This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TSMC.2018.2875452, IEEE Transactions on Systems, Man, and Cybernetics: Systems

A Fast Non-negative Latent Factor Model based on Generalized Momentum Method

Xin Luo, *Senior Member, IEEE*, Zhigang Liu, Shuai Li, *Senior Member, IEEE*, Mingsheng Shang, and Zidong Wang, *Fellow, IEEE*

***Abstract*—Non-negative latent factor (NLF) models can efficiently acquire useful knowledge from high-dimensional and sparse (HiDS) matrices filled with non-negative data. Single latent factor-dependent, non-negative and multiplicative update (SLF-NMU) is an efficient algorithm for building an NLF model on an HiDS matrix, yet it suffers slow convergence. A momentum method is frequently adopted to accelerate a learning algorithm, but it is incompatible with those implicitly adopting gradients like SLF-NMU. To build a fast NLF model, we propose a generalized momentum method compatible with SLF-NMU. With it, we further propose a single latent factor-dependent, non-negative, multiplicative and momentum-incorporated update (SLF-NM^{2}U) algorithm, thereby achieving a fast non-negative latent factor (FNLF) model. Empirical studies on six HiDS matrices from industrial application indicate that an FNLF model outperforms an NLF model in terms of both convergence rate and prediction accuracy for missing data. Hence, compared with an NLF model, an FNLF model is more practical in industrial applications.**

***Index Terms*—Big Data, Latent Factor Analysis, Non-negative Latent Factor Model, High-dimensional and Sparse Matrix, Recommender System, Missing Data Estimation.**

I. Introduction

A BIG DATA-RELATED application commonly involves a huge amount of entities, e.g., millions of users are involved in a social network service system [1-3]. In such an application, interactions among numerous entities result in high-dimensional relationships. Moreover, since each entity cannot establish complete interactions with the others, e.g., a user cannot touch all items in a recommender system [4-6], such high dimensional relationships can be very sparse. High-dimensional and sparse (HiDS) matrices, where numerous entries are unknown rather than zeroes, are frequently adopted to describe such relationships [1-10].

Although an HiDS matrix can be extremely sparse, it contains plenty of knowledge describing various patterns. For instance, a user-item rating matrix from a recommender system implicitly reflects users' preferences on items [4-6], which are highly useful for important knowledge acquisition tasks like community detection [43-49]. Great efforts have been made to extract such useful knowledge from an HiDS matrix, resulting in various big-data analysis models [1-10]. Among them, latent factor (LF) models are highly efficient [11-17, 50].

An LF model works by extracting LFs from observed data of an HiDS matrix. It maps the involved entities into a unique low-dimensional LF space, builds a cost function based on the target matrix's known entries and desired LFs, and then minimizes this cost function with regard to these LFs. The obtained LFs are treated as the entity features hidden in the observed part of an HiDS matrix, and can be applied to various data analysis tasks. For instance, in a recommender system, they form the low-rank approximation to an HiDS matrix for estimating its missing data, thereby predicting users' potential preferences [11-17, 50].

As shown in prior research [11-17, 50], current LF models are highly efficient in both computation and storage when addressing an HiDS matrix. However, most of them do not fulfill the non-negativity constraints. Note that industrial data like rating data in a recommender system [4-6] and interaction weights in protein interactome mapping [7-9] are commonly defined to be non-negative. When addressing a non-negative HiDS matrix, it is significant to constrain desired LFs to be non-negative for describing its non-negative data more precisely [18-20]. Moreover, non-negative LFs can precisely describe hidden patterns like community tendency and user preferences [1, 3, 19, 20]. Hence, it is vital to develop LF models which fulfill the non-negativity constraints.

Given a complete matrix, non-negative matrix factorization (NMF) models can acquire non-negative LFs from it [18-26]. Paatero and Tappe [22] adopt alternating least squares (ALS) to build an NMF model. It truncates negative LFs to be zeroes during the ALS-based training process to maintain their

This research is supported in part by the National Key Research and Development Program of China under grant 2017YFC0804002, in part by the National Natural Science Foundation of China under grants 61772493 and 91646114, in part by the Chongqing Research Program of Technology Innovation and Application under grants cstc2017rgzn-zdyfX0020, cstc2017zdcy-zdyf0554 and cstc2017rgzn-zdyf0118, 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 cx2018011, and in part by the Pioneer Hundred Talents Program of Chinese Academy of Sciences. X. Luo and Z. Liu contributed equally to this work and should be considered co-first authors. Z. Wang and X. Luo are the corresponding authors.

X. Luo is with the School of Computer Science and Network Security, Dongguan University of Technology, Dongguan, Guangdong 523808, China (e-mail: luoxin21@dgut.edu.cn).

Z. Liu and M. Shang 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, China (e-mail: {liuzhigang, msshang}@cigit.ac.cn).

S. Li is with the Department of Computing, Hong Kong Polytechnic University, Hong Kong, HK 999077, China (e-mail: shuaili@polyu.edu.hk).

Z. Wang is with the Department of Computer Science, Brunel University London, Uxbridge, Middlesex UB8 3PH, United Kingdom (e-mail: Zidong.Wang@brunel.ac.uk).

non-negativity. Lee and Seung [18, 21] derive the non-negative and multiplicative update (NMU) algorithm, which keeps the negativity of desired LFs if they are initially non-negative. Lin [23] proposes projected gradient decent to implement an NMF model. It also works by truncating negative LFs to be zeroes during the training process, but adopts the gradient descent as the learning algorithm. These models and their extensions [21-28] well analyze a complete matrix, but cannot address an HiDS one with numerous missing data.

For adapting existing NMF techniques to HiDS matrices, Zhang *et al*. propose a weighted NMF (WNMF) [19] model. It constructs an intermediate matrix by filling the unknown entries of the target matrix with zeroes, and then applies the NMU algorithm to it to obtain non-negative LFs. Xu *et al*. [20] propose a non-negative matrix completion (NMC) model, which adopts a full approximation to the target matrix and then applies projected ALS to it for non-negative LFs. These models can address an HiDS matrix [21-28], but are defined on its full approximation. As a consequence, they suffer unacceptably high computational and storage costs. For example, the MovieLens 20M matrix [42] has 20,000,263 instances scattering in 138,493 rows and 26,744 columns. Its data density is 0.54% only, but its entry count (including the unknown ones) is more than 3.7 billion. To manipulate its full approximation with so many entries is extremely expensive in both computation and storage. Moreover, with careful model design, such burden can be greatly alleviated [20, 30].

For performing non-negative latent factor analysis on HiDS matrices efficiently, Luo et al. [20, 30] propose the single latent factor-dependent, non-negative and multiplicative update (SLF-NMU) algorithm. It relies on an HiDS matrix's known data only, thereby alleviating the computational and storage burden greatly. Based on this algorithm, they further propose a non-negative latent factor (NLF) model. Given an HiDS matrix, NLF's computational cost is linear with its known entry count only, and its storage cost is linear with the sum of its user and item counts only [20, 30].

In spite of its low cost in both computation and storage, SLF-NMU makes an NLF model suffer low-tail convergence [20, 30], i.e., it takes many iterations to make the model converge. In industrial applications, a model's convergence rate is vital. Hence, how to further accelerate an NLF model becomes an emerging issue.

As unveiled by prior work [31, 32], a gradient descent (GD)-based algorithm's convergence rate can be significantly improved by a momentum method. A momentum method works by recording the update of involved parameters in each iteration, and deciding their next update by linearly combining the current gradient and previous update [12, 31, 32]. However, it is designed for algorithms explicitly depending on gradients. In contrast, an SLF-NMU algorithm trains non-negative LFs multiplicatively for keeping their non-negativity. It depends on gradients implicitly. Therefore, an SLF-NMU algorithm is incompatible with a standard momentum method.

Is it possible to accelerate an SLF-NMU algorithm by incorporating momentum effects into it, thereby achieving a fast non-negative latent factor (FNLF) model? To answer this question, this work proposes a generalized momentum method, which keeps the principle of a standard momentum method but implicitly depends on gradients. Based on this generalized momentum method, we propose a single latent factor-dependent, non-negative, multiplicative and momentum-incorporated update (SLF-NM^{2}U) algorithm. With it, we further achieve an FNLF model for HiDS matrices. The main contributions of this paper include:

a) A generalized momentum method which is compatible with learning algorithms implicitly depending on gradients;
b) An FNLF model, which applies a generalized momentum method with an NLF model to achieve better performance; and
c) Algorithm design and analysis for an FNLF model.

For validating the performance of an FNLF model, we have conducted empirical studies on six HiDS matrices generated by real applications. To the authors' best knowledge, such efforts have been never seen in any previous work.

The rest of this paper is organized as follows. Section II gives the preliminaries. Section III presents our methods. Section IV gives the experimental results. Section V discusses some related issues. Finally, Section VI concludes this paper.

II. Preliminaries

A. Problem Formulation

An HiDS matrix describes certain relationships among entities involved in big data-related application, which is defined as follows:

***Definition* 1:** Let M and N be two large entity sets; $R^{|M|\times|N|}$ be a matrix whose entry $t_{m,n}$ describes certain relationship between $m\in M$ and $n\in N$; Λ and Γ be known and unknown entry sets of R; R is an HiDS matrix if $|\Lambda|\ll|\Gamma|$.

