Randomized Latent Factor Model for High-dimensional and Sparse Matrices from Industrial Applications

Mingsheng Shang, Xin Luo, *Senior Member, IEEE,* Zhigang Liu, Jia Chen, Ye Yuan, and MengChu Zhou, *Fellow, IEEE*

***Abstract*—Latent factor (LF) models are highly effective in extracting useful knowledge from High-Dimensional and Sparse (HiDS) matrices which are commonly seen in various industrial applications. An LF model usually adopts iterative optimizers, which may consume many iterations to achieve a local optima, resulting in considerable time cost. Hence, determining how to accelerate the training process for LF models has become a significant issue. To address this, this work proposes a randomized latent factor (RLF) model. It incorporates the principle of randomized learning techniques from neural networks into the LF analysis of HiDS matrices, thereby greatly alleviating computational burden. It also extends a standard learning process for randomized neural networks in context of LF analysis to make the resulting model represent an HiDS matrix correctly. Experimental results on three HiDS matrices from industrial applications demonstrate that compared with state-of-the-art LF models, RLF is able to achieve significantly higher computational efficiency and comparable prediction accuracy for missing data. I provides an important alternative approach to LF analysis of HiDS matrices, which is especially desired for industrial applications demanding highly efficient models.**

***Index Terms*—Big data, high-dimensional and sparse matrix, latent factor analysis, latent factor model, randomized learning.**

I. Introduction

With the large amount of information released and shared every second on the Internet, the world has entered an era of big data and very complex systems [1]. People are inundated by the big data generated by various industrial applications, e.g., user-item preferences in recommender systems [2]−[5], user-service QoS in Web-service analysis [6], [7], user interactions in social networks [8], [9], and node relationships in wireless sensor networks [10], [11]. These big data describe the relationship among numerous entities, which is unlikely to be fully observed [2], [3], [6], [7], [11]. Hence, high-dimensional and sparse (HiDS) matrices are often used to describe such relationships in practice [2], [12].

In spite of their sparsity, these HiDS matrices contain rich information regarding various desired patterns, e.g., users' potential preferences in recommender systems [2], [3]. However, their missing data result in a great obstacle that prevents people from extracting useful knowledge from them [2], [12]. Latent factor (LF) analysis has proven to be highly efficient in addressing such missing data issue in practice [12].

Similar to matrix factorization-based models, an LF model also maps a target HiDS matrix into a low-dimensional feature space [12]. Then, a series of loss functions are built on the known data of the target matrix and the desired LFs. These LFs are trained by minimizing the loss functions to represent the known data of the matrix. Afterwards, various useful patterns can be acquired based on trained LFs [12], [13].

Owing to their efficiency in LF analysis of HiDS matrices, LF models have been thoroughly investigated, and many sophisticated models have been established. However, as defined in the previous research, LF analysis of HiDS matrices is a non-convex problem that cannot be analytically solved [12], [14]. Hence, current LF models all adopt iterative optimizers, which may take many iterations to converge [12], [15], thereby resulting in considerable time cost. On the other hand, with more efficient optimizers requiring fewer training iterations, the efficiency of an LF model can be greatly improved, making it more practical for industrial applications in dynamic data environments.

Randomized learning is an efficient learning scheme for building a single-hidden layer feed-forward networks (SLFN) [16]−[19]. As proven in big data-related areas [20]−[22], extensions of such a learning scheme enable a model to approximate any continuous functions with enough hidden neurons [16]−[19]. With randomized learning, an SLFN can be built with very high computational efficiency by generating hidden neurons randomly and analytically solving the output weights directly [16]−[19]. As discussed in [15], building an

Manuscript received February 5, 2018; revised April 9, 2018; accepted April 22, 2018. This work was supported in part by the National Natural Science Foundation of China (61772493; 91646114), Chongqing research program of technology innovation and application (cstc2017rgzn-zdyfX0020), and in part by the Pioneer Hundred Talents Program of Chinese Academy of Sciences. Recommended by Associate Editor Jian Yang. Mingsheng Shang and Xin Luo are co-first authors. *(Corresponding author: Xin Luo and MengChu Zhou.)*

Citation: M. S. Shang, X. Luo, Z. G. Liu, J. Chen, Y. Yuan, and M. C. Zhou, "Randomized latent factor model for high-dimensional and sparse matrices from industrial applications," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 1, pp. 131−141, Jan. 2019.

M. Shang, X. Luo, Z. Liu and Y. Yuan are with the Chongqing Engineering Research Center of Big Data Application for Smart Cities, and Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China (e-mail: {msshang, luoxin21, liuzhigang, yuanye}@cigit.ac.cn).

J. Chen is with the School of Computer Science and Engineering, Beihang University, Beijing 100191, China (e-mail: chenjia@buaa.edu.cn).

M. Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: zhou@njit.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JAS.2018.7511189

LF model on an HiDS matrix can be transformed into building an SLFN on incomplete input features. Since randomized learning is an effective learning scheme for building an SLFN, it becomes possible to adapt its principle to LF analysis of HiDS matrices. However, current randomized learning techniques are all designed for complete inputs [18], [19]. Therefore, further efforts are necessary for incorporating the principle of randomized learning into an LF model, which are designed for HiDS matrices with a mass of unknown data [13].

In this work, we aim to develop a randomized latent factor (RLF) model with high computational efficiency in the model building process. The main idea is to incorporate the principle of randomized learning into an LF model and make necessary adaptions to the model building process for handling the sparsity of an HiDS matrix. The main contributions of this work include:

1) RLF model and biased and randomized latent factor (BRLF) model, which are achieved by incorporating the principle of randomized learning into the LF analysis of HiDS matrices;

2) Algorithm design and analysis for the RLF/BRLF models;

3) Empirical studies on three HiDS matrices from industrial applications.

To the authors' best knowledge, such efforts have been never seen in any previous work.

The rest of this paper is organized as follows: Section II investigates the related works; Section III presents the RLF/BRLF models; Section IV provides and analyzes the experimental results; and finally, Section V concludes this paper.

II. Related Work

A. LF Models for HiDS Matrices

An LF model approximates an HiDS matrix by extracting its LFs based on its known entries only, thereby achieving high computational efficiency that is linearly related to the number of its known entries [12], [13]. Meanwhile, since the dimension of an LF space can be set low without impairing an LF model's performance, an LF model's storage cost is also linearly related to the number of involved entities, e.g., user and item counts in recommender systems [2]. Hence, the computational and storage burden of an LF model is easy to resolve even in context of big data [12], [15].

Let $\boldsymbol{R}^{|U|\times|I|}$ denote the target HiDS matrix where U and I denote its row and column entity sets, Λ and Ω denote its known and unknown entry sets, and $r_{u,i}$ denote its entry at the uth row and ith column describing the relationship between entities $u \in U$ and $i \in I$, respectively. To represent $\boldsymbol{R}$, an LF model builds its rank-f approximation $\hat{\boldsymbol{R}} = \boldsymbol{H}\beta$ where f denotes the LF space dimension and $f \ll \min\{|U|, |I|\}$. Note that $\boldsymbol{H}^{|U|\times f}$ and $\beta^{f\times|I|}$ denote the LF matrices corresponding to U and I, respectively. Since $\boldsymbol{R}$ is an HiDS matrix, we have $|\Lambda| \ll |\Omega|$, i.e., known entries are far fewer than unknown ones. $\boldsymbol{H}$ and β are built based on Λ only to achieve high efficiency.

To do so, an objective method to measure the differences between Λ and the corresponding entry set in $\hat{\boldsymbol{R}}$ is desired. With Euclidean distance, such an objective function is given as:

$$\begin{aligned}\arg\min_{\boldsymbol{H},\beta} \beta\varepsilon(\boldsymbol{H},\beta) = \frac{1}{2}[&\sum_{r_{u,i}\in\Lambda}(r_{u,i} - \sum_{k=1}^{f} h_{u,k}\beta_{k,i})^2 \\ &+\frac{1}{C}(\sum_{k=1}^{f} h_{u,k}^2 + \sum_{k=1}^{f}\beta_{k,i}^2)\ ] \end{aligned} \quad (1)$$

where the term behind $1/C$ is the regularization term to avoid overfitting, and the constant C controls the regularization effect.

By solving (1), $\boldsymbol{H}$ and β are achieved on Λ only [14], [16], [18], [23]. Note that (1) is bilinear and non-convex. Hence, it is impossible to solve it analytically. To obtain desired LFs, i.e., $\boldsymbol{H}$ and β, current models all rely on iterative solvers like the stochastic gradient decent (SGD) [12], alternating least squares (ALS) [24], single element dependent and non-negative multiplicative update [13] and the second-order solver for LF analysis [25]. Representative LF models include the biased regularized incremental simultaneous MF (BRISMF) model [15], SVD++ model [12], probabilistic MF model [26], nonparametric MF model [27], weighted trace-norm regularization-based model [28], non-negative LF model [13], and second-order LF model [25]. These models perform LF analysis of HiDS matrices effectively. Meanwhile, the idea of LF analysis is also widely adopted in various HiDS data analysis tasks, like the community detection in social networks [8], [9], user preference modeling in recommender systems [2], [4], [5], [12], and mechanical fault detection in mechanical systems [29].

