Robust Latent Factor Analysis for Precise Representation of High-Dimensional and Sparse Data

Di Wu, *Member, IEEE* and Xin Luo, *Senior Member, IEEE*

***Abstract*—High-dimensional and sparse (HiDS) matrices commonly arise in various industrial applications, e.g., recommender systems (RSs), social networks, and wireless sensor networks. Since they contain rich information, how to accurately represent them is of great significance. A latent factor (LF) model is one of the most popular and successful ways to address this issue. Current LF models mostly adopt L_2-norm-oriented *Loss* to represent an HiDS matrix, i.e., they sum the errors between observed data and predicted ones with L_2-norm. Yet L_2-norm is sensitive to outlier data. Unfortunately, outlier data usually exist in such matrices. For example, an HiDS matrix from RSs commonly contains many outlier ratings due to some heedless/malicious users. To address this issue, this work proposes a smooth L_1-norm-oriented latent factor (SL-LF) model. Its main idea is to adopt smooth L_1-norm rather than L_2-norm to form its *Loss*, making it have both strong robustness and high accuracy in predicting the missing data of an HiDS matrix. Experimental results on eight HiDS matrices generated by industrial applications verify that the proposed SL-LF model not only is robust to the outlier data but also has significantly higher prediction accuracy than state-of-the-art models when they are used to predict the missing data of HiDS matrices.**

***Index Terms*—High-dimensional and sparse matrix, L_1-norm, L_2-norm, latent factor model , recommender system, smooth L_1-norm.**

I. Introduction

In this era of information explosion, people are frequently inundated by big data from various industrial applications [1]–[5], e.g., recommender systems (RSs), social networks, and wireless sensor networks. Among these applications, matrices are commonly adopted to represent the relationship between two types of entities. For example, a user-item rating matrix is frequently seen in recommender systems (RSs) [6]–[9], where each row indicates a specific user, each column indicates a specific item (e.g., movie, electronic product, and music), and each entry indicates a user's preference on an item.

In big data-related applications like Amazon [10], since the relation among numerous entities is unlikely to be fully observed in practice, matrices from these applications are usually high-dimensional and sparse (HiDS) [7], [11]. Yet these HiDS matrices contain rich formation regarding various valuable knowledge, e.g., user's potential preferences on items in RSs. Hence, how to precisely extract useful information from an HiDS matrix becomes a hot yet thorny issue in industrial applications.

Up to now, various approaches and models are proposed to address this issue [6]–[9]. Among them, a latent factor (LF) model, which originates from matrix factorization techniques [11], [12], is becoming increasingly popular due to its high accuracy and scalability in industrial applications [13], [14]. Given an HiDS matrix, an LF model represents it by training two low-dimensional LF matrices based on its observed data only [13], [15]. To model an accurate LF model, how to design its *Loss* function is very crucial, where *Loss* denotes the sum of errors between observed data (ground truths) and predicted ones computed by a specific norm on an HiDS matrix [13], [14].

Currently, most LF models adopt L_2-norm-oriented *Loss* [12]–[18] while few ones adopt L_1-norm-oriented *Loss* [19]. Fig. 1(a) illustrates the differences between L_1-norm-oriented and L_2-norm-oriented *Losses* [19]–[21] where *Error* denotes the errors between predicted results and ground truths: 1) the former is less sensitive to *Error* than the latter, thereby enhancing the robustness of a resultant model [19], [21], [22], and 2) the latter is smoother than the former when the absolute value of *Error* is small (less than 1), thereby enhancing the stability of a resultant model [23].

Hence, although an LF model with L_2-norm-oriented *Loss* can achieve a steady and accurate prediction for the missing data of an HiDS matrix space [23], its robustness cannot be guaranteed when such a matrix is mixed with outlier data. Unfortunately, outlier data usually exist in an HiDS matrix. For example, an HiDS matrix from RSs commonly contains many outlier ratings due to some heedless/malicious users (e.g., a user rates an item randomly in the feedback or badmouths a specific item) [24], [25].

Manuscript received September 4, 2020; revised October 31, 2020; accepted November 17, 2020. This work was supported in part by the National Natural Science Foundation of China (61702475, 61772493, 61902370, 62002337), in part by the Natural Science Foundation of Chongqing, China (cstc2019jcyj-msxmX0578, cstc2019jcyjjqX0013), in part by the Chinese Academy of Sciences "Light of West China" Program, in part by the Pioneer Hundred Talents Program of Chinese Academy of Sciences, and by Technology Innovation and Application Development Project of Chongqing, China (cstc2019jscx-fxydX0027). Recommended by Associate Editor Shangce Gao. *(Corresponding author: Xin Luo.)*

Citation: D. Wu and X. Luo, "Robust latent factor analysis for precise representation of high-dimensional and sparse data," *IEEE/CAA J. Autom. Sinica*, vol. 8, no. 4, pp. 796–805, Apr. 2021.

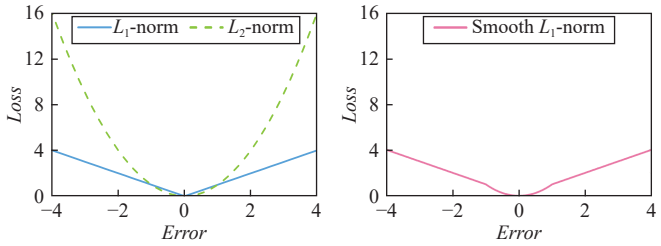
The authors are with the Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, and also with the Chongqing School, University of Chinese Academy of Sciences, Chongqing 400714, China (e-mail: wudi@cigit.ac.cn; luoxin21@cigit.ac.cn).

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.2020.1003533

On the other hand, although an LF model with L_1-norm-oriented *Loss* has intrinsic robustness, its solution space for predicting the missing data of an HiDS matrix is multimodal. The reason is that L_1-norm-oriented *Loss* is not smooth when the predicted results and ground truths are close to each other. As a result, an LF model with L_1-norm-oriented *Loss* may be stacked by some "bad" solutions, making it unable to guarantee high prediction accuracy.

From the aforementioned discussions, we see that both L_1 and L_2-norm-oriented *Losses* are not the best choice for modeling an LF model. Then, do we have an alternative one? This work aims to answer it. Fig. 1(b) illustrates the smooth L_1-norm-oriented *Loss*, where we observe that *Loss* not only is robust to *Error* but also has a smooth gradient when the absolute value of *Error* is smaller than 1. Motivated by this observation, we propose a smooth L_1-norm-oriented latent factor (SL-LF) model. Its main idea is to adopt smooth L_1-norm to form its *Loss*, making it have both strong robustness and high accuracy in predicting the missing data of an HiDS matrix.



(a) L_1-norm and L_2-norm (b) Smooth L_1-norm

Fig. 1. The relationship between *Loss* and *Error* with different norms to solve a regression problem.

Main contributions of this work include:

1) Proposing an SL-LF model with strong robustness and high accuracy in predicting the missing data of an HiDS matrix.

2) Performing a suite of theoretical analyses and algorithm designs for the proposed SL-LF model.

3) Conducting extensive empirical studies on eight HiDS matrices generated by industrial applications to evaluate the proposed model and other state-of-the-art ones.

To the author's best knowledge, this is the first study to employ smooth L_1-norm to implement an LF model for predicting the missing data of an HiDS matrix. Experimental results demonstrate that compared with state-of-the-art models, SL-LF achieves significant accuracy gain when predicting missing data of an HiDS matrix. Its computational efficiency is also highly competitive when compared with the most efficient LF models.

The rest of the paper is organized as follows. Section II states preliminaries. Section III presents an SL-LF model. Section IV reveals experimental results. Section V discusses related work. Finally, Section VI concludes this paper.

