A Deep Latent Factor Model for High-Dimensional and Sparse Matrices in Recommender Systems

Di Wu, *Student Member, IEEE*, Xin Luo, *Senior Member, IEEE*, Mingsheng Shang, Yi He, Guoyin Wang, *Senior Member, IEEE*, and MengChu Zhou, *Fellow, IEEE*

***Abstract*—Recommender systems (RSs) commonly adopt a user-item rating matrix to describe users' preferences on items. With users and items exploding, such a matrix is usually high-dimensional and sparse (HiDS). Recently, the idea of deep learning has been applied to RSs. However, current deep-structured RSs suffer from high computational complexity. Enlightened by the idea of deep forest, this paper proposes a deep latent factor model (DLFM) for building a deep-structured RS on an HiDS matrix efficiently. Its main idea is to construct a deep-structured model by sequentially connecting multiple latent factor (LF) models instead of multilayered neural networks through a nonlinear activation function. Thus, the computational complexity grows linearly with its layer count, which is easy to resolve in practice. The experimental results on four HiDS matrices from industrial RSs demonstrate that when compared with state-of-the-art LF models and deep-structured RSs, DLFM can well balance the prediction accuracy and computational efficiency, which well fits the desire of industrial RSs for fast and right recommendations.**

***Index Terms*—Big data, deep model, high-dimensional and sparse (HiDS) matrix, latent factor (LF) analysis, recommender system (RS).**

Manuscript received May 31, 2019; accepted July 19, 2019. Date of publication August 15, 2019; date of current version June 16, 2021. This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFC0804002, in part by the National Natural Science Foundation of China under Grant 61702475, Grant 61772493, and Grant 91646114, in part by the Chongqing Basic Research and Frontier Exploration under Grant cstc2019jcyj-msxm1750, in part by the Chongqing Overseas Scholars Innovation Program under Grant cx2017012 and Grant cx2018011, in part by the Chongqing Research Program of Key Standard Technologies Innovation of Key Industries under Grant cstc2017zdcy-zdyfX0076 and Grant cstc2018jszx-cyztzxX0025, in part by the Chongqing Research Program of Technology Innovation and Application under Grant cstc2017rgzn-zdyfX0020, Grant cstc2017zdcy-zdyf0554, and Grant cstc2017rgzn-zdyf0118, and in part by the Pioneer Hundred Talents Program of Chinese Academy of Sciences. This paper was recommended by Associate Editor X. Wang. *(Di Wu and Mingsheng Shang are co-first authors of this paper.) (Corresponding author: Xin Luo.)*

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

X. Luo is with the Chongqing Engineering Research Center of Big Data Application for Smart Cities, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China, also with the Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China, and also with the Department of Computing, Hong Kong Polytechnic University, Hong Kong 999077 (e-mail: luoxin21@cigit.ac.cn).

M. Shang and G. Wang are with the Chongqing Engineering Research Center of Big Data Application for Smart Cities, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China, and also with the Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China (e-mail: msshang@cigit.ac.cn; wanggy@ieee.org).

Y. He is with the School of Computing and Informatics, University of Louisiana at Lafayette, Lafayette, LA 70503 USA (e-mail: yi.he1@louisiana.edu).

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

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSMC.2019.2931393.

Digital Object Identifier 10.1109/TSMC.2019.2931393

I. Introduction

IN THIS era of big data, people suffer the serious problem of information overload [1]. They are inundated by big data and have been desiring for intelligent systems that can truly find useful information from big data for them [2]. Recommender systems (RSs), which can guide people to obtain desired information out of billions of bytes, have been proven to be highly useful in addressing this issue [3]. Nowadays, RSs have been widely adopted in numerous online services, such as social media sites, e-commerce, and online news [4], [5].

So far, various models are proposed for implementing RSs, where a very important branch is collaborative filtering (CF) [6]–[17]. Commonly, a CF-based RS is developed based on a user-item rating matrix [18]–[20], where each column denotes a specified item (e.g., book, music, and movie), each row denotes a specified user, and each entry denotes the corresponding user's evaluation on the corresponding item. Since the number of items can be very large in an industrial RS like Taobao [21], it is impossible for a user to touch all of them. Therefore, high-dimensional and sparse (HiDS) matrices with numerous missing data are frequently encountered in CF-based RSs [22]–[24].

A CF-based RS essentially exploits an HiDS matrix to predict involved users' potential preferences. In other words, the main task of a CF-based RS is to perform missing data estimation for an HiDS matrix subject to globally high accuracy and other requirements, like high computational and storage efficiency [18], [19]. Great efforts have been made to address this issue, resulting in a pyramid of CF-based models [3]. Among them, one highly important and successful branch is

the latent factor (LF) models that are highly accurate and scalable under many circumstances [18], [19], [25]–[27].

An LF model (LFM) works by building a low-rank approximation to a given HiDS matrix based on its known entries only. It maps both users and items into the same low-dimensional LF space, trains desired LFs on the existing ratings, and then predicts the missing ratings in the given HiDS matrix heavily relying on these obtained LFs. So far, many sophisticated LFMs are developed to handle large-scale HiDS matrices, including a bias-based LFM [25], a probabilistic one [28], a nonparametric LFM [29], a regularized LFM [30], a nonnegativity-constrained LFM [18], a neighborhood-integrated LFM [13], a clustering-preconditioned LFM [31], and a generalized momentum-incorporated LFM [14].

Recently, deep neural networks (DNNs) have attracted much attention from both academic and industrial communities owing to their incredible abilities in representation learning [32]. To date, they have been successfully applied to many fields [33]–[37] including CF-based RSs [37]–[45]. Representative models of this kind include an autoencoder-based model [38], a denoising autoencoder-based model [39], a collaborative denoising autoencoder-based model [40], a stacked autoencoder-based model [44], a neural CF-based model [41], a recurrent neural network-based model [42], a deep matrix factorization with neural network-based model [43], and a hybrid deep structure-based model [45].

Although DNNs have achieved great success in various applications, they have obvious defects [46] which also exist in the CF-based RSs with DNNs: 1) they require massive training instances, resulting in extremely high computational cost; 2) they contain too many hyper-parameters, which requires very careful parameter tuning; and 3) they take complete data as inputs, which are unavailable in RS. A CF-based RS with DNNs needs to prefill a given HiDS matrix's unknown data with zeroes or other statistics of the observed data, resulting in unnecessarily high cost in both computation and storage [41], [43]. For instance, the MovieLens 20M matrix (used in this paper) collected by GroupLens has 20 000 263 instances scattering in 138 493 rows and 26 744 columns [47]. Its data density is 0.54% only, but the total number of its entries is more than 3.7 billion. To manipulate such a huge and full matrix is greatly difficult and sometimes impossible.

For adapting the principle of deep learning to more general machine learning scenarios, Zhou and Feng [46] proposed a deep forest to address the aforementioned defects of DNNs. It connects a series of decision trees to obtain a deeply structured model. With such designs, a general learning model can also achieve highly competitive representation learning ability. Thus, it becomes possible to improve a general learning model's ability in representation learning via adopting the principle of the deep forest [46]. Enlightened by the principle of deep forest, we propose a deep LF model (DLFM) for processing HiDS matrices from RSs for the first time. The main idea of DLFM is to construct an LF hierarchy for enhancing the resultant model's representation learning ability. The main contributions of this paper include.

1) A DLFM with excellent abilities in representing an HiDS matrix.
2) Detailed algorithm design and analyzes of DLFM.
3) Detailed empirical studies regarding DLFM's performance on four HiDS matrices generated by RSs currently in use.

Note that DLFM is significantly different from the existing CF-based RSs with DNNs in the following aspects: 1) its deep structure is implemented following the principle of a deep forest, thereby its learning process does not depend on a back-propagation process and 2) it carefully handles incomplete data of an HiDS matrix, thereby achieving high computational and storage efficiency.

The rest of this paper is organized as follows. Section II states the problem formulation. Section III presents DLFM. Section IV provides the experimental results. Section V discusses related issues. Finally, Section VI concludes this paper.

II. Problem Formulation

An RS commonly involves two large entity sets, i.e., a user set U and an item set I. Given them, we recall the definitions of an HiDS rating matrix [18], [26].

Definition 1: A rating matrix $R^{|U|\times|I|}$ has its each element $r_{u,i}$ quantify the preference by user $u \in U$ on item $i \in I$. Let R_K and R_U, respectively, denote its known and unknown entry sets, it is an HiDS rating matrix with U and I being large and $|R_K| \ll |R_U|$.

Note that in an HiDS rating matrix R, its most entries are unknown rather than zeroes in a traditional sparse matrix. To extract desired LFs from it, we recall the definition of an LFM.