B. Randomized Learning for SLFN

Randomized learning techniques are originally designed for building SLFN-based learning models with high efficiency [18], [19]. With a randomized learning scheme, an SLFN is trained as follows [18], [19]: 1) the mapping step, where the input data is mapped into the hidden layer consisting of the randomly-generated input-weights and biases; and 2) the solving step, where the output matrix is analytically solved based on these randomly generated weights and biases to minimize the generalized error.

The key point of randomized learning is that the input-weights and biases are randomly generated, without the need of back propagation [16]−[19]. Hence, the hidden weights are actually pre-guessed based on the information in the input data, making the outputs analytically savable. From this point of view, it avoids solving the desired parameters iteratively, thereby achieving very high computational efficiency. However, current randomized learning schemes are all designed for full inputs without missing data, so they are not applicable for LF analysis where the inputs are HiDS matrices with a very high percentage of unknown data [2], [12]. Hence, it is attractive to adapt randomized learning to LF analysis of HiDS matrices.

III. Randomized Latent Factor Models

A. Symbols of an RLF Model

To implement an RLF model, the first step is to describe the process of LF extraction with an SLFN. Such an SLFN is depicted in Fig. 1. As shown in Fig. 1, the SLFN denotes that an RLF model has $|U|$ inputs nodes in the input layer, f hidden nodes in the hidden layer, and $|I|$ output nodes in the output layer. Hence, the target matrix $\boldsymbol{R}^{|U|\times|I|}$ is taken as the inputs and training outputs simultaneously, which is similar to the structure of an auto-encoder [15], [23]. Consequently, the hidden outputs $\boldsymbol{H}^{|U|\times f}$ and hidden weights matrix $\beta^{f\times|I|}$ are the desired LF matrices that describe the output matrix $\boldsymbol{R}$ with $\boldsymbol{H}\beta$. We summarize the symbols used in an RLF model in Table I.

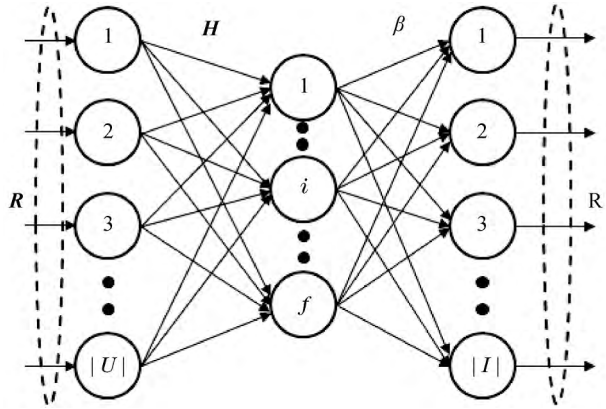


Fig. 1. Structure of the RLF model for HiDS matrices.

TABLE I
Parameters Involved in an RLF Model

Parameter	Description
$\boldsymbol{R}^{\|U\|\times\|I\|}$	Inputs
	Training outputs
$\|U\|$	Number of nodes in input layer
$\|I\|$	Number of nodes in output layer
f	Number of nodes in hidden layer
	Dimension of the LF space
$\boldsymbol{H}^{\|U\|\times f}$	Output matrix of the hidden layer
	LF matrix corresponding to entity set U
$\beta^{f\times\|I\|}$	Output weight matrix
	LF matrix corresponding to entity set I

By combining Fig. 1 and Table I, we reformulate an RLF model's objective function as follows:

$$\arg\min_{\boldsymbol{H},\beta} \varepsilon(\boldsymbol{H},\beta) = \boldsymbol{C}(\boldsymbol{R}-\boldsymbol{H}\beta)^2 + \|\boldsymbol{H}\|_F^2 + \|\beta\|_F^2 \qquad (2)$$

where $\|\cdot\|_F$ computes the Frobenius norm of a matrix. Note that the Frobenius norms of $\boldsymbol{H}$ and β play the role of regularization terms. Note that the objective function (2) and one in a classical SLFN [23] is generally the same; however, in our context, the target matrix $\boldsymbol{R}$ is HiDS with numerous missing data, making our problem sharply different from an SLFN in the following aspects:

a) $\boldsymbol{R}$ is an HiDS matrix with numerous missing entries. As a consequence, (2) is not analytically solvable with randomized learning directly; and

b) $\boldsymbol{H}$ is one of the desired LF matrices and vital in deciding the performance of the output model. Hence, to make $\boldsymbol{H}$ depend on randomly-generated weights and biases would only impair the performance of the resultant model.

Next we show how to address these two issues.

B. Addressing the Sparsity of an HiDS Matrix

First of all, we consider adapting the SLFN-style objective (1) to an HiDS matrix to obtain a solvable objective for an RLF model. Therefore, (1) should be reformulated to concentrate on Λ rather than $\boldsymbol{R}$ as in an LF model. To do so, we expand the Euclidean distance along with the regularization terms with respect to each single element $r_{u,i} \in \Lambda$ to obtain:

$$\arg\min_{\boldsymbol{H},\beta} \varepsilon_{\mathbf{RLF}}(\boldsymbol{H},\beta) = \sum_{r_{u,i}\in\Lambda} \left(\mathbf{C}\left(r_{u,i} - h_u\beta_{(i)}\right)^2 + \|h_u\|^2 + \|\beta_{(i)}\|^2 \right) \qquad (3)$$

Note that in (3), we apply the regularization to each error corresponding to an instance $r_{u,i} \in \Lambda$. Through such a strategy, we let the regularization term represent the sparsity of $\boldsymbol{R}$ [13], [24].

Based on the above inferences, we successfully build the objective function of our RLF model such that it depends on the known entries of an HiDS matrix only.

C. Guessing Step

According to Fig. 1, given $\boldsymbol{R}$, the hidden outputs $\boldsymbol{H}$ depends on the activation function with respect to the hidden weight matrix $\boldsymbol{A}^{f\times|I|}$ and length-f bias vector $\boldsymbol{B}$, formulated as:

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{h}(\boldsymbol{r}_1) \\ \vdots \\ \boldsymbol{h}(\boldsymbol{r}_{|U|}) \end{bmatrix} = \begin{bmatrix} g(\boldsymbol{a}_1\boldsymbol{r}_1^T + b_1) & \cdots & g(\boldsymbol{a}_f\boldsymbol{r}_1^T + b_f) \\ \vdots & \ddots & \vdots \\ g(\boldsymbol{a}_1\boldsymbol{r}_{|U|}^T + b_1) & \cdots & g(\boldsymbol{a}_f\boldsymbol{r}_{|U|}^T + b_f) \end{bmatrix} \qquad (4)$$

where $\boldsymbol{r}_u$ denotes the uth row vector of $\boldsymbol{R}$, $\boldsymbol{a}_k$ denotes the kth row vector of $\boldsymbol{A}$, $\boldsymbol{b}_k$ denotes the kth element of $\boldsymbol{B}$, and g denotes the activation function, respectively.

Algorithm 1: PROCEDURE PROD_AR

Input: u,$\boldsymbol{a}_k$	
Operation	Cost
initialize RST=0	$\Theta(1)$
for $r_{u,i} \in \Lambda(u)$	$\times\|\Lambda(u)\|$
RST = RST + $r_{u,i} \times a_{k,i}$	$\Theta(1)$
end for	–
Output: RST	$\Theta(1)$

In a standard SLFN, $\boldsymbol{A}$ and $\boldsymbol{B}$ should be trained via the backward propagation algorithms [14]. However, following the principle of randomized learning, they are generated randomly for alleviating the computational burden. Nonetheless, due to the sparsity of $\boldsymbol{R}$, $\boldsymbol{r}_u$ is also incomplete for $\forall u \in U$, making (4) intractable. To address this issue, we adopt the Procedure **PROD_AR** to compute $\mathbf{a}_k{}^T\boldsymbol{R}_u$. Note that in Procedure **PROD_AR**, $\Lambda(u)$ denotes the subset of Λ related to entity u.

Since $\boldsymbol{R}$ is an HiDS matrix, for $\forall u \in U$ we have $|\Lambda(u)| \ll |I|$. With **PROD_AR**, we make $\boldsymbol{H}$ focus on Λ only.

In terms of g, there are many choices as discussed in prior work [14], [30]−[32], e.g., the sigmoidal functions as well as the radial basis, sine, cosine, exponential, and many nonregular functions. It is interesting to validate RLF's performance with different activation functions. However, thorough investigations into this issue goes beyond the scope of this paper and will be done in our future works. In this paper we simply choose the exponential function as the activation function, i.e., we let $g(x) = e^x$ where e denotes the Euler's constant.