II. Preliminaries

In this section, we give the related definitions and descriptions of an HiDS matrix and an LF model. According to [9], [16], [18], they are defined as follows.

Definition 1 (HiDS matrix): Given a set U and a set I, Z denotes a $|U| \times |I|$ matrix where each element $z_{u,i}$ denotes the relation between element u ($u \in U$) and element i ($i \in I$). Z_K and Z_U respectively denote the known and the unknown entities sets of Z. Z is an HiDS matrix with $|Z_K| << |Z_U|$.

Built on an HiDS matrix, an LF model is defined as follows [13], [14], [26].

Definition 2 (Latent factor model): Given Z, f, a $|U| \times f$ latent factor matrix X, and an $|I| \times f$ latent factor matrix Y, a latent factor model is to search for X and Y to achieve Z's rank-f approximation $\widehat{Z}$ by minimizing the errors between Z and $\widehat{Z}$ on Z_K with the condition of $f << \min\{|U|, |I|\}$, where $\widehat{Z}$ is given by $\widehat{Z} = XY^T$. $\widehat{Z}$'s each element $\hat{z}_{u,i}$ is the prediction for each $z_{u,i}$.

To implement an LF model, a *Loss* function needs to be carefully designed to measure the difference between Z and $\widehat{Z}$ [14]. Currently, L_2-norm is most commonly used [7], [17]:

$$\arg\min_{X,Y} \varepsilon(X,Y) = \left\| \Omega \odot \left(Z - \widehat{Z} \right) \right\|_{L_2}^2 = \left\| \Omega \odot \left(Z - XY^T \right) \right\|_{L_2}^2 \quad (1)$$

where $\|\cdot\|_{L_2}$ denotes the L_2-norm of a matrix, $\odot$ denotes the Hadamard product (the component-wise multiplication), and Ω is a $|U| \times |I|$ binary index matrix:

$$\Omega_{u,i} = \begin{cases} 1, & \text{if } z_{u,i} \text{ is known} \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

As analyzed in [14], an LF model is easy to overfit on Z_K of an HiDS matrix by minimizing (1). Tikhonov regularization is thus adopted to prevent such an overfitting problem [9], [16], [18]. Then, with it, an LF model has the following objective function:

$$\arg\min_{X,Y} \varepsilon(X,Y) = \left\| \Omega \odot \left(Z - XY^T \right) \right\|_{L_2}^2 + \lambda \left(\|X\|_{L_2}^2 + \|Y\|_{L_2}^2 \right) \quad (3)$$

where λ is a regularization controlling parameter. Next, we present an SL-LF model.

III. Smooth L_1-Norm-Oriented Latent Factor Model

A. Objective Formulation

As L_2-norm is sensitive to outlier data [20], an LF model with a *Loss* function of (1) lacks robustness. On the other hand, an LF model with an L_1-norm-oriented *Loss* function cannot guarantee a high prediction accuracy because its solution space is not smooth. As shown in Fig. 1(b), smooth L_1-norm has the merits of both robustness and smoothness. Yet it remains open whether an LF model with a smooth L_1-norm-oriented *Loss* function can achieve a highly robust and accurate prediction for the missing data of an HiDS matrix. To answer it, we first define the smooth L_1-norm of a matrix.

Definition 3 (Smooth L_1-norm of a matrix): Given a matrix R with M rows and N columns, symbol $\|R\|_{SL_1}$ denotes the smooth L_1-norm of R.

$$\|R\|_{SL_1} = \sum_{m=1}^{M} \sum_{n=1}^{N} v, \; v = \begin{cases} \sqrt{r_{m,n}^2}, & \text{if } |r_{mn}| \le 1 \\ |r_{mn}|, & \text{if } |r_{mn}| > 1 \end{cases} \quad (4)$$

where $r_{m,n}$ denotes R's element at m-th row and n-th column,

$m\in\{1, 2, \ldots, M\}$, $n\in\{1, 2, \ldots, N\}$.

Then, to implement an SL-LF model, we design the following objective function:

$$\arg\min_{X,Y}\varepsilon(X,Y)=\left\|\Omega\odot\left(Z-XY^T\right)\right\|_{SL_1}^2+\lambda\left(\|X\|_{L_2}^2+\|Y\|_{L_2}^2\right) \quad (5)$$

where $\|\cdot\|_{SL_1}$ denotes the smooth L_1-norm of a matrix. As Z contains numerous unknown data, (5) should be expanded into the following data density-oriented form for high efficiency in both storage and computation [13], [15]:

$$\arg\min_{X,Y}\varepsilon(X,Y)=\begin{cases}\displaystyle\sum_{(u,i)\in Z_K}\left(\Delta_{u,i}\right)^2+\lambda\sum_{(u,i)\in Z_K}\left(\sum_{k=1}^{f}x_{u,k}^2+\sum_{k=1}^{f}y_{i,k}^2\right)\\ \qquad\text{if } \left|\Delta_{u,i}\right|\le 1\\ \displaystyle\sum_{(u,i)\in Z_K}\left|\Delta_{u,i}\right|+\lambda\sum_{(u,i)\in Z_K}\left(\sum_{k=1}^{f}x_{u,k}^2+\sum_{k=1}^{f}y_{i,k}^2\right)\\ \qquad\text{if } \left|\Delta_{u,i}\right|>1\end{cases} \quad (6)$$

where $x_{u,k}$ denotes the specific entity at the u-th row and k-th column in X, $y_{i,k}$ denotes the specific entity at the i-th row and k-th column in Y, and $\Delta_{u,i}$ is the prediction error between $z_{u,i}$ and $\hat{z}_{u,i}$ calculated as

$$\Delta_{u,i}=z_{u,i}-\hat{z}_{u,i}=z_{u,i}-\sum_{k=1}^{f}x_{u,k}y_{i,k}. \quad (7)$$

With such an objective function of (6), we next check whether an SL-LF model can achieve a highly robust and accurate prediction for the missing data of an HiDS matrix.

B. Model Optimization

As analyzed in [13], [15], a stochastic gradient descent (SGD) algorithm has the advantages of fast convergence and ease of implementation in optimizing a bilinear objective. Hence, we employ it to minimize (6) by training desired X and Y. First, we consider the instant loss of (6) on a single entity $z_{u,i}$:

$$\varepsilon_{u,i}=\begin{cases}\left(\Delta_{u,i}\right)^2+\lambda\left(\displaystyle\sum_{k=1}^{f}x_{u,k}^2+\sum_{k=1}^{f}y_{i,k}^2\right), & \text{if } \left|\Delta_{u,i}\right|\le 1\\ \left|\Delta_{u,i}\right|+\lambda\left(\displaystyle\sum_{k=1}^{f}x_{u,k}^2+\sum_{k=1}^{f}y_{i,k}^2\right), & \text{if } \left|\Delta_{u,i}\right|>1.\end{cases} \quad (8)$$

To solve (8) with SGD, we need to respectively move each single LF $x_{u,k}$ and $y_{i,k}$ along the opposite of the stochastic gradient of (8). To do so, we make

$$\begin{cases}x_{u,k}=x_{u,k}-\eta\dfrac{\partial\varepsilon_{u,i}}{\partial x_{u,k}}\\ y_{i,k}=y_{i,k}-\eta\dfrac{\partial\varepsilon_{u,i}}{\partial y_{i,k}}\\ \qquad\forall k\in\{1,2,\ldots,f\}\end{cases} \quad (9)$$

where η is the learning rate. Since (8) has the absolute deviation term of $|\Delta_{u,i}|$, there are a total of three situations for (8) in computing (9) as follows:

$$\varepsilon_{u,i}=\begin{cases}\left(\Delta_{u,i}\right)^2+\lambda\left(\displaystyle\sum_{k=1}^{f}x_{u,k}^2+\sum_{k=1}^{f}y_{i,k}^2\right), & \text{if } \left|\Delta_{u,i}\right|\le 1\\ \Delta_{u,i}+\lambda\left(\displaystyle\sum_{k=1}^{f}x_{u,k}^2+\sum_{k=1}^{f}y_{i,k}^2\right), & \text{if } \Delta_{u,i}>1\\ -\Delta_{u,i}+\lambda\left(\displaystyle\sum_{k=1}^{f}x_{u,k}^2+\sum_{k=1}^{f}y_{i,k}^2\right), & \text{if } \Delta_{u,i}<-1.\end{cases} \quad (10)$$

