Non-Negative Latent Factor Model Based on β-Divergence for Recommender Systems

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

***Abstract*—Non-negative latent factor (NLF) models well represent high-dimensional and sparse (HiDS) matrices filled with non-negative data, which are frequently encountered in industrial applications like recommender systems. However, current NLF models mostly adopt Euclidean distance in their objective function, which represents a special case of a β-divergence function. Hence, it is highly desired to design a β-divergence-based NLF (β-NLF) model that uses a β-divergence function, and investigate its performance in recommender systems as β varies. To do so, we first model β-NLF's learning objective with a β-divergence function. Subsequently, we deduce a general single latent factor-dependent, non-negative and multiplicative update scheme for β-NLF, and then design an efficient β-NLF algorithm. The experimental results on HiDS matrices from industrial applications indicate that by carefully choosing the value of β, β-NLF outperforms an NLF model with Euclidean distance in terms of accuracy for missing data prediction without increasing computational time. The research outcomes show the necessity of using an optimal β-divergence function in order to achieve the best performance of an NLF model on HiDS matrices. Hence, the proposed model has both theoretical and application significance.**

***Index Terms*—β-divergence, big data, high-dimensional and sparse (HiDS) matrix, industrial application, learning algorithm, non-negative latent factor (NLF) analysis, recommender system.**

Manuscript received June 3, 2019; accepted July 18, 2019. Date of publication August 21, 2019; date of current version July 19, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61772493, Grant 91646114, Grant 51609229, Grant 61872065, and Grant 61702475, in part by the National Key Research and Development Program of China under Grant 2017YFC0804002, in part by the Chongqing Cultivation Program of Innovation and Entrepreneurship Demonstration Group under Grant cstc2017kjrc-cxcytd0149, in part by the Chongqing Overseas Scholars Innovation Program under Grant cx2017012 and Grant cx2018011, in part by the Chongqing Research Program of Technology Innovation and Application under Grant cstc2017zdcy-zdyfX0076, Grant cstc2018jszx-cyztzxX0025, Grant cstc2017rgzn-zdyfX0020, Grant cstc2017zdcy-zdyf0554, and Grant cstc2017rgzn-zdyf0118, and in part by the Pioneer Hundred Talents Program of Chinese Academy of Sciences. This paper was recommended by Associate Editor J. A. Lozano. *(Xin Luo and Ye Yuan are co-first authors.) (Corresponding authors: MengChu Zhou; Mingsheng Shang.)*

X. Luo is with the School of Computer Science and Technology, Dongguan University of Technology, Dongguan 523808, China (e-mail: luoxin21@gmail.com).

Y. Yuan is with the Chongqing Engineering Research Center of Big Data Application for Smart Cities, Chinese Academy of Sciences, Chongqing 400714, China, also with the Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: yuanye@cigit.ac.cn).

M. Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA, and also with the Center of Research Excellence in Renewable Energy and Power Systems, King Abdulaziz University, Jeddah 21589, Saudi Arabia (e-mail: zhou@njit.edu).

Z. Liu and M. Shang are with the Chongqing Engineering Research Center of Big Data Application for Smart Cities, Chinese Academy of Sciences, Chongqing 400714, China, and also with the 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: liuzhigang@cigit.ac.cn; msshang@cigit.ac.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSMC.2019.2931468.

Digital Object Identifier 10.1109/TSMC.2019.2931468

I. Introduction

RAPID expansion of World Wide Web brings people great convenience. Online consumptions become indispensable in their daily life since millions of products are provided online. However, such massive data lead to the severe problem of information overload: it becomes highly difficult for people to pick up their truly desired information from big data. Intelligent, efficient, and robust models for addressing this issue are greatly desired. In such context, recommender systems, which can perform efficient information filtering, have attracted great interests [1]–[4]. They address the problem of information overload by connecting valuable information to right people actively, rather than passively, according to their information utility history [1]–[4].

High-dimensional and sparse (HiDS) matrices are frequently adopted to describe people's information utility history in recommender systems [5]–[8] because: 1) numerous entities are involved in a recommender system, e.g., billions of commodities are available to billions of users on Taobao [9], making the resultant relationship extremely high-dimensional and 2) each user can only touch a tiny subset of numerous items and each item can only be touched by a tiny subset of users, making the resultant relationship extremely sparse. Note that unlike a traditional sparse matrix whose entries are mostly zeroes, an HiDS matrix is filled with numerous unknown data with only a tiny subset of its whole entry set being observed.

In spite of their sparsity, HiDS matrices generated by recommender systems contain rich knowledge regarding various desired patterns like user community [10], [11], item clusters [12], [13], users' potential favorites [2]–[4], and key profile features [5]–[8]. Latent factor (LF) models can analyze HiDS matrices efficiently, yet they do not meet non-negativity constraints [5], [6], [14]–[16]. As proven by prior research, non-negative features can better represent HiDS matrices filled with non-negative data in most recommender systems [1]–[4],

and describe hidden patterns like user profile features [5], [6] and community tendencies [11], [17] more precisely.

Given a complete matrix, a non-negative matrix factorization (NMF) model can extract non-negative LFs from it efficiently for various pattern analysis tasks [18]–[24]. Paatero and Tapper [25] applied alternating least squares to train desired LFs and truncate negative LFs to zeroes for keeping them non-negative. Lee and Seung [18] derived the non-negative and multiplicative update (NMU) for the desired LF matrices, which maintains the non-negativity of involved LFs if they are initially non-negative. Lin [26] adopted projected gradient decent to implement an NMF model. This model also works by truncating negative LFs to zero during its training process, but adopts gradient descent in its training scheme. These models and their extensions [25]–[29] can well analyze a complete matrix, extracting non-negative LFs from it. However, they are not applicable to HiDS matrices filled with numerous missing data.

Great efforts have been made for applying existing NMF models to LF analysis on HiDS matrices [7], [8], [17], [30]. Zhang *et al.* [17] proposed a weighted NMF (WNMF) model to construct an intermediate matrix by filling the unknown entries of the target matrix with zeroes, and then apply NMU to it to obtain desired non-negative LFs. Xu *et al.* [30] proposed a non-negative matrix completion model, which adopts a full approximation to the target matrix and then applies the projected alternating least squares on it for acquiring non-negative LFs. These models are able to address HiDS matrices with existing NMF algorithms [18]–[29], but suffer unacceptably high computational and storage costs because they rely on full matrices with the same row and column counts as an HiDS one from the start to the end. Note that an HiDS matrix can be extremely sparse with numerous rows and columns. For instance, the dating agency matrix collected by LibimSeTi [43] has 135 359 rows and 168 791 columns. It contains only 17 359 346 known entries, thus resulting in very low data density at 0.076%. However, its full approximation consists of nearly 23 billion entries. To handle such a huge matrix can be extremely time-consuming at the cost of huge computational and storage burden, and sometimes even intractable in industrial applications [5]–[8].

For implementing non-negative LF analysis on HiDS matrices with high efficiency and low cost, Luo *et al.* [7], [33] proposed a single LF-dependent, NMU (SLF-NMU) for extracting non-negative LFs from an HiDS matrix on its known entries only, thereby greatly improving the computational efficiency as well as alleviating the computational and storage burden. Based on SLF-NMU, they further proposed a non-negative LF (NLF) model. Given an HiDS matrix, an NLF model's computational cost is linear with its known entries only, and its storage cost is linear with its involved entity count only [7], [33].

An NLF model can concisely represent an HiDS matrix efficiently. It relies on an objective function defined via Euclidean distance, which is only a special case of $\beta = 2$ in a β-divergence function [32], [34], [35], [56]–[59]. Note that for an NLF model, the objective function has vital effects on its various characteristics, such as the convergence rate and prediction accuracy for missing data [7], [8], [17], [30], [33]. Consequently, a highly interesting question arises: can an NLF model with a β-divergence function in its objective function offer the optimal performance exceeding that of a model with commonly used Euclidean distance? Since the Euclidean distance is the most frequently adopted objective function for building an NLF model, investigations into this issue is highly innovative and useful for industrial applications seeking for the most accurate NLF model on an HiDS matrix [7], [8], [17], [30], [33]. This paper aims to develop a generalized NLF (β-NLF) model with a β-divergence function as its objective function for the first time. The main contributions include the following.

1) A β-NLF model whose objective function is a β-divergence function.
2) A β-divergence and SLF-NMU (β-SLF-NMU) scheme for a β-NLF model as β varies by analyzing its Karush–Kuhn–Tucker (KKT) conditions.
3) Algorithm design and analysis for a β-NLF model.
4) Empirical validations on four HiDS matrices from industrial recommender systems currently in use.

Note that as shown in prior research [57]–[59], NMF models based on β-divergence or more generalized α-β-divergence are well explored. They have proven to be efficient in extracting useful knowledge from complete matrices. However, a β-NLF model differs from them in the following aspects.

1) A β-NLF model takes an HiDS matrix as the input, however, generalized NMF models proposed in [57]–[59] focuses on complete matrices and cannot address an HiDS matrix directly.
2) For addressing an HiDS matrix, this paper proposes a β-SLF-NMU scheme, which is a generalized form of an SLF-NMU scheme [7]. It is defined on the known data of an HiDS matrix only, and updates involved LFs in a single element-dependent way to achieve high efficiency in both computation and storage. In comparison, NMF models proposed in [57]–[59] generalize an NMU scheme for minimizing a learning objective based on the β-divergence or α-β-divergence defined on a complete matrix. Hence, their learning schemes are not compatible with an HiDS matrix concerned in this paper.

The rest of this paper is organized as follows. Section II gives the preliminaries. Section III presents the methods. Section IV gives the experiments. Section V discusses some similar work and future plan. Finally, Section VI concludes this paper.

II. Preliminaries

A. Problem Statement

For an LF-based recommender, a user-item rating-matrix built on people's information usage history is taken as the fundamental data source. Given an item set I and a user set U, a user-item rating-matrix R is defined as follows.

Definition 1: R is a $|U| \times |I|$ matrix where each element $r_{u,i}$ is connected with user u's preference on item i.

Let R_K and R_W denote the known and whole entry sets of R, respectively. An LF model seeks for R's low-rank approximation $\hat{R}$ as defined in [5]–[8] and [12]–[16].

Definition 2: $\hat{R}$ is R's rank-f approximation built on R_K.