D. β-Step

Based on the randomly-generated $\boldsymbol{A}$ and $\boldsymbol{B}$, $\boldsymbol{H}$ is directly obtained. However, due to the sparsity of $\boldsymbol{R}$, RLF's objective function (2) relies on Λ along with related row and column vectors from $\boldsymbol{H}$ and β, respectively. Hence, we cannot adopt the matrix-dependent analytic solution as in an SLFN to obtain β. Instead, we solve (2) depending on each column vector of β by denoting $\beta = [\beta_{(1)}, \ldots, \beta_{(|I|)}]$. Naturally, since each $\boldsymbol{h}_u$ is obtained with randomly-generated $\boldsymbol{A}$ and $\boldsymbol{B}$ through Procedure **PROD_AR**, (3) is analytically solvable with respect to each single LF vector $\beta_{(i)}$ with $\forall i \in I$. The derivative of ε_{RLF} with respect to $\beta_{(i)}$ is:

$$\frac{\partial \varepsilon_{RLF}}{\partial \beta_{(i)}} = 2\beta_{(i)}|\Lambda(i)| - 2C \sum_{r_{u,i} \in \Lambda(i)} \boldsymbol{h}_u^T r_{u,i} + 2C \sum_{u \in U(i)} \boldsymbol{h}_u^T \boldsymbol{h}_u \beta_{(i)} \tag{5}$$

Note that in (5), $U(i)$ denotes the subset of U where each $u \in U(i))$ corresponds to a unique instance $r_{u,i} \in \Lambda(i)$. To achieve a local minimum of (3), we can set (5) at zero to achieve the following expression:

$$\left(\frac{|\Lambda(i)|}{C} \boldsymbol{I} + \sum_{u \in U(i)} \boldsymbol{h}_u^T \boldsymbol{h}_u \right) \beta_{(i)} = \sum_{r_{u,i} \in \Lambda(i)} \boldsymbol{h}_u^T r_{u,i} \tag{6}$$

where $\boldsymbol{I}$ denotes an f-rank identity matrix whose diagonal elements are equal to 1.

As discussed in the prior work [12], [15], when building an LF model on an HiDS matrix, the condition $f \ll \min\{|U|, |I|\}$ is always fulfilled since with $f > \min\{|U|, |I|\}$ the resulting $\hat{\boldsymbol{R}}$ is no longer a low rank approximation to $\boldsymbol{R}$. Hence, $\sum_{u \in U(i)} \boldsymbol{h}_u^T \boldsymbol{h}_u$ is full-rank, making the following solution to (6) exist:

$$\beta_{(i)} = \left(\frac{|\Lambda(i)|}{C} \boldsymbol{I} + \sum_{u \in U(i)} \boldsymbol{h}_u^T \boldsymbol{h}_u \right)^{-1} \sum_{r_{u,i} \in \Lambda(i)} \boldsymbol{h}_u^T r_{u,i} \tag{7}$$

with (7), we solve β based on $\boldsymbol{H}$.

E. $\boldsymbol{H}$-Step

With randomized learning, a standard SLFN solves the output weights β only to fit the training outputs. However, as depicted in Fig. 1 and Table I, we see that both $\boldsymbol{H}$ and β are desired LF matrices deciding the performance of the resultant model. Hence, similar to the ALS-based solver [24], we design the $\boldsymbol{H}$-step after the β-step.

Note that in the $\boldsymbol{H}$-step, $\boldsymbol{H}$ is taken as the decision parameter in (2). To deal with the sparsity of $\boldsymbol{R}$, we have to analytically solve each row vector of $\boldsymbol{H}$, i.e., $\boldsymbol{h}_u \in \{h_1, \cdots, h_{|U|}\}$ with $\boldsymbol{H} = [h_1, \cdots, h_{|U|}]^T$,thereby resulting in:

$$\frac{\partial \varepsilon_{RLF}}{\partial \boldsymbol{h}_u} = 2\boldsymbol{h}_u|\Lambda(u)| - 2C \sum_{r_{u,i} \in \Lambda(u)} r_{u,i}\beta_{(i)}^T + 2C \sum_{i \in I(u)} h_u \beta_{(i)} \beta_{(i)}^T. \tag{8}$$

In (8), $\Lambda(u)$ denotes the subset of Λ related with the active entity $u \in U$, and $I(u)$ denotes the subset of I where each $u \in I(u)$ corresponds to a unique instance $r_{u,i} \in \Lambda(u)$, respectively. Similar to (6), we solve (8) to obtain:

$$\boldsymbol{h}_u = \left(\sum_{r_{u,i} \in \Lambda(u)} r_{u,i} \beta_{(i)}^T \right) \left(\frac{|\Lambda(u)|}{C} \boldsymbol{I} + \sum_{i \in I(u)} \beta_{(i)} \beta_{(i)}^T \right)^{-1}. \tag{9}$$

Note that (9) is also based on the condition $f \ll \min\{|U|, |I|\}$ which ensures the full rank of $\sum_{i \in I(u)} \beta_{(i)} \beta_{(i)}^T$.

It should be pointed out that (6) and (9) depends on the inverse of $\left(\frac{|\Lambda(i)|}{C} \boldsymbol{I} + \sum_{u \in U(i)} \boldsymbol{h}_u^T \boldsymbol{h}_u \right)$ and $\left(\frac{|\Lambda(u)|}{C} \boldsymbol{I} + \sum_{i \in I(u)} \beta_{(i)} \beta_{(i)}^T \right)$.However, note that both matrices are of dimension $f \times f$. Consequently, the cost by inversing them is only $\Theta(f^2)$ and easy to resolve in practice when f is small, e.g., 20 as frequently adopted in prior works [12], [15].

F. RLF with Linear Biases

As discussed in prior works [12], [13], [15], an LF model can be extended by integrating linear bias schemes into it. As discussed in [12], [13], [15], [26], given $\boldsymbol{R}^{|U| \times |I|}$, commonly adopted biases include the global average μ, the row bias $\boldsymbol{C}^{|U|}$ and column bias $\boldsymbol{D}^{|I|}$. With them, an LF model approximates $r_{u,i} \in \Lambda$ as follows:

$$\hat{r}_{u,i} = \mu + c_u + d_i + \boldsymbol{h}_u \beta_{(i)}. \tag{10}$$

where c_u denotes the uth element of $\boldsymbol{C}$, and b_i denotes the ith element of $\boldsymbol{D}$, respectively. Then μ is computed as follows:

$$\mu = \frac{\sum_{r_{u,i} \in \Lambda}}{|\Lambda|} \tag{11}$$

After that, linear biases in $\boldsymbol{C}$ and $\boldsymbol{D}$ should be trained along with the desired LFs. A convenient strategy to do so is fixing the first column of $\boldsymbol{H}$ and second row of β at one [15]. Note that with such a strategy, $\boldsymbol{H}$ and β are transformed into the following expressions:

$$\begin{cases} \boldsymbol{H} = [1, \boldsymbol{C}^T, \boldsymbol{h}_{(1)}, \ldots, \boldsymbol{h}_{(f)}] \\ \beta = [\boldsymbol{D}^T, 1, \beta_1, \ldots, \beta_{|f|}]^T. \end{cases} \tag{12}$$

In (12) we slightly simplify the symbols by letting 1 denote a row/column vector filled with one. Thus, the effects of linear biases will be included in the training. For example, given $r_{u,i}$, the model computes its approximation $\hat{r}_{u,i}$ as follows:

$$\begin{aligned} \hat{r}_{u,i} &= \mu + \boldsymbol{h}_u \beta_{(i)} \\ &= [1, c_u, h_{u,1}, \ldots, h_{u,f}] \cdot [d_i, 1, \beta_{1,i}, \ldots, \beta_{f,i}]^T \end{aligned}$$

$$= \quad \mu + c_u + d_i + \sum_{k=1} f h_{u,k}\beta_{k,i}. \tag{13}$$

with (12) and (13), we do not need to modify the $\boldsymbol{H}$ and β steps. However, the guessing step should be applied to $\boldsymbol{C}$. Hence, we slightly adjust the guessing step as follows:

$$\boldsymbol{H}=\begin{bmatrix} 1,\ g(a_C r_1^T + b_C) & g(a_1 r_1^T + b_1) & \cdots & g(a_f r_1^T + b_f) \\ \vdots & \vdots & \ddots & \vdots \\ 1,\ g(a_C r_{|U|}^T + b_C) & g(a_1 r_{|U|}^T + b_1) & \cdots & g(a_f r_{|U|}^T + b_f) \end{bmatrix} \tag{14}$$

where a_C and b_C denotes the randomly generated weights and biases for the linear bias vector $\boldsymbol{C}$, respectively. With (12)−(14), we achieve the biased and randomized latent factor (BRLF) model.

