Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Performance of latent factor models with extended linear biases

Jia Chen[a,1], Xin Luo[b,c,1,*], Ye Yuan[b], Mingsheng Shang[b], Zhong Ming[c], Zhang Xiong[a]

[a] *School of Computer Science, Beihang University, Beijing 100191, China*
[b] *Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China*
[c] *College of Computer Science and Engineering, Shenzhen University, Shenzhen 518060, China*

ARTICLE INFO

Article history:
Received 19 February 2016
Revised 6 February 2017
Accepted 7 February 2017
Available online 14 February 2017

KeyWords:
Big data
Recommender system
Latent factor model
Linear bias
Industrial application

ABSTRACT

High-dimensional and sparse (HiDS) matrices are frequently encountered in various industrial applications, due to the exploding number of involved entities and great needs to describe the relationships among them. Latent factor (LF) models are highly effective and efficient in extracting useful knowledge from these HiDS matrices. They well represent the known data of a HiDS matrix with high computational and storage efficiency. When building an LF model, the incorporation of linear biases has proven to be effect in further improving its performance on HiDS matrices in many applications. However, prior works all propose to assign a single bias to each entity, i.e., a single bias for each user/movie from a user-movie HiDS matrix. In this work we argue that to extend the linear biases, i.e., to assign multiple biases to each involved entity, can further improve an LF model's performance in some applications. To verify this hypothesis, we first extended the linear biases of an LF model, and then deduced the corresponding training rule of involved LFs. Subsequently, we conducted experiments on ten HiDS matrices generated by different industrial applications, evaluating the resulting LF models' prediction accuracy for the missing data of involved HiDS matrices. The experimental results indicate that on most testing cases an LF model needs to extend its linear biases to achieve the highest prediction accuracy. Hence, the number of linear biases should be chosen with care to make an LF model achieve the best performance in practice.

© 2017 Elsevier B.V. All rights reserved.

1. Introduction

With the world-wild-web widely expanding with a mass of information released, shared and accessed at every moment, people are living in the era of information explosion. They are no longer lack of information but inundated by it. With such information overload, numerous entities and their corresponding high-dimensional relationships are frequently processed in many industrial applications, e.g., users, items and user-item preferences in recommender systems [1–5], users, services and user-service QoS history in Web-service QoS analysis systems [6–8], users and user-user trust networks in social network-based services [9–13], and proteins and protein-protein interactome mappings in bioinformatics [14–15]. Due to the exploding numbers of involved entities, e.g., hundreds of thousands of users and items in recommender systems, it becomes impossible to observe the full relationship among them. Consequently, high-dimensional and sparse (HiDS) matrices are frequently encountered to describe such relationships in practice [1–3, 6–12, 14–15].

In spite of their incompleteness, these HiDS matrices contain rich information regarding various desired patterns, e.g., user preferences in recommender systems [1–3] and protein interactome features in bioinformatics [16]. Hence, how to extract such useful knowledge from them becomes a vital yet thorny issue [1–16]. Great efforts are made regarding it, resulting in various knowledge acquisition models, where one important branch of such models is based on latent-factor (LF) analysis of HiDS matrices [3, 10, 11]. Originated from matrix-factorization (MF) techniques, an LF model also works by approximating a target HiDS matrix in a low-rank way. They first map entities corresponding to the column and row entities of a given HiDS matrix into the same low-dimensional LF space. Then, a series of loss functions are built based on its known entry set and the desired LFs. Finally, the generalized loss measured by these loss functions is minimized with respect to these LFs. Nonetheless, traditional MF models deal with full matrices, while the target HiDS matrix of an LF model is always highly sparse, i.e., only a very small part of its entries are observed.

Due to the great need of acquiring useful knowledge from HiDS matrices, many sophisticated LF models are proposed. Representative models of this kind include the biased regularized incre-

* Corresponding author.
E-mail addresses: chenjia@buaa.edu.cn (J. Chen), luoxin21@cigit.ac.cn, luoxin21@gmail.com (X. Luo), yuanye@cigit.ac.cn (Y. Yuan), msshang@cigit.ac.cn (M. Shang), mingz@szu.edu.cn (Z. Ming), xiongz@buaa.edu.cn (Z. Xiong).
[1] J. Chen and X. Luo contributed equally to this work.

http://dx.doi.org/10.1016/j.knosys.2017.02.010
0950-7051/© 2017 Elsevier B.V. All rights reserved.

mental simultaneous MF (BRISMF) model [17], singular value decomposition plus-plus model [18, 19], probabilistic MF model [20], nonparametric MF model [21], weighted trace-norm regularization-based model [22], and non-parametric Bayesian-based LF model [23]. These models may differ from each other vastly in their objective functions and training schemes, yet share two similar principles: 1) to address HiDS matrices, an LF-based model only focuses on its known entry set to build the objective function, which measuring the generalized loss with respect to related LFs; and 2) to achieve high computational efficiency, the optimization process is taken with respect to LFs related to known entries only. These extracted LFs can well represent the observed data in the HiDS matrices, and be interpreted as the entity features hidden in a target HiDS matrix. The resulting LF models have been successfully applied to various important tasks, e.g., missing-data estimation [3, 17–23], community detection [11, 24], image tagging [25], video reindexing [26], and mobile-user tracking [27].

LF models are highly scalable and compatible with many performance-enhancing strategies [17–23]. Among them, a well-known and widely-adopted one is the linear bias integration, which has proven to be highly effective in further improving an LF model's performance [28–30]. However, previous works all propose to assign a single bias for each entity, i.e., a single bias for each user/movie in a user-movie HiDS matrix. In this work, we argue that the extension of the linear biases, i.e., assigning multiple biases on each entity, can further improve an LF model's performance. To verify this hypothesis, we first split the linear biases of an LF model, and then deduce the corresponding training rule of involved LFs. Subsequently, we conduct experiments on ten HiDS matrices generated by different industrial applications, to check the prediction accuracy of an LF model with different extended numbers of linear biases. The main contributions of this paper include:

(1) Extended linear bias scheme, which enables multiple linear biases for each involved entity in latent factor-based models;
(2) Algorithm design and analysis for a latent factor-based model with extended linear biases; and
(3) Empirical validations on ten high-dimensional and sparse matrices generated by real applications.

The rest of this paper is organized as follows. Section 2 gives the preliminaries. Section 3 states our methods. Section 4 gives and analyzes the experimental results. Finally, Section 5 discusses and concludes this paper.

2. Preliminaries

In our context, a HiDS matrix is the data source, which is defined as follows.

Definition 1. Let U and I denote two large entity sets, R denote an $|U|\times|I|$ matrix whose element $r_{u,i}$ describing some relationships between entities $u\in U$ and $i\in I$, R_K and R_U denote the known and unknown entry sets of R. R is a HiDS matrix if $|R_K|\ll|R_U|$.

Given a HiDS target R, an LF model seeks for a rank-f approximation $\hat{R}$ to R. According to [17-30], we define $\hat{R}$ as follows.

Definition 2. Given R and R_K, an LF model seeks for R's rank-f approximation $\hat{R}$ fulfilling $f\ll\min\{|U|,|I|\}$ based on R_K.

Note that f can also be interpreted as the dimension of the LF space, and an LF model is usually composed of several LF matrices. In this work we only consider LF matrices corresponding to involved entity sets, i.e., U and I. As unveiled by prior works [18, 19], further LF matrices like the time temporal LF matrix and neighborhood LF matrix can be introduced into an LF model for performance gain. Nonetheless, the relative simplicity of the adopted model does not prevent us from establishing the problem. Therefore, we defined the concerned model as follows,

Definition 3. Given U, I and f; given a $|U|\times f$ matrix P for U, and a $|I|\times f$ matrix Q for I; the desired rank-f approximation is given by $\hat{R}=PQ^T$.

According to Definition 3, each row vector in P and Q denotes a specific LF vector corresponding to an entity in U and I, respectively. Meanwhile, since R is a HiDS matrix,$\hat{R}$should be built based on R_K rather than the whole R. Let T and V be the subsets of R_K as the training and validating datasets, then we define the problem of building an LF model as follows.