Note that f denotes the rank of $\hat{R}$ and we naturally have $f \ll \min\{|U|, |I|\}$. To obtain $\hat{R}$, an LF model extracts LF matrices $P^{|U|\times f}$ and $Q^{f\times|I|}$ from R_K. Note that P and Q are actually interpreted as the user and item LF matrices reflecting user and item characteristics hidden in R_K. Under such circumstances, f is also interpreted as the dimension of an LF space.

The LF extraction is taken by minimizing an objective function measuring the difference between R_K and the corresponding entry set in $\hat{R}$. For such purposes, the Euclidean distance is mostly adopted in prior work [5]–[8], [12]–[17]. Moreover, when R is defined on the non-negative field of real numbers, P and Q are expected to be non-negative for representing the target matrix and its hidden patterns more precisely. Thus, P and Q should be subject to the non-negativity constraints to achieve an NLF model. The Euclidean distance-based objective function for an NLF model is formulated as

$$\varepsilon = \sum_{(u,i)\in R_K}\left(\left(r_{u,i} - \sum_{m=1}^{f} p_{u,m}q_{m,i}\right)^2 + \lambda_P \sum_{m=1}^{f} p_{u,m}^2 + \lambda_Q \sum_{m=1}^{f} q_{m,i}^2\right)$$
$$\text{s.t.} \quad P \geq 0, Q \geq 0 \tag{1}$$

where ε denotes the objective function with respect to the desired LF matrices P and Q, constants λ_P and λ_Q denote the regularization coefficients, and terms behind λ_P and λ_Q are the regularization terms for avoiding overfitting, respectively. Note that we apply the modified l_2-norm based regularization discussed in [5]–[8] for accurately describing the sparsity of R.

B. Non-Negative Latent Factor Model

For extracting non-negative LFs from R_K, NLF adopts an SLF-NMU scheme to minimize (1). This scheme applies additive gradient descent (AGD) with respect to each desired LF in (1), resulting in the following parameter learning rules:

$$\arg\min_{P,Q} \varepsilon(P,Q) \overset{\text{AGD}}{\Rightarrow}$$
$$\begin{cases} p_{u,m} \leftarrow p_{u,m} - \eta_{u,m}\sum_{i\in I_u}\left(\lambda_P p_{u,m} - q_{m,i}\left(r_{u,i} - \sum_{m=1}^{f} p_{u,m}q_{m,i}\right)\right) \\ q_{m,i} \leftarrow q_{m,i} - \eta_{m,i}\sum_{u\in U_i}\left(\lambda_Q q_{m,i} - p_{u,m}\left(r_{u,i} - \sum_{m=1}^{f} p_{u,m}q_{m,i}\right)\right) \end{cases} \tag{2}$$

where I_u denotes the subset of I related to user u, U_i denotes the subset of U related to item i, and $\eta_{u,m}$ and $\eta_{m,i}$ denote the learning rates corresponding to $p_{u,m}$ and $q_{m,i}$, respectively. With (2), $p_{u,m}$ and $q_{m,i}$ might become negative because $-\eta_{u,m}\sum_{i\in I_u}(q_{m,i}\hat{r}_{u,i} + \lambda_P p_{u,m})$ and $-\eta_{m,i}\sum_{u\in U_i}(p_{u,m}\hat{r}_{u,i} + \lambda_Q q_{m,i})$ are negative terms for $p_{u,m}$ and $q_{m,i}$, respectively. For cancelling them, SLF-NMU manipulates $\eta_{u,m}$ and $\eta_{m,i}$ to cancel the negative terms in the resultant AGD-based updating rules

$$\eta_{u,m} = \frac{p_{u,m}}{\left(\sum_{i\in I_u} q_{m,i}\hat{r}_{u,i} + \lambda_P |I_u| p_{u,m}\right)}$$
$$\eta_{m,i} = \frac{q_{m,i}}{\left(\sum_{u\in U_i} p_{u,m}\hat{r}_{u,i} + \lambda_Q |U_i| q_{m,i}\right)}. \tag{3}$$

By substituting (3) to (2) following [7], we achieve:

$$\arg\min_{P,Q} \varepsilon(P,Q) \overset{\text{SLF-NMU}}{\Rightarrow} \forall u \in U, i \in I, m \in \{1,\ldots,f\}$$
$$\begin{cases} p_{u,m} \leftarrow p_{u,m}\frac{\sum_{i\in I_u} q_{m,i}r_{u,i}}{\sum_{i\in I_u} q_{m,i}\hat{r}_{u,i} + \lambda_P |I_u| p_{u,m}} \\ q_{m,i} \leftarrow q_{m,i}\frac{\sum_{u\in U_i} p_{u,m}r_{u,i}}{\sum_{u\in U_i} p_{u,m}\hat{r}_{u,i} + \lambda_Q |U_i| q_{m,i}}. \end{cases} \tag{4}$$

With (4), all LFs in P and Q are kept non-negative if so initially. Thus, the NLF model is achieved, which is frequently adopted in various data analysis tasks requiring non-negative LF analysis [33], [36]–[39].

Nonetheless, NLF's objective function (1) only considers the special case of $\beta = 2$ in a β-divergence function. Naturally, with different β-divergence-based objective functions, we achieve different NLF models. In the next section, we present a β-NLF model that fully considers all cases in a β-divergence function.

III. Generalized Non-Negative Latent Factor Model

A. Generalized β-Divergence for β-NLF

The frequently adopted Euclidean distance in LF models is only a special case of a β-divergence function [32], [34], [35], [56]–[59]. Given U, I, R, and R_K, the generalized β-divergence between R and $\hat{R} = PQ$ for an LF model is defined as follows:

$$\begin{cases} \beta = 0 : d_0 = \sum_{r_{u,i}\in R_K}\left(\frac{r_{u,i}}{\hat{r}_{u,i}} - \log\frac{r_{u,i}}{\hat{r}_{u,i}} - 1\right) \\ \beta = 1 : d_1 = \sum_{r_{u,i}\in R_K}\left(r_{u,i}\log\frac{r_{u,i}}{\hat{r}_{u,i}} - r_{u,i} + \hat{r}_{u,i}\right) \\ \beta \neq 0 \text{ or } 1 : d_\beta = \sum_{r_{u,i}\in R_K}\frac{\left(r_{u,i}^\beta + (\beta-1)\hat{r}_{u,i}^\beta - \beta r_{u,i}\hat{r}_{u,i}^{\beta-1}\right)}{\beta(\beta-1)}. \end{cases} \tag{5}$$

By integrating the regularization terms and non-negativity constraints into (5), we have the objective function for a β-NLF model as follows:

$$\begin{cases} \varepsilon_0 = \sum_{r_{u,i}\in R_K}\left(\left(\frac{r_{u,i}}{\hat{r}_{u,i}} - \log\frac{r_{u,i}}{\hat{r}_{u,i}} - 1\right) + \lambda_P\sum_{m=1}^{f} p_{u,m}^2 + \lambda_Q\sum_{m=1}^{f} q_{m,i}^2\right) \\ \varepsilon_1 = \sum_{r_{u,i}\in R_K}\left(\left(r_{u,i}\log\frac{r_{u,i}}{\hat{r}_{u,i}} - r_{u,i} + \hat{r}_{u,i}\right) + \lambda_P\sum_{m=1}^{f} p_{u,m}^2 + \lambda_Q\sum_{m=1}^{f} q_{m,i}^2\right) \\ \varepsilon_\beta = \sum_{r_{u,i}\in R_K}\left(\frac{\left(r_{u,i}^\beta + (\beta-1)\hat{r}_{u,i}^\beta - \beta r_{u,i}\hat{r}_{u,i}^{\beta-1}\right)}{\beta(\beta-1)} + \lambda_P\sum_{m=1}^{f} p_{u,m}^2 + \lambda_Q\sum_{m=1}^{f} q_{m,i}^2\right) \end{cases}$$
$$\text{s.t.} \quad P \geq 0, Q \geq 0 \tag{6}$$

where ε_0, ε_1, and ε_β denote learning objectives with $\beta = 0$, $\beta = 1$, and $\beta \notin \{0, 1\}$, respectively. Next, we deduce the β-SLF-NMU scheme for each case in (6).

B. Case 1: β = 0

SLF-NMU proposed in [7] manipulates the learning rates in AGD-based update to achieve a multiplicative training scheme as in (4). To achieve a β-NLF-NMU scheme with more solid deduction, this paper analyzes the KKT conditions

of the objective function for the same purposes. When $\beta = 0$, a β-NLF model's objective function is given by

$$\varepsilon_0 = \sum_{r_{u,i}\in R_K}\left(\left(\frac{r_{u,i}}{\hat{r}_{u,i}} - \log\frac{r_{u,i}}{\hat{r}_{u,i}} - 1\right) + \lambda_P\sum_{m=1}^{f} p_{u,m}^2 + \lambda_Q\sum_{m=1}^{f} q_{m,i}^2\right)$$
$$\text{s.t.}\quad P \geq 0, Q \geq 0. \tag{7}$$

Let $\Gamma^{|U|\times f}$ and $K^{f\times|I|}$ be the Lagrangian multipliers corresponding to the constraints $P = 0$ and $Q = 0$, respectively. Note that (7) also denotes the Itakura–Saito divergence between R and $\hat{R}$ on R_K. Then, we build the Lagrangian function corresponding to (7)

$$\begin{aligned} L_0 &= \varepsilon_0 + \mathrm{tr}\left(\Gamma\ P^T\right) + \mathrm{tr}\left(\mathrm{K}^T Q\right) \\ &= \varepsilon_0 + \sum_u\sum_m \gamma_{u,m}p_{u,m} + \sum_i\sum_m \kappa_{m,i}q_{m,i} \end{aligned} \tag{8}$$

where the operator $\mathrm{tr}(\cdot)$ computes the trace of the enclosed matrix. Considering the partial derivatives of L_0 with respect to the single LFs $p_{u,m}$ and $q_{m,i}$, we obtain

$$\begin{cases} \frac{\partial L_0}{\partial p_{u,m}} = \frac{\partial \varepsilon_0}{\partial p_{u,m}} + \gamma_{u,m} = \sum_{i\in I_u}\left(-\frac{r_{u,i}q_{m,i}}{\hat{r}_{u,i}^2} + \frac{q_{m,i}}{\hat{r}_{u,i}} + 2\lambda_P p_{u,m}\right) + \gamma_{u,m} \\ \frac{\partial L_0}{\partial q_{m,i}} = \frac{\partial \varepsilon_0}{\partial q_{m,i}} + \kappa_{m,i} = \sum_{u\in U_i}\left(-\frac{r_{u,i}p_{u,m}}{\hat{r}_{u,i}^2} + \frac{p_{u,m}}{\hat{r}_{u,i}} + 2\lambda_Q q_{m,i}\right) + \kappa_{m,i}. \end{cases} \tag{9}$$

From (8) and (9), we have the following observations.

