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: DOI 10.1109/TKDE.2020.3033324, IEEE Transactions on Knowledge and Data Engineering

# Position-Transitional Particle Swarm Optimization-Incorporated Latent Factor Analysis

Xin Luo, *Senior Member, IEEE*, Ye Yuan, Sili Chen, Nianyin Zeng, *Member, IEEE*, and Zidong Wang, *Fellow*, *IEEE*

**Abstract**—High-dimensional and sparse (HiDS) matrices are frequently found in various industrial applications. A latent factor analysis (LFA) model is commonly adopted to extract useful knowledge from an HiDS matrix, whose parameter training mostly relies on a stochastic gradient descent (SGD) algorithm. However, an SGD-based LFA model's learning rate is hard to tune in real applications, making it vital to implement its self-adaptation. To address this critical issue, this study firstly investigates the evolution process of a particle swarm optimization algorithm with care, and then proposes to incorporate more dynamic information into it for avoiding accuracy loss caused by premature convergence without extra computation burden, thereby innovatively achieving a novel position-transitional particle swarm optimization ($P^2SO$) algorithm. It is subsequently adopted to implement a $P^2SO$-based LFA (PLFA) model that builds a learning rate swarm applied to the same group of LFs. Thus, a PLFA model implements highly efficient learning rate adaptation as well as represents an HiDS matrix precisely. Experimental results on four HiDS matrices emerging from real applications demonstrate that compared with an SGD-based LFA model, a PLFA model no longer suffers from a tedious and expensive tuning process of its learning rate, and it can achieve even higher prediction accuracy for missing data of an HiDS matrix. On the other hand, compared with state-of-the-art adaptive LFA models, a PLFA model's prediction accuracy and computational efficiency are highly competitive. Hence, it has high potential in addressing real industrial issues.

**Index Terms**—Data Science, Computational Intelligence, Learning Rate Adaptation, Hyper Parameter Adaptation, Latent Factor Analysis, Particle Swarm Optimization, High-dimensional and Sparse Data, Adaptive Algorithm, Industrial Application

## 1 Introduction

IN MANY BIG-DATA-RELATED FIELDS like wireless sensor networks [1-3], bioinformatic applications [4-6], social networks [7-9] and electronic commerce systems [10-12], people usually encounter a mass of entities and their high-dimensional and sparse (HiDS) relationships [10-12]. An HiDS matrix is commonly adopted to describe such specific relationships among them [4-15], where only a small fractions of its entries are given and the most others are unknown.

Although an HiDS matrices is extremely sparse, it involves a mountain of valuable information regarding various patterns. For instance, a user-item matrix generated by an electronic commerce system describes user-item preferences according to users' historical experiences [10-12]. In order to extract such knowledge from an HiDS matrix, researchers have made great efforts to develop various sophisticated models. Representative ones include a probabilistic MF model [16], a multi-dimensional probabilistic model [17], an SVD++ [18], a non-parametric Bayesian-based latent factor model [19], and a weighted trace-norm regularization-based model [20]. He *et.al* [64] propose a co-learning LFA model for graph analysis, which efficiently figures out an optimal arrangement of clusters for vertices in an attributed graph by formulating the task as a fuzzy constrained optimization problem. Meanwhile, He *et.al* [65] propose a multi-view LFA model, which uses an effective multi-view learning scheme [65] to learn the latent space from multi-view vertex features, there modeling the inter-relationship between pairwise vertices.

Considering existing HiDS matrix analysis models, a latent factor analysis (LFA) model is very popular owing to its high scalability and efficiency [21-25, 47, 48]. An LFA model essentially targets at constructing a HiDS matrix's low-rank approximation based on its known data only in the following working-flow:

a) Mapping the row and column entities of a target HiDS matrix into a unique and low-dimensional LF space;
b) Constructing a learning objective on an HiDS matrix's observed data and their related LFs;

✧ *This research is supported in part by the National Natural Science Foundation of China under grants 61772493 and 61933007, in part by the Guangdong Province Universities and College Pearl River Scholar Funded Scheme (2019), and in part by the Natural Science Foundation of Chongqing (China) under grant cstc2019jcyjjqX0013 (Corresponding authors: N. Zeng and Z. Wang).*

✧ *X. Luo is with the School of Computer Science and Technology, Dongguan University of Technology, Dongguan, Guangdong 523808, China, and also with the Hengrui (Chongqing) Artificial Intelligence Research Center, Department of Big Data Analyses Techniques, Cloudwalk, Chongqing 401331, China (e-mail: luoxin21@gmail.com).*

✧ *Ye Yuan and S. Chen are with the Chongqing Engineering Research Center of Big Data Application for Smart Cities, and Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China (e-mail: yeyuan@cigit.ac.cn, chensili06@outlook.com).*

✧ *N. Zeng is with the Department of Instrumental and Electrical Engineering, Xiamen University, Xiamen 361005 China. (e-mail: zny@xmu.edu.cn).*

✧ *Z. Wang is with the Department of Information Systems and Computing, Brunel University, Uxbridge, Middlesex UB8 3PH, U.K. (e-mail: Zidong.Wang@brunel.ac.uk).*

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: DOI 10.1109/TKDE.2020.3033324, IEEE Transactions on Knowledge and Data Engineering

c) Minimizing the learning objective with the desired LFs to make them precisely represent the known data of an HiDS matrix; and
d) Constructing the desired low-rank approximation to the target HiDS matrix with the achieved LFs.

As illustrated by prior research [22, 25, 26], a stochastic gradient descent (SGD) algorithm is highly efficient in building an LFA model. However, the performance of an SGD-based LFA model is greatly influenced by its learning rate [16, 25, 26]. An inappropriately large learning rate makes the model diverge while a too small one leads to slow model convergence, i.e., many iterations are required to make the model converge. Note that an HiDS matrix generated by an industrial application can be huge. For instance, the Douban matrix collected by the Chinese largest online book, movie and music database Douban [27] has 16,830,839 entries scattering in 129,490 rows and 58,541 columns. Although its data density is 0.22% only, it contains 7.58 billion instances. When handling such a large-scale HiDS matrix, it becomes very difficult and expensive in time and computation to manually tune the learning rate of an SGD-based LFA model.

Great efforts have been paid to address the issue of learning rate adaptation in an SGD algorithm [28-30]. Duchi *et.al* [28] propose the AdaGrad algorithm that automatically adjusts the learning rate according to the sum squares of past gradients at each update point. Zeiler *et.al* [29] propose the AdaDelta algorithm that implements learning rate adaptation based on the decaying average of all past squared gradients instead of the accumulated squared gradients adopted by AdaGrad. Li *et.al* [30] propose the AdaError algorithm that is more robust with noises. It adopts smaller learning rate to cope with lager training errors, and vice versa. Kingma and Ba [49] propose the widely-adopted Adam algorithm. It dynamically estimates the learning rate based on the exponentially decaying square average and exponentially decaying average of past stochastic gradients, as well as dynamically adjusts the learning direction according to last update. It is also frequently adopted to build a deep neural network.

Although the aforementioned methods correctly implement leaning rate adaptation in an SGD algorithm, they drastically increases the time cost per iteration when building a learning model. This defect is even more significant when applying them to building an LFA model on an HiDS matrix. As unveiled in [12], AdaDelta, AdaGrad and Adam all make an LFA model's time cost per iteration increase significantly. Although they indeed implement learning rate adaption as well as accelerate an LFA model's convergence rate, i.e., make it converge with less iterations, they still lead to increase of total time cost [12]. It should be pointed out that with its efficient learning rate adaptation, an Adam-like algorithm actually outperforms a standard SGD-based LFA model in terms of computational efficiency in spite of the aforementioned defect. This is because to achieve the best performance, a standard SGD-based LFA model calls for manual tuning of its learning rate, which is extremely time-consuming. However, can we find another path to learning rate adaptation in an SGD-based LFA model without additional computation burden?

As unveiled by prior study [31], a particle swarm optimization (PSO) algorithm can enable the adaptation of hyper parameters in a learning algorithm. Meanwhile, a PSO algorithm further has the advantages of high efficiency and compatibility [32-34, 53-59]. Nonetheless, a standard PSO algorithm often suffers from premature convergence when dealing with complex tasks [35-37], such as the bi-convex problem in an LFA model. To address this defect of a PSO algorithm, various efforts have been made, resulting in a pyramid of PSO extensions. They can be mainly divided into two categories:

a) Hyper-parameter incorporation. Ratnaweera *et al*. [54] incorporate time-varying acceleration coefficients and inertia weights into a PSO algorithm to efficiently control its local search and global convergence. Dong *et al*. [55] model PSO as a dynamic quadratic programming model with box constraints and set its parameters with a dynamic forgetting factor to improve its exploration competence.
b) Searching strategy adjustment. Liang *et al*. [56] adopt particles' historical information to improve the swarm searching ability. Zhan *et al*. [57] propose an orthogonal learning strategy to adjust the flying direction of each particle for enhancing the swarm searching ability. Behnamian and Ghomi. [58] propose to compoundthe population-based evolutionary searching and local searching of simulated annealing to enhance the searching ability of PSO. Chen *et al*. [60] integrate an aging strategy into the particle evolution for addressing the problem of premature convergence. Xia *et al*. [62] propose a multi-swarm PSO algorithm that well balances the exploration and exploitation during the swarm searching process. Gong *et al*. [59] adopt genetic operators to generate exemplars that interact with particles bi-directionally, thereby enhancing the swarm searching ability. Li *et al*. [61] enable information sharing among particles for enhancing their mutual interaction, thereby improving the swarm searching ability.

The aforementioned PSO extensions, in spite of their efficiency, all perplex the evolution process of a standard PSO algorithm with more parameters or more complex searching strategies. When applying them to a learning model, a lot of tuning tasks are necessary to make them function. When the target model is computationally expensive to build (like an LFA model defined on a large-scale HiDS matrix), such tuning can be highly expensive. Thus, the critical issue of complexity again arises: is it possible to implement a PSO algorithm with good searching ability without increasing its complexity, thereby building an LFA model with efficient learning rate adaptation?