An LF model tries to build a low-rank approximation to an HiDS matrix. Given R, it is defined as follows:

***Definition* 2.** Given R and Λ, an LF model builds R's rank-d approximation $\hat{R}=PQ^{\mathrm{T}}$, where $P^{|M|\times d}$ and $Q^{|N|\times d}$ are LF matrices and $d\ll\min\{|M|, |N|\}$.

Note that P and Q are defined as the LF matrices reflecting characteristics of M and N represented by Λ, and d is the dimension of the LF space. To obtain P and Q, an objective function is designed to measure the difference between R and $\hat{R}$ with respect to Λ only. With the Euclidean distance, such an objective function is formulated by [20, 30]:

$$\arg\min_{P,Q}\varepsilon(P,Q)=\frac{1}{2}\sum_{r_{m,n}\in\Lambda}\left(r_{m,n}-\sum_{k=1}^{d}p_{m,k}q_{n,k}\right)^2, \tag{1}$$

where $r_{m,n}$, $p_{m,k}$ and $q_{n,k}$ denote specified entries in R, P and Q, respectively.

Let $\hat{r}_{m,n}=\sum_{k=1}^{d}p_{m,k}q_{n,k}$, we extend objective (1) into the following form for correctly describing non-negative data:

$$\begin{aligned}&\arg\min_{P,Q}\varepsilon(P,Q)=\frac{1}{2}\sum_{r_{m,n}\in\Lambda}\left(r_{m,n}-\sum_{k=1}^{d}p_{m,k}q_{n,k}\right)^2,\\ &s.t.\quad \forall m\in M,n\in N,k\in\{1,2,...,d\}:p_{m,k}\geq 0,\ q_{n,k}\geq 0.\end{aligned} \tag{2}$$

Meanwhile, LF analysis on an HiDS matrix is ill-posed [11-17,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TSMC.2018.2875452, IEEE Transactions on Systems, Man, and Cybernetics: Systems

20], calling for regularization design. So we regularize (2) as:

$$\arg\min_{P,Q}\varepsilon(P,Q)=\frac{1}{2}\sum_{r_{m,n}\in\Lambda}\left(\left(r_{m,n}-\hat{r}_{m,n}\right)^2+\lambda_P\|P\|_F^2+\lambda_Q\|Q\|_F^2\right)$$
$$=\frac{1}{2}\sum_{r_{m,n}\in\Lambda}\left(\left(r_{m,n}-\hat{r}_{m,n}\right)^2+\lambda_P\sum_{k=1}^{d}p_{m,k}^2+\lambda_Q\sum_{k=1}^{d}q_{n,k}^2\right), \quad (3)$$
$$s.t.\quad \forall m\in M,n\in N,k\in\{1,2,...,d\}:p_{m,k}\geq 0,\ q_{n,k}\geq 0;$$

where $\|\cdot\|_F$ computes the Frobenius norm of an enclosed matrix, and λ_P and λ_Q are the regularization coefficients for P and Q, respectively.

B. Non-negative Latent Factor Model

For extracting non-negative LFs from Λ, NLF adopts the SLF-NMU algorithm to minimize objective (3). It firstly applies the additive gradient descent (AGD) to each desired LF, resulting in the following update rule:

$$\arg\min_{P,Q}\varepsilon(P,Q)\overset{AGD}{\Rightarrow}$$
$$\begin{cases}p_{m,k}\leftarrow p_{m,k}-\eta_{m,k}\sum_{n\in\Lambda(m)}\left(\lambda_P p_{m,k}-q_{n,k}\left(r_{m,n}-\hat{r}_{m,n}\right)\right),\\ q_{n,k}\leftarrow q_{n,k}-\eta_{n,k}\sum_{m\in\Lambda(n)}\left(\lambda_Q q_{n,k}-p_{m,k}\left(r_{m,n}-\hat{r}_{m,n}\right)\right).\end{cases} \quad (4)$$

where $\Lambda(m)$ and $\Lambda(n)$ denote the subsets of Λ related to m and n, $\eta_{m,k}$ and $\eta_{n,k}$ denote the learning rates corresponding to $p_{m,k}$ and $q_{n,k}$, respectively. With (4), $p_{m,k}$ and $q_{n,k}$ can be negative due to $-\eta_{m,k}\sum_{n\in\Lambda(m)}\left(q_{n,k}\hat{r}_{m,n}+\lambda_P p_{m,k}\right)$ and $-\eta_{n,k}\sum_{m\in\Lambda(n)}\left(p_{m,k}\hat{r}_{m,n}+\lambda_Q q_{n,k}\right)$, which are negative terms in the AGD-based learning rules. For cancelling them, SLF-NMU manipulates $\eta_{m,k}$ and $\eta_{n,k}$ as:

$$\eta_{m,k}=p_{m,k}\Big/\left(\sum_{n\in\Lambda(m)}q_{n,k}\hat{r}_{m,n}+\lambda_P\left|\Lambda(m)\right|p_{m,k}\right),$$
$$\eta_{n,k}=q_{n,k}\Big/\left(\sum_{m\in\Lambda(n)}p_{m,k}\hat{r}_{m,n}+\lambda_Q\left|\Lambda(n)\right|q_{n,k}\right). \quad (5)$$

By instituting (5) into (4), the learning rules for $p_{m,k}$ and $q_{n,k}$ are reformulated as follows [20]:

$$\arg\min_{P,Q}\varepsilon(P,Q)\overset{SLF-NMU}{\Rightarrow}$$
$$\begin{cases}p_{m,k}\leftarrow p_{m,k}\sum_{n\in\Lambda(m)}q_{n,k}r_{m,n}\Big/\left(\sum_{n\in\Lambda(m)}q_{n,k}\hat{r}_{m,n}+\lambda_P\left|\Lambda(m)\right|p_{m,k}\right),\\ q_{n,k}\leftarrow q_{n,k}\sum_{m\in\Lambda(n)}p_{m,k}r_{m,n}\Big/\left(\sum_{m\in\Lambda(n)}p_{m,k}\hat{r}_{m,n}+\lambda_Q\left|\Lambda(n)\right|q_{n,k}\right).\end{cases} \quad (6)$$

With (6), the original NLF model is achieved, which is frequently adopted in various data analysis tasks [33-36].

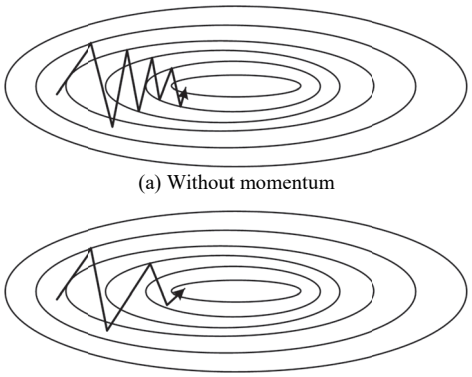
C. Momentum Method

Gradient descent (GD) is a widely-adopted optimization algorithm [31, 32]. Given the decision parameter θ of the objective $J(\theta)$, a standard GD algorithm updates θ as follows:

$$\theta_t=\theta_{t-1}-\eta\nabla_\theta J(\theta) \quad (7)$$

where θ_t and θ_{t-1} denote the states of θ at the tth and (t-1)th iterations, η denotes the learning rate, respectively. As indicated by prior work [11-17], GD is also frequently adopted by LF models.

However, for a complex model built on large-scale data, adopting GD can usually result in slow convergence [31, 32]. This is because GD can be easily trapped by ravines scattering around local optima, where the surface curves in one dimension are far steeper than those in another [31, 32]. When encountering such a ravine, GD makes a model oscillate across its slopes [31, 32], resulting in slow convergence as in Fig. 1.



(a) Without momentum

(b) With momentum

Fig. 1. Training process by GD with/without momentum.

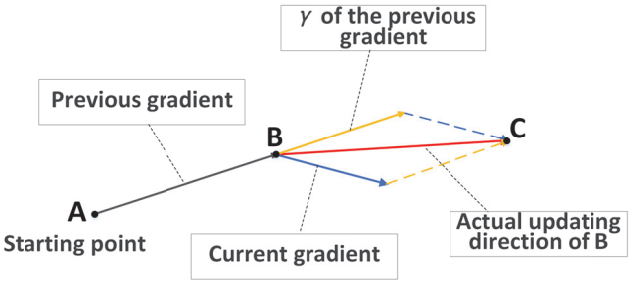


Fig. 2. Illustration of the fundamentals of momentum.

For accelerating a GD algorithm, prior researchers propose the momentum method [31, 32], whose principle is depicted in Fig. 2. With it, a GD algorithm records the update of the latest iteration, and determines the update of the current iteration as a linear combination of the current gradient and the latest update. Formally, given the decision parameter θ of objective $J(\theta)$, a momentum-incorporated GD algorithm updates it as follows:

$$v_0=0,$$
$$v_t=\gamma v_{t-1}+\eta\nabla_\theta J(\theta_{t-1}), \quad (8)$$
$$\theta_t=\theta_{t-1}-v_t.$$

In (8), v_0 denotes the initial state of the update velocity and is set at 0, v_t and v_{t-1} denote the update velocity vectors of the tth and (t-1)th iterations, and γ denotes the constant balancing the effects of the previous update velocity vector and current gradient, respectively. Therefore, a momentum-incorporated GD algorithm updates the desired parameters along the direction between the actual gradient and the latest update velocity vector, thereby dampening oscillations to achieve fast convergence as depicted in Fig. 2.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TSMC.2018.2875452, IEEE Transactions on Systems, Man, and Cybernetics: Systems

D. Summery

An NLF model extracts non-negative LFs from an HiDS matrix via SLF-NMU, which is actually implemented by manipulating the learning rates in a GD algorithm to obtain the multiplicative update rules (6). On the other hand, a momentum method combines the current gradient and latest update velocity vector to accelerate a GD algorithm. Intuitively, we expect to accelerate an NLF model's training process via incorporating the momentum method into an SLF-NMU algorithm. Nonetheless, as shown in (6), the SLF-NMU-based learning rules in an NLF model depend on gradients implicitly, while momentum-incorporated learning rules (8) explicitly rely on the gradients of decision parameters in each iteration. Hence, we need to generalize the standard momentum method, making it depend on the decision parameters' update increments rather than their gradients, thus compatible with an NLF model. Next we present our method.