G. Algorithm Design and Analysis

Based on the above design, we develop Algorithm RLF. Note that for the BRLF model, the algorithm is exactly the same except that the first column of $\boldsymbol{H}$ and second row of β are fixed at one during the training process. As discussed in Sections III(A)−(E), RLF consists of three phases: a) guessing the hidden weight matrix $\boldsymbol{H}$ with randomly generated $\boldsymbol{a}$ and $\boldsymbol{b}$; b) executing the β-step to solve β based on $\boldsymbol{H}$ and (7); c) recomputing $\boldsymbol{H}$ based on the solved β and (9). Note that similar to the ALS-based solver for latent factor models, the RLF algorithm is essentially an iterative process. However, RLF only runs one iteration to train β and $\boldsymbol{H}$. Moreover, in each phase, Algorithm RLF focuses on Λ rather than the whole $\boldsymbol{R}$ to achieve high computational efficiency. Hence, the model's computational burden can be reduced greatly.

Based on Algorithm 2, we summarize the computational cost of RLF as follows:

$$\begin{aligned} T_{RLF} =& \Theta(|U| \times f + 2|I| \times f + f^2 + 3f) + \Theta(f \times |\Lambda|) \\ &+ \Theta(|I| \times (f^3 + 2f^2 + f) + |\Lambda| \times (f^2 + f)) \\ &+ \Theta(|U| \times (f^3 + 2f^2 + f) + |\Lambda| \times (f^2 + f)) \\ \approx& \Theta((|U| + |\Lambda| + |I|) \times f) \\ &+ \Theta((|U| + |I|) \times f^3 + |\Lambda| \times f^2) \\ \approx& \Theta((|U| + |I|) \times f^3 + |\Lambda| \times f^2) \\ =& \Theta((|\Lambda| + (|U| + |I|) \times f) \times f^2). \end{aligned} \tag{15}$$

Note that in (15) we reasonably omit the lower-order-terms to achieve the final result. Note that f is a positive constant and can be set small without impairing an LF model's performance. Hence, we conclude that **RLF**'s computational cost is linear with the term $(|\Lambda| + (|U| + |I|) \times f)$. Moreover, in a HiDS matrix $\boldsymbol{R}$ we usually have $\max\{|U|, |I|\} \ll |\Lambda|$.Hence, with small f, e.g., 20, RLF's computational cost is linear with $|\Lambda|$. Nonetheless, this inference cannot hold as f grows larger and larger.

In terms of its storage cost, as depicted in **RLF**, we adopt three auxiliary matrices/arrays, i.e., $\boldsymbol{c}$, $\boldsymbol{d}$ and $\boldsymbol{s}$, to cache the interim results in the algorithm for improving its efficiency. Hence, RLF's storage cost is:

$$S_{RLF} = \Theta((|U| + |I|) \times f)) \tag{16}$$

As depicted in (16), RLF's storage cost is linear with the term $(|U| + |I|)$, i.e., the total number of involved entities in $\boldsymbol{R}$.

Algorithm 2: ALGORITHM RLF

Input: Λ, I, U, g(x), f

Operation	Cost
initialize $\boldsymbol{H}^{|U| \times f}$	$\Theta(|U| \times f)$
initialize $\beta^{f \times |I|}$	$\Theta(f \times |I|)$
initialize $\boldsymbol{a}^{f \times |I|}$ randomly	$\Theta(f \times |I|)$
initialize $\boldsymbol{b}^{f}$ randomly	$\Theta(f)$
initialize $\boldsymbol{c}^{f \times f}$	$\Theta(f \times f)$
initialize $\boldsymbol{d}^{f \times 1}$	$\Theta(f)$
initialize $\boldsymbol{s}^{f}$	$\Theta(f)$
/-Note: Guessing step-/	
for $k = 1$ **to** f	$\times f$
for $u \in U$	$\sum_{u \in U}$
$h_{u,k} = g(\textbf{PROD\_AR}(u, \mathbf{a}_k) + b_k)^*$	$\Theta(|\Lambda(u)|)$
end for	–
end for	–
/-Note: β-step-/	
for $i \in I$	$\sum_{i \in I}$
reset c = zero	$\Theta(f \times f)$
reset d = zero	$\Theta(f)$
for $u \in U(i)$	$\times |\Lambda(i)|^{**}$
$\boldsymbol{c} = \boldsymbol{c} + \boldsymbol{h}_u^T \boldsymbol{h}_u$	$\Theta(f \times f)$
$\boldsymbol{d} = \boldsymbol{d} + \boldsymbol{h}_u^T r_{u,i}$	$\Theta(f)$
end for	–
$\beta_{(i)} = \left(\frac{|\Lambda(i)|}{C}\boldsymbol{I} + \boldsymbol{c}\right)^{-1} \boldsymbol{d}$	$\Theta(f^3 + f^2)$
end for	–
/-Note: $\boldsymbol{H}$-step-/	
for $u \in U$	$\sum_{u \in U}$
reset c = zero	$\Theta(f \times f)$
reset s = zero	$\Theta(f)$
for $i \in I(u)$	$\times |\Lambda(u)|^{***}$
$\boldsymbol{c} = \boldsymbol{c} + \beta_{(i)}\beta_{(i)}^T$	$\Theta(f \times f)$
$\boldsymbol{s} = \boldsymbol{s} + r_{u,i}\beta_{(i)}^T$	$\Theta(f)$
end for	–
$\boldsymbol{h}_u = \boldsymbol{p}\left(\frac{|\Lambda(u)|}{C}\boldsymbol{I} + \boldsymbol{c}\right)^{-1} \boldsymbol{d}$	$\Theta(f^3 + f^2)$
end for	–

Output: LF matrices $\boldsymbol{H}$ and β

IV. Experimental Results

A. General Settings

Evaluation Protocol: For industrial applications [12], [13], [28], one major motivation to factorize an HiDS matrix is to predict its missing data for implementing the full relationship mapping among involved entities. Owing to its popularity and usefulness, we adopt it as the evaluation protocol in our experiments. A model's prediction accuracy for the missing entries of an HiDS matrix is measured by the root mean squared error (RMSE) and mean absolute error (MAE) [11],

[12]:

$$\begin{cases} RMSE = \sqrt{\dfrac{\left(\sum_{r_{v,j}\in\Gamma}(r_{v,j}-\hat{r}_{v,j})^2\right)}{|\Gamma|}}, \\ MAE = \dfrac{\left(\sum_{r_{v,j}\in\Gamma}|r_{v,j}-\hat{r}_{v,j}|_{abs}\right)}{|\Gamma|}. \end{cases} \tag{17}$$

where Γ denotes the validation set and is disjoint with Λ, $\hat{r}_{v,j}$ denotes the generated prediction for the testing instance $r_{v,j} \in \Gamma$, which simulates the missing entry at the *v*th row and *j*th column of a target HiDS matrix, and $|\cdot|_{abs}$ computes the absolute value of a given number, respectively.

Meanwhile, we are concerned with the computational efficiency of tested models. Hence, we record their consumed time to converge. All experiments are conducted on a Tablet with a 2.5 GHz i7 CPU and 16 GB RAM. The programming language is JAVA SE 7U60.

Datasets: Three HiDS matrices are included in our experiments. All are collected by industrial companies.

1) D1: Douban dataset. It is collected from the largest online book, movie and music database in China Douban [33]. It contains 16 830 839 ratings in the scale of [1, 5] from 129 490 users on 58 541 items. Its known data density is 0.22 % only.

2) D2: ML20M dataset. It is collected by the MovieLens system [34] maintained by the GroupLens research team, and contains 20 000 263 entries in [0.5, 5], by 138 493 users on 26 744 movies. Its known data density is 0.54 % only.

3) D3: Crystm dataset. It is from the University of Florida sparse matrix collection [35], and contains 583 770 entries denoting the vibrational stiffness data of a piece of specific kinds of material among its 24 696 nodes. It is known data density is 0.096 % only.

Note that D1 and D2 are from recommender systems, while D3 is from more conventionally industrial applications.

For obtaining objective and fair results, we adopt the 80 %−20 % train-test settings and five-fold cross-validations on all datasets. More specifically, we split the known entry set of each HiDS matrix into five equally-sized, disjoint subsets at random. Each time, we select four subsets as the training set Λ to train a model and the remaining subset as the testing set Γ. This process is sequentially repeated for five times to obtain the final results.

B. Effects of C and f

As depicted in (7)−(12) and Algorithm RLF, the RLF/BRLF model's performance relies on f and C, which decide the LF dimension and regularization effect, respectively. Hence, it is necessary to check the effects of these hyper parameters on the performance of the resulting model.

