A Multilayered-and-Randomized Latent Factor Model for High-Dimensional and Sparse Matrices

Ye Yuan, Qiang He, *Member*, *IEEE*, Xin Luo, *Senior Member*, *IEEE*, and Mingsheng Shang

Abstract—How to extract useful knowledge from a high-dimensional and sparse (HiDS) matrix efficiently is critical for many big data-related applications. A latent factor (LF) model has been widely adopted to address this problem. It commonly relies on an iterative learning algorithm like stochastic gradient descent. However, an algorithm of this kind commonly consumes many iterations to converge, resulting in considerable time cost on large-scale datasets. How to accelerate an LF model's training process without accuracy loss becomes a vital issue. To address it, this study innovatively proposes a multilayered-and-randomized latent factor (MLF) model. Its main idea is two-fold: a) adopting randomized-learning to train LFs for implementing a 'one-iteration' training process for saving time; and 2) adopting the principle of a generally multilayered structure as in a deep forest or multilayered extreme learning machine to structure its LFs, thereby enhancing its representative learning ability. Empirical studies on six HiDS matrices from real applications demonstrate that compared with state-of-the-art LF models, an MLF model achieves significantly higher computational efficiency with satisfactory prediction accuracy. It has the potential to handle LF analysis on a large scale HiDS matrix with real-time requirements.

Index Terms—Big Data, Latent Factor Analysis, Generally Multilayered Structure, Deep Forest, Multilayered Extreme Learning Machine, Randomized-learning, High-dimensional and Sparse Matrix, Stochastic Gradient Descent, Randomized Model

1 Introduction

THE RAPID expanding World-Wide-Web brings human society into an era of big data. In big data–related applications, interaction data among numerous entities commonly provide rich information reflecting their behaviors, e.g., user experience and service performance in quality-of-service data from service-oriented applications [1-3], node connections and characteristics in node interaction data from wireless sensor networks [4-6], and user preferences and item popularity in user-item ratings data from recommender systems [7-10]. Moreover, due to the exploding numbers of involved entities, it is impossible to observe the full relationship among them [1, 2, 5-7].

Under such circumstances, High-dimensional and sparse (HiDS) matrices [54-59] are commonly adopted to model highly-sparse relationships [7, 11]. For instance, the Douban matrix [33] collected by the Chinese largest online book, movie and music database includes 58,541 items and 129,490 users, while only contains 16,830,839 known ratings. Its data density is 0.22% only. Note that in an HiDS matrix, most entries are 'unknown' rather than 'zeroes' in conventional sparse matrices, e.g., an unobserved connection between a specified user-item tuple is missing rather than zero in recommender systems [7-10].

In spite of its sparsity, an HiDS matrix contains rich information regarding various desired patterns, e.g., user community tendency in a social network service system [61-64]. Therefore, how to extract these patterns from an HiDS matrix becomes a highly interesting issue. Various knowledge acquisition models are proposed to address it [7, 63, 64], where a latent factor (LF) analysis-based approach is considered highly efficient [11, 13, 14, 16, 29].

Given an HiDS matrix, an LF model maps its row and column entities into a unique LF space to build a learning objective. This objective is defined on its known data only with respect to desired LFs [11-14]. These LFs are subsequently trained to minimize the objective so that they can represent the known data of the HiDS matrix precisely. This optimization process is commonly non-convex [11, 15] and solved via an iterative learning algorithm like a stochastic gradient descent (SGD) algorithm. Although such an algorithm is widely adopted and achieves low time cost per iteration on an HiDS matrix, it commonly consumes many iterations to converge [11, 13, 14, 28, 29]. Hence, its time cost can be still considerable on large-scale datasets [11-14, 16].

To make an LF model achieve even higher computation efficiency on an HiDS matrix, it turns out to be vital to replace an iterative algorithm with a non-iterative one,

- *Y. Yuan, X. Luo, 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: {yuanye, luoxin21, msshang}@cigit.ac.cn).*
- *Q. He is with the School of Software and Electrical Engineering, Swinburne University of Technology, Melbourne, 3122, Australia. (email: qhe@swin.edu.au)*
- *Y. Yuan is also with the University of Chinese Academy of Sciences, Beijing 100049, China.*
- *X. Luo is also with the Hong Kong Polytechnic University, Kowloon, Hong Kong 999077, China, and with the Hengrui (Chongqing) Artificial Intelligence Research Center, Department of Big Data Analyses Techniques, Cloudwalk, Chongqing 401331, China.*
- *Y. Yuan, Q. He and X. Luo are co-first authors of this study.*
- *Corresponding Authors: X. Luo and M. Shang.*

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/TBDATA.2020.2988778, IEEE Transactions on Big Data

thereby reducing the iteration count. But its non-convex learning process is incompatible with analytical solvers [15]. From this point of view, a randomized learning algorithm becomes a potential solution to this problem [17-25].

According to prior research [17-25], a randomized learning algorithm has attracted much attention from both academic and industrial communities. It is extremely efficient in building a single-hidden layer feed-forward network (SLFN). For instance, an extreme learning machine-based approach has been widely adopted in various industrial applications [21-25]. When building an SLFN, a randomized learning algorithm randomly generates weights and biases for its hidden neurons to compute its output weights and biases according to the inputs analytically [18-25]. Therefore, the training process becomes non-iterative and thus highly efficient.

As discussed in [16], an LF model can be considered as an SLFN taking the target HiDS matrix as the inputs and outputs simultaneously. Since a randomized learning algorithm is highly efficient in building an SLFN, it can probably improve an LF model's computational efficiency greatly. Nonetheless, existing randomized algorithms suffer from the following defects when performing LF analysis on an HiDS matrix:

1) Existing randomized learning algorithms are mostly designed for complete inputs like images [21-25]. They are incompatible with an HiDS matrix; and
2) A randomized learning algorithm suffers from the dilemma between representative learning ability and complexity. Its representative learning ability significantly decreases with limited hidden neurons [21, 22], while its computational and storage burden grows drastically with numerous ones [16].

Aiming at addressing the above mentioned issues, this paper proposes a multilayered-and-randomized latent factor (MLF) model. Its main idea is to incorporate a generally multilayered structure into an LF model, where the input information of each layer is activated through randomly-generated weights and biases without backward propagations, thereby greatly improving the computational efficiency. On the other hand, its representative learning ability is guaranteed via its specifically-designed multilayered structure, where the input information at each layer keeps enhanced via analytical parameter solutions. Main contributions of this paper include:

1) A Multilayered-and-randomized Latent Factor (MLF) model with high computational efficiency and satisfactory prediction accuracy for missing data of an HiDS matrix; and
2) Algorithm design and analysis of an MLF model.

Extensive experiments are conducted on six HiDS matrices generated by real applications to evaluate the proposed MLF model's performance in comparison with five state-of-the-art models relying on randomized or iterative learning algorithms.

Note that an iterative learning algorithm like SGD is widely adopted to build deeply-structured models for LF analysis on an HiDS matrix in many learning model repositories like TensorFlow [67]. However, this study considers the same task via combining the principle of randomized learning and generally deep structure, thereby implementing non-iterative and highly efficient model training at the cost of slight loss in representative learning ability only. Thus, it actually provides researchers and engineers with a choice that stresses on model efficiency. It can be highly useful for real systems with the need of analyzing an HiDS matrix in real time. To the authors' best knowledge, such efforts have been never seen in any previous study.

The remainder of this paper is organized as: Section 2 gives preliminaries, Section 3 presents an MLF model, Section 4 provides empirical studies, and finally, Section 6 concludes this paper.

2 PRELIMINARIES

2.1 Symbol Appointment

Necessary symbols are summarized in Table I.

TABLE I. SYMBOL APPOINTMENT

Parameter	Description
U, I	Involved entity sets.
$U(i)$, $I(u)$	Subsets of U and I connected to entities $i\in I$ and $u\in U$.
f	LF space dimension/Node count in hidden layers.
R, $r_{u,:}$	A $\|U\|\times\|I\|$ HiDS matrix as input, and its uth row vector.
$\hat{R}$, $r_{u,i}$	R's rank-f approximation, and its single element.
R_K, R_U	Known and unknown entry sets in R and $\|R_K\|\ll\|R_U\|$.
$R_K(s)$	Subset of R_K related to entity s.
n	Number of layers.
A, $a_{k,:}$	A $f\times\|I\|$ hidden weight matrix for the first hidden layer hidden layer in an MLF model/the only hidden layer in an SLF model, and its kth row vector.
b, b_k	A length-f bias vector for the first hidden layer hidden layer in an MLF model/the only hidden layer in an SLF model, and its kth element.
W^n, $w^n_{k,:}$	A $f\times f$ hidden weight matrix for the nth ($n\geqslant 2$) hidden layer in an MLF model, and its kth row vector.
d^n, d^n_k	A length-f bias vector for the nth (n h 2) hidden layer in an MLF model, and its kth element.
P, $p_{u,:}$	A $\|U\|\times f$ weight matrix describing the input layer in an SLF model/LF matrix corresponding to entity set U, and its uth row vector.
P^n, $p^n_{u,:}$	A $\|U\|\times f$ output matrix of the nth input layer in an MLF model/ LF matrix corresponding to entity set U at the nth hidden layer, and its uth row vector.
Q, $q_{i,:}$	A $\|I\|\times f$ output weight matrix in an SLF model/LF matrix corresponding to entity set I, and its ith row vector.
Q^n, $q^n_{i,:}$	A $\|I\|\times f$ output weight matrix corresponding to P^n in an MLF model, and its ith row vector.
e	Euler's constant.
$g(x)$	Activation function.
z^T	Transpose of a matrix z.
C	Regularization coefficient.
In	An indicator matrix.
N	Maximum number of layers.
$\|\cdot\|_{abs}$	Absolute value of a given number.
$\|\|\cdot\|\|_F$	Frobenius norm of a given vector.

2.2 Randomized learning in Neural Networks

A randomized learning algorithm is initially designed to build an SLFN with high efficiency [17-25]. With it, an SLFN is trained within two steps only, 1) input mapping, where the input data is mapped into the hidden layer via a nonlinear activation function and randomly-generated weights and biases; and 2) parameter solving, where the output weights are analytically solved. Such an algorithm avoids solving the weights and biases via iterative backward propagations [15, 17-19], thereby achieving high computational efficiency. However, existing randomized learning algorithms like an extreme learning machine [2, 7, 11, 14, 29] is not compatible with an HiDS matrix since they are designed to take complete inputs like images.

3 An MLF Model

3.1 A Single-layered-and-randomized Case

Before building an MLF model, it is necessary to demonstrate its special case in a single-layered form, i.e., a Single-layered-and-randomized Latent Factor (SLF) model. Considering an SLF model, its structure is shown in Fig. 1. From it, we see that it works like an autoencoder, i.e., the inputs and outputs are both data from R. The hidden weight matrix P and output weight matrix Q form R's rank-f approximation based on R_K only, and the learning objective is to make P and Q well represent R_K. Next we discuss an SLF model in detail.