Definition 4. Given T, V, U, I and f, an LF model seeks for P and Q based on T to obtain $\hat{R}$ where each entry $\hat{r}_{u,i}=p_u\cdot q_i$ to approximate $\arg\min \sum_{\forall r_{w,j}\in V} |\hat{r}_{w,j}-r_{w,j}|_{abs}$.

Note that in Definition 4, p_u and q_i denote the uth row of P and ith row of Q, and $|\cdot|_{abs}$ computes the absolute value of the enclosed number, respectively. To extract P and Q from R based on T, we define a loss function measuring the difference between R and$\hat{R}$. Without loss of generality, this work adopts the most commonly seen Euclidean distance [17-20, 28-33] to formulate the problem. Note that the methods to be described also work with other loss functions like the Kullback-Leibler divergence. So the problem of building an LF model is formulated by:

$$\begin{aligned}\arg\min_{P,Q}\varepsilon(P,Q)&=\frac{1}{2}\sum_{\forall r_{u,i}\in T}\left(r_{u,i}-\hat{r}_{u,i}\right)^2\\&=\frac{1}{2}\sum_{\forall r_{u,i}\in T}\left(r_{u,i}-\sum_{k=1}^{f}p_{u,k}q_{i,k}\right)^2,\end{aligned}\tag{1}$$

where $p_{u,k}$ and $q_{i,k}$ denote the kth elements of p_u and q_i, respectively. Note that (1) is obtained by expanding the inner product between LF vectors p_u and q_i when computing$\hat{r}_{u,i}$. The robustness of (1) can be greatly improved through integrating the Tikhonov regularization as follow,

$$\begin{aligned}&\arg\min_{P,Q}\varepsilon(P,Q)=\frac{1}{2}\sum_{\forall r_{u,i}\in T}\\&\quad\times\left(\left(r_{u,i}-\sum_{k=1}^{f}p_{u,k}q_{i,k}\right)^2+\lambda\left(\|p_u\|_F^2+\|q_i\|_F^2\right)\right)\\&=\frac{1}{2}\sum_{\forall r_{u,i}\in T}\left(\left(r_{u,i}-\sum_{k=1}^{f}p_{u,k}q_{i,k}\right)^2+\lambda\left(\sum_{k=1}^{f}p_{u,k}^2+\sum_{k=1}^{f}q_{i,k}^2\right)\right),\end{aligned}\tag{2}$$

where $\|\cdot\|_F$ computes the l_2 norm of the enclosed vector, λ is the constant controlling the regularization effect, and the terms behind λ are the regularization terms preventing a solver from overfitting the loss function (1), respectively. Note that (2) means introduces the l_2 norms of involved LFs as the penalty factor, thereby making a solver minimize (1) and involved LFs' l_2 norms simultaneously, preventing it from overfitting the generalized error.

In order to further improve the performance of an LF model, most previous works suggest integrating linear biases into it. We set the length-$|U|$ vector B as the linear bias vector for U and length-$|I|$ vector C as the linear bias vector for I to expand (2) into:

$$\begin{aligned}\arg\min_{B,C,P,Q}\varepsilon(B,C,P,Q)&=\frac{1}{2}\sum_{\forall r_{u,i}\in T}\left(\left(r_{u,i}-b_u-c_i-\sum_{k=1}^{f}p_{u,k}q_{i,k}\right)^2\right.\\&\quad\left.+\lambda\left(b_u^2+c_i^2+\sum_{k=1}^{f}p_{u,k}^2+\sum_{k=1}^{f}q_{i,k}^2\right)\right),\end{aligned}\tag{3}$$

where b_u and c_i denote the uth and ith element of B and C, respectively. Then (3) can be minimized with respect to B, C, P and Q on T with an optimizer, e.g., stochastic gradient descent (SGD).

Previous works regarding linear bias incorporation into an LF model all assign a single linear bias to each entity, e.g., b_u for $u\in U$ and c_i for $i\in C$ as in (3). However, in this work we argue that to assign multiple linear biases to each entity may further improve the performance of an LF model under some circumstances. The main motivation of this work is to answer the following questions:

(1) Is it possible to extend the linear biases, enabling multiple linear biases for each involved entity in a latent factor-based model? If yes, then how to design highly efficient algorithms to do so?
(2) Does the common accepted implementation of linear biases, i.e., assigning one single linear bias to each involved entity, always enable the best performance of a latent factor-based model? Or extended linear biases are desired for further improving the performance of a latent factor-based model?

Next we give the corresponding methods.

3. Methods

3.1. Extended linear biases

According to [17], assigning a single linear bias to each involved entity as in (3) is equivalent to fixing the first column of P and second column of Q at one. Corresponding to (3), this description is formulated by:

$$P' = \begin{bmatrix} 1 & B^T & P \end{bmatrix}, Q' = \begin{bmatrix} C^T & 1 & Q \end{bmatrix}, \hat{R} = P'\left(Q'\right)^T; \tag{4}$$

Note that in (4) we slightly abuse the notations by making ‘1’ denote a column vector filled with one. According to (4), the linear biases, i.e., B and C, play a similar role to the principle components of the desired LFs [17–20, 34].

(3) and (4) are based on the assumption that to assign a single linear bias to each involved entity is the optimal choice. However, here we argue that the optimal count of linear biases may vary according to the characteristics of data in R. From this point of view, it is necessary to assign multiple linear biases to each involved entity, i.e., to extend B and C as follows,

$$B = \begin{bmatrix} b_{1,1} & \dots & b_{1,|U|} \\ \vdots & & \vdots \\ b_{x,1} & \dots & b_{x,|U|} \end{bmatrix}, C = \begin{bmatrix} c_{1,1} & \dots & c_{1,|I|} \\ \vdots & & \vdots \\ c_{y,1} & \dots & c_{y,|I|} \end{bmatrix}; \tag{5}$$

In other words, we transform B and C from length-$|U|$ and $|I|$ vectors to $x\times|U|$ and $y\times|I|$ matrices. Thus, (4) is reformulated into:

$$P'' = \begin{bmatrix} J & B^T & P \end{bmatrix}, Q'' = \begin{bmatrix} C^T & K & Q \end{bmatrix}, \hat{R} = P''\left(Q''\right)^T; \tag{6}$$

where the $|U|\times y$ matrix J and $|I|\times x$ matrix K are both matrices filled with one. From (6), we see that by assigning multiple linear biases on each involved entity, the effect of principle components in (4) is strengthen with $x>1$ and $y>1$. Moreover, we probably have $x\neq y$, since the optimal number of linear biases may depend on the kind of corresponding entities.

By expanding $\hat{R} = P''(Q'')^T$ in (6) into the single-element-dependent form, we obtain the expression of $\hat{r}_{u,i}$ with extended linear biases as follows,

$$\hat{r}_{u,i} = \sum_{j=1}^{x} b_{u,j} + \sum_{l=1}^{y} c_{i,l} + \sum_{k=1}^{f} p_{u,k}q_{i,k} \tag{7}$$

Table 1
Training algorithm for an LF model with extended linear biases and SGD.