III. Fast Non-negative Latent Factor Model

A. Generalized Momentum Method

A standard momentum method is designed for a GD algorithm [31, 32] which adopts gradients explicitly as in (8). However, from (8) we see that the parameter update in a momentum-incorporated GD algorithm naturally consists of the following components:

a) Initial state of decision parameters, i.e., θ_{t-1} in (8);

b) Update increment by the adopted algorithm. In (8), the update increment by GD is $\eta\nabla_\theta J(\theta)$; and

c) Velocity momentum by the latest velocity, i.e., γv_{t-1} in (8).

Actually, in an algorithm implicitly depending on gradients, we can also calculate the update increment by the adopted algorithm. Let θ'_t denote the expected state of the decision parameter on the adopted algorithm after the tth iteration, then we calculate the update increment as follows:

$$\Delta_t = \theta'_t - \theta_{t-1}. \tag{9}$$

By replacing the gradient-dependent term in (8) with the parameter update (9), we obtain a generalized form of the update velocity vector in the tth iteration:

$$v_t = \gamma v_{t-1} - \Delta_t = \gamma v_{t-1} - \left(\theta'_t - \theta_{t-1}\right). \tag{10}$$

For verifying (10), with GD as the optimization algorithm we have the following inferences:

$$\begin{aligned} &\theta'_t = \theta_{t-1} - \eta\nabla_\theta J\left(\theta_{t-1}\right), \\ &\Rightarrow \Delta_t = \theta'_t - \theta_{t-1} = -\eta\nabla_\theta J\left(\theta_{t-1}\right), \\ &\Rightarrow v_t = \gamma v_{t-1} - \Delta_t = \gamma v_{t-1} + \eta\nabla_\theta J\left(\theta_{t-1}\right); \end{aligned} \tag{11}$$

where we see the equivalence of the velocity update rules in (8) and (10). Thus, by combining (8) and (10), we achieve a generalized momentum method:

$$\begin{aligned} &v_0 = 0, \\ &v_t = \gamma v_{t-1} - \left(\theta'_t - \theta_{t-1}\right), \\ &\theta_t = \theta_{t-1} - v_t. \end{aligned} \tag{12}$$

By comparing (12) and (8), we see the main difference between a generalized momentum method and a standard one: a standard momentum method explicitly adopt the gradient of the objective function with respect to the decision parameter, while a generalized momentum method relies on the latest status of the decision parameter and the expected state of the decision parameter relying on an adopted learning algorithm. Hence, a generalized momentum method is compatible with an NLF model in the following aspects:

a) As shown in (6), NLF adopts SLF-NMU as the learning algorithm, which depends on gradients implicitly. With it, the update increment of decision parameters can still be achieved. Hence, we can deduce the expression for the velocity update in (12) based on SLF-NMU, making the generalized momentum method compatible with an NLF model; and

b) As shown in (6), SLF-NMU is also a gradient-dependent learning algorithm: it adopts carefully-chosen learning rates to achieve the multiplicative update. Consequently, it still suffers ravine traps. By applying the generalized momentum method to it, we can probably improve its convergence rate and prediction accuracy for missing data by adjusting its learning direction during the optimization process, following the principle of a momentum method given in Section II(C).

B. SLF-NM^{2}U Algorithm

By combining Sections II(A-B), we design the SLF-NM^{2}U algorithm. Let P_{t-1} and Q_{t-1} denote the status of P and Q after the (t-1)th iteration, and P'_t and Q'_t denote the status after the tth iteration with SLF-NMU, respectively. Thus, we write the training process of the tth iteration as follows,

$$\left(P'_t, Q'_t\right) = \underset{P,Q}{\overset{\text{SLF-NMU}}{\arg\min}}\, \varepsilon\left(P_{t-1}, Q_{t-1}\right). \tag{13}$$

Then the update caused by SLF-NMU is computed as:

$$\Delta_t = \begin{bmatrix} P'_t \\ Q'_t \end{bmatrix} - \begin{bmatrix} P_{t-1} \\ Q_{t-1} \end{bmatrix}. \tag{14}$$

On the other hand, based on (10) and (12), we calculate v_1, i.e., the update velocity vector for the first iteration as:

$$v_1 = \gamma v_0 - \Delta_1 = -\begin{bmatrix} P'_1 \\ Q'_1 \end{bmatrix} + \begin{bmatrix} P_0 \\ Q_0 \end{bmatrix}, \tag{15}$$

where P_0 and Q_0 denote the initial state of P and Q. As shown in prior work [20, 30], P_0 and Q_0 are random and non-negative initial guesses for P and Q, which are generated before the first iteration. By combining (12) and (15), we achieve the expression of the update rule for P_1 and Q_1:

$$\begin{bmatrix} P_1 \\ Q_1 \end{bmatrix} = \begin{bmatrix} P_0 \\ Q_0 \end{bmatrix} - v_1 = \begin{bmatrix} P_0 \\ Q_0 \end{bmatrix} + \begin{bmatrix} P'_1 \\ Q'_1 \end{bmatrix} - \begin{bmatrix} P_0 \\ Q_0 \end{bmatrix} = \begin{bmatrix} P'_1 \\ Q'_1 \end{bmatrix}. \tag{16}$$

By substituting (16) into (15), we see that

$$v_1 = -\begin{bmatrix} P'_1 \\ Q'_1 \end{bmatrix} + \begin{bmatrix} P_0 \\ Q_0 \end{bmatrix} = -\begin{bmatrix} P_1 \\ Q_1 \end{bmatrix} + \begin{bmatrix} P_0 \\ Q_0 \end{bmatrix}, \tag{17}$$

which is consistent with the third equation in (12). Afterwards, considering the second iteration, we have:

$$\left(P'_2, Q'_2\right) = \underset{P,Q}{\overset{\text{SLF-NMU}}{\arg\min}}\, \varepsilon\left(P_1, Q_1\right) \Rightarrow \Delta_2 = \begin{bmatrix} P'_2 \\ Q'_2 \end{bmatrix} - \begin{bmatrix} P_1 \\ Q_1 \end{bmatrix}, \tag{18}$$

where P'_2 and Q'_2 denote the states of P and Q after the second iteration with SLF-NMU. By combining (12), (16) and (18), we achieve the update velocity vector v_2 for the second iteration:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TSMC.2018.2875452, IEEE Transactions on Systems, Man, and Cybernetics: Systems

$$\begin{aligned} v_2 &= \gamma v_1 - \Delta_2 = \gamma\left(-\begin{bmatrix} P_1 \\ Q_1 \end{bmatrix} + \begin{bmatrix} P_0 \\ Q_0 \end{bmatrix}\right) - \begin{bmatrix} P_2' \\ Q_2' \end{bmatrix} + \begin{bmatrix} P_1 \\ Q_1 \end{bmatrix} \\ \Rightarrow \begin{bmatrix} P_2 \\ Q_2 \end{bmatrix} &= \begin{bmatrix} P_1 \\ Q_1 \end{bmatrix} - v_2 = \begin{bmatrix} P_2' \\ Q_2' \end{bmatrix} + \gamma\left(\begin{bmatrix} P_1 \\ P_1 \end{bmatrix} - \begin{bmatrix} P_0 \\ Q_0 \end{bmatrix}\right) \end{aligned} \tag{19}$$

By analogy, the update rules for the third~tth iterations with SLF-NM^{2}U is similar to (19). Hence, with (16) and (18), we arrive at the following training scheme:

$$\begin{cases} t=1: \begin{bmatrix} P_1 \\ Q_1 \end{bmatrix} = \begin{bmatrix} P_1' \\ Q_1' \end{bmatrix}, \\ t \geq 2: \begin{bmatrix} P_t \\ Q_t \end{bmatrix} = \begin{bmatrix} P_t' \\ Q_t' \end{bmatrix} + \gamma\left(\begin{bmatrix} P_{t-1} \\ Q_{t-1} \end{bmatrix} - \begin{bmatrix} P_{t-2} \\ Q_{t-2} \end{bmatrix}\right). \end{cases} \tag{20}$$