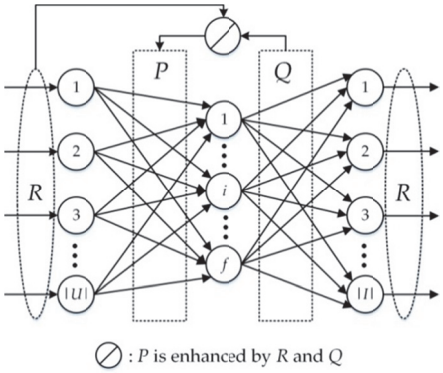


Fig. 1. Structure of an SLF model.

3.1.1 Sparse-data Oriented Learning Objective

To let P and Q precisely represent an HiDS matrix R, a learning objective should concentrate on R_K rather than R itself. To do so, we expand the Euclidean distance with the L_2-norm-based regularization $\forall r_{u,i}\in R_K$:

$$\varepsilon(P,Q)=\sum_{r_{u,i}\in R_K}\left(C\left(r_{u,i}-p_{u,\cdot}q_{i,\cdot}\right)^2+\left\|p_{u,\cdot}\right\|_F^2+\left\|q_{i,\cdot}\right\|_F^2\right). \quad (1)$$

Note that in (1), the regularization is applied to each single error $\forall r_{u,i}\in R_K$. With such a strategy, the regularization effects are connected with R's data density [13, 28].

3.1.2 *P*-Mapping Step

According to Fig. 1, when R is given, then P depends on the activation function with the hidden weight matrix A and the bias vector b as:

$$P=\begin{bmatrix}p_{1,\cdot}\\ \vdots\\ p_{|U|,\cdot}\end{bmatrix}=\begin{bmatrix}g\left(a_{1,\cdot}r_{1,\cdot}^{\mathrm{T}}+b_1\right) & \cdots & g\left(a_{f,\cdot}r_{1,\cdot}^{\mathrm{T}}+b_f\right)\\ \vdots & \ddots & \vdots\\ g\left(a_{1,\cdot}r_{|U|,\cdot}^{\mathrm{T}}+b_1\right) & \cdots & g\left(a_{f,\cdot}r_{|U|,\cdot}^{\mathrm{T}}+b_f\right)\end{bmatrix}. \quad (2)$$

Note that in a standard SLFN, A and b should be trained with a backward propagation algorithm [15]. However, following the principle of randomized learning, they are generated randomly for high efficiency. Thus, P is directly tractable as R_K, A and b are all given. Nonetheless, due to the sparsity of R, $r_{u,\cdot}$ is also incomplete $\forall u\in U$, making (2) intractable. To address this issue, we adopt the Procedure **COMPUTE_*P*** to compute $a_{k,\cdot}r_{u,\cdot}^{\mathrm{T}}$. Since R is an HiDS matrix, $\forall u\in U$ we have $|R_K(u)|\ll|I|$. With it, P can be generated on R_K only.

PROCEDURE **COMPUTE_*P***

Input: u, $a_{k,\cdot}$	
Operation	**Cost**
initialize RST = 0	$\Theta(1)$
for $r_{u,j}\in R_K(u)$	$\times\|R_K(u)\|$
RST = RST+$r_{u,j}\times a_{k,j}$	$\Theta(1)$
end for	--
Output: RST	$\Theta(1)$

Considering g(x), we set it as the hyperbolic tangent function [25], i.e., $g(x)=(1-e^{-x})/(1+e^{-x})$. However, our methodology works compatibly with other activation functions.

3.1.3 *Q*-Solving Step

After the P-Mapping step, Q can be solved analytically from (1). However, due to the sparsity of R, objective function (1) cannot be solved in a matrix-dependent way. Instead, Q is solved from (1) with respect to its each row vector, i.e., $q_{i,\cdot}$ ($\forall i\in I$). Note that the derivative of (1) with respect to $q_{i,\cdot}$ is given as:

$$\frac{\partial\varepsilon}{\partial q_{i,\cdot}}=2q_{i,\cdot}\left|R_K(i)\right|-2C\sum_{u\in U(i)}p_{u,\cdot}^{\mathrm{T}}r_{u,i}+2C\sum_{u\in U(i)}p_{u,\cdot}^{\mathrm{T}}p_{u,\cdot}q_{i,\cdot}. \quad (3)$$

By setting (3) at zero, we have the following inference:

$$\left(\frac{\left|R_K(i)\right|}{C}In+\sum_{u\in U(i)}p_{u,\cdot}^{\mathrm{T}}p_{u,\cdot}\right)q_{i,\cdot}=\sum_{u\in U(i)}p_{u,\cdot}^{\mathrm{T}}r_{u,i}, \quad (4)$$

where the indicator matrix In is $f\times f$, and $\sum_{u\in U(i)}p_{u,\cdot}^{\mathrm{T}}p_{u,\cdot}$ is a full $f\times f$ matrix. Thus, the solution to (4) is formulated by:

$$q_{i,\cdot}=\left(\frac{\left|R_K(i)\right|}{C}In+\sum_{u\in U(i)}p_{u,\cdot}^{\mathrm{T}}p_{u,\cdot}\right)^{-1}\sum_{r_{u,i}\in R_K(i)}p_{u,\cdot}^{\mathrm{T}}r_{u,i}. \quad (5)$$

By traversing $\forall i\in I$ to execute (5), Q can be solved then.

3.1.4 *P*-Enhancing Step

After the Q-solving step, we enhance P as depicted in Fig. 1. This is achieved by solving (1) with P with R_K and solved Q. Similar to the Q-solving step, the derivative of (1) with respect to $p_{u,\cdot}$ as $\forall u\in U$ is formulated by:

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/TBDATA.2020.2988778, IEEE Transactions on Big Data

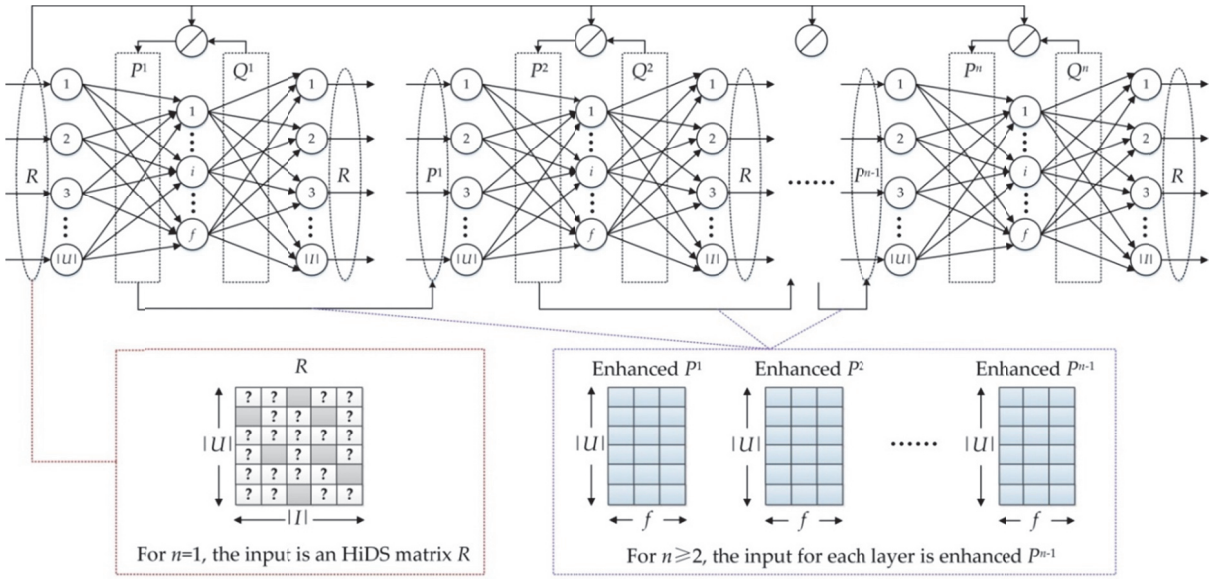


Fig. 2. Structure of an MLF model.

$$\frac{\partial \varepsilon}{\partial p_{u,\cdot}} = 2p_{u,\cdot}\left|R_K(u)\right| - 2C\sum_{i\in I(u)} r_{u,i}q_{i,\cdot}^{\mathrm{T}} + 2C\sum_{i\in I(u)} p_{u,\cdot}q_{i,\cdot}q_{i,\cdot}^{\mathrm{T}}. \quad (6)$$

Note that (6) can be analytically solved as follows:

$$p_{u,\cdot} = \left(\sum_{r_{u,i}\in R_K(u)} r_{u,i}q_{i,\cdot}^{\mathrm{T}}\right)\left(\frac{\left|R_K(u)\right|}{C}In + \sum_{i\in I(u)} q_{i,\cdot}q_{i,\cdot}^{\mathrm{T}}\right)^{-1}. \quad (7)$$

It should be pointed out that (5) and (7) inverses $\left(\frac{\left|R_K(i)\right|}{C}In + \sum_{u\in U(i)} p_{u,\cdot}^{\mathrm{T}}p_{u,\cdot}\right)$ and $\left(\frac{\left|R_K(u)\right|}{C}In + \sum_{i\in I(u)} q_{i,\cdot}q_{i,\cdot}^{\mathrm{T}}\right)$ to cost $\Theta(f^3)$. As small f (in the scale of [10, 500]) is commonly adopted by an LF model for well balancing the computational cost and representative learning ability [11, 16, 29], such a cost is easy to resolve in practice.

3.2 A Multilayered-and-randomized Case

Based on an SLF model, we generalize it to the multilayered-and-randomized case, i.e., an MLF model. Although a standard multilayered architecture can be adopted for such a purpose [37, 41, 48, 53], they can yield huge time cost. To address this issue, we adopt the principle of input connection skipping as in a deep forest [26] and a multilayered extreme learning machine [22] to implement a generalized-multilayer structure for an MLF model. The structure of an MLF model is illustrated in Fig. 2. From Fig. 2, we see that its learning process is executed as:

1) As shown in Fig. 2, the first layer of an MLF model has exactly the same structure as that of an SLF model. Hence, its parameters in the first layer is learnt in a similar way as in an SLF model, i.e., a) its input is R_K for the 1st layer, b) P^1 is achieved via P-mapping with R_K, c) Q^1 is achieved via Q-solving with R_K and P^1, and d) P^1 is enhanced via P-enhancing with R_K and Q^1.
2) Different from the first layer, at the nth layer as $n\geqslant 2$, the enhanced P^{n-1} is adopted as its input, as illustrated in Fig. 2 where the layer count $n\geqslant 2$. Thus, involved parameters are learnt as follows, a) P^n is achieved via P-Mapping with P^{n-1} , b) Q^n is solved via Q-solving with R_K and P^n, and c) P^n is enhanced via P-enhancing with R_K and Q^n.

