7.92 % and the recall@5 11.21 %. More obvious improvements on Gowalla is due to the local property. For example, there are some business districts in a city and business POIs are geographically close to each other within each business district. And it shows that the average cover rate of anchor points on *Gowalla* dataset is about 6.5 % while

Table 2. Precision and recall comparison on *Gowalla*

Top-N	Precision				Recall			
	5	10	20	30	5	10	20	30
WMF rank5	0.0646	0.0489	0.0363	0.0302	0.0587	0.0848	0.1229	0.1502
WMF rank10	0.0752	0.0570	0.0432	0.0359	0.0676	0.0991	0.1457	0.1786
WMF rank15	0.0819	0.0628	0.0471	0.0393	0.0748	0.1096	0.1593	0.1955
WMF rank20	0.0862	0.0659	0.0494	0.0412	0.0797	0.1165	0.1659	0.2053
LWMF rank3	0.0872	0.0662	0.0495	0.0415	0.0801	0.1152	0.1650	0.2033
	1.17 %	0.34 %	0.33 %	0.73 %	0.57 %	−1.12 %	−0.57 %	−0.96 %
LWMF rank5	0.0986	0.0750	0.0548	0.0459	0.0899	0.1292	0.1827	0.2239
	14.37 %	13.67 %	10.97 %	11.18 %	12.82 %	10.93 %	10.14 %	9.11 %
LWMF rank10	0.1101	0.0843	0.0614	0.0511	0.0992	0.1444	0.2045	0.2491
	27.70 %	27.81 %	24.45 %	23.89 %	24.48 %	23.94 %	23.25 %	21.34 %
LWMF rank15	**0.1167**	**0.0877**	**0.0638**	**0.0522**	**0.1047**	**0.1504**	**0.2131**	**0.2588**
	35.37 %	33.03 %	29.29 %	26.49 %	31.37 %	29.12 %	28.48 %	26.08 %

Table 3. Precision and recall comparison on *YES*

Top-N	Precision				Recall			
	5	10	20	30	5	10	20	30
WMF rank5	0.0793	0.0641	0.0487	0.0409	0.1030	0.2130	0.2506	0.3171
WMF rank10	0.1046	0.0842	0.0624	0.0501	0.1349	0.2765	0.3245	0.3913
WMF rank15	0.1104	0.0890	0.0655	0.0525	0.1421	0.2913	0.3393	0.4083
WMF rank20	0.1111	0.0895	0.0659	0.0530	0.1429	0.2931	0.3414	0.4118
LWMF rank3	0.1121	0.0899	0.0666	0.0542	0.1467	0.2935	0.3399	0.4116
	0.88 %	0.53 %	1.11 %	2.29 %	2.68 %	0.11 %	−0.45 %	−0.04 %
LWMF rank5	0.1167	0.0919	0.0671	0.0538	0.1517	0.3009	0.3495	0.4218
	5.02 %	2.74 %	1.90 %	1.65 %	6.18 %	2.65 %	2.36 %	2.42 %
LWMF rank10	0.1182	0.0923	**0.0684**	**0.0545**	0.1569	0.3082	0.3552	0.4245
	6.36 %	3.19 %	3.79 %	2.84 %	9.84 %	5.14 %	4.04 %	3.09 %
LWMF rank15	**0.1199**	**0.0938**	**0.0682**	**0.0545**	**0.1589**	**0.3118**	**0.3591**	**0.4282**
	7.92 %	4.89 %	3.49 %	2.93 %	11.21 %	6.37 %	5.17 %	3.98 %

it is about 11.6 % on *YES* dataset. To validate the local property in *Gowalla*, we calculate the correlation between similarity of locations ($sim(m_1, m_2)$) and their geographic distance ($distance(m_1, m_2)$), where m_1 and m_2 are each pair of locations.

As shown in Fig. 2, we make statistics about correlation between similarity and geographic distance among the locations. Correlation coefficient between two locations is negative and the average is about −0.41. Therefore, two near locations are more similar than two further locations. So the *local* property leads that the user-location matrix is not globally low-rank but locally low-rank.