Input: U, I, R_K	
Operation	*Cost*
Initialize x, y, f, λ, η, N = **max_iteration_number**	$\Theta(1)$
Initialize $P^{\|U\|\times f}$, $Q^{\|I\|\times f}$, $B^{\|U\|\times x}$, $C^{\|I\|\times y}$	$\Theta(\|U\|\times(f+x)+\|I\|\times(f+y))$
while $n\leq N$ and **not** converge	$\times N$
for $\forall r_{u,i}\in R_K$	$\times\|R_K\|$
compute $\hat{r}_{u,i}$ according to (7)	$\Theta(f+x+y)$
$err_{u,i} = r_{u,i} - \hat{r}_{u,i}$	$\Theta(1)$
for $k=1\sim f$	$\times f$
update $p_{u,k}$ according to (10)	$\Theta(1)$
update $q_{i,k}$ according to (10)	$\Theta(1)$
end for	–
for $j=1\sim x$	$\times x$
update $b_{u,j}$ according to (10)	$\Theta(1)$
end for	–
for $l=1\sim y$	$\times y$
update $c_{i,l}$ according to (10)	$\Theta(1)$
end for	–
end for	–
$n=n+1$	$\Theta(1)$
end while	–
Output*: desired LF matrices *B, C, P, Q	

By substituting (7) into (4), the objective function is reformulated into the following form:

$$\arg\min_{B,C,P,Q} \varepsilon(B,C,P,Q) = \frac{1}{2}\sum_{\forall r_{u,i}\in T} \times\left(\left(r_{u,i}-\hat{r}_{u,i}\right)^2 + \lambda\left(\sum_{j=1}^{x} b_{u,j}^2 + \sum_{l=1}^{y} c_{i,l}^2 + \sum_{k=1}^{f}\left(p_{u,k}^2+q_{i,k}^2\right)\right)\right) \tag{8}$$

As indicated by prior works [17–19], SGD has proven to be effective and efficient in solving bilinear loss functions. So we adopt it to solve (8) as follows,

$$for\ \forall r_{u,i}\in T, j=1\sim x, l=1\sim y, k=1\sim f: \begin{cases} b_{u,j}^* \leftarrow b_{u,j} - \eta\frac{\partial\varepsilon_{u,i}}{\partial b_{u,j}} \\ c_{i,l}^* \leftarrow c_{i,l} - \eta\frac{\partial\varepsilon_{u,i}}{\partial c_{i,l}} \\ p_{u,k}^* \leftarrow p_{u,k} - \eta\frac{\partial\varepsilon_{u,i}}{\partial q_{u,k}} \\ q_{i,k}^* \leftarrow q_{i,k} - \eta\frac{\partial\varepsilon_{u,i}}{\partial q_{i,k}} \end{cases} \tag{9}$$

where $\varepsilon_{u,i}$ denotes the partial loss associated with the training data $r_{u,i}$ in (8), i.e., $\varepsilon_{u,i} = \frac{1}{2}((r_{u,i}-\hat{r}_{u,i})^2 + \lambda(\sum_{j=1}^{x} b_{u,j}^2 + \sum_{l=1}^{y} c_{i,l}^2 + \sum_{k=1}^{f}(p_{u,k}^2+q_{i,k}^2)))$. Let $err_{u,i} = r_{u,i} - \hat{r}_{u,i}$, then we expand (9) as follows,

$$for\ \forall r_{u,i}\in T, j=1\sim x, l=1\sim y, \\ k=1\sim f: \begin{cases} b_{u,j}^* \leftarrow b_{u,j} + \eta\left(err_{u,i} - \lambda b_{u,j}\right) \\ c_{i,l}^* \leftarrow c_{i,l} + \eta\left(err_{u,i} - \lambda c_{i,l}\right) \\ p_{u,k}^* \leftarrow p_{u,k} + \eta\left(err_{u,i}q_{i,k} - \lambda p_{u,k}\right) \\ q_{i,k}^* \leftarrow q_{i,k} + \eta\left(err_{u,i}p_{u,k} - \lambda q_{i,k}\right) \end{cases} \tag{10}$$

Table 1 gives the algorithm adopting the SGD-based training scheme (10) for LFs in (8). As shown in Table 1, for an Extended Linear Biases-enhanced LF (ELBLF) model, its training process with SGD is very similar to the original LF model whose loss function is given in (3). The differences lie in the prediction to each training instance in R_K, and the update procedures of linear biases. However, these two differences will not result in considerable computational burden. Based on the computational cost of each step recorded Table 1, we formulate its computational cost as follows,

$$\begin{aligned} T &= \Theta(1 + |U| \times (f + x) + |I| \times (f + y) + N \\ &\quad \times (|R_K| \times (f + x + y + 1 + 2f + x + y) + 1)) \\ &= \Theta(1 + |U| \times (f + x) + |I| \times (f + y) + N \\ &\qquad \times |R_K| \times (3f + 2x + 2y + 1) + N) \\ &\qquad\quad \approx \Theta(N \times |R_K| \times (f + x + y)) \end{aligned} \tag{11}$$

Note that the last step of (11) reasonably omits the lower order terms on the premise that $\max\{|U|,|I|\}<<(N\times|R_K|)$, along with the constant coefficients. As discussed in prior works [17–20, 28–30], the computational cost of an LF model with SGD is $\Theta(N\times|R_K|\times f)$. Compared with it, the ELBLF's computational cost is a bit higher due to the update of multiple linear biases. However, this additional cost is linear with respect to x and y. As shown in the experiments, it is unnecessary to adopt large x and y for achieving the best performance, so this additional cost is easy to resolve in practice.

Naturally, with different x and y, the local optima of (8) vary due to the change in linear biases, thereby affecting the performance of the resulting model. Next we conduct detailed experiments to investigate this issue.

4. Empirical studies

4.1. General settings

4.1.1. Evaluation protocol

Given a HiDS matrix, one important task is to predict its unknown entries based on the known ones due to the great need to implement the full relationship mapping between entity sets [1–3, 6–12, 14–16]. Hence, this work adopts the task of missing-data-estimation as the evaluation protocol. Formally, given the known entry set T, we make a tested model predict data in the validation set V to checking its prediction accuracy, so naturally $T \cap V = \phi$. To measure the prediction accuracy of a tested model, we adopt the mean absolute error (MAE) and root mean squared error (RMSE) [1–3, 17–23, 28–30, 35], which are also widely adopted for evaluating the statistical accuracy of an LF model when predicting missing data in a HiDS matrix. These two metrics are formulated by:

$$\begin{aligned} MAE &= \left(\sum_{\forall r_{w,j} \in V} \left| \hat{r}_{w,j} - r_{w,j} \right|_{abs} \right) \Big/ |V|, \\ RMSE &= \sqrt{\sum_{\forall r_{w,j} \in V} \left(\hat{r}_{w,j} - r_{w,j} \right)^2 \Big/ |V|}, \end{aligned} \tag{12}$$

All experiments are conducted on a PC with a 2.5 GHz i5 CPU and 4 GB RAM. All models are implemented in JAVA SE 7U60 to check their suitability for industrial usage.

4.1.2. Experimental datasets

In order to present a detailed report, we have conducted experiments on 10 HiDS matrices collected by industrial applications, whose details are given below:

(1) D1: Jester-3 dataset. It is collected through the joke-recommender Jester [36], and contains 616,912 continuous ratings in the scale of [−10.00, 10.00] from 24,938 users on 100 jokes. The data density of D1 is 24.74%.
(2) D2: MovieLens 1 M dataset. It contains 1,000,209 ratings applied to 3900 movies by 6040 users with a rating scale of [0, 5], which is collected by the movie recommender MovieLens developed by the GroupLens research group at the University of Minnesota [37]. Its data density is 4.25%.
(3) D3: Extended Epinion dataset. It is collected by Trustlet website [38], and contains 13,668,320 known entries in the range of [1, 5], by 120,492 users on 755,760 articles. Its data density is 0.015% only.
(4) D4: EachMovie dataset. It is collected through the EachMovie system by the DEC Systems Research Center, and contains 2,811,718 ratings ranging in the scale of [0.0, 1.0], entered by 72,916 users on 1,628 different movies. Its data density is 2.37%.
(5) D5: Dating Agency data set. It is collected by the online dating website LibimSeTi [39], and contains 17,359,346 known entries in the range of [1, 10], by 135,359 users on 168,791 profiles. Its data density is 0.076%.
(6) D6: Response-time data in WS-Dream dataset 2. Collected by the WS-Dream system [7, 40], it contains 1,873,838 continuous response-time data by 339 users on 5,825 Web services. Its data density is 94.89%.
(7) D7: Throughput data in WS-Dream dataset 2 [7, 40]. It contains 1831,253 continuous throughput data by 339 users on 5,825 Web services. Its data density is 92.74%.
(8) D8: Response-time data in WS-Dream dataset 3 [7, 40]. It contains 491,360 continuous response-time data by 134 users on 4,532 Web services. Its data density is 80.91%.
(9) D9: Throughput data in WS-Dream dataset 3 [7, 40]. It contains 491,360 continuous throughput data by 134 users on 4,532 Web services. Its data density is 80.91%.
(10) D10: Tag Genome dataset. Collected by GroupLens [41], it contains 10,979,952 continuous movie-tag relevance data in the scale of (0, 1) among 9,734 movies and 1,138 tags. Its data density is 99.12%.