1) To achieve the local optima of L_0 with respect to $p_{u,m}$ and $q_{m,i}$, the partial derivatives in (9) should be set at zero simultaneously.
2) The KKT condition of the Lagrangian function (8) is $\forall u \in U,\ i \in I,\ m \in \{1,\ldots,f\} : \gamma_{u,m}p_{u,m} = 0$, and $\kappa_{m,i}q_{m,i} = 0$.

With 1) and 2), we have the following deduction based on (9):

$$\begin{cases} p_{u,m}\sum_{i\in I_u}\left(-\frac{r_{u,i}q_{m,i}}{\hat{r}_{u,i}^2} + \frac{q_{m,i}}{\hat{r}_{u,i}} + 2\lambda_P p_{u,m}\right) = 0 \\ q_{m,i}\sum_{u\in U_i}\left(-\frac{r_{u,i}p_{u,m}}{\hat{r}_{u,i}^2} + \frac{p_{u,m}}{\hat{r}_{u,i}} + 2\lambda_Q q_{m,i}\right) = 0. \end{cases} \tag{10}$$

Note that (10) can be rearranged as follows:

$$\begin{cases} p_{u,m}\sum_{i\in I_u}\frac{r_{u,i}q_{m,i}}{\hat{r}_{u,i}^2} = p_{u,m}\sum_{i\in I_u}\left(\frac{q_{m,i}}{\hat{r}_{u,i}} + \lambda_P p_{u,m}\right) \\ q_{m,i}\sum_{u\in U_i}\frac{r_{u,i}p_{u,m}}{\hat{r}_{u,i}^2} = q_{m,i}\sum_{u\in U_i}\left(\frac{p_{u,m}}{\hat{r}_{u,i}} + \lambda_Q q_{m,i}\right). \end{cases} \tag{11}$$

Note that in (11) we slightly abuse the symbols by folding the constant two into the regularization coefficients λ_P and λ_Q. With it, the iterative expressions of $p_{u,m}$ and $q_{m,i}$ are formulated as

$$\begin{cases} p_{u,m} = p_{u,m}\frac{\sum_{i\in I_u}\frac{r_{u,i}q_{m,i}}{\hat{r}_{u,i}^2}}{\sum_{i\in I_u}\frac{q_{m,i}}{\hat{r}_{u,i}} + \lambda_P|I_u|p_{u,m}} \\ q_{m,i} = q_{m,i}\frac{\sum_{u\in U_i}\frac{r_{u,i}p_{u,m}}{\hat{r}_{u,i}^2}}{\sum_{u\in U_i}\frac{p_{u,m}}{\hat{r}_{u,i}} + \lambda_Q|U_I|q_{m,i}} \end{cases} \tag{12}$$

which is the β-SLF-NMU rule for LFs with $\beta = 0$ in β-NLF.

C. Case 2: $\beta = 1$

When $\beta = 1$, a β-NLF model's objective function is given as

$$\varepsilon_1 = \sum_{r_{u,i}\in R_K}\left(\left(r_{u,i}\log\frac{r_{u,i}}{\hat{r}_{u,i}} - r_{u,i} + \hat{r}_{u,i}\right) + \lambda_P\sum_{m=1}^{f} p_{u,m}^2 + \lambda_Q\sum_{m=1}^{f} q_{m,i}^2\right)$$
$$\text{s.t.}\quad P \geq 0, Q \geq 0. \tag{13}$$

Note that (13) also denotes the Kullback–Leibler divergence [34] between R and $\hat{R}$ on R_K. We build (13)'s Lagrangian function as

$$L_1 = \varepsilon_1 + \sum_u\sum_m \gamma_{u,m}p_{u,m} + \sum_i\sum_m \kappa_{m,i}q_{m,i}. \tag{14}$$

Considering the partial derivatives of L_1 with respect to the single LFs $p_{u,m}$ and $q_{m,i}$, we obtain

$$\begin{cases} \frac{\partial L_1}{\partial p_{u,m}} = \frac{\partial \varepsilon_1}{\partial p_{u,m}} + \gamma_{u,m} \\ \qquad = \sum_{i\in I_u}\left(-\frac{r_{u,i}q_{m,i}}{\hat{r}_{u,i}} + q_{m,i} + 2\lambda_P p_{u,m}\right) + \gamma_{u,m} \\ \frac{\partial L_1}{\partial q_{m,i}} = \frac{\partial \varepsilon_1}{\partial q_{m,i}} + \kappa_{m,i} \\ \qquad = \sum_{u\in U_i}\left(-\frac{r_{u,i}p_{u,m}}{\hat{r}_{u,i}} + p_{u,m} + 2\lambda_Q q_{m,i}\right) + \kappa_{m,i}. \end{cases} \tag{15}$$

By combining (15) with the KKT condition of (14), i.e., $\forall u \in U,\ i \in I, m \in \{1,\ldots,f\}\ \gamma_{u,m}p_{u,m} = 0,\ \kappa_{m,i}q_{m,i} = 0$, we have

$$\begin{cases} p_{u,m}\sum_{i\in I_u}\left(-\frac{r_{u,i}q_{m,i}}{\hat{r}_{u,i}} + q_{m,i} + 2\lambda_P p_{u,m}\right) = 0 \\ q_{m,i}\sum_{u\in U_i}\left(-\frac{r_{u,i}p_{u,m}}{\hat{r}_{u,i}} + p_{u,m} + 2\lambda_Q q_{m,i}\right) = 0 \end{cases} \tag{16}$$

which can be rearranged to result in the following equation:

$$\begin{cases} p_{u,m}\sum_{i\in I_u}\frac{r_{u,i}q_{m,i}}{\hat{r}_{u,i}} = p_{u,m}\sum_{i\in I_u}\left(q_{m,i} + \lambda_P p_{u,m}\right) \\ q_{m,i}\sum_{u\in U_i}\frac{r_{u,i}p_{u,m}}{\hat{r}_{u,i}} = q_{m,i}\sum_{u\in U_i}\left(p_{u,m} + \lambda_Q q_{m,i}\right). \end{cases} \tag{17}$$

Based on (17), $p_{u,m}$ and $q_{m,i}$ are iteratively expressed as

$$\begin{cases} p_{u,m} = p_{u,m}\frac{\sum_{i\in I_u}\frac{r_{u,i}q_{m,i}}{\hat{r}_{u,i}}}{\sum_{i\in I_u} q_{m,i} + \lambda_P|I_u|p_{u,m}} \\ q_{m,i} = q_{m,i}\frac{\sum_{u\in U_i}\frac{r_{u,i}p_{u,m}}{\hat{r}_{u,i}}}{\sum_{u\in U_i} p_{u,m} + \lambda_Q|U_I|q_{m,i}} \end{cases} \tag{18}$$

which is exactly the β-SLF-NMU rule with $\beta = 1$ in β-NLF.

D. Case 3: $\beta \notin \{0, 1\}$

When $\beta \notin \{0, 1\}$, the objective function becomes

$$\varepsilon_\beta = \sum_{r_{u,i}\in R_K}\left(\frac{\left(r_{u,i}^\beta + (\beta-1)\hat{r}_{u,i}^\beta - \beta r_{u,i}\hat{r}_{u,i}^{\beta-1}\right)}{\beta(\beta-1)} + \lambda_P\sum_{m=1}^{f} p_{u,m}^2 + \lambda_Q\sum_{m=1}^{f} q_{m,i}^2\right)$$
$$\text{s.t.}\quad P \geq 0, Q \geq 0. \tag{19}$$

Then, we build the Lagrangian function of (19) as follows:

$$L_\beta = \varepsilon_\beta + \sum_u\sum_m \gamma_{u,m}p_{u,m} + \sum_i\sum_m \kappa_{m,i}q_{m,i}. \tag{20}$$

Considering the partial derivatives of L_β with respect to $p_{u,m}$ and $q_{m,i}$, we obtain

$$\begin{cases} \frac{\partial L_\beta}{\partial p_{u,m}} = \frac{\partial \varepsilon_\beta}{\partial p_{u,m}} + \gamma_{u,m} = \sum_{i\in I_u}\Big(q_{m,i}\hat{r}_{u,i}^{\beta-1} - q_{m,i}r_{u,i}\hat{r}_{u,i}^{\beta-2} \\ \qquad\qquad + 2\lambda_P p_{u,m}\Big) + \gamma_{u,m} \\ \frac{\partial L_\beta}{\partial q_{m,i}} = \frac{\partial \varepsilon_\beta}{\partial q_{m,i}} + \kappa_{m,i} = \sum_{u\in U_i}\Big(p_{u,m}\hat{r}_{u,i}^{\beta-1} - p_{u,m}r_{u,i}\hat{r}_{u,i}^{\beta-2} \\ \qquad\qquad + 2\lambda_Q q_{m,i}\Big) + \kappa_{m,i}. \end{cases} \tag{21}$$

With (21) and the KKT condition of (20), i.e., $\forall u \in U, i \in I, m \in \{1, \ldots, f\} : \gamma_{u,m}p_{u,m} = 0$, $\kappa_{m,i}q_{m,i} = 0$, we obtain

$$\begin{cases} p_{u,m}\sum_{i\in I_u}\Big(q_{m,i}\hat{r}_{u,i}^{\beta-1} - q_{m,i}r_{u,i}\hat{r}_{u,i}^{\beta-2} + 2\lambda_P p_{u,m}\Big) = 0 \\ q_{m,i}\sum_{u\in U_i}\Big(p_{u,m}\hat{r}_{u,i}^{\beta-1} - p_{u,m}r_{u,i}\hat{r}_{u,i}^{\beta-2} + 2\lambda_Q q_{m,i}\Big) = 0 \end{cases} \tag{22}$$

which can be rearranged to achieve

$$\begin{cases} p_{u,m}\sum_{i\in I_u} q_{m,i}r_{u,i}\hat{r}_{u,i}^{\beta-2} = p_{u,m}\sum_{i\in I_u}\Big(q_{m,i}\hat{r}_{u,i}^{\beta-1} + \lambda_P p_{u,m}\Big) \\ q_{m,i}\sum_{u\in U_i} p_{u,m}r_{u,i}\hat{r}_{u,i}^{\beta-2} = q_{m,i}\sum_{u\in U_i}\Big(p_{u,m}\hat{r}_{u,i}^{\beta-1} + \lambda_Q q_{m,i}\Big). \end{cases} \tag{23}$$