Note that the generally multilayered structure of MLF differs from a standard multilayer architecture in neural networks mainly in two perspectives:

1) Its hidden weights and biases at each layer are random, which greatly reduces the computational cost.
2) It does not adopt backward propagation to connect its layers. Instead, it adopts the LF matrix sequence $\{P^1, P^2, \ldots, P^{n-1}\}$ as the input of layers 2-n to connect its layers. Note that at layer n, information of P^n is further enhanced via a P-enhancing step to well represent R_K. Moreover, such information gain is passed to the next layer via a P-mapping step to enhance the representative learning ability of the whole model.

Based on the above inferences, we summarize the expressions of P-mapping, Q-solving and P-enhancing steps in an MLF model as follows:

1) *P*-Mapping of MLF. For layers 1-n, it is formulated by:

$$P^n = \begin{cases} n=1, \\ \begin{bmatrix} p_{1,\cdot} \\ \vdots \\ p_{|U|,\cdot} \end{bmatrix} = \begin{bmatrix} g\left(a_{1,\cdot}r_{1,\cdot}^{\mathrm{T}} + b_1\right) & \cdots & g\left(a_{f,\cdot}r_{1,\cdot}^{\mathrm{T}} + b_f\right) \\ \vdots & \ddots & \vdots \\ g\left(a_{1,\cdot}r_{|U|,\cdot}^{\mathrm{T}} + b_1\right) & \cdots & g\left(a_{f,\cdot}r_{|U|,\cdot}^{\mathrm{T}} + b_f\right) \end{bmatrix}, \\ n\geq 2, \\ \begin{bmatrix} p_{1,\cdot}^n \\ \vdots \\ p_{|U|,\cdot}^n \end{bmatrix} = \begin{bmatrix} g\left(w_{1,\cdot}^n\left(p_{1,\cdot}^{n-1}\right)^{\mathrm{T}} + d_1^n\right) & \cdots & g\left(w_{f,\cdot}^n\left(p_{1,\cdot}^{n-1}\right)^{\mathrm{T}} + d_f^n\right) \\ \vdots & \ddots & \vdots \\ g\left(w_{1,\cdot}^n\left(p_{|U|,\cdot}^{n-1}\right)^{\mathrm{T}} + d_1^n\right) & \cdots & g\left(w_{f,\cdot}^n\left(p_{|U|,\cdot}^{n-1}\right)^{\mathrm{T}} + d_f^n\right) \end{bmatrix}; \end{cases} \quad (8)$$

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/TBDATA.2020.2988778, IEEE Transactions on Big Data

PROCEDURE **COMPUTE_P^N**

Input: m, $w_{k,\cdot}^{n-1}$	
Operation	**Cost**
initialize RST = 0	$\Theta(1)$
for m = 1 *to* f	$\times f$
RST = RST+ $p_{u,m}^{n-1} \times w_{k,m}^{n-1}$	$\Theta(1)$
end for	--
Output: RST	$\Theta(1)$

ALGORITHM **MLF**

Input: R_K, I, U, $g(x)$, f, N	
Operation	**Cost**
Initialize P^n	$\Theta(\|U\|\times f)$
Initialize Q^n	$\Theta(\|I\|\times f)$
Initialize A randomly	$\Theta(\|I\|\times f)$
Initialize b randomly	$\Theta(f)$
Initialize E	$\Theta(f\times f)$
Initialize h	$\Theta(f)$
Initialize s	$\Theta(f)$
for n = 1 **to** N	$\times N$
If n = 1	$\times 1$
for k = 1 **to** f	$\times f$
for $u \in U$	$\times \|U\|$
$p_{u,k}^n = g$(**COMPUTE_P^N**$(u,\ a_{k,\cdot})+b_k$)	$\Theta(\|R_K(u)\|)$
end for	--
end for	--
else	--
Initialize W^{n-1} randomly	$\Theta(f\times f)$
Initialize d^{n-1} randomly	$\Theta(f)$
for k = 1 **to** f	$\times f$
for m = 1 **to** f	$\times f$
$p_{u,k}^n = g$(**COMPUTE_P^N**$(m, w_{k,\cdot}^{n-1})+\ d_k^{n-1}$)	$\Theta(f)$
end for	--
end for	--
end if	--
for $i \in I$	$\times \|I\|$
reset E = *zero*	$\Theta(f\times f)$
reset h = *zero*	$\Theta(f)$
for $u \in U(i)$*	$\times \|R_K(i)\|$
$E = E + \left(p_{u,\cdot}^n\right)^{\mathrm{T}} p_{u,\cdot}^n$	$\Theta(f\times f)$
$h = h + \left(p_{u,\cdot}^n\right)^{\mathrm{T}} r_{u,i}$	$\Theta(f)$
end for	--
$q_{i,\cdot}^n = \left(\frac{\left\|R_K(i)\right\|}{C} In + E\right)^{-1} h$	$\Theta(f^3+f^2)$
end for	--
for $u \in U$	$\times \|U\|$
reset E = *zero*	$\Theta(f\times f)$
reset s = *zero*	$\Theta(f)$
for $i \in I(u)$**	$\times \|R_K(u)\|$
$E = E + q_{i,\cdot}^n \left(q_{i,\cdot}^n\right)^{\mathrm{T}}$	$\Theta(f\times f)$
$s = s + r_{u,i}\left(q_{i,\cdot}^n\right)^{\mathrm{T}}$	$\Theta(f)$
end for	--
$p_{u,\cdot}^n = s\left(\frac{\left\|R_K(u)\right\|}{C} In + E\right)^{-1}$	$\Theta(f^3+f^2)$
end for	--
end for	--
Output: P^n and Q^n	

*As given in Section 3.1, we have $|R_K(i)|=|U(i)|$.

**As given in Section 3.1, we have $|R_K(u)|=|I(u)|$.

where the weight matrices A and W^n, and bias vectors b and d^n are all randomly generated. Note that Procedure **COMPUTE_P^N** is proposed for executing the P-Mapping step when $n \geqslant 2$.

2) Q-Solving of MLF. Firstly, note that the learning objective at layer n is formulated by:

$$\varepsilon_{MLF}\left(P^n, Q^n\right) = \sum_{r_{u,i} \in R_K} \left(C\left(r_{u,i} - p_{u,\cdot}^n q_{i,\cdot}^n\right)^2 + \left\|p_{u,\cdot}^n\right\|_F^2 + \left\|q_{i,\cdot}^n\right\|_F^2 \right). \quad (9)$$

Thus, based on (3) and (8), the derivative of (9) with $q_{i,\cdot}^n$ is:

$$\frac{\partial \varepsilon_{MLF}}{\partial q_{i,\cdot}^n} = 2q_{i,\cdot}^n \left|R_K(i)\right| - 2C \sum_{r_{u,i} \in R_K(i)} \left(p_{u,\cdot}^n\right)^{\mathrm{T}} r_{u,i} + 2C \sum_{u \in U(i)} \left(p_{u,\cdot}^n\right)^{\mathrm{T}} p_{u,\cdot}^n q_{i,\cdot}^n. \quad (10)$$

According to (10), Q^n is solved as follows:

$$q_{i,\cdot}^n = \left(\frac{\left|R_K(i)\right|}{C} In + \sum_{u \in U(i)} \left(p_{u,\cdot}^n\right)^{\mathrm{T}} p_{u,\cdot}^n \right)^{-1} \sum_{r_{u,i} \in R_K(i)} \left(p_{u,\cdot}^n\right)^{\mathrm{T}} r_{u,i}. \quad (11)$$

3) P-Enhancing of MLF. Similar to (7), at the nth layer we enhance P^n with Q^n and R_K as follows:

$$p_{u,\cdot}^n = \left(\sum_{r_{u,i} \in R_K(u)} r_{u,i} \left(q_{i,\cdot}^n\right)^{\mathrm{T}} \right) \left(\frac{\left|R_K(u)\right|}{C} In + \sum_{i \in I(u)} q_{i,\cdot}^n \left(q_{i,\cdot}^n\right)^{\mathrm{T}} \right)^{-1}. \quad (12)$$

Then $\hat{R}_n$ at the nth layer of an MLF model is given as:

$$\hat{R}_n = P^n Q^n. \quad (13)$$

3.3 Algorithm Design and Analysis

Based on the above inferences, we develop Algorithm MLF. Its computational cost is given as follows:

$$\begin{aligned} T_{MLF} &= \Theta\left(|U|\times f\right) + 2\Theta\left(|I|\times f\right) + \Theta\left(f^2\right) + 3\Theta\left(f\right) \\ &\quad + \Theta\left(|R_k|\times f\right) + (N-1)\left[\Theta\left(f^2\right) + \Theta\left(f\right) + \Theta\left(f^3\right)\right] \\ &\quad + N\times\left[|I|\times\left(\Theta\left(f^2\right) + \Theta\left(f\right) + \Theta\left(f^3+f^2\right)\right)\right] \\ &\quad + N\times\left[|U|\times\left(\Theta\left(f^2\right) + \Theta\left(f\right) + \Theta\left(f^3+f^2\right)\right)\right] \\ &\quad + 2N\times\left[|R_k|\times\left(\Theta\left(f^2\right) + \Theta\left(f\right)\right)\right] \\ &\approx \Theta\left(N\times\left(\left(|U|+|I|\right)\times f + |R_k|\right)\times f^2\right) \end{aligned} \quad (14)$$

Note (14) reasonably omits the lower-order-terms to achieve its final result. Since in practice we commonly have $(|U|+|I|) \ll |R_K|$ and f being a small constant, MLF's computational cost is actually linear with $|R_K|$.

4 EXPERIMENTAL RESULTS

4.1 General Settings

Evaluation Protocol. For real applications [9-11, 13, 14, 16], one major motivation to analyze an HiDS matrix is to estimate its missing data to recover full relationship among involved entities. Hence, this study sets missing data estimation as the evaluation protocol. According to prior research [11, 13, 14, 39], an LF model's accuracy for missing data estimation can be measured by root mean squared error (RMSE) and mean absolute error (MAE):

$$\begin{cases} RMSE = \sqrt{\left(\sum_{r_{u,i} \in T} \left(r_{u,i} - \hat{r}_{u,i}\right)^2\right) \Big/ |T|}, \\ MAE = \left(\sum_{r_{u,i} \in T} \left|r_{u,i} - \hat{r}_{u,i}\right|_{abs}\right) \Big/ |T|; \end{cases}$$

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/TBDATA.2020.2988778, IEEE Transactions on Big Data

where T denotes the validation set and is disjoint with R_K, $\hat{r}_{u,i}$ denotes the prediction for testing instance $r_{u,i} \in T$, respectively.

Datasets. The experiments are conducted on a series of real-world HiDS matrices which are high-dimensional, extremely sparse and collected by real applications currently in use. Their details are given below.