To answer this question, this study presents a position-transitional particle swarm optimization (P$^2$SO) algorithm, which integrates more 'dynamic' information into the evolution rule of each particle for enhancing the vitality of the whole swarm, preventing it from premature convergence. Then we integrate P$^2$SO into an SGD-based LFA model for making its learning rate self-adaptive, and further propose a P$^2$SO-based LFA (PLFA) model with high computational efficiency and prediction accuracy. More

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: DOI 10.1109/TKDE.2020.3033324, IEEE Transactions on Knowledge and Data Engineering

importantly, it does not bring additional computation burden into its evolution process, thereby achieving high efficiency. Main contributions of this paper include:

a) A P²SO algorithm is proposed. It amplifies the effect of evolution velocity in a PSO algorithm, thereby improving the swarm searching ability. A P²SO algorithm successfully overcomes the defect of premature in a standard PSO algorithm without perplexing its evolution process, thereby exhibits ease of implementation and good compatibility with complex and expensive models like an LFA model concerned in this study;
b) Based on a P²SO algorithm, a PLFA model is proposed. It makes its learning rate self-adaptive, thereby avoiding expensive and tedious tuning process for its learning rate. Owing to the good searching ability and high computational efficiency of a P²SO algorithm, a PLFA model's computational efficiency and prediction accuracy for missing data of an HiDS matrix is ensured;
c) Theoretical convergence analysis for a PLFA model; and
d) Detailed algorithm design and analysis for a PLFA model, illustrating that it is highly efficient in both time and storage.

Empirical studies on four HiDS matrices from industrial applications demonstrate that when compared with LFA models relying on an SGD algorithm, a standard PSO-incorporated SGD algorithm and an Adam algorithm, a PLFA model possesses the following virtues:

a) Its learning rate adaptation is implemented more efficiently, which makes its computational efficiency highly competitive. Its time cost is much less than an LFA model relying on standard SGD or Adam, in spite that the latter is a highly popular algorithm with learning rate adaptation; and
b) Its prediction accuracy for missing data of an HiDS matrix is impressive. It generates more accurate predictions for missing data than its peers do on most testing cases, indicating that the enhanced swarm searching ability of a P²SO algorithm enables an PLFA model to represent an HiDS matrix more precisely.

According to the authors' best knowledge, a PLFA model implements learning rate adaptation in an SGD-based LFA model more efficiently than any other models do. Section 2 gives the preliminaries. Section 3 presents the methods. Section 4 provides and analyses the empirical study results. Finally, Section 5 concludes this paper.

## 2 Preliminaries

### 2.1 Problem Statement

Firstly, let us review the following definitions:

***Definition* 1: *An HiDS matrix.*** Given two large entity sets $U$ and $I$, $Z$ is a $|U|\times|I|$ matrix where each entry $z_{u,i}$ denotes a certain relationship between entities $u\in U$ and $i\in I$. Let $\Lambda$ and $\Gamma$ denote the known and unknown entry sets of $Z$, $Z$ is an HiDS matrix if $|\Lambda|\ll|\Gamma|$.

***Definition* 2: *An LFA model.*** Given $Z$ and $\Lambda$, an LFA model constructs an estimator $\hat{Z}=PQ^{\mathrm{T}}$ with rank-$f$ LF matrices $P$ and $Q$, where $P$ is $|U|\times f$, $Q$ is $|I|\times f$ and $f\ll\min\{|U|,|I|\}$.

Note that $f$ represents the dimension of an LF space, $P$ and $Q$ consist of LFs reflecting characteristics of $U$ and $I$ described by $\Lambda$, respectively.

To acquire LFs hidden in $\Lambda$, an objective function is desired. We commonly adopt the Euclidean distance to model the objective function [21, 22] for generality (while our methods are adaptable with other objective functions):

$$\varepsilon\left(P,Q\right)=\sum_{z_{u,i}\in\Lambda}\left(z_{u,i}-p_u q_i^{\mathrm{T}}\right)^2, \tag{1}$$

where $z_{u,i}$ denotes a specified element in $Z$, $p_u$ and $q_i$ denote the $u$-th and $i$-th row vector of $P$ and $Q$, respectively.

As described in prior study [16, 21, 22, 25, 26], to perform LFA on an HiDS matrix is ill-posed. Consequently, it is essential to apply regularization to its objective function for avoiding overfitting. The Tikhonov regularization is a widely-accepted choice for such a purpose [21, 22]. Let $\hat{z}_{u,i}=p_u q_i^{\mathrm{T}}$, with the Tikhonov regularization (1) is extended as:

$$\varepsilon\left(P,Q\right)=\sum_{z_{u,i}\in\Lambda}\left(\left(z_{u,i}-\hat{z}_{u,i}\right)^2+\lambda_P\left\|p_u\right\|_2^2+\lambda_Q\left\|p_u\right\|_2^2\right), \tag{2}$$

where $\lambda_P$ and $\lambda_Q$ are regularization constants.

### 2.2 An SGD-based LFA Model

When performing LFA on an HiDS matrix, an SGD algorithm enjoys its fast convergence and ease of implementation [22, 25, 26]. With it, (2) is minimized with desired LF matrices $P$ and $Q$ as:

$$\arg\min_{P,Q}\varepsilon\left(P,Q\right)\overset{SGD}{\Rightarrow}\forall z_{u,i}\in\Lambda:\begin{cases}p_u^{\tau}\leftarrow p_u^{\tau-1}-\eta\cdot\nabla\varepsilon_{u,i}\left(p_u^{\tau-1}\right),\\ q_i^{\tau}\leftarrow q_i^{\tau-1}-\eta\cdot\nabla\varepsilon_{u,i}\left(q_i^{\tau-1}\right).\end{cases} \tag{3}$$

where $\tau$ and $(\tau\text{-}1)$ denote the $\tau$-th and $(\tau\text{-}1)$-th update points by SGD, $\varepsilon_{u,i}=\left(z_{u,i}-p_u q_i^{\mathrm{T}}\right)^2+\lambda_P\left\|p_u\right\|_2^2+\lambda_Q\left\|q_i\right\|_2^2$ denotes the instant error on the training instance $z_{u,i}\in\Lambda$ and LF vectors $p_u$ and $q_i$, $\eta$ is the learning rate, respectively. By making $e_{u,i}=z_{u,i}-p_u q_i^{\mathrm{T}}$ and expanding the gradients in (3), we achieve the following learning scheme:

$$\begin{aligned}&\arg\min_{P,Q}\varepsilon\left(P,Q\right)\overset{SGD}{\Rightarrow}\forall z_{u,i}\in\Lambda:\\ &\begin{cases}p_u^{\tau}\leftarrow p_u^{\tau-1}+\eta\cdot\left(e_{u,i}^{\tau-1}\cdot q_i^{\tau-1}-\lambda_P\cdot p_u^{\tau-1}\right),\\ q_i^{\tau}\leftarrow q_i^{\tau-1}+\eta\cdot\left(e_{u,i}^{\tau-1}\cdot p_u^{\tau-1}-\lambda_Q\cdot q_i^{\tau-1}\right).\end{cases}\end{aligned} \tag{4}$$

As proven in [38, 39], an SGD-based LFA model converges well with appropriate chosen values of $\eta$, which actually arises from the Robbins-Siegmund theorem [40].

### 2.3 A Standard PSO Algorithm

A standard PSO is an evolutionary meta-heuristics algorithm adopted in many fields [32-37]. In it, $S$ particles of a swarm fly around $D$-dimensional search space at a certain speed. Each particle serves as a potential solution to the current optimization task. Their movements depend on two factors, i.e., their velocity and position. More specifically, the velocity and position of the $j$-th particle at the $t$-th iteration are denoted by two vectors, i.e., $v_j(t)=(v_{j,1}(t), v_{j,2}(t), \ldots, v_{j,D}(t))$ and $x_j(t)=(x_{j,1}(t), x_{j,2}(t), \ldots, x_{j,D}(t))$ where $v_{j\max}=(v_{j\max,1}, v_{j\max,2}, \ldots, v_{j\max,D})$, $x_{j,d}(t)\in[x_{\min,d}, x_{\max,d}]$, and $1\le d\le D$. Note that

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: DOI 10.1109/TKDE.2020.3033324, IEEE Transactions on Knowledge and Data Engineering

$v_{j\max}$ is the maximum velocity of the whole dimension, $x_{\min,d}$ and $x_{\max,d}$ are lower and upper bounds for the $d$-th dimension, respectively.

During this evolution process, each particle determines its next flying through its own flying experience and the best flying experience among its peers. One is named *pbest*, i.e., $pb_j=(pb_{j,1}, pb_{j,2}, \ldots, pb_{j,D})$, which is the best position discovered by itself, and the other is named *gbest*, i.e., $gb=(gb_1, gb_2,\ldots,gb_D)$, which is the best position in the whole swarm. Thus, the $j$-th particle evolves at the $t$-th iteration as:

$$\begin{aligned} v_j(t) &= wv_j(t-1)+c_1r_1\left(pb_j(t-1)-x_j(t-1)\right) \\ &\quad +c_2r_2\left(gb(t-1)-x_j(t-1)\right), \\ x_j(t) &= x_j(t-1)+v_j(t); \end{aligned} \tag{5}$$

where $w$ is the non-negative inertia constant, $c_1$ and $c_2$ are cognitive and social coefficients, $r_1$ and $r_2$ are two uniform random numbers in the range of [0, 1], respectively.

## 3 METHODS

### 3.1 A P²SO Algorithm

In a standard PSO algorithm relying on (5), each particle's evolution depends on three learning factors:

a) Its latest evolution velocity $v_j(t\text{-}1)$,
b) An adjusting velocity from its latest position $x_j(t\text{-}1)$ to its latest locally optimal position $pb_j(t\text{-}1)$, and
c) An adjusting velocity from $x_j(t\text{-}1)$ and the latest globally optimal position of all particles $gb(t\text{-}1)$.

Thus, each particle evolves based on its latest evolution velocity which reflects the historically learning inertia, and flies towards its historically best position and the globally best position of the whole swarm, thereby making the whole swarm finally achieve a globally steady solution.

Meanwhile, randomness is injected into this process to strengthen the searching ability of each particle, preventing the whole swarm from being stacked by some local solution. In other words, a standard PSO algorithm makes its particles evolve based on their own experience and swarm experience simultaneously. However, during its evolution process, the 'statistic' positions, i.e., $x_j(t\text{-}1)$, $pb_j(t\text{-}1)$ and $gb(t\text{-}1)$, heavily affect the evolution of each particle, making the whole swarm suffer from the premature issue, i.e., become steady too early. From this point of view, we propose to incorporate more 'dynamic' information into the evolution scheme of each particle. Thus, we make the $j$-th particle evolve at the $t$-th iteration as:

$$\begin{cases} v_j(t) = wv_j(t-1)+c_1r_1\left(pb_j(t-1)-x_j(t-1)\right) \\ \qquad +c_2r_2\left(gb(t-1)-x_j(t-1)\right) \\ \qquad +\rho\left(c_1r_1+c_2r_2\right)\left(x_j(t-1)-v_j(t-1)\right), \\ x_j(t) = x_j(t-1)+v_j(t); \end{cases} \tag{6}$$

where $\rho$ denotes a non-negative coefficient in the range of [0, 1]. Next we will analyze the effect of (6) in the cases of $\rho$=0, $\rho$=1 and 0<$\rho$<1 in detail.