Note that D6-D10 are actually not sparse due to the dense data in the corresponding high-dimensional matrices. We involve them in the experiments for checking the effect of extended linear biases on different types of data. On D1-D5, we employ the 80%−20% train-test and five-fold cross-validation settings, i.e., we split the dataset into five equally-sized, disjoint subsets at random, and select four subsets as T to train a model predicting the remaining one subset as V for performance test, and sequentially repeat this process is for five times to obtain the final results. On D6-D10, we employ the 20%−80% train-test where we select one subset as T to train a model to predict the other four remaining subsets as V, along with the five-fold cross-validation settings.

4.1.3. Model settings

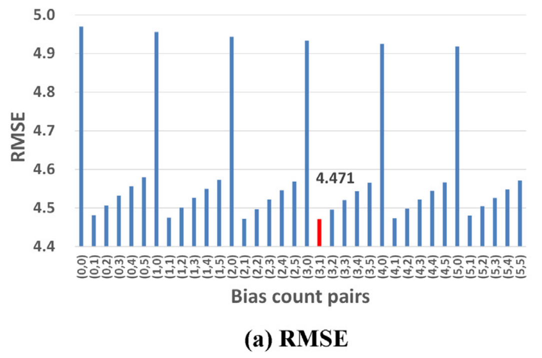
The purpose of the experiments is to validate the effect of extended linear biases in the performance of an LF model. Therefore, on all datasets we fix f at 20 and test the performance of an ELBLF model with x and y ranging from 0 to 5. Note that such settings also include two special cases, i.e., an LF model without linear biases with $x = y = 0$, and an LF model with commonly employed single linear biases with $x = y = 1$. On each dataset, with a grid search with respect to x and y, we test 36 different combinations of extended linear biases.

In terms of the other hyper parameters, i.e., η and λ, we first tune their values on one fold according to instructions in prior works [17–20, 28–33], and then employ the same setting on the other four folds. Note that we adopt consistent η and λ with different x and y, ensuring that the experimental results do not rely on these two hyper parameters.

4.2. Results

We first depict the RMSE and MAE of an ELBLF model with different x and y on all datasets in Figs. 1–10. In the following we analyze these results in detail.

On D1, ELBLF achieves the lowest RMSE at 4.471 with $x = 3$ and $y = 1$, as shown in Fig. 1(a). From Fig. 1(a), we also observe that the prediction accuracy of ELBLF relies more on C rather than on B, e.g., with $y = 0$ its RMSE increases significantly, as shown at points (0, 0), (1, 0), (2, 0), (3, 0), (4, 0) and (5, 0) in Fig. 1(a). Nonetheless,



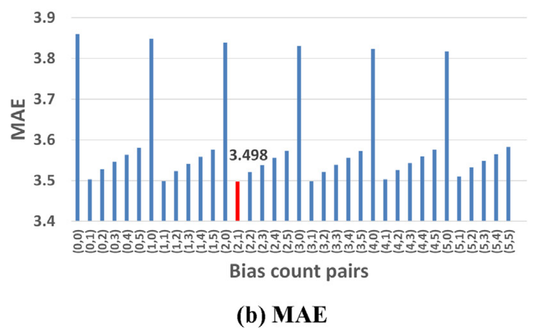
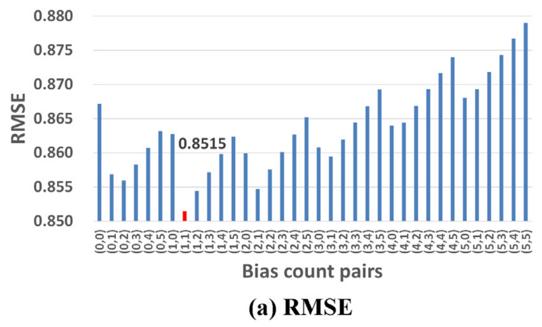


Fig. 1. Results on D1. Note that the bar colored in red denotes the lowest error. Labels in the *X*-axis denote the combinations of *x* and *y*, e.g., (0, 0) represents for $x = 0$ and $y = 0$. Figs. 2–10 share the same settings.



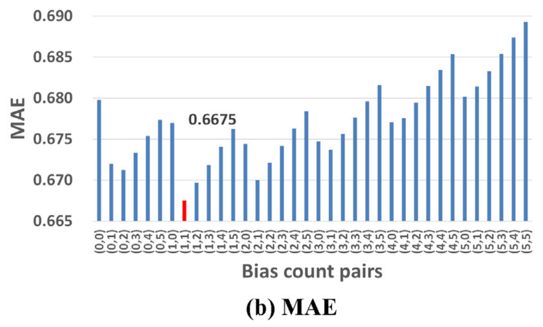


Fig. 2. Results on D2.

by incorporating linear biases in $\underline{C}$, ELBLF's prediction accuracy increases. For instance, its RMSE is 4.481 at point (0, 1), 9.85% lower than that 4.971 at (0, 0). However, this does not mean that the increase of *y* always ensures high prediction accuracy. As shown in Fig. 1(a), with $y > 1$, ELBLF's RMSE again increases.

When measured by MAE, the lowest prediction error of ELBLF is 3.498 with $x = 2$ and $y = 1$. Except for the optimal values of *x* and *y*, the situation with MAE is very similar to that with RMSE, as depicted in Fig. 1(b).

On D2, ELBLF achieves the lowest RMSE at 0.8515 and the lowest MAE at 0.6675 with $x = 1$ and $y = 1$. In other words, on this dataset the strategy of assigning a single linear bias to each involved entity, which is commonly adopted by prior works, is the most effective one out of all possible combinations of *x* and *y*. Meanwhile, we also notice that with $x > 1$ and $y > 1$, ELBLF's prediction accuracy decreases as *x* and *y* increase. For instance, ELBLF's MAE at points (1, 1), (2, 1), (3, 1), (4, 1) and (5, 1) is 0.6675, 0.6700, 0.6737, 0.6776 and 0.6814, respectively. The increasing tendency of the prediction error as *x* increases is obvious. We can observe similar phenomena when fixing *x* fixed and enlarging *y*, as depicted in Fig. 2.

On D3, ELBLF achieves the lowest RMSE at 1.180, and the lowest MAE at 0.8788 with $x = 5$ and $y = 1$, as depicted in Fig. 3. Moreover, we see that its prediction accuracy relies heavily on the value of *x*. Its RMSE is 1.950 with $x = 0$ and $y = 0$, and 1.221 with $x = 5$ and $y = 0$; the accuracy gain is 37.38%. In terms of *y*, we see that its optimal choice is always 1; the prediction error increases with $y = 0$ or > 1.

On D4, ELBLF achieves the lowest RMSE at 0.2226 with $x = 0$ and $y = 1$, indicating that it does not need to assign any bias to entities in *U*, and need to single bias on entities in *I* to achieve the highest prediction accuracy. Moreover, as recorded in Fig. 4(a), improper design of linear biases will result in obvious accuracy loss, as shown by points (1, 3), (1, 4) and (1, 5) in Fig. 4(a). When measuring its prediction error with MAE, we obtain exactly the same results. The lowest MAE of ELBLF on D4 is 0.1717 with $x = 0$ and $y = 1$.