From (23), we obtain the iterative expressions of $p_{u,m}$ and $q_{m,i}$ with $\beta \notin \{0, 1\}$ as follows:

$$\begin{cases} p_{u,m} = p_{u,m}\frac{\sum_{i\in I_u} q_{m,i}r_{u,i}\hat{r}_{u,i}^{\beta-2}}{\sum_{i\in I_u} q_{m,i}\hat{r}_{u,i}^{\beta-1} + \lambda_P|I_u|p_{u,m}} \\ q_{m,i} = q_{m,i}\frac{\sum_{u\in U_i} p_{u,m}r_{u,i}\hat{r}_{u,i}^{\beta-2}}{\sum_{u\in U_i} p_{u,m}\hat{r}_{u,i}^{\beta-1} + \lambda_Q|U_I|q_{m,i}} \end{cases} \tag{24}$$

which is the β-SLF-NMU rule for LFs with $\beta \notin \{0, 1\}$ in β-NLF. Note that by substituting $\beta = 2$ into (24), we have the SLF-NMU rule (4) for the NLF model. From this point of view, NLF is certainly a special case of β-NLF, and SLF-NMU is also a special case of β-SLF-NMU.

E. β-NLF Algorithm Design and Analysis

By combining (12), (18), and (24), we obtain the general expression of the β-SLF-NMU rule for β-NLF

$$\begin{cases} \beta = 0: \begin{cases} p_{u,m} = p_{u,m}\sum_{i\in I_u}\frac{r_{u,i}q_{m,i}}{\hat{r}_{u,i}^2}\Big/\Big(\sum_{i\in I_u}\frac{q_{m,i}}{\hat{r}_{u,i}} + \lambda_P|I_u|p_{u,m}\Big) \\ q_{m,i} = q_{m,i}\sum_{u\in U_i}\frac{r_{u,i}p_{u,m}}{\hat{r}_{u,i}^2}\Big/\Big(\sum_{u\in U_i}\frac{p_{u,m}}{\hat{r}_{u,i}} + \lambda_Q|U_I|q_{m,i}\Big) \end{cases} \\ \beta = 1: \begin{cases} p_{u,m} = p_{u,m}\sum_{i\in I_u}\frac{r_{u,i}q_{m,i}}{\hat{r}_{u,i}}\Big/\Big(\sum_{i\in I_u} q_{m,i} + \lambda_P|I_u|p_{u,m}\Big) \\ q_{m,i} = q_{m,i}\sum_{u\in U_i}\frac{r_{u,i}p_{u,m}}{\hat{r}_{u,i}}\Big/\Big(\sum_{u\in U_i} p_{u,m} + \lambda_Q|U_I|q_{m,i}\Big) \end{cases} \\ \beta \notin \{0, 1\}: \begin{cases} p_{u,m} = p_{u,m}\sum_{i\in I_u} q_{m,i}r_{u,i}\hat{r}_{u,i}^{\beta-2}\Big/\Big(\sum_{i\in I_u} q_{m,i}\hat{r}_{u,i}^{\beta-1} \\ \qquad\qquad + \lambda_P|I_u|p_{u,m}\Big) \\ q_{m,i} = q_{m,i}\sum_{u\in U_i} p_{u,m}r_{u,i}\hat{r}_{u,i}^{\beta-2}\Big/\Big(\sum_{u\in U_i} p_{u,m}\hat{r}_{u,i}^{\beta-1} \\ \qquad\qquad + \lambda_Q|U_I|q_{m,i}\Big). \end{cases} \end{cases} \tag{25}$$

Based on (25), we design the algorithm for β-NLF as Algorithm 1. Note that β-NLF adopts four auxiliary matrices, i.e., $A^{|U|\times f}$, $B^{|U|\times f}$, $C^{f\times|I|}$, and $D^{f\times|I|}$, for updating desired parameters efficiently. The additional storage caused by these auxiliary matrices is $\Theta((|U| + |I|) \times f)$. Given that $f \ll \min\{|U|, |I|\}$, this additional cost is easy to resolve in practice.

Algorithm 1 β-NLF

Input: $\boldsymbol{U}$, $\boldsymbol{I}$, $\boldsymbol{R_K}$, $\boldsymbol{f}$, $\boldsymbol{\beta}$

Operation	Complexity
initialize $P^{\lvert U\rvert\times f}, A^{\lvert U\rvert\times f}, B^{\lvert U\rvert\times f}$ non-negatively	$\Theta(\lvert U\rvert \times f)$
initialize $Q^{f\times\lvert I\rvert}, C^{f\times\lvert I\rvert}, D^{f\times\lvert I\rvert}$ non-negatively	$\Theta(\lvert I\rvert \times f)$
initialize $\lambda_P, \lambda_Q, t = 0$, *Max-training-round* $= n$	$\Theta(1)$
while not converge **and** $t = n$ **do**	$\times t$
reset $A = 0, B = 0$	$\Theta(\lvert U\rvert \times f)$
reset $C = 0, D = 0$	$\Theta(\lvert I\rvert \times f)$
for each $r_{u,i}$ **in** R_K	$\times\lvert R_K\rvert$
$\hat{\mathbf{r}}_{u,i} = \sum_{m=1}^{f} p_{u,m}q_{m,i}$	$\times f$
for $m = 1$ **to** f	$\times f$
$(a_{u,m}, b_{u,m}, c_{m,i}, d_{m,i}) =$ **PROC_UPDATE**$(r_{u,i}, r_{u,i}, p_{u,m}, a_{u,m}, b_{u,m}, q_{m,i}, c_{m,i}, d_{m,i}, \beta, \lambda_P, \lambda_Q)$	$T_{\text{PROC\_UPDATE}}$
end for	-
end for	-
for $u \in U$	$\times\lvert U\rvert$
for $m = 1$ **to** f	$\times f$
$p_{u,m} = p_{u,m}(a_{u,m}/b_{u,m})$	$\Theta(1)$
end for	-
end for	-
for $i \in I$	$\times\lvert I\rvert$
for $m = 1$ **to** f	$\times f$
$q_{m,i} = q_{m,i}(c_{m,i}/d_{m,i})$	$\Theta(1)$
end for	-
end for	-
$t = t + 1$	$\Theta(1)$
end while	-

Output: $\boldsymbol{P}, \boldsymbol{Q}$

Moreover, β-NLF relies on Procedure 1, which computes the update increment relying on each $r_{u,i} \in R_K$ during the traverse on R_K. Its pseudo code is given in Procedure 1. As depicted in PROC_UPDATE, the computation of the update increment depends on β, corresponding to the three different cases discussed in Sections III-B–III-D. Meanwhile, as analyzed in Procedure 1, $T_{\text{PROC\_UPDATE}} = \Theta(1)$.

Based on β-NLF and PROC_UPDATE, we summarize β-NLF's computational cost as follows:

$$\begin{aligned} T_{GNLF} &= \Theta(n \times (|U| + |I|) \times f + n \times |R_K| \times f) \\ &\approx \Theta(n \times |R_K| \times f). \end{aligned} \tag{26}$$

Note that (26) adopts the condition $|R_K| \gg \max\{|M|, |N|\}$, which is constantly fulfilled in industrial applications to drop the lower-order-terms. Since both t and f are positive constants in practice, the computational complexity of β-NLF is linear with $|R_K|$. Meanwhile, β-NLF only uses six matrices, i.e., P, Q, A, B, C, and D, along with U, I, and R_K. The whole storage cost of these data structures takes $\Theta((|U| + |I|) \times f + |R_K|))$ only. Therefore, it is highly efficient in both computation and storage.

IV. Experimental Results and Discussion

A. General Settings

1) Evaluation Protocol: For industrial applications [1]–[8], [14]–[16], [33], [40], [41], one major motivation to analyze an HiDS matrix is to perform missing data estimation. Owing to its popularity and usefulness, we adopt it as the evaluation protocol. For an LF model, its prediction accuracy is commonly measured by root mean squared error (RMSE)

Procedure 1 PROC_UPDATE

Input: $\hat{r}_{u,i}, r_{u,i}, p_{u,m}, a_{u,m}, b_{u,m}, q_{m,i}, c_{m,i}, d_{m,i}, \beta, \lambda_P, \lambda_Q$

Operation	Complexity
if $\beta = 0$	-
$a_{u,m} = a_{u,m} + r_{u,i} q_{m,i} / \hat{r}_{u,i}^2$	$\Theta(1)$
$b_{u,m} = b_{u,m} + q_{m,i} / \hat{r}_{u,i} + \lambda_P p_{u,m}$	$\Theta(1)$
$c_{m,i} = c_{m,i} + r_{u,i} p_{u,m} / \hat{r}_{u,i}^2$	$\Theta(1)$
$d_{m,i} = d_{m,i} + p_{u,m} / \hat{r}_{u,i} + \lambda_Q q_{m,i}$	$\Theta(1)$
else if $\beta = 1$	-
$a_{u,m} = a_{u,m} + r_{u,i} q_{m,i} / \hat{r}_{u,i}$	$\Theta(1)$
$b_{u,m} = b_{u,m} + q_{m,i} + \lambda_P p_{u,m}$	$\Theta(1)$
$c_{m,i} = c_{m,i} + r_{u,i} p_{u,m} / \hat{r}_{u,i}$	$\Theta(1)$
$d_{m,i} = d_{m,i} + p_{u,m} + \lambda_Q q_{m,i}$	$\Theta(1)$
else	-
$a_{u,m} = a_{u,m} + q_{m,i} r_{u,i} \hat{r}_{u,i}^{\beta-2}$	$\Theta(1)$
$b_{u,m} = b_{u,m} + q_{m,i} \hat{r}_{u,i}^{\beta-1} + \lambda_P p_{u,m}$	$\Theta(1)$
$c_{m,i} = c_{m,i} + p_{u,m} r_{u,i} \hat{r}_{u,i}^{\beta-2}$	$\Theta(1)$
$d_{m,i} = d_{m,i} + p_{u,m} \hat{r}_{u,i}^{\beta-1} + \lambda_Q q_{m,i}$	$\Theta(1)$
end if	-