Fig. 2 depicts RLF's prediction accuracy as f increases from 5 to 80 and C increases from 2^{-10} to 2^{10}. Table II shows RLF's time cost as f increases from 5 to 80. Note that Fig. 2 and Table II show the results of the RLF model on D1. We can reach the similar conclusions on D2 and D3 for the RLF model and on D1−D3 for the BRLF model. From them, we have the following findings:

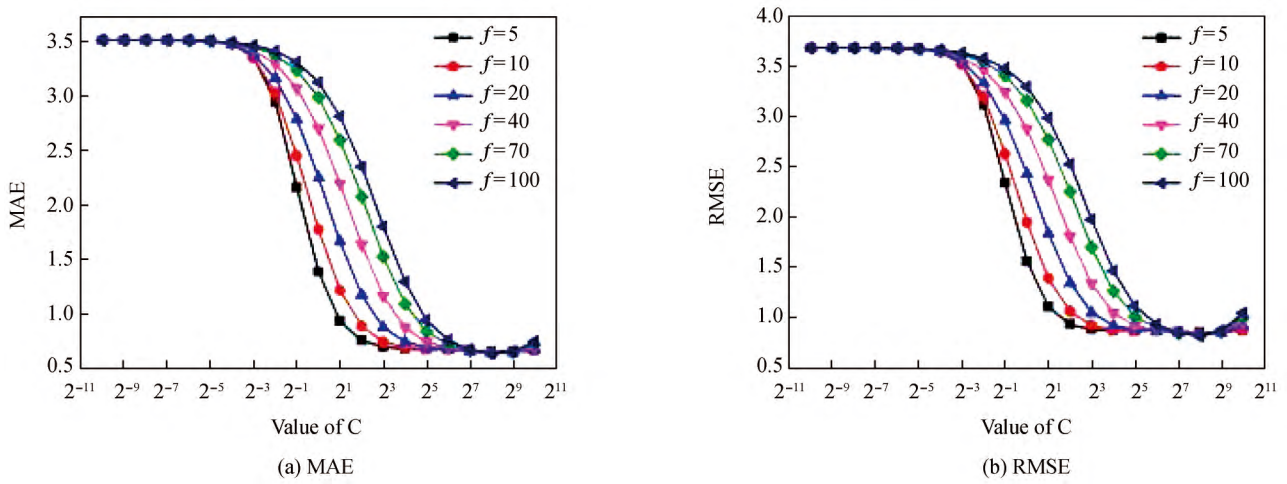
a) An RLF/BRLF model's prediction accuracy is sensitive to *C*. As depicted in Fig. 2, with different *f*, RLF's prediction error is always closely connected with the value of *C*. We can see that with different values of *f*, RLF's prediction error with large *C*, i.e., 2^{10}, is obviously higher than that with the optimal value of *C*, i.e., 2^8. For instance, as shown in Fig. 2(a), with $f = 5$ on D1, RLF's MAE is 0.5758 with the optimal value 2^8 for *C*, but 0.5844 with $C = 2^{10}$; the accuracy gap is 1.47 %. Similar situations can also be found in Fig. 2(b). Note that the regularization effects in RLF/BRLF model decrease as *C* increases as depicted in (7)−(12). Hence, this phenomenon actually supports the necessity to apply regularization to RLF/BRLF model.

On the other hand, with too much regularization brought on by a small *C*, RLF cannot achieve a local optimum. As depicted in Fig. 2(a), with $f = 5$ on D1, RLF's MAE is larger than two with $C \ll 0.5$, much higher than its lowest MAE 0.5758 with $C = 2^8$. Similar conclusions can be made according to Fig. 2(b). Note that with *C* being too small, objective (2) is bent to fit the regularization terms, resulting in poor prediction accuracy. To summarize, it is necessary to tune *C* with care to ensure RLF's high prediction accuracy for missing data.

b) Like other LF models [12], [13], RLF's prediction accuracy also relies on the LF dimension, i.e., *f*. Note that the optimal value of f is data dependent, which can be seen from Table IV. However, it is very interesting that RLF tends to achieve high prediction accuracy with a small *f*, as depicted in Fig. 2. For instance, on D1, it achieves the lowest MAE 0.5742 and RMSE 0.7351 with $f = 20$ and $C = 256$ (which is the optimal value of *C* with $f = 20$), as shown in Figs. 2a and 2b. However, its MAE and RMSE are 0.5872 and 7481 with $f = 100$ and $C = 256$ (which is the optimal value of *C* with $f = 100$). Thus, we see that as f increase from 20 to 100, RLF's MAE and RMSE increases at 2.21 % and 1.74 %.

Note that as discussed in the prior work [12], [15], [26], an LF model's feature dimension f actually decides the rank of the resulting approximation to a target HiDS matrix. Its prediction accuracy for the target matrix's missing data usually increases as f increases under its actual rank. Nonetheless, this rule does not hold true for RLF. To be shown later, D1's actual rank is obviously larger than the optimal value of f in RLF. As shown in Fig. 1, RLF is actually a special case of the SLFN. Its objective function depends not only on the linear combinations of $\boldsymbol{H}$ and β but also on the non-linear mapping from $\boldsymbol{R}$, $\boldsymbol{a}$ and $\boldsymbol{b}$ to $\boldsymbol{H}$ through g. Such a mapping process makes $\boldsymbol{H}$ and β approximate Λ in a non-linear way, thereby changing the characteristics of the resulting low-rank approximation to $\boldsymbol{R}$. Hence, although its rank still increases as f increases, its representativeness of Λ is not improved.

c) From Section III-G, we see that RLF's time cost is asymptotically quadratic with *f*. This inference is also supported by the experimental results, as depicted in Table II. We also see that RLF's time cost is not strictly quadratic with *f*, due to the lower-order terms in its time cost. As a result, the time cost of the model increases drastically as f becomes larger, without positive effects on the accuracy of the resultant model.



(a) MAE (b) RMSE

Fig. 2. RLFs prediction error with f in [5, 80] and C in $[2^{-10}, 2^{10}]$.

TABLE II
RLF'S TIME COST AS F INCREASES FROM 5 TO 80 ON D1

Value of f	5	10	20	40	70	100
Time Cost (s)	5.00	8.66	20.99	63.93	239.05	483.97

C. Comparison Between RLF and State-of-the-art Models

In this part of the experiment, we compare RLF with three sophisticated and state-of-the-art LF models in terms of both prediction accuracy and computational efficiency on all datasets. Their details are given in Table III.

TABLE III
DETAILS OF COMPARED MODELS

No.	M1	M2	M3	M4	M5
Name	RLF	BRLF	BRISMF (Takacs *et al.* 2009)	WALS (Hu *et al.* 2008)	SVD++ (Koren *et al.* 2009)

Note that we choose M3−M5 as the rival models based on the following considerations: a) they are representative and highly efficient LF models able to address HiDS matrices; and b) M3 and M5 relies on stochastic gradient descent, while M4 relies on alternating least squares. Both optimization techniques are frequently employed to address the optimization tasks in LF models. Hence, comparing RLF with M3−M5 can provide us with objective and fair results relying on different optimization techniques.

In terms of the hyper parameters in each model, to ensure a fair comparison, we adopt the following settings:

1) For all compared models, we compare their prediction accuracy and computational efficiency as f increases from 5 to 100 to conduct a thorough comparison;

2) M3 and M5 depends on the learning rate and regularization coefficient. Following instructions in [12], [15], on all datasets we first draw a grid search to seek optimal values on one fold, and then apply the same settings to the other four. The search scale for them is in the scale of [0.01, 0.05] as demonstrated in [12], [15];

3) M4 depends on a regularization coefficient. Following [24], on all datasets, we first tune it on one fold, and then apply the same settings to the other four. Its tuning scale is also [0.01, 0.05], as discussed in [24]; and As shown in Section IV-B, M1 and M2 rely on C. On all datasets, we set $C = 2^8$.

Fig. 3 depicts the comparison among tested models in prediction accuracy and computational efficiency. Table IV records the time cost of each model with $f = 20$, respectively. From these results, we have the following findings:

1) RLF and BRLF model's prediction accuracy is comparable to their peers. On D1, M1 and M2 are able to outperform M3−M5 in fact. Moreover, M2 is able to generate the most accurate predictions for missing entries on D1. As depicted in Fig. 3(a), The lowest MAE achieved by M2 is 0.5728 with $f = 20$, which is about 0.24 % lower than 0.5742 by M1 with $f = 20$, 0.62 % lower than 0.5764 by M3 with $f = 100$, 0.75 % lower than 0.5771 by M4 with $f = 100$, and 0.52 % lower than 0.5758 by M5 with $f = 100$. Similar observations can be made with the RMSE, as depicted in Fig. 3(b).

On D2, the situation is different: M1 and M2 are outperformed by M3−M5 in prediction accuracy. As shown in Fig. 3(c), M5 has the lowest MAE at 0.6150 with $f = 100$, about 2.46 % lower than 0.6305 by M1 with $f = 40$, 1.87 % lower than 0.6267 by M2 with $f = 40$, 0.18 % lower than 0.6161 by M3 with $f = 100$, and 0.74 % lower than 0.6196 by M4 with $f = 100$. This holds true in terms of the RMSE as depicted in Fig. 3(d).