On D5, we have encountered different situations: ELBLF achieves the lowest RMSE at 1.901 with $x = 4$ and $y = 1$, and the lowest MAE at 1.297 with $x = 0$ and $y = 5$. This phenomenon is interesting, since it indicates that the optimal choice of *x* and *y* may also depend on the evaluation metrics. However, we also observe that the accuracy gap by different *x* and *y* is not large on D5. For instance, ELBLF's RMSE with $x = 0$ and $y = 5$ (which are the optimal values for the lowest MAE) is 1.932, only 1.6% higher than the lowest RMSE at 1.901.

On D6, the results are similar to those on D1-D5. ELBLF's lowest RMSE 1.160 and MAE 0.446 appear by fixing *x* at 0 and setting $y = 3$ and 2 respectively, as shown in Fig. 6. On D7, ELBLF's prediction accuracy keeps increasing as *x* and *y* increases, as depicted in Fig. 7. It achieves the lowest values of both RMSE and MAE with $x = 5$ and $y = 5$, which is the right bound of our tested scale. This phenomenon indicates that on some HiDS matrices we need assign several linear biases to each involved entity to achieve the highest prediction accuracy.

On D8, ELBLF's lowest RMSE 3.443 and MAE 1.940 occur with $x = 1$ and $y = 1$, and $x = 0$ and $y = 5$, respectively. On D9, we' ve encountered a special case: ELBLF achieves its lowest RMSE at 42.01 with $x = 0$ and $y = 0$, indicating that the lowest error is obtained without linear biases. With regard to MAE, the lowest value 6.205 appears with $x = 0$ and $y = 5$. As a matter of fact, when evaluated by MAE, ELBLF suffers accuracy loss when assigning linear biases to entities in *U*, and obtains accuracy gain when assigning linear biases to entities in *I* on D6, as depicted in Fig. 9(b). Note that D6-D9 are all QoS data collected by the WS-Dream system [7, 40], but the optimal values of *x* and *y* keep varying on them. This phenomenon unveils that the optimal values of *x* and *y* are probably

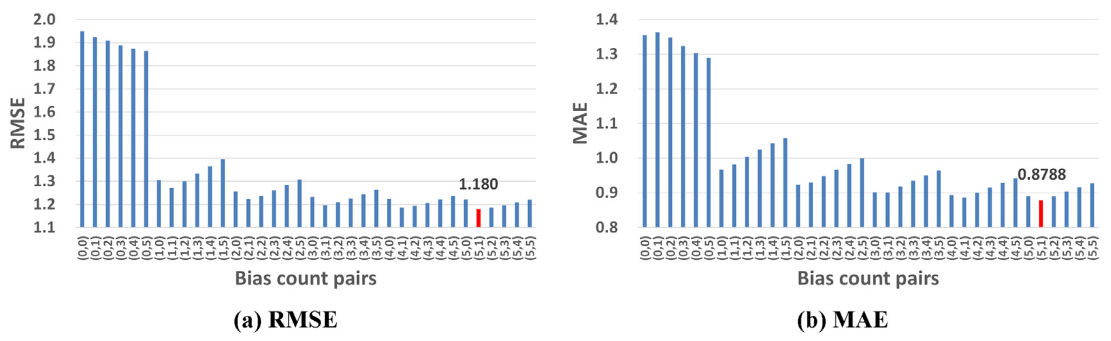


Fig. 3. Results on D3.

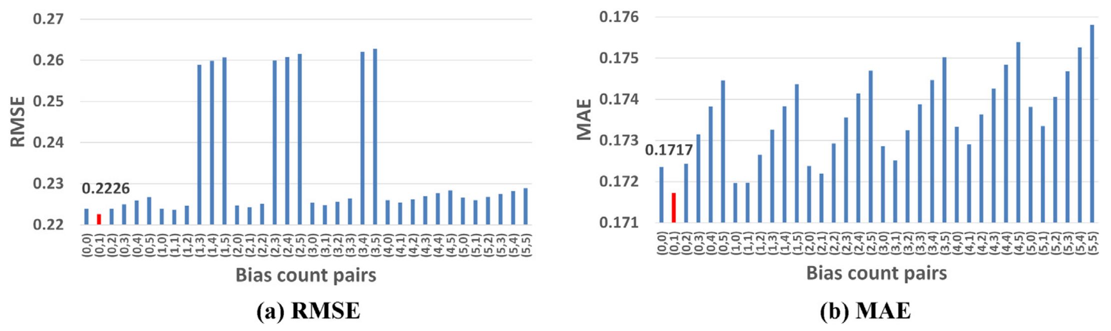


Fig. 4. Results on D4.

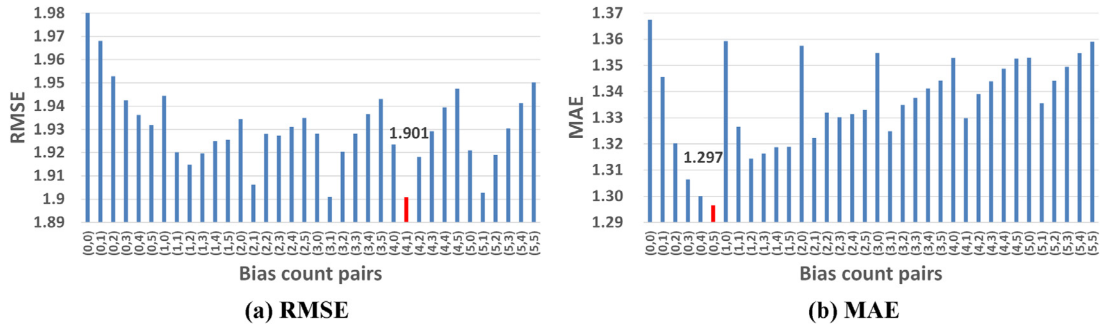


Fig. 5. Results on D5.

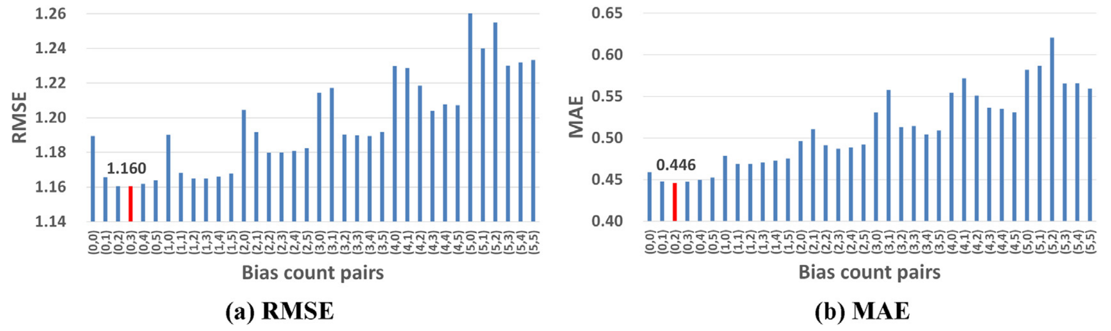


Fig. 6. Results on D6.

not connected with the general data type, but the detailed data distribution in *T*.

On D10, ELBLF again requires setting $x = 0$ and $y = 0$ to achieve the lowest RMSE 0.0947. During our experiments, in two cases out of all 20, ELBLF's prediction accuracy will decrease with the integration of linear biases. They indicate that to incorporate linear biases into an LF model will not always improve its prediction accuracy. In terms of MAE, ELBLF achieves the lowest value at 0.0512 with $x = 1$ and $y = 0$. Compared with that at 0.0622 with x = 0 and $y = 0$, and 0.0728 with $x = 1$ and $y = 1$, the accuracy gain is 17.68% and 29.67%, respectively.