Output: updated $a_{u,m}, b_{u,m}, c_{m,i}, d_{m,i}$

and mean absolute error (MAE)

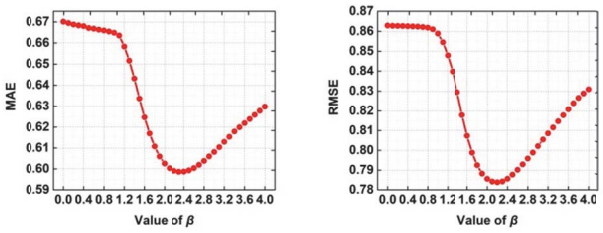
$$\text{RMSE} = \sqrt{\left(\sum_{r_{v,j} \in R_V} \left(r_{v,j} - \hat{r}_{v,j}\right)^2\right) \Big/ |R_V|}$$
$$\text{MAE} = \left(\sum_{r_{v,j} \in R_V} \left|r_{v,j} - \hat{r}_{v,j}\right|_{abs}\right) \Big/ |R_V| \tag{27}$$

where R_V denotes the validation set and naturally $R_K \cap R_V = \phi$, $\hat{r}_{v,j}$ denotes the prediction for the testing instance $r_{v,j} \in R_V$, $|\cdot|$ calculates the cardinality of a given set, 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 have recorded their iteration count at convergence and time cost per iteration. All experiments are conducted on a tablet with a 2.6-GHz i7 CPU and 128-GB RAM, and implemented in JAVA SE 7U60.

2) Datasets: Four HiDS matrices are involved in our tests, which are real datasets collected by industrial applications.

1) *D1 (MovieLens 20M):* It is collected by the MovieLens system [31] maintained by the GroupLens research team. It has 20 000 263 entries in [0.5, 5], from 138 493 users on 26 744 movies. Its data density is 0.54% only.
2) *D2 (Douban):* It is collected from the Chinese largest online book, movie and music database Douban [42]. It includes 16 830 839 ratings in the scale of [1] and [5] from 129 490 users on 58 541 items. Its density is 0.22% only.
3) *D3 (Dating Agency):* It is collected by an online dating Website LibimSeTi [43], with 17 359 346 known entries in the range of [1] and [10], from 135 359 users on 168 791 profiles. Its data density is 0.076% only.



(a) (b)

Fig. 1. β-NLF's prediction error for missing data in HiDS matrices as β varies. (a) MAE on D1. (b) RMSE on D1. Note that similar situations are also encountered on D2–D4.

4) *D4 (Extended Epinion):* It is collected by Trustlet Website [44], with 13 668 320 known entries in the range of [1] and [5], from 120 492 users on 755 760 articles. Its data density is 0.015% only.

All datasets are: 1) high-dimensional; 2) extremely sparse; and 3) collected by industrial applications currently in use. Hence, results on them are highly representative and useful.

The known entry set of each HiDS matrix is randomly split into five equally sized, disjoint subsets. In all experiments, we adopt the 80%–20% train-test settings and fivefold cross-validations, i.e., each time we select four subsets as R_K to train a model predicting the remaining one subset as R_V. This process is sequentially repeated for five times to obtain the final results. The training process of a tested model terminates if: 1) the number of consumed iterations reaches a preset threshold, i.e., 1000 or 2) the model converges, i.e., the error difference between two consecutive iterations is smaller than 10^{-5}. Note that such early stopping settings can improve the generality of a resultant model, thereby achieving fair comparisons.

3) Model Settings: As discussed in Section IV-D, NLF is only a special case of β-NLF with $\beta = 2$. Actually, β-NLF covers all possible variations of NLF models relying on β-divergence and β-SLF-NMU. To obtain objective and precise results, we adopt the following settings.

1) Fixing the regularization coefficients $\lambda_P = \lambda_Q = 0.05$ according to the empirical studies in [7].
2) Initializing P and Q with the same randomly generated and non-negative arrays as β varies, thus eliminating the impact caused by random guesses shown in [5]–[8], [14]–[16], and [33].
3) Repeating each set of experiments for ten times and taking the average of the experimental outputs as their final results.

B. Effects of β

In this set of experiments, we validate the effects of β, which is a key parameter deciding β-NLF's performance. The tested range of β is [0,4] (by noting that a β-NLF model suffers performance degeneration when $\beta > 4$ according to our experiences) and its step size is 0.1. We have recorded β-NLF's RMSE, MAE, and iteration count when the algorithm is converged, and time cost per iteration as β varies.

1) Prediction Accuracy for Missing Data: Fig. 1 depicts β-NLF's prediction error as β varies. From it, we have the following findings.

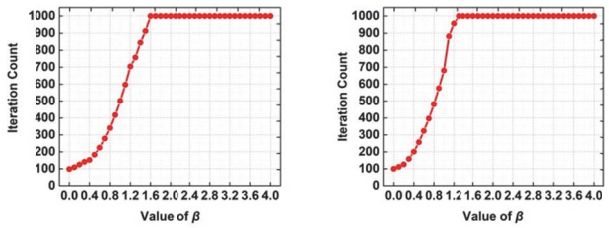
TABLE I
PREDICTION ERROR RELYING ON β

Dataset	Metric	*O. β	**Err. O. β	***Err. β=2	Gap
D1	MAE	2.3	0.5986	0.6027	0.68%
	RMSE	2.2	0.7838	0.7856	0.23%
D2	MAE	2.1	0.5579	0.5583	0.07%
	RMSE	1.9	0.7094	0.7098	0.06%
D3	MAE	1.7	1.2767	1.2939	1.33%
	RMSE	1.5	1.8367	1.8941	3.03%
D4	MAE	2.4	0.3045	0.3102	1.84%
	RMSE	1.7	0.5146	0.5273	2.41%

*Optimal value of β on each testing case.
**β-NLF's prediction error with the optimal value of β on each testing case.
***β-NLF's prediction error with β=2 on each testing case.

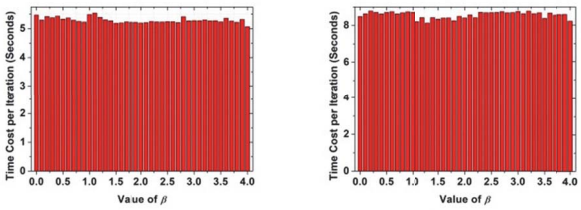
1) β-NLF's prediction accuracy for missing data in HiDS matrices is closely connected with the value of β. As depicted in Fig. 1, the prediction error of β-NLF varies in β. Meanwhile, although this connection is data-dependent, we observe a fixed pattern, i.e., there is always an optimal value of β such that it enables β-NLF to achieve the lowest RMSE/MAE on each experimental dataset. As we expect, the optimal value of β is not two (which corresponds to the frequently adopted Euclidean distance) on all testing cases as summarized in Table I. As β deviates from this optimal value, an GNLF model suffers loss of accuracy, as shown in Fig. 1.
2) As discussed in the previous sections, by making $\beta = 2$ we turn β-NLF into NLF [7]. However, as shown in Fig. 1, β-NLF never achieves the lowest prediction error on all eight testing cases with $\beta = 2$. Table I summarizes the difference in β-NLF's prediction error relying on β. As depicted in Table I, on D1 and D2, this difference can be small, and the optimal value of β is very close to two. On the other hand, on D3 and D4, this accuracy difference is more significant and the optimal value of β is far from two. This phenomenon demonstrates that the frequently adopted Euclidean distance may not be the best choice for β-NLF to achieve the highest accuracy for missing data prediction.
3) On the other hand, from Fig. 1 we also see that when $\beta \in \{0, 1\}$, β-NLF generally cannot achieve high accuracy for missing data prediction. As discussed in the previous sections, the feature update rule for β-NLF with $\beta \in \{0, 1\}$ is far different from the cases where $\beta \notin \{0, 1\}$. Hence, this phenomenon indicates that when addressing the task of missing data estimation, we should adopt a β-NLF model with $\beta \notin \{0, 1\}$ for achieving the highest prediction accuracy. Nonetheless, as discussed in previous research [5], [6], [14]–[18], a cost function with Itakura–Saito divergence or Kullback–Leibler divergence (corresponding to the case of $\beta \in \{0, 1\}$ in β-NLF, respectively) may enable an LF model's excellent performance in addressing other pattern analysis tasks like community detection [11], [12]. Therefore, it is important to consider a β-NLF model with $\beta \in \{0, 1\}$ when addressing such tasks.

2) Convergence Rate: Fig. 2 depicts β-NLF's iteration count at convergence as β varies on our experimental datasets.



(a) (b)

Fig. 2. β-NLF's iteration count to converge as β varies. (a) With MAE on D1. (b) With RMSE on D1. Note that similar situations are also encountered on D2–D4.



(a) (b)

Fig. 3. β-NLF's time cost per iteration as β varies. (a) D1. (b) D2. Note that similar situations are also encountered on D3 and D4.

From Fig. 2, we see that as β increases, β-NLF tends to consume more iterations to converge. By combining Figs. 1 and 2 along with Table I, we see that with the optimal value of β, β-NLF always consumes the preset threshold of training iterations, i.e., 1000, to converge at an optimum of (6), which is not necessarily global. The meaning of this phenomenon is twofold.

1) When β is smaller than the optimal value, a β-NLF model tends to be trapped by some saddle points of the objective function, making the resultant model not accurate enough to address the task of missing data estimation.
2) As β increases over the optimal value, the convergence rate of the resultant model becomes slow, making it hard to achieve an optimum of the objective function.

3) In Time Cost Per Iteration: Fig. 3 depicts β-NLF's time cost per iteration as β varies on our experimental datasets. From Fig. 3, we see that variations of β hardly affect β-NLF's time cost per iteration.