#### 3.2.1 Case $\rho$=0

By substituting $\rho$=0 into (6), we clearly achieve (5), i.e., a standard PSO algorithm. As analyzed before, its evolution of each particle in this case is mostly decided by the adjusting velocities, i.e., ($pb_j(t\text{-}1)$-$x_j(t\text{-}1)$) and ($gb(t\text{-}1)$-$x_j(t\text{-}1)$), as shown in Fig. S1(a) in the Supplementary File. Hence, each particle tends to approach its local optimal position and global optimal position of the swarm quickly, making the whole swarm suffer from premature convergence.

#### 3.2.2 Case $\rho$=1

By substituting $\rho$=0 into (6), we achieve another border case as follows:

$$\begin{cases} v_j(t) = wv_j(t-1)+c_1r_1\left(pb_j(t-1)\right)-c_1r_1v_j(t-1) \\ \qquad +c_2r_2\left(gb(t-1)\right)-c_2r_2v_j(t-1), \\ x_j(t) = x_j(t-1)+v_j(t); \end{cases} \tag{7}$$

With (7), the evolution of each particle depends on the following factors: a) the latest evolution velocity, i.e., $v_j(t\text{-}1)$, b) the adjusting velocity decided by the direction vector from the origin point to the locally optimal position and the latest evolution velocity, i.e., ($pb_j(t\text{-}1)$-0) and $v_j(t\text{-}1)$, and c) the adjusting velocity decided by the direction vector from the origin point to the globally optimal position and the latest evolution velocity, i.e., ($gb(1)$-0) and $v_j(t\text{-}1)$. The whole process is illustrated in Fig. S1(b) in the Supplementary File. Note that as shown in Fig. S1(b), with (7) more dynamic information relying on $v_j(t\text{-}1)$ is injected into the particle evolution, thereby making the whole swarm more active to avoiding premature convergence.

#### 3.2.3 Case 0<$\rho$<1

When $\rho$ lies in the scale of (0, 1), (6) is reformulated as:

$$\begin{cases} v_j(t) = wv_j(t-1)+c_1r_1\left(pb_j(t-1)-x_j(t-1)\right) \\ \qquad +c_2r_2\left(gb(t-1)-x_j(t-1)\right) \\ \qquad +\rho c_1r_1\left(x_j(t-1)-v_j(t-1)\right) \\ \qquad +\rho c_2r_2\left(x_j(t-1)-v_j(t-1)\right), \\ x_j(t) = x_j(t-1)+v_j(t); \end{cases} \tag{8}$$

where we clearly see the combination of (5) and (7): the evolution velocity does not only depend on the adjusting velocities decided by the static positions, but also depend on the dynamic evolution velocities as analyzed in the previous section. Its evolution process is depicted in Fig. S1(c) in the Supplementary File. From it, we see that with 0<$\rho$<1 we arrive at an intermediate state between two extreme cases of $\rho$=0 and $\rho$=1.

#### 3.2.4 Summary

Based on the above inferences, we achieve the particle evolution scheme for an P²SO algorithm. It contains a standard PSO algorithm as one of its special case. It also brings more dynamic information into the evolution process for preventing the swarm from premature convergence when $\rho\neq0$. With such characteristics, next we implement efficient learning rate adaptation in a PLFA model.

### 3.3 A PLFA Model

Following the principle of a P²SO algorithm, we

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: DOI 10.1109/TKDE.2020.3033324, IEEE Transactions on Knowledge and Data Engineering

manipulate a swarm consisting of $S$ particles, where the $j$-th particle is the learning rate $\eta_j$ for the same group of LFs. Then following (6), $\eta_j$'s evolution scheme is given by:

$$\forall j \in \{1,\ldots,S\}: \begin{cases} v_j(t) = wv_j(t-1) + c_1 r_1\left(pb_j(t-1) - \eta_j(t-1)\right) \\ \qquad + c_2 r_2\left(gb(t-1) - \eta_j(t-1)\right) \\ \qquad + \rho\left(c_1 r_1 + c_2 r_2\right)\left(\eta_j(t-1) - v_j(t-1)\right), \\ \eta_j(t) = \eta_j(t-1) + v_j(t). \end{cases} \tag{9}$$

Thus, each particle searches for the optimal solution in an one-dimensional space for the learning rate only.

According to prior research [31], the velocity and searching space of each particle is constrained to be in a certain range:

$$v_j^t = \begin{cases} v_{\max}, \text{ if } v_j^t > v_{\max}, \\ v_{\min}, \text{ if } v_j^t < v_{\min}; \end{cases} \quad \eta_j^t = \begin{cases} \eta_{\max}, \text{ if } \eta_j^t > \eta_{\max}, \\ \eta_{\min}, \text{ if } \eta_j^t < \eta_{\min}. \end{cases} \tag{10}$$

where $v_{\max}$ and $v_{\min}$ denote the upper and lower bounds for $v$, $\eta_{\max}$ and $\eta_{\min}$ denote the upper and lower bounds for $\eta$, respectively. Note that these threshold values are mostly empirical and problem-dependent. In an PLFA model, we set $v_{\max}$=1, $v_{\min}$=-1, $\eta_{\max}$=$2^{-8}$, and $\eta_{\min}$=$2^{-12}$, respectively.

For better acquiring LFs from HiDS matrices, we adopt the following two fitness functions for the $j$-th particle:

$$F_{1(j)} = \sqrt{\left(\sum_{z_{u,i}\in\Omega}\left(z_{u,i} - \hat{z}_{(j)u,i}\right)^2\right) \Big/ |\Omega|}, \quad F_{2(j)} = \left(\sum_{z_{u,i}\in\Omega}\left|z_{u,i} - \hat{z}_{(j)u,i}\right|_{abs}\right) \Big/ |\Omega|; \tag{11}$$

where $|\cdot|_{abs}$ calculates the absolute value of a given value, $\Omega$ denotes the validation set and is disjoint with $\Lambda$, and $\hat{z}_{(j)u,i}$ denotes the estimation value regarding to the known element $z_{u,i}\in\Omega$ generated by the $j$-th particle, respectively. To be shown later, a PLFA model adopts $F_{1(j)}$ or $F_{2(j)}$ according to the performance evaluation metrics.

Note that $\forall j\in\{1,\ldots,S\}$, $\eta_j$ is linked with the same group of LF matrices, i.e., $P$ and $Q$, which are trained by an SGD algorithm. Thus, its $t$-th iteration actually consists of $S$ sub-iterations, where $P$ and $Q$ are updated in the $j$-th sub-iteration as follows:

$$\arg\min_{P,Q}\varepsilon(P,Q) \overset{P^2SO-SGD}{\Rightarrow} \forall z_{u,i}\in\Lambda: \begin{cases} p_{(j)u}^{\tau} \leftarrow p_{(j)u}^{\tau-1} + \eta_j^{t-1}\cdot\left(e_{(j)u,i}^{\tau-1}\cdot q_{(j)i}^{\tau-1} - \lambda_P\cdot p_{(j)u}^{\tau-1}\right), \\ q_{(j)i}^{\tau} \leftarrow q_{(j)i}^{\tau-1} + \eta_j^{t-1}\cdot\left(e_{(j)u,i}^{\tau-1}\cdot p_{(j)u}^{\tau-1} - \lambda_Q\cdot q_{(j)i}^{\tau-1}\right). \end{cases} \tag{12}$$

where the footnote $(j)$ on $p_u$, $q_i$ and $e_{u,i}$ denotes that their current updates are linked with the $j$-th particle, i.e., $\eta_j$. Based on (9)-(12), we achieve a PLFA model for HiDS matrices with learning rate adaptation. Next we analyze its convergence in theory.

### 3.4 Convergence Analysis for a PLFA Model

We start with recalling the definitions of L-smooth and strong convexity of a continuously differentiable function $f(x)$ [66] as follows.

***Definition* 3: *L-Smooth***. $f(x)$ is L-smooth if $\forall x, y\in R^f$ the following inequality holds:

$$\left\|\nabla f(x) - \nabla f(y)\right\|_2 \le L\left\|x - y\right\|_2. \tag{13}$$

***Definition* 4: *Strongly Convex.*** $f(x)$ is strongly convex if there exists a constant $\beta>0$ such that $\forall x, y\in R^f$ the following inequality holds:

$$f(x) \ge f(y) + \nabla f(y)(x-y)^{\mathrm{T}} + \frac{1}{2}\beta\left\|x - y\right\|_2^2. \tag{14}$$

Then in order to conveniently analyze PLFA's convergence, we reformulate the objective function (2) as:

$$\varepsilon = \sum_{z_{u,i}\in\Lambda}\left(\left(z_{u,i} - p_u q_i^{\mathrm{T}}\right)^2 + \lambda_P\left\|p_u\right\|_2^2 + \lambda_Q\left\|q_i\right\|_2^2\right) = \sum_{z_{u,i}\in\Lambda}\varepsilon_{u,i}, \tag{15}$$

where $\varepsilon_{u,i} = \left(z_{u,i} - p_u q_i^{\mathrm{T}}\right)^2 + \lambda_P\left\|p_u\right\|_2^2 + \lambda_Q\left\|q_i\right\|_2^2$ denotes the instant loss related with a single instance $z_{u,i}\in\Lambda$ and LF vector $p_u$ and $q_i$.

Although PLFA makes its learning rate self-adaptive with P²SO, it still adopts SGD-based learning scheme to update the desired LFs. Hence, we aim to show that with this P²SO-incoporated and SGD-based learning scheme, a PLFA model converges. Meanwhile, note that its learning objective (15) is non-convex. According to prior research in the optimization community [68, 70, 76], necessary relaxations are made to achieve the final proof:

a) We focus on each instant loss $\varepsilon_{u,i}$ instead of the sum loss $\varepsilon$ according to the single-pass of an SGD algorithm;
b) We analyze the model convergence with respect to an LF vector while fixing its coupled LF vectors in the non-convex terms of (15). For instance, $q_i$ is fixed to show the model convergence with the update of $p_u$. Note that the model convergence with the update of $q_i$ can be achieved in the same way.

Then considering $\varepsilon_{u,i}$, we first have the following lemmas.

***Lemma* 1**. *$\varepsilon_{u,i}$ is L-smooth when L is the maximum singular value of* $(q_i^{\mathrm{T}}q_i+\lambda_P I_f)$ as $I_f$ be an $f\times f$ identity matrix.