6.3.2 Comparison with Different Discounts for *DCGASC*

Figure 3 shows the performance of *LWMF* with different anchor numbers. For both datasets, the precision and recall of both *LWMF* and *WMF* improve while r increases and *LWMF* performances better than *WMF* with rank $r \geq 3$. For *Gowalla* dataset, *LWMF* with rank $r = 15$ and anchor number $K \geq 20$ outperforms *WMF* with rank $r = 20$. While the same performance on *YES* dataset needs $K \geq 65$ anchor points. We can see that as the number of anchor points increases, the performance gets better. Although the training time increases, the gap of running time of matrix factorization between *LWMF* and *WMF* is small. Because the running time of *WMF* is $O(F^2\mathcal{N} + F^3N)$ and the sub-matrices of *LWMF* are much smaller than the original matrix (i.e., in both datasets, each sub-matrix is about 10 % of original matrix averagely). Only one sub-matrix factorization is much faster than original matrix factorization. Despite all this, *LWMF* costs more time on calculating the KDE between users and items and selecting anchor points.

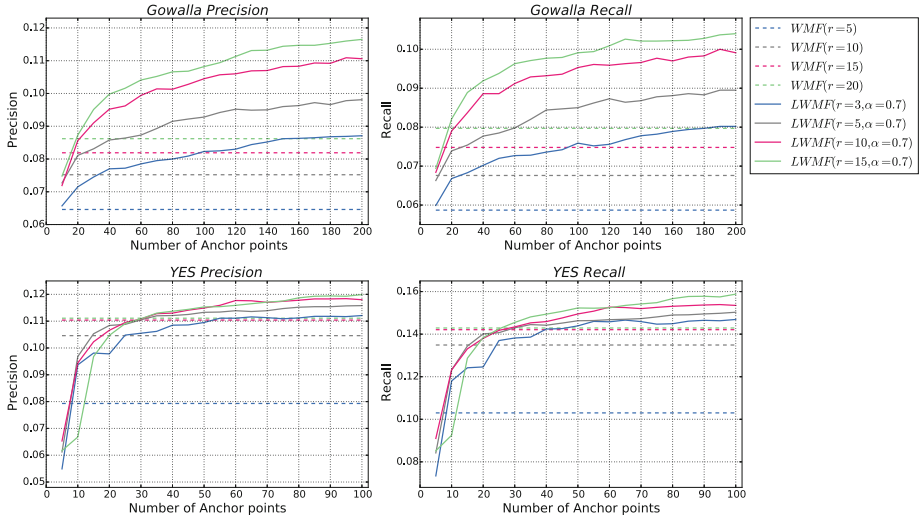


Fig. 3. Comparison with different discounts for *DCGASC* (Color figure online)

6.3.3 Anchor Point Set Selection Methods Comparison

Next, we compare the performance of *LWMF_Random*, *LWMF_ASC* and *LWMF_DCGASC* in Fig. 4. The discount parameter α is set 0.7. The method *ASC* may cover all training data before selecting K anchor points. After covering all training data, we use *Random* method to select the remaining anchor points. From Fig. 4, when the number of anchor points is small, *LWMF_DCGACS* and *LWMF_ACS* perform better in precision and recall. When the number of anchor points increases, the gap of performance among three gets less. Despite of this, *LWMF_DCGACS* and *LWMF_ACS* outperform *LWMF_Random* on both datasets and *LWMF_DCGACS* is the best.