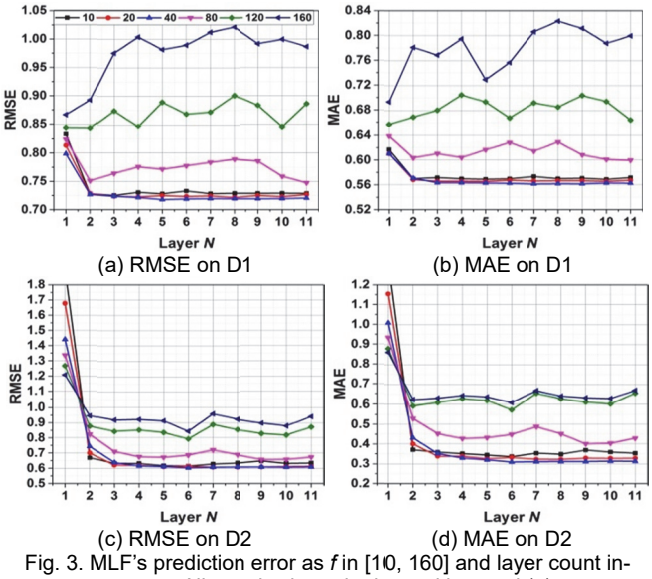
1) D1: Douban dataset [33]. It is collected from the Chinese largest online book, movie and music vendor Douban. It contains 16,830,839 ratings within the range of [1, 5] from 129,490 users on 58,541 items. Its density is 0.22% only.
2) D2: Extended Epinion dataset [34]. It is collected by Trustlet, with 13,668,320 known entries in the range of [1, 5], provided by 120,492 users on 755,760 articles. Its density is 0.015% only.
3) D3: Dating Agency dataset [35]. It is collected by an online dating website LibimSeTi, with 17,359,346 known entries in the range of [1, 10], from 135,359 users on 168,791 profiles. Its density is 0.076% only.
4) D4: Viscoplastic dataset [40]. It is provided by the University of Florida and contains 381,326 known entries denoting the vibration stiffness data of a piece of specific kind of material among its 32,769 nodes. Its density is 0.035% only.
5) D5: Maragal dataset [40]. It is collected by the University of Florida and contains 1,308,415 known entries with 33,212 rows and 75,077 columns. The dataset is used to in solving rank-deficient problems [49] and its density is 0.052% only.
6) D6: Delor dataset [40]. It is also collected by the University of Florida and contains 4,211,599 known entries with 343,236 rows and 887,058 columns, which denotes the mixed perturbation in underdetermined systems [51]. The data density is 0.00138% only.

To facilitate fair and square evaluation, we adopt the 80%-20% train-test setting and five-fold cross-validation in the experiments on all six datasets. More specifically, we randomly split the set of known entries of each HiDS matrix into five equally-sized, disjoint subsets. Each time we select four subsets as the training set R_K to train a model and the remaining subset as the testing set T. This process is repeated for five times and the results are averaged to obtain the final results.

4.2 Parameter Sensitivity

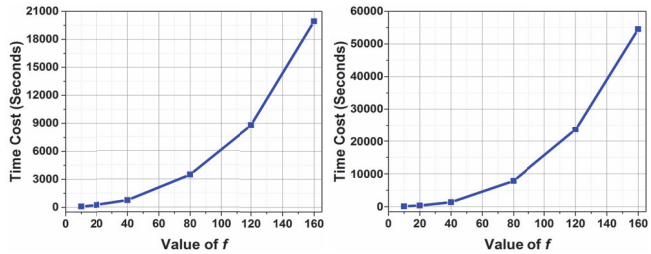
Naturally, an MLF model's performance is affected by its regularization coefficient C and LF dimension f. In this part of experiments, we analyze the performance of an MLF model with respect to them.

1) **An MLF model's prediction accuracy for missing data of an HiDS matrix is not proportional to f.** Fig. 3 depicts an MLF model's prediction error as f increases from 10 to 160 and layer count N increases from 1 to 11.



(a) RMSE on D1 (b) MAE on D1
(c) RMSE on D2 (d) MAE on D2

Fig. 3. MLF's prediction error as f in [10, 160] and layer count increases. All panels share the legend in panel (a).



(a) Time cost on D1 (b) Time cost on D2

Fig. 4. MLF's time cost as f in [10, 160].

Note that C is fixed at 2^4 in Fig. 3. From it, we see that MLF's prediction accuracy is not proportional to f. For instance, as shown in Fig. 3(a), on D1 with C=2^4 its RMSE is 0.7278 with f=10, 0.7224 with f=20, 0.7176 with f=40, 0.7714 with f=80, 0.8452 with f=120, and 0.9816 with f=160, respectively. Its lowest RMSE 0.7176 occurs when f=40, which is about 26.89% lower than 0.9816 with f=160. Similar situations are also encountered on the other testing cases, as in Fig. 3.

This property of an MLF model may rely on its generally multilayered structure, which has significantly positive effects in its prediction accuracy. For instance, as shown in Figs. 3(a)-(b), with f=40 on D1, its RMSE and MAE are 0.7987 and 0.6103 on the 1st layer 1, while its lowest RMSE at 0.7176 and MAE at 0.5632 occur on its 5th layer. The accuracy gain brought by its generalized-multilayered structure is 10.15% in RMSE and 7.71% in MAE, respectively

2) **An MLF model's computational cost grows non-linearly as f increases.** As analyzed in previous sections, an MLF model's computational complexity is proportional to f^3. Hence, it is necessary to validate its scalability as f increase. Fig. 4 depicts its time cost as f increases from 10 to 160 on D1-2. Note that similar situations are found on other datasets. From it, we clearly see that an MLF model's time cost is non-linearly related with its LF dimension f. For instance, as depicted in

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/TBDATA.2020.2988778, IEEE Transactions on Big Data

Fig. 4(a), on D1, it takes 97.9, 234.3, 727.1, 3448.5, 8801.1, and 19936.4 seconds as f=10, 20, 40, 80, 120, and 160, respectively. As f grows bigger, its time cost increases drastically. It appears necessary to adopt parallelization or distribution [55-59] for further reducing an MLF model's time cost. We plan to address this issue in the future.

3) **Regularization effects are vital for an MLF model to improve it generality, but should be controlled with care.** Fig. 5 depicts an MLF model's prediction error as C in $\{2^0, 2^2, 2^4, 2^6, 2^8, +\infty\}$ and layer count increases. Note that in Fig. 5 f is fixed at 40. Fig. 6 depicts an MLF model's prediction error as C in $\{2^0, 2^2, 2^4, 2^6, 2^8, +\infty\}$ and f increases from 10 to 160.

It should be pointed out that as C=+∞, an MLF model's learning objective (9) actually turns into the following non-regularized form:

$$\varepsilon_{MLF}\left(P^n, Q^n\right) = \sum_{r_{u,i} \in R_K} \left(r_{u,i} - p^n_{u,\cdot} q^n_{i,\cdot}\right)^2. \tag{16}$$

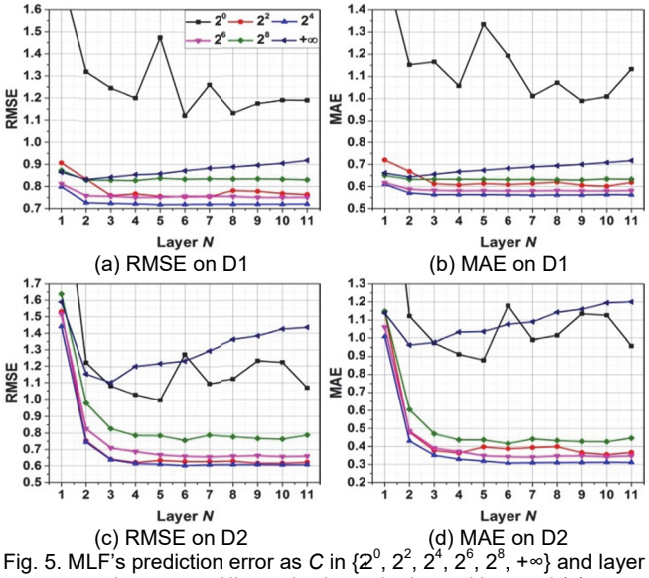
With (16), we will obtain a non-regularized MLF model which might suffer from overfitting. On the other hand, as C decreases, from (9) we see that the regularization effects are amplified. Thus, with inappropriately small C, a resultant MLF model will swing to another extreme to suffer from underfitting. Both overfitting and underfitting will result in accuracy loss, which we are expecting to observe from Figs. 4 and 5.

From Fig. 5, we clearly see that regularization effects should be controlled with care for an MLF model. With inappropriately chosen C, it will inevitably suffer from accuracy loss caused by underfitting or overfitting. For instance, as depicted in Fig. 5(c), on D2, its lowest RMSE is 0.6023 with C=2^4. In contrast, as we expected, it suffers from underfitting with C=2^0 to achieve the RMSE at 0.9989, which is about 39.70% higher than its lowest RMSE.

On the other hand, it suffers overfitting as C grows inappropriately large. Its RMSE is 0.7545 as C=2^8, indicating an accuracy loss at 20.17% with inappropriately shrunk regularization effects. Moreover, its RMSE increases to 1.1053 as C=+∞, indicating that the accuracy loss increases to 45.51% as regularization effects vanish. Similar situations are also encountered on the other testing cases, as shown in Figs. 5(a), (b) and (d). Moreover, from Fig. 6, we clearly observe the same phenomena are encountered as f decreases or increases.

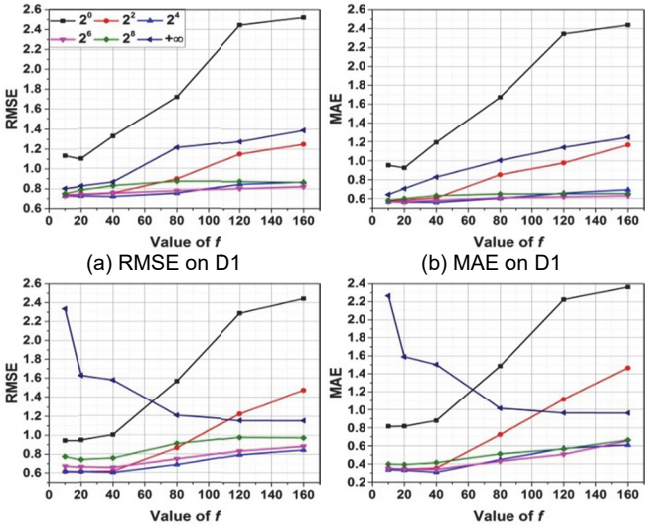
Hence, it turns out to be vital to incorporate regularization into an MLF model for preventing it from overfitting. Nonetheless, the incorporated regularization effects should be controlled with care. According to our experiments, the empirical values of C are in the interval of $[2^3, 2^5]$.

4) **An MLF model achieves slight accuracy gain with diversified regularization coefficients.** According to



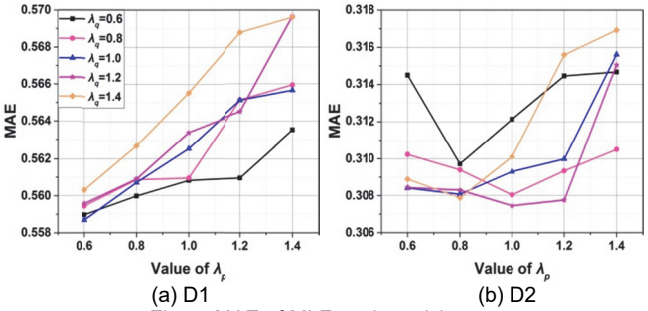
(a) RMSE on D1 (b) MAE on D1
(c) RMSE on D2 (d) MAE on D2

Fig. 5. MLF's prediction error as C in $\{2^0, 2^2, 2^4, 2^6, 2^8, +\infty\}$ and layer count increases. All panels share the legend in panel (a).