*Proof*. By replacing $p_u$ in $\varepsilon_{u,i}$ with arbitrary vectors $p_g$ and $p_h$ which are independent of each other, we have

$$\begin{aligned} &\nabla\varepsilon_{u,i}(p_g) - \nabla\varepsilon_{u,i}(p_h) \\ &= -\left(z_{u,i} - p_g q_i^{\mathrm{T}}\right)q_i + \lambda_P p_g + \left(z_{u,i} - p_h q_i^{\mathrm{T}}\right)q_i - \lambda_P p_h \\ &= \left(p_g - p_h\right)\left(q_i^{\mathrm{T}}q_i + \lambda_P I_f\right), \\ &\Rightarrow \left\|\nabla\varepsilon_i(p_g) - \nabla\varepsilon_i(p_h)\right\|_2 = \left\|\left(p_g - p_h\right)\left(q_i^{\mathrm{T}}q_i + \lambda_P I_f\right)\right\|_2 \\ &\quad \le \left\|\left(p_g - p_h\right)\right\|_2\left\|\left(q_i^{\mathrm{T}}q_i + \lambda_P I_f\right)\right\|_2. \end{aligned} \tag{16}$$

According to the property of a matrix's $L_2$-norm [69-72], the term $\left\|\left(q_i^{\mathrm{T}}q_i + \lambda_P I_f\right)\right\|_2$ denotes the largest singular value of $(q_i^{\mathrm{T}}q_i+\lambda_P I_f)$. Based on the above inferences, *Lemma* 1 stands.□

***Lemma* 2**. *$\varepsilon_{u,i}$ is strongly convex when $\beta$ is the minimum singular value* of $(q_i^{\mathrm{T}}q_i+\lambda_P I_f)$.

*Proof*. Given arbitrary vectors $p_g$ and $p_h$, we consider the state

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: DOI 10.1109/TKDE.2020.3033324, IEEE Transactions on Knowledge and Data Engineering

of $\varepsilon_{u,i}$ at $p_g$, and then consider its its second-order Taylor-expansion at $p_h$ as:

$$\begin{aligned}&\varepsilon_{u,i}\left(p_g\right)\approx\varepsilon_{u,i}\left(p_h\right)+\nabla\varepsilon_{u,i}\left(p_h\right)\left(p_g-p_h\right)^{\mathrm{T}}\\&\qquad+\frac{1}{2}\left(p_g-p_h\right)\nabla^2\varepsilon_{u,i}\left(p_h\right)\left(p_g-p_h\right)^{\mathrm{T}}.\\&\Rightarrow\varepsilon_{u,i}\left(p_g\right)-\varepsilon_{u,i}\left(p_h\right)\\&\quad=\nabla\varepsilon_{u,i}\left(p_h\right)\left(p_g-p_h\right)^{\mathrm{T}}+\frac{1}{2}\left(p_g-p_h\right)\nabla^2\varepsilon_{u,i}\left(p_h\right)\left(p_g-p_h\right)^{\mathrm{T}}.\end{aligned}\tag{17}$$

Then considering *Lemma* 2, it states that

$$\varepsilon_{u,i}\left(p_g\right)-\varepsilon_{u,i}\left(p_h\right)\geq\nabla\varepsilon_i\left(p_h\right)\left(p_g-p_h\right)^{\mathrm{T}}+\frac{1}{2}\beta\left\|p_g-p_h\right\|_2^2.\tag{18}$$

Thus, it is desired to appropriately select $\beta$ to make the following inequality stand,

$$\left(p_g-p_h\right)\nabla^2\varepsilon_{u,i}\left(p_h\right)\left(p_g-p_h\right)^{\mathrm{T}}\geq\beta\left\|p_g-p_h\right\|_2^2.\tag{19}$$

From the expression of $\varepsilon_{u,i}$, we have:

$$\nabla^2\varepsilon_{u,i}\left(p_h\right)=q_i^{\mathrm{T}}q_i+\lambda_P I_f.\tag{20}$$

By substituting (20) into (19), we reformulate it as follows:

$$\left(p_g-p_h\right)\left(q_i^{\mathrm{T}}q_i+\lambda_P I_f-\beta I_f\right)\left(p_g-p_h\right)^{\mathrm{T}}\geq0,\tag{21}$$

which actually requires the matrix ($q_i^{\mathrm{T}}q_i+\lambda_P I_f-\beta I_f$) to be positive semi-definite (PSD). As unveiled by [70-73], when $\beta$ is the minimum singular value of the matrix ($q_i^{\mathrm{T}}q_i+\lambda_P I_f$), ($q_i^{\mathrm{T}}q_i+\lambda_P I_f-\beta I_f$) is PSD. Hence, ***Lemma* 2** stands.□

Thus, considering the $j$-th sub-iteration in the $t$-th iteration of an PLFA model, on $z_{u,i}\in\Lambda$ $p_u$ is updated with the following scheme:

$$p_{(j)u}^{\tau}\leftarrow p_{(j)u}^{\tau-1}-\eta_j^{t-1}\cdot\nabla\varepsilon_{u,i}\left(p_{(j)u}^{\tau-1}\right).\tag{22}$$

Assume that $p_u^*$ is the optimal state of $p_u$, we have

$$\begin{aligned}\left\|p_{(j)u}^{\tau}-p_u^*\right\|_2^2&=\left\|p_{(j)u}^{\tau-1}-\eta_j^{t-1}\nabla\varepsilon_i\left(p_u^{t-1}\right)-p_u^*\right\|_2^2\\&=\left\|p_{(j)u}^{\tau-1}-p_u^*\right\|_2^2-2\eta_j^{t-1}\nabla\varepsilon_i\left(p_{(j)u}^{\tau-1}\right)\left(p_{(j)u}^{\tau-1}-p_u^*\right)^{\mathrm{T}}\\&\quad+\left(\eta_j^{t-1}\right)^2\left\|\nabla\varepsilon_i\left(p_{(j)u}^{\tau-1}\right)\right\|_2^2.\end{aligned}\tag{23}$$

Note that the expectation of (23) is given by:

$$\begin{aligned}\mathrm{E}\left[\left\|p_{(j)u}^{\tau}-p_u^*\right\|_2^2\right]&=\mathrm{E}\left[\left\|p_{(j)u}^{\tau-1}-p_u^*\right\|_2^2\right]-2\eta_j^{t-1}\mathrm{E}\left[\nabla\varepsilon_i\left(p_{(j)u}^{\tau-1}\right)\left(p_{(j)u}^{\tau-1}-p_u^*\right)^{\mathrm{T}}\right]\\&\quad+\left(\eta_j^{t-1}\right)^2\mathrm{E}\left[\left\|\nabla\varepsilon_i\left(p_{(j)u}^{\tau-1}\right)\right\|_2^2\right].\end{aligned}\tag{24}$$

Based on (14) and *Lemma* 2, we achieve that

$$\varepsilon_{u,i}\left(p_u^*\right)\geq\varepsilon_{u,i}\left(p_{(j)u}^{\tau-1}\right)+\nabla\varepsilon_i\left(p_{(j)u}^{\tau-1}\right)\left(p_u^*-p_{(j)u}^{\tau-1}\right)^{\mathrm{T}}+\frac{1}{2}\beta\left\|p_u^*-p_{(j)u}^{\tau-1}\right\|_2^2.\tag{25}$$

Moreover, since $p_u^*$ is the optimal state of $p_u$, we further have

$$\varepsilon_{u,i}\left(p_u^*\right)<\varepsilon_{u,i}\left(p_{(j)u}^{\tau-1}\right).\tag{26}$$

Then following *Lemma* 2, we evidently have:

$$\begin{aligned}&\nabla\varepsilon_i\left(p_{(j)u}^{\tau-1}\right)\left(p_u^*-p_{(j)u}^{\tau-1}\right)^{\mathrm{T}}+\frac{1}{2}\beta\left\|p_u^*-p_{(j)u}^{\tau-1}\right\|_2^2>0\\&\Rightarrow\nabla\varepsilon_i\left(p_{(j)u}^{\tau-1}\right)\left(p_{(j)u}^{\tau-1}-p_u^*\right)^{\mathrm{T}}>\frac{1}{2}\beta\left\|p_{(j)u}^{\tau-1}-p_u^*\right\|_2^2.\end{aligned}\tag{27}$$

By substituting (27) into (24), we have:

$$\begin{aligned}&\mathrm{E}\left[\left\|p_{(j)u}^{\tau}-p_u^*\right\|_2^2\right]\\&<\left(1-\eta_j^{t-1}\beta\right)\mathrm{E}\left[\left\|p_{(j)u}^{\tau-1}-p_u^*\right\|_2^2\right]+\left(\eta_j^{t-1}\right)^2\mathrm{E}\left[\left\|\nabla\varepsilon_i\left(p_{(j)u}^{\tau-1}\right)\right\|_2^2\right].\end{aligned}\tag{28}$$

Following [68], let the positive real number $r$ fulfill the following condition:

$$\mathrm{E}\left[\left\|\nabla\varepsilon_i\left(p_{(j)u}^{\tau-1}\right)\right\|_2^2\right]\leq r^2,\tag{29}$$

and substituting (29) into (28), we infer that

$$\mathrm{E}\left[\left\|p_{(j)u}^{\tau}-p_u^*\right\|_2^2\right]<\left(1-\eta_j^{t-1}\beta\right)\mathrm{E}\left[\left\|p_{(j)u}^{\tau-1}-p_u^*\right\|_2^2\right]+\left(\eta_j^{t-1}r\right)^2.\tag{30}$$

Let $\eta_j^{t-1}=\alpha/\beta t$ for a constant $\alpha$>1, then (30) is rewritten as:

$$\mathrm{E}\left[\left\|p_{(j)u}^{\tau}-p_u^*\right\|_2^2\right]<\left(1-\frac{\alpha}{t}\right)\mathrm{E}\left[\left\|p_{(j)u}^{\tau-1}-p_u^*\right\|_2^2\right]+\left(\frac{\alpha}{\beta t}r\right)^2.\tag{31}$$

As discussed in [68], $\mathrm{E}\left[\left\|p_{(j)u}^{\tau}-p_u^*\right\|_2^2\right]$ denotes $p_u$'s expected squared distance to $p_u^*$ at the $\tau$-th update point of the $j$-th sub-iteration in the $t$-th iteration. By expanding the iterative expression of (31), it satisfies the following condition:

$$\mathrm{E}\left[\left\|p_{(j)u}^{\tau}-p_u^*\right\|_2^2\right]\leq\frac{1}{t}\max\left\{\left\|p_{(j)u}^{1}-p_u^*\right\|_2^2,\frac{\alpha^2r^2}{\beta\alpha-1}\right\},\tag{32}$$

where $p_{(j)u}^1$ denotes the initial state of $p_{(j)u}$ at the $j$-th sub-iteration of the $t$-th iteration.