Note that with SLF-NMU, the parameter update should be taken with respect to each LF. Thus, by combining (6) and (20), we obtain the following update rule:

$$\begin{aligned} &\arg\min_{P,Q} \varepsilon(P,Q) \overset{SLF-NM^2U}{\Rightarrow} \\ &\begin{cases} t=1: \begin{cases} p_{m,k} \leftarrow p_{m,k} \dfrac{\sum_{n\in\Lambda(m)} q_{n,k}\, r_{m,n}}{\sum_{n\in\Lambda(m)} q_{n,k}\, \hat{r}_{m,n} + \lambda_P |\Lambda(m)| p_{m,k}}, \\ q_{n,k} \leftarrow q_{n,k} \dfrac{\sum_{m\in\Lambda(n)} p_{m,k}\, r_{m,n}}{\sum_{m\in\Lambda(n)} p_{m,k}\, \hat{r}_{m,n} + \lambda_Q |\Lambda(n)| q_{n,k}}; \end{cases} \\ t\geq 2: \begin{cases} p_{m,k} \leftarrow p_{m,k} \dfrac{\sum_{n\in\Lambda(m)} q_{n,k}\, r_{m,n}}{\sum_{n\in\Lambda(m)} q_{n,k}\, \hat{r}_{m,n} + \lambda_P |\Lambda(m)| p_{m,k}} + \gamma\left(p_{m,k} - p_{m,k}\right), \\ q_{n,k} \leftarrow q_{n,k} \dfrac{\sum_{m\in\Lambda(n)} p_{m,k}\, r_{m,n}}{\sum_{m\in\Lambda(n)} p_{m,k}\, \hat{r}_{m,n} + \lambda_Q |\Lambda(n)| q_{n,k}} + \gamma\left(q_{n,k} - q_{n,k}\right); \end{cases} \end{cases} \end{aligned} \tag{21}$$

Velocity terms $\gamma\left(p_{m,k} - p_{m,k}\right)$ and $\gamma\left(q_{n,k} - q_{n,k}\right)$ in (21) can become negative since we cannot guarantee that each LF is non-decreasing during the training process. For keeping resultant LFs non-negative, we truncate these velocity terms to zeroes once they become negative, resulting in the following training scheme:

$$\begin{aligned} &\arg\min_{P,Q} \varepsilon(P,Q) \overset{SLF-NM^2U}{\Rightarrow} \\ &\begin{cases} t=1: \begin{cases} p_{m,k} \leftarrow p_{m,k} \dfrac{\sum_{n\in\Lambda(m)} q_{n,k}\, r_{m,n}}{\sum_{n\in\Lambda(m)} q_{n,k}\, \hat{r}_{m,n} + \lambda_P |\Lambda(m)| p_{m,k}}, \\ q_{n,k} \leftarrow q_{n,k} \dfrac{\sum_{m\in\Lambda(n)} p_{m,k}\, r_{m,n}}{\sum_{m\in\Lambda(n)} p_{m,k}\, \hat{r}_{m,n} + \lambda_Q |\Lambda(n)| q_{n,k}}; \end{cases} \\ t\geq 2: \begin{cases} p_{m,k} \leftarrow p_{m,k} \dfrac{\sum_{n\in\Lambda(m)} q_{n,k}\, r_{m,n}}{\sum_{n\in\Lambda(m)} q_{n,k}\, \hat{r}_{m,n} + \lambda_P |\Lambda(m)| p_{m,k}} + \max\left\{\gamma\left(p_{m,k} - p_{m,k}\right), 0\right\}, \\ q_{n,k} \leftarrow q_{n,k} \dfrac{\sum_{m\in\Lambda(n)} p_{m,k}\, r_{m,n}}{\sum_{m\in\Lambda(n)} p_{m,k}\, \hat{r}_{m,n} + \lambda_Q |\Lambda(n)| q_{n,k}} + \max\left\{\gamma\left(q_{n,k} - q_{n,k}\right), 0\right\}. \end{cases} \end{cases} \end{aligned} \tag{22}$$

Based on (22), we achieve the SLF-NM^{2}U algorithm.

C. Incorporation of Linear Biases

As indicated in prior research [30], to incorporate linear biases into an NLF model can make it achieve a stable training process as well as high prediction accuracy for missing data of an HiDS matrix [30]. Considering objective (3), with linear biases $B^{|M|}$ for M and $C^{|N|}$ for N, we reformulate it as follows:

$$\begin{aligned} &\arg\min_{B,C,P,Q} \varepsilon(B,C,P,Q) \\ &= \frac{1}{2} \sum_{r_{m,n}\in\Lambda} \left(\left(r_{m,n} - \hat{r}_{m,n}\right)^2 + \lambda_B b_m^2 + \lambda_C c_n^2 + \lambda_P \sum_{k=1}^{d} p_{m,k}^2 + \lambda_Q \sum_{k=1}^{d} q_{n,k}^2 \right), \\ &s.t. \quad \forall m \in M, n \in N, k \in \{1,2,...,d\}: \\ &\qquad b_m \geq 0,\ c_n \geq 0, p_{m,k} \geq 0,\ \ q_{n,k} \geq 0; \end{aligned} \tag{23}$$

where the approximation $\hat{r}_{m,n}$ to each $r_{m,n} \in \Lambda$ is given by:

$$\hat{r}_{m,n} = b_m + c_n + \sum_{k=1}^{d} p_{m,k} q_{n,k}. \tag{24}$$

Based on the inferences shown in (13)~(20), we achieve the momentum-incorporated raining scheme for B, C, P and Q as:

$$\begin{cases} t=1: \begin{bmatrix} B_1 \\ C_1 \\ P_1 \\ Q_1 \end{bmatrix} = \begin{bmatrix} B_1' \\ C_1' \\ P_1' \\ Q_1' \end{bmatrix}, \\ t\geq 2: \begin{bmatrix} B_t \\ C_t \\ P_t \\ Q_t \end{bmatrix} = \begin{bmatrix} B_t' \\ C_t' \\ P_t' \\ Q_t' \end{bmatrix} + \gamma\left(\begin{bmatrix} B_{t-1} \\ C_{t-1} \\ P_{t-1} \\ Q_{t-1} \end{bmatrix} - \begin{bmatrix} B_{t-2} \\ C_{t-2} \\ P_{t-2} \\ Q_{t-2} \end{bmatrix}\right). \end{cases} \tag{25}$$

Then based on (6), (22) and (25), we achieve the SLF-NM^{2}U algorithm for a fast and biased non-negative latent factor (FBNLF) model. Its update rules are highly close to that of an FNLF given in (22), but is expanded with the update rules for linear biases in B and C.

D. Algorithm Design and Analysis

From the previous sections, we design the Algorithm FNLF. As shown in Algorithm FNLF, we adopt several auxiliary matrices for a) caching the training increment, and b) caching the intermediate results for computing the momentum effects. For instance, we adopt four auxiliary matrices, i.e., *PU*, *PD*, *PO* and *PT* for P. Among them, a) *PU* and *PD* cache the learning increment on each instance $r_{m,n} \in \Lambda$, making the algorithm able to record the training increments on all involved LFs within a single traverse on Λ; and b) *PO* and *PT* cache the intermediate states of P in the (t-1)th and (t-2)th iterations, for helping the algorithm computing the momentum effects correctly. Similar settings are applied to B, C and Q. Hence, the storage cost of Algorithm FNLF relies on Λ, M, N, B, C, P and Q, along with the auxiliary arrays. Their costs sum up to:

$$S_{FNLF} = |\Lambda| + 5\times(|M| + |N|)\times d + 6\times(|M| + |N|). \tag{26}$$

From (26), we see that the storage cost of FNLF is linear with the known entry count in R and number of involved entities. For industrial applications, such storage burden is easy to resolve.

Based on Algorithm FNLF, we summarize FNLF's

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TSMC.2018.2875452, IEEE Transactions on Systems, Man, and Cybernetics: Systems

computational cost as follows,

$$T_{FNLF} = \Theta\left(n \cdot \left(\left|M\right| + \left|N\right|\right) \cdot d + n \cdot \left|\Lambda\right| \cdot d\right) \approx \Theta\left(n \cdot \left|\Lambda\right| \cdot d\right). \quad (27)$$