Tables 2 and 3 summarize the experimental results. Figs. 11 and 12 depicts the distribution of optimal *x* and *y* to achieve the lowest RMSE and MAE on all datasets. From them, we summerize that *x* and *y* should be chosen with care to achieve the highest prediction accuracy. When measured by RMSE, with $x = 1$ and $y = 1$ ELBLF achieves the lowest error on only one dataset, i.e., D2. Meanwhile, on D9 and D10 linear biases are not needed for obtaining the low-

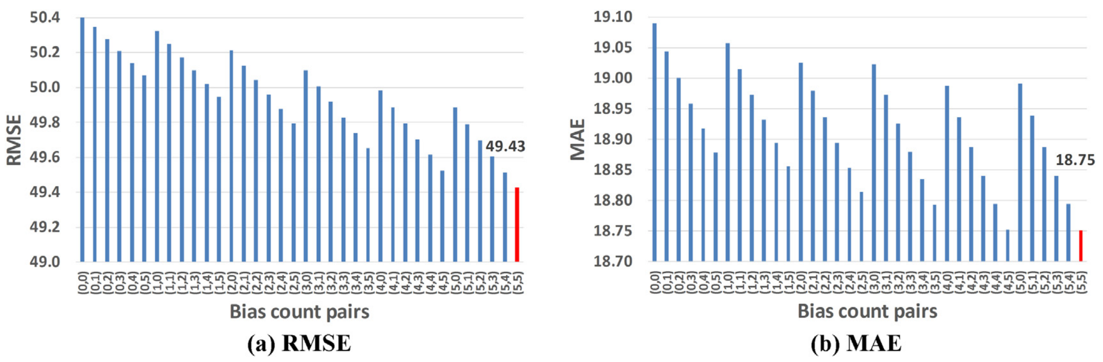


Fig. 7. Results on D7.

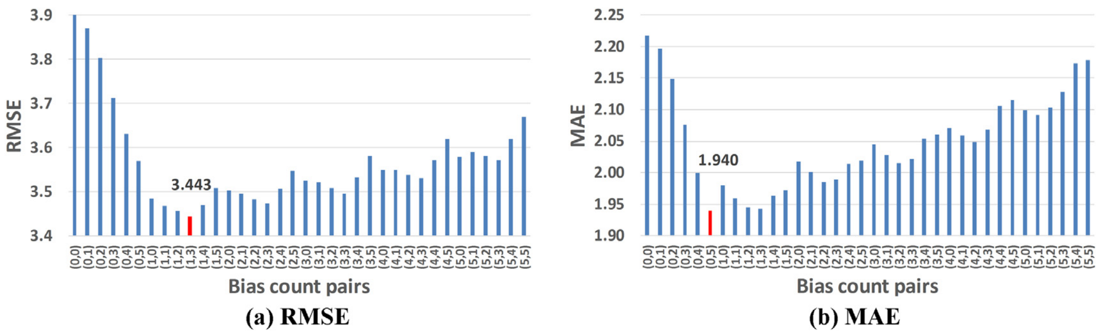


Fig. 8. Results on D8.

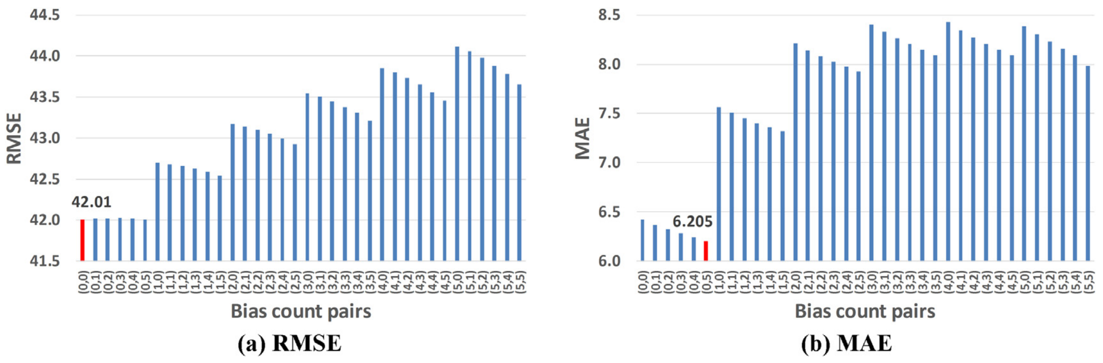


Fig. 9. Results on D9.

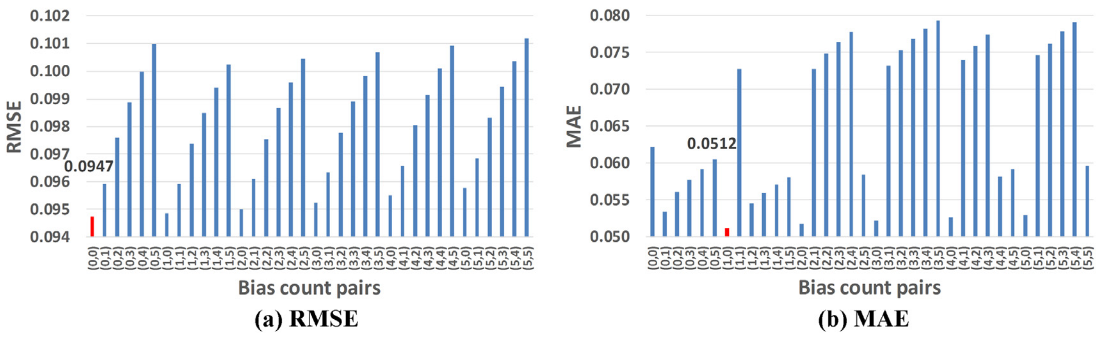


Fig. 10. Results on D10.

est RMSE. On the other 7 datasets, the optimal values of x and y keep varying, as depicted in Table 2 and Fig. 11. Similar situations are also encountered when we measure ELBLF's prediction error with MAE. This phenomenon indicates that the usual adopted setting, i.e., $x = 1$ and $y = 1$ by prior works [17–20, 28–30], is not always the optimal choice in practice.

5. Conclusions and future works

5.1. Conclusions drawn based on the experimental results

Based on the detailed experimental results, we conclude that 1) An LF model achieve its highest prediction accuracy for miss-

Table 2
Result summary in RMSE.

RST No.	Min	$(x, y)_{min}$	Max	$(x, y)_{max}$	Value of (1, 1)	Gain over (1, 1)
D1	4.471	(3,1)	4.970	(0,0)	4.475	**0.09%**
D2	0.8515	(1,1)	0.8790	(5,5)	0.8515	–
D3	1.180	(5,1)	1.950	(0,0)	1.271	**7.16%**
D4	0.2226	(0,1)	0.2628	(3,5)	0.2237	**0.49%**
D5	1.901	(4,1)	1.980	(0,0)	1.920	**0.99%**
D6	1.160	(0,3)	1.261	(5,0)	1.168	**0.68%**
D7	49.43	(5,5)	50.42	(0,0)	50.25	**1.63%**
D8	3.443	(1,3)	3.905	(0,0)	3.469	**0.75%**
D9	42.01	(0,0)	44.15	(5,0)	42.68	**1.57%**
D10	0.0947	(0,0)	0.1012	(5,5)	0.0959	**1.25%**

Table 3
Result summary in MAE.