Recall *Lemma* 1, $\varepsilon_{u,i}$ is $L$-smooth. Thus, by utilizing the property of an $L$-smooth function, we have:

$$\varepsilon_{u,i}\left(p_{(j)u}^{\tau}\right)-\varepsilon_{u,i}\left(p_u^*\right)\leq\frac{L}{2}\left\|p_{(j)u}^{\tau}-p_u^*\right\|_2^2.\tag{33}$$

Note that $\varepsilon_{u,i}\left(p_{(j)u}^{\tau}\right)\geq\varepsilon_{u,i}\left(p_u^*\right)$ since $p_u^*$ is the optimal state of $p_u$. So by taking the expectation of (33), we achieve:

$$\mathrm{E}\left[\varepsilon_{u,i}\left(p_{(j)u}^{\tau}\right)-\varepsilon_{u,i}\left(p_u^*\right)\right]\leq\frac{L}{2}\mathrm{E}\left[\left\|p_{(j)u}^{\tau}-p_u^*\right\|_2^2\right].\tag{34}$$

By substituting (32) into (34), we infer the following inequality:

$$\mathrm{E}\left[\varepsilon_{u,i}\left(p_{(j)u}^{\tau}\right)-\varepsilon_{u,i}\left(p_u^*\right)\right]\leq\frac{LQ(\alpha)}{2t},\tag{35}$$

where

$$Q(\alpha)=\max\left\{\left\|p_{(j)u}^{1}-p_u^*\right\|_2^2,\frac{\alpha^2r^2}{\beta\alpha-1}\right\}.\tag{36}$$

Let $\Lambda(u)$ denote the subset of $\Lambda$ related to $u\in U$, then (35) can be expanded with the consideration on $\forall z_{u,i}\in\Lambda(u)$ to achieve:

$$\begin{aligned}&\mathrm{E}\left[\sum_{z_{u,i}\in\Lambda(u)}\left(\varepsilon_{u,i}\left(p_{(j)u}^{\tau}\right)-\varepsilon_{u,i}\left(p_u^*\right)\right)\right]\\&=\sum_{z_{u,i}\in\Lambda(u)}\mathrm{E}\left[\varepsilon_{u,i}\left(p_{(j)u}^{\tau}\right)-\varepsilon_{u,i}\left(p_u^*\right)\right]\leq\frac{\left|\Lambda(u)\right|LQ(\alpha)}{2t}.\end{aligned}\tag{37}$$

Similarly, $\forall u\in U$ whose related LF vector $p_u$ is updated with the learning scheme (22), (37) can be further expanded as follows:

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: DOI 10.1109/TKDE.2020.3033324, IEEE Transactions on Knowledge and Data Engineering

$$\begin{aligned}&\mathrm{E}\left[\sum_{u\in U}\sum_{z_{u,i}\in\Lambda(u)}\left(\varepsilon_{u,i}\left(p_{(j)u}^{\tau}\right)-\varepsilon_{u,i}\left(p_u^*\right)\right)\right]\\&=\sum_{u\in U}\sum_{z_{u,i}\in\Lambda(u)}\mathrm{E}\left[\varepsilon_{u,i}\left(p_{(j)u}^{\tau}\right)-\varepsilon_{u,i}\left(p_u^*\right)\right]\le\frac{|\Lambda|LQ(\alpha)}{2t},\end{aligned}\tag{38}$$

where the last step is achieved with the condition of $\sum_{u\in U}|\Lambda(u)|=|\Lambda|$. With (38), we see that with the learning scheme of PLFA, $p_u$ converges at $p_u^*$ $\forall u\in U$ with sufficient training iterations, i.e., $t$ is big enough. Following the same principle, we can achieve that $q_i$ converges at $q_i^*$ $\forall i\in I$. Thus, we see that an PLFA model converges with its P2SO-incorporated learning scheme (12).

### 3.5 Algorithm Design and Analysis

Based on the inferences previous, we design the Algorithm PLFA in the Supplementary File. From it, we see that the storage cost of a PLFA model mainly depends on two factors: a) auxiliary matrices for LFA, and b) auxiliary vectors for PSO. More specifically, 1) $P^{|M|\times f}$ and $Q^{|N|\times f}$ cache the LFs; 2) $V^{|S|}$ caches the evaluation velocity, $H^{|S|}$ caches the latest position, and $pb^{|S|}$ caches the best local position of each particle. Thus, a PLFA model's storage cost is given as:

$$S=\Theta\left(\left(|M|+|N|\right)\times f+3S\right)\approx\Theta\left(\left(|M|+|N|\right)\times f\right),\tag{39}$$

which is linear with the number of involved entities.

On the other hand, via the cost analysis of each step in Algorithm PLFA, we achieve its time complexity as follows:

$$\begin{aligned}T&=\Theta\left(\left(|M|+|N|\right)\times f+S+2S\times D+C\times\left(S\times|\Lambda|\times 3f+14S\right)\right)\\&\approx\Theta\left(|\Lambda|\times C\times S\times f\right),\end{aligned}\tag{40}$$

where the last step is obtained by eliminating the constants and lower-order terms. Hence, the time complexity of PLFA is linearly related to $|\Lambda|$. Based on the above analysis, a PLFA model has proven to be highly efficient in both computation and storage.

## 4 Experimental Results and Analysis

### 4.1 General Settings

**Evaluation protocol.** In industrial applications [1-12], in order to recover the unknown relationships among involved entities, it is highly significant to precisely estimate missing data of an HiDS matrix. Thus, the task of missing data prediction is adopted as the evaluation protocol.

**Evaluation Metrics.** For a tested model, root mean squared error (RMSE) and mean absolute error (MAE) are frequently used to validate a tested model's prediction accuracy for missing data of an HiDS matrix [4, 12, 13, 16, 21, 22, 47, 48]:

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

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

where $\Gamma$ denotes the testing set, which is disjoint with $\Lambda$ and $\Omega$. Note that we set $F_{1(j)}$ as the fitness function for PLFA with RMSE as the evaluation metric, and $F_{2(j)}$ with MAE.

Meanwhile, the computational efficiency of tested models is concerned. So we record the converging time cost of each model. All experiments are performed on the same bare machine with an Xeon 2.40-GHz E5-2680 CPU, 512GB memory, and Windows 7 OS. All tested models are executed in JAVA SE 7U60.

**Datasets.** We adopt four HiDS datasets in the experiments.

a) D1: MovieLens 10M. It includes 10,000,054 ratings of 10,681 various movies by 71,567 users. Its rating scale is in the range of [0.5, 5], and rating density is 1.31%. It is from the MovieLens system [41] maintained by the GroupLens research team.
b) D2: Extended Epinion. It contains 13,668,320 observed comments from 120,492 users on 775,760 articles, which is collected by Trustlet website [42]. Its rating scale is [1, 5]. Its density is only 0.015%.
c) D3: Flxiter. It has a density of 0.11% only and a rating scale of [0.5, 5]. It includes 8,196,077 known entries from 147,612 on 48,794 movies, which is collected from the Flixter commercial website [43].
d) D4: Douban. Collected from Chinese largest online book, movie and music database Douban [27], it includes 16,830,839 known entries in the range of [1, 5], by 129,490 on 58,541 items. It has density of 0.22% only.

Note that all datasets are also extremely sparse and high-dimensional. Hence, the experimental results on them are highly representative.

All HiDS matrices are randomly divided into ten disjoint and equally-sized subsets. In each experiment, we adopt the 70%-10%-20% train-validation-test settings, i.e., we adopt seven subsets as training set $\Lambda$ to train a model, one as validation set $\Omega$ to validate the training process, and finally, the last two as the testing set $\Gamma$ for showing the performance of each model. Ten-fold cross-validation settings are applied to such a process for objective results.

The training process of a tested model terminates if a) its consumed iteration counts reaches the preset maximum, i.e., 1000; b) the model converges, i.e., the error difference between two consecutive iterations is smaller than $10^{-5}$.

**Model Settings.** In order to obtain objective results, we adopt the following settings:

a) The performance of an LFA model is easily influenced by the LF dimension. Thus, the LF dimension of each model involved in the experiments is set at 20 uniformly, and initialized with the same randomly generated array whose elements scatter in the range of [-0.05, 0.05] according to prior research [21, 22];
b) Note that as discussed in [63], for LFA models depending on $l_2$ norm-based regularization, diversifying $\lambda_P$ and $\lambda_Q$ can yield slight accuracy gain. However, it calls for time-consuming search with $\lambda_P$ and $\lambda_Q$. Hence, in this study we adopt the same empirical value for regularization coefficients on $P$ and $Q$, i.e., setting $\lambda_P=\lambda_Q=0.03$ uniformly following [12].
c) In PSO and P²SO, we choose the same settings for the hyper parameters: $w$, $r_1$ and $r_2\in(0,1)$ and is generated by a

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: DOI 10.1109/TKDE.2020.3033324, IEEE Transactions on Knowledge and Data Engineering

uniform distribution, $S$=10, $c_1$=$c_2$=2 [32], $v$∈[-1,1], and $\forall j \in \{1,\ldots,S\}$: $\eta_j \in [2^{-12}, 2^{-8}]$.

d) Each set of experiment is repeated for 20 times independently for eliminating the biased results causing by randomness in LF initialization, PSO and P²SO.

### 4.2 Effect of $\rho$

As discussed in Section 3, a P²SO tunes the static and dynamic adjusting velocities through the hyper parameter $\rho$. Hence, it is necessary to conduct parameter sensitivity tests with it for a PLFA model. Results of sensitivity tests are given in Figs. S2 and S3 in the Supplementary File. From them, we have the following findings:

a) **A PLFA model's prediction error decreases as $\rho$ increases.** For instance, as depicted in Fig. S2(a), its RMSE on D1 with $\rho$=0.0 and 1.0 is 0.7993 and 0.7858, respectively. The gap ratio between its lowest and highest RMSE is 1.69%. In terms of its MAE on D1, from Fig. S3(a) we see that its MAE is 0.6147 and 0.6040 when $\rho$=0 and 1.0, where the gap ratio is 1.74%. Similar situations are found on D2-D4. As discussed in Section 3, when $\rho$=0, a P²SO algorithm becomes a standard PSO algorithm which solely depends on 'static' adjusting velocities solely depending on the static positions connected with each particle. On the other hand, when $\rho$=1, its adjusting velocities are mostly 'dynamic' decided by the direction vectors linked with its locally optimal position, globally optimal position of the whole swarm, along with its latest evolution velocity. Hence, from this phenomenon, we see that a PLFA model makes a particle's evolution depend on such 'dynamic' adjusting velocities solely is helpful in improving its prediction accuracy for missing data.

b) **A PLFA model's converging iteration count increases as $\rho$ increases.** For instance, as shown in Fig. S2(b), the converging iteration count of a PLFA model in RMSE on D1 is 10 and 17 as $\rho$=0 and 1.0, respectively. In MAE, a PLFA model's converging iteration count is 10 and 17 respectively, as depicted in Fig. S3(b). Similar situations are also found on the other datasets, as shown in Figs. S2(b) and S3(b). As analyzed in Section 3.2, when $\rho$≠0, a P²SO algorithm can preventing the swarm from premature convergence. Hence, it is reasonable that a PLFA model consumes more iterations as it increases. Meanwhile, as the converging iteration count increases, a PLFA model's converging time cost also increases correspondingly, as depicted in Figs. S2(c) and S3(c).