On D3, M2, i.e., the BRLF model, is able to achieve higher prediction accuracy for missing data than its peers do. As depicted in Fig. 3(e), M2 achieves the lowest MAE at 0.0857 with $f = 20$, about 7.55 % lower than 0.0927 by M1 with $f = 5$, 4.46 % lower than 0.0897 by M3 with $f = 100$, 8.24 % lower than 0.0934 by M4 with $f = 100$, and 4.03 % lower than 0.0893 by M5 with $f = 100$. We can make similar conclusions in terms of the RMSE as depicted in Fig. 3(f). Considering M1, i.e., the unbiased RLF model, its prediction accuracy is higher than that of M4 but lower than that of M1 and M3 on D3.

2) From Fig. 3 we see that M3−M5's prediction error keeps decreasing as f increases. However, for M1 and M2 it is

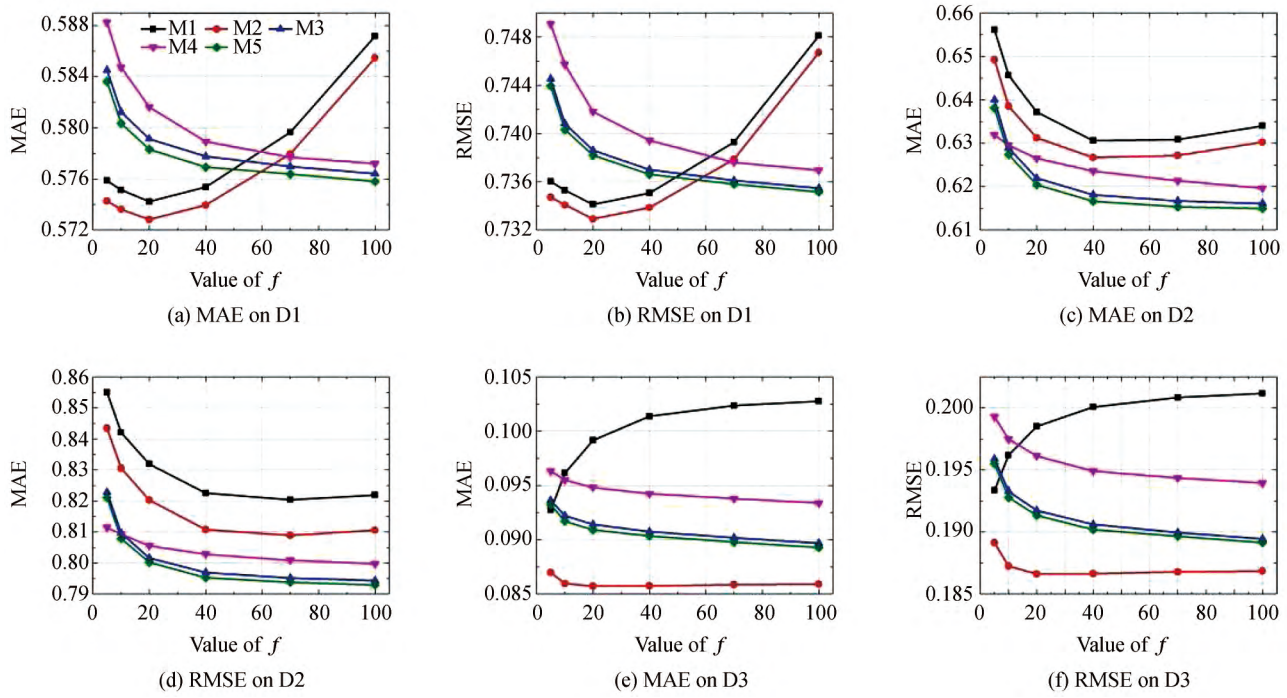


Fig. 3. Comparison among Tested Models in Prediction Accuracy. All panels share the legend in panel(a).

significantly different. As discussed in Section III, RLF and BRLF models integrate the non-linearity into the output LFs through the guessing step. As a result, they perform differently from linear LF models such as M3−M5: their small prediction error is related to the large value of f.

3) RLF and BRLF's computational efficiency is significantly higher than that of their peers, as recorded in Table IV. For instance, M1 consumes 20.99 seconds to achieve its lowest MAE on D1. This is only 2.42 % of the 866.72 seconds required by M3, 0.45 % of the 4639.35 required by M4, and 2.75 % of the 763.35 required by M5. In terms of M2, it needs 22.47 seconds to achieve the lowest MAE on D1, which is close to the time cost of M1. Similar results can also be found on D2 and D3 as recorded in Table IV.

TABLE IV
TIME COST OF EACH TESTED MODEL TO ACHIEVE THE LOWEST PREDICTION ERROR IN THE EXPERIMENTS (IN SECONDS)

Case	M1	M2	M3	M4	M5
D1(MAE)	**20.99**	**22.47**	866.72	4639.35	763.35
Value of f	**20**	**20**	1000	1000	1000
D1 (RMSE)	**20.99**	**22.47**	809.75	4343.25	719.58
Value of f	**20**	**20**	1000	1000	1000
D2 (MAE)	**73.82**	**79.04**	668.95	3655.62	713.84
Value of f	**40**	**40**	1000	1000	1000
D2 (RMSE)	**226.27**	**242.06**	640.15	3323.85	684.55
Value of f	**70**	**70**	1000	1000	1000
D3 (MAE)	**1.27**	**7.46**	77.30	493.14	68.76
Value of f	**5**	**20**	1000	1000	1000
D3 (RMSE)	**1.27**	**7.46**	71.25	471.85	60.98
Value of f	**5**	**20**	1000	1000	1000

The reason for such drastic computational time reduction is two-fold: a) as depicted in Algorithm RLF, the training process of RLF and BRLF executes only one iteration consisting of three steps, i.e., the guessing step, β-step and $\boldsymbol{H}$-step; and b) RLF and BRLF models do not require large f to achieve the lowest prediction error, as shown in Fig. 3 and Table IV.

4) In general, integrating the linear biases into the RLF model is very helpful in achieving higher prediction accuracy, as depicted in Fig. 3. As recorded in Table IV, such extension leads to a higher time cost.

5) To summarize, when compared with state-of-the-art LF models, the proposed RLF and BRLF models achieve significantly higher computational efficiency as well as competitive prediction accuracy for missing data. Hence, they provide us with a novel, effective, and highly efficient approach to LF analysis of HiDS matrices.

V. DISCUSSIONS

As shown in Section IV, the proposed RLF and BRLF models are able to achieve high computational efficiency as well as competitive accuracy for missing data prediction when performing LF analysis of HiDS matrices. In this section, we discuss several important issues regarding the characteristics of the proposed models.

1) Pros and Cons of RLF/BRLF Models: Based on Section IV, we see that the virtues of RLF/BRLF model include:

a) High computational efficiency. Owing to the training scheme based on randomized learning, RLF/BRLF models are able to achieve high computational efficiency. Compared with state-of-the-art LF models, they consume much less time.

b) Ease of implementation. RLF/BRLF models relies on a training process with only one iteration, which is very convenient to implement with any programming language.

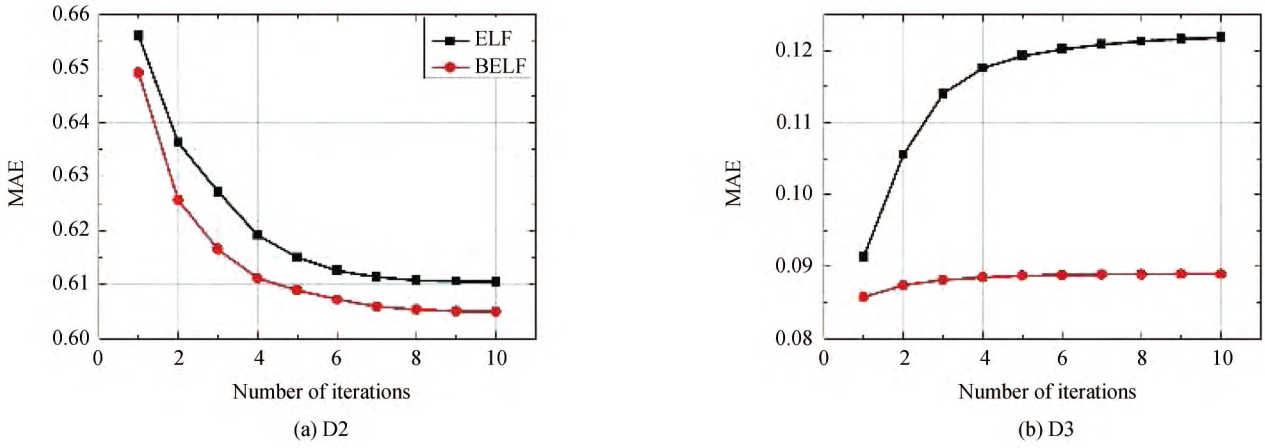


Fig. 4. MAE of RLF and BRLF models by executing the β-step and $\boldsymbol{H}$-step for multiple times on D2 and D3. Both panels share the legend in panel (a).