Then, by incorporating (10) into (9), we have the training rules for each single LF $x_{u,k}$ and $y_{i,k}$ respectively on a single entity $z_{u,i}$ as follows:

For $z_{u,i}$ $\forall k\in\{1,2,\ldots,f\}$:

$$\begin{cases}\begin{cases}x_{u,k}=x_{u,k}+\eta y_{i,k}\Delta_{u,i}-\eta\lambda x_{u,k},\\ y_{i,k}=y_{i,k}+\eta x_{u,k}\Delta_{u,i}-\eta\lambda y_{i,k},\end{cases} & \text{if } \left|\Delta_{u,i}\right|\le 1\\ \begin{cases}x_{u,k}=x_{u,k}+\eta y_{i,k}-\eta\lambda x_{u,k},\\ y_{i,k}=y_{i,k}+\eta x_{u,k}-\eta\lambda y_{i,k},\end{cases} & \text{if } \Delta_{u,i}>1\\ \begin{cases}x_{u,k}=x_{u,k}-\eta y_{i,k}-\eta\lambda x_{u,k},\\ y_{i,k}=y_{i,k}-\eta x_{u,k}-\eta\lambda y_{i,k},\end{cases} & \text{if } \Delta_{u,i}<-1.\end{cases} \quad (11)$$

After all the known entities in Z_K are employed to train by (11), we get the desired X and Y. Finally, Z's rank-f approximation $\widehat{Z}$, which can predict the missing data of Z, is obtained as $\widehat{Z}=XY^T$.

C. Incorporating Linear Biases into SL-LF

According to [14], [27], [28], linear biases can be incorporated into an LF model to improve its prediction accuracy. Since SL-LF is also an LF-based model, we can extend it with linear biases. As analyzed in [14], [27], [28], linear biases commonly include the global average and the observed deviations on users and items. With them, then, an SL-LF model approximates $\hat{z}_{u,i}$ as follows:

$$\hat{z}_{u,i}=\mu+b_u+b_i+\sum_{k=1}^{f}x_{u,k}y_{i,k} \quad (12)$$

where μ denotes the global average value in Z_K, b_u denotes the observed deviations on user u, and b_i denotes the observed deviations on item i, respectively. Given an HiDS matrix Z, μ is computed as follows:

$$\mu=\frac{\displaystyle\sum_{(u,i)\in Z_K}r_{u,i}}{|Z_K|}. \quad (13)$$

After that, b_u and b_i can be estimated in an easy way [27], [28]:

$$b_i=\frac{\displaystyle\sum_{(u,i)\in Z(i)}(r_{u,i}-\mu)}{\theta_1+|Z(i)|},\ b_u=\frac{\displaystyle\sum_{(u,i)\in Z(u)}(r_{u,i}-\mu-b_i)}{\theta_2+|Z(u)|} \quad (14)$$

where $Z(i)$ denotes the known entities set in the i-th column of Z, $Z(u)$ denotes the known entities set in the u-th row of Z, θ_1

and θ_2 denote the threshold constant and can be determined by cross-validation [28]. Then, by incorporating linear biases into SL-LF, (7) is changed into:

$$\Delta_{u,i} = z_{u,i} - \hat{z}_{u,i} = z_{u,i} - \mu - b_u - b_i - \sum_{k=1}^{f} x_{u,k} y_{i,k}. \quad (15)$$

Finally, by combining (15) into (11), we can obtain the training rules for SL-LF with linear biases on a single entity $z_{u,i}$.

D. Algorithm Design and Analysis

From the above analyses, we see that SL-LF has two versions, i.e., without and with linear biases. We respectively name SL-LF without and with linear biases as SL-LF$_{\bar{b}}$ and SL-LF$_b$. First, we design Algorithm 1 for SL-LF$_{\bar{b}}$, whose computational complexity $C_{\bar{b}}$ can be derived as follows:

$$\begin{aligned} C_{\bar{b}} &= \Theta(1) + \Theta(|U| \times f) + \Theta(|I| \times f) \\ &\quad + N_{mtr} \times (|Z_K| \times f \times 2 \times \Theta(1) + \Theta(1)) \\ &\approx \Theta(N_{mtr} \times |Z_K| \times f) \end{aligned} \quad (16)$$

where we drop the lower order terms due to $(|U| + |I|) << |Z_K|$ in real-world applications. Second, we design Algorithm 2 for SL-LF$_b$, whose computational complexity C_b can be derived as follows:

$$\begin{aligned} C_b &= \Theta(1) + \Theta(|U| \times f) + \Theta(|I| \times f) + \Theta(|Z_K|) + \Theta(|Z_K|) \\ &\quad + \Theta(|Z_K|) + N_{mtr} \times (|Z_K| \times f \times 2 \times \Theta(1) + \Theta(1)) \\ &\approx \Theta(N_{mtr} \times |Z_K| \times f). \end{aligned} \quad (17)$$

From (16) and (17), we see that SL-LF$_{\bar{b}}$ and SL-LF$_b$ have the same computational complexity. The maximal-training-round count N_{mtr}, known entity count $|Z_K|$, and LF space dimension f are crucial in deciding their computational complexity.

Besides, to implement SL-LF$_b$, we need the following data structures: 1) two arrays with length $|Z_K|$ to cache training data and corresponding predictions, 2) an array with length $|U|$ to cache observed deviations on users, 3) an array with length $|I|$ to cache observed deviations on items, 4) a matrix with size $|U| \times f$ to cache X, and 5) a matrix with size $|I| \times f$ to cache Y. Thus, SL-LF$_b$'s space complexity is

$$\begin{aligned} S_B &= 2|Z_K| + |U| + |I| + |U| \times f + |I| \times f \\ &\approx \Theta\left(f \times \max\left\{\frac{|Z_K|}{f}, |U|, |I|\right\}\right). \end{aligned} \quad (18)$$

Although SL-LF$_{\bar{b}}$ does not need arrays to cache linear biases, it has the same space complexity as SL-LF$_b$.

In real-world applications, both N_{mtr} and f are positive constants. Hence, SL-LF's computational and space complexity is linear with $|Z_K|$, which is highly efficient and applicable to big data-related industrial applications.

IV. Experiments and Results

A. General Settings

Datasets: Eight benchmark datasets are selected to conduct the experiments. Table I summarizes their properties. They are real HiDS datasets generated by industrial applications and frequently adopted by prior studies [13], [29]. Dating is collected by an online dating website LibimSeTi [30], Douban is collected by Douban.com [13], [29], Eachmovie is collected by the EachMovie system by the DEC Systems Research Center [1], [31], Epinion is collected by Trustlet website [29], Flixter is collected by the Flixter website [32], Jester is collected by the joke-recommender Jester [32], and MovieLens_10M and MovieLens_20M are collected by the MovieLens system [33].