1) With $\beta \notin \{0, 1\}$, β-NLF's time cost per iteration appears quite stable on each dataset. From (24), we see that in this situation, β-NLF tends to raise each $\hat{r}_{u,i}$ to the power of β. In an industrial programming language like JAVA in our implementation, such an operation costs nearly the same constant time. Hence, it is reasonable that β-NLF achieves steady time cost per iteration when $\beta \notin \{0, 1\}$.
2) When $\beta \notin \{0, 1\}$, β-NLF's time cost per iteration is very close to that with $\beta \in \{0, 1\}$. From Sections III-B and III-C, we see that β-NLF's objective function is very different from that with $\beta \notin \{0, 1\}$. However, by comparing (12) and (18) with (24), we find that the β-SLF-NMU scheme with $\beta \in \{0, 1\}$ also relies on simple operations with respect to $\hat{r}_{u,i}$ and $r_{u,i}$, similar

TABLE II
COMPARED MODELS IN EXPERIMENTS

No.	β	Description
M1	0	β-NLF relying on the Itakura-Saito divergence.
M2	1	β-NLF relying on the Kullback-Leibler divergence.
M3	2	Original NLF model proposed in [7].
M4	O. β in Table I	β-NLF with the optimal value of β on each dataset.
M5	WNMF	Weighted non-negative matrix factorization-based model recommender [17].
M6	AutoRec	A neural network-based approach to recommender systems under the autoencoder framework [45].

to the case with $\beta \notin \{0, 1\}$. Hence, it is reasonable that β-NLF's time cost per iteration keeps stable as β varies.

3) However, note that as β changes, the update rule of desired LFs can depend on different operation, e.g., raising an arbitrary real number to the power of different values of β. Such difference can introduce some cost difference in the computational cost of a resultant β-NLF model. Hence, from Fig. 3 we also observe some slight fluctuations of time cost per iteration as β changes in β-NLF.

C. Comparison With State-of-the-Art LF Models

In this set of experiments, we compare the performance of β-NLF models with recent LF modes. Four β-NLF models with typical values of β, i.e., $\beta = 0$, 1, 2 and the optimal value on each dataset, are first involved as M1–M4 as summarized in Table II. Note that when $\beta = 2$, we actually achieve the NLF mode proposed in [7], which is a most popular model adopted in various non-negative LF analysis [33], [36]–[39]. Moreover, we include WNMF [17] and AutoRec [45] models into our comparison. The former [17] is a classical model for non-negative LF analysis based on NMF algorithm as introduced in Section I. We include it into our experiments as the baseline to show the virtues of a β-NLF model. The latter [45] is a recent model designed for recommender systems based on the principle of an autoencoder. We adopt it as a rival model to see whether or not β-NLF can achieve performance gain when compared with the newest recommenders. Note that WNMF and AutoRec are, respectively, marked as M5 and M6, as shown in Table II.

We set the latent dimension $f = \{20, 40, 80, 120, 160, 300, 500\}$. The prediction error of compared models as f increases is depicted in Fig. 4. Their time cost per iteration as f increases is depicted in Fig. 5. From them, we have the following findings.

1) As f increases, all models' prediction accuracy for missing data in an HiDS matrix increases. However, M4, i.e., β-NLF with the optimal value of β on each data set, keeps outperforming its peers in terms of prediction accuracy. For instance, as shown in Fig. 4(g), with $f = 20$ on D4, M1–M6's MAE is 0.3435, 0.3422, 0.3102, 0.3045, 0.3927, and 0.3562, respectively; M4's MAE is 11.35%, 11.02%, 1.84%, 22.45%, and 14.51% lower than that of M1–M3, M5, and M6. In addition, as shown in Fig. 4(h), with $f = 160$, M1–M6's RMSE is 0.5294, 0.5138, 0.5148, 0.4925, 0.6188, and 0.5301, respectively; M1–M3, M5, and M6's RMSE is 6.97%, 4.14%, 4.33%, 20.41%, and 7.09% higher than that of M4. The situation is similar for other cases on D1–D3, as shown in Fig. 4(a)–(f). This phenomenon indicates that with optimal β, a β-NLF model keeps its advantage in prediction accuracy for missing data of HiDS matrix as f increases.

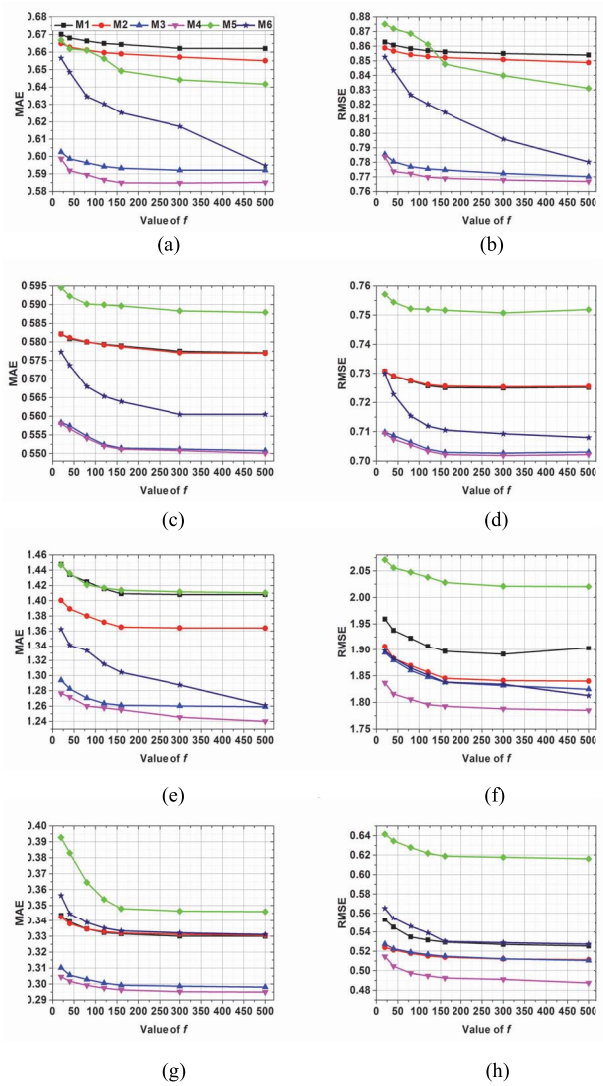


Fig. 4. Prediction error of compared models as f increases. Note that all panels share the same legend in panel (a). (a) MAE on D1. (b) RMSE on D1. (c) MAE on D2. (d) RMSE on D2. (e) MAE on D3. (f) RMSE on D3. (g) MAE on D4. (h) RMSE on D4.

Note that as mentioned in prior research [45], M6, i.e., the AutoRec model, is able to achieve high prediction accuracy for missing data. However, we also note that M6 requires setting the hidden neuron count at 500 to achieve the highest prediction accuracy. Note that the hidden neuron count actually plays the role of LF dimension as analyzed in [6]. As shown in Fig. 4, as f increases to 500, M6's prediction accuracy for missing data can outperform M3 in some testing cases, as shown in Fig. 4(e) and (f). However, it cannot achieve generalized error as low as M4 does.

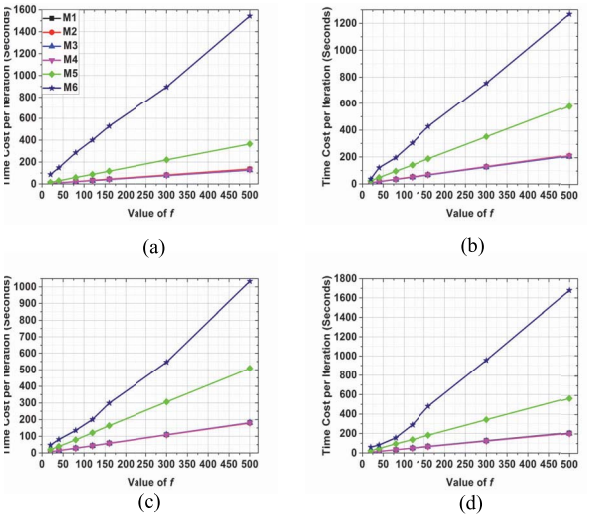


Fig. 5. Compared models' time cost per iteration as f increases. Note that all panels share the same legend in panel (a). (a) D1. (b) D2. (c) D3. (d) D4.

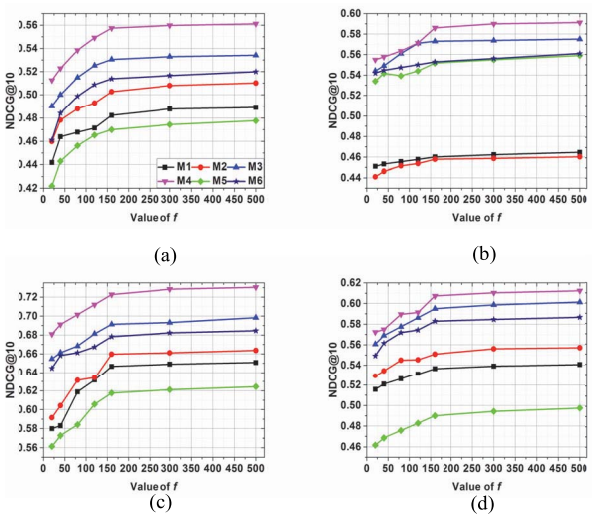


Fig. 6. Compared models' NDCG as f increases. Note that all panels share the same legend in panel (a). (a) D1. (b) D2. (c) D3. (d) D4.

1) As f increases, all compared model's time cost per iteration increases, but β-NLF has advantage in terms of computational efficiency. Moreover, as shown in Fig. 5, we see that all involved models' time cost per iteration is strictly linear with f except M6. As given in Section III-E, β-NLF's (including M1–M4) time cost per iteration is $\Theta(|\mathrm{R}_K|\times f)$ and the time costs are highly close and their cost curves overlap, which are highly consistent with the results given in Fig. 5. It means β-NLF's time cost per iteration is not sensitive to the value of β. Meanwhile, M5's time cost is about three times of β-NLF. As indicated in [17], M5 adopts a binary weight matrix for adapting the standard NMU proposed in [18] to an HiDS matrix. Hence, its training is taken on a full matrix filled with lots of zeroes, which consumes much more time than the β-NLF scheme proposed in this paper. For M6, its time cost per iteration is the highest among its peers, and grows nonlinear with respect to f, as shown in Fig. 5. As indicated in [45], M6 relies on many operations about matrix manipulation and non-linear activation functions. Without GPU-based acceleration as in our experiment, it is indeed expensive to train a multilayered neural network-based LF model like M6 on a large-scale HiDS matrix.
2) As indicated in recent work, the ranking performance of a recommender is highly important for real applications [46]–[51]. Hence, we further adopt the normalized discounted cumulative gain (NDCG) as the evaluation metric to validate M1–M6's ranking performance. In this test, we use the popular leave-one-out settings [46]–[51].
   a) The latest item touched by each user is put into a candidate set.
   b) The corresponding rating is removed from the training data.
   c) Fifty items are selected at random from the candidate set to form the validation set.
   d) An involved recommender's ranking performance is validated by ranking items in the validation set according to its rating predictions using NDCG.