Definition 2: Given R, U, I, f, an LFM seeks for LF matrices $P^{|U|\times f}$ and $Q^{|I|\times f}$ to form R's rank-f approximation $\hat{R} = PQ$ where $f \ll \min\{|U|, |I|\}$.

Note that f denotes the LF space dimension, and LF matrices P and Q actually reflect the characteristics of U and I hidden in R_K, respectively. To extract P and Q from R_K, an objective function that measures the difference between R and $\hat{R}$ is desired. Commonly, such an objective function can be modeled by the Euclidean distance or Kullback–Leibler divergence [18]–[20]. With the former, we have

$$\varepsilon = \frac{1}{2}\sum_{r_{u,i}\in R_K}\left(r_{u,i}-\hat{r}_{u,i}\right)^2 = \frac{1}{2}\sum_{r_{u,i}\in R_K}\left(r_{u,i}-\sum_{k=1}^{f}p_{u,k}q_{k,i}\right)^2 \quad (1)$$

where $r_{u,i}$, $\hat{r}_{u,i}$, $p_{u,k}$, and $q_{k,i}$ denote involved entries in R, $\hat{R}$, P, and Q, respectively. As indicated in [25] and [26], it is important to integrate the Tikhonov regularization into (1) to improve its generality

$$\varepsilon = \frac{1}{2}\sum_{r_{u,i}\in R_K}\left(\left(r_{u,i}-\sum_{k=1}^{f}p_{u,k}q_{k,i}\right)^2+\lambda\left(\sum_{k=1}^{f}p_{u,k}^2+\sum_{k=1}^{f}q_{k,i}^2\right)\right) \quad (2)$$

where λ is the regularization constant controlling the regularization effects. Note that in (2) we follow the strategies

mentioned in [48] and [49] to connect a specified regularization term, i.e., $(\sum_{k=1}^{f} p_{u,k}^2 + \sum_{k=1}^{f} q_{k,i}^2)$ on each $r_{u,i} \in R_K$, for describing the imbalanced density of known data in R.

With (2) we formulate the problem of an LFM. Next, we present our method of a DLFM.

III. Deep Latent Factor Model

A. Learning Rule With Stochastic Gradient Descent

As discussed in prior studies [25], [26], stochastic gradient descent (SGD) is highly efficient in solving a bilinear objective function (2). With it, we consider the instant loss $\forall r_{u,i} \in R_K$

$$\varepsilon_{u,i} = \frac{1}{2}\left(r_{u,i} - \sum_{k=1}^{f} p_{u,k} q_{k,i}\right)^2 + \frac{\lambda}{2}\left(\sum_{k=1}^{f} p_{u,k}^2 + \sum_{k=1}^{f} q_{k,i}^2\right). \tag{3}$$

Then, LFs involved in (3) are trained by moving them along the opposite direction against the stochastic gradient of (3) with respect to each single LF

$$\forall k \in \{1, 2, \ldots, f\} : \begin{cases} p_{u,k} \leftarrow p_{u,k} - \eta \frac{\partial \varepsilon_{u,i}}{\partial p_{u,k}} \\ q_{k,i} \leftarrow q_{k,i} - \eta \frac{\partial \varepsilon_{u,i}}{\partial q_{k,i}} \end{cases} \tag{4}$$

where η denotes the learning rate. Let $\hat{r}_{u,i} = \sum_{k=1}^{f} p_{u,k} q_{k,I}$, then stochastic gradients in (4) are given by

$$\begin{aligned} \frac{\partial \varepsilon_{u,i}}{\partial p_{u,k}} &= -q_{k,i}\left(r_{u,i} - \hat{r}_{u,i}\right) + \lambda p_{u,k} = -q_{k,i}\Delta_{u,i} + \lambda p_{u,k} \\ \frac{\partial \varepsilon_{u,i}}{\partial q_{k,i}} &= -p_{u,k}\Delta_{u,i} + \lambda q_{k,i} \end{aligned} \tag{5}$$

where $\Delta_{u,i} = r_{u,i} - \hat{r}_{u,i}$ is the prediction error on $r_{u,i}$. Thus, we formulate the sum error between R and $\hat{R}$ on R_K as follows:

$$E = \sum_{r_{u,i} \in R_K} \left(r_{u,i} - \hat{r}_{u,i}\right)^2. \tag{6}$$

B. Deep Structure of DLFM

Following the principle of a deep forest, we sequentially connecting LF models with different N layers and $N-1$ nonlinear activation functions. Given $N|U| \times f$ user LF matrices $\{P_1, P_2, \ldots, P_N\}$, $Nf \times |I|$ item LF matrices $\{Q_1, Q_2, \ldots, Q_N\}$, and $N|U| \times |I|$ low-rank approximation matrices $\{\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_N\}$, DLFM seeks for P_n and Q_n to obtain approximation $\hat{R}_n$ for R first and then selects the best approximation $\hat{R}_f$ as the final output, where $n \in \{1, 2, \ldots, N\}$. Its structure is given in Fig. 1.

As shown in Fig. 1, a DLFM works as follows.

1) R_K is taken as the initial input of the first layer.
2) Training LF matrices P_n and Q_n based on the input of the nth layer and the learning rule mentioned in Section III-A, we obtain $\hat{R}_n$.
3) Mixing the information in $\hat{R}_n$ and R with a nonlinear activation function for enhancing the generality of a DLFM.
4) The output of the activation function is taken as the input of the next layer.
5) Repeating steps 2)–4) to achieve $\{\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_N\}$, and the best approximation $\hat{R}_f$ is selected from them as the final output.

Formally, the above process is given by

$$\begin{aligned} &\arg\min_{P_n, Q_n} \varepsilon(P_n, Q_n) \\ &= \begin{cases} \sum_{r_{u,i} \in R_K} \left(\left(r_{u,i} - \sum_{k=1}^{f} p_{u,k}^1 q_{k,i}^1\right)^2 \right. \\ \qquad \left. + \lambda\left(\sum_{k=1}^{f}\left(p_{u,k}^1\right)^2 + \sum_{k=1}^{f}\left(q_{k,i}^1\right)^2\right)\right), & \text{if } n = 1 \\ \sum_{r_{u,i} \in R_K} \left(\left(\tilde{r}_{u,i}^{n-1} - \sum_{k=1}^{f} p_{u,k}^n q_{k,i}^n\right)^2 \right. \\ \qquad \left. + \lambda\left(\sum_{k=1}^{f}\left(p_{u,k}^n\right)^2 + \sum_{k=1}^{f}\left(q_{k,i}^n\right)^2\right)\right), & \text{otherwise} \end{cases} \end{aligned} \tag{7}$$

where $p_{u,k}^n$ and $q_{k,i}^n$ are single LFs in P_n and Q_n with $n \in \{1, 2, \ldots, N\}$, and $\tilde{r}_{u,i}^{n-1}$ is the output of the nonlinear activation function designed as follows:

$\forall (u, i) \in R_K \ \forall n \in \{2, 3, \ldots, N\}$:

$$\tilde{r}_{u,i}^{n-1} = \Phi\left(\hat{r}_{u,i}^{n-1}, r_{u,i}\right) = \begin{cases} r_{u,i} & \text{if } \hat{r}_{u,i}^{n-1} < r_{\min} \\ r_{u,i} & \text{if } \hat{r}_{u,i}^{n-1} > r_{\max} \\ \hat{r}_{u,i}^{n-1} & \text{if otherswise} \end{cases} \tag{8}$$

where $\hat{r}_{u,i}^{n-1}$ is a single element in $\hat{R}_{n-1}$, and $r_{\min}$ and $r_{\max}$ denote the maximum and minimum values in R_K, respectively. Note that if $\hat{r}_{u,i}^{n-1} < r_{\min}$ or $\hat{r}_{u,i}^{n-1} > r_{\max}$, it is obvious that the predictions generated by the $n-1$th layer are not correct. Hence, the physical meaning of (8) is to reset the extremely unreasonable predictions generated by the current layer to make the next layer describe R more precisely.