c) **Based on the above results and analyses, we see that $\rho$ plays an important role in performance of a PLFA model.** When it is small, a PLFA model converges with fewer iterations and less time cost but suffering a loss of prediction accuracy. With it increases, the converging iteration count and time cost of a PLFA model increases, but it obtains an accuracy gain. However, to be shown next, owing to its self-adaptation of learning rate, a PLFA model is able to achieve very competitive prediction accuracy for missing data, as well as very high computational efficiency when compared with a SGD-based LFA model.

d) **As $\rho$ set appropriately, PLFA implements learning rate adaptation efficiently.** Fig. S4 in the Supplementary File depicts a PLFA model's learning rate adaptation and learning curves as $\rho$ varies on D1. Note that similar situations are also encountered on the other datasets. From Fig. S4(a)-(c), we clearly see that the rate adaptation trend of $\eta$ varies in $\rho$. According to Fig. S4(a), as $\rho$=0, $\eta$ maintained by particles 1-10 all are set to $\eta_{\min}$ initially, and then grows gradually with more iterations. Finally, $\eta$ maintained by particles 1-10 stabilize at $\eta_{\max}$, making the resultant model also converge. As shown in Fig. S4(b), as $\rho$=0.6, $\eta$ maintained by particles 1-10 fluctuate between $\eta_{\min}$ and $\eta_{\max}$ during the training process, and then converge at $\eta_{\max}$ along with the convergence of the resultant model. As depicted in Fig. S4(c), as $\rho$=1.0, $\eta$ maintained by particles 1-10 is adjusted in a similar way with the case of $\rho$=0.6. However, the evolution of $\eta$ by particles 1-10 is implemented more carefully (which is reflected by the perplexed evolution curves of $\eta$ by different particles in Fig. S4(c). Thus, as $\rho$=1, a PLFA model is able to better represent an HiDS matrix at the cost of more training iterations, as shown in Fig. S4(d). This phenomenon is consistent with our discoveries discussed in points a)-c) mentioned above.

### 4.3 Effects of Evolution Parameters

In Section 4.1, we have already provided the empirical scale of evolution parameters in an PLFA model. However, it is still necessary to test their effects in an PLFA model's performance. Hence, in this part of experiments we analyze the effects of swarm size $S$, cognitive and social coefficients $c_1$ and $c_2$, bound of velocity $v_{\min}$ and $v_{\max}$, $\eta$'s bound $\eta_{\min}$ and $\eta_{\max}$, in the performance of an PLFA model. Note that $w$, $r_1$ and $r_2$ are excluded since they all are random numbers generated from a uniform distribution in the range of (0, 1) like other PSO algorithms [32-37]. Meanwhile, only results on D1 are reported for a concise report since results encountered on the other datasets are highly similar.

#### *4.3.1 Swarm size: S*

$S$'s effects in a PLFA model's performance as RMSE and MAE as the evaluation metric are recorded in Tables S1 and S2 in the Supplementary File, respectively. From Tables S1 and S2, we clearly see that a PLFA model's performance is affected by the swarm size of P²SO. More specifically, we have the following findings:

a) **As $S$ increases, a PLFA model's time cost per iteration increases, while its converging iteration count decreases.** According to Algorithm PLFA, the swarm size $S$ actually affects the 'inner loop' of a PLFA model's each training iteration: it will traverse $\Lambda$ for $S$ times in a single iteration to update LFs with the learning rate maintained by the incorporated P²SO algorithm. Hence, it actually traverses $\Lambda$ for $S \times t$ times as $t$ denotes the converging iteration count, as recorded in the column of '$\Lambda$ Traversing Counts' in Tables S1 and S2. From it, we further see that as S increases, a PLFA model actually traverses $\Lambda$ for more times to converge, which indicates that the increasing swarm size increase the exploration ability of PLFA, preventing it from premature convergence. From this point of view, it is reasonable that a PLFA model's time

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: DOI 10.1109/TKDE.2020.3033324, IEEE Transactions on Knowledge and Data Engineering

cost per iteration and total time cost both increases as $S$ increases.

b) **As $S$ increases, a PLFA model's prediction accuracy for missing data of an HiDS matrix slightly increases.** From Tables S1 and S2, we see that an PLFA model's RMSE and MAE decreases as $S$ increases. For instance, on D1 its RMSE is 0.7885 when $S$=2, and 0.7858 when $S$=10. The accuracy gain by increasing $S$ from 2 to 10 is 0.34%. Note that this slight accuracy gain is achieved at the cost of 40% more time consumption. From this point of view, small $S$ is suitable for applications with real time needs, while large $S$ can be adopted in applications seeking for highly accurate representation of an HiDS matrix.

#### 4.3.2 Cognitive and social coefficients: $c_1$ and $c_2$

$c_1$ and $c_2$'s effects in a PLFA model's performance are depicted in Fig. S5 in the Supplementary File. From it, we see that a PLFA model's prediction error fluctuates drastically when $c_1$ and $c_2$ are less than 1.8. However, as $c_1$ and $c_2$ increases over 2.0, the prediction error of a PLFA model becomes stable, as shown in Figs. S5(a) and (b). Considering a PLFA model's convergence rate, from Figs. S5(c) and (d) we find its increasing trend as $c_1$ and $c_2$ increase. Thus, the time cost of a PLFA model also tends to increase as $c_1$ and $c_2$ increase, as depicted in Figs. S5(e) and (f). Based on these results, we conclude that the widely adopted empirical values of $c_1$=$c_2$=2.0 [32-37] can also be adopted in a PLFA model to ensure its steady performance.

#### 4.3.3 Bound of velocity: $v_{\min}$ and $v_{\max}$

$v_{\min}$ and $v_{\max}$'s effects in a PLFA model's performance are depicted in Fig. S6 in the Supplementary File. Note that we set $|v_{\min}|=|v_{\max}|$ in our experiments. As shown in Fig. S6, a PLFA model generally achieves good balance between prediction accuracy and convergence rate as $|v_{\min}|=|v_{\max}|=1$. Meanwhile, by comparing Figs. S6(a), (c) and (e), we see that as $v_{\min}$ and $v_{\max}$ changes, a PLFA model's accuracy is closely related to its convergence rate and time cost: it can converge with less iterations, while suffers from slight loss in prediction accuracy. When adopting MAE as the evaluation metric, the phenomenon is the same, as shown in Figs. S6(b), (d) and (f). Therefore, it is suggested to adopt the empirical values of $|v_{\min}|=|v_{\max}|=1$ and $v_{\min}=-v_{\max}$ as suggested in [31].

#### 4.3.4 Bound of $\eta$: $\eta_{\min}$ and $\eta_{\max}$

In this part of experiments, we keep the relative magnitude difference from $\eta_{\min}$ to $\eta_{\max}$, i.e., making $\eta_{\max}=24\times\eta_{\min}$, and then let $\eta_{\min}$ increase from $2^{-15}$ to $2^{-6}$ to validate how do $\eta_{\min}$ and $\eta_{\max}$ affect a PLFA model's performance. The results are depicted in Fig. S7 in the Supplementary File. From it, we see that as $\eta_{\min}$ and $\eta_{\max}$ set too small, e.g., $\eta_{\min}=2^{-15}$ and $\eta_{\max}=2^{-11}$ in our experiments, a PLFA model's converging iteration count increases, leading to much time cost as shown in Figs. S7(c)-(f). However, as $\eta_{\min}$ and $\eta_{\max}$ set too large, e.g., $\eta_{\min}=2^{-6}$ and $\eta_{\max}=2^{-2}$, a PLFA model suffers from significant accuracy loss, as shown in Figs. S7(a)-(b). Furthermore, from Figs S7(g)-(h), a PLFA model is even unable converge when the learning rate is beyond a certain bound. Fortunately, from Fig. S7 we see that a PLFA model's performance is not very sensitive to $\eta_{\min}$ to $\eta_{\max}$. As $\eta_{\min}\in[2^{-12}, 2^{-8}]$ and $\eta_{\min}\in[2^{-8}, 2^{-4}]$, a PLFA model can well balance its time cost and prediction accuracy for missing data.

#### 4.3.5 Summary of parameter sensitivity tests

Based on the results of this section, we conclude that the evolution parameter setting in a PLFA model can follow the empirical guidance achieved in previous research regarding PSO algorithms [31-37], which is given in the model setting descriptions of Section 4.1.

### 4.4 Comparison Results

In this part, our goal is to compare the performance of a PLFA model with that of an SGD-based LFA model and an Adam-based LFA model. The following models are included in this set of experiments:

**M1**: An SGD-based LFA model. Note that we choose it as the baseline because a PLFA model's each particle takes it as the base model, as shown in Section 3.3. Hence, it is reasonable to compare M1 with a PLFA model to see the effect brought by a P²SO algorithm into it. Naturally, P2SO is compatible with other LFA models relying on a learning rate-dependent training process. Further investigations into this issue are highly interesting and we plan to make them in the future;

**M2**: A PLFA model with $\rho$=0. As discussed in Section 3.2, it is equivalent with an LFA model relying on a standard PSO algorithm; and

**M3**: An Adam-based LFA model. Note that an Adam algorithm adjusts the learning rate during an SGD process based on both the exponentially decaying square average and exponentially decaying average of past stochastic gradients [49]. It is a state-of-the-art adaptive SGD algorithm frequently adopted to train deep neural networks. Hence, it is highly representative to compare an Adam-based LFA model with the proposed PLFA model to validate its efficiency.

**M4**: A PLFA model with $\rho$=1. As shown in the previous section, M4 is able to achieve highly accurate predictions for missing data from an HiDS matrix. We hope to see its performance gain when compared with M1-3.

Note that the learning rate of M1 is not self-adaptive. Hence, we have tuned its learning rate on each dataset to make it achieve the highest prediction accuracy on one fold of each dataset, and adopt the same settings on the other folds. Its learning rate settings on each dataset are recorded in Table S3 in the Supplementary File.