Algorithm 1 SL-LF$_{\bar{b}}$

Input: Z_K	
Operation	**Cost**
initializing f, λ, η, $Nmtr$ (*Max-training-round count*)	$\Theta(1)$
initializing X randomly	$\Theta(\|U\| \times f)$
initializing Y randomly	$\Theta(\|I\| \times f)$
while $t \le Nmtr$ && not converge	$\times Nmtr$
for each entity $z_{u,i}$ in Z_K	$\times \|Z_K\|$
for k=1 to f	$\times f$
computing $x_{u,k}$ according to (7) and (11)	$\Theta(1)$
computing $y_{i,k}$ according to (7) and (11)	$\Theta(1)$
end for	–
end for	–
t=t+1	$\Theta(1)$
end while	–
Output: X, Y	

Algorithm 2 SL-LF$_b$

Input: Z_K	
Operation	**Cost**
initializing f, λ, η, $Nmtr$ (*Max-training-round count*)	$\Theta(1)$
initializing X randomly	$\Theta(\|U\| \times f)$
initializing Y randomly	$\Theta(\|I\| \times f)$
computing μ according to (13)	$\Theta(\|Z_K\|)$
for each entity $z_{u,i}$ in Z_K	$\times \|Z_K\|$
computing b_i according to (14)	$\Theta(1)$
end for	–
for each entity $z_{u,i}$ in Z_K	$\times \|Z_K\|$
computing b_u according to (14)	$\Theta(1)$
end for	–
while $t \le Nmtr$ && not converge	$\times Nmtr$
for each entity $z_{u,i}$ in Z_K	$\times \|Z_K\|$
for k=1 to f	$\times f$
computing $x_{u,k}$ according to (11) and (15)	$\Theta(1)$
computing $y_{i,k}$ according to (11) and (15)	$\Theta(1)$
end for	–
end for	–
t=t+1	$\Theta(1)$
end while	–
Output: X, Y	

Evaluation Metrics: Missing data prediction is a common but important task in representing an HiDS matrix [34]. To evaluate prediction accuracy, mean absolute error (MAE) and root mean squared error (RMSE) are widely adopted:

$$MAE = \left(\sum_{(u,i) \in \Gamma} \left| z_{u,i} - \hat{z}_{u,i} \right|_{abs} \right) / |\Gamma|$$

TABLE I
PROPERTIES OF ALL THE DATASETS

No.	Name	$\|U\|$	$\|I\|$	$\|Z_K\|$	Density
D1	Dating	135 359	168 791	17 359 346	0.08%
D2	Douban	129 490	58 541	16 830 839	0.22%
D3	Eachmovie	72 916	1 628	2 811 718	2.37%
D4	Epinion	755 760	120 492	13 668 321	0.02%
D5	Flixter	147 612	48 794	8 196 077	0.11%
D6	Jester	24 983	100	1 186 324	47.49%
D7	MovieLens_10M	71 567	65 133	10 000 054	0.21%
D8	MovieLens_20M	138 493	26 744	20 000 263	0.54%

$$RMSE = \sqrt{\left(\sum_{(u,i)\in\Gamma}(z_{u,i}-\hat{z}_{u,i})^2\right)/|\Gamma|}$$

where Γ denotes the testing set and $|\cdot|_{abs}$ denotes the absolute value of a given number. Lower MAE and RMSE indicate higher missing data prediction accuracy. Besides, to evaluate the computational efficiency of missing data prediction, we measure CPU running time.

Experimental Designs: For each dataset, its 80% known data are used as a training dataset and the remaining 20% ones as a testing dataset. Five-fold cross-validations are adopted. All the experiments are run on a PC with 3.4 GHz i7 CPU and 64 GB RAM. In the next experiments, we aim at answering the following research questions:

1) How do the hyper-parameters of an SL-LF model impact its prediction performance?

2) How do the outlier data impact the prediction performance of an SL-LF model?

3) Does the proposed SL-LF model outperform related state-of-the-art models?

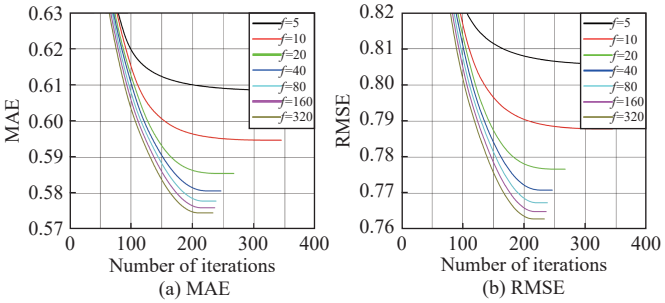
B. Hyper-Parameter Sensitivity Tests

From Section III, we know that SL-LF has three parameters, i.e., LF space dimension f, regularization parameter λ, and learning rate η. Next, we analyze the behaviors of SL-LF$_{\bar{b}}$ (without linear biases) and SL-LF$_b$ (with linear biases) with respect to these parameters.

1) Impacts Off

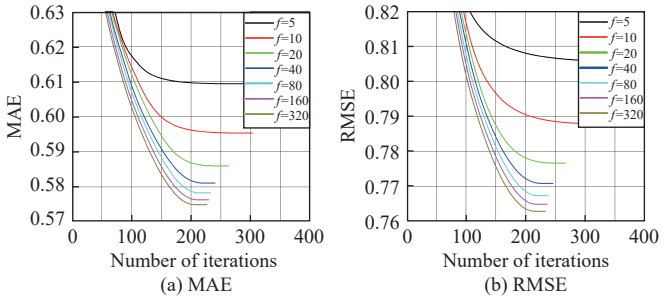
This set of experiments increase f from 5 to 320. Figs. 2 and 3 record the results on D8. The complete results on all the datasets are recorded in the Supplementary File[1] of this paper. From these results, we find that the larger f makes both SL-LF$_{\bar{b}}$ and SL-LF$_b$ have better representation learning ability on most cases, which benefits for achieving a higher prediction accuracy. However, such prediction accuracy gain slows down when f is larger than a threshold, such as 20. Moreover, we find exceptions that prediction accuracy decreases when f increases from 5 to 40 on D4. One reason may be that underfitting is caused by such hyper-parameters. Besides, the larger f requires more computational cost as analyzed in (16) and (17). Hence, f should be set appropriately to balance prediction accuracy and computational cost according to a specific task. As a rule of thumb, f is usually set to fall into 10–20 [14], [15], [35].

[1]https://pan.baidu.com/s/1o_8sKP0HRluNH1a4IWHW8w, Code: t3sw



(a) MAE (b) RMSE

Fig. 2. The training process of SL-LF$_{\bar{b}}$ with different f on D8, where λ = 0.01 and η = 0.001.

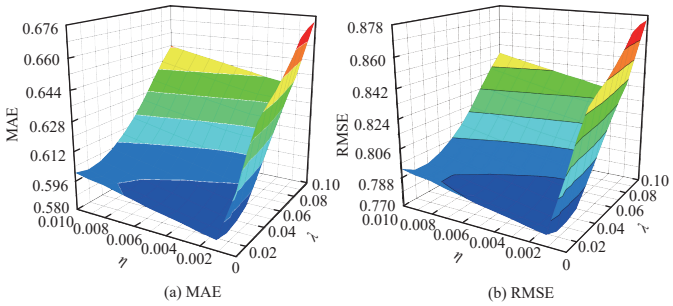


(a) MAE (b) RMSE

Fig. 3. The training process of SL-LF$_b$ with different f on D8, where λ = 0.01 and η = 0.001.