   For each user, we keep top ten items corresponding to the highest rating predictions generated by a recommender to compute its final NDCG, which is consistent with the commonly adopted settings [46]–[51]. Note the high NDCG stands for strong ranking ability of a recommender, and vice versa.

The NDCG of compared models as f increases is depicted in Fig. 6. From it, we see that although all compared model's NDCG increases as f increases, M4, i.e., a β-NLF model with the optimal β, always achieves the highest NDCG on all datasets, outperforming its peers significantly. For instance, with $f = 20$ on D3, the NDCG of M4 is 0.6809. Compared with 0.5799 by M1, 0.5914 by M2, 0.6548 by M3, 0.5614 by M5, and 0.6447 by M6, the improvement by M4 is 14.83%, 13.14%, 3.83%, 17.55%, and 5.31%, respectively. Similar situations are also found on the other datasets, as in Fig. 6.

D. Summaries

Based on the above experimental results, we summarize the following.

1) When addressing the task of missing data estimation, β-NLF's prediction accuracy is highly sensitive to β. By carefully tuning β to its optimal value, β-NLF's prediction accuracy for missing data can be certainly improved and sometimes significant. It can also achieve obvious accuracy gain when compared with a state-of-the-art neural network-based LF model.s
2) β-NLF's convergence rate is also sensitive to β. On an HiDS matrix, when β is smaller than its optimal value, a β-NLF model tends to be trapped by some bad local optimum; when β is larger than its optimal value, a β-NLF model can hardly achieve a desired solution, either.
3) β-NLF's time cost per iteration is insensitive to β, but linear with f. Its computational efficiency is high when

compared with state-of-the-art models in a bare machine environment. Meanwhile, β-NLF's prediction accuracy for missing data always increases with f.

4) When addressing ranking problems, β-NLF also achieves the highest NDCG among its peers.

Hence, we conclude that with properly tuned β, a β-NLF model can well handle the task of missing data estimation or collaborative ranking with high computational efficiency.

V. Discussion

Based on the experimental results, we see the effective of a β-NLF model on HiDS matrices. In this section, we discuss the following issues.

1) *Self-Adaptation of β:* For a β-NLF model, the value of β is vital in deciding its performance. Currently, we can tune it on a probe set to achieve a relatively accurate β-NLF model on a given dataset. However, the issue of making β self-adaptive to achieve a most accurate β-NLF model remains open. According to prior research [60], an evolutionary-computation-based algorithm like particle swarm optimization can be useful in implementing efficient adaptations of hyper-parameters like β in our scene. Moreover, Lu *et al.* [59] proposed a nonmaximum-likelihood estimator called score matching (SM) to select β in an NMF model designed for full matrices. Based on these prior studies, it seems highly promising to implement self-adaptation of β. Nonetheless, great efforts are needed for redirecting such strategies to a β-NLF model defined on an HiDS matrix.
2) *Training Acceleration:* As shown in the experimental results, a β-NLF model tends to consume many iterations to converge with an optimal β. Is it possible to further accelerate the training process of a β-NLF model, making it consume less iterations to converge? As presented in Section III, a β-NLF model relies on the β-SLF-NMU scheme, which is a generalized form of the SLF-NMU scheme. As shown in prior study [61], an SLF-NMU scheme can be accelerated by a generalized momentum method. Hence, it is probable to make a β-NLF model converge faster with it. Further efforts are required to validate the compatibility between a β-SLF-NMU scheme and a generalized momentum method.
3) *A More Generalized NLF Model Based on α-β-Divergence:* As discussed in [58], given two complete matrices, β-divergence can be further generalized into α-β-divergence to measure their differences. Based on it, Cichoki *et al.* [58] proposed to build a series of robust NMF models for extracting non-negative LFs from a complete matrix. Given that α-β-divergence covers the cases of β-divergence, it is promising to achieve more accurate NLF models based on it. To do so, it is desired to develop more generalized learning scheme for α-β-divergence-based NLF model following the principle of SLF-NMU to handle an HiDS matrix.

On the other hand, α-β-divergence has two hyper parameters, i.e., α and β, for tuning the characteristics of an objective function. As discussed in Section IV-B, β decides the performance of a β-NLF model, making it vital to pretune its value on a probe dataset for ensuring high prediction accuracy of a resultant model. From this point of view, with the more generalized α-β-divergence, it becomes more difficult to obtain optimal hyper parameters to achieve the most accurate model. Consequently, to make hyper parameters self-adaptive becomes vital in such a model. We plan to address such issues in the future.

VI. Conclusion

An NLF model adopts an SLU-NMU scheme for extracting NLFs from HiDS matrices efficiently in both storage and computation. However, current NLF models mostly adopt Euclidean distance as their objective function, which is only a special case of a β-divergence function. Since the objective function has dominant effects on an NLF model's performance, it is of great importance to study whether a more generalized NLF model with a β-divergence function-based optimization objective can outperform an NLF model, as well as validating its performance on HiDS matrices arising from industrial recommender systems [2]–[7], [48]–[55], [62].

For such purposes, this paper proposes a generalized NLF (β-NLF) model whose learning objective is designed according to a β-divergence function. Corresponding to each case of its objective function, a β-SLF-NMU scheme is for the first time deduced by analyzing the KKT conditions of its constraints for training desired LFs under non-negativity constraints. With an appropriately designed algorithm, β-NLF is able to achieve high efficiency in both computation and storage. Moreover, with properly tuned β, it outperforms several state-of-the-art LF models in terms of prediction accuracy and ranking ability.

Future studies include: 1) implementing self-adaptation of β; 2) implementing a more efficient model with the principle of accelerated optimization algorithms like a generalized momentum method; and 3) developing more generalized NLF models based on α-β-divergence.

References

[1] 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.
[2] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, "Recommender system application developments: A survey," *Decis. Support Syst.*, vol. 74, pp. 12–32, Jun. 2015.
[3] M. Erdt, A. Fernández, and C. Rensing, "Evaluating recommender systems for technology enhanced learning: A quantitative survey," *IEEE Trans. Learn. Technol.*, vol. 8, no. 4, pp. 326–344, Oct./Dec. 2015.
[4] Z. Yang, B. Wu, K. Zheng, X. Wang, and L. Lei, "A survey of collaborative filtering-based recommender systems for mobile Internet applications," *IEEE Access*, vol. 4, pp. 3273–3287, 2016.
[5] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Comput.*, vol. 42, no. 8, pp. 30–37, Aug. 2009.
[6] G. Takács, I. Pilászy, B. Németh, and D. Tikky, "Scalable collaborative filtering approaches for large recommender systems," *J. Mach. Learn. Res.*, vol. 10, pp. 623–656, Mar. 2009.
[7] 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. Informat.*, vol. 10, no. 2, pp. 1273–1284, May 2014.

[8] 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.

[9] C. X. Ou and R. M. Davison, "Technical opinion: Why eBay lost to TaoBao in China: The glocal advantage," *Commun. ACM*, vol. 52, no. 1, pp. 145–148, 2009.

[10] C. L. Liu, J. Liu, and Z. Z. Jiang, "A multiobjective evolutionary algorithm based on similarity for community detection from signed social networks," *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2274–2287, Dec. 2014.

[11] L. Yang, X. C. Cao, D. Jin, X. Wang, and D. Meng, "A unified semi-supervised community detection framework using latent space graph regularization," *IEEE Trans. Cybern.*, vol. 45, no. 11, pp. 2585–2598, Nov. 2015.

[12] P. Yang, Q. Zhu, and B. Huang, "Spectral clustering with density sensitive similarity function," *Knowl. Based Syst.*, vol. 24, no. 5, pp. 621–628, 2011.

[13] S. Hui and P. N. Suganthan, "Ensemble and arithmetic recombination-based speciation differential evolution for multimodal optimization," *IEEE Trans. Cybern.*, vol. 46, no. 1, pp. 64–74, Jan. 2016.

[14] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 20, 2008, pp. 1257–1264.

[15] Q. Zhao, L. Zhang, and A. Cichocki, "Bayesian CP factorization of incomplete tensors with automatic rank determination," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1751–1763, Sep. 2015.

[16] D. Rafailidis and A. Nanopoulos, "Modeling users preference dynamics and side information in recommender systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 6, pp. 782–792, Jun. 2016.

[17] S. Zhang, W. Wang, J. Ford, and F. Makedon, "Learning from incomplete ratings using non-negative matrix factorization," in *Proc. SIAM Int. Conf. Data Min.*, Bethesda, MD, USA, 2006, pp. 549–553.

[18] D. D. Lee and S. H. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788–791, Oct. 1999.

[19] O. Nicoletti, F. de la Pena, R. K. Leary, D. J. Holland, C. Ducati, and P. A. Midgley, "Three-dimensional imaging of localized surface plasmon resonances of metal nanoparticles," *Nature*, vol. 502, pp. 80–84, Oct. 2013.

[20] C. H. Q. Ding, L. Tao, and M. I. Jordan, "Convex and semi-nonnegative matrix factorizations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 1, pp. 45–55, Jan. 2010.

[21] I. Meganem, Y. Deville, S. Hosseini, P. Deliot, and X. Briottet, "Linear-quadratic blind source separation using NMF to unmix urban hyperspectral images," *IEEE Trans. Signal Process.*, vol. 62, no. 7, pp. 1822–1833, Apr. 2014.

[22] J. P. Brunet, P. Tamayo, T. R. Golub, and J. P. Mesirov, "Metagenes and molecular pattern discovery using matrix factorization," *Proc. Nat. Acad. Sci. USA*, vol. 101, no. 12, pp. 4164–4169, 2003.

[23] J. J. Pan, S. J. Pan, Y. Jie, L. M. Ni, and Y. Qiang, "Tracking mobile users in wireless networks via semi-supervised co-localization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 3, pp. 587–600, Mar. 2012.

[24] C. Fidel, C. Victor, F. Diego, and F. Vreixo, "Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems," *ACM Trans. Web*, vol. 5, no. 1, pp. 1–33, 2011.

[25] P. Paatero and U. Tapper, "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values," *Environmetrics*, vol. 5, no. 2, pp. 111–126, 1994.