RST. No.	Min	$(x, y)_{min}$	Max	$(x, y)_{max}$	Value of (1, 1)	Gain over (1, 1)
D1	3.498	(2,1)	3.856	(0,0)	3.499	**0.03%**
D2	0.6675	(1,1)	0.6893	(5,5)	0.6675	–
D3	0.8788	(5,1)	1.363	(0,1)	0.9815	**10.46%**
D4	0.1717	(0,1)	0.1758	(5,5)	0.1720	**0.17%**
D5	1.297	(0,5)	1.367	(0,0)	1.327	**2.26%**
D6	0.446	(0,2)	0.621	(5,2)	0.4688	**4.86%**
D7	18.75	(5,5)	19.09	(0,0)	19.014	**1.39%**
D8	1.940	(0,5)	2.218	(0,0)	1.960	**1.02%**
D9	6.205	(0,5)	8.431	(4,0)	7.504	**17.31%**
D10	0.0512	(1,0)	0.0793	(3,5)	0.0728	**29.67%**

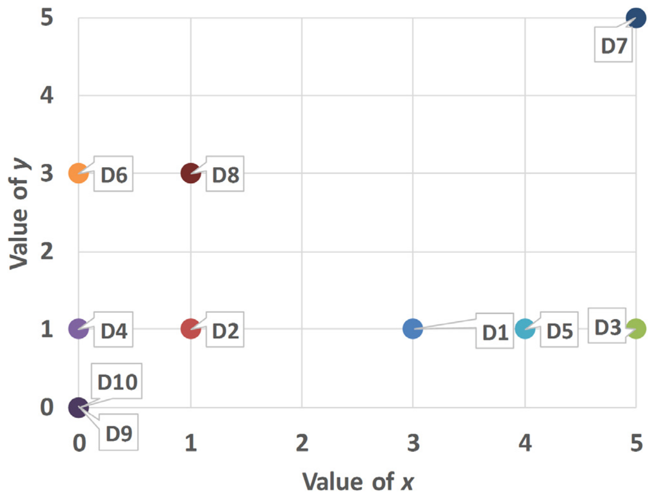


Fig. 11. Distribution of combinations of *x* and *y* achieving the lowest RMSE on all data sets.

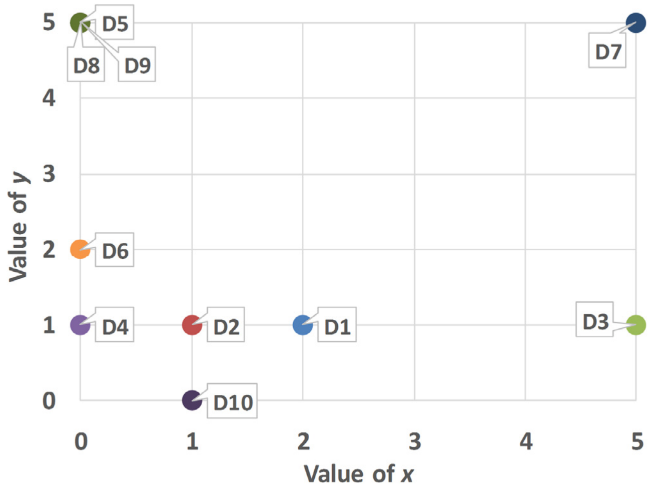


Fig. 12. Distribution of combinations of *x* and *y* achieving the lowest MAE on all data sets.

ing data with extended linear biases on most HiDS matrices involved in our experiments, indicating the need for this strategy in reall applications; 2) optimal combinations of extended linear biases mainly depend on the data distribution of the known entry set in a HiDS matrix. For identify them, it is necessary to conduct grid search on probe data in real applications; and 3) from the algorithm of ELBLF we see that to incorporate linear biases into an LF model will only cause slight computational burden which is easy to resolve in practice. Hence, it can be considered as a convenient way to improve the prediction accuracy of any LF model.

5.2. Possible extensions of ELBLF

For further improving the performance of ELBLF, we plan to address the following issues in the future: 1) How to choose the optimal value of *x* and *y*? This question remains open, since from the experimental results we have not deduced stationary rules regarding this issue. One possible way is to tune *x* and *y* along with other hyper parameters, i.e., *f*, η and λ on probe data, seeking for their optimal combinations and then adopt these tuned values to build an accurate model. However, theoretical guidance regarding this issue calls for our future efforts; and 2) How to apply extended linear biases into other LF matrices, i.e., the time temporal LF matrix and the neighborhood LF matrix [18, 19]? This is also a highly interesting issue and included in our future plan. Meanwhile, we also plan to investigate ELBLF's performance in other big data-related areas [42–51].

Acknowledgements

This research is supported in part by the Pioneer Hundred Talents Program of Chinese Academy of Sciences, in part by the Young

Scientist Foundation of Chongqing under Grant No. cstc2014kjrc-qnrc40005, in part by the National Natural Science Foundation of China under Grant No. 91646114 and No. 61370150, and in part by the Fundamental Research Funds for the Central Universities under Grant No. 106112016CDJXY180005 and No. CDJZR12180012, and in part by the project of the State Key Laboratory of Software Development Environment under Grant No. SKLSDE-2015ZX-25.

References

[1] J. Breese, D. Heckerman, C. Kadie, Empirical analysis of predictive algorithms for collaborative-filtering, in: Proceedings of the 14th International Conference on Uncertainty in Artificial Intelligence, San Francisco, California, USA, 1998, pp. 43–52.

[2] B. Sarwar, G. Karypis, J. Konstan, J. Reidl, Item based collaborative-filtering recommendation algorithms, in: Proceedings of the 10th International Conference on World Wide Web, Hong Kong, PRC, 2001, pp. 285–295.

[3] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions, IEEE Trans. Knowl. Data Eng. 17 (2005) 734–749.

[4] D. Rosaci, G.M.L. Sarné, Recommending multimedia web services in a multi--device environment, Inf. Syst. 38 (2) (2013) 198–212.

[5] D. Rosaci, CILIOS: connectionist inductive learning and inter-ontology similarities for recommending information agents, Inf. Syst. 32 (6) (2007) 793–825.

[6] L.-S. Shao, J. Zhang, Y. Wei, J.-F. Zhao, B. Xie, H. Mei, Personalized QoS-prediction for web services via collaborative filtering, in: Proceedings of the 14th IEEE International Conference on Web Services, Salt Lake City, USA, 2007, pp. 439–446.

[7] Z.-B. Zheng, H. Ma, M.R. Lyu, I. King, QoS-aware web service recommendation by collaborative filtering, IEEE Trans. Serv. Comput. 4 (2011) 140–152.

[8] J. Wu, L. Chen, Y.-P. Feng, Z.-B. Zheng, M.-C. Zhou, Z. Wu, Predicting quality of service for selection by neighborhood-based collaborative-filtering, IEEE Trans. Syst. Man Cybern 43 (2013) 428–439.

[9] R. Narayanam, Y. Narahari, A shapley value-based approach to discover influential nodes in social networks, IEEE Trans. Automat. Sci. Eng. 8 (1) (2011) 130–147 Jan.

[10] P. Kazienko, K. Musiał, T. Kajdanowicz, Multidimensional social network in the social recommender system, IEEE Trans. Syst. Man Cybern Part A 41 (4) (2011) 746–759.

[11] X. Cao, X. Wang, D. Jin, Y. Cao, D. He, Identifying overlapping communities as well as hubs and outliers via nonnegative matrix factorization, Sci. Rep. 3 (2013) 2993.

[12] X. Qian, H. Feng, G. Zhao, T. Mei, Personalized recommendation combining user interest and social circle, IEEE Trans. Knowl. Data Eng. 26 (7) (2014) 1763–1777.

[13] P. De Meo, A. Nocera, D. Rosaci, D. Ursino, Recommendation of reliable users, social networks and high-quality resources in a social internetworking system, AI Commun. 24 (1) (2011) 29–50.

[14] H.N. Chua, L. Wong, Increasing the reliability of protein interactomes, Drug Discov. Today 13 (15-16) (2008) 652–658 Aug.

[15] X. Luo, Z. Ming, Z.-H. You, S. Li, Y.-N. Xia, Q.-S. Zhu, H. Leung, Improving network topology-based protein interactome mapping via collaborative filtering, Knowl.-Based Syst. 90 (2015) 23–32.

[16] X. Luo, Z. You, M. Zhou, S. Li, H. Leung, Y. Xia, Q. Zhu, A highly efficient approach to protein interactome mapping based on collaborative filtering framework, Sci. Rep. 5 (2015) 7702.

[17] G. Takács, I. Pilászy, Bottyán Németh, D. Tikky, Scalable collaborative filtering approaches for large recommender systems, J. Mach. Learn. Res. 10 (2009) 623–656.

[18] Y. Koren, R. Bell, C. Volinsky, Matrix-factorization techniques for recommender systems, IEEE Comput. 42 (2009) 30–37.