Then we consider the instant loss of (7) on each $r_{u,i}$

$$\varepsilon_{u,i}^n = \begin{cases} \left(r_{u,i} - \sum_{k=1}^{f} p_{u,k}^1 q_{k,i}^1\right)^2 + \lambda\left(\sum_{k=1}^{f}\left(p_{u,k}^1\right)^2 + \sum_{k=1}^{f}\left(q_{k,i}^1\right)^2\right) \\ \qquad \text{if } n = 1 \\ \left(\tilde{r}_{u,i}^{n-1} - \sum_{k=1}^{f} p_{u,k}^n q_{k,i}^n\right)^2 + \lambda\left(\sum_{k=1}^{f}\left(p_{u,k}^n\right)^2 + \sum_{k=1}^{f}\left(q_{k,i}^n\right)^2\right) \\ \qquad \text{otherwise.} \end{cases} \tag{9}$$

We train each desired LF associated with $r_{u,i}$ as follows:

$$\forall k \in \{1, 2, \ldots, f\} : \begin{cases} p_{u,k}^n \leftarrow p_{u,k}^n - \eta \frac{\partial \varepsilon_{u,i}^n}{\partial p_{u,k}^n} \\ q_{k,i}^n \leftarrow q_{k,i}^n - \eta \frac{\partial \varepsilon_{u,i}^n}{\partial q_{k,i}^n} \end{cases} \tag{10}$$

where the stochastic gradients are, respectively, given by

$$\begin{aligned} \text{if } n = 1 : & \begin{cases} \frac{\partial \varepsilon_{u,i}^1}{\partial p_{u,k}^1} = -q_{k,i}^1\left(r_{u,i} - \hat{r}_{u,i}^1\right) + \lambda p_{u,k}^1 \\ \qquad\qquad = -q_{k,i}^1 \Delta_{u,i}^1 + \lambda p_{u,k}^1 \\ \frac{\partial \varepsilon_{u,i}^1}{\partial q_{k,i}^1} = -p_{u,k}^1 \Delta_{u,i}^1 + \lambda q_{k,i}^1 \end{cases} \\ \text{otherwise} : & \begin{cases} \frac{\partial \varepsilon_{u,i}^n}{\partial p_{u,k}^n} = -q_{k,i}^n\left(\tilde{r}_{u,i}^{n-1} - \hat{r}_{u,i}^n\right) + \lambda p_{u,k}^n \\ \qquad\qquad = -q_{k,i}^n \Delta_{u,i}^n + \lambda p_{u,k}^n \\ \frac{\partial \varepsilon_{u,i}^n}{\partial q_{k,i}^n} = -p_{u,k}^n \Delta_{u,i}^n + \lambda q_{k,i}^n \end{cases} \end{aligned} \tag{11}$$

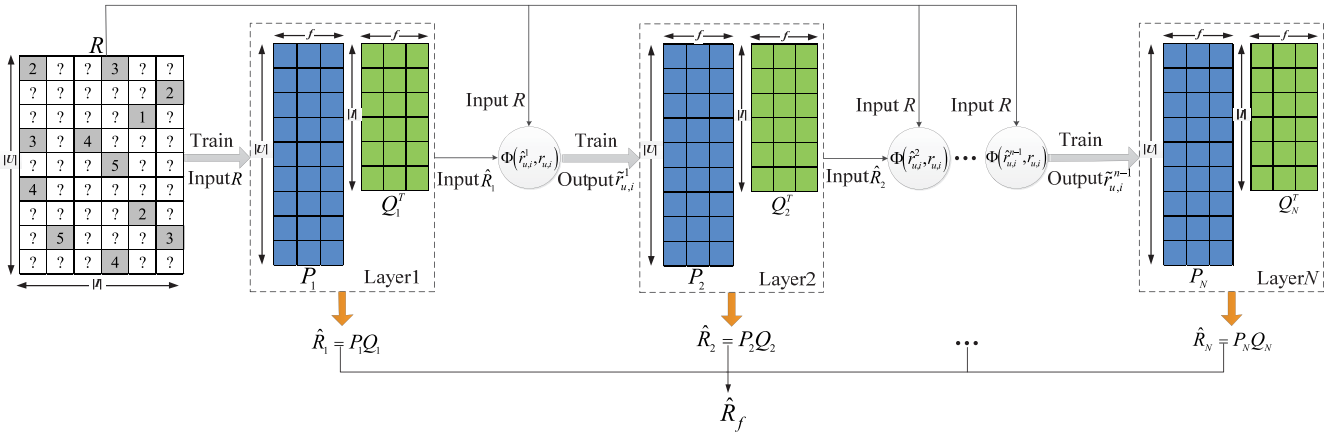

Fig. 1. Structure and processing flow of DLFM.

where $\Delta_{u,i}^{n}$ denotes the prediction error between $\hat{r}_{u,i}^{n}$ and $\hat{r}_{u,i}^{n}$. Thus, $\hat{R}_n$ is obtained as

$$\hat{R}_n = P_n Q_n \tag{12}$$

and the generalized error E_n between R and $\hat{R}_n$ is given by

$$E_n = \sum_{r_{u,i} \in R_K} \left(r_{u,i} - \hat{r}_{u,i}^{n}\right)^2. \tag{13}$$

Finally, we select the best approximation $\hat{R}_f$ for R with the lowest generalized error E_f

$$E_f \Leftarrow \min\{E_1, E_2, \ldots, E_N\}. \tag{14}$$

C. Algorithm Design and Analysis

Based on Section III-B, we design Algorithm 1. We can derive its computational complexity C as follows:

$$\begin{aligned} C &= \Theta(1) + N \times (\Theta(|U| \times f) + \Theta(f \times |I|)) \\ &\quad + N_m \times (|R_K| \times N \times (f \times 2 \times \Theta(1) + \Theta(f)) + 1) \\ &\quad + N \times \Theta(|R_K|) + \Theta(N-1) \\ &= \Theta(N \times f \times (|U| + |I|) \\ &\quad + N_m \times (|R_K| \times N \times 3f + 1) + N \times |R_K| + N) \\ &\approx \Theta(N_m \times |R_K| \times N \times f). \end{aligned} \tag{15}$$

From (15), we see that the maximum iteration count N_m, known entry count $|R_K|$, layer count N, and LF space dimension f decide the computational complexity of Algorithm 1. As analyzed in [26], the computational complexity of an LFM with SGD is $\Theta(N_m \times |R_K| \times f)$. Thus, the computational cost of DLFM is N times than that of an LFM. Obviously, the extra computing burden of DLFM is caused by its deep structure and decided by its layer count. However, such extra cost is linear with N. Given that an LFM is highly efficient in addressing an HiDS, the additional cost by DLFM is acceptable in practice.

IV. Empirical Studies

A. Evaluation Protocol

We choose the task of missing data estimation on HiDS matrices [18], [19] as an evaluation protocol. When addressing such a task, each involved model's prediction accuracy for missing data is the key performance indicator, which is commonly reflected by the root mean squared error (RMSE) and the mean absolute error (MAE) [14], [18], [19], [22]–[24]

$$\text{RMSE} = \sqrt{\left(\sum_{r_{w,j} \in \Gamma} \left(r_{w,j} - \hat{r}_{w,j}\right)^2\right) \Big/ |\Gamma|}$$

$$\text{MAE} = \left(\sum_{r_{w,j} \in \Gamma} \left|r_{w,j} - \hat{r}_{w,j}\right|_{abs}\right) \Big/ |\Gamma|$$

where Γ denotes the testing set and $|\cdot|_{abs}$ denotes the absolute value of a given number. The lower RMSE and MAE denote higher prediction accuracy. In this paper, we use RMSE_n and MAE_n to denote the RMSE and MAE of nth layer of DLFM

$$\text{RMSE}_n = \sqrt{\left(\sum_{r_{w,j} \in \Gamma} \left(r_{w,j} - \hat{r}_{w,j}^{n}\right)^2\right) \Big/ |\Gamma|}$$

$$\text{MAE}_n = \left(\sum_{r_{w,j} \in \Gamma} \left|r_{w,j} - \hat{r}_{w,j}^{n}\right|_{abs}\right) \Big/ |\Gamma|.$$

All experiments are run on a PC with 3.4 GHz i7 CPU and 64 GB RAM. In addition, all models are implemented in JAVA SE 7U60 to check their suitability for industrial usage.

B. Datasets

Four benchmark datasets, which are HiDS matrices from industrial RSs, are selected to conduct the experiments to validate the effectiveness of DLFM.

1) *D1 (Flixter Dataset):* Collected by the Flixter website [47], it contains 8 196 077 ratings in the range of [0.5, 5] from 147 612 users on 48 794 movies. Its density is 0.11% only.
2) *D2 (Jester 1.1 M Dataset):* Collected by the joke-recommender Jester [47], it contains 1 186 324 continuous ratings in the range of $[-10, 10]$ from 24 983 users on 100 jokes. Its data density is 47.32%. Note that D2 is relatively denser than the other three datasets. We choose

Algorithm 1: DLFM

Input: R_K

Operation	Cost
Initializing $f, \lambda, \eta, N_m = \max - \text{training} - \text{round}$	$\Theta(1)$
for $n = 1$ to N	$\times N$
Initializing P_n randomly	$\Theta(\lvert U\rvert \times f)$
Initializing Q_n randomly	$\Theta(f \times \lvert I\rvert)$
end for	$--$
while $t = N_m$ && not converge	$\times N_m$
for $\forall r_{u,i} \in R_K$	$\times \lvert R_K\rvert$
for $n = 1$ to N	$\times N$
for $k = 1$ to f	$\times f$
computing $p^n_{u,k}$ according to (10) and (11)	$\Theta(1)$
computing $q^n_{k,i}$ according to (10) and (11)	$\Theta(1)$
end for	$--$
computing $\hat{r}^n_{u,i}$ according to (12)	$\Theta(f)$
end for	$--$
end for	$--$
$t = t + 1$	$\Theta(1)$
end while	$--$
for $n = 1$ to N	$\times N$
computing E_n according to (13)	$\Theta(\lvert R_K\rvert)$
end for	$--$
selecting $\hat{R}_f$ according to (14)	$\Theta(N-1)$

Output: $\hat{R}_f$

it to test the performance of DLFM on a different kind of HiDS matrices.