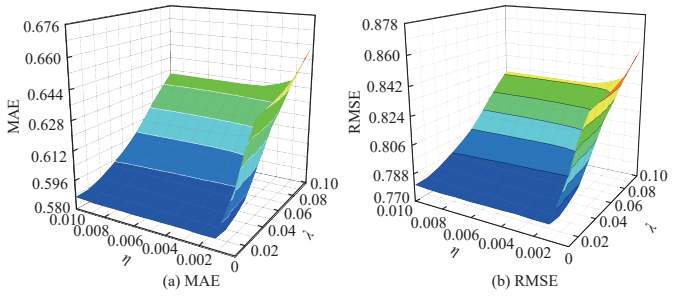
2) Impacts of λ and η

In this set of experiments, we increase λ from 0.01 to 0.1 and η from 0.0001 to 0.01 by performing a grid-based search [36]. Figs. 4 and 5 show the results on D8. The complete results on all the datasets are presented in the Supplementary File[1]. From them, we conclude that:



(a) MAE (b) RMSE

Fig. 4. The experimental results of SL-LF$_{\bar{b}}$ with respect to λ and η on D8, where f = 20.



(a) MAE (b) RMSE

Fig. 5. The experimental results of SL-LF$_b$ with respect to λ and η on D8, where f = 20.

TABLE II
DESCRIPTIONS OF ALL THE COMPARISON MODELS

Model	Description
BLF	The basic LF model proposed in 2009 [14]. It has been extensively used in RSs.
NLF	The regularized non-negative LF model proposed in 2016 [15]. It improves the L2-LF model by introducing the non-negative constraint into objective function design.
FNLF	A fast non-negative LF model based on generalized momentum proposed in 2018 [7]. It improves the NLF model.
AutoRec	A DNN-based model proposed in 2015 [37]. It is an autoencoder [38] framework for CF. It is a representative model in DNN-based RSs.
DCCR	A DNN-based model proposed in 2019 [39]. It improves AutoRec by using two different neural networks.
SL-LF$_{\bar{b}}$	The proposed SL-LF model without linear biases.
SL-LF$_{b}$	The proposed SL-LF model with linear biases.

a) Both λ and η have a significant impact on prediction accuracy of both SL-LF$_{\bar{b}}$ and SL-LF$_{b}$. As λ and η increase, MAE and RMSE decrease at first and then increase in general. For example, on D8, RMSE of SL-LF$_{\bar{b}}$ decreases from 0.8116 to 0.7729 at the beginning. Then, it increases up to 0.8451 as λ and η continue to increase.

b) λ and η have different situations in searching their optimal values on the tested datasets. On the different datasets, the optimal value of η is a small value like 0.001. However, the optimal value of λ is different for the different datasets. It distributes in the range from 0.02 to 0.09. Hence, λ should be carefully tuned on the target dataset.

C. Outlier Data Sensitivity Tests

In this section, we compare an SL-LF model with a basic LF (BLF) model when outlier data are added to the datasets. BLF is modeled based on the L_2-norm-oriented *Loss* while SL-LF is done by the smooth L_1-norm-oriented *Loss*. The specific method of adding outlier data is: 1) randomly selecting an unknown entity between two known entities as the outlier entity for the input HiDS matrix Z, 2) assigning a value (maximum or minimum known value) to the outlier entity, 3) the percentage that outlier entities account for known entities is increased from 0% to 100% with an interval of 10%, and 4) the outlier entities are only added into the training set. To illustrate this method, an example is given in Fig. 6.

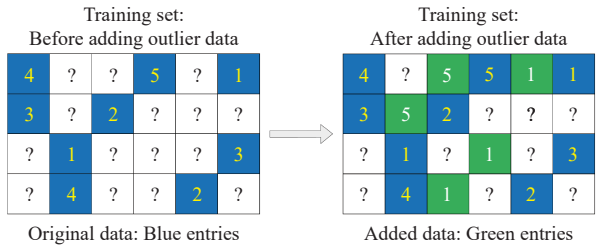


Fig. 6. An example of adding outlier data.

Fig. 7 records the experimental results on D8. Supplementary File[1] records the complete results on all the datasets. Since smooth L_1-norm is less sensitive to outlier data than L_2-norm, we find that both SL-LF$_{\bar{b}}$ and SL-LF$_{b}$ become much more robust than BLF as the percentage of outlier data increases. For example, on D8, RMSEs of SL-LF$_{\bar{b}}$ and BLF are 0.7767 and 0.7761, respectively when there are no outlier data, and then become 0.8536 and 1.1244, respectively when the percentage of outlier data is 100%. The improvement of RMSE of BLF is 0.3483, which is about 4.53 times as large as that of SL-LF$_{\bar{b}}$ at 0.0769. Therefore, we conclude that an SL-LF model is robust to the outlier data.

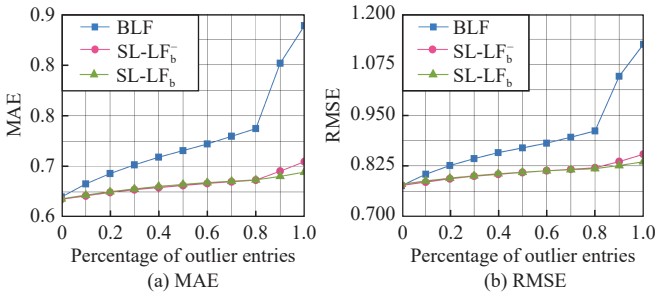


Fig. 7. The outlier data sensitivity tests results of BLF, SL-LF$_{\bar{b}}$, and SL-LF$_{b}$ on D8, where $\lambda = 0.01$, $\eta = 0.001$, and $f = 20$.

D. Comparison Between SL-LF and State-of-the-Art Models

We compare an SL-LF model with five related state-of-the-art models, including three LF-based models (basic latent factor (BLF), non-negative latent factor (NLF), and fast non-negative latent factor (FNLF)) and two deep neural network (DNN)-based models (AutoRec and deep collaborative conjunctive recommender (DCCR)). Table II gives a brief introduction to these models. To make a fair comparison, f is set as 20 for all the LF-based models and the proposed SL-LF model. Besides, we tune the other hyper-parameters for all the involved models to make them achieve their highest prediction accuracy.

1) Comparison of Prediction Accuracy

Table III presents the detailed comparison results. Statistical analysis is conducted on these comparison results. First, the win/loss counts of SL-LF$_{\bar{b}}$/SL-LF$_{b}$ versus other models are summarized in the third/second-to-last row of Table III. Second, we perform Friedman test [40] on these comparison results. The result is recorded in the last row of Table III, where it accepts the hypothesis that these comparison models have significant differences with a significance level of 0.05. From these comparisons and statistical results, we find that a) both SL-LF$_{\bar{b}}$ and SL-LF$_{b}$ achieve lower RMSE/MAE than the other models on most testing cases, and b) SL-LF$_{b}$ achieves the lowest F-rank value among all the models. Hence, we conclude that SL-LF$_{b}$ has the highest prediction accuracy

TABLE III
THE COMPARISON RESULTS ON PREDICTION ACCURACY, INCLUDING WIN/LOSS COUNTS STATISTIC AND FRIEDMAN TEST, WHERE ● AND ✪ RESPECTIVELY INDICATE THAT SL-LF$_{\bar{b}}$, AND SL-LF$_b$ HAVE A HIGHER PREDICTION ACCURACY THAN COMPARISON MODELS