[19] Y. Koren, R. Bell, Advances in collaborative filtering, in: F. Ricci, L. Rokach, B. Shapira, P.B. Kantor (Eds.), Recommender Systems Handbook, Springer, New York, US, 2011, pp. 145–186.

[20] R. Salakhutdinov, A. Mnih, Probabilistic matrix-factorization, Adv. Neural Inf. Process. Syst. 20 (2008) 1257–1264.

[21] K. Yu, S. Zhu, J. Lafferty, Y. Gong, Fast nonparametric matrix-factorization for large-scale collaborative-filtering, in: Proceedings of the 32nd ACM SIGIR International Conference on Research and Development in Information Retrieval, Boston, Massachusetts, USA, 2009, pp. 211–218.

[22] N. Srebro, R. Salakhutdinov, Collaborative filtering in a non-uniform world: learning with the weighted trace norm, in: Advances in Neural Information Processing Systems 23 (NIPS), Vancouver, British Columbia, Canada, 2010, pp. 2056–2064.

[23] S. Chatzis, Nonparametric Bayesian multitask collaborative filtering, in: Proceedings of the 22nd ACM International Conference on Information and Knowledge Management, San Francisco, California, USA, 2013, pp. 2149–2158.

[24] Y.-R. Lin, J. Sun, P. Castro, R. Konuru, H. Sundaram, A. Kelliher, MetaFac: community discovery via relational hypergraph factorization, in: Proceedings of the 15th ACM SIGKDD International Conferenec on Knowledge Discovery and Data Mining, Paris, France, 2009.

[25] Z. Ning, W.K. Cheung, Q. Guoping, X. Xiangyang, A hybrid probabilistic model for unified collaborative and content-based image tagging, IEEE Trans. Pattern Anal. Mach. Intell. 33 (7) (2011) 1281–1294.

[26] M.-F. Weng, Y.-Y. Chuang, Collaborative video reindexing via matrix factorization, ACM Trans. Multimed. Comput. Commun. Appl. 8 (2) (2012) 1–20.

[27] J.J. Pan, S.J. Pan, Y. Jie, L.M. Ni, Y. Qiang, Tracking mobile users in wireless networks via semi-supervised colocalization, IEEE Trans. Pattern Anal. Mach. Intell. 34 (3) (2012) 587–600.

[28] X. Luo, Y.-N. Xia, Q.-S. Zhu, Incremental collaborative filtering recommender based on regularized matrix factorization, Knowled.-Based Syst. 27 (2012) 271–280.

[29] X. Luo, Y.-N. Xia, Q.-S. Zhu, Applying the learning rate adaptation to the matrix factorization based collaborative filtering, Knowled.-Based Syst. 37 (2013) 154–164.

[30] X. Luo, M.-C. Zhou, Y.-N. Xia, Q.-S. Zhu, A.C. Ammari, A. Alabdulwahab, Generating highly accurate predictions for missing qos data via aggregating nonnegative latent factor models, IEEE Trans. Neural Netw. Learn. Syst. 27 (3) (2016) 579–592.

[31] N.-Y. Guan, D.-C. Tao, Z.-G. Luo, B. Yuan, Online nonnegative matrix factorization with robust stochastic approximation, IEEE Trans. Neural Netw. Learn. Syst. 23 (2012) 1087–1099.

[32] S. Zhang, W. Wang, J. Ford, F. Makedon, Learning from incomplete ratings using non-negative matrix-factorization, in: Proceedings of the SIAM International Conference on Data Mining, Bethesda, Maryland, USA, 2006, pp. 549–553.

[33] Y. Xu, W. Yin, Z. Wen, Y. Zhang, An alternating direction algorithm for matrix completion with nonnegative factors, Front. Math. China 7 (2) (2012) 365–384.

[34] A.M. Buchanan, A.W. Fitzgibbon, Damped Newton algorithms for matrix factorization with missing data, in: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Diego, CA, USA, 2005, pp. 316–322.

[35] J. Herlocker, J. Konstan, L. Terveen, J. Riedl, Evaluating collaborative filtering recommender systems, ACM Trans. Inf. Syst. 22 (1) (2004) 5–53.

[36] K. Goldberg, T. Roeder, D. Gupta, C. Perkins, Eigentaste: a constant time collaborative-filtering algorithm, Inf. Retrieval 4 (2001) 133–151.

[37] J.A. Konstan, B.N. Miller, D. Maltz, J.L. Herlocker, L.R. Gordon, J. Riedl, GroupLens: applying collaborative filtering to usenet news, Commun. ACM 40 (1997) 77–87.

[38] P. Massa, P. Avesani, Trust-aware recommender systems, in: Proceedings of the 2007 ACM Conference on Recommender Systems, Minneapolis, MN, USA, 2007, pp. 17–24.

[39] A. Krzywicki, W. Wobcke, X. Cai, A. Mahidadia, M. Bain, P. Compton, Y.S. Kim, Interaction-based collaborative filtering methods for recommendation in online dating, in: Proceedings of the 11th International Conference on Web Information Systems Engineering, Hong Kong, China, 2010, pp. 342–356.

[40] Y.-L. Zhang, Z.-B. Zheng, M.R. Lyu, Exploring latent features for memory-based QoS-prediction in cloud computing, in: Proceedings of the 30th IEEE International Symposium on Reliable Distributed Systems, Madrid, Spain, 2011, pp. 1–10.

[41] V. Jesse, S. Shilad, R. John, The tag genome: encoding community knowledge to support novel interaction, ACM Trans. Interact. Intell. Syst. 2 (3) (2012) 1–44.

[42] Z. Zhang, H. Lin, K. Liu, D. Wu, G. Zhang, J. Lu, A hybrid fuzzy-based personalized recommender system for telecom products/services, Inf. Sci. 235 (2013) 117–129.

[43] Q. Shambour, J. Lu, A hybrid trust-enhanced collaborative filtering recommendation approach for personalized government-to-business e-services, Int. J. Intell. Syst. 26 (9) (2011) 814–843.

[44] J. Li, M. Qiu, Z. Ming, G. Quan, X. Qin, Z. Gu, Online optimization for scheduling preemptable tasks on IaaS cloud systems, J. Parallel Distrib. Comput. 72 (5) (2012) 666–677.

[45] X. Luo, M.-C. Zhou, M.-S. Shang, S. Li, Y. Xia, A novel approach to extracting non-negative latent factors from big sparse matrices, IEEE Access 4 (2016) 2649–2655.

[46] X. Luo, M.-C. Zhou, Y.-N. Xia, Q.-S. Zhu, Generating highly accurate predictions for missing QoS-data via aggregating non-negative latent factor models, IEEE Trans. Neural Netw. Learn. Syst. 27 (2016) 579–592.

[47] X. Luo, M.-C. Zhou, Y.-N. Xia, Q.-S. Zhu, An incremental-and-static-combined scheme for matrix-factorization-based collaborative filtering, IEEE Trans. Autom. Sci. Eng. 13 (2016) 333–343.

[48] X. Luo, M.-C. Zhou, S. Li, Z.-H. You, Y.-N. Xia, Q.-S. Zhu, H. Leung, An efficient second-order approach to factorizing sparse matrices in recommender systems, IEEE Trans. Indus. Inf. 11 (2015) 946–956.

[49] X. Luo, M.-C. Zhou, Y.-N. Xia, Q.-S. Zhu, An efficient non-negative matrix-factorization-based approach to collaborative-filtering for recommender systems, IEEE Trans. Indus. Inf. 10 (2014) 1273–1284.

[50] Z.-H. You, M.-C. Zhou, X. Luo, S. Li, Highly efficient framework for predicting interactions between proteins, IEEE Trans. Cybern. 47 (2017) 721–733.

[51] S. Li, Z.-H. You, H.L. Guo, X. Luo, Z.-Q. Zhao, Inverse-free extreme learning machine with optimal information updating, IEEE Trans. Cybern. 46 (2016) 1229–1241.