3) *D3 (MovieLens 10M Dataset):* Collected by the MovieLens system [50], it contains 10 000 054 ratings in the range of [0, 5] from 69 878 users on 10 677 movies. Its rating density is 1.34% only.
4) *D4 (MovieLens 20M Dataset):* Collected by the MovieLens system [50], it contains 20 000 263 ratings in the range of [0, 5] from 138 493 users on 26 744 movies. Its rating density is 0.54% only.

To eliminate the influence of ratings with different value domains, we map the ratings of D2 into the range of [0, 5]. On all datasets, we adopt the 80%–20% train-test settings and fivefold cross-validations. The detailed evaluation is executed as follows: 1) each dataset is randomly split into five disjoint subsets, each of which contains its 20% data; 2) four subsets are selected as the training set and the remaining one as the testing set; 3) build each tested model on the training set and test its performance on the testing set; 4) sequentially repeating steps 2) and 3) for five times to ensure that each fold can serve as the testing set once; and 5) compute the average of outcomes on all five folds to achieve final results.

C. Effects of Layer Count in DLFM

First, we analyze the performance of DLFM regarding its layer count. In this set of experiments, the hyper-parameters of DLFM are set as $\lambda = 0.01$, $\eta = 0.01$, $f = 20$, and $N = 10$, uniformly. Figs. 2 and 3 show the training process of DLFM at different layers. Figs. 4 and 5 denote the RMSE and MAE of DLFM as the layer becomes deeper. Note that with only one layer, a DLFM degenerates into an original LFM (OLFM). Hence, we adopt the results at the first layer of DLFM as the baseline. From Figs. 2–5, we have the following important findings.

1) At each layer of DLFM, RMSE, and MAE keep decreasing with more training iterations. As the layer count increases, the decreasing rate of RMSE and MAE becomes lower. However, the model convergence is always ensured at each layer, as shown in Fig. 2. Hence, DLFM's deep design does not affect its base model's convergence at each layer.
2) RMSE and MAE of DLFM decrease first and then increase with its layer count. The reason for this phenomenon is quite complex, and we provide related discussions regarding it in Section V discussions of this paper. For RMSE, the best layer count of DLFM is the 5 on D1 (0.9041), 7 on D2 (1.0038), 4 on D3 (0.7875), and 4 on D4 (0.7802), respectively. Compared with the baseline, the accuracy gain by DLFM with the best layer count is 3.90%, 0.51%, 1.83%, and 2.35% on D1–D4, respectively. For MAE, DLFM has its best layer count at 4 on D1 (0.6552), 2 on D2 (0.7761), 3 on D3 (0.6066), and 3 on D4 (0.5968), respectively. With the best layer count, DLFM outperforms the baseline with 3.46%, 0.31%, 1.61%, and 1.97% in MAE on D1–D4, respectively.

D. Effects of LF Dimension in DLFM

This set of experiments focus on the prediction accuracy of DLFM as f increases from 5 to 80, as shown in Figs. 6 and 7. The other hyper-parameters are set as $\lambda = 0.01$, $\eta = 0.01$, and $N = 10$, uniformly.

As depicted in Figs. 6 and 7, as f increases, the representation learning ability of DLFM gets better, making the resultant model achieve higher prediction accuracy. However, such accuracy gain is much more obvious when f increases from 5 to 20. As f is over 20, DLFM's improvement in RMSE and MAE becomes small. One possible reason for this phenomenon is that when $f = 20$, the representation learning ability of DLFM becomes strong enough, making it represent a target HiDS matrix precisely. Consequently, the continuous increase of f after 20 cannot bring significant improvement in prediction accuracy.

E. Effects of Regularization Constant λ in DLFM

This set of experiments focus on the effects of λ in DLFM. The other hyper-parameters are set as $\eta = 0.01$ and $f = 20$, uniformly. Figs. 8 and 9 show the experimental results as λ increases from 0.002 to 0.02. We have recorded the RMSE and MAE measured for DLFM with the best layer count as λ changes, along with the corresponding layer count. From these results, we have the following interesting findings.

1) The best layer count of DLFM decreases as λ increases. For example, on D1, DLFM achieves the lowest RMSE with 21 layers when $\lambda = 0.002$. However, when $\lambda = 0.02$, the best layer count decreases from 21 to 3. Similar results are observed on D2-4. They indicate that an under-regularized DLFM requires more layers

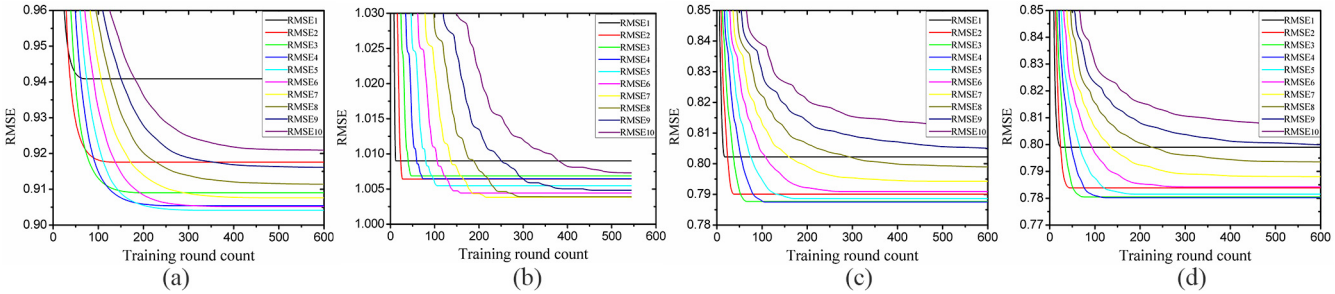

Fig. 2. Training process of DLFM at different layers in RMSE. Pannel settings: (a) D1, (b) D2, (c) D3, and (d) D4.

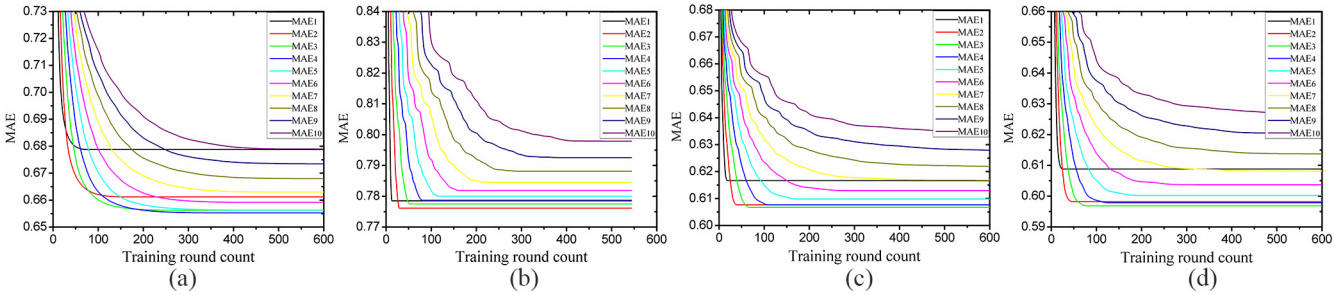

Fig. 3. Training process of DLFM at different layers in MAE. Pannel settings: (a) D1, (b) D2, (c) D3, and (d) D4.