(a) RMSE on D1 (b) MAE on D1
(c) RMSE on D2 (d) MAE on D2

Fig. 6. MLF's lowest prediction error as C in $\{2^0, 2^2, 2^4, 2^6, 2^8, +\infty\}$ and f in [10, 160]. All panels share the legend in panel (a).



(a) D1 (b) D2

Fig. 7. MAE of MLF as λ_P and λ_Q vary.

(9), with a unique C we apply the same regularization coefficients to P and Q. Naturally, we can control the regularization effects from a finely-grained perspective via assigning diversified regularization coefficients to P and Q, thereby reformulating (9) as follows:

$$\varepsilon_{MLF}\left(P^n, Q^n\right) = \sum_{r_{u,i} \in R_K} \left(C\left(r_{u,i} - p^n_{u,\cdot} q^n_{i,\cdot}\right)^2 + \lambda_P \left\|p^n_{u,\cdot}\right\|_F^2 + \lambda_Q \left\|q^n_{i,\cdot}\right\|_F^2\right). \tag{17}$$

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: DOI 10.1109/TBDATA.2020.2988778, IEEE Transactions on Big Data

We have tested the performance of an MLF model relying on (17) as λ_P and λ_Q vary in the scale of (0.6, 1.4). Fig. 7 depicts its MAE on D1-2. Note that similar results can be observed on other datasets or with RMSE. From Fig. 7, we clearly see that accuracy gain can be achieved via applying different regularization coefficients on P and Q. For instance, the lowest RMSE achieved by an MLF model is 0.5587 with λ_P=0.6 and λ_Q=1.0. On the other hand, by setting λ_P=λ_Q=1.0, we arrive at the initial design of an MLF model with objective (9), whose MAE is 0.5625. Thus, the accuracy gain by finely-grained regularization coefficients is 0.68% in MAE. Similar situations are found on D2 as in Fig. 7(b). However, to achieve this slight accuracy gain requires grid search as in Fig. 7, which is time-inefficient. Hence, in the following experiments, we adopt (9) with a uniform regularization coefficient for an MLF model.

4.3 Comparison with State-of-the-art Models

In this part of tests, we compare the proposed MLF with several state-of-the-art LF models, whose details are summarized below.

1) **M1: SVD++.** A sophisticated LF model designed for HiDS data [11]. It adopts an SGD algorithm.
2) **M2: ELM.** An extreme learning machine model [24]. Its training process includes a mapping step and a solving step. However, it takes complete inputs. Therefore, we prefill missing data with zeroes for it.
3) **M3: ML-ELM.** A multilayered ELM model, which takes the output weights of the previous layer as the input of the next layer [22]. Similar to M2, M3 also takes complete inputs. Consequently, we fill the missing data in an HiDS matrix with zeroes for it.
4) **M4: I-AutoRec.** An LF model based on an autoencoder paradigm [39]. It takes the target HiDS matrix as its input and output simultaneously, and trains desired LFs via SGD-based backward propagation.
5) **M5: Adam-based I-AutoRec.** It is exactly the same with M4 except that the learning algorithm SGD is replaced with an Adaptive Moment Estimation (Adam) algorithm. M5 is included to show the effects brought by another iterative learning algorithm, i.e., Adam, which is more advanced and popular than a standard SGD algorithm.
6) **M6: NeuMF.** An LF model relying on deep neural network [41]. It explores the linear latent representation through LF analysis, and learns the nonlinear interaction hidden in HiDS data through a deeply-structure neural network. Its learning algorithm is the same with that of M4. Note that M6 is originally designed for ranking optimization issues with cross entropy loss as its learning objective to predict missing implicit feedbacks, which is not compatible with missing data estimation. Hence, we replace its original learning objective with a Euclidean distance-based one to make it compatible with our evaluation protocol.
7) **M7: DCCR.** A deep collaborative conjunctive model consists of two different types of neural network models, i.e., an autoencoder and a deeply-structured perceptron [65]. The former extracts user and item LFs, while the latter fuses user and item LFs to model their nonlinear interactions. Its learning algorithm is the same with that of M4.
8) **M8: MLF**. The proposed model of this study.

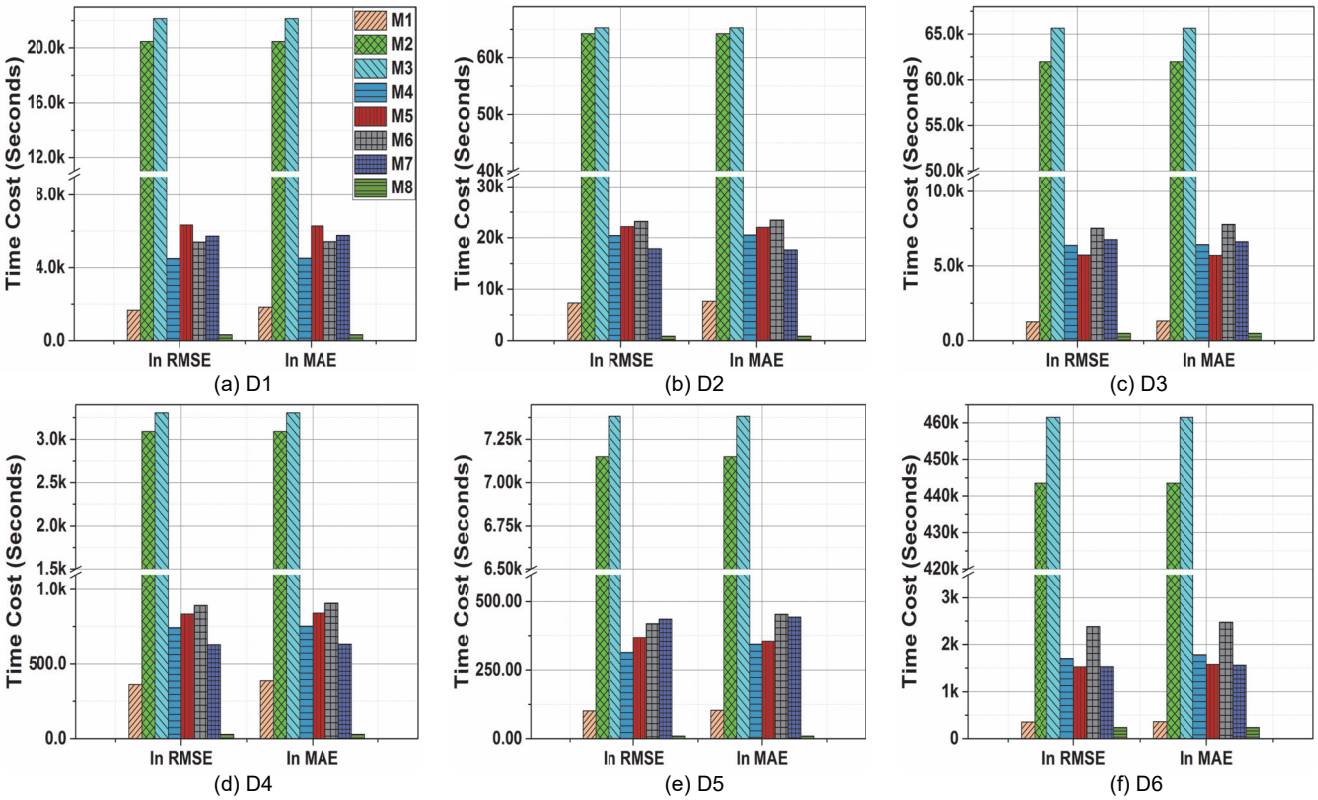
All experiments of these deep LF models are conducted on a server with a 3.60GHz Core i7-7820X CPU, 32GB RAM and GeForce RTX2080ti GPU. Note that M4-7 are all deeply-structured LF models relying on iterative learning algorithms like SGD or Adam. Moreover, M2-7 all relies on matrix-dependent operations during their training process, making them compatible with GPU framework. Hence, their training process are accelerated through GPU if the server and enjoys the speedup in the scale of [10.1, 11.2]. Considering the hyper parameters of each model, to ensure a fair and objective comparison, we adopt the following settings:

1) For M1-8, we compare their prediction accuracy and computational efficiency as f increases from 10 to 160 for thorough comparisons;
2) As analyzed in [11, 22, 24, 39, 41] and section 4.2, hyper parameters are vital for M1-8. On each data set for each involved model, we tune its hyper-parameters to achieve its highest performance on one fold, and then adopt the same setting on the remaining four folds. The final outcome is achieved by averaging the results from five folds of experiments. We aim to achieve objective results based on such settings.

Figs. 8-10 depict the time cost, RMSE and MAE of M1-8. Time II records the lowest prediction error of M1-8 along with the corresponding time costs. From these results, we have the following important findings:

1) **An MLF model's computational efficiency is significantly higher than its peers.** As depicted in Fig. 8 and Table II, M8, i.e., a proposed MLF model, takes 333.2 seconds to achieve its lowest RMSE on D1, which is only 20.04% of 1662.1 seconds by M1, 1.63% of 20473.5 seconds by M2, 1.50% of 22146.8 seconds by M3, 7.41% of 4495.6 seconds by M4, 5.27% of 6328.5 seconds by M5, 6.17% of 5394.8 seconds by M6, and 5.83% of 5712.3 seconds by M7, respectively. On D2, M8 consumes 885.3 seconds to achieve its lowest RMSE, which is only 12.06% of 7339.4 seconds by M1, 1.38% of 64217.6 seconds by M2, 1.36% of 65251.3 seconds by M3, 4.33% of 20430.2 seconds by M4, 3.99% of 22153.7 seconds by M5, 3.82% of 23187.1 seconds by M6, and 5.02% of 17853.6 seconds by M7, respectively. Similar results are found on D3-6, as in Fig. 8 and Table II.