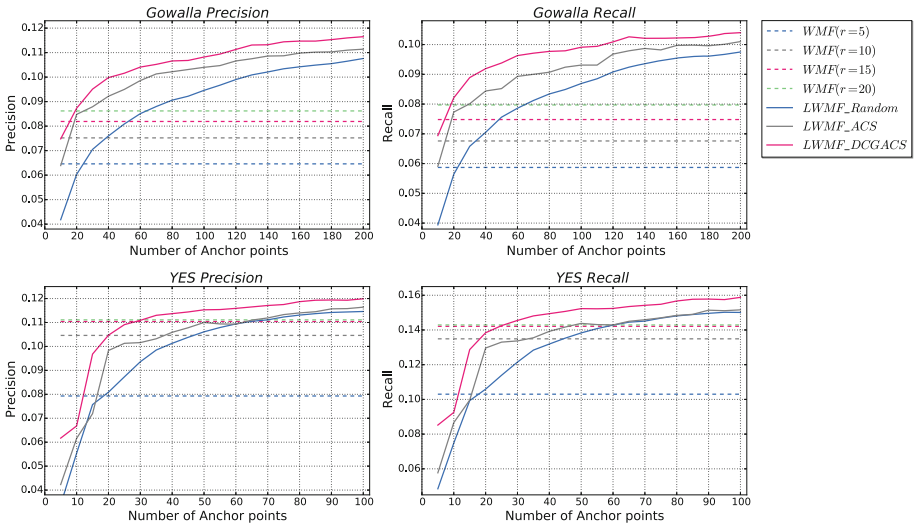


Fig. 4. Anchor point set selection methods comparison (Color figure online)

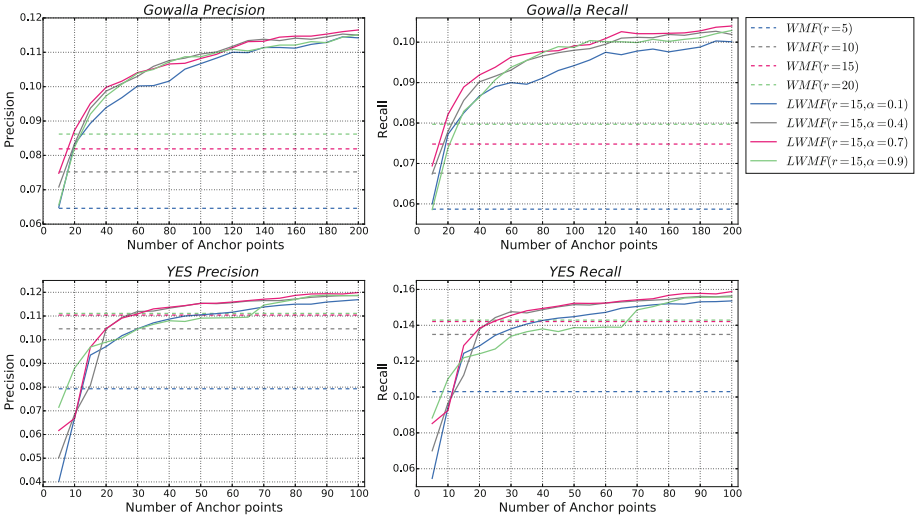


Fig. 5. Comparison with different discounts for *DCGASC* (Color figure online)

6.3.4 Comparison with Different Discounts for *DCGASC*

Finally, we study the performance of *LWMF_DCGACS* with different discount parameters. For each α, we explore results obtained by varying the parameter in the range $(0, 1]$ with decimal steps. Because the results with discount parameter $\alpha \in [0.2, 0.8]$ are similar, we only plot the curves with $\alpha \in \{0.1, 0.4, 0.7, 0.9\}$ in Fig. 5. The gap of performance with four discount parameters is small. The

performance with discount parameter $\alpha = 0.1$ or 0.9 is a little worse. In general, the performance of *LWMF* is not sensitive to the discount parameter but mainly depends on the number of anchor points.

7 Conclusion and Future Work

In this paper, we propose *LWMF* which selects sub-matrices to model the user behavior better. *LWMF* relieves the sparsity problem by sub-matrix factorization. Moreover, we propose *DCGASC* to select sub-matrix set which improves the performance of *LWMF*. The extensive experiments on two real datasets demonstrate the effectiveness of our approach compared with state-of-the-art method *WMF*.

We want to study the three further directions: (1) To speed up selecting sub-matrices; (2) In this paper, we first select the sub-matrix set by selecting anchor points, then do the weighted matrix factorization for each sub-matrix. So we need two steps to optimize the objective function. We can try to find the methods to optimize the local matrix factorization in only one objective function; (3) We can further leverage other special additional information into *LWMF* in some special scenarios (such as, the geographical information in POI recommender).

Acknowledgement. This work was supported by the NSFC grants (No. 61472141, 61370101 and 61021004), Shanghai Leading Academic Discipline Project (No. B412), and Shanghai Knowledge Service Platform Project (No. ZF1213).