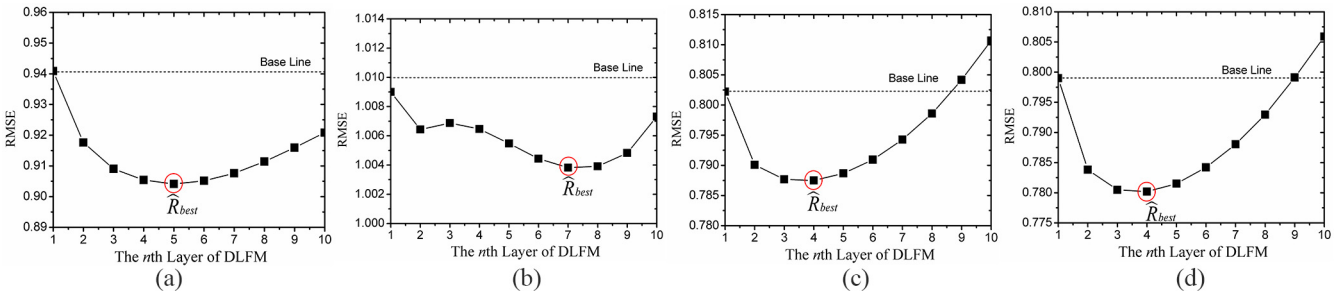

Fig. 4. RMSE of DLFM as layer count N changes. Pannel settings: (a) D1, (b) D2, (c) D3, and (d) D4.

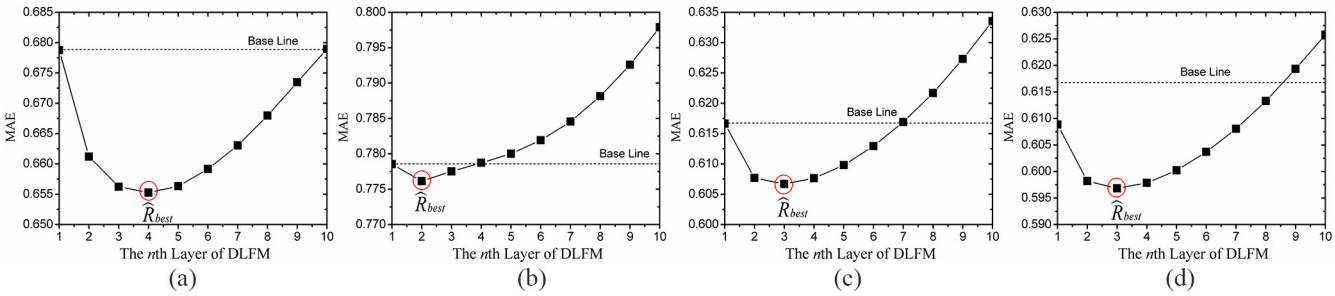

Fig. 5. MAE of DLFM as layer count N changes. Pannel settings: (a) D1, (b) D2, (c) D3, and (d) D4.

for improving the model generality; however, an over-regularized DLFM cannot improve its representativeness for an HiDS matrix with more layers.

2) λ is crucial for the prediction accuracy of DLFM. As λ increases from 0.002 to 0.02, the RMSE and MAE of DLFM decreases at first and then increases on all datasets. For RMSE, the best λ is 0.004 on D1, D2, and D4, and 0.006 on D3. For MAE, the best λ is 0.004 on D1 and D4, and 0.006 on D2 and D3. Like in standard LFMs, λ should be chosen with care to ensure high prediction accuracy for missing data of an HiDS matrix.

F. Comparison Between DLFM and State-of-the-Art Models

Five models, i.e., an OLFM [25], a regularized single-element-based non-negative matrix factorization (RSNMF) [19], a fast non-negative LFM based on generalized momentum (FNLF) [14], an item-oriented auto encoder-based recommender (I-AutoRec) [38], and a neural network-based matrix factorization (NeuMF) [41],

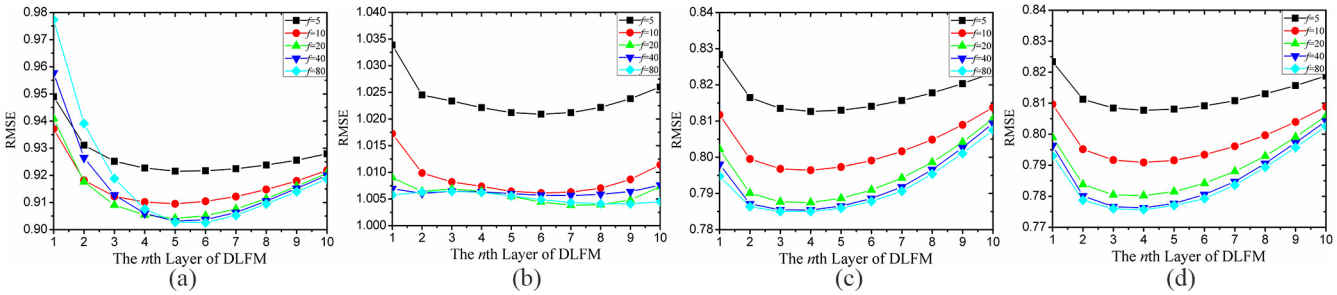

Fig. 6. RMSE of DLFM as LF dimension f changes. Pannel settings: (a) D1, (b) D2, (c) D3, and (d) D4.

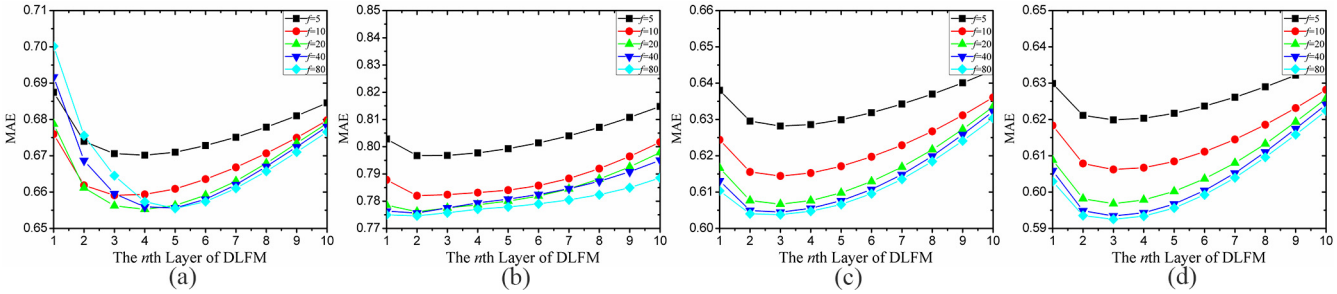

Fig. 7. MAE of DLFM as LF dimension f changes. Pannel settings: (a) D1, (b) D2, (c) D3, and (d) D4.

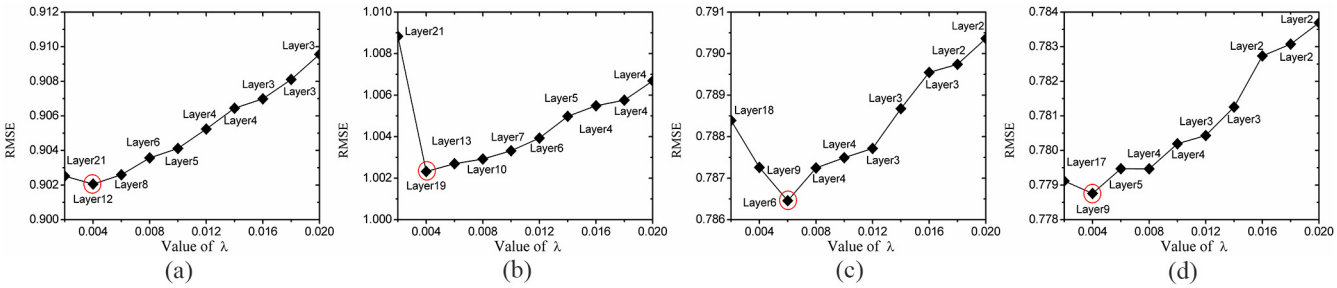

Fig. 8. RMSE of DLFM as regularization constant λ changes. Pannel settings: (a) D1, (b) D2, (c) D3, and (d) D4.

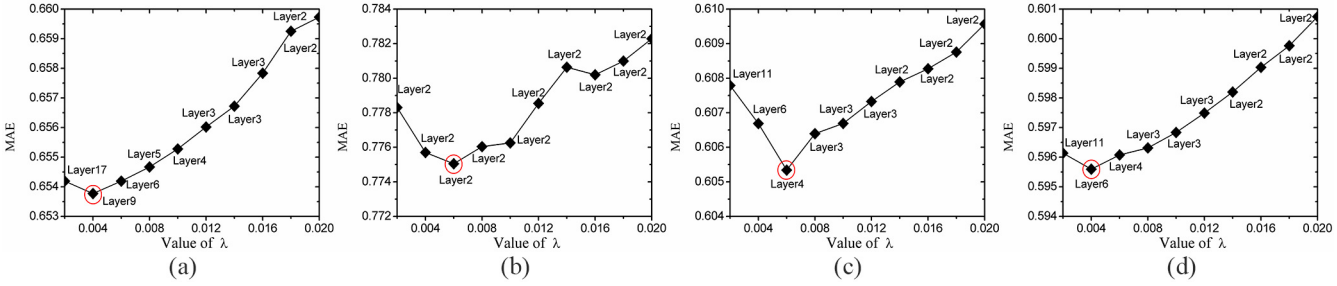

Fig. 9. MAE of DLFM as regularization constant λ changes. Pannel settings: (a) D1, (b) D2, (c) D3, and (d) D4.

are compared with DLFM in terms of prediction accuracy for missing data along with computational efficiency. All are sophisticated models able to well represent an HiDS matrix. Note that OLFM, RSNMF, and FNLF are single-layer models while I-AutoRec and NeuMF are DNNs-based ones.