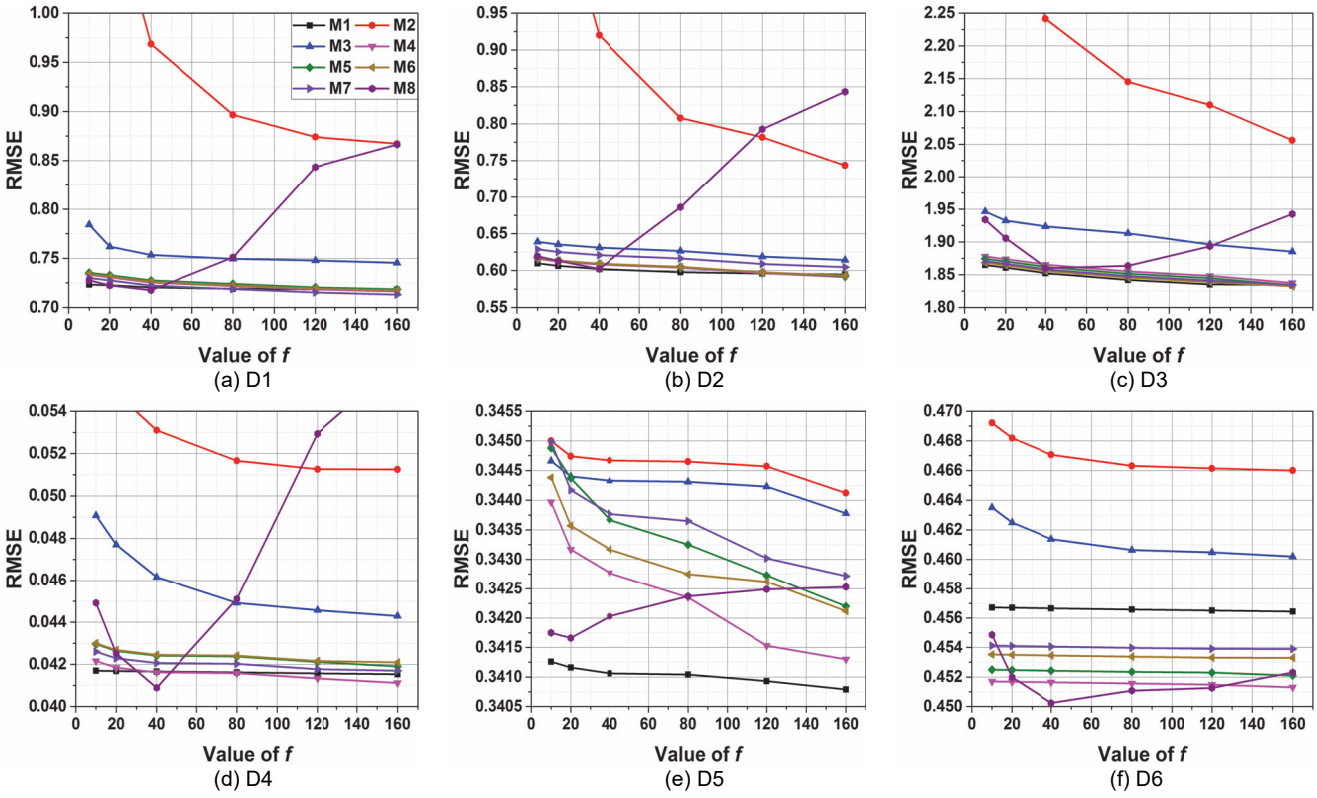
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2020.2988778, IEEE Transactions on Big Data



(a) D1 (b) D2 (c) D3

(d) D4 (e) D5 (f) D6

Fig. 8. Time cost of M1-8. All panels share the legend in panel (a).



(a) D1 (b) D2 (c) D3

(d) D4 (e) D5 (f) D6

Fig. 9. RMSE of M1-8. All panels share the legend in panel (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/TBDATA.2020.2988778, IEEE Transactions on Big Data

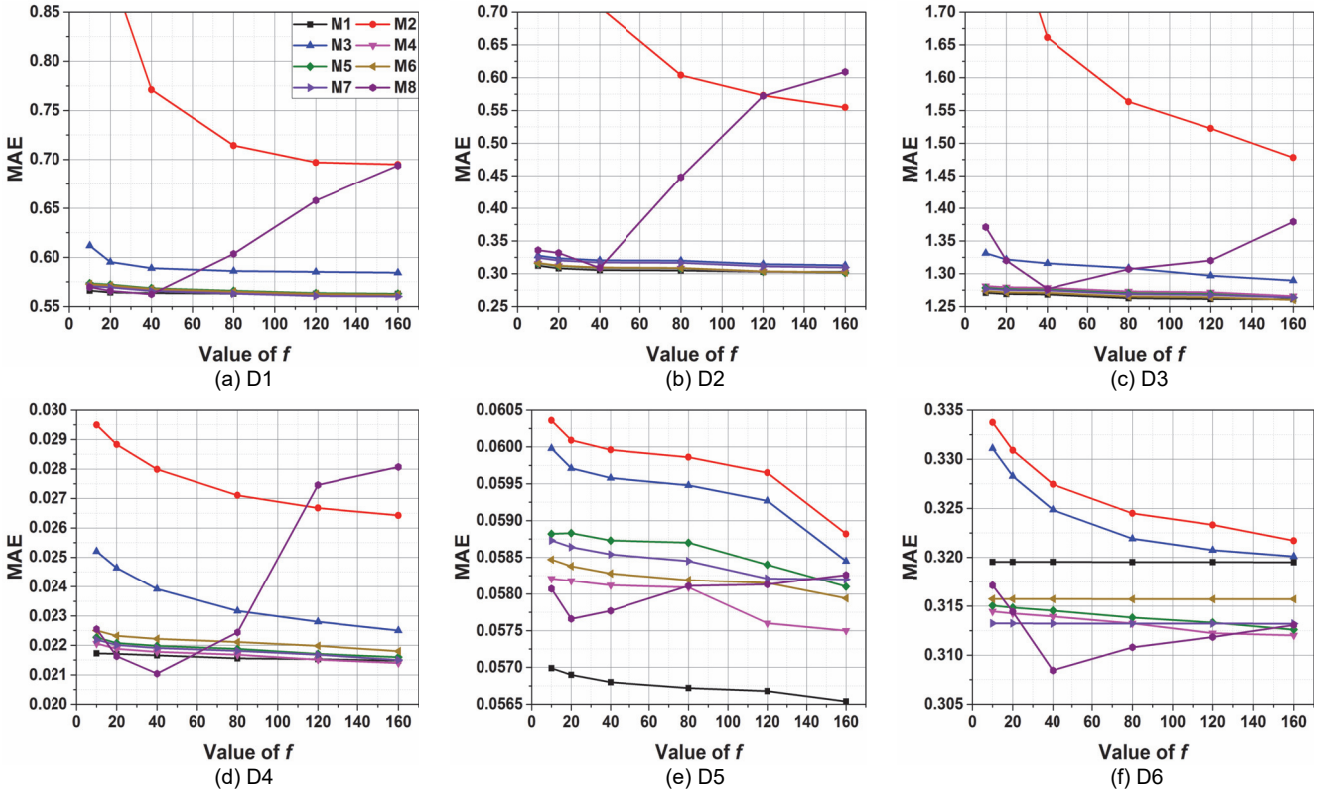


Fig. 10. MAE of M1-8. All panels share the legend in panel (a).

TABLE II. LOWEST PREDICTION ERROR ACHIEVED BY M1-8 AND THE CORRESPONDING TIME COSTS (SECONDS).

Case	M1	M2	M3	M4	M5	M6	M7	M8
D1: RMSE	0.7181	0.8672	0.7454	0.7163	0.7185	0.7171	**0.7132**	0.7176
D1: Time	1662.1	220090.2	224125.3	4495.6	6328.5	5394.8	5712.3	**333.2**
D1: MAE	0.5628	0.6949	0.5846	0.5616	0.5630	0.5621	**0.5602**	0.5625
D1: Time	1833.3	220090.2	224125.3	4514.1	6277.1	5418.4	5756.8	**333.2**
D2: RMSE	0.5946	0.7434	0.6144	**0.5912**	0.5922	0.5933	0.6045	0.6023
D2: Time	7339.4	664010.6	666871.5	20430.2	22153.7	23187.1	17853.6	**885.3**
D2: MAE	0.3026	0.5546	0.3127	**0.3013**	0.3015	0.3019	0.3095	0.3087
D2: Time	7637.2	664010.6	666871.5	20557.6	22011.3	23449.5	17648.2	**885.3**
D3: RMSE	1.8344	2.0562	1.8852	1.8376	1.8341	**1.8324**	1.8347	1.8599
D3: Time	1260.6	676650.7	679312.3	6367.1	5722.6	7513.1	6749.5	**485.7**
D3: MAE	1.2612	1.4781	1.2894	1.2658	1.2632	**1.2602**	1.2637	1.2767
D3: Time	1315.8	676650.7	679312.3	6422.5	5689.3	7768.5	6612.8	**485.7**
D4: RMSE	0.0415	0.0513	0.0443	0.0411	0.0419	0.0421	0.0417	**0.0409**
D4: Time	361.7	33449.2	33711.6	742.3	833.1	890.76	627.5	**28.4**
D4: MAE	0.0215	0.0264	0.0225	0.0214	0.0216	0.0218	0.0215	**0.0210**
D4: Time	387.3	33449.2	33711.6	751.7	838.7	906.44	631.3	**28.4**
D5: RMSE	**0.3407**	0.3441	0.3437	0.3413	0.3422	0.3421	0.3427	0.3417
D5: Time	101.1	75060.8	75302.3	313.9	368.2	419.2	435.7	**9.2**
D5: MAE	**0.0564**	0.0591	0.0584	0.0575	0.0581	0.0579	0.0582	0.0577
D5: Time	103.6	75060.8	75302.3	344.4	354.9	453.7	443.1	**9.2**
D6: RMSE	0.4564	0.4660	0.4602	0.4513	0.4521	0.4533	0.4539	**0.4502**
D6: Time	355.5	4923501.3	4924127.2	1701.8	1527.8	2382.5	1528.7	**237.7**
D6: MAE	0.3195	0.3217	0.3201	0.3120	0.3126	0.3157	0.3132	**0.3084**
D6: Time	360.5	4923501.3	4924127.2	1784.5	1579.3	2476.7	1561.2	**237.7**

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/TBDATA.2020.2988778, IEEE Transactions on Big Data

The reason why M8 achieves such a drastic efficiency gain over its peers is mainly owing to its carefully-designed learning scheme. Compared with M2 and M3, it implements its training process on the know entries of an HiDS matrix only, which is far less than its full size, i.e., $|U|\times|I|$. Compared with M1 and M4-7, its training process executes 'one iteration' only at each layer. Moreover, although it has a multilayered architecture, the hidden weights and biases are randomly generated instead of being updated by backward propagation. Hence, it is not surprising that M8, i.e., a proposed MLF model, outperforms its peers in terms of computational efficiency significantly.

2) **An MLF model's prediction accuracy for missing data of an HiDS matrix is comparable with that of the most accurate LF models relying on iterative learning algorithms.**

First of all, considering all the three randomized learning-based models, i.e., M2, M3 and M8, we clearly see that M8, i.e., a proposed MLF model, outperforms M2 and M3 in terms of prediction accuracy. For instance, as shown in Table II and Fig. 9(a), M8's RMSE is 0.7176, which is 3.73% lower than 0.7454 by M3, and 17.25% lower than 0.8672 by M2. Considering MAE, according to Table II and Fig. 10(a), M8's MAE is 0.5625, which is about 3.78% lower than 0.5846 by M3, and 19.05% lower than 0.6949 by M2. Similar situations can also found on the other testing cases. From these results, we can draw the following conclusions, a) the multilayered structure of M3 and M8 indeed enhance their representative learning ability over a single-layered model, i.e., M2, and b) owing to its specifically-designed learning process with three steps, i.e., *P*-Mapping, *Q*-Solving, and *P*-Enhancing, M8, i.e, a proposed MLF model, can better represent an HiDS matrix than M3, i.e., a multilayered ELM model does.