On the other hand, compared with state-the-art models, the limitation of RLF/BRLF is that its prediction accuracy for missing data might be lower than iterative models. As depicted in Fig. 3, RLF and BRLF model's prediction accuracy is comparable to their peers in some case, i.e., on D1. However, on D2, the situation is different: RLF and BRLF are outperformed by other iterative LF models.

Note that in static applications like recommender systems, LF analysis can be done offline. In such cases, the most concerning characteristic of a model is its accuracy, such as in deep learning scenarios. To make RLF/BRLF models fit such cases, strategies like the ensemble and hierarchical model structure [36], [37] are required for further boosting their accuracy. Nonetheless, in dynamic data environments like service selection, the training time is limited to enable fast feedback of the system in response to data variations. Under such circumstances, the high computational efficiency of the RLF/BRLF models is desired.

2) Iterative Training in RLF/BRLF: As depicted in Algorithm RLF, an RLF/BRLF models executes the guessing step, β-step, and $\boldsymbol{H}$-step only once to obtain the desired LFs based on the principle of an ELM model. Naturally, the β-step and $\boldsymbol{H}$-step can be iteratively executed like in iterative LF models. Thus, how will the RLF/BRLF models act with such settings?

Fig. 4 depicts the MAE of RLF and BRLF models by executing the β-step and $\boldsymbol{H}$-step iteratively on D2 and D3. From Fig. 4, we see that the accuracy gain brought the iterative settings in RLF/BRLF models is data-dependent. For instance, on D2 the MAE of RLF/BRLF decreases with more iterations, as shown in Fig. 4(a). Nonetheless, on D3 the MAE increases with more iterations, as shown in Fig. 4(b). However, by making RLF/BRLF iterative, the time cost also grows linearly with respect to the iteration count. Therefore, whether or not to apply the iterative settings to the RLF/BRLF model should be considered with care, according to the detailed requirements raised by the target applications.

VI. Conclusion

This work proposes to perform the LF analysis of HiDS matrices based on the principle of randomized learning. Owing to the novel training scheme based on randomized learning, the proposed RLF/BRLF models are able to achieve high computational efficiency. Compared with state-of-the-art LF models, their time cost is significantly reduced. However, their prediction accuracy for missing data might be slightly lower than the iterative LF models.

Note that in static applications [2], [12], LF analysis can be done offline. In such cases, the most significant characteristic of a model is its accuracy such as in deep learning scenarios [36], [37]. To make RLF/BRLF models fit such cases, strategies like the ensemble [38] and hierarchical model structure [31] are required to further boost their accuracy. On the other hand, in dynamic data environments like QoS prediction [6], [7], [39], [40], online recommendation [41], wireless sensor analysis [42] and other applications [43]−[45], the training time is limited to enable fast feedbacks of the system according to data variations. Under such circumstances, the high computational efficiency of the RLF/BRLF models is essentially needed.

As demonstrated by prior research [46]−[48], further constraints on the output LFs like non-negativity and symmetry are essential for boosting the resultant model's representativeness to the target data. It is interesting to develop RLF extensions under such circumstances. We plan to address these issues in our future research.

References

[1] V. Mayer-Schïnberger and K. Cukier, *Big Data: A Revolution That Will Transform How We Live, Work, and Think*. Boston, MA: Houghton Mifflin Harcourt, 2013.

[2] G. Adomavicius and 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.*, vol. 17, no. 6, pp. 734−749, Jun. 2005.

[3] X. M. Qian, H. Feng, G. H. Zhao, and T. Mei, "Personalized recommendation combining user interest and social circle," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 7, pp. 1763−1777, Jul. 2014.

[4] W. J. Luan, G. J. Liu, C. J. Jiang, and L. Qi, "Partition-based collaborative tensor factorization for poi recommendation," *IEEE/CAA J. Autom. Sinica*, vol. 4, no. 3, pp. 437−446, Jul. 2017.

[5] S. X. Gao, Z. T. Yu, L. B. Shi, X. Yan, and H. X. Song, "Review expert collaborative recommendation algorithm based on topic relationship," *IEEE/CAA J. Autom. Sinica*, vol. 2, no. 4, pp. 403−411, Oct. 2015.

[6] J. Wu, L. Chen, Y. P. Feng, Z. B. Zheng, M. C. Zhou, and Z. H. Wu, "Predicting quality of service for selection by neighborhood-based collaborative filtering," *IEEE Trans. Syst. Mand Cybern.: Syst.*, vol. 43, no. 2, pp. 428−439, Mar. 2013.

[7] Z. B. Zheng, H. Ma, M. R. Lyu, and I. King, "Qos-aware web service recommendation by collaborative filtering," *IEEE Trans. Servic. Comput.*, vol. 4, no. 2, pp. 140−152, Apr-Jun. 2011.

[8] R. Narayanam and Y. Narahari, "A shapley value-based approach to discover influential nodes in social networks," *IEEE Trans. Autom. Sci. Eng.*, vol. 8, no. 1, pp. 130−147, Jan. 2011.

[9] X. C. Cao, X. Wang, D. Jin, Y. X. Cao, and D. X. He, "Erratum: Identifying overlapping communities as well as hubs and outliers via nonnegative matrix factorization," *Sci. Rep.*, vol. 4, pp. Article No. 5788, Nov. 2014.

[10] X. L. Piao, Y. L. Hu, Y. F. Sun, B. C. Yin, and J. B. Gao, "Correlated spatio-temporal data collection in wireless sensor networks based on low rank matrix approximation and optimized node sampling," *Sensors*, vol. 14, no. 12, pp. 23137−23158, Dec. 2014.

[11] T. L. N. Nguyen and Y. Shin, "Matrix completion optimization for localization in wireless sensor networks for intelligent iot," *Sensors*, vol. 16, no. 5, pp. 722, May 2016.

[12] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30−37, Aug. 2009.

[13] X. Luo, M. C. Zhou, Y. N. Xia, and Q. S. Zhu, "An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems," *IEEE Trans. Ind. Inf.*, vol. 10, no. 2, pp. 1273−1284, May 2014.

[14] S. B. Vandenberghe, *Convex Optimization*. Cambridge: Cambridge University Press, 2009.

[15] G. Takács, I. Pilászy, B. Németh, and D. Tikk, "Scalable collaborative filtering approaches for large recommender systems," *J. Mach. Learn. Res.*, vol. 10, pp. 623−656, Dec. 2009.

[16] Y. H. Pao and Y. Takefuji, "Functional-link net computing: theory, system architecture, and functionalities," *Computer*, vol. 25, no. 5, pp. 76−79, May 1992.

[17] B. Igelnik and Y. H. Pao, "Stochastic choice of basis functions in adaptive function approximation and the functional-link net," *IEEE Trans. Neural Netw.*, vol. 6, no. 6, pp. 1320−1329, Nov. 1995.

[18] A. N. Gorban, I. Y. Tyukin, D. V. Prokhorov, and K. I. Sofeikov, "Approximation with random bases: Pro et contra," *Inf. Sci.*, vol. 364−365, pp. 129−145, Oct. 2016.

[19] S. Scardapane and D. H. Wang, "Randomness in neural networks: an overview" *Wiley Interdiscipl. Rev.: Data Min. Knowl. Discovery*, vol. 7, no. 2, Mar. 2017. DOI: 10.1002/widm.1200.

[20] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1−3, pp. 489−501, Dec. 2006.

[21] J. X. Tang, C. W. Deng, and G. B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 4, pp. 809−821, Apr. 2016.

[22] S. Li, Z. H. You, H. L. Guo, X. Luo, and Z. Q. Zhao, "Inverse-free extreme learning machine with optimal information updating," *IEEE Trans. Cybern.*, vol. 46, no. 5, pp. 1229−1241, May 2016.

[23] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. Int. Conf. Conference on Learning Representations (ICLR)*, Banff, Canada, 2014.

[24] Y. F. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Proc. 8th IEEE Int. Conf. Data Mining*, Pisa, Italy, 2009, pp. 263−272.

[25] X. Luo, M. C. Zhou, S. Li, Y. N. Xia, Z. H. You, Q. S. Zhu, and H. Leung, "An efficient second-order approach to factorize sparse matrices in recommender systems," *IEEE Trans. Ind. Inf.*, vol. 11, no. 4, pp. 946−956, Aug. 2015.

[26] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Proc. 20th Int. Conf. Neural Information Processing Systems*, Vancouver, British Columbia, Canada, 2007, pp. 1257−1264.

[27] K. Yu, S. H. Zhu, J. Lafferty, and Y. H. Gong, "Fast nonparametric matrix factorization for large-scale collaborative filtering," in *Proc. 32nd Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, Boston, MA, USA, 2009, pp. 211−218.