[26] C.-J. Lin, "Projected gradient methods for nonnegative matrix factorization," *Neural Comput.*, vol. 19, no. 10, pp. 2756–2779, Oct. 2007.

[27] P. O. Hoyer, "Non-negative matrix factorization with sparseness constraints," *J. Mach. Learn. Res.*, vol. 5, pp. 1457–1469, Jan. 2004.

[28] H. Kim and H. Park, "Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis," *Bioinformatics*, vol. 23, no. 12, pp. 1495–1502, 2007.

[29] W. W. Wang, A. Cichocki, and J. A. Chambers, "A multiplicative algorithm for convolutive non-negative matrix factorization based on squared Euclidean distance," *IEEE Trans. Signal Process.*, vol. 57, no. 7, pp. 2858–2864, Jul. 2009.

[30] Y. Xu, W. Yin, Z. Wen, and Y. Zhang, "An alternating direction algorithm for matrix completion with nonnegative factors," *Front. Math. China*, vol. 7, no. 2, pp. 365–384, 2012.

[31] 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, 1997.

[32] I. S. Dhillon and S. Sra, "Generalized nonnegative matrix approximations with Bregman divergences," in *Proc. 18th Int. Conf. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2005, pp. 283–290.

[33] 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.

[34] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2009.

[35] M. M. Deza and E. Deza, *Encyclopedia of Distances*. Heidelberg, Germany: Springer-Verlag, 2009.

[36] A. Che, P. Wu, F. Chu, and M. Zhou, "Improved quantum-inspired evolutionary algorithm for large-size lane reservation," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 12, pp. 1535–1548, Dec. 2015.

[37] Q. Kang, J. Wang, M. Zhou, and A. C. Ammari, "Centralized charging strategy and scheduling algorithm for electric vehicles under a battery swapping scenario," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 3, pp. 659–669, Mar. 2015.

[38] P. Wu, A. Che, F. Chu, and M. Zhou, "An improved exact ε-constraint and cut-and-solve combined method for biobjective robust lane reservation," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 3, pp. 1479–1492, Jun. 2015.

[39] L. Feng and B. Bhanu, "Semantic concept co-occurrence patterns for image annotation and retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 4, pp. 785–799, Apr. 2016.

[40] M. Li and Z. Yin, "Debugging object tracking by a recommender system with correction propagation," *IEEE Trans. Big Data*, vol. 3, no. 4, pp. 429–442, Dec. 2017.

[41] L. Song, C. Tekin, and M. V. D. Schaar, "Online learning in large-scale contextual recommender systems," *IEEE Trans. Services Comput.*, vol. 9, no. 3, pp. 433–445, May/Jun. 2017.

[42] H. Ma, I. King, and M. R. Lyu, "Learning to recommend with social trust ensemble," in *Proc. 32nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Boston, MA, USA, 2009, pp. 203–210.

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

[44] P. Massa and P. Avesani, "Trust-aware recommender systems," in *Proc. 1st ACM Conf. Recommender Syst.*, Minneapolis, MN, USA, 2007, pp. 17–24.

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

[46] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua, "Fast matrix factorization for online recommendation with implicit feedback," in *Proc. 39th ACM SIGIR Int. Conf. Res. Develop. Inf. Retrieval*, Pisa, Italy, 2016, pp. 549–558.

[47] I. Bayer, X. He, B. Kanagal, and S. Rendle, "A generic coordinate descent framework for learning from implicit feedback," in *Proc. 26th Int. Conf. World Wide Web*, Perth, WA, Australia, 2017, pp. 1341–1350.

[48] A.-M. Elkahky, Y. Song, and X. He, "A multi-view deep learning approach for cross domain user modeling in recommendation systems," in *Proc. 24th Int. Conf. World Wide Web*, Florence, Italy, 2015, pp. 278–288.

[49] X. He, T. Chen T, M.-Y. Kan, and X. Chen, "TriRank: Review-aware explainable recommendation by modeling aspects," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manag.*, Melbourne, VIC, Australia, 2015, pp. 1661–1670.

[50] F. Ricci, L. Rokach, and B. Shapira, "Recommender systems: Introduction and challenges," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, and B. Shapira, Eds. Boston, MA, USA: Springer, 2015, pp. 1–34.

[51] B. Suhrid and C. Sumit, "Collaborative ranking," in *Proc. 15th ACM Int. Conf. Web Search Data Min.*, Seattle, WA, USA, 2012, pp. 143–152.

[52] J. Castro, J. Lu, G. Zhang, Y. 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.

[53] 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. Automatica Sinica*, vol. 6, no. 2, pp. 584–595, Mar. 2019.

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

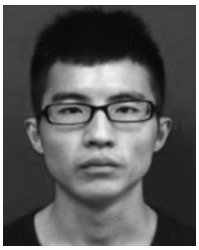
[55] Q. Zhao, C. Wang, P. Wang, M. Zhou, and C. Jiang, "A novel method on information recommendation by hybrid similarity," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 3, pp. 448–459, Mar. 2018.

[56] R. Hennequin, B. David, and R. Badeau, "Beta-divergence as a subclass of Bregman divergence," *IEEE Signal Process. Lett.*, vol. 18, no. 2, pp. 83–86, Feb. 2011.
[57] D. L. Sun and C. Févotte, "Alternating direction method of multipliers for non-negative matrix factorization with the beta-divergence," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2014, pp. 6201–6205.
[58] A. Cichocki, S. Cruces, and S.-I. Amari, "Generalized alpha-beta divergences and their application to robust nonnegative matrix factorization," *Entropy*, vol. 13, no. 1, pp. 134–170, 2011.
[59] Z. Lu, Z. Yang, and E. Oja, "Selecting β-divergence for nonnegative matrix factorization by score matching," in *Proc. Int. Conf. Artif. Neural Netw.*, Lausanne, Switzerland, 2012, pp. 419–426.
[60] P. R. Lorenzo, J. Nalepa, M. Kawulok, L. S. Ramos, and J. R. Paster, "Particle swarm optimization for hyper-parameter selection in deep neural networks," in *Proc. ACM Int. Conf. Genet. Evol. Comput.*, 2017, pp. 481–488.
[61] 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.
[62] 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.

Xin Luo (M'14–SM'17) received the B.S. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2005, and the Ph.D. degree from Beihang University, Beijing, China, in 2011, both in computer science.

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. He is currently also a Distinguished Professor of Computer Science with the Dongguan University of Technology, Dongguan, China. His current research interests include big data analysis and intelligent control. He has published over 100 papers (including over 30 IEEE TRANSACTIONS papers) in the above areas.

Dr. Luo 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, the Advanced Support of the Pioneer Hundred Talents Program of Chinese Academy of Sciences in 2018, and the Outstanding Associate Editor reward of the IEEE ACCESS in 2018. He is currently serving as an Associate Editor for the IEEE/CAA JOURNAL OF AUTOMATICA SINICA, the IEEE ACCESS, and *Neurocomputing*. He has also served as a Program Committee Member for over 20 international conferences.



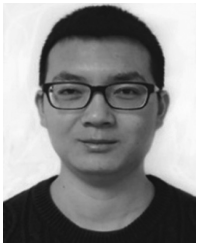
Ye Yuan received the B.S. degree in electronic information engineering and the M.S. degree in signal processing from the University of Electronic Science and Technology, Chengdu, China, in 2010 and 2013, respectively. He is currently pursuing the Ph.D. degree in computer science with the University of Chinese Academy of Sciences, Chinese Academy of Sciences, Beijing, China.

He is currently an Assistant Professor with the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China. His current research interests include data mining, recommender system, and intelligent computing.

MengChu Zhou (S'88–M'90–SM'93–F'03) received the B.S. degree in control engineering from the Nanjing University of Science and Technology, Nanjing, China, in 1983, the M.S. degree in automatic control from the Beijing Institute of Technology, Beijing, China, in 1986, and the Ph.D. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, USA, in 1990.

He joined the New Jersey Institute of Technology, Newark, NJ, USA, in 1990, where he is currently a Distinguished Professor of Electrical and Computer Engineering. He has over 800 publications, including 12 books, over 500 journal papers (over 380 in the IEEE TRANSACTIONS), 12 patents, and 29 book-chapters. His current research interests include Petri nets, intelligent automation, Internet of Things, big data, Web services, and intelligent transportation.

Prof. Zhou was a recipient of the Humboldt Research Award for U.S. Senior Scientists from Alexander von Humboldt Foundation, the Franklin V. Taylor Memorial Award, and the Norbert Wiener Award from the IEEE Systems, Man and Cybernetics Society for which he serves as the VP for Conferences and Meetings. He is the founding Editor of the IEEE Press Book Series on Systems Science and Engineering, and the Editor-in-Chief of the IEEE/CAA JOURNAL OF AUTOMATICA SINICA. He was the General Chair of 2008 IEEE Conference on Automation Science and Engineering, the General Co-Chair of 2003 and 2019 IEEE International Conference on System, Man and Cybernetics (SMC), the Founding General Co-Chair of 2004 IEEE International Conference on Networking, Sensing and Control, and the General Chair of 2006 IEEE International Conference on Networking, Sensing and Control. He was the Program Chair of the 2010 IEEE International Conference on Mechatronics and Automation, the 1998 and 2001 IEEE International Conference on SMC, and the 1997 IEEE International Conference on Emerging Technologies and Factory Automation. He organized and chaired over 150 technical sessions and served on program committees for many conferences. He has led or participated in over 50 research and education projects with total budget over $ 12M, funded by National Science Foundation, Department of Defense, NIST, New Jersey Science and Technology Commission, and industry. He is a Life Member of the Chinese Association for Science and Technology, USA, and served as its President in 1999. He is a fellow of the International Federation of Automatic Control, the American Association for the Advancement of Science, and the Chinese Association of Automation.



Zhigang Liu received the B.S. degree in geographical information system from the Chongqing University of Posts and Telecommunications, Chongqing, China, in 2013. He is currently pursuing the M.E. degree in computer technology with Chongqing University, Chongqing.

He is an Exchange Scholar with the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing. His current research interests include big data analysis and algorithm design.

Mingsheng Shang received the B.E. degree in management from Sichuan Normal University, Chengdu, China, in 1995, and the Ph.D. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, in 2007.

He is currently a Professor of Computer Science and Technology with the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China. His current research interests include complex network analysis and big data applications.