Data Sci. Eng. (2016) 1:252–264
DOI 10.1007/s41019-017-0032-6

Local Weighted Matrix Factorization for Top-*n* Recommendation with Implicit Feedback

Keqiang Wang[1] · **Hongwei Peng**[1] · **Yuanyuan Jin**[1] · **Chaofeng Sha**[2] · **Xiaoling Wang**[1]

Received: 4 July 2016 / Accepted: 14 December 2016 / Published online: 7 February 2017

Abstract Item recommendation helps people to discover their potentially interested items among large numbers of items. One most common application is to recommend top-*n* 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 (LWMF) for top-*n* recommendation 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 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 about 30% comparing with the best case of weighted matrix factorization (WMF).

Keywords Recommendation systems · Local matrix factorization · Implicit feedback · Weighted matrix factorization · Item recommender · Sub-modular

1 Introduction

MF [4] projects users and items into a latent low-dimensional space. Further, the missing entries in the original matrix can be recovered using the dot product between user and item latent vectors. Recently, LLORMA [7] has been shown to be more effective than the traditional MF. The original matrix is divided into several smaller sub-matrices, in which we can exploit local structures for better low-rank approximation. In each sub-matrix, the standard MF technique is applied to generate sub-matrix-specific latent vectors for both users and items.

The above techniques can achieve good performance in rating prediction when high-quality explicit feedback is available. For example, ratings are explicit feedbacks which indicate users' preference. However, explicit feedbacks are not easy to get and rating prediction cannot be used in top-*n* item recommendation directly. Compared with the explicit feedback, the implicit feedbacks are more common and larger. User discovers the item if her behaviors are implicit feedbacks, such as listening, watching or visiting the item. Otherwise, user is unaware of the item. Different from the explicit feedback, the numerical value to describe implicit feedback is nonnegative and very likely to be noisy [10].

Therefore, we consider doing top-*n* item recommendation based on implicit feedback datasets. Specifically, we also assume that the implicit feedback matrix is not

✉ Xiaoling Wang
xlwang@sei.ecnu.edu.cn

Keqiang Wang
sei.wkq2008@gmail.com

Hongwei Peng
penghognwei_phw@163.com

Yuanyuan Jin
2547124038@qq.com

Chaofeng Sha
cfsha@fudan.edu.cn

[1] Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China

[2] Shanghai Key Laboratory of Intelligent Information Processing, Fudan University, Shanghai 200433, China

globally low rank but some sub-matrices are low rank. 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. Two key issues of such a sub-matrix-ensemble method are (1) how to generate the sub-matrices and (2) how to set the ensemble weights for sub-matrices. For the first problem, we propose a heuristic method DCGASC to select sub-matrices which approximate the original matrix well. For the second problem, we adopt the kernel function to model local property and explore user preferences by the weight function.

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 sub-modularity analysis of the DCGASC objective function.
- Based on item recommendation problem, we further propose a variant method user-based LWMF, which is more reasonable for item recommendation and get better performance.
- 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 including the heuristic method DCGASC to select sub-matrices and the learning algorithm of local latent vectors. Experimental evaluations using real datasets are given in Sect. 5. Conclusion and future work are followed in Sect. 6.

2 Related Work

One of the most traditional and popular ways 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, several studies focus on using the ensemble of sub-matrices for better low-rank approximation, including DFC [5], LLORMA [7, 8], ACCAMS [9] and WEMAREC [26]. These methods partition the original matrix into several smaller sub-matrices, and a local MF is applied to each sub-matrix individually. The final predictions are obtained using the ensemble of multiple local MFs. Typically, clustering-based techniques with heuristic adaptations are used for sub-matrix generation. We give a brief review of these studies. Mackey et al. [5] introduces a Divide-Factor-Combine (DFC) framework, in which the expensive task of matrix factorization is randomly divided into smaller subproblems. LLORMA [7, 8] uses a non-parametric kernel smoothing method to search nearest neighbors; WEMAREC [26] employs Bregman co-clustering [30] techniques to partition the original matrix. 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 explicit feedbacks are not always available, while implicit feedbacks are large and common. 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). For details, Hu et al. [10] present a whole-data-based learning approach setting a uniform weight to missing entries, i.e., giving all zero entries the same weight. Pan et al. [11, 12] propose a sample-based approach which samples negative instances from missing data and adopts nonuniform weighting.

To improve the efficiency of WMF, several approaches have been proposed. Pilaszy et al. [27] design an approximate solution to ALS presenting novel and fast ALS variants both for the implicit and for the explicit feedback datasets. Recently, Devooght et al. [28] propose the randomized block coordinate descent (RCD) learner, which is a dynamic framework and reduces the complexity. Further, He et al. [24] design an algorithm based on the element-wise alternating least squares (eALS) technique to optimize a MF model with variably weighted missing data. Other related work on implicit feedback datasets is 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, for BPR framework, there are a lot of 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.

Our method employs the kernel function to intensify local property and the weight function to explore user preferences. As for parameter learning, we adopt eALS skillfully to learn the latent factors.

3 Preliminary

In this section, we present some preliminaries about basic MF, weighted MF for implicit datasets and local matrix factorization method LLORMA. A glossary of notations used in the paper is listed in Table 1. In what follows, we denote matrices by bold capital letters and sets by hand-written form. Superscripts of different forms, such as $\mathbf{R}^h$, denote different sub-matrices. Subscripts on matrices mean the indices of data. For example, $\mathbf{R}^h_{um}$ denotes the entry for the u-th user and the m-th item of the h-th data sub-matrix. In addition, row vectors are represented by having the transpose superscript$^\top$, otherwise by default they are column vectors.

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 its 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] Given a sparse matrix $\mathbf{R} \in \mathbb{R}^{N \times M}$ with indicator matrix $\mathbf{I}$, and latent factor number $K \ll \min\{N, M\}$. The aim of MF is:

$$\min_{\mathbf{P},\mathbf{Q}} \sum_{u=1}^{N} \sum_{m=1}^{M} \mathbf{I}_{um}\left(\mathbf{R}_{um} - \mathbf{P}_u^\top \mathbf{Q}_m\right)^2 \tag{1}$$

where $\mathbf{R}_{um}$ is the observed score by the u-th user for the m-th item. $\mathbf{P}_u$ and $\mathbf{Q}_m$ are the latent vectors of the u-th user and the m-th item, respectively. In order to avoid overfitting, regularization terms are usually added to the objective function to modify the squared error. So the task is to minimize $\sum_{u=1}^{N} \sum_{m=1}^{M} \mathbf{I}_{um}(\mathbf{R}_{um} - \mathbf{P}_u^\top \mathbf{Q}_m)^2 + \lambda_\mathbf{P} \|\mathbf{P}\|_F^2 + \lambda_\mathbf{Q} \|\mathbf{Q}\|_F^2$. The parameters $\lambda_\mathbf{P}$ and $\lambda_\mathbf{Q}$ are used to control the magnitudes of the latent feature matrices (*i.e.,* $\mathbf{P}$ and $\mathbf{Q}$). Stochastic gradient descent is often used to learn the parameters [4].