As indicated by prior research [19], [25], [38] and analyzes in previous sections, the regularization constant λ plays a vital role in deciding an LFM's performance. To draw fair comparisons, we tune it to achieve the best performance for all five models as summarized in Tables I and II. Regarding the other hyper-parameters, we set $f = 20$ for OLFM, RSNMF, FNLF, and DLFM; $\eta = 0.01$ for OLFM and DLFM; $\gamma = 1.1$ for FNLF following [14]. For I-AutoRec, we set its hidden unit count at 500 following [33]. Note that NeuMF is originally designed for ranking optimization with a cross-entropy function as its objective, which cannot guarantee its prediction accuracy for missing data of an HiDS matrix. For making it comparable with its peers, we replace its objective function with a Euclidean distance-based one. Moreover, we set predictive factors at 64 and tune its hyper-parameters following [41].

TABLE I
LOWEST RMSE OF COMPARED MODELS

Dataset	OLFM	RSNMF	FNLF	I-AutoRec	NeuMF	DLFM
D1	0.9224	0.9056	0.9038	**0.8938**	0.9174	0.9020
	(λ=0.04)	(λ=0.05)	(λ=0.06)	(λ=100)	(λ=0.01)	(λ=0.004)
D2	1.0087	1.0049	1.0025	1.0078	1.1047	**1.0023**
	(λ=0.02)	(λ=0.08)	(λ=0.08)	(λ=100)	(λ=0.01)	(λ=0.004)
D3	0.7981	0.7893	0.7881	0.7872	0.7977	**0.7864**
	(λ=0.03)	(λ=0.05)	(λ=0.04)	(λ=100)	(λ=0.01)	(λ=0.006)
D4	0.7935	0.7819	0.7798	0.7802	0.7922	**0.7787**
	(λ=0.03)	(λ=0.05)	(λ=0.04)	(λ=100)	(λ=0.01)	(λ=0.004)

TABLE II
LOWEST MAE OF COMPARED MODELS

Dataset	OLFM	RSNMF	FNLF	I-AutoRec	NeuMF	DLFM
D1	0.6697	0.6550	0.6520	**0.6472**	0.6626	0.6537
	(λ=0.03)	(λ=0.03)	(λ=0.04)	(λ=100)	(λ=0.01)	(λ=0.004)
D2	0.7801	0.7769	0.7778	0.7905	0.7982	**0.7750**
	(λ=0.01)	(λ=0.04)	(λ=0.03)	(λ=100)	(λ=0.01)	(λ=0.006)
D3	0.6144	0.6080	0.6068	0.6062	0.6121	**0.6053**
	(λ=0.02)	(λ=0.04)	(λ=0.03)	(λ=100)	(λ=0.01)	(λ=0.006)
D4	0.6067	0.5977	0.5961	**0.5947**	0.6054	0.5956
	(λ=0.02)	(λ=0.04)	(λ=0.03)	(λ=100)	(λ=0.01)	(λ=0.004)

TABLE III
TIME COST (IN SECONDS) OF COMPARED MODELS

Dataset	OLFM	RSNMF	FNLF	I-AutoRec	NeuMF	DLFM
D1	58.78	406.94	900.12	5806.43	7014.32	1560.32
D2	19.65	35.67	10.3530	6990.14	5932.18	163.24
D3	56.23	442.21	902.87	50318.24	60267.25	549.42
D4	120.32	850.35	1305.48	60282.63	72168.56	1153.56

Tables I and II record the prediction accuracy of all compared models. We see that DLFM achieves the lowest RMSE on D2-4, and the lowest MAE on D2-3. In comparison, I-AutoRec achieves the lowest RMSE on D1, and the lowest MAE on D1 and D4. OLFM, RSNMF, FNLF, and NeuMF cannot achieve so high prediction accuracy as their peers do.

Table III records the time cost of all compared models. We find that: 1) I-AutoRec and NeuMF consume much more time than their peers do due to their DNNs-based learning strategy; 2) DLFM has relatively higher time cost than OLFM and RSNMF because of its deep structure; and 3) DLFM has comparable time cost to FNLF.

To better understand the comparison results, we conduct the Friedman test [51], which is effective in validating the performance of multiple models on multiple datasets, to analyze their statistical significance. First, we compute the average rank on prediction accuracy (RMSE and MAE recorded in Tables I and II) and computational efficiency (time cost recorded in Table III) of all compared models. The computed average ranks are shown in Table IV.

TABLE IV
TESTED MODELS' AVERAGE RANKING IN PREDICTION ACCURACY AND COMPUTATIONAL EFFICIENCY

Average rank in	OLFM	RSNMF	FNLF	I-AutoRec	NeuMF	DLFM
Prediction Accuracy	5.63	3.63	2.63	2.38	5.25	1.50
Time Cost	1.25	2.25	3.00	5.25	5.75	3.50

Let a_i^j be the rank of the jth one of p tested models on the ith one of q testing cases; the Friedman test compares the average ranks of involved models, i.e., $A_j = \sum_{i=1}^{q} a_i^j/q$. Under the null hypothesis, which states that all the models are equivalent and so their average ranks A_j should be equal, the Friedman value is

$$\chi_F^2 = \frac{12q}{p(p+1)}\left[\sum_{j=1\sim p} A_j^2 - \frac{p(p+1)^2}{4}\right]. \tag{16}$$

With (16), the testing score is given by

$$F_F = \frac{(q-1)\chi_F^2}{q(p-1)-\chi_F^2}. \tag{17}$$

Note that (17) is distributed according to the F-distribution with $p-1$ and $(p-1)(q-1)$ degrees of freedom [51]. Thus, we can reject the null hypothesis with the critical level α if F_F is greater than the corresponding critical value.

In the experiments, six models are tested on four datasets. Note that, on each dataset, we have two independent testing cases for prediction accuracy while one testing case for computational efficiency. Hence, we have $p = 6$ and $q = 8$, and F_F with (5, 35) degrees of freedom for prediction accuracy. For computational efficiency, we have $p = 6$ and $q = 4$, and F_F with (5, 15) degrees of freedom. The critical value of $F(5, 35)$ and $F(5, 15)$ for $\alpha = 0.05$ is 2.49 and 2.90, respectively. Therefore, if the testing scores of our experiments are greater than them, we reject the null hypothesis.

According to Table IV and following (16) and (17), we compute the testing scores for prediction accuracy and computational efficiency. They are 24.61 and 18. Both testing scores are far greater than 2.49 and 2.90, respectively. Thus, we assert that the tested models are significantly different in prediction accuracy with a confidence level at 95%.

Moreover, to check whether DLFM has the significantly better performance than each single compared model, we also conduct the Wilcoxon signed-ranks test [51], [52] on prediction accuracy (RMSE and MAE recorded in Tables I and II) and computational efficiency (time cost recorded in Table III) of all compared models. Wilcoxon signed-ranks test is a nonparametric pairwise comparison procedure and has three indicators: $R+$, $R-$, and p-value. The larger $R+$ value indicates higher performance and p-value indicates the significance level. Table V records the statistical results. From it, we find that: 1) DLFM has significantly better prediction accuracy than OLFM, RSNMF, FNLF, and NeuMF with a confidence level at 95%; 2) DLFM achieves higher $R+$ rankings

TABLE V
RESULTS OF WILCOXON SIGNED-RANKS TEST

Comparison	Prediction accuracy			Time cost		
	R+	*R*-	*p*-value	*R*+	*R*-	*p*-value
DLFM vs. OLFM	36	0	**0.0039**	0	10	1.0000
DLFM vs. RSNMF	36	0	**0.0039**	0	10	1.0000
DLFM vs. FNLF	30.5	5.5	**0.0430**	6	4	0.6875
DLFM vs. I-AutoRec	20.5	15.5	0.3867	10	0	**0.0625**
DLFM vs. NeuMF	36	0	**0.0039**	10	0	**0.0625**

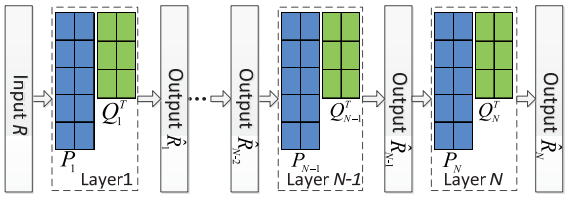

Fig. 10. Training process of DLFM.