Tables S4 and S6 in the Supplementary File summarize their lowest RMSE, MAE and converging iteration counts; Tables S5 and S7 in the Supplementary File summarize the average time cost per iteration and total time cost. From them, we have the following findings.

a) **Owing to its incorporation of a P²SO algorithm, a PLFA model implement efficient self-adaptation of learning rate, and most importantly, without loss of prediction accuracy for missing data of an HiDS matrices**. For instance, as described in Table S4, on D1, M4 has the lowest RMSE at 0.7858, which is 0.18% lower than the RMSE at 0.7872 by M1. Similar situations are also encountered on the other datasets. The same results are

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: DOI 10.1109/TKDE.2020.3033324, IEEE Transactions on Knowledge and Data Engineering

obtained in MAE, as recorded in Table S6. This phenomenon is very impressive since M1's learning rate is carefully tuned to enable it to achieve the highest prediction accuracy for missing data on each testing case. However, M4, with its P²SO-based self-adaptation of learning rate, can even achieve higher prediction accuracy than M1 does (although the accuracy gain is slight). This is indeed a significant progress. In comparison, as shown in tables S4 and S5, e.g., M2's prediction error is always higher than M1, which is considered as its compromise between the adaptive learning rate and prediction accuracy for missing data. Nonetheless, such compromise is never seen in M4. It turns out that with a P²SO algorithm, it is possible to implement the self-adaptation of learning rate in an LFA model without loss of prediction accuracy. Moreover, M3 is also a learning rate-adaptive algorithm, but it still suffers prediction accuracy loss except on D2.

b) **A PLFA model's convergence rate is very fast.** For instance, as summarized in Table S4, M4 takes 17 iterations to converge in RMSE on D1, while M1 takes 1000. M4's converging iteration count is only 1.7% that of M1's. Similar situations are also encountered on other testing cases, as shown in Tables S4 and S6. However, M4's converging iteration count is higher than that of M2. The reason for this phenomenon is analyzed in Section 4.2. In addition, M2-4 all outperform M1 in terms of convergence rate owing to their self-adaptation of learning rate.

c) **A PLFA model's computational efficiency is high**. First of all, it should be pointed out that the time cost per iteration by M2 and M4 is higher than that of M1. This is because in M2 and M4, there are $S$ particles which make each evolutionary iteration consist of $S$ sub-iterations, thereby making M2 and M4's time cost per iteration about $S$ times that of M1's. E.g., in our experiments we set $S$=10 for M2 and M4. Hence, their time cost per iteration is about ten times that of M1's.

d) **Considering their total time cost, however, the situation is very different**. Since M4 converges much faster than M1, its total time cost is much less than that of M1. For instance, as recorded in Table S5, on D1 M4 consumes 153 seconds to achieve the lowest RMSE, only 17.51% of 874 seconds by M1 does. The same situations are also found on the other datasets. As shown in Tables S5 and S7, M2 consumes less total time than M4 does, but it suffers significant accuracy loss which is discussed above. On the other hand, M3, i.e., the widely-adopted Adam algorithm, makes compromise of time cost for its self-adaptive learning rate. Its total time cost is even significantly higher than M1.

### 4.5 Summary

Based on the above results and analyses, we summarize that by incorporating a P²SO algorithm into its learning scheme, a PLFA model achieves efficient learning rate adaptation, high prediction accuracy for missing data of an HiDS matrix and low computation burden:

a) Owing to the highly dynamic evolution scheme of a P²SO algorithm, a PLFA model no longer suffers from premature convergence; and

b) With its efficient learning rate adaptation, its prediction accuracy is even higher than that of an LF model with finely-tuned learning rate.

Hence, a PLFA model shows great potential to solve real LFA problems raised in industrial applications.

## 5 Conclusions

How to appropriately select the learning rate is an essential issue for an SGD-based LFA model. Existing approaches to it mostly rely on a pre-tuning process on a probe set, whose defects are two-fold: a) it is extremely time-consuming to conducting linear or grid search for setting the learning rate properly, and b) due to the data-dependence of learning rate, the tuned results on a probe set may not enable the best performance of a resultant model on the target data. Existing approaches (like Adam) to this issue all bring extra computation burden to the model training process, which makes a resultant model cost even more time than an standard SGD-based model does.

However, with a PLFA model proposed in this study, it becomes possible to implement efficient learning rate adaptation in an SGD-based LFA model without any accuracy loss nor extra computation burden. A PLFA model's learning rate adaptation is implemented via a swarm consisting of different learning rates applied to the same group of LFs, which evolves via a newly-proposed P²SO algorithm without the risk of premature convergence.

In the future, we aim to address the following issues:

a) As unveiled by prior research, improved PSO algorithms, e.g., adaptive PSO [37], switching delayed PSO [44], switching local evolutionary PSO [45], swarm intelligence PSO [49], and quantum mechanism based PSO [50], have shown their impressive ability in addressing general optimization problems. Are they compatible with our problem, and if so, can we achieve further accuracy and efficiency gain by incorporating their principle into a P2SO algorithm for more efficient hypermeter-free LFA models? This question remains open.

b) On the other hand, do other intelligent optimization algorithms have the potential to build an LFA models with learning rate adaptation? For instance, a beetle antennae search algorithm [46] can solve an optimization problem in way of evolution conveniently, yet it commonly leads to accuracy loss. A brain storm optimization algorithm enables a learning model's high representation learning ability [52, 53], but it relies on an inner clustering process which can yield high computational complexity. It is highly interesting to implement hyper parameter-free LFA models based on their principles.

## References

[1] T. Nguyen and Y. Shin, "Matrix completion optimization for localization in wireless sensor networks for intelligent IoT," *Sensors*, vol. 16, no. 5, pp. 722-733, 2016.

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

[3] 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. on Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 587-600, 2012.

[4] X. Luo, Z. You, M. Zhou, S. Li, H. Leung, Y. Xia, and Q. Zhu, "A Highly Efficient Approach to Protein Interactome Mapping Based on Collaborative Filtering Framework," *Scientific Reports*, vol. 5, pp. 7702, 2015.

[5] 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.

[6] 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.

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

[8] X. Cao, X. Wang, D. Jin, Y. Cao, and D. He, "Identifying overlapping communities as well as hubs and outliers via nonnegative matrix factorization," *Scientific Reports*, vol. 3, pp. 2993, 2013

[9] 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

[10] Y. Li et al., "An efficient recommendation method for improving business process modeling," *IEEE Trans. on Industrial Informatics*, vol. 10, no. 1, pp. 502-513, Feb. 2014.

[11] 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, Jun. 2005.

[12] X. Luo, D.-X. Wang, M.-C. Zhou, H.-Q. Yuan, "Latent factor-based recommenders relying on extended stochastic gradient descent algorithms," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, DOI 10.1109/TSMC.2018.2884191.

[13] X. Luo, H. Wu, H.-Q Yuan, M.-C Zhou, "Temporal pattern-aware QoS prediction via biased non-negative latent factorization of tensors," *IEEE Trans. on cybernetics*, vol. 50, no. 5, pp. 1798-1809, May. 2020.

[14] 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, Nov. 2015.

[15] D. Szklarczyk, A. Franceschini, S. Wyder, K. Forslund, D. Heller, J. Huerta-Cepas, M. Simonovic, A. Roth, A. Santos, K. 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. 447-452, 2015.

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

[17] W. Chu, Z, Ghahramani, "Probabilistic models for incomplete multi-dimensional arrays," in *Proc. of the 12th Int. Conf. on Artificial Intelligence and Statistics*, Clearwater Beach, FL, Apr. 2009, pp. 89-96.

[18] Y. Koren and R. Bell, "Advances in collaborative-filtering," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. New York, NY, USA: Springer, 2011, pp. 145-186.

[19] S. Chatzis, "Nonparametric Bayesian multitask collaborative filtering," in *Proc. of the 22nd ACM Int. Conf. on Information and Knowledge Management*, San Francisco, CA, Oct. 2013, pp. 2149-2158.

[20] N. Srebro, and R. Salakhutdinov, "Collaborative filtering in a non-uniform world: learning with the weighted trace norm," *Advances in Neural Information Processing System*, pp. 2056-2064, 2010.

[21] X. Luo, M.-C Zhou, Z.-D Wang, Y.-N Xia, and Q.-S Zhu, "An effective QoS estimating scheme via alternating direction method-based matrix factorization," *IEEE Trans. on Services Computing*, vol. 12, no. 4, pp. 503-518, Nov. 2019.

[22] X. Luo, M.-C Zhou, S. Li, M.-S Shang, "An inherently non-negative latent factor model for high-dimensional and sparse matrices from industrial applications". *IEEE Trans. on Industrial Informatics*, vol. 14, no. 5, pp. 2011-2022, 2018.

[23] 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.

[24] J.-J. Pan, S.-J. Pan, Y. Jie, L.-M. Ni, and Q. Yang, "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, Mar. 2012.

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

[26] 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, 2009.

[27] 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, Jul. 2009, pp. 203-210.

[28] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. 7, pp. 2121-2159, 2011.

[29] M. D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method," *Computer Science*, 2012.

[30] D.-S. Li, C. Chen, Q. Lv, H.-S. Gu, T. Lu, L. Shang, N. Gu, and S.-M C, "AdaError: An Adaptive Learning Rate Method for Matrix Approximation-based Collaborative Filtering," in *Proc. of the 27th World Wide Web Conf.*, 2018, pp. 741-751.

[31] S. Hsieh, T. Sun, C. Lin, and C. Liu, "Effective Learning Rate Adjustment of Blind Source Separation Based on an Improved Particle Swarm Optimizer," *IEEE Trans. on Evolutionary Computation*, vol. 12, no. 2, pp. 242-251, 2008.

[32] J. Kennedy, and R.-C Eberhart, "Particle swarm optimization," in *Proc. of IEEE Int. Conf. Neural Networks*, Perth, Australia, Nov. 1995, vol. 4, pp. 1942–1948.

[33] Shi Y, and Eberhart RC, "Empirical study of particle swarm optimization," in *Proc. of the 1999 IEEE Congress on Evolutionary Computation*, Washington, DC, USA, USA, Jul. 1999, pp. 1945–1950.

[34] R. Cheng, and Y.-C. Jin, "A social learning particle swarm optimization algorithm for scalable optimization", *Information Sciences*, vol.291, pp. 43-60, 2015.

[35] T Yang, Z.-D Wang, J.-A Fang, "Parameters identification of unknown delayed genetic regulatory networks by a switching particle swarm optimization algorithm," *Expert Systems with Applications*, vol. 38, no. 3, pp. 2523–35, 2011.

[36] Valle Y, Venayagamoorthy G, Mohagheghi S, Hernandez J, Harley R, "Particle swarm optimization: basic concepts, variants and applications in power systems," *IEEE Trans. on Evolutionary Computation*, vol. 12, no. 2, pp. 171-195, 2008.

[37] Z.-H Zhan, J Zhang, Y Li, H Chung, "Adaptive particle swarm optimization," *IEEE Trans. on Systems Man and Cybernetics part B-Cybernetics*, vol. 39, no. 6, pp. 1362-1381, 2009.

[38] K. C. Kiwiel, "Convergence and efficiency of subgradient methods for quasiconvex minimization," *Mathematical Programming*, vol. 90, no. 1, pp. 1-25, 2001.

[39] C. K. Krzysztof, "Convergence of approximate and incremental subgradient methods for convex optimization," *SIAM Journal on Optimization*, vol. 14, no. 3, pp. 807-840, 2004.