Dataset	Metric	BLF	NLF	FNLF	AutoRec	DCCR	SL-LF$_{\bar{b}}$	SL-LF$_b$
D1	MAE	1.2392●✪	1.2617●✪	1.2588●✪	1.2610●✪	1.2574●✪	1.1702	1.1686
	RMSE	1.8066✪	1.8245✪	1.8215✪	1.8027✪	1.8013✪	1.8275	1.7829
D2	MAE	0.5537●✪	0.5590●✪	0.5592●✪	0.5606●✪	0.5581●✪	0.5516	0.5458
	RMSE	0.7139●✪	0.7150●✪	0.7139●✪	0.7080✪	0.7074✪	0.7094	0.6957
D3	MAE	0.1732●✪	0.1767●✪	0.1763●✪	0.1784●✪	0.1775●✪	0.1731	0.1729
	RMSE	0.2251●	0.2264●✪	0.2259●	0.2305●✪	0.2289●✪	0.2242	0.2260
D4	MAE	0.3011●✪	0.3047●✪	0.3036●✪	0.3014●✪	0.3036●✪	0.2967	0.2766
	RMSE	0.5958✪	0.5994●✪	0.5977✪	0.5946✪	0.5952✪	0.5992	0.4812
D5	MAE	0.6447●✪	0.6550●✪	0.6520●✪	0.6295✪	0.6308✪	0.6348	0.6084
	RMSE	0.8961●✪	0.9056●✪	0.9038●✪	0.8682✪	0.8792✪	0.8949	0.8324
D6	MAE	0.7664●✪	0.7769●✪	0.7778●✪	0.7905●✪	0.7883●✪	0.7552	0.7573
	RMSE	0.9957✪	1.0049●✪	1.0003●✪	1.0078●✪	1.0042●✪	0.9988	0.9936
D7	MAE	0.5999●✪	0.6080●✪	0.6068●✪	0.6048●✪	0.6002●✪	0.5950	0.5980
	RMSE	0.7819●	0.7893●✪	0.7881●	0.7865●	0.7847●	0.7806	0.7887
D8	MAE	0.5886●✪	0.5977●✪	0.5961●✪	0.5947●✪	0.5902●✪	0.5841	0.5857
	RMSE	0.7737●	0.7819●✪	0.7798●✪	0.7802●✪	0.7789●	0.7730	0.7790
Statistical analysis	● Win/Loss	13/3	15/1	14/2	11/5	11/5	–	–
	✪ Win/Loss	13/3	16/0	14/2	15/1	14/2	–	–
	F-rank*	3.281	6.313	5.125	4.813	3.969	**2.625**	**1.875**

* The smaller F-rank value denotes a higher prediction accuracy.

among all the models.

Next, we check whether SL-LF$_b$ achieves significantly higher prediction accuracy than each single model. To do so, we conduct the Wilcoxon signed-ranks test [41], [42] on the comparison results of Table III. Wilcoxon signed-ranks test is a nonparametric pairwise comparison procedure and has three indicators – $R+$, $R-$, and p-value. The larger $R+$ value indicates higher performance and the p-value indicates the significance level. Table IV records the test results, where we see that SL-LF$_b$ has a significantly higher prediction accuracy than all the comparison models with a significance level of 0.05 except for SL-LF$_{\bar{b}}$. However, SL-LF$_b$ achieves a much larger $R+$ value than SL-LF$_{\bar{b}}$, which verifies that linear biases can boost an SL-LF model's prediction accuracy.

2) Comparison of Computational Efficiency

To compare the computational efficiency of all the tested models, we measure their CPU running times on all the datasets. Fig. 8 presents the results. From it, we observe that:

a) DNN-based models (AutoRec and DCCR) cost much more CPU running time than the other models due to their time-consuming DNN-based learning strategy [43].

b) SL-LF costs slightly more CPU running time than BLF. The reason is that SL-LF has the additional computational procedures of discrimination (11) while BLF does not.

c) SL-LF costs less or more CPU running time than NLF and FNLF on the different datasets.

Therefore, these results verify that SL-LF's computational efficiency is higher than those of DNN-based models and comparable to those of other LF-based models.

TABLE IV
STATISTICAL RESULTS ON TABLE III BY CONDUCTING THE WILCOXON SIGNED-RANKS TEST

Comparison	$R+$	$R-$	p-Value
SL-LF$_b$ vs. BLF	121	15	**0.0033**
SL-LF$_b$ vs. NLF	136	0	**0.0002**
SL-LF$_b$ vs. FNL	133	3	**0.0004**
SL-LF$_b$ vs. AutoRec	134	2	**0.0004**
SL-LF$_b$ vs. DCCR	131	5	**0.0006**
SL-LF$_b$ vs. SL-LF$_{\bar{b}}$	100	37	0.0544

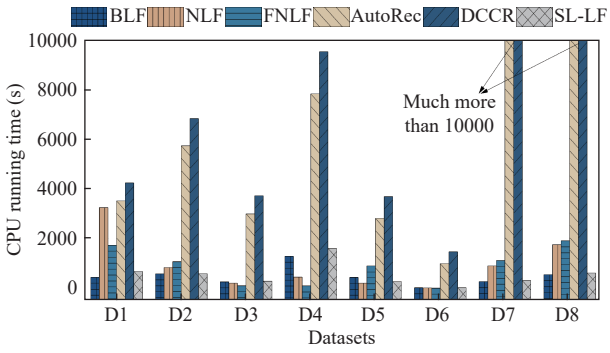


Fig. 8. The comparison CPU running time of involved models on D1–D8.

E. Summary of Experiments

Based on the above experimental results and analyses, we have the following conclusions:

1) An SL-LF model's prediction accuracy is closely connected with λ and η. As a rule of thumb, we can set η=0.001 while λ should be fine-tuned according to a specific target dataset.

2) SL-LF has significantly higher prediction accuracy than state-of-the-art models for the missing data of an HiDS matrix.

3) SL-LF's computational efficiency is much higher than those of DNN-based models and comparable to those of most efficient LF-based models.

4) Linear biases have positive effects on improving SL-LF's prediction accuracy.

V. Related Work

An LF model is one of the most popular and successful ways to efficiently predict the missing data of an HiDS matrix [13], [14]. Up to now, various approaches are proposed to implement an LF model, including a bias-based one [14], non-negativity-constrained one [15], randomized one [17], probabilistic one [44], dual-regularization-based one [45], posterior-neighborhood-regularized one [16], graph regularized one [18], neighborhood-and-location integrated one [6], data characteristic-aware one [35], confidence-driven one [46], deep-latent-factor based one [47], and nonparametric one [48]. Although they are different from one another in terms of model design or learning algorithms, they all adopt an L_2-norm-oriented *Loss*, making them sensitive to outlier data [20]. Since outlier data are frequently found in an HiDS matrix [24], [25], their robustness cannot be guaranteed.

To make an LF model less sensitive to outlier data, Zhu *et al.* [19] proposed to adopt an L_1-norm-oriented *Loss*. However, such an LF model is multimodal because L_1 norm is less smooth than L_2 norm, as shown in Fig. 1(a). Hence, an LF model with an L_1 norm-oriented *Loss* tends to be stacked by some "bad" solutions, resulting in its failure to achieve high prediction accuracy. Differently from these approaches, the proposed SL-LF model adopts smooth L_1-norm-oriented *Loss*, making its solution space smoother and less multimodal than that of an LF model with L_1 norm-oriented *Loss*. Meanwhile, its robustness is also higher than that of an LF model with L_2 norm-oriented *Loss*.

Recently, DNN-based approaches to represent an HiDS matrix have attracted extensive attention [49]. According to a recent review regarding DNN-based studies [34], various models are proposed to address the task of missing data prediction for an HiDS matrix. Representative models include an autoencoder-based model [37], hybrid autoencoder-based model [39], multitask learning framework [50], neural factorization machine [51], attentional factorization machine [52], deep cooperative neural network [53], and convolutional matrix factorization model [54]. However, DNN-based models have the limit of high computational cost caused by their learning strategies. For example, they take complete data rather than known data of an HiDS matrix as input. Unfortunately, an HiDS matrix generated by RSs commonly has a very low rating density. In comparison, SL-LF trains only on the known data of an HiDS matrix, thereby achieving highly computational efficiency.

As analyzed in [13], [16], an LF model can not only predict the missing data of an HiDS matrix but also be used as a data representation approach. Hence, SL-LF has some potential applications in representation learning, such as community detection, autonomous vehicles [5], and medical image analysis [55]–[57]. Besides, some researchers incorporate non-negative constraints into an LF model to improve its performance [17]. Similarly, we plan to improve SL-LF by considering non-negative constraints [58] in the future.