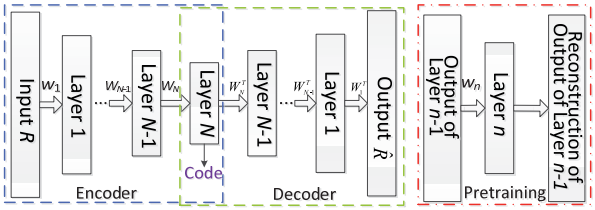

Fig. 11. Training process of I-AutoRec.

than I-AutoRec, which means that DLFM has slightly better prediction accuracy than I-AutoRec; 3) DLFM's computational efficiency is significantly higher than that of I-AutoRec and NeuMF with a confidence level at 90%; 4) OLFM and RSNMF have significantly higher computational efficiency than DLFM; and 5) DLFM has slightly lower computational efficiency than FNLF.

V. DISCUSSION

A. Regarding the Training Process of DLFM and DNNs-Based Model

For illustrating this point, we first depict the training processes of DLFM and I-AutoRec (which is a representative DNNs-based recommender proposed in [38]). From Fig. 10, we see that at each layer of a DLFM, desired LF matrices are fully trained with an iterative algorithm, i.e., SGD, to well approximate the input data. Afterward, the achieved LF matrices can produce the "raw" predictions, which will pass through a nonlinear activation function as given in (8) to form the input data for the next layer. Thus, in a DLFM, LF matrices at each layer are trained within this layer to represent its input matrix well; no cross-layer interactions are produced during the training process. In other words, decision parameters are well-trained to fit multiple targets in DLFM. In comparison, as shown in Fig. 11, a DNNs-based model implements back-propagation on each layer to adjust the hidden weights for approximating a unique input, like the user-rating vectors in a u-autoender [38] model. During such a process, cross-layer interactions are made in a backward way for tuning all decision parameters to fit its unique input. From this point of view, a DLFM model works in a significantly different way from a DNNs-based deep model.

B. Effects of the Activation Function (8)

Considering the effects of the activation function (8). From Fig. 1 we see that a DLFM's multiple targets, besides the original HiDS matrix, are generated as follows: 1) the input data pass through nonlinear activation (8) for rescaling them into the given range; 2) LFs in the active layer are iteratively trained to fit the input data which are bent through the activation function already; and 3) the well-trained LFs are adopted to generate the input data for the next layer. During such a process, we see that the original learning objective is gradually bent through the rescaling by the nonlinear activation function (8), thereby improving the generality of a resultant DLFM model.

C. Relationship Between DLFM's Prediction Accuracy and Its Layer Count

As analyzed in the previous section, the learning objective of DLMF keeps being bent as the layer count increases owing to its activation function (8). Initially, this can improve the generality of a resultant DLFM. However, as the layer count grows over the optimal threshold, a resultant DLFM can suffer under-fitting since its learning objective is bent too much. This can be also justified by the compound effects regarding deep structure and regularization in a DLFM model: as shown in Section IV-E, its best layer count decreases as λ increases and vice versa.

From this point of view, DLFM's each layer bent its learning objective and works compatibly with its regularization terms. It can initially improve the generality of a resultant DLFM but also make it suffer under-fitting as the layer count goes over the optimal threshold. Under such circumstances, a resultant DLFM becomes unable to approximate the learning objective well.

Also, the optimal layer count of DLFM is data-dependent, as depicted in Figs. 4–7. This is because the experimental datasets differ from each other vastly in user count, item count, instance count, and data density. DLFM's optimal layer count on a dataset, as its most important hyper-parameter, depends heavily on such data characteristics. Hence, it keeps changing as dataset changes.

D. How to Choose DLFM's Layer Count Wisely?

From Sections IV-C and V-A, we learn that the optimal layer of DLFM is dataset-dependent. To develop a practical application, it is necessary to make the number of DLFM's cascade layers self-adaptive. We can achieve this by doing the following steps when training a DLFM on a specific dataset: 1) splitting the training set into two parts, i.e., growing set and validation set; 2) using the growing set to grow the cascade

layer, and the validation set to estimate the performance. If growing a new layer cannot improve DLFM's performance, the growth terminates and the current layer count is used as the final number of DLFM's cascade layers; and 3) retraining DLFM on the whole training set. To check whether DLFM can self-adaptively decide the optimal number of cascade layers on different datasets by doing the above steps, we conduct some experiments on D1–D4. Concretely, we split 80% of the training set as the growing set and the remain 20% as the validation set. The experimental results show that DLFM finds the optimal number of cascade layers is five on D1, seven on D2, four on D3, and four on D4 for RMSE, respectively, which is consistent with the results obtained in Section IV-C.

E. Why Can DLFM Handle HiDS Data Efficiently Given That It Is Indeed "Deep" Model?

Commonly, an existing deep recommender [37]–[45] adopts a conventional sparse matrix as the input. Since it depends on frequent matrix manipulations, it needs to prefill all entries in R_U with zeroes. We draw this conclusion based on careful investigations into the source codes of existing deep recommenders. In comparison, considering a DLFM, its learning objective and training process are defined on R_K from R, thereby achieving low computational and storage costs. This requires great efforts in handling the parameter update and reconstruction of algorithm codes, but making us achieve the most efficient deep model, i.e., DLFM, which can handle large-scale HiDS matrices with the lowest computational burden than its peers.

VI. Conclusion

HiDS matrices are frequently encountered in RSs, while they contain rich information regarding various useful patterns [18], [19], [54]. To represent an HiDS matrix more precisely, this paper proposes a DLFM, which takes advantage of a generalized deep learning structure, i.e., a deep forest [46]. DLFM's main idea is to construct a deep learning structure by sequentially connecting multiple LF models, each of which acts as a layer of the whole model. Each layer communicates with its subsequent layer with a nonlinear activation function. With such designs, the data representation ability of the resultant model is significantly enhanced. The experimental results on four HiDS matrices from industrial RSs indicate that when compared with state-of-the-art LF models and deep-structured RSs, DLFM can well balance the prediction accuracy and computational efficiency.

In this paper, we choose a homogeneous LFM model as the base model at each layer of DLFM. Can we expect further accuracy gain by adopting heterogeneous LFM models at its different layers? This issue remains open and we plan to investigate into it in the future.

References

[1] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 97–107, Jan. 2014.

[2] R. Lu, X. Jin, S. Zhang, M. Qiu, and X. Wu, "A study on big knowledge and its engineering issues," *IEEE Trans. Knowl. Data Eng.*, to be published.

[3] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, Jun. 2005.

[4] A. Boutet, D. Frey, R. Guerraoui, A. Jégou, and A.-M. Kermarrec, "WHATSUP: A decentralized instant news recommender," in *Proc. 27th Int. Parallel Distrib. Process. Symp. (IPDPS)*, Boston, MA, USA, 2013, pp. 741–752.

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

[6] Y. Cai, H.-F. Leung, Q. Li, H. Min, J. Tang, and J. Li, "Typicality-based collaborative filtering recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 3, pp. 766–779, Mar. 2014.

[7] J. L. D. L. Rosa, N. Hormazabal, S. Aciar, G. A. Lopardo, A. Trias, and M. Montaner, "A negotiation-style recommender based on computational ecology in open negotiation environments," *IEEE Trans. Ind. Electron.*, vol. 58, no. 6, pp. 2073–2085, Jun. 2011.

[8] Y. Koren and R. Bell, "Advances in collaborative filtering," in *Recommender Systems Handbook*. Boston, MA, USA: Springer, 2015, pp. 77–118.

[9] H. Chen and J. Li, "Learning multiple similarities of users and items in recommender systems," in *Proc. IEEE Int. Conf. Data Min.*, New Orleans, LA, USA, 2017, pp. 811–816.

[10] S. Sathe, C. C. Aggarwal, X. Kong, and X. Liu, "Kernel-based feature extraction for collaborative filtering," in *Proc. IEEE Int. Conf. Data Min.*, New Orleans, LA, USA, 2017, pp. 1057–1062.

[11] J. Zhu, P. He, Z. Zheng, and M. R. Lyu, "Online QoS prediction for runtime service adaptation via adaptive matrix factorization," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 10, pp. 2911–2924, Oct. 2017.

[12] A. Liu *et al.*, "Differential private collaborative Web services QoS prediction," in *World Wide Web*. New York, NY, USA: Springer 2018, pp. 1–24. doi: 10.1007/s11280-018-0544-7.

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

[14] X. Luo, Z. Liu, S. Li, M. Shang, and Z. Wang, "A fast non-negative latent factor model based on generalized momentum method," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published.