[40] H. Robbins, and D. Siegmund, "A convergence theorem for non-negative almost super martingales and some applications," *Optimizing Methods in Statistics*, pp. 233-257, 1971.

[41] 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.

[42] P. Massa, and P. Avesani, "Trust-aware recommender systems," in *Proc. of the 1st ACM Conf. on Recommender Systems*, Minneapolis, MN, USA, Oct. 2007, pp. 17-24.

[43] J. Mohsen, and E. Martin, "A matrix factorization technique with trust propagation for recommendation in social networks," in *Proc. of the 4th ACM Conf. on Recommender Systems*, Barcelona, Spain, 2010, pp. 135-142.

[44] N.-Y. Zeng, Z.-D. Wang, H. Zhang, and F.-E. Alsaadi. "A Novel Switching Delayed PSO Algorithm for Estimating Unknown Parameters of Lateral Flow Immunoassay," *Cognitive Computation*. vol. 8, no. 2, pp. 143-152, 2016.

[45] N.-Y. Zeng, Y.-S. Hung, Y.-R. Li, and M. Du. "A novel switching local evolutionary PSO for quantitative analysis of lateral flow immunoassay," *Expert Systems with Applications*, vol. 41, no. 4, pp. 1708-1715, 2014.

[46] X.-Y Jiang, S. Li. "Beetle antennae search without parameter tuning (BAS-WPT) for multi-objective optimization," arXiv:1711.02395v1, 2017.

[47] X. Luo, M.-C. Zhou, S. Li, Z.-H. You, Y.N. Xia, and Q.-S. Zhu, "A non-negative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method". *IEEE Trans. on Neural Networks and Learning Systems*, vol. 27, no. 3, pp. 524-537, 2016.

[48] X. Luo, M.-C Zhou, Y.-N Xia, Q.-S Zhu, "Generating highly accurate predictions for missing QoS-data via aggregating non-negative latent factor models". *IEEE Trans. on Neural Networks and Learning Systems*, vol. 27, no. 3, pp. 579-592, 2016.

[49] S. García-Galán, R. P. Prado, and J. E. M. Expósito, "Swarm Fuzzy Systems: Knowledge Acquisition in Fuzzy Systems and Its Applications in Grid Computing," *IEEE Trans. on Knowledge and Data Engineering*, vol. 26, no. 7, pp. 1791-1804, 2014.

[50] H. Wang, and C. Qiao, "A Nodes' Evolution Diversity Inspired Method to Detect Anomalies in Dynamic Social Networks," *IEEE Trans. on Knowledge and Data Engineering*, DOI: 10.1109/TKDE.2019.2912574, 2019.

[51] D. P. Kingma, and J. Ba, "Adam: A Method for Stochastic Optimization," arXiv:1412.6980, 2014.

[52] Y.-H. Shi, "Brain Storm Optimization Algorithm," in *Proc. of the 2nd Int. Conf. on Advances in Swarm Intelligence*, Chongqing, China, 2011, pp. 303-309.

[53] Z.-H. Zhan, J. Zhang, Y.-H. Shi, and H. Lin, "A modified brain storm optimization," in *Proc. of the 2012 IEEE Congress on Evolutionary Computation*, Brisbane, Australia, 2012, pp. 1-8.

[54] A. Ratnaweera, S.-K. Halgamuge, and H.-C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Trans. on Evolutionary Computation*, vol. 8, no. 3, pp. 240-255, 2004.

[55] W. Dong and M.-C. Zhou, "A Supervised Learning and Control Method to Improve Particle Swarm Optimization Algorithms," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 7, pp. 1135-1148, 2017.

[56] J. Liang, A.-K. Qin, P.-N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. on Evolutionary Computation*, vol. 10, no. 3, pp. 281-295, 2006.

[57] Z.-H. Zhan, J. Zhang, Y. Li, and Y.-H. Shi, "Orthogonal Learning Particle Swarm Optimization," *IEEE Trans. on Evolutionary Computation*, vol. 15, no. 6, pp. 832-847, 2011.

[58] J. Behnamian and S.-M Ghomi, "Development of a PSO-SA hybrid metaheuristic for a new comprehensive regression model to time-series forecasting," *Expert Systems With Applications*, vol. 37, no. 2, pp. 974-984, 2010.

[59] Y.-J. Gong, J.-J. Li, Y.-C. Zhou, Y. Li, H.-S. Chung, Y.-H. Shi, and J. Zhang, "Genetic Learning Particle Swarm Optimization," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 46, no. 10, pp. 2277-2290, 2016.

[60] W.-N. Chen, J. Zhang, Y. Lin, N. Chen, Z.-H. Zhan, H.-S. Chung, and Y.-H. Shi, "Particle Swarm Optimization With an Aging Leader and Challengers." *IEEE Trans. on Evolutionary Computation*, vol. 17, no. 2, pp. 241-258, 2013.

[61] Y.-H. Li, Z.-H. Zhan, S.-J. Lin, J. Zhang, and X.-N. Luo, "Competitive and cooperative particle swarm optimization with information sharing mechanism for global optimization problems," *Information Sciences*, vol. 293, 370-382, 2015.

[62] X.-W. Xia, G. Ling, and Z.-H. Zhan, "A multi-swarm particle swarm optimization algorithm based on dynamical topology and purposeful detecting," *Applied Soft Computing*, vol. 67, no. 126-140, 2018.

[63] Y. Yuan, Q. He, X. Luo, and M. Shang, "A Multilayered and Randomized Latent Factor Model for High-Dimensional and Sparse Matrices", *IEEE Trans. on Big Data*, DOI: 10.1109/TBDATA.2020.2988778

[64] T.-T. He and K.-C. Chan, "Discovering Fuzzy Structural Patterns for Graph Analytics," *IEEE Trans. on Fuzzy Systems*, vol. 26, no. 5, pp. 2785-2796, 2018.

[65] T.-T. He, Y. Liu, T.-H. Ko, K.-C. Chan, and Y.-S. Ong, "Contextual Correlation Preserving Multi-view Featured Graph Clustering," *IEEE Trans. Cybernetics*, DOI 10.1109/TCYB.2019.2926431.

[66] C. Xu, D.-C. Tao, and C. Xu, "A Survey on Multi-view Learning," *arXiv: Learning*, 2013.

[67] Y. Nesterov, "Introductory Lectures on Convex Optimization: A Basic Course," *Kluwer Academic Publishers*, 2014.

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: DOI 10.1109/TKDE.2020.3033324, IEEE Transactions on Knowledge and Data Engineering

[68] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, "Robust Stochastic Approximation Approach to Stochastic Programming," *Siam Journal on Optimization*, vol. 19, no. 4, pp. 1574-1609, 2013。
[69] S. Boyd and L. Vandenberghe, *Convex Optimization*, *Cambridge: Cambridge University Press*, 2009.
[70] X. D. Zhang, "Matrix analysis and applications," *Cambridge University Press*, 2017.
[71] G. Strang, "Introduction to Linear Algebra," *Wellesley-Cambridge Press*, 1993.
[72] S. J. Leon, "Linear Algebra with Applications," *Prentice Hall*, 1980.
[73] D. C. Lay, "Linear Algebra and Its Applications," *Pearson*, 1994.
[74] E. Hazan, A. Agarwal, and S. Kale, "Logarithmic regret algorithms for online convex optimization," *Machine Learning*, vol. 69, no. 2, pp. 169–192, 2007.
[75] A. Rakhlin, O. Shamir, and K. Sridharan, "Making Gradient Descent Optimal for Strongly Convex Stochastic Optimization," in Proc. of Int. Conf. on Machine Learning, pp. 1571-1578, 2012.
[76] C. Jin, R. Ge, P. Netrapalli, S. M. Kakade, and M. I. Jordan, "How to escape saddle points efficiently," *In Proc. of the 34th Int. Conf. on Machine Learning*, vol. 70, pp. 1724-1732, 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. 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 50 IEEE TRANS. 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, and the Advanced Support of the Pioneer Hundred Talents Program of Chinese Academy of Sciences in 2018. He is currently serving as an Associate Editor for the IEEE/CAA JOURNAL OF AUTOMATICA SINICA, IEEE ACCESS, and *Neurocomputing*. He has received the Outstanding Associate Editor award of IEEE ACCESS in 2018.

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

**Sili Chen** received the B.S. degree in network engineering from China West Normal University, Nanchong, China, in 2017. She is currently pursuing her M.S. degree at China West Normal University and is a visiting student at Chongqing Institute of Green and Intelligent Technology. Her research interests include big data analysis and algorithm design for large scale data applications, especially for recommender systems.

**Nianyin Zeng** (M' 17) received the B.Eng. degree in electrical engineering and automation in 2008 and the Ph.D. degree in electrical engineering in 2013, both from Fuzhou University. From October 2012 to March 2013, he was a RA in the Department of Electrical and Electronic Engineering, the University of Hong Kong. From September 2017 to August 2018, he was an ISEF Fellow founded by the Korea Foundation for Advance Studies and also a Visiting Professor at the Korea Advanced Institute of Science and Technology. Currently, he is an Associate Professor with the Department of Instrumental & Electrical Engineering of Xiamen University. His current research interests include intelligent data analysis, computational intelligent, time-series modeling and applications. He is the author or coauthor of several technical papers including 7 ESI Highly Cited Papers according to the most recent Clarivate Analytics ESI report and also a very active reviewer for many international journals and conferences. Dr. Zeng is currently serving as an Associate Editor for Neurocomputing, Editorial Board members for Computers in Biology and Medicine, Biomedical Engineering Online, and also a Guest Editor for Frontiers in Neuroscience. He also serves as a technical program committee member for ICBEB 2014, an Invited Session Chair of ICCSE 2017.

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

He is currently a Professor of Dynamical Systems and Computing in the Department of Computer Science, Brunel University London, U.K. From 1990 to 2002, he held teaching and research appointments in universities in China, Germany and the UK. Prof. Wang's research interests include dynamical systems, signal processing, bioinformatics, control theory and applications. He has published more than 500 papers in refereed international journals. He is a holder of the Alexander von Humboldt Research Fellowship of Germany, the JSPS Research Fellowship of Japan, William Mong Visiting Research Fellowship of Hong Kong.

Prof. Wang serves (or has served) as the Editor-in-Chief for Neurocomputing, the Deputy Editor-in-Chief for International Journal of Systems Science, and an Associate Editor for 12 international journals, including IEEE Transactions on Automatic Control, IEEE Transactions on Control Systems Technology, IEEE Transactions on Neural Networks, IEEE Transactions on Signal Processing, and IEEE Transactions on Systems, Man, and Cybernetics-Part C. He is a Fellow of the IEEE, a Fellow of the Royal Statistical Society and a member of program committee for many international conferences.