Algorithm FNLF

Input: $\boldsymbol{M}$, $\boldsymbol{N}$, $\boldsymbol{\Lambda}$, $\boldsymbol{f}$, $\boldsymbol{\gamma}$,	
Operation	**Cost**
initialize $P^{\|M\|\times d}$, $PU^{\|M\|\times d}$, $PD^{\|M\|\times d}$, $PO^{\|M\|\times d}$=$PT^{\|M\|\times d}$=P, λ_P	$\Theta(\|M\|\times d)$
initialize $Q^{\|N\|\times d}$, $QU^{\|N\|\times d}$, $QD^{\|N\|\times d}$, $QO^{\|N\|\times d}$=$QT^{\|N\|\times d}$=Q, λ_Q	$\Theta(\|N\|\times d)$
initialize t = 0, *Max-training-round* = n	$\Theta(1)$
if biased **then**	-
initialize $B^{\|M\|}$, $BU^{\|M\|}$, $BD^{\|M\|}$, $BO^{\|M\|}$=$BT^{\|M\|}$=B, λ_B	$\Theta(\|M\|)$
initialize $C^{\|N\|}$, $CU^{\|N\|}$, $CD^{\|N\|}$, $CO^{\|N\|}$=$CT^{\|N\|}$=C, λ_C	$\Theta(\|N\|)$
end if	-
while not converge **and** $t \leq n$ **do**	$\times t$
reset PU=0, PD=0, QU=0, QD=0	$\Theta((\|M\|+\|N\|)\times d)$
if biased **then**	-
reset BU=0, BD=0, CU=0, CD=0	$\Theta(\|M\|+\|N\|)$
end if	-
/-Recording train increment-/	
for each $r_{m,n}$ **in** Λ	$\times\|\Lambda\|$
$\hat{r}_{m,n} = \sum_{k=1}^{d} p_{m,k} q_{n,k}$	$\Theta(d)$
if biased then	-
$\hat{r}_{m,n} = \hat{r}_{m,n} + b_m + c_n$	$\Theta(1)$
end if	-
for k=1 **to** d	$\times d$
$pu_{m,k} = pu_{m,k} + q_{n,k} r_{m,n}$	$\Theta(1)$
$pd_{m,k} = pd_{m,k} + q_{n,k} \hat{r}_{m,n} + \lambda_P p_{m,k}$	$\Theta(1)$
$qu_{n,k} = qu_{n,k} + p_{m,k} r_{m,n}$	$\Theta(1)$
$qd_{n,k} = qd_{n,k} + p_{m,k} \hat{r}_{m,n} + \lambda_Q q_{n,k}$	$\Theta(1)$
end for	-
if biased then	-
$bu_m = bu_m + r_{m,n}$	$\Theta(1)$
$bd_m = bd_m + \hat{r}_{m,n} + \lambda_B b_m$	$\Theta(1)$
$cu_n = cu_n + r_{m,n}$	$\Theta(1)$
$cd_n = cd_n + \hat{r}_{m,n} + \lambda_C c_n$	$\Theta(1)$
end if	-
end for	-
/-Updating-/	
for $m \in M$	$\times\|U\|$
if biased **then**	-
b_m=$b_m \cdot (bu_m/bd_m)$+max{0, $\gamma \cdot (bo_m - bt_m)$}	$\Theta(1)$
end if	-
for k=1 **to** d	$\times f$
$p_{m,k}$=$p_{m,k} \cdot (pu_{m,k}/pd_{m,k})$+max{0, $\gamma \cdot (po_{m,k} - pt_{m,k})$}	$\Theta(1)$
end for	-
end for	-
for $n \in N$	$\times\|I\|$
if biased **then**	-
c_n=$c_n \cdot (cu_n/cd_n)$+max{0, $\gamma \cdot (co_n - ct_n)$}	$\Theta(1)$
end if	-
for k=1 **to** d	$\times f$
$q_{n,k}$=$q_{n,k} \cdot (qu_{n,k}/qd_{n,k})$+max{0, $\gamma \cdot (qo_{n,k} - qt_{n,k})$}	$\Theta(1)$
end for	-
end for	-
/-Recording state of current LFs for momentum-/	
PT= PO, PO=P, QT=QO, QO=Q	$\Theta((\|M\|+\|N\|)\times d)$
if biased **then**	-
BT= BO, BO=B, CT=CO, CO=C	$\Theta((\|M\|+\|N\|))$
end if	-
t=t+1	$\Theta(1)$
end while	-
Output $\boldsymbol{P}$, $\boldsymbol{Q}$ **for FNLF** $\boldsymbol{B}$, $\boldsymbol{C}$, $\boldsymbol{P}$, $\boldsymbol{Q}$ **for FBNLF**	

Note that (27) adopts the condition $|\Lambda| \gg \max\{|M|,|N|\}$ to drop the lower-order-terms, which is constantly fulfilled in industrial applications. Since both n and d are positive constants in practice, the computational cost of FNLF is linear with $|\Lambda|$.

Based on the above inferences, we see that Algorithm FNLF is highly efficient in both computation and storage. Next we validate the performance of FNLF and FBNLF on HiDS matrices generated by industrial applications.

IV. Experimental Results and Analysis

A. General Settings

Evaluation Protocol. For industrial applications, one major motivation to analyze an HiDS matrix is for missing data estimation, owing to the great desire for discovering the full connections among involved entities [1-9]. Hence, we adopt it as the evaluation protocol, where a tested model's prediction accuracy for missing data of an HiDS matrix is concerned. It can be measured by root mean squared error (RMSE) [11-17, 20, 30, 37]:

$$RMSE = \sqrt{\left(\sum_{r_{u,v} \in \Gamma} \left(r_{u,v} - \hat{r}_{u,v}\right)^2\right) \Big/ |\Gamma|}, \quad (28)$$

where Γ denotes the validation set and is disjoint with Λ, $\hat{r}_{u,v}$ denotes the estimated value generated by the tested model corresponding to the instance $r_{u,v} \in \Gamma$, and $|\cdot|$ calculates the cardinality of a given set, respectively

Meanwhile, we are also interested in the computational cost of each tested model. We have recorded their converging iteration count and time cost per iteration. All experiments are conducted on a Tablet with a 2.6 GHz i7 CPU and 16 GB RAM, and implemented in JAVA SE 7U60.

Datasets. Six HiDS matrices are adopted in our experiments. Note that for validating the feasibility of the FNLF model in addressing big data from real applications, all of them are real datasets collected by industrial companies. Their details are summarized in Table I.

TABLE I
DETAILS OF EXPERIMENTAL DATASETS

No.	Name	$\|\Lambda\|+\|\Gamma\|$	$\|M\|$	$\|N\|$	Source
D1	ML20M	20,000,263	138,493	26,744	MovieLens [42]
D2	Flixter	8,196,077	147,612	48,794	Flixter [38]
D3	Douban	16,830,839	129,490	58,541	Douban [40]
D4	EM	2,811,718	72,916	1,628	EachMovie
D5	Dating	17,359,346	135,359	168,791	LibimSeTi [41]
D6	NetFlix	54,782,019	478,350	10,000	NetFlix [11-13]

As shown in Table I, all datasets are a) high-dimensional, b) extremely sparse, and c) collected by industrial applications in use. Hence, results on them are highly representative.

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 five-fold cross-validations, i.e., each time we select four subsets as the training set Λ to train a model predicting the remaining one subset as the testing set Γ. This process is sequentially repeated for five times to obtain the final results. The training process of a tested model terminates if a) the number of consumed iterations reaches a preset threshold, i.e., 1000, and b) the model converges, i.e., the error difference between two

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TSMC.2018.2875452, IEEE Transactions on Systems, Man, and Cybernetics: Systems

consecutive iterations is smaller than 10^{-5}.

Model Settings. Four models are involved in our experiments, whose details are given in Table II.

TABLE II
COMPARED MODELS IN OUR EXPERIMENTS

No.	Name	Description
M1	NLF	Original NLF model relying on SLF_NMU proposed in [20].
M2	FNLF	Fast NLF model relying on SLF_NM^{2}U proposed in Section III(B).
M3	BNLF	Biased NLF model relying on SLF_NMU proposed in [30].
M4	FBNLF	Biased NLF model relying on SLF_NM^{2}U proposed in Section III(C).

To obtain objective results, we adopt the following settings:

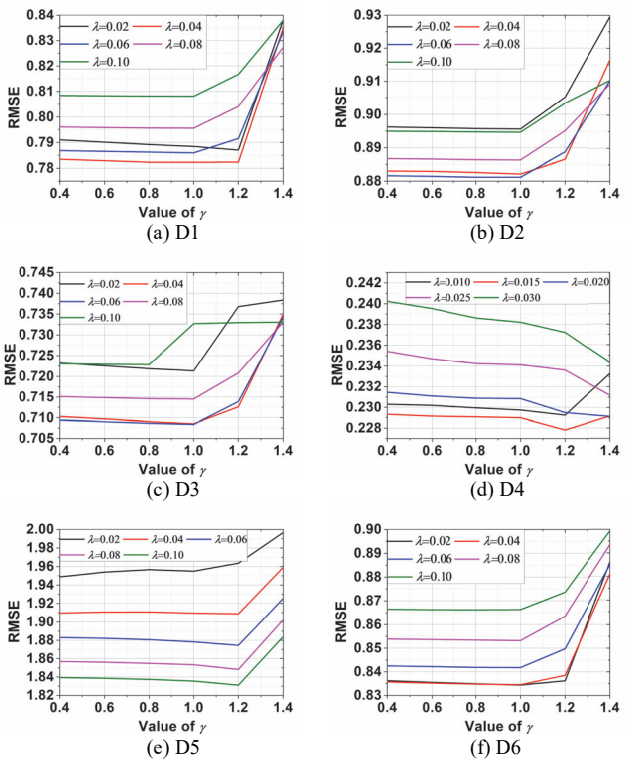
a) For all models, we make $\lambda_P = \lambda_Q = \lambda_B = \lambda_C = \lambda$, following the instructions in [20, 30];

b) We initialize B, C, P and Q with the same randomly-generated and non-negative arrays for all involved models to eliminate the impact caused by random initial guesses as discussed in [11-17, 20, 30]; and

c) We repeat each set of experiments for 10 times and take the average of the outputs as the final results.

B. Parameter Sensitivity Tests

As discussed in Section III, FNLF and FBNLF rely on the regularization coefficient λ and momentum coefficient γ. Hence, we firstly conduct the model sensitivity tests with respect to them. Note that both M2 and M4 rely on SLF-NM^{2}U, so we present the results for M4 on all six datasets for a concise report. However, similar situations can be found for M2. The tested scale for λ is (0.01, 0.10), and for γ is (0.4, 1.4), respectively.