VI. Conclusions

This study proposes, for the first time, a smooth L_1-norm-oriented latent factor (SL-LF) model to robustly and accurately predict the missing data of a high-dimensional and sparse (HiDS) matrix. Its main idea is to employ smooth L_1-norm rather than L_2-norm to form its *Loss* (the error between observed data and predicted ones), making it achieve highly robust and accurate prediction of missing data in a matrix. Extensive experiments on eight HiDS matrices from industrial applications are conducted to evaluate the proposed model. The experimental results verify that 1) it is robust to the outlier data, 2) it significantly outperforms state-of-the-art models in terms of prediction accuracy for the missing data of an HiDS matrix, and 3) its computational efficiency is much higher than those of DNN-based models and comparable to those of most efficient LF models. Although it has shown promising prospects, how to make its hyper-parameter λ self-adaptive and improve its performance by considering non-negative constraint remains open. We plan to fully investigate these issues in the future.

References

[1] R. Q. Lu, X. L. Jin, S. M. Zhang, M. K. Qiu, and X. D. Wu, "A study on big knowledge and its engineering issues," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 9, pp. 1630–1644, Sep. 2019.

[2] S. C. Gao, M. C. Zhou, Y. R. Wang, J. J. Cheng, H. Yachi, and J. H. Wang, "Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 2, pp. 601–614, Feb. 2019.

[3] D. P. Bertsekas, "Feature-based aggregation and deep reinforcement learning: A survey and some new implementations," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 1, pp. 1–31, Jan. 2019.

[4] H. Zahid, T. Mahmood, A. Morshed, and T. Sellis, "Big data analytics in telecommunications: Literature review and architecture recommendations," *IEEE/CAA J. Autom. Sinica*, vol. 7, no. 1, pp. 18–38, Jan. 2020.

[5] Y. F. Ma, Z. Y. Wang, H. Yang, and L. Yang, "Artificial intelligence applications in the development of autonomous vehicles: A survey," *IEEE/CAA J. Autom. Sinica*, vol. 7, no. 2, pp. 315–329, Mar. 2020.

[6] D. Ryu, K. Lee, and J. Baik, "Location-based web service QoS prediction via preference propagation to address cold start problem," *IEEE Trans. Serv. Comput.*, to be published. DOI: 10.1109/TSC.2018.2821686

[7] X. Luo, Z. G. Liu, S. Li, M. S. Shang, and Z. D. Wang, "A fast non-negative latent factor model based on generalized momentum method," *IEEE Trans. Syst., Man, Cybern.: Syst.*, to be published. DOI: 10.1109/TSMC.2018.2875452

[8] J. D. Zhang, C. Y. Chow, and J. Xu, "Enabling kernel-based attribute-aware matrix factorization for rating prediction," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 4, pp. 798–812, Apr. 2017.

[9] J. Castro, J. Lu, G. G. Zhang, Y. C. Dong, and L. Martínez, "Opinion

dynamics-based group recommender systems," *IEEE Trans. Syst.*, *Man*, *Cybern.*: *Syst.*, vol. 48, no. 12, pp. 2394–2406, Dec. 2018.

[10] B. Smith and G. Linden, "Two decades of recommender systems at Amazon.com," *IEEE Int. Comput.*, vol. 21, no. 3, pp. 12–18, May-Jun. 2017.

[11] M. G. Gong, X. M. Jiang, H. Li, and K. C. Tan, "Multiobjective sparse non-negative matrix factorization," *IEEE Trans. Cybern.*, vol. 49, no. 8, pp. 2941–2954, Aug. 2019.

[12] X. N. He, J. H. Tang, X. Y. Du, R. C. Hong, T. W. Ren, and T. S. Chua, "Fast matrix factorization with nonuniform weights on missing data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 8, pp. 2791–2804, Aug. 2020.

[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. Inform.*, vol. 10, no. 2, pp. 1273–1284, May 2014.

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

[15] 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.

[16] D. Wu, Q. He, X. Luo, M. S. Shang, Y. He, and G. Y. Wang, "A Posterior-neighborhood-regularized latent factor model for highly accurate web service QoS prediction," *IEEE Trans. Serv. Comput.*, to be published. DOI: 10.1109/TSC.2019.2961895

[17] 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.

[18] C. C. Leng, H. Zhang, G. R. Cai, I. Cheng, and A. Basu, "Graph regularized *Lp* smooth non-negative matrix factorization for data representation," *IEEE/CAA J. Automa. Sinica*, vol. 6, no. 2, pp. 584–595, Mar. 2019.

[19] X. K. Zhu, X. Y. Jing, D. Wu, Z. Y. He, J. C. Cao, D. Yue, and L. N. Wang, "Similarity-maintaining privacy preservation and Location-aware Low-rank matrix factorization for QoS prediction based web service recommendation," *IEEE Trans. Serv. Comput.*, to be published. DOI: 10.1109/TSC.2018.2839741

[20] L. Wang, M. D. Gordon, and J. Zhu, "Regularized least absolute deviations regression and an efficient algorithm for parameter tuning," in *Proc. 6th IEEE Int. Conf. Data Mining*, Hong Kong, China, 2006, pp. 690–700.

[21] W. T. Ma, N. Li, Y. H. Li, J. D. Duan, and B. D. Chen, "Sparse normalized least mean absolute deviation algorithm based on unbiasedness criterion for system Identification with noisy input," *IEEE Access*, vol. 6, pp. 14379–14388, Feb. 2018.

[22] Q. F. Ke and T. Kanade, "Robust subspace computation using *L*1 norm: School of computer science," Carnegie Mellon University, Pittsburgh, PA, CMU-CS-03-172, Aug. 2003.

[23] R. Koenker and K. F. Hallock, "Quantile regression," *J. Econom. Perspect.*, vol. 15, no. 4, pp. 143–156, 2001.

[24] C. Wu, W. W. Qiu, Z. B. Zheng, X. Y. Wang, and X. H. Yang, "Qos prediction of web services based on two-phase K-means clustering," in *Proc. IEEE Int. Conf. Web Services*, New York, USA, 2015, pp. 161–168.

[25] B. Lakshminarayanan, G. Bouchard, and C. Archambeau, "Robust Bayesian matrix factorisation," in *Proc. the 14th Int. Conf. Artificial Intelligence and Statistics*, Ft. Lauderdale, USA, 2011, pp. 425–433.

[26] D. Wu, Y. He, X. Luo, M. S. Shang, and X. D. Wu, "Online feature selection with capricious streaming features: A general framework," in *Proc. IEEE Int. Conf. Big Data*, Los Angeles, USA, 2019, pp. 683–688.

[27] Y. Yuan, X. Luo, and M. S. Shang, "Effects of preprocessing and training biases in latent factor models for recommender systems," *Neurocomputing*, vol. 275, pp. 2019–2030, Jan. 2018.

[28] Y. Koren and R. Bell, "Advances in collaborative filtering," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, P. B. Kantor, Eds. Boston, USA: Springer, 2015, pp. 77–118.

[29] P. Massa and P. Avesani, "Trust-aware recommender systems," in *Proc. ACM Conf. Recommender Systems*, Minneapolis, USA, 2007, pp. 17–24.

[30] L. Brozovsky and V. Petricek, "Recommender system for online dating service," arXiv: cs/0703042, 2007.

[31] Y. Shi, M. Larson, and A. Hanjalic, "Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges," *ACM Comput. Surv.*, vol. 47, no. 1, pp. 3, May 2014.

[32] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *Inform. Retrieval*, vol. 4, no. 2, pp. 133–151, Jul. 2001.

[33] 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.