[15] L. Wu, P. Sun, R. Hong, Y. Ge, and M. Wang, "Collaborative neural social recommendation," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published.

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

[17] J. Castro, J. Lu, G. Zhang, Y. Dong, and L. Martínez, "Opinion dynamics-based group recommender systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 12, pp. 2394–2406, Dec. 2018.

[18] X. Luo, M. Zhou, S. Li, Z. You, Y. Xia, and Q. Zhu, "A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 3, pp. 579–592, Mar. 2016.

[19] X. Luo, M. Zhou, Y. Xia, and Q. Zhu, "An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1273–1284, May 2014.

[20] Y. Shi, M. Larson, and A. Hanjalic, "Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges," *ACM Comput. Surveys*, vol. 47, no. 1, pp. 1–45, 2014.

[21] Q. Liu, S. Wu, D. Wang, Z. Li, and L. Wang, "Context-aware sequential recommendation," in *Proc. IEEE 16th Int. Conf. Data Min. (ICDM)*, Barcelona, Spain, 2016, pp. 1053–1058.

[22] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, "Personalized QoS prediction for Web services via collaborative filtering," in *Proc. 14th IEEE Int. Conf. Web Services*, Salt Lake City, UT, USA, 2007, pp. 439–446.

[23] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Qos-aware Web service recommendation by collaborative filtering," *IEEE Trans. Services Comput.*, vol. 4, no. 2, pp. 140–152, Apr./Jun. 2015.

[24] J. Wu, L. Chen, Y. Feng, Z. Zheng, M. C. Zhou, and Z. Wu, "Predicting quality of service for selection by neighborhood-based collaborative filtering," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 2, pp. 428–439, Mar. 2013.

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

[26] J. Chen, X. Luo, Y. Yuan, M. Shang, Z. Ming, and Z. Xiong, "Performance of latent factor models with extended linear biases," *Knowl. Based Syst.*, vol. 123, pp. 128–136, May 2017.
[27] J. M. Hernández-Lobato, N. Houlsby, and Z. Ghahramani, "Probabilistic matrix factorization with non-random missing data," in *Proc. 31st Int. Conf. Mach. Learn. (JMLR)*, 2014, pp. 1512–1520.
[28] X. Ren, M. Song, E. Haihong, and J. Song, "Context-aware probabilistic matrix factorization modeling for point-of-interest recommendation," *Neurocomputing*, vol. 241, pp. 38–55, Jun. 2017.
[29] K. Yu, S. Zhu, J. Lafferty, and Y. Gong, "Fast nonparametric matrix factorization for large-scale collaborative filtering," in *Proc. 32nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Boston, MA, USA, 2009, pp. 211–218.
[30] H. Wu, Z. Zhang, K. Yue, B. Zhang, J. He, and L. Sun, "Dual-regularized matrix factorization with deep neural networks for recommender systems," *Knowl. Based Syst.*, vol. 145, pp. 46–58, Apr. 2018.
[31] D. Wu, X. Luo, M. Shang, Y. He, G. Wang, and X. Wu, "A data-aware latent factor model for Web service QoS prediction," in *Proc. 23rd Pac.–Asia Conf. Knowl. Disc. Data Min. (PAKDD)*, 2019, pp. 384–399.
[32] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.
[33] J. Huang and B. Kingsbury, "Audio-visual deep learning for noise robust speech recognition," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Vancouver, BC, Canada, 2013, pp. 7596–7599.
[34] J. Lemley, S. Bazrafkan, and P. Corcoran, "Deep learning for consumer devices and services: Pushing the limits for machine learning, artificial intelligence, and computer vision," *IEEE Consum. Electron. Mag.*, vol. 6, no. 2, pp. 48–56, Apr. 2017.
[35] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[36] D. Yu and J. Li, "Recent progresses in deep learning based acoustic models," *IEEE/CAA J. Automatica Sinica*, vol. 4, no. 3, pp. 396–409, Jul. 2017.
[37] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Comput. Surveys*, vol. 52, no. 1, p. 5, 2019.
[38] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "AutoRec: Autoencoders meet collaborative filtering," in *Proc. ACM 24th Int. Conf. World Wide Web*, Florence, Italy, 2015, pp. 111–112.
[39] S. Li, J. Kawale, and Y. Fu, "Deep collaborative filtering via marginalized denoising auto-encoder," in *Proc. ACM 24th ACM Int. Conf. Inf. Knowl. Manag.*, Melbourne, VIC, Australia, 2015, pp. 811–820.
[40] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-N recommender systems," in *Proc. 9th ACM Int. Conf. Web Search Data Min.*, San Francisco, CA, USA, 2016, pp. 153–162.
[41] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. 26th Int. World Wide Web Conf.*, Perth, WA, Australia, 2017, pp. 173–182.
[42] S. Okura, Y. Tagami, S. Ono, and A. Tajima, "Embedding-based news recommendation for millions of users," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2017, pp. 1933–1942.
[43] H.-J. Xue, X.-Y. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems," in *Proc. 26th Int. Joint Conf. Artif. Intell. (IJCAI)*, Melbourne, VIC, Australia, 2017, pp. 3203–3209.
[44] S. Cao, N. Yang, and Z. Liu, "Online news recommender based on stacked auto-encoder," in *Proc. IEEE/ACIS 16th Int. Conf. Comput. Inf. Sci. (ICIS)*, Wuhan, China, 2017, pp. 721–726.
[45] X. Dong, L. Yu, Z. Wu, Y. Sun, L. Yuan, and F. Zhang, "A hybrid collaborative filtering model with deep structure for recommender systems," in *Proc. 31st AAAI Conf. Artif. Intell.*, San Francisco, CA, USA, 2017, pp. 1309–1315.
[46] Z.-H. Zhou and J. Feng, "Deep forest: Towards an alternative to deep neural networks," in *Proc. 26th Int. Joint Conf. Artif. Intell. (IJCAI)*, Melbourne, VIC, Australia, 2017, pp. 3553–3559.
[47] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *Inf. Retrieval*, vol. 4, no. 2, pp. 133–151, 2001.
[48] X. Luo, M. Zhou, Y. Xia, Q. Zhu, A. C. Ammari, and A. Alabdulwahab, "Generating highly accurate predictions for missing QoS data via aggregating nonnegative latent factor models," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 3, pp. 524–537, Mar. 2016.
[49] X. Luo *et al.*, "Incorporation of efficient second-order solvers into latent factor models for accurate prediction of missing QoS data," *IEEE Trans. Cybern.*, vol. 48, no. 4, pp. 1216–1228, Apr. 2018.
[50] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, "GroupLens: Applying collaborative filtering to Usenet news," *Commun. ACM*, vol. 40, no. 3, pp. 77–87, 1997.
[51] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.
[52] D. Wu, X. Luo, G. Wang, M. Shang, Y. Yuan, and H. Yan, "A highly accurate framework for self-labeled semisupervised classification in industrial applications," *IEEE Trans. Ind. Informat.*, vol. 14, no. 3, pp. 909–920, Mar. 2018.
[53] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
[54] W. Luan, G. Liu, C. Jiang, and L. Qi, "Partition-based collaborative tensor factorization for POI recommendation," *IEEE/CAA J. Automatica Sinica*, vol. 4, no. 3, pp. 437–446, 2017.

Di Wu (S'18) received the B.S. degree in applied physics from the Nanjing University of Science and Technology, Jiangsu, China, in 2009, the M.S. degree in optical engineering from Chongqing University, Chongqing, China, in 2012, and the Ph.D. degree in computer science from the University of Chinese Academy of Sciences, Chinese Academy of Sciences, Beijing, China, in 2019.

He is currently an Assistant Professor with the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing. From April 2018 to April 2019, he was a Visiting Scholar with the University of Louisiana at Lafayette, Lafayette, LA, USA. His current research interests include data mining and machine learning.

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. He has published over 100 papers (including over 30 IEEE TRANSACTIONS papers) in the above areas. His current research interests include big data analysis and intelligent control.

Prof. Luo was a recipient of the Hong Kong Scholar Program jointly by the Society of Hong Kong Scholars and China Post-Doctoral Science Foundation in 2014, the Pioneer Hundred Talents Program of Chinese Academy of Sciences in 2016, the Advanced Support of the Pioneer Hundred Talents Program of Chinese Academy of Sciences in 2018, and the Outstanding Associate Editor reward of IEEE ACCESS in 2018. He is currently serving as an Associate Editor for the IEEE/CAA JOURNAL OF AUTOMATICA SINICA, IEEE ACCESS, and *Neurocomputing*. He has also served as the Program Committee Member for over 20 international conferences.

Mingsheng Shang received the Ph.D. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2007.

He joined the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China, in 2015, where he is currently a Professor of Computer Science and Engineering. His current research interests include data mining