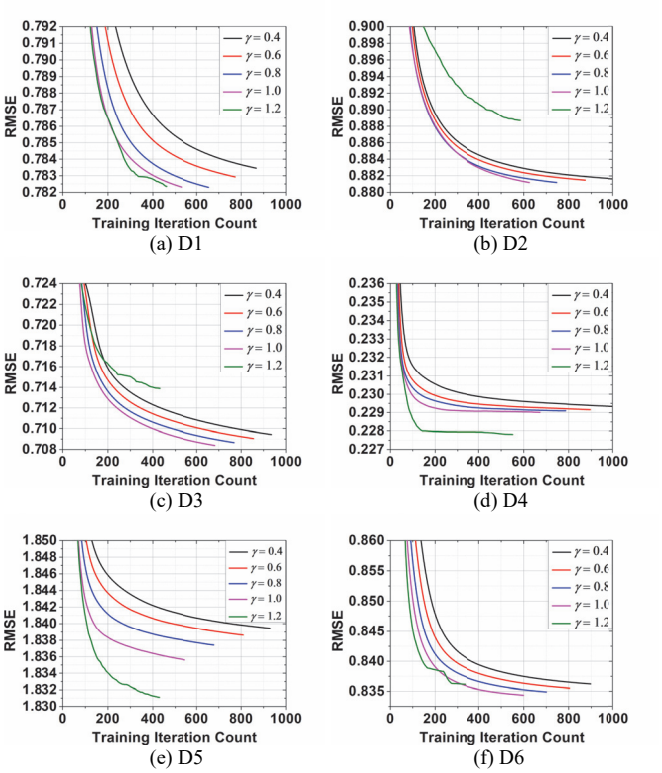


(a) D1 (b) D2 (c) D3 (d) D4 (e) D5 (f) D6

Fig. 3. Relationship among RMSE, λ and γ in M4.

Fig. 3 depicts the relationship among RMSE, λ and γ in M4; Fig. 4 depicts M4's training process with optimal λ as γ varies;

Table III summarizes optimal λ and γ for M4. From them, we have the following findings:



(a) D1 (b) D2 (c) D3 (d) D4 (e) D5 (f) D6

Fig. 4. M4's training process with optimal value of λ as γ varies.

a) With SLU-NM^{2}U as the learning algorithm, M4's RMSE is sensitive to both λ and γ. As shown in Fig. 3, with optimal λ and γ, M4 achieves low RMSE. On the other hand, inappropriate λ and γ result in accuracy loss. As shown in Fig. 3(b), by setting γ=1, M4's RMSE with λ=0.002, 0.004, 0.006, 0.008 and 0.010 is 0.8957, 0.8826, 0.8812, 0.8864 and 0.8947, respectively. The lowest RMSE is 0.8812 with λ=0.006, and the highest is 0.8957 with λ=0.002. The gap between the highest RMSE and lowest RMSE, i.e., the ratio calculated by $(RMSE_{highest}\text{-}RMSE_{lowest})/RMSE_{highest}$, is 1.62%, which is obvious. Similar situations can be found on the other datasets, as shown in Figs. 3(b)-(f).

b) Small γ eliminates the momentum effects, while large γ makes a learning algorithm overshoot a local optimum. This assertion is supported by the training process of M4, as shown in Fig. 4. On D1, by setting γ=0.4, M4 converges after 867 iterations, achieving the RMSE at 0.7834. In contrast, with the optimal γ=0.8 on D1, M4 consumes 654 iterations to achieve the RMSE at 0.7823. Compared with the case of γ=0.4, the converging iteration count decreases at 24.6% while the prediction accuracy increases at 0.14%.
Nonetheless, as γ grows too large, M4 suffers overshooting, which results in accuracy loss. For instance, as shown in Fig. 4(b), with γ=1.2, M4 suffers fluctuations during the training process, and converges with the RMSE at 0.8887. Compared with the RMSE at 0.8812 caused by the optimal γ=1.0, the prediction accuracy decreases at 0.84%.

c) For an FNLF/FBNLF model, the optimal values for λ and γ are data dependent. As summarized in Table III, for M4, the optimal λ and γ keep changing on different datasets. For γ,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TSMC.2018.2875452, IEEE Transactions on Systems, Man, and Cybernetics: Systems

the optimal value is around 1.0 and may increase/decrease slightly in our experiments. For λ, its optimal value changes vastly on different datasets. However, on the same dataset, the optimal value of λ is not model-dependent. For instance, Fig. 5 depicts the optimal λ on D1 and D2 for M1-M4. From it, we see that on D1 the optimal λ is consistently 0.04 for M1-M4. Similarly, the optimal λ on D2 is 0.06 for M1-M4.

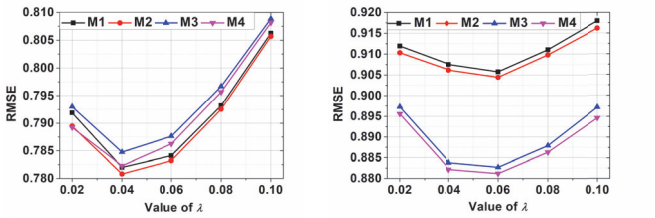


Fig. 5. Optimal value of λ on D1 and D2 for M1-M4.

TABLE III
OPTIMAL VALUES OF λ AND γ FOR M4 ON D1-D6.

Dataset.	Density	λ	γ	Lowest RMSE
D1	0.54%	0.040	0.80	0.7823
D2	0.11%	0.060	1.00	0.8812
D3	0.22%	0.060	1.00	0.7084
D4	2.37%	0.015	1.20	0.2278
D5	0.076%	0.100	1.20	1.8311
D6	1.14%	0.020	1.00	0.8344

d) From these results, we see that FNLF and FBNLF's performance is closely connected with λ and γ, whose optimal values are data-dependent. γ balances the training direction between the intuitive update by the adopted learning algorithm and momentum effects describing the latest update. Inappropriate γ makes the model converge slow, or overshoot a local optimum. In our experiments, the optimal γ is around 1.0, as shown in Table III. Considering λ, it controls the regularization effects. Although its optimal value varies significantly on different dataset, it keeps consistent for M1-M4 on the same dataset.

C. Comparison with NLF models

In this part, we compare the proposed models with the NLF models proposed in [20, 30]. Fig. 6 depicts the training process of M1-M4; Tables IV and V summarize the lowest RMSE and time cost of M1-M4; Fig. 7 depicts LF distributions of M1-M4 on D2. From them, we have the following findings:

a) Owing to the generalized momentum method, an FNLF/FBNLF model converges faster than an NLF/BNLF model does. For instance, as shown in Fig. 6(a), M2 and M4 respectively take 677 and 654 iterations to converge on D1. In contrast, both M1 and M3 take 1,000 iterations. These outcomes indicate that by adopting the generalized momentum method, an SLF-NM^{2}U algorithm incorporates the momentum effects correctly into it, thereby accelerating the training process of an FNLF/FBNLF model.

b) Owing to the momentum effects, an FNLF/FBNLF model achieves higher prediction accuracy than an NLF/BNLF model does. For instance, on D1, M2's RMSE is 0.7808, about 0.15% lower than M1's RMSE at 0.7820. M4's RMSE is 0.7823, about 0.31% lower than M3's RMSE at 0.7847. Similar situations can be found on D2-D6, as depicted in Fig. 6 and Table IV. In general, an NLF/BNLF model is constantly outperformed by an FNLF/FBNLF model in terms of prediction accuracy.

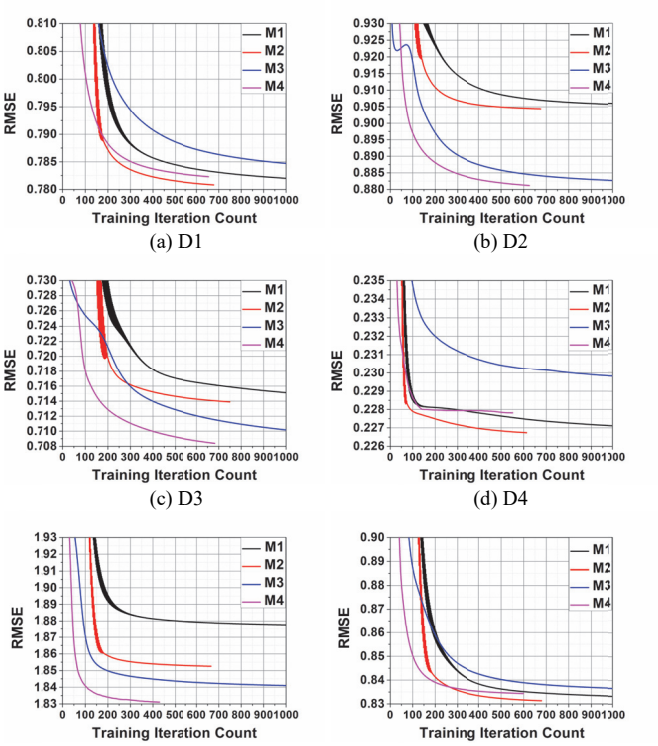


Fig. 6. Training process of M1-M4 on D1-D6.