When compared with the most accurate LF models relying on widely-adopted iterative learning algorithms, i.e., SGD and Adam, an MLF model also achieves competitive prediction accuracy for missing data of an HiDS matrix. As shown in Table II, on all twelve testing cases, M8 achieves the lowest prediction error on four cases, i.e., on D4 and D6 in both RMSE and MAE. In comparison, M1, M4, M6 and M7 achieve the lowest prediction error respectively on two testing cases. Therefore, we reasonably conclude that M8, i.e., an MLF model's prediction accuracy is comparable with that of the most accurate LF models relying on iterative learning algorithms.

3) **Significance analyses demonstrate that an MLF model is highly efficient on an HiDS matrix.**

To better understand the comparison results, we conduct the Friedman test [44] to validate the statistical significance of the experimental results.

Let r_i^j be the rank of the jth of k compared models on the ith of N testing cases. The Friedman test compares the average rank of each involved model, i.e., $A_j = \sum_{j=1}^{N} r_i^j \Big/ N$. Hence, we summarize the average rank of each model in terms of its computational efficiency and prediction accuracy in Table III.

TABLE III. AVERAGE RANK OF EACH INVOLVED MODEL.

Average rank	M1	M2	M3	M4	M5	M6	M7	M8
Efficiency	2	7	8	3.83	4.42	5.58	4.17	1
Accuracy	3.63	8	7	2.33	3.92	3.58	4.21	3.33

Based on Table III, we compute the Friedman value as:

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j A_j^2 - \frac{k(k+1)^2}{4}\right],$$

and the testing score as:

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2}.$$

Note that F_F is distributed following the F-distribution with k–1 and (k–1)(N–1) degrees of freedom [45]. We can reject the null hypothesis that states the equivalence of compared models with the critical level α if F_F is greater than the corresponding critical value.

In our experiments, eight models are tested on six datasets. And on each dataset we have two independent cases respectively in RMSE and MAE. So we have k=8 and N=12, and F_F with (7, 77) degrees. The critical value of F_F(7, 77) for α=0.05 is 2.34. Thus, if the testing scores of our experiments are higher than 2.34, we reject the null hypothesis.

By substituting the average ranks of involved models in Table III into (18) and (19), the testing scores for efficiency and accuracy comparisons are 130.55 and 18.97, respectively. Both of them are much higher than the critical value 2.34. Thus, we assert that M1-8 are different with a confidence level at 95%.

For further identifying the performance of tested models, we adopt the Nemenyi test [45], where two models are significantly different if the difference between their performance ranks is greater than the critical difference value [45] given as:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}},$$

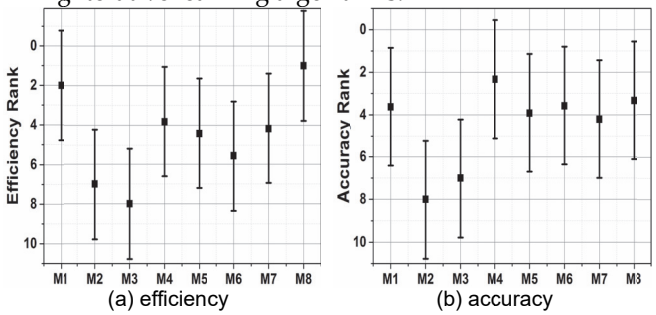
where q_α relies on the Studentized range statistic [45]. With eight models in our experiment, the critical value q_α is 2.78 with the critical level α=0.1 [45]. By substituting k=8, N=12, and q_α=2.78 into (20), we obtain that CD=2.78, which indicates that any pair of models with a rank difference higher than 2.78 have significant performance difference with the confidence of 90%.

The results of Nemenyi analysis are shown in Fig. 11. From Fig. 11(a), we sM8, i.e., a proposed MLF model, significantly outperforms M2-7 in terms of computa-

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/TBDATA.2020.2988778, IEEE Transactions on Big Data

tional efficiency. Its efficiency is clearly higher than that of M1 according to Fig. 8 and Tables II-III, but the rank difference is less than the critical value.

Considering prediction accuracy, as illustrated in Fig. 11(b), M1 and M4-8 significantly outperform M2 and M3; however, they do not have significance difference in prediction accuracy from each other. Hence, we arrive at the conclusion that M8's prediction accuracy is as high as that of the most accurate LF models adopting iterative learning algorithms.



(a) efficiency (b) accuracy

Fig. 11. Results of Nemenyi analysis.

4.4 Summary

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

1) It is critical to incorporate regularization effects into an MLF model for improving its generality. The regularization effects should be controlled with care to enable its best performance. To diversify its regularization coefficients on different LF matrices is helpful in further improvement of its prediction accuracy. However, the tuning process can be time consuming. Self-adaptive regularization strategies are highly desired and should be investigated in the future.
2) Owing to its carefully designed learning scheme, an MLF model's computational efficiency is significantly higher than that of the state-of-the-art LF models.
3) Owing to its generally deep structure, an MLF model's prediction accuracy is comparable with that of the most accurate LF models.
4) An MLF model's perdition accuracy for missing data of an HiDS matrix is not always proportional to its LF dimension. In contrast, it suffers accuracy loss as its LF dimension becomes too large. This might be caused by the non-linearity brought by the activation function, however, detailed reasons for this phenomenon remains unveiled and calls for our further efforts.

5 Conclusions

An MLF model has high computational efficiency as well as competitive prediction accuracy for missing data of an HiDS matrix. Compared with existing LF and randomized learning models, its virtues lay in the following perspectives:

1) It combines the virtues of randomized learning in computational efficiency and generalized multilayered structure in representative learning ability.
2) Different from an ELM model or its multilayered extension that take complete data as inputs, an MLF model is specifically designed for an HiDS matrix with numerous missing data. It addresses an HiDS matrix at the computational and storage costs linear with its known entry count. Moreover, its training process consists of three steps, i.e., *P*-mapping, *Q*-solving, and *P*-enhancing, thereby greatly improving its representation learning ability on sparse data.

However, the following issues remain open:

1) An MLF model's prediction accuracy decreases when its LF dimension exceeds a certain threshold. However, as indicated in prior research [43, 46], an LF model's representative learning ability should be enhanced as its LF dimension increases in a linear LF model. This phenomenon may rely on the non-linearity introduced by an activation function during the *P*-mapping step, or inappropriately tuned regularization effects. It is desired to figure out the connections among an MLF models' representative learning ability, activation function, randomized mapping rules and regularization scheme.
2) It will be highly interesting to integrate constraints like symmetry [43] and non-negativity [14] into an MLF model for improving its representative learning ability on specific kinds of sparse data like an undirected, weighted and sparse network [43].
3) To implement the domain-specific self-adaptation [47, 50, 52] of an MLF model's diversified regularization parameters can be helpful in improving its practicability and model generality.
4) To incorporate parallelization or distribution mechanism [55-59] into its learning scheme can further improve its computational efficiency, especially when f grows large. Moreover, according to the previous study [56], the compressed storage strategies for sparse data can greatly reduce the computational and communication costs for a paralleled or distributed learning model. Hence, it becomes necessary to investigate the compatibility between an MLF model and sparse data compressing strategies for efficiency gain.
5) To extend the methodology of an MLF model for addressing the issue of latent factorization of HiDS tensors [66] is highly interesting.

We plan to address the above issues in the future.

Code Availability

The code of an MLF model along can be found via https://github.com/yuanye1080/MLF.

ACKNOWLEDGMENTS

This research is supported in part by the National Natural Science Foundation of China under grants 61772493, 91646114, 51609229 and 61872065, in part by the Natural Science Foundation of Chongqing (China) under grant cstc2019jcyjjqX0013, and in part by the Pioneer Hundred Talents Program of Chinese Academy of Sciences

REFERENCES

[1] J. Wu, L. Chen, Y. P. Feng, Z. B. Zheng, M. C. Zhou, and Z. Wu, "Predicting quality of service for selection by neighborhood-based collaborative-filtering," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 443, pp. 428-439, Mar., 2013.

[2] Z. B. Zheng, H. Ma, M. R. Lyu, and I. King, "QoS-aware web service recommendation by collaborative filtering," *IEEE Trans. on Services Computing*, vol. 4, pp. 140-152, Apr., 2011.

[3] G. Resta and P. Santi, "WiQoSM: An Integrated QoS-Aware Mobility and User Behavior Model for Wireless Data Networks," *IEEE Trans. on Mobile Computing*, vol. 7, no. 2, pp. 187-198, Feb., 2007.

[4] 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. 1-8, Oct., 2013.

[5] 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, May, 2016.

[6] X. Fu, L. Wei, Z. Li, et al, "Noise-tolerant Wireless Sensor Networks Localization via Multi-norms Regularized Matrix Completion," *IEEE Trans. on Vehicular Technology*, vol. 67, no. 3, pp. 2409-2419, Mar., 2018.

[7] 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, pp. 734-749, Jun., 2005.

[8] C. Shulong and P. Yuxing, "Matrix Factorization for Recommendation with Explicit and Implicit Feedback," *Knowledge-Based Systems*, vol. 158, pp. 109-117, Oct., 2018.

[9] S. Seo, J. Huang, H. Yang, et al, "Representation learning of users and items for review rating prediction using attention-based convolutional neural network," *in Proc. of the 3rd Int. Conf. on Machine Learning Methods for Recommender Systems*, 2017.

[10] Z. Yang, B. Wu, K. Zheng, X. Wang, and L. Lei, "A Survey of Collaborative Filtering-Based Recommender Systems for Mobile Internet Applications," *IEEE Access*, vol. 4, pp. 3273-3287, May, 2016.

[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] B. K. Patra, R. Launonen, V. Ollikainen, et al, "A new similarity measure using Bhattacharyya coefficient for collaborative filtering in sparse data," *Knowledge-Based Systems*, vol. 82, pp. 163-177, Jul., 2015.

[13] X. Luo, M. C. Zhou, Y. N. Xia, and Q. S. Zhu, "An Efficient Non-Negative Matrix-Factorization-Based Approach to Collaborative Filtering for Recommender Systems," *IEEE Trans. on Industrial Informatics*, vol. 10, no. 2, pp. 1273-1284, May., 2014.

[14] X. Luo, M. C. Zhou, S. Li, Z. H. You, Y. N. Xia, and Q. S. Zhu, "A Nonnegative Latent Factor Model for Large-Scale Sparse Matrices in Recommender Systems via Alternating Direction Method," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 27, no. 3, pp. 579-592, Mar., 2016.

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

[16] G. Takács, I. Pilászy, B. Németh, and D. Tikk, "Scalable collaborative filtering approaches for large recommender systems," *Journal of Machine Learning Research*, vol. 10, pp. 623-656, 2009.