[28] R. Salakhutdinov and N. Srebro, "Collaborative filtering in a non-uniform world: Learning with the weighted trace norm," in *Proc. 24th Ann. Conf. Neural Information Processing Systems*, Vancouver, British Columbia, Canada, 2010, pp. 2056−2064.

[29] B. Gao, W. L. Woo, Y. T. Gui, and Z. Hong, "Unsupervised diagnostic and monitoring of defects using waveguide imaging with adaptive sparse representation," *IEEE Trans. Ind. Inf.*, vol. 12, no. 1, pp. 405−416, Feb. 2016.

[30] N. Y. Zeng, Z. D. Wang, H. Zhang, W. B. Liu, and F. E. Alsaadi, "Deep belief networks for quantitative analysis of a gold immunochromatographic strip," *Cogn. Comput.*, vol. 8, no. 4, pp. 684−692, Aug. 2016.

[31] N. Y. Zeng, H. Zhang, B. Y. Song, W. B. Liu, Y. R. Li, and A. M. Dobaie, "Facial expression recognition via learning deep sparse autoencoders," *Neurocomputing*, vol. 273, pp. 643−649, Jan. 2018.

[32] N. Y. Zeng, H. Zhang, Y. R. Li, J. L. Liang, and A. M. Dobaie, "Denoising and deblurring gold immunochromatographic strip images via gradient projection algorithms," *Neurocomputing*, vol. 247, pp. 165−172, Jul. 2017.

[33] H. Ma, I. King, and M. R. Lyu, "Learning to recommend with social trust ensemble," in *Proc. 32nd Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, Boston, MA, USA, 2009, pp. 203−210.

[34] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, "Grouplens: applying collaborative filtering to usenet news," *Commun. ACM*, vol. 40, no. 3, pp. 77−87, Mar. 1997.

[35] T. A. Davis and Y. F. Hu, "The university of Florida sparse matrix collection," *ACM Trans. Math. Softw.*, vol. 38, no. 1, pp. Article No. 1, Nov. 2011.

[36] C. Y. Lei, D. Liu, W. P. Li, Z. J. Zha, and H. Q. Li, "Comparative deep learning of hybrid representations for image recommendations," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 2016, pp. 2545−2553.

[37] H. Lee, Y. Ahn, H. Lee, S. Ha, and S. G. Lee, "Quote recommendation in dialogue using deep neural network," in *Proc. 39th Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, Pisa, Italy, 2016, pp. 957−960.

[38] X. Luo, M. C. Zhou, Y. N. Xia, Q. S. Zhu, A. C. Ammari, and A. Alabdulwahab, "Generating highly accurate predictions for missing QoS data via aggregating nonnegative latent factor models," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 3, pp. 524−537, Mar. 2016.

[39] X. Luo, M. C. Zhou, Z. D. Wang, Y. N. Xia, and Q. S. Zhu, "An

effective scheme for QoS estimation via alternating direction method-based matrix factorization," *IEEE Trans. Servic. Comput.*, 2016. DOI: 10.1109/TSC.2016.2597829.

[40] X. Luo, M. C. Zhou, S. Li, Y. N. Xia, Z. H. You, Q. S. Zhu, and H. Leung, "Incorporation of efficient second-order solvers into latent factor models for accurate prediction of missing QoS data," *IEEE Trans. Cybern.*, vol. 48, no. 4, pp. 1216−1228, Apr. 2018.

[41] X. Luo, M. C. Zhou, H. Leung, Y. N. Xia, Q. S. Zhu, Z. H. You, and S. Li, "An incremental-and-static-combined scheme for matrix-factorization-based collaborative filtering," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 1, pp. 333−343, Jan. 2016.

[42] Z. X. Liu, Y. Z. Yuan, X. P. Guan, and X. B. Li, "An approach of distributed joint optimization for cluster-based wireless sensor networks," *IEEE/CAA J. Autom. Sinica*, vol. 2, no. 3, pp. 267−273, Jul. 2015.

[43] B. X. Zhang and Z. B. Zhu, "Linearized proximal alternating direction method of multipliers for parallel magnetic resonance imaging," *IEEE/-CAA J. Autom. Sinica*, vol. 4, no. 4, pp. 763−769, Oct. 2017.

[44] J. Cheng, M. Chen, M. Zhou, S. Gao, C. Liu, and C. Liu, Overlapping Community Change Point Detection in an Evolving Network, *IEEE Trans. on Big Data*, doi: 10.1109/TBDATA.2018.2880780, Nov. 2018.

[45] S. Gao, M. Zhou, Y. Wang, J. Cheng, H. Yachi, and J. Wang, "Dendritic neuron model with effective learning algorithms for classification, approximation and prediction," *IEEE Transactions on Neural Networks and Learning Systems*, doi: 10.1109/TNNLS.2018.2846646, 2018.

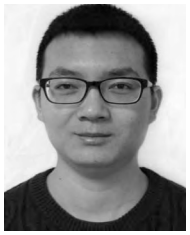
[46] X. Luo, M. C. Zhou, M. S. Shang, S. Li, and Y. N. Xia, "A novel approach to extracting non-negative latent factors from non-negative big sparse matrices," *IEEE Access*, vol. 4, pp. 2649−2655, Apr. 2016.

[47] X. Luo, M. C. Zhou, S. Li, Z. H. You, Y. N. Xia, and Q. S. Zhu, "A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 3, pp. 579−592, Mar. 2016.

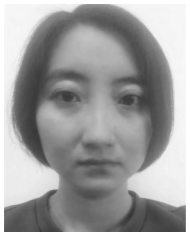
[48] X. Luo, J. P. Sun, Z. D. Wang, S. Li, and M. S. Shang, "Symmetric and nonnegative latent factor models for undirected, high-dimensional, and sparse networks in industrial applications," *IEEE Trans. Ind. Inf.*, vol. 13, no. 6, pp. 3098−3107, Dec. 2017

Mingsheng Shang received the B.E. degree in management in Sichuan Normal University in Chengdu, China in 1995, and Ph.D. degree in computer science from University of Electronic Science and Technology of China in Chengdu, China in 2007. He is currently a Professor at the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China. His research interests include complex network analysis and big data applications.

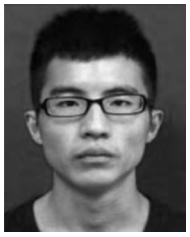
Xin Luo (M'14−SM'17) received the B.S. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2005, and the Ph.D. degree in computer science from Beihang University, Beijing, China, in 2011. In 2016, he joined the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China, where he is currently a Professor of computer science and engineering. His research interests include big data analysis and intelligent control. He has published more than 90 papers (including 30+ IEEE Transactions papers) in his related areas.



Zhigang Liu received the B.S. degree in geographical information system from Chongqing University of Posts and Telecommunications, Chongqing, China, in 2013. He is currently pursuing an M.E. degree in computer technology at Chongqing University, Chongqing, China, and studying at the Chongqing institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China as an exchange scholar. His research interests include big data analysis and algorithm design for large scale data applications.



Jia Chen received the B.S. degree in electronic information engineering from Hunan University, Hunan, China, in 2007. She is a doctoral candidate in the college of computer science and a faculty member in asset management department, Beihang University. Her research interests include big data analysis, social relationship and asset management.



Ye Yuan received the B.S. degree in electronic information engineering and the M.S. degree in signal processing from the University of Electronic Science and technology, Chengdu, China, in 2010 and 2013, respectively. He is currently working toward the Ph.D. degree in computer science from Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China. His research interests include data mining, recommender system, intelligent computing, and their applications.

MengChu Zhou (S'88−M'90−SM'93−F'03) received his B.S. degree in Control Engineering from Nanjing University of Science and Technology, Nanjing, China in 1983, M.S. degree in Automatic Control from Beijing Institute of Technology, Beijing, China in 1986, and Ph. D. degree in Computer and Systems Engineering from Rensselaer Polytechnic Institute, Troy, NY in 1990. He joined New Jersey Institute of Technology (NJIT), Newark, NJ in 1990, and is now a Distinguished Professor of Electrical and Computer Engineering. His research interests are in Petri nets, intelligent automation, Internet of Things, big data, web services, and intelligent transportation. He has over 800 publications including 12 books, 460+ journal papers (360+ in IEEE transactions), 12 patents and 28 book-chapters. He is the founding Editor of IEEE Press Book Series on Systems Science and Engineering and Editor-in-Chief of *IEEE/CAA Journal of Automatica Sinica*. He is a recipient of Humboldt Research Award for US Senior Scientists from Alexander von Humboldt Foundation, Franklin V. Taylor Memorial Award and the Norbert Wiener Award from IEEE Systems, Man and Cybernetics Society. He is a life member of Chinese Association for Science and Technology-USA and served as its President in 1999. He is a Fellow of International Federation of Automatic Control (IFAC), American Association for the Advancement of Science (AAAS), and Chinese Association of Automation (CAA).