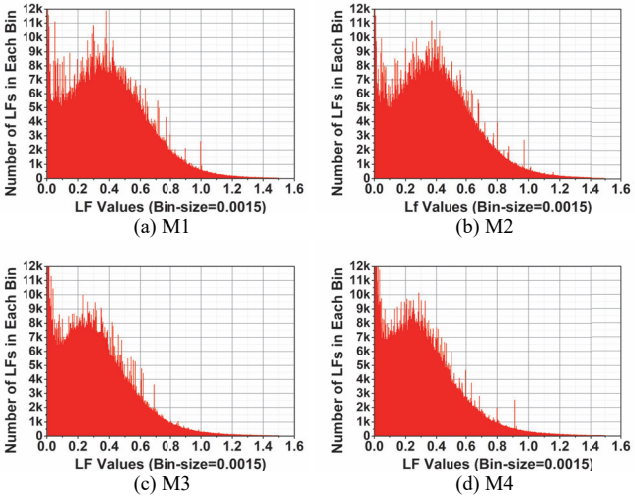


Fig. 7. LF Distributions of M1-M4 on D2.

c) With SLF-NM^{2}U, an FNLF/FBNLF model's time cost per iteration is a bit higher than that of an NLF/BNLF model. From Table V we see that the time cost per iteration on D1 by M1, M2, M3 and M4 is 1,701, 1,804, 1,892 and 1,987 milliseconds, respectively. M2's time cost per iteration is about 6.06% higher than that of M1, and M4's time cost per iteration is about 5.02% higher than that of M3. Similar situations are also encountered on D2-D6. By comparing Algorithm FNLF with the Algorithm NLF/BNLF proposed in [20, 30], we see that the former consumes more constant time in addressing the momentum term than the latter. Hence, it is reasonable that FNLF/FBNLF's time cost per iteration is slightly higher than that of NLF/BNLF.

d) LFs extracted by FNLF/BNLF are constantly non-negative.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TSMC.2018.2875452, IEEE Transactions on Systems, Man, and Cybernetics: Systems

As shown in Fig. 7, all LFs extracted by M1-M4 on D2 are scattering in the non-negative real field. Hence, SLF-NM^{2}U guarantees the non-negativity of output LFs.

TABLE IV
LOWEST RMSE OF M1-M4 ON D1-D6

Dataset.	M1	M2	M3	M4
D1	0.7820	**0.7808**	0.7847	0.7823
D2	0.9058	0.9044	0.8827	**0.8812**
D3	0.7151	0.7139	0.7102	**0.7084**
D4	0.2271	**0.2267**	0.2298	0.2278
D5	1.878	1.853	1.841	**1.831**
D6	0.8333	**0.8314**	0.8366	0.8344

TABLE V
TIME COST PER ITERATION OF M1-M4 ON D1-D6 (MILLISECONDS)

Dataset.	M1	M2	M3	M4
D1	**1,701**	1,804	1,892	1,987
D2	**657**	782	772	826
D3	**1,597**	1,741	1,679	1,855
D4	**226**	237	257	269
D5	**3,127**	3,285	3,879	4,115
D6	**94,107**	99,261	108,009	113,035

D. Summary

Based on the experimental results, we summarize that by incorporating the momentum effects into the training process, an FNLF/FBNLF model outperforms an NLF/BNLF model in both convergence rate and prediction accuracy for missing data of an HiDS matrix. However, their performance relies on hyper parameters λ and γ, which should be chosen with care. In general, the optimal values of γ on different datasets scatter around 1.0, which is quite stable. So we can adopt such empirical values in practice. However, like in an NLF/BNLF model, optimal value of λ in an FNLF/FBNLF model varies significantly on different datasets. How to make λ self-adaptive remains an open issue. We plan to address it in the future.

V. Discussions

A generalized momentum method. A standard momentum method is initially designed for a learning algorithm explicitly depending on gradients. It is inapplicable to learning algorithms implicitly depending on gradients, like an SLF-NMU algorithm for NLF models. In contrast, the proposed generalized momentum method calculates the components of 'current update' in an iteration via subtracting the initial state of the parameters from their expected state by the adopted algorithm. With it, we successfully integrate the momentum effects into SLF-NMU. In this work we empirically validated the effects by a generalized momentum method in non-negative LF analysis on HiDS matrices. However, it would be highly significantly to show its soundness by rigorously theoretical study. We plan to investigate this issue in the future.

Effects of momentum terms in FNLF/FBNLF. As shown in Section IV, the momentum effects in an FNLF/FBNLF model are significant. Compared with the original NLF/BNLF model, an FNLF/FBNLF model converges much faster with higher prediction accuracy for missing data of an HiDS matrix. Note that the momentum method works by making the searching process navigate across ravines, as shown in Fig. 1. Hence, it turns out that the original SLF-NMU algorithm can be easily affected by encountered ravines, but an SLF-NM^{2}U algorithm is less sensitive to them. It makes an FNLF/FBNLF model achieve better local optima with much higher convergence rate.

As indicated by prior research, LBs in an LF model can be extended in various aspects [51, 52]. Deep learning-based approaches [53, 54] are also becoming increasingly attractive in LF analysis. Moreover, Hessian-free optimization based LF analysis is also highly efficient [6]. It would be interesting to validate the compatibility between the principle of this work and these recent techniques.

VI. Conclusions

A non-negative latent factor (NLF) model adopts the single latent factor-dependent, non-negative and multiplicative update (SLF-NMU) as the learning algorithm. It suffers slow convergence on high-dimensional and sparse (HiDS) matrices. Meanwhile, due to its implicit dependence on gradients, it is incompatible with a standard momentum method, which proves to be highly effective in accelerating a learning algorithm explicitly depending on gradients.

We firstly design a generalized momentum method compatible with learning algorithms implicitly depending on gradients. We subsequently apply it to the SLF-NMU algorithm, thereby achieving a single latent factor-dependent, non-negative, multiplicative and momentum-incorporated update (SLF-NM^{2}U) algorithm. With it, we further propose the fast non-negative latent factor (FNLF) model and its biased version, along with careful algorithm design and analysis. Empirical studies show that an FNLF model outperforms an NLF model in terms of both convergence rate and prediction accuracy for missing data. Hence, it is highly useful for industrial applications desiring highly efficient, accurate and non-negative latent factor analysis.

References

[1] 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. on Cybernetics,* vol. 45, no. 11, pp. 2585-2598, 2015.

[2] Z. Ghahramani, "Probabilistic machine learning and artificial intelligence," *Nature,* vol. 521, no. 7553, pp. 452-459, 2015.

[3] D. He, D. Jin, Z. Chen, and W. Zhang, "Identification of hybrid node and link communities in complex networks," *Scientific Reports,* vol. 5, pp. 8638, 2015.

[4] 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. on Knowledge and Data Engineering,* vol. 17, no. 6, pp. 734-749, 2005.

[5] Y. Li, B. Cao, L. Xu, J. Yin, S. Deng, Y. Yin, and Z. Wu, "An Efficient Recommendation Method for Improving Business Process Modeling," *IEEE Trans. on Industrial Informatics,* vol. 10, no. 1, pp. 502-513, 2014.

[6] X. Luo, M.-C. Zhou, S. Li, Z.-H. You, Y.-N. Xia, Q.-S. Zhu, and H. Leung, "An Efficient Second-order Approach to Factorizing Sparse Matrices in Recommender Systems," *IEEE Trans. on Industrial Informatics,* vol. 11, no. 4, pp. 946 - 956, Aug., 2015.

[7] Z.-H. You, Y.-K. Lei, J. Gui, D.-S. Huang, and X.-B. Zhou, "Using manifold embedding for assessing and predicting protein interactions from high-throughput experimental data," *Bioinformatics,* vol. 26, no. 21, pp. 2744-2751, Nov, 2010.

[8] M. Hofree, J. P. Shen, H. Carter, A. Gross, and T. Ideker, "Network-based stratification of tumor mutations," *Nature Methods,* vol. 10, no. 11, pp. 1108-1115, 2013.

[9] D. Szklarczyk, A. Franceschini, S. Wyder, K. Forslund, D. Heller, J. Huerta-Cepas, M. Simonovic, A. Roth, A. Santos, K. P. Tsafou, M. Kuhn, P. Bork, L. J. Jensen, and C. v. Mering, "STRING v10: protein–protein interaction networks, integrated over the tree of life," *Nucleic Acids Research,* vol. 43, no. 1, pp. D447-D452, 2015.

[10] C.-X. Ou, and R.-M. Davison, "Technical opinion: Why eBay lost to TaoBao in China: the Glocal advantage," *Communications of the ACM,* vol. 52, no. 1, pp. 145-148, 2009.

[11] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," *IEEE Computer,* vol. 42, no. 8, pp. 30-37, Aug., 2009.

[12] G. Takács, I. Pilászy, Bottyán Németh, and D. Tikky, "Scalable Collaborative Filtering Approaches for Large Recommender Systems," *Journal of Machine Learning Research,* vol. 10, pp. 623-656, Mar., 2009.

[13] R. Salakhutdinov, and A. Mnih, "Probabilistic matrix factorization," *Advances in Neural Information Processing Systems,* vol. 20, pp. 1257-1264, 2008.

[14] J. Wu, L. Chen, Y.-P. Feng, Z.-B. Zheng, M.-C. Zhou, and Z.-H. Wu, "Predicting Quality of Service for Selection by Neighborhood-Based Collaborative Filtering," *IEEE Trans. on Systems, Man, and Cybernetics: Systems,* vol. 43, no. 2, pp. 428-439, 2013.

[15] X. Luo, M.-C. Zhou, H. Leung, Y.-N. Xia, Q.-S. Zhu, Z.-H. You, and S. Li, "An Incremental-and-Static-Combined Scheme for Matrix-Factorization-Based Collaborative Filtering," *IEEE Trans. on Automation Science and Engineering,* vol. 13, no. 1, pp. 333-343, Jan., 2016.