[34] S. Zhang, L. N. Yao, A. X. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 5, Feb. 2019.

[35] D. Wu, X. Luo, M. S. Shang, Y. He, G. Y. Wang, and X. D. Wu, "A data-characteristic-aware latent factor model for web services QoS prediction," *IEEE Trans. Knowl. Data Eng.*, to be published. DOI: 10.1109/TKDE.2020.3014302

[36] P. Y. Zhang, S. Shu, and M. C. Zhou, "An online fault detection model and strategies based on SVM-Grid in clouds," *IEEE/CAA J. Autom. Sinica*, vol. 5, no. 2, pp. 445–456, Mar. 2018.

[37] S. Sedhain, A. K. Menon, S. Sanner, and L. X. Xie, "AutoRec: Autoencoders meet collaborative filtering," in *Proc. the 24th Int. Conf. World Wide Web*, Florence, Italy, 2015, pp. 111–112.

[38] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, Article no. 7553, pp. 436−444, May 2015. DOI: 10.1038/nature 14539.

[39] Q. X. Wang, B. B. Peng, X. Y. Shi, T. Q. Shang, and M. S. Shang, "DCCR: Deep collaborative conjunctive recommender for rating prediction," *IEEE Access*, vol. 7, pp. 60186–60198, May 2019.

[40] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Dec. 2006.

[41] B. Rosner, R. J. Glynn, and M. L. T. Lee, "The Wilcoxon signed rank test for paired comparisons of clustered data," *Biometrics*, vol. 62, no. 1, pp. 185–192, Mar. 2006.

[42] D. Wu, X. Luo, G. Y. Wang, M. S. Shang, Y. Yuan, and H. Y. Yan, "A highly accurate framework for self-labeled semisupervised classification in industrial applications," *IEEE Trans. Ind. Inform.*, vol. 14, no. 3, pp. 909–920, Mar. 2018.

[43] Z. H. Zhou and J. Feng, "Deep forest: Towards an alternative to deep neural networks," in *Proc. the 26th Int. Joint Conf. on Artificial Intelligence*, Melbourne, Australia, 2017, pp. 3553–3559.

[44] X. Y. Ren, M. N. Song, E. Haihong, and J. D. Song, "Context-aware probabilistic matrix factorization modeling for point-of-interest recommendation," *Neurocomputing*, vol. 241, pp. 38–55, Jun. 2017.

[45] H. Wu, Z. X. Zhang, K. Yue, B. B. Zhang, J. He, and L. C. Sun, "Dual-regularized matrix factorization with deep neural networks for recommender systems," *Knowl.-Based Syst.*, vol. 145, pp. 46–58, Apr. 2018.

[46] C. Wang, Q. Liu, R. Z. Wu, E. H. Chen, C. R. Liu, X. P. Huang, and Z. Y. Huang, "Confidence-aware matrix factorization for recommender systems, " in *Proc. the 32nd AAAI Conf. Artificial Intelligence*, New Orleans, USA, 2018, pp. 434–442.

[47] D. Wu, X. Luo, M. S. Shang, Y. He, G. Y. Wang, and M. C. Zhou, "A deep latent factor model for high-dimensional and sparse matrices in recommender systems," *IEEE Trans. Syst., Man, Cybern.: Syst.*, to be published. DOI: 10.1109/TSMC.2019.2931393

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

[49] X. N. He, L. Z. Liao, H. W. Zhang, L. Q. Nie, X. Hu, and T. S. Chua, "Neural collaborative filtering," in *Proc. the 26th Int. Conf. World Wide Web*, Perth, Australia, 2017, pp. 173–182.

[50] P. J. Li, Z. H. Wang, Z. C. Ren, L. D. Bing, and W. Lam, "Neural rating regression with abstractive tips generation for recommendation," in *Proc. the 40th Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, Shinjuku, Japan, 2017, pp. 345–354.

[51] X. N. He and T. S. Chua, "Neural factorization machines for sparse predictive analytics," in *Proc. the 40th Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, Shinjuku, Japan, 2017, pp. 355–364.

[52] J. Xiao, H. Ye, X. N. He, H. W. Zhang, F. Wu, and T. S. Chua, "Attentional factorization machines: Learning the weight of feature interactions via attention networks," in *Proc. the 26th Int. Joint. Conf. on Artificial Intelligence*, Melbourne, Australia, 2017, pp. 3119–3125.

[53] L. Zheng, V. Noroozi, and P. S. Yu, "Joint deep modeling of users and items using reviews for recommendation," in *Proc. the 10th ACM Int. Conf. Web Search and Data Mining*, Cambridge, UK, 2017, pp. 425–434.

[54] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware recommendation," in *Proc. the 10th ACM Conf. Recommender Systems*, Boston, USA, 2016, pp. 233–240.

[55] N. Y. Zeng, Z. D. Wang, H. Zhang, K. E. Kim, Y. R. Li and X. H. Liu, "An improved particle filter with a novel hybrid proposal distribution for quantitative analysis of gold immunochromatographic strips," *IEEE Trans. Nanotechnol.*, vol. 18, pp. 819–829, Aug. 2019.

[56] N. Y. Zeng, Z. D. Wang, B. Zineddin, Y. R. Li, M. Du, L. Xiao, X. H. Liu, and T. Young, "Image-based quantitative analysis of gold immunochromatographic strip via cellular neural network approach," *IEEE Trans. Med. Imag.*, vol. 33, no. 5, pp. 1129–1136, May 2014.

[57] N. Y. Zeng, H. Li, Z. D. Wang, W. B. Liu, S. M. Liu, F. E. Alsaadi, and X. H. Liu, "Deep-reinforcement-learning-based images segmentation for quantitative analysis of gold immunochromatographic strip," *Neurocomputing*, to be published. DOI: 10.1016/j.neucom.2020.04.001

[58] Z. Q. Shu, X. J. Wu, C. Z. You, Z. Liu, P. Li, H. H. Fan, and F. Y. Ye, "Rank-constrained nonnegative matrix factorization for data representation," *Inform. Sci.*, vol. 528, pp. 133–146, Aug. 2020.

Di Wu (M'19) received the Ph.D. degree in computer application technology from Chongqing Institute of Green and Intelligent Technology (CIGIT), Chinese Academy of Sciences (CAS), Chongqing, China in 2019. Currently, he is an Associate Professor with the CIGIT, CAS. He was a visiting scholar during the time from April 2018 to April 2019 at the University of Louisiana, Lafayette, USA. His research interests include machine learning and data mining. He has published over 40 papers in *IEEE Trans. Syst, Man, Cybern.: Syst., IEEE Trans. Knowl. Data Eng., IEEE Trans. Ind. Inform., IEEE Trans. Serv. Comput., IEEE Trans. Neural Netw. Learn. Syst., IEEE International Conference on Data Mining, ACM International World Wide Web Conferences, International Joint Conference on Artificial Intelligence*, etc.

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. He is currently also a distinguished Professor of computer science with the Dongguan University of Technology, Dongguan, China. In 2016, he joined the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China, as a Professor of computer science and engineering. His current research interests include big data analysis and intelligent control. He has published over 100 papers (including over 50 IEEE transactions papers) in the above areas. He was a recipient of the Hong Kong Scholar Program jointly by the Society of Hong Kong Scholars and China Post-Doctoral Science Foundation in 2014, the Pioneer Hundred Talents Program of Chinese Academy of Sciences in 2016, and the Advanced Support of the Pioneer Hundred Talents Program of Chinese Academy of Sciences in 2018. He is currently serving as an Associate Editor for the *IEEE/CAA Journal of Automatica Sinica, IEEE Access*, and *Neurocomputing*. He received the Outstanding Associate Editor award of *IEEE Access* in 2018. He has also served as the Program Committee Member for over 20 international conferences.