[17] N. J. Guliyev and V. E. Ismailov, "A Single Hidden Layer Feedforward Network with Only One Neuron in the Hidden Layer Can Approximate Any Univariate Function," *Neural Computation*, vol. 28, no. 7, pp. 1-16, 2015.

[18] B. Igelnik and Y. H. Pao, "Stochastic Choice of Basis Functions in Adaptive Function Approximation and the Functional-Link Net," *IEEE Trans. on Neural Networks*, vol. 6, no. 6, pp. 1320-1329, 1995.

[19] A. N. Gorban, I. Y. Tyukin, D. V. Prokhorov, and K. I. Sofeikov, "Approximation with random bases: Pro et Contra," *Information Sciences*, vol. 364, pp.129-145, 2016.

[20] D. Bacciu, M. Colombo, D. Morelli, et al, "Randomized neural networks for preference learning with physiological data," *Neurocomputing*, vol. 298, pp. 9-20, 2018.

[21] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, pp. 489-501, 2006.

[22] J. X. Tang, C. Deng, and G. B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 27, no. 4, pp. 809-821, 2016.

[23] S. Li, Z. H. You, H. Guo, X. Luo, and Z. Q. Zhao, "Inverse-Free Extreme Learning Machine With Optimal Information Updating," *IEEE Trans. on Cybernetics*, vol. 46, no. 5, pp. 1229-1241, 2016.

[24] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks[C]," *in Proc. Of the IEEE Int. Joint Conf. on Neural Networks*, pp. 985-990, 2005.

[25] G. Huang, G. B. Huang, S. Song, et al, "Trends in extreme learning machines: A review," *Neural Networks*, vol. 61, pp. 32-48, Jan., 2015.

[26] Z. H. Zhou and J. Feng, "Deep forest: Towards an alternative to deep neural networks," *arXiv preprint arXiv*:1702.08835, 2017.

[27] B. Yi, X. Shen, H. Liu, et al, "Deep Matrix Factorization with Implicit Feedback Embedding for Recommendation System," *IEEE Trans. on Industrial Informatics*, pp. 1-1. Jan., 2019.

[28] Y. F. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," *in Proc. of the 8th IEEE Int. Conf. on Data Mining*, pp. 263-272, 2008.

[29] 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, Mar., 2016.

[30] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," *Advances in Neural Information Processing Systems*, vol. 29, no. 2, pp. 1257–1264, 2007.

[31] Z. Yang, Y. Zhang, W. Yan, et al, "A Fast Non-Smooth Nonnegative Matrix Factorization for Learning Sparse Representation," *IEEE Access*, vol. 4, pp. 5161-5168, Sep., 2016.

[32] T. Himabindu, V. Padmanabhan, and K. Pujari, "Conformal Matrix Factorization based Recommender System," *Information Sciences*, vol. 467, pp. 685-707, Oct., 2018.

[33] S. Zhang, W. Wang, J. Ford J, and F. Makedon, "Learning from Incomplete Ratings Using Non-negative Matrix Factorization," *in Proc. of the 6th SIAM Int. Conf. on Data Mining*, pp. 549—553, Apr., 2006.

[34] J. Yang and J. Leskovec, "Overlapping community detection at scale: a nonnegative matrix factorization approach," *in Proc. of the 6th ACM Int. Conf. on Web search and data mining*, pp. 587-596, Feb., 2013.

[35] R. Narayanam and Y. Narahari, "A shapley value-based approach to discover influential nodes in social networks," *IEEE Trans. on Automa-*

tion Science and Engineering, vol. 8, no. 1, pp. 130–147, Jul., 2010.

[36] S. Miao, Z. J. Wang, and R. Liao, "A CNN Regression Approach for Real-time 2D/3D Registration," *IEEE Trans. on Medical Imaging*, vol. 35, pp. 5, pp. 1352-1363, 2016.

[37] K. Zhang, W. Zuo, and Y. Chen, "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising," *IEEE Trans. on Image Processing*, vol. 26, no. 7, pp. 3142-3155, 2017.

[38] H. Wu, Z. Zhang, J. Luo, et al, "Multiple Attributes QoS Prediction via Deep Neural Model with Contexts," *IEEE Trans. on Services Computing*, 2018.

[39] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "AutoRec: Autoencoders Meet Collaborative Filtering," *in Proc. of the 24th Int. Conf. on World Wide Web*, pp. 111-112, 2015.

[40] T. A. Davis and Y. Hu, "The university of florida sparse matrix collection," *ACM Trans. on Mathematical Software*, vol. 38, no. 1, pp. 1–25, 2011.

[41] X. He, L. Liao, H. Zhang, et al, "Neural collaborative filtering," *in Proc. Of the 26th Int. Conf. on World Wide Web*, pp. 173-182, 2017.

[42] X. Ge, Q. L. Han, and Z. Wang, "A Dynamic Event-Triggered Transmission Scheme for Distributed Set-Membership Estimation Over Wireless Sensor Networks," *IEEE Trans. on Cybernetics*, vol. 49, no. 1, pp. 171-183, Jan., 2019.

[43] X. Luo, J. Sun, Z. Wang, S. Li, and M. Shang, "Symmetric and non-negative latent factor models for undirected, high dimensional and sparse networks in industrial applications," *IEEE Trans. on Industrial Informatics*, vol. 13, no. 6, pp. 3098-3107, Dec., 2017.

[44] M. Friedman, "A Comparison of Alternative Tests of Significance for the Problem of m Rankings," *Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86-92, 1940.

[45] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. 1, pp. 1-30, 2006.

[46] Y. Koren, "Collaborative filtering with temporal dynamics," *in Proc of the 15th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining*, pp. 447-456, 2010.

[47] P. R. Lorenzo, J. Nalepa, M. Kawulok, L. S. Ramos, and J. R. Paster, "Particle swarm optimization for hyper-parameter selection in deep neural networks," *in Proc. of the ACM Int. Conf. on Genetic and Evolutionary Computation*, 2017, pp. 481-488.

[48] Y. Bengio, A. Courville, and P. Vincent, "Representation Learning: A Review and New Perspectives," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798-1828, 2013.

[49] P. C. Hansen, "Rank-deficient and discrete ill-posed problems: numerical aspects of linear inversion," *Society for Industrial and Applied Mathematics*, 1998.

[50] G. Fernando, F. Cláudio F, and M. Zbigniew, "Parameter Setting in Evolutionary Algorithms," *Springer Science and Business Media*, 2007.

[51] D. L. Donoho, Y. Tsaig, I. Drori, et al, "Sparse Solution of Underdetermined Systems of Linear Equations by Stagewise Orthogonal Matching Pursuit," *IEEE Trans. on Information Theory*, vol. 58, no. 2, pp. 1094-1121, 2012.

[52] J. J. Wang and T. Kumbasar, "Parameter optimization of interval Type-2 fuzzy neural networks based on PSO and BBBC methods," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 1, pp. 247–257, 2019.

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

[54] X. Luo, Y. Yuan, M. C. Zhou, Z. G. Liu, and M. S. Shang, "Non-Negative Latent Factor Model Based on β-Divergence for Recommender Systems," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, 10.1109/TSMC.2019.2931468, 2019.

[55] H. Li, K. Li, J. An, W. Zheng, and K. Li, "An efficient manifold regularized sparse non-negative matrix factorization model for large-scale recommender systems on GPUs," *Information Sciences*, vol. 496, pp. 464-484, Sep., 2019.

[56] Y. Chen, K. Li, W. Yang, G. Xiao, X. Xie, and T. Li, "Performance-aware model for sparse matrix-matrix multiplication on the sunway TaihuLight supercomputer," *IEEE Trans. on Parallel and Distributed Systems*, vol. 30, no. 4, pp. 923-938, Sep., 2018.

[57] G. Xiao, K. Li, Y. Chen, W. He, A. Zomaya, and T. Li, "CASpMV: a customized and accelerative SPMV framework for the sunway TaihuLight," *IEEE Trans. on Parallel and Distributed Systems*, 10.1109/TPDS.2019.2907537, Mar., 2019.

[58] H. Li, K. Li, J. An, and K. Li, "MSGD: A novel matrix factorization approach for large-scale collaborative filtering recommender systems on gpus," *IEEE Trans. on Parallel and Distributed Systems*, vol. 29, no. 7, pp. 1530-1544, Jun., 2017.

[59] H. Li, K. Li, J. An, and K. Li, "An Online and Scalable Model for Generalized Sparse Non-negative Matrix Factorization in Industrial Applications on Multi-GPU," *IEEE Trans. on Industrial Informatics*, 10.1109/TII.2019.2896634, Jan., 2019.

[60] P. R. Lorenzo, J. Nalepa, M. Kawulok, L. S. Ramos, and J. R. Paster, "Particle swarm optimization for hyper-parameter selection in deep neural networks," *in Proc. of the ACM Int. Conf. on Genetic and Evolutionary Computation*, pp. 481-488, 2017.

[61] T. Nathanson, E. Bitton, and K. Goldberg, "Eigentaste 5.0: constant-time adaptability in a recommender system using item clustering," *in Proc. of the ACM Conf. on Recommender systems*, pp. 149-152., 2007.

[62] 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.

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

[64] L. Yang, X. Cao, D. Jin, X. Wang, and D. Meng, "A unified semisupervised community detection framework using latent space graph regularization," *IEEE Trans. on Cybernetics*, vol. 45, no. 11, pp. 2585–2598, 2015

[65] Q. Wang, B. Peng, X. Shi, T. Shang, and M. Shang, "DCCR: deep collaborative conjunctive recommender for rating prediction," *IEEE Access*, vol. 7, pp. 60186-60198, 2019.

[66] X. Luo, H. Wu, H. Q. Yuan, and M. C. Zhou, "Temporal Pattern-aware QoS Prediction via Biased Non-negative Latent Factorization of Tensors," *IEEE Trans. on Cybernetics*, DOI 10.1109/TCYB.2019.2903736.

[67] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. A. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: A System for Large-Scale Machine Learning," *in Proc. of the 12th USENIX Symp. on Operating Systems Design and Implementation*, pp. 265-283, 2016.

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.

Qiang He (M'11) received his first PhD degree from Swinburne University of Technology, Australia, in 2009 and his second PhD degree in computer science and engineering from Huazhong University of Science and Technology, China, in 2010. He is a senior lecturer at Swinburne. His research interests include service computing, software engineering, cloud computing and edge computing. More details about his research can be found at https://sites.google.com/site/heqiang/.

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 40 IEEE TRANSACTIONS papers) in the above areas. Dr. Luo was a recipient of the Hong Kong Scholar Program jointly by the Society of Hong Kong Scholars and China Post-Doctoral Science Foundation in 2014, the Pioneer Hundred Talents Program of Chinese Academy of Sciences in 2016, 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 reward of IEEE ACCESS in 2018. He has also served as the Program Committee Member for over 20 international conferences.

Mingsheng Shang received his Ph.D Degree in Computer Science from University of Electronic Science and Technology of China (UESTC). He joined the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China, in 2015, and is currently a Professor of computer science and engineering. His research interests include data mining, complex networks, cloud computing and their applications.