[16] J.-J. Pan, S.-J. Pan, Y. Jie, L.-M. Ni, and Y. Qiang, "Tracking Mobile Users in Wireless Networks via Semi-Supervised Colocalization," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. 34, no. 3, pp. 587-600, 2012.

[17] M.-F. Weng, and Y.-Y. Chuang, "Collaborative video reindexing via matrix factorization," *ACM Trans. Multimedia Comput. Commun. Appl.,* vol. 8, no. 2, pp. 1-20, 2012.

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

[19] S. Zhang, W. Wang, J. Ford, and F. Makedon, "Learning from Incomplete Ratings Using Non-negative Matrix Factorization," in *Proc. of the SIAM Int. Conf. on Data Mining,* Bethesda, USA, 2006, pp. 549-553.

[20] 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. on Industrial Informatics,* vol. 10, no. 2, pp. 1273 - 1284, May., 2014.

[21] D.-D. Lee, and S.-H. Seung, "Algorithms for Non-negative Matrix Factorization," *Advances in Neural Information Processing Systems,* vol. 13, no. 2000, pp. 556-562, 2000.

[22] 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.

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

[24] P. Hoyer, "Non-negative matrix factorization with sparseness constraints," *Journal of Machine Learning Research,* vol. 5, pp. 1457-1469, 2004.

[25] 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.

[26] 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. on Signal Processing,* vol. 57, no. 7, pp. 2858-2864, 2009.

[27] D. Rafailidis, and A. Nanopoulos, "Modeling Users Preference Dynamics and Side Information in Recommender Systems," *IEEE Trans. on Systems, Man, and Cybernetics: Systems,* vol. 46, no. 6, pp. 782-792, 2016.

[28] 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. on Signal Processing,* vol. 62, no. 7, pp. 1822-1833, 2014.

[29] Y. Xu, W. Yin, Z. Wen, and Y. Zhang, "An alternating direction algorithm for matrix completion with nonnegative factors," *Frontiers of Mathematics in China,* vol. 7, no. 2, pp. 365-384, 2012.

[30] 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. on Neural Networks and Learning Systems,* vol. 27, no. 3, pp. 524-537, 2016.

[31] D.-E. Rumelhart, G.-E. Hinton, and R.-J. Williams, "Learning representations by back-propagating errors," *Nature,* vol. 323, pp. 533, 1986.

[32] S. Boyd, and L. Vandenberghe, *Convex Optimization*, Cambridge: Cambridge University Press, 2009.

[33] A. Che, P. Wu, F. Chu, and M. Zhou, "Improved Quantum-Inspired Evolutionary Algorithm for Large-Size Lane Reservation," *IEEE Trans. on Systems, Man, and Cybernetics: Systems,* vol. 45, no. 12, pp. 1535-1548, 2015.

[34] 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. on Intelligent Transportation Systems,* vol. 17, no. 3, pp. 659-669, 2015.

[35] 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. on Intelligent Transportation Systems,* vol. 16, no. 3, pp. 1479-1492, 2015.

[36] L. Feng, and B. Bhanu, "Semantic Concept Co-Occurrence Patterns for Image Annotation and Retrieval," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. 38, no. 4, pp. 785-799, 2016.

[37] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Trans. on Information Systems,* vol. 22, no. 1, pp. 5-53, 2004.

[38] J. Mohsen, and E. Martin, "A matrix factorization technique with trust propagation for recommendation in social networks," in *Proc. of the Fourth ACM Conf. on Recommender Systems*, Barcelona, Spain, 2010, pp. 135-142.

[39] P. Massa, and P. Avesani, "Trust-aware recommender systems," in *Proc. of the First ACM Conf. on Recommender Systems*, Minneapolis, MN, USA, 2007, pp. 17-24.

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

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

[42] J.-A. Konstan, B.-N. Miller, D. Maltz, J.-L. Herlocker, L.-R. Gordon, and J. Riedl, "Grouplens: applying collaborative filtering to UseNet news," *Communications of the ACM,* vol. 40, no. 3, pp. 77-87, 1997.

[43] J. Cao, Z.-A. Wu, J.-J. Wu, and H. Xiong, "SAIL: Summation-based incremental learning for information-theoretic text clustering," *IEEE Trans. on Cybernetics*, vol. 43, no. 2, pp. 570-584, 2013.

[44] J. Cao, Z.-A. Wu, and J.-J. Wu, "Scaling up cosine interesting pattern discovery: A depth-first method," *Information Sciences*, vol. 266, pp. 31-46, 2014.

[45] J. Cao, B. Wang, and B. Douglas, "Similarity based leaf image retrieval using multiscale r-angle description," *Information Sciences*, vol. 374, pp. 51-64,2016.

[46] J. Cao, Z.-A Wu, J.-J. Wu, and W.-J. Liu, "Towards information-theoretic K-means clustering for image indexing," *Signal Processing*, vol. 93, no. 7, pp. 2026-2037, 2013.

[47] J. Cao, Z.-A. Wu, Y.-Q. Wang, and Y. Zhuang, "Hybrid Collaborative Filtering algorithm for bidirectional Web service recommendation," *Knowledge and information systems*, vol. 36, no. 3, pp. 607-627, 2013.

[48] J. Cao, Z.-A. Wu, B. Mao, and Y.-C. Zhang, "Shilling attack detection utilizing semi-supervised learning method for collaborative recommender system," *World Wide Web Journal: Internet and Web Information Systems*, vol. 16, no. 5-6, pp. 729-748, 2013.

[49] J. Cao, Z. Bu, G.-L. Gao, and H.-C. Tao, "Weighted modularity optimization for crisp and fuzzy community detection in large-scale networks," *Physica A: Statistical Mechanics and its Applications*, vol. 462, pp. 386-395, 2016.

[50] W.-J. Luan, G.-J. Liu, C.-J. Jiang, and L. Qi, "Partition-based Collaborative Tensor Factorization for POI Recommendation," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 3, pp. 442-451, 2017.

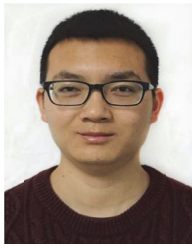
[51] Y. Yuan, X. Luo, and M.-S. Shang:, "Effects of Preprocessing and Training Biases in Latent Factor Models for Recommender Systems," *Neurocomputing,* vol. 275, pp. 2019-2030, 2018.

[52] J. Chen, X. Luo, Y. Yuan, M.-S. Shang, Z. Ming, and Z. Xiong:, "Performance of Latent Factor Models with Extended Linear Biases," *Knowledge-based Systems,* vol. 123, pp. 128-136, 2017.

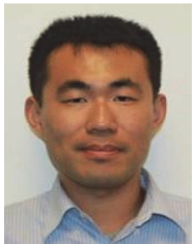
[53] Q. Li, B. Shen, Z. Wang, T. Huang, and J. Luo, "Synchronization Control for A Class of Discrete Time-Delay Complex Dynamical Networks: A Dynamic Event-Triggered Approach," *IEEE Transactions on Cybernetics,* vol. DOI: 10.1109/TCYB.2018.2818941.

[54] B. Shen, Z. Wang, and H. Qiao, "Event-triggered State Estimation for Discrete-time Multidelayed Neural Networks with Stochastic Parameters and Incomplete Measurements," *IEEE Trans. on Neural Networks and Learning Systems,* vol. 28, no. 5, pp. 1152-1163, 2017.

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 joined the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China in 2016, as a professor of computer science and engineering. In 2018, he joined the Dongguan University of Technology, Dongguan, Guangdong, China, as a distinguished professor of computer science. His research interests include big data analysis, artificial intelligence and intelligent control. He has published more than 90 papers (including 20+ IEEE Transactions papers) in his related areas.



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



Shuai Li (M'14-SM'17) received the B.E. degree in Precision Mechanical Engineering from Hefei University of Technology, China in 2005, the M.E. degree in Automatic Control Engineering from University of Science and Technology of China, China in 2008, and the Ph.D. degree in Electrical and Computer Engineering from Stevens Institute of Technology, USA in 2014. He is currently with Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong. His current research interests include dynamic neural networks, recommender systems, cyber-physical systems and multi-robotics.

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

Zidong Wang (SM'03-F'14) was born in Jiangsu, China, in 1966. He received the B.Sc. degree in mathematics from Soochow University, Suzhou, China, and the M.Sc. degree in applied mathematics and the Ph.D. degree in electrical engineering from Nanjing University of Science and Technology, Nanjing, China, in 1986, 1990, and 1994, respectively.

He is currently a Professor of dynamical systems and computing in the Department of Information Systems and Computing, Brunel University London, Middlesex, U.K. From 1990 to 2002, he held teaching and research appointments in universities in China, Germany, and the U.K. He has published more than 300 papers in refereed international journals. He is a holder of the Alexander von Humboldt Research Fellowship of Germany, the JSPS Research Fellowship of Japan, and the William Mong Visiting Research Fellowship of Hong Kong. His research interests include dynamical systems, signal processing, bioinformatics, and control theory and applications. Prof. Wang serves (or has served) as the Editor-in-Chief for Neurocomputing and an Associate Editor for 12 international journals, including the IEEE TRANSACTIONS ON AUTOMATIC CONTROL, the IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, the IEEE TRANSACTIONS ON NEURAL NETWORKS, the IEEE TRANSACTIONS ON SIGNAL PROCESSING, and the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS--Part C. He is a Fellow of the Royal Statistical Society and a member of program committee for many international conferences.