References

1. Deshpande, M., Karypis, G.: Item-based top-n recommendation algorithms. ACM Trans. Inf. Syst. (TOIS) **22**(1), 143–177 (2004)
2. Paterek, A.: Improving regularized singular value decomposition for collaborative filtering. In: Proceedings of KDD Cup and Workshop, vol. 2007, pp. 5–8 (2007)
3. Mnih, A., Salakhutdinov, R.: Probabilistic matrix factorization. In: Advances in Neural Information Processing Systems, pp. 1257–1264 (2007)
4. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **8**, 30–37 (2009)
5. Mackey, L.W., Jordan, M.I., Talwalkar, A.: Divide-and-conquer matrix factorization. In: Advances in Neural Information Processing Systems, pp. 1134–1142 (2009)
6. Zhang, Y., Zhang, M., Liu, Y., et al.: Localized matrix factorization for recommendation based on matrix block diagonal forms. In: Proceedings of the 22nd International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, pp. 1511–1520 (2013)
7. Lee, J., Kim, S., Lebanon, G., et al.: Local low-rank matrix approximation. In: Proceedings of the 30th International Conference on Machine Learning, pp. 82–90 (2013)
8. Lee, J., Bengio, S., Kim, S., et al.: Local collaborative ranking. In: Proceedings of the 23rd International Conference on World Wide wWeb, pp. 85–96. ACM (2014)

9. Beutel, A., Ahmed, A., Smola, A.: Additive co-clustering to approximate matrices succinctly. In: Proceedings of the 24th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, pp. 119–129 (2015)
10. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: 8th IEEE International Conference on Data Mining: ICDM 2008, pp. 263–272. IEEE (2008)
11. Pan, R., Zhou, Y., Cao, B., et al.: One-class collaborative filtering. In: 8th IEEE International Conference on Data Mining, pp. 502–511. IEEE (2008)
12. Pan, R., Scholz, M.: Mind the gaps: weighting the unknown in large-scale one-class collaborative filtering. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 667–676. ACM (2009)
13. Rendle, S., Freudenthaler, C., Gantner, Z., et al.: BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, pp. 452–461. AUAI Press (2009)
14. Rendle, S., Freudenthaler, C.: Improving pairwise learning for item recommendation from implicit feedback. In: Proceedings of the 7th ACM International Conference on Web Search, Data Mining, pp. 273–282. ACM (2014)
15. Yang, D., Chen, T., Zhang, W., et al.: Local implicit feedback mining for music recommendation. In: Proceedings of the 6th ACM Conference on Recommender Systems, pp. 91–98. ACM (2012)
16. Ilievski, I., Roy, S.: Personalized news recommendation based on implicit feedback. In: Proceedings of the 2013 International News Recommender Systems Workshop, Challenge, pp. 10–15. ACM (2013)
17. Zibriczky, D., Hidasi, B., Petres, Z., et al.: Personalized recommendation of linear content on interactive TV platforms: beating the cold start and noisy implicit user feedback. UMAP Workshops (2012)
18. Lian, D., Zhao, C., Xie, X., et al.: GeoMF: joint geographical modeling and matrix factorization for point-of-interest recommendation. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 831–840. ACM (2014)
19. Liu, Y., Wei, W., Sun, A., et al.: Exploiting geographical neighborhood characteristics for location recommendation. In: Proceedings of the 23rd ACM International Conference on Information, Knowledge Management, pp. 739–748. ACM (2014)
20. Cho, E., Myers, S.A., Leskovec, J.: Friendship and mobility: user movement in location-based social networks. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1082–1090. ACM (2011)
21. Chen, S., Moore, J.L., Turnbull, D., et al.: Playlist prediction via metric embedding. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery, Data Mining, pp. 714–722. ACM (2012)
22. Chapelle, O., Metlzer, D., Zhang, Y., et al.: Expected reciprocal rank for graded relevance. In: Proceedings of the 18th ACM Conference on Information, Knowledge Management, pp. 621–630. ACM (2009)
23. Clarke, C.L.A., Kolla, M., Cormack, G.V., et al.: Novelty, diversity in information retrieval evaluation. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research, Development in Information Retrieval, pp. 659–666. ACM (2008)
24. Nemhauser, G., Wolsey, L.A., Fisher, M.: An analysis of approximations for maximizing submodular set functions - I. Math. Program. **14**(1), 265–294 (1978)