Table 1 Notations used in the paper

Symbols	Descriptions
N, M	Number of rows (users) and columns (items)
K	The number ($\ll \min(N, M)$) of dimensions for local latent vectors
H	The number of sub-matrices
$\mathbf{R}$	Data matrix ($\in \mathbb{R}^{N \times M}$) (with missing values)
$\mathbf{C}$	Binarized data matrix ($\in \mathbb{R}^{N \times M}$) of data matrix $\mathbf{R}$ (with missing values)
$\mathbf{W}$	The confidence weight matrix of $\mathbf{C}$
$\mathbf{P}, \mathbf{Q}$	The local latent matrix for all users (items) *w.r.t.* the data matrix $\mathbf{R}$
$\mathbf{R}^h$	The h-th data sub-matrix
$\mathbf{C}^h$	The h-th binarized data sub-matrix
$\mathbf{W}^h$	The confidence weight matrix of binarized data sub-matrix $\mathbf{C}^h$
$\mathbf{T}^h$	The sub-matrix weight matrix of binarized data sub-matrix $\mathbf{C}^h$
$\mathbf{P}^h_u, \mathbf{Q}^h_m$	The local latent vector ($\in \mathbb{R}^K$) for the u-th user (the m-th item) *w.r.t.* the data sub-matrix $\mathbf{R}^h$
$\mathcal{A}$	The data point set (nonzero user-item pair set)
$a_i = \langle u_i, m_i \rangle$	The data point $\langle u_i, m_i \rangle$ (nonzero user-item pair, $\in \mathcal{A}$)
$\hat{\mathcal{A}}$	The anchor point set ($\subset \mathcal{A}$)
$\hat{a}_h = \langle \hat{u}_h, \hat{m}_h \rangle$	The anchor point $(\hat{u}_h, \hat{m}_h)$ ($\in \hat{\mathcal{A}}$)
$E(a_i, a_j)$	The kernel value between two data points

3.2 Weighted Matrix Factorization

Hu et al. [10] and Pan et al. [11, 12] argue that original MF is applied on explicit feedback datasets, especially for rating prediction and is not suitable on implicit feedback. So they propose weighted matrix factorization (WMF) to handle the cases with implicit feedback. Recently, WMF has been widely used in TV show, music and point-of-interest recommendation. To utilize the undiscovered items and to distinguish between discovered and undiscovered items, weight matrix is added to the MF:

$$\mathbf{W}_{um} = 1 + log(1 + \mathbf{R}_{um} \times 10^{\varepsilon}) \tag{2}$$

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:

$$\min_{\mathbf{P},\mathbf{Q}} \sum_{u=1}^{N} \sum_{m=1}^{M} \mathbf{W}_{um}\left(\mathbf{C}_{um} - \mathbf{P}_u^\top \mathbf{Q}_m\right)^2 + \lambda_\mathbf{P} \|\mathbf{P}\|_F^2 + \lambda_\mathbf{Q} \|\mathbf{Q}\|_F^2 \tag{3}$$

where each entry $\mathbf{C}_{um}$ in the 0/1 matrix $\mathbf{C}$ indicates whether the u-th user has discovered the m-th item, which can be defined as a binarized matrix:

$$\mathbf{C}_{um} = \begin{cases} 1 & \mathbf{R}_{um} > 0 \\ 0 & \mathbf{R}_{um} = 0 \end{cases}. \tag{4}$$

---

[1] http://www.netflixprize.com/.

[2] http://www.kdd.org/kdd2011/kddcup.shtml.

[3] https://102.alibaba.com/competition/addDiscovery/index.htm.

3.3 Low-rank Matrix Approximation

Lee et al. [7, 8] proposed LLORMA, which 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 $\mathbf{R}$ is not low rank but a sub-matrix $\mathbf{R}^h$ is low rank. It is to say that the entire matrix $\mathbf{R}$ is composed by a set of low-rank sub-matrices $\mathcal{R} = \{\mathbf{R}^1, \mathbf{R}^2, \ldots, \mathbf{R}^H\}$ with weight matrix set $\mathcal{T} = \{\mathbf{T}^1, \mathbf{T}^2, \ldots, \mathbf{T}^H\}$ of sub-matrices, where $\mathbf{T}^h_{um}$ indicates the sub-matrix weight of $\mathbf{R}^h_{um}$ in $\mathbf{R}^h$:

$$\mathbf{R}_{um} \approx \frac{1}{\mathbf{Z}_{um}} \sum_{h=1}^{H} \mathbf{T}^h_{um} \mathbf{R}^h_{um} \tag{5}$$

where $\mathbf{Z}_{um} = \sum_{h=1}^{H} \mathbf{T}^h_{um}$. LLORMA uses the MF introduced in Sect. 3.1 to approximate the sub-matrix $\mathbf{R}^h$. If the matrix has local property, it can achieve good accuracy in predicting ratings following the paper [7].

4 Local Weighted Matrix Factorization

In this section, we introduce our proposed model LWMF and further propose a heuristic method to select sub-matrices. Finally, we adopt fast element-wise ALS to learn the local latent vectors $\mathbf{P}^h$ and $\mathbf{Q}^h$.

4.1 Our Proposed Model

Following the LLORMA, we first select sub-matrices from the original matrix, and 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 binarized sub-matrix $\mathbf{C}^h$ by WMF in Sect. 3.2 as follows:

$$\min_{\mathbf{P}_h, \mathbf{Q}_h} \sum_{u=1}^{N} \sum_{m=1}^{M} \mathbf{T}^h_{um} \mathbf{W}^h_{um} (\mathbf{C}^h_{um} - \mathbf{P}^{h\top}_u \mathbf{Q}^h_m)^2 + \lambda^h_{\mathbf{P}} \|\mathbf{P}^h\|^2_F + \lambda^h_{\mathbf{Q}} \|\mathbf{Q}^h\|^2_F \tag{6}$$

where $\lambda^h_{\mathbf{P}}$ and $\lambda^h_{\mathbf{Q}}$ are the regularization of user and item in the sub-matrix. So the original binarized Matrix $\mathbf{C}$ can be approximated by the set of sub-matrices $\mathcal{C} = \{\mathbf{C}^1, \mathbf{C}^2, \ldots, \mathbf{C}^H\}$:

$$\mathbf{C}_{um} \approx \frac{1}{\mathbf{Z}_{um}} \sum_{h=1}^{H} \mathbf{T}^h_{um} \mathbf{P}^{h\top}_u \mathbf{Q}^h_m \tag{7}$$

where $\mathbf{Z}_{um} = \sum_{h=1}^{H} \mathbf{T}^h_{um}$ is the normalizer and $\mathbf{T}^h_{um}$ indicates the weight for the entry $\mathbf{C}^h_{um}$ in the sub-matrix $\mathbf{C}^h$. Two key issues of such a sub-matrix-ensemble method are (1) how to generate the sub-matrices and (2) how to set the ensemble weights for sub-matrices.

Following LLORMA to get the sub-matrix, we firstly find a data point $a_i = \langle u_i, m_i \rangle$ in the data point set $\mathcal{A} = \{a_1, a_2, \ldots, a_{|\mathbf{R}|}\}$ as the anchor point $\hat{a}_h$. Then we calculate the relevant degree between anchor point and other data points by a similarity measure or kernel function. Finally, we choose the data points whose relevant degree is larger than a constant to compose the sub-matrix. So the data points in this selected sub-matrix are similar. In addition, we can select more anchor points to get more sub-matrices.

Actually, we use the Epanechnikov kernel to calculate the relationship between two data point pairs $a_i = (u_i, m_i)$ and $a_j = (u_j, m_j)$. It is computed as the product of user Epanechnikov kernel ($E_b(u_i, u_j)$) and item Epanechnikov kernel ($E_b(m_i, m_j)$) as follows:

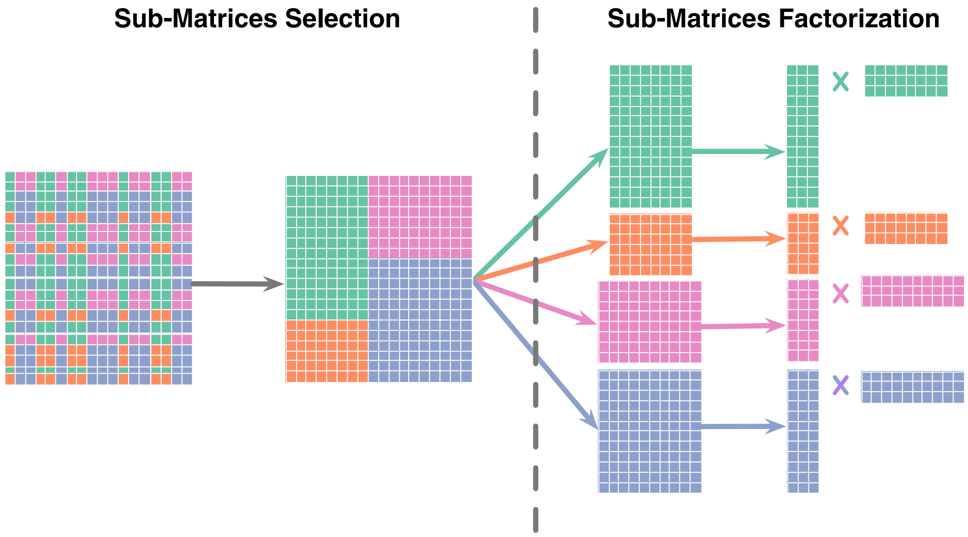


Fig. 1 Local matrix factorization

$$E(a_i, a_j) = E_b(u_i, u_j) \times E_b(m_i, m_j) \tag{8}$$

where

$$E_b(u_i, u_j) \propto (1 - d(u_i, u_j)^2)\, \mathbf{1}_{\{d(u_i,u_j) \leq b\}}$$
$$E_b(m_i, m_j) \propto (1 - d(m_i, m_j)^2)\, \mathbf{1}_{\{d(m_i,m_j) \leq b\}}$$

and b is the bandwidth parameter 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 $d(u_i, u_j) = \arccos(\frac{\mathbf{P}_{u_i} \cdot \mathbf{P}_{u_j}}{\|\mathbf{P}_{u_i}\| \cdot \|\mathbf{P}_{u_j}\|})$, where $\mathbf{P}_{u_i}$, $\mathbf{P}_{u_j}$ are the local latent vector for the u_i-th user and the u_j-th user. The distance between items is computed in the same way. So with the anchor point $\hat{a}_h$ we set the weight $\mathbf{T}^h_{u_j m_j} = E(\hat{a}_h, a_j)$ of user-item pair $\langle u_j, m_j \rangle$ for sub-matrix $\mathbf{R}^h$, the sub-matrix regularization $\lambda^h_{\mathbf{P}} = \lambda_{\mathbf{P}} E_b(\hat{u}_h, u_j)$ and $\lambda^h_{\mathbf{Q}} = \lambda_{\mathbf{Q}} E_b(\hat{m}_h, m_j)$.

Therefore, each anchor point stands for a sub-matrix. Selecting the sub-matrix set $\mathcal{C}$ is in fact to select a set of anchor points $\hat{\mathcal{A}} = \{\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_H\}$. The details of selecting anchor point set are discussed in next section.

4.2 Anchor Point Set Selection

Intuitively, the sub-matrix set $\mathcal{C} = \{\mathbf{C}^1, \mathbf{C}^2, \ldots, \mathbf{C}^H\}$ should cover the original matrix $\mathbf{C}$, that is $\mathbf{C} = \cup_{\mathbf{C}^h \in \mathcal{C}} \mathbf{C}^h$, so that these sub-matrix sets $\mathcal{C}$ can approximate the original matrix $\mathbf{C}$ better than the set that does not cover. Therefore, the anchor points selection problem can be reduced to the set cover problem.

4.2.1 Anchor Point Set Cover (ASC)

We treat all the nonzero user-item pairs, *i.e.,* data point set $\mathcal{A} = \{a_1, a_2, \ldots, a_{|\mathbf{R}|}\}$ as the candidate anchor point set. Every candidate point a_i can cover itself several other candidate points denoted by $\mathcal{A}^i = \{a_i, a_{i1}, a_{i2}, \ldots, a_{iD}\} \subset \mathcal{A}$. Then, we propose the naive anchor points cover method, called Anchor Point Set Cover that returns an anchor point set $\hat{\mathcal{A}} \subset \mathcal{A}$ such that

$$\begin{aligned} &\max J(\hat{\mathcal{A}}) = | \cup_{i \in \hat{\mathcal{A}}} \mathcal{A}^i | \\ &\quad s.t. |\hat{\mathcal{A}}| = H \end{aligned} \tag{9}$$

Obviously, the ASC problem is sub-modular and monotone [25]. So the greedy algorithm can achieve $1 - \frac{1}{e}$ approximation ratio of the optimized result.

4.2.2 Discounted Cumulative Gain Anchor Point Set Cover (DCGASC)

However, set cover problem only needs to cover a point only once while covering all training data only once is not enough. Covering the training data more times is also helpful for the final recommendation. 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) [22] and ERR (expected reciprocal rank) [21] in IR(information retrieval). The premise of NDCG and ERR is that highly relevant documents appearing lower in a search result list should be penalized as the graded relevance value is reduced proportional to the position of the result. Learning from this discounted approach, we propose a heuristic method to model this situation, called Discounted Cumulative Gain Anchor Point Set Cover (DCGASC) that returns an anchor point order list $\hat{\mathcal{A}} = \{\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_H\} \subset \mathcal{A}$ such that

$$\begin{aligned} &\max J(\hat{\mathcal{A}}) = \sum_{h=1}^{H} \sum_{a_l \in \hat{\mathcal{A}}^h} \alpha^{o_{lh}-1} (1 - \max\nolimits_{h' \in \{1,\ldots,h-1\}} E_b(\hat{a}_h, \hat{a}_{h'})) \\ &s.t. |\hat{\mathcal{A}}| = H \end{aligned} \tag{10}$$

where o_{lh} denotes the covered times of a_l by the selected anchor points $\{\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_h\}$. $\alpha \in (0, 1)$ is the discount parameter. When point a_l has been covered by a anchor point before, the covered gain will be reduced next time. When $\alpha = 0$, this problem reduces to the set cover problem. And when $\alpha = 1$, it just gets the anchor point which covers the other points at most every time. The $(1 - \max_{h' \in \{1,\ldots,h-1\}} E_b(\hat{a}_h, \hat{a}_{h'}))$ term means DCGASC tends to select the point which is far from the selected anchor points. Below we prove that $J(\cdot)$ is sub-modular and monotone.

Theorem 1 *DCGASC function is sub-modular and also monotone nondecreasing.*

Proof Let $\mathcal{S} = \{\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_{H-1}\}$ and $\mathcal{V} = \{\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_{H-1}, \ldots, \hat{a}_{X-1}\}$ are the anchor point sets, $X \geq H$ and $a_i = \hat{a}_X \in \mathcal{A} \backslash \mathcal{V}$ is the next selected anchor point. We have that

$$\begin{aligned} &J(\mathcal{V} \cup \{\hat{a}_X\}) - J(\mathcal{V}) \\ &\quad = \sum_{h=1}^{X} \sum_{a_l \in \hat{\mathcal{A}}^h} \alpha^{o_{lh}-1} (1 - \max\nolimits_{h' \in \{1,\ldots,h-1\}} E_b(\hat{a}_h, \hat{a}_{h'})) \\ &\quad\quad - \sum_{h=1}^{X-1} \sum_{a_l \in \hat{\mathcal{A}}^h} \alpha^{o_{lh}-1} (1 - \max\nolimits_{h' \in \{1,\ldots,h-1\}} E_b(\hat{a}_h, \hat{a}_{h'})) \\ &\quad = \sum_{a_l \in \mathcal{A}^X} \alpha^{o_{lX}-1} (1 - \max\nolimits_{h' \in \{1,\ldots,X-1\}} E_b(\hat{a}_X, \hat{a}_{h'})) \geq 0 \end{aligned} \tag{11}$$

and

$$\begin{aligned} &J(\mathcal{S} \cup \{\hat{a}_X\}) - J(S) - (J(\mathcal{V} \cup \{\hat{a}_X\}) - J(\mathcal{V})) = \\ &\quad = \sum_{a_l \in \mathcal{A}^X} \alpha^{o_{lX'}-1} (1 - \max\nolimits_{h' \in \{1,\ldots,H-1\}} E_b(\hat{a}_X, \hat{a}_{h'})) \\ &\quad\quad - \sum_{a_l \in \mathcal{A}^X} \alpha^{o_{lX}-1} (1 - \max\nolimits_{h' \in \{1,\ldots,X-1\}} E_b(\hat{a}_X, \hat{a}_{h'})) \end{aligned} \tag{12}$$

where $o_{lX'}$ means the covered times of a_l by the anchor points $S \cup \{\hat{a}_X\}$. Because the number of anchor points covered satisfies that $o_{lX'} \leqslant o_{lX}$, discount parameter $\alpha \in [0,1]$ and $\max_{h' \in \{1,\ldots,H-1\}} E_b(\hat{a}_X, \hat{a}_{h'}) \le \max_{h' \in \{1,\ldots,X-1\}} E_b(\hat{a}_X, \hat{a}_{h'})$, we know that $J(\mathcal{S} \cup \{\hat{a}_X\}) - J(\mathcal{S}) - (J(\mathcal{V} \cup \{\hat{a}_X\}) - J(\mathcal{V})) \geq 0$. Therefore, it is proved that the DCGASC function is monotone and sub-modular.

Due to the monotonicity and sub-modularity of DCGASC function, the greedy algorithm 1 can provide a theoretical approximation guarantee of factor $1 - \frac{1}{e}$ as described in [23]. Algorithm 1 shows the greedy algorithm: It first obtains the anchor point which cover other points at most and then uses Eq. (12) to get the following anchor points in turn.

Algorithm 1: `DCGASC Greedy Algorithm`

```
Input : Set of data points $\mathcal{A}$, anchor number $H$,
        DCGASC function $f$ and sets $A^i$ covered by
        each data point $a_i$
Output: An anchor point order list $\hat{\mathcal{A}} \subseteq \mathcal{A}$ with
        $|\hat{\mathcal{A}}| = H$
$\hat{a}_1 \leftarrow \arg\max_{a_i \in \mathcal{A}} |\mathcal{A}^i|$;
$\hat{\mathcal{A}} \leftarrow \{\hat{a}_1\}$;
for $h$ from 2 to $H$ do
    $\hat{a}_h \leftarrow \arg\max_{a'_i \in \mathcal{A} \setminus \hat{\mathcal{A}}} f(\hat{\mathcal{A}} \cup \{a'_i\}) - f(\hat{\mathcal{A}})$
    $\hat{\mathcal{A}} \leftarrow \hat{\mathcal{A}} \cup \{\hat{a}_h\}$
end
return $\hat{\mathcal{A}}$
```

4.3 Learning Algorithm

Alternating least square (ALS) is a popular approach to optimize weighted matrix factorization [10]. [24] proposed a fast element-wise ALS learning algorithm which optimizes each coordinate of the latent vector with the other fixed ones and speeds up learning by avoiding the massive repeated computations introduced by the weighted missing data. In this paper, we use the element-wise ALS learning algorithm to learn the sub-matrix latent vectors. More specifically, the latent factors of the u-th user are updated based on

$$\mathbf{P}^h_{uk} = \frac{\sum_{m \in \mathcal{M}^h} \left(\mathbf{C}_{um} - \hat{\mathbf{C}}^h_{um,k}\right) \mathbf{T}^h_{um} \mathbf{W}_{um} \mathbf{Q}^h_{mk}}{\sum_{m \in \mathcal{M}^h} \mathbf{T}^h_{um} \mathbf{W}_{um} \mathbf{Q}^h_{mk} \mathbf{Q}^h_{mk} + \lambda^h_{\mathbf{P}}} \tag{13}$$

where $\mathcal{M}^h$ denotes item indices set in the h-th sub-matrix, *i.e.,* the prediction without the component of latent factor k $\hat{\mathbf{C}}^h_{um,k} = \hat{\mathbf{C}}^h_{um} - \mathbf{P}^h_{uk} \mathbf{Q}^h_{mk}$, where $\hat{\mathbf{C}}^h_{um}$ is the predict score. Noted that $\mathbf{C}_{um}$ and $\mathbf{W}_{um}$ are all the same in the different sub-matrices. The sub-matrix weight $\mathbf{T}^h_{um}$ is the only difference in Eq. (13) with the original WMF, which may lead to high running time. Fortunately, due to $\mathbf{T}^h_{um} = E_b(\hat{u}_h, u) \times E_b(\hat{m}_h, m)$ and $\lambda^h_{\mathbf{P}} = \lambda_{\mathbf{P}} E_b(\hat{u}_h, u)$, we also can speed up learning by memorizing the massive repeated computations. Firstly, $E_b(\hat{u}_h, u)$ is both in the numerator and in the denominator so it can be canceled. Noted that if $E_b(\hat{u}_h, u) = 0$, it does not need to calculate the latent vector $\mathbf{P}^h_u$. Then, we focus on the numerator:

$$\begin{aligned}
&\sum_{m \in \mathcal{M}^h} \left(\mathbf{C}_{um} - \hat{\mathbf{C}}^h_{um,k}\right) E_b(\hat{m}_h, m) \mathbf{W}_{um} \mathbf{Q}^h_{mk} \\
&\quad = \sum_{m \in \mathcal{M}^h_u} \left[\mathbf{W}_{um} \mathbf{C}_{um} - (\mathbf{W}_{um} - 1) \hat{\mathbf{C}}^h_{um,k}\right] E_b(\hat{m}_h, m) \mathbf{Q}^h_{mk} \\
&\quad - \sum_{m \in \mathcal{M}^h} E_b(\hat{m}_h, m) \hat{\mathbf{C}}^h_{um,k} \mathbf{Q}^h_{mk}
\end{aligned} \tag{14}$$

where $\mathcal{M}^h_u$ means the set of items discovered by the u-th user in the h-th sub-matrix. Because $E_b(\hat{m}_h, m)$ is the same for different users, the cache method can also be utilized here. The $\sum_{m \in \mathcal{M}^h} E_b(\hat{m}_h, m) \hat{\mathbf{C}}^h_{um,k} \mathbf{Q}^h_{mk}$ term can be speeded up:

$$\begin{aligned}
&\sum_{m \in \mathcal{M}^h} E_b(\hat{m}_h, m) \hat{\mathbf{C}}^h_{um,k} \mathbf{Q}^h_{mk} \\
&\quad = \sum_{f \neq k} \mathbf{P}_{uf} \sum_{m \in \mathcal{M}^h} E_b(\hat{m}_h, m) \mathbf{Q}^h_{mk} \mathbf{Q}^h_{mf}
\end{aligned} \tag{15}$$

So the $\sum_{m \in \mathcal{M}^h} E_b(\hat{m}_h, m) \mathbf{Q}^h_{mk} \mathbf{Q}^h_{mf}$ can be pre-computed and used in updating the latent vectors for all users. Similarly, the same cache method can be used in the calculation of denominator. We define the $\mathbf{S}^{\mathbf{Q}^h}$ as $\mathbf{S}^{\mathbf{Q}^h} = \sum_{m \in \mathcal{M}^h} E_b(\hat{m}_h, m) \mathbf{Q}^h_m \mathbf{Q}^{h\top}_m$, so Eq. (13) can be calculated as:

$$\begin{aligned}
\mathbf{P}^h_{uk} = &\left\{ \sum_{m \in \mathcal{M}^h_u} \left[\mathbf{W}_{um} \mathbf{C}_{um} - (\mathbf{W}_{um} - 1) \hat{\mathbf{C}}^h_{um,k}\right] E_b(\hat{m}_h, m) \mathbf{Q}^h_{mk} \right. \\
&\left. - \sum_{f \neq k} \mathbf{P}^h_{uf} \mathbf{S}^{\mathbf{Q}^h}_{fk} \right\} \Big/ \left\{ \sum_{m \in \mathcal{M}^h_u} E_b(\hat{m}_h, m) (\mathbf{W}_{um} - 1) \mathbf{Q}^h_{mk} \mathbf{Q}^h_{mk} \right. \\
&\left. + \mathbf{S}^{\mathbf{Q}^h}_{kk} + \lambda_{\mathbf{P}} \right\}
\end{aligned} \tag{16}$$

where $\mathbf{S}^{\mathbf{Q}^h}_{fk}$ is the (f, k)-th element of the $\mathbf{S}^{\mathbf{Q}^h}$. Similarly, we define the $\mathbf{S}^{\mathbf{P}^h}$ as $\mathbf{S}^{\mathbf{P}^h} = \sum_{u \in \mathcal{U}^h} E_b(\hat{u}_h, u) \mathbf{P}^h_u \mathbf{P}^{h\top}_u$ and the update of item latent vectors is:

$$\begin{aligned}
\mathbf{Q}^h_{mk} = &\left\{ \sum_{u \in \mathcal{U}^h_m} \left[\mathbf{W}_{um} \mathbf{C}_{um} - (\mathbf{W}_{um} - 1) \hat{\mathbf{C}}^h_{um,k}\right] E_b(\hat{u}_h, u) \mathbf{P}^h_{uk} \right. \\
&\left. - \sum_{f \neq k} \mathbf{Q}^h_{mf} \mathbf{S}^{\mathbf{P}^h}_{fk} \right\} \Big/ \left\{ \sum_{u \in \mathcal{U}^h_m} E_b(\hat{u}_h, u) (\mathbf{W}_{um} - 1) \mathbf{P}^h_{uk} \mathbf{P}^h_{uk} \right. \\
&\left. + \mathbf{S}^{\mathbf{P}^h}_{kk} + \lambda_{\mathbf{Q}} \right\}
\end{aligned} \tag{17}$$

So with the local sub-matrix weights, one iteration takes $O(NK^2 + MK^2 + |\mathbf{R}|K)$ time as the same as the fast element-wise ALS [24].

Algorithm 2: LWMF Learning Algorithm

```
Input : data matrix $\mathbf{R}$, anchor number $H$, DCGASC
        function $J$ and sets $\mathcal{A}^i$ covered by each data
        point $a_i$,W,$\lambda_\mathbf{P}$ and $\lambda_\mathbf{Q}$, the number $K$ of
        dimensions for latent vectors
Output: Latent featue matrix sets
        $\mathcal{P} = \{\mathbf{P}^1, \mathbf{P}^2, ..., \mathbf{P}^H\}$ and
        $\mathcal{Q} = \{\mathbf{Q}^1, \mathbf{Q}^2, ..., \mathbf{Q}^H\}$ and the sub-matrix
        weight matrices $\mathcal{T} = \mathbf{T}^1, \mathbf{T}^2, ..., \mathbf{T}^H$
1  Use Eq. 4 to calculate binarized data matrix $\mathbf{C}$ from
   original data matrix $\mathbf{R}$;
2  Use fast element-wise ALS [24] to learn the whole
   latent vectors $\mathbf{P}$ and $\mathbf{Q}$;
3  Use algorithm. 1 to get anchor set
   $\hat{\mathcal{A}} = \{\hat{a}_1, \hat{a}_2, ..., \hat{a}_H\}$;
4  for $h \leftarrow 1$ to $H$ do
5      for $u \leftarrow 1$ to $N$ do
6          if $E_b(\hat{u}_h, u) > 0$ then
7              $\mathcal{U}^h \leftarrow \mathcal{U}^h \cup \{u\}$
8          end
9      end
10     for $m \leftarrow 1$ to $M$ do
11         if $E_b(\hat{m}_h, m) > 0$ then
12             $\mathcal{M}^h \leftarrow \mathcal{M}^h \cup \{m\}$
13         end
14     end
15     //update local user latent vectors
       $\mathbf{S}^{\mathbf{Q}^h} = \sum_{m \in \mathcal{M}^h} E_b(\hat{m}_h, m)\mathbf{Q}^h_m \mathbf{Q}^{h\top}_m$;
16     for $u \in \mathcal{U}^h$ do
17         foreach $m \in \mathcal{M}^h_u$ do $\hat{\mathbf{C}}^h_{um} \leftarrow \mathbf{P}^{h\top}_u \mathbf{Q}^h_m$
18         for $k \leftarrow 1$ to $K$ do
19             foreach $m \in \mathcal{M}^h_u$ do
20                 $\hat{\mathbf{C}}^h_{um,k} \leftarrow \hat{\mathbf{C}}^h_{um} - \mathbf{P}^h_{uk}\mathbf{Q}^h_{mk}$
21             calculate $\mathbf{P}^h_{uk}$ using Eq. 16;
22             foreach $m \in \mathcal{M}^h_u$ do
23                 $\hat{\mathbf{C}}^h_{um,k} \leftarrow \hat{\mathbf{C}}^h_{um} + \mathbf{P}^h_{uk}\mathbf{Q}^h_{mk}$
24         end
25     end
26     //update local item latent vectors
       $\mathbf{S}^{\mathbf{P}^h} = \sum_{u \in \mathcal{U}^h} E_b(\hat{u}_h, u)\mathbf{P}^h_u \mathbf{P}^{h\top}_u$;
27     for $m \in \mathcal{M}^h$ do
28         foreach $u \in \mathcal{U}^h_m$ do $\hat{\mathbf{C}}^h_{um} \leftarrow \mathbf{P}^{h\top}_u \mathbf{Q}^h_m$
29         for $k \leftarrow 1$ to $K$ do
30             foreach $u \in \mathcal{U}^h_m$ do
31                 $\hat{\mathbf{C}}^h_{um,k} \leftarrow \hat{\mathbf{C}}^h_{um} - \mathbf{P}^h_{uk}\mathbf{Q}^h_{mk}$
32             calculate $\mathbf{Q}_{mk}$ using Eq. 17;
33             foreach $u \in \mathcal{U}^h_m$ do
34                 $\hat{\mathbf{C}}^h_{um,k} \leftarrow \hat{\mathbf{C}}^h_{um} + \mathbf{P}^h_{uk}\mathbf{Q}^h_{mk}$
35         end
36     end
37 end
38 return $\mathbf{P}, \mathbf{Q}$
```

Algorithm 2 summarizes the process of learning local weighted latent vectors. First, we use the fast element-wise ALS [23] to learn the global latent vectors (Line 1). Then we obtain the anchor set by Algorithm 1. At last, we adopt the fast element-wise ALS to learning every sub-matrix latent vectors.

4.4 User-based Local Weighted Matrix Factorization

The above method LWMF uses the selected sub-matrices to model the local property and ignore global feature. Especially for the item recommendation problem, we should recommend items for a user from all the items. So we propose a variant method, called User-based Local Weighted Matrix Factorization, which only considers users to select the anchor points and puts all items into the sub-matrix. Given the user set $\mathcal{U} = \{u_1, u_2, \ldots, u_N\}$ (all users) while every user u_i can cover itself several other users denoted by $\mathcal{U}^i = \{u_i, u_{i1}, u_{i2}, \ldots, u_{iD}\}$, we need to find user anchor set $\hat{\mathcal{U}} = \{\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_H\}$ to maximize the user anchor set cover function:

$$\begin{aligned} &\max J(\hat{\mathcal{U}}) \\ &= \sum_{h=1}^{H} \sum_{u_l \in \mathcal{U}^h} \alpha^{o_{lh}-1}\left(1 - \max_{h' \in \{1,\ldots,h-1\}} E_b(\hat{u}_h, \hat{u}_{h'})\right) \\ &\quad s.t. |\hat{\mathcal{U}}| = H \end{aligned} \tag{18}$$

Obviously, this user-based DCGASC function is also submodular and also monotone nondecreasing. Figure 2 shows the user-based LWMF to select the sub-matrices. Because we do not need to consider the items, it is much faster to the user anchor point set. Moreover, user-based LWMF is more reasonable for item recommendation problem. As a direct comparison of user-based LWMF, we also implement item-based LWMF, which only considers items to select the anchor points and lets all users into the sub-matrix.

5 Experiments

In this section, we evaluate the method proposed in this paper using real datasets. We first introduce the datasets

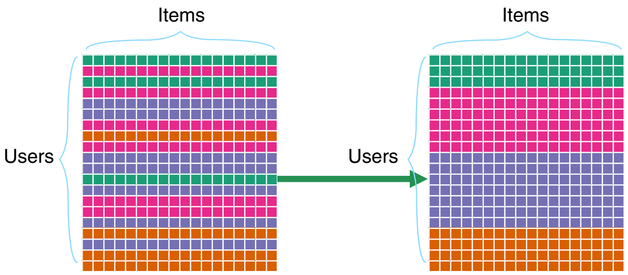


Fig. 2 User-based local matrix factorization

and experimental settings. Then, we compare our method with WMF under specific parameter settings. We also compare results with different anchor numbers and two anchor points selection methods.

5.1 Dataset

We choose two real-world datasets from [29]. One is the Foursquare check-in data made in Singapore between August 2010 and July 2011, and another is the Gowalla check-in data made in California and Nevada between February 2009 and October 2010. Both are popular online LBSNs datasets.

The Foursquare check-in data comprises 194,108 check-ins made by 2312 users at 5596 POIs, and the density is 1.50×10^{-2}. The Gowalla check-in data comprises 456,967 check-ins made by 10,162 users at 24,238 POIs, and the density is 1.86×10^{-3}. Two datasets are very sparse (Table 2).

More details about two datasets are listed in Table 3. We randomly select 80% of each user's visiting locations as the training set and the rest 20% as the testing set.

Table 3 Detail information of Gowalla and Foursquare

	Foursquare	Gowalla
#users	2321	10,162
#locations	5596	24,238
#check-ins	194,108	456,967
avg. #users per loc	34.69	18.85
avg. #loc. per user	83.63	44.97
max #users per loc	695	2,195
max #loc. per user	311	1,113

5.2 Setting

Next, we show the parameter values. The regularization λ is set to 10, and the performance of recommendation is not sensitive to this parameter. The weight parameter ε for Fousquare is set to 2 and for Gowalla is set to 3. We set the bandwidth parameter in Epanechnikov kernel as $b = 0.8$. The discount α of DCGASC is set to 0.4. We select 100 anchor points for both datasets. 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 the u-th user, we set $\mathcal{I}_u^P$ as the predicted item list and $\mathcal{I}_u^T$ as the true list in the testing dataset. So the Precision@n and Recall@n are:

$$\text{Precision@}n = \frac{1}{N}\sum_{u=1}^{N}\frac{|\mathcal{I}_u^P \bigcap \mathcal{I}_u^T|}{n}$$

$$\text{Recall@}n = \frac{1}{N}\sum_{u=1}^{N}\frac{|\mathcal{I}_u^P \bigcap \mathcal{I}_u^T|}{|\mathcal{I}_u^T|}$$

where $|\mathcal{I}_u^P| = n$. In our base experiments, we choose top 10 as evaluation metrics.

We compare seven methods for implicit feedback datasets:

Table 2 Precision and recall comparison on Foursquare and Gowalla, where column "improve" indicates the relative improvements that our approach LWMF achieves relative to the basic WMF results

ALL	Metrics	MP	KNN$_u$	KNN$_m$	WMF	LWMF$_{both}$	LWMF$_m$	LWMF$_u$	Improve (%)
Foursquare	Precision	0.0615	0.0741	0.0698	0.0792	0.0823	**0.0869**	0.0852	9.80
$d = 5$	Recall	0.0680	0.8212	0.7975	0.0905	0.0952	0.0962	**0.0999**	10.34
$d = 10$	Precision	0.0615	0.0741	0.0698	0.0847	0.0847	0.0878	**0.0898**	6.03
	Recall	0.0680	0.8212	0.7975	0.0993	0.0995	0.0990	**0.1047**	5.44
$d = 20$	Precision	0.0615	0.0741	0.0698	0.0844	0.0832	0.0893	**0.0915**	8.39
	Recall	0.0680	0.8212	0.7975	0.0980	0.0982	0.1021	**0.1067**	8.85
$d = 40$	Precision	0.0615	0.0741	0.0698	0.0741	0.0828	**0.0907**	**0.0902**	22.45
	Recall	0.0680	0.8212	0.7975	0.0922	0.0945	0.1028	**0.1054**	14.27
Gowalla	Precision	0.0203	0.0552	0.0587	0.0321	0.0489	0.0478	0.0445	52.56
$d = 5$	Recall	0.0460	**0.1055**	0.1014	0.0664	0.0923	0.0884	0.0881	39.05
$d = 10$	Precision	0.0203	0.0552	**0.0587**	0.0385	0.0528	0.0526	0.0504	37.10
	Recall	0.0460	**0.1055**	0.1014	0.0779	0.0990	0.0936	0.0989	27.01
$d = 20$	Precision	0.0203	0.0552	**0.0587**	0.0442	0.0558	0.0565	**0.0581**	31.44
	Recall	0.0460	0.1055	0.1014	0.0871	0.1035	0.1006	**0.1110**	27.41
$d = 40$	Precision	0.0203	0.0552	0.0587	0.0485	0.0578	0.0584	**0.0623**	28.36
	Recall	0.0460	0.1055	0.1014	0.0953	0.1067	0.1034	**0.1191**	25.04

Bold values indicate the best performance among all the methods

- Most popular: This is the most basic method, which recommends the most popular items to the target user.
- KNN$_u$: This is user-based CF method, where user-user similarity is calculated based on the training data.
- KNN$_m$: This method is similar to KNN$_u$, and the difference is that KNN$_m$ calculates item-item similarity based on the training data. Specifically, we set the neighbor numbers in KNN$_u$ and KNN$_m$ to 100.
- WMF: This is the state-of-the-art method, which is a whole-data-based learning approach setting a uniform weight to missing entries [10, 24].
- LWMF$_{\mathrm{both}}$: This is our proposed method that employs the kernel function to intensify local property and the weight function to explore user preferences.
- LWMF$_u$: A variant method of LWMF$_{\mathrm{both}}$ which only considers users to select the anchor points and puts all items into the sub-matrix.
- LWMF$_m$: A variant method of LWMF$_{\mathrm{both}}$ which only considers items to select the anchor points and puts all users into the sub-matrix.

Then, we compare two anchor points selection methods to study the performance of LWMF:

- Random: Sampling anchor points uniformly from training dataset as paper [7] does.
- Discounted Cumulative Gain Anchor Set Cover (DCGASC): Discounting cumulative gain of covering the points which is also sub-modular and monotone.

So LWMF can be expanded into two sub-methods: LWMF_Random and LWMF_DCGASC. By default, LWMF means LWMF_DCGASC. Each method is conducted five times independently. Therefore, the average score indicates the performance of the recommendation methods.

5.3 Experimental Results

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

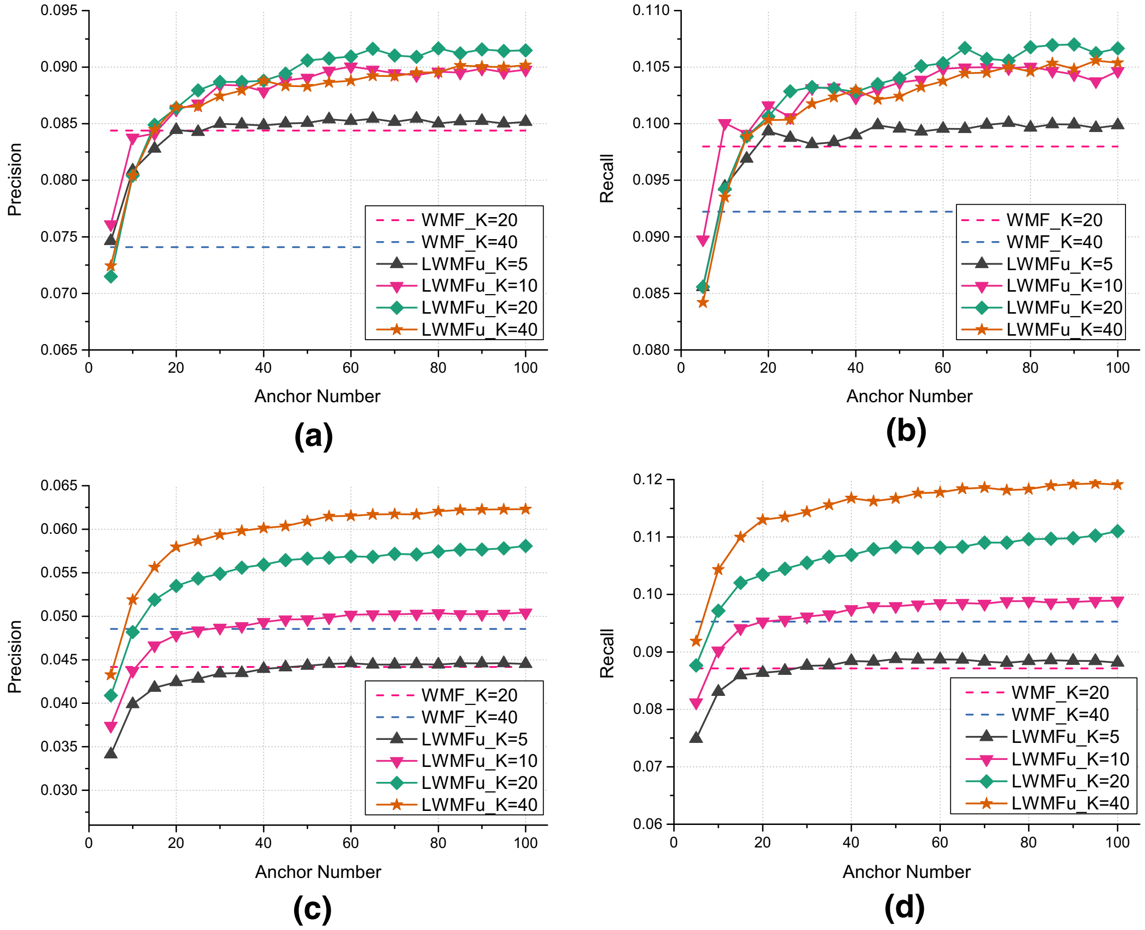


Fig. 3 Comparison with different number of anchor points. **a** Precision on Foursquare, **b** recall on Foursquare, **c** precision on Gowalla, **d** recall on Gowalla

5.3.1 Recommendation Methods Comparison

Table 2 lists the precision and recall of seven methods mentioned above on Foursquare and Gowalla datasets. It shows the same result as [7] that LORMA outperforms SVD, and LWMF always outperforms WMF. The performances of WMF and LWMF are increasing with the increase of K. However, on Foursquare, when K gets to 40, the performances both fall, which indicates that the value of K has resulted in overfitting. So we choose K to be 20. On the other hand, the experiments based on Gowalla dataset show that the value of K is bigger than 40 when the performance is best. It is obvious that performances of all LWMF methods are better than WMF methods in all dimensions. Especially on Gowalla dataset, the precision and recall of LWMF are more than 25% better than WMF. Specifically when K equals to 5, the precision of LWMF_{both} are 52.56% better than WMF. More obvious improvements on Foursquare and 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. Additionally as for our three approaches, we can find that the differences between their performances are not very obvious. But from an overall view, LWMF_u are better than the other two methods. LWMF_u does the recommendation task based on users, so it can be inferred that selecting points based on users are more reasonable than the other two methods. We also do the comparison of three basic methods, which are MostPopular, KNN_u and KNN_m. The experimental results indicate that our methods are better than these three basic methods. Although KNN_u and KNN_m are better than LWMF when K is low on Gowalla, the performance of LWMF goes up with the increase of K and is far more better than KNN_u and KNN_m.

5.3.2 Comparison with Different Number of Anchor Points

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 K increases and LWMF performs better than WMF with $K \geq 20$. For

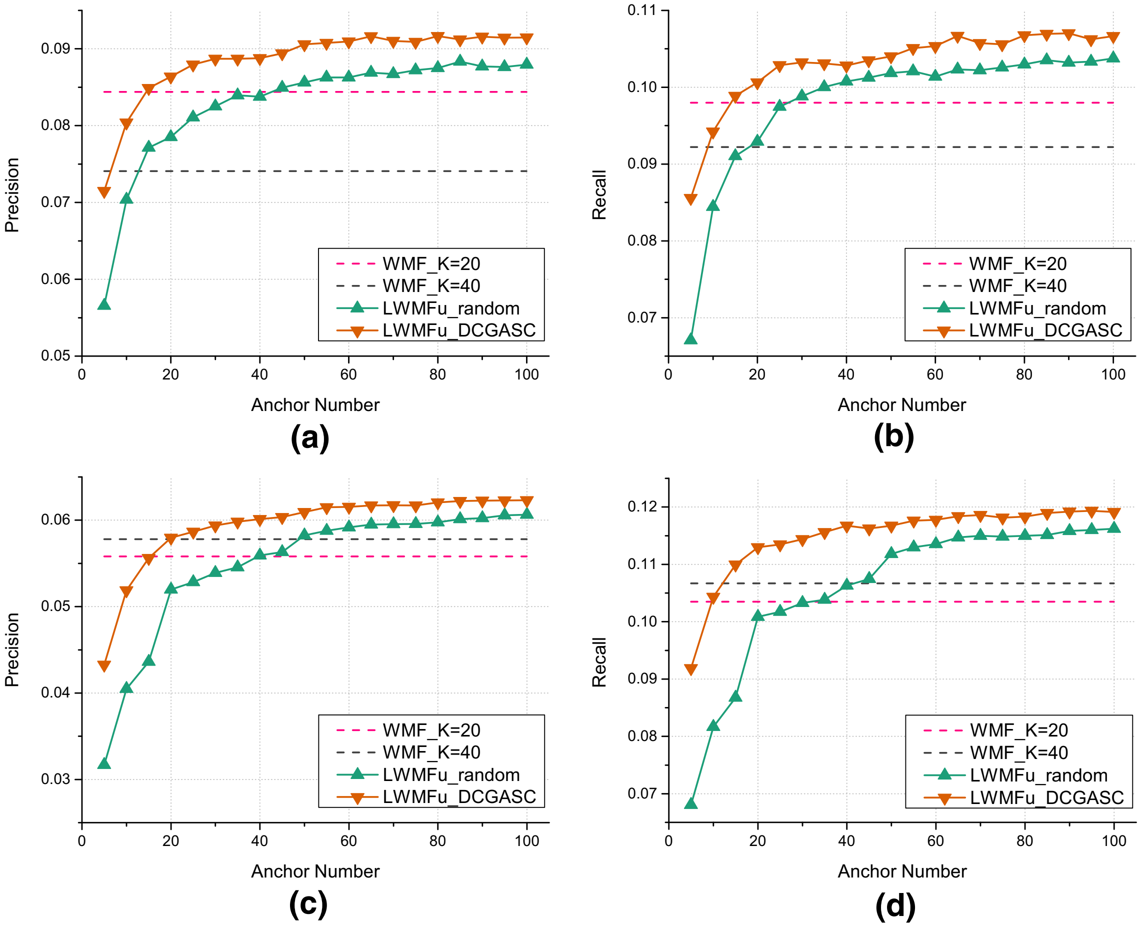


Fig. 4 Anchor point set selection methods comparison. **a** Precision on Foursquare, **b** recall on Foursquare, **c** precision on Gowalla, **d** recall on Gowalla

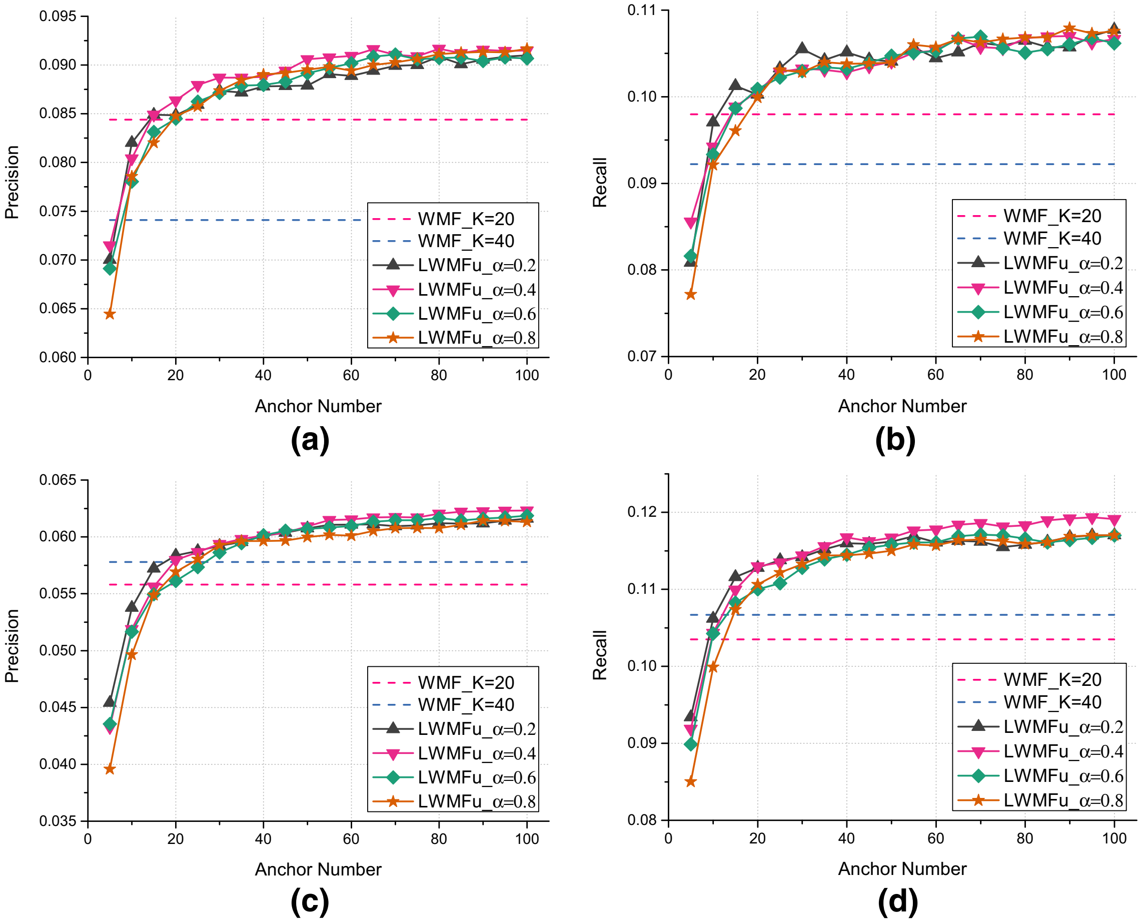
Foursquare dataset, LWMF with $K = 20$ and anchor number $H \geq 20$ outperforms WMF with $K = 20$, while the same performance on Gowalla dataset needs $H \geq 40$ anchor points. We can see that as the number of anchor points increases, the performance gets better. When the number of anchor points gets to 50, we can get a good performance. 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 $O(NK^2 + MK^2 + |\mathbf{R}|K)$ 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.

5.3.3 Anchor Point Set Selection Methods Comparison

Next, we compare the performance of LWMF$_u$_Random and LWMF$_u$ in Fig. 4. The discount parameter α is set 0.4. K is set to 20 for Foursquare dataset, while 40 for Gowalla dataset. From Fig. 4, when the number of anchor points is small, LWMF$_u$ performs better in precision and recall. When the number of anchor points increases, the gap of performance among three gets less. Despite this, LWMF$_u$ outperforms LWMF$_u$_Random on both datasets.

5.3.4 Comparison with Different Discounts for DCGASC

Finally, we study the performance of LWMF$_u$ with different discount parameters. K is set to 20 for Foursquare dataset, while 40 for Gowalla dataset. 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 $\alpha \in \{0.2, 0.4, 0.6, 0.8\}$ in Fig. 5. The gap of performance with four discount parameters is small. The performance with discount parameter $\alpha = 0.4$ is better slightly. In general, the performance of LWMF is not sensitive to the discount parameter but mainly depends on the number of anchor points.



(a) **(b)** **(c)** **(d)**

Fig. 5 Comparison with different discounts of anchor points. **a** Precision on Foursquare, **b** recall on Foursquare, **c** precision on Gowalla, **d** recall on Gowalla

6 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 will 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.

Acknowledgements This work was supported by National Natural Science Foundation of China Grants (Nos. 61472141, U1501252 and 61021004), Shanghai Knowledge Service Platform Project (No. ZF1213), Shanghai Leading Academic Discipline Project (Project Number: B412) and Shanghai Agriculture Science Program (2015) Number 3-2.

References

1. Deshpande M, Karypis G (2004) Item-based top-n recommendation algorithms. ACM Trans Inf Syst 22(1):143–177
2. Paterek A (2007) Improving regularized singular value decomposition for collaborative filtering. In: Proceedings of KDD cup and workshop, vol 2007, pp 5–8
3. Salakhutdinov R, Mnih A (2011) Probabilistic matrix factorization. In: Shawe-Taylor J, Zemel RS, Bartlett PL, Pereira F, Weinberger KQ (eds) Advances in neural information processing systems, vol 20. Curran Associates Inc, Vancouver, BC, Canada, pp 1257–1264
4. Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. Computer 42(8):30–37
5. Mackey LW, Jordan MI, Talwalkar A (2011) Divide-and-conquer matrix factorization. In: Platt JC, Koller D, Singer Y, Roweis ST (eds) Advances in neural information processing systems, vol 24. Curran Associates Inc, Granada, Spain, pp 1134–1142
6. Zhang Y, Zhang M, Liu Y, et al (2013) Localized matrix factorization for recommendation based on matrix block diagonal forms. In: Proceedings of the 22nd international conference on world wide web, pp 1511–1520
7. Lee J, Kim S, Lebanon G, et al (2013) Local low-rank matrix approximation. In: Proceedings of the 30th international conference on machine learning, pp 82–90
8. Lee J, Bengio S, Kim S, et al (2014) Local collaborative ranking. In: Proceedings of the 23rd international conference on world wide web, pp 85–96
9. Beutel A, Ahmed A, Smola AJ (2015) ACCAMS: Additive co-clustering to approximate matrices succinctly. In: Proceedings of the 24th international conference on world wide web, pp 119–129
10. Hu Y, Koren Y, Volinsky C (2008) Collaborative filtering for implicit feedback datasets. In: Eighth IEEE international conference on data mining, pp 263–272
11. Pan R, Zhou Y, Cao B, et al (2008) One-class collaborative filtering. In: Eighth IEEE international conference on data mining, pp 502–511
12. Pan R, Scholz M (2009) 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
13. Rendle S, Freudenthaler C, Gantner Z, et al (2009) BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence, pp 452–461
14. Rendle S, Freudenthaler C (2014) Improving pairwise learning for item recommendation from implicit feedback. In: Proceedings of the 7th ACM international conference on web search and data mining, pp 273–282
15. Yang D, Chen T, Zhang W, et al (2012) Local implicit feedback mining for music recommendation. In: Proceedings of the sixth ACM conference on recommender systems, pp 91–98
16. Ilievski I, Roy S (2013) Personalized news recommendation based on implicit feedback. In: Proceedings of the 2013 international news recommender systems workshop and challenge, pp 10–15
17. Zibriczky D, Hidasi B, Petres Z, et al (2012) Personalized recommendation of linear content on interactive TV platforms: beating the cold start and noisy implicit user feedback. In: UMAP workshops
18. Lian D, Zhao C, Xie X, et al (2014) 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
19. Liu Y, Wei W, Sun A, et al (2014) Exploiting geographical neighborhood characteristics for location recommendation. In: Proceedings of the 23rd ACM conference on information and knowledge management, pp 739–748
20. Cho E, Myers SA, Leskovec J (2011) 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
21. Chapelle O, Metlzer D, Zhang Y, et al (2009) Expected reciprocal rank for graded relevance. In: Proceedings of the 18th ACM conference on information and knowledge management, pp 621–630
22. Clarke CLA, Kolla M, Cormack GV, et al (2008) Novelty and diversity in information retrieval evaluation. In: Proceedings of the 31st annual international ACM SIGIR conference on research and development in information retrieval, pp 659–666
23. Nemhauser GL, Wolsey LA, Fisher ML (1978) An analysis of approximations for maximizing submodular set functions I. Math Program 14(1):265–294
24. He X, Zhang H, Kan MY, et al (2016) Fast matrix factorization for online recommendation with implicit feedback. In: Proceedings of the 39th annual international ACM SIGIR conference on research and development in information retrieval, pp 549–558

25. Guillory A, Bilmes J (2010) Interactive submodular set cover. arXiv preprint arXiv:1002.3345
26. Chen C, Li D, Zhao Y, et al (2015) WEMAREC: Accurate and scalable recommendation through weighted and ensemble matrix approximation. In: Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval, pp 303–312
27. Pilszy I, Zibriczky D, Tikk D (2010) Fast als-based matrix factorization for explicit and implicit feedback datasets. In: Proceedings of the fourth ACM conference on recommender systems, pp 71–78
28. Devooght R, Kourtellis N, Mantrach A (2015) Dynamic matrix factorization with priors on unknown values. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pp 189–198
29. Yuan Q, Cong G, Ma Z, et al (2013) Time-aware point-of-interest recommendation. In: Proceedings of the 36th international ACM SIGIR conference on research and development in information retrieval, pp 363–372
30. Dhillon, Inderjit S, Mallela Subramanyam, Modha Dharmendra S (2003) Information-theoretic co-clustering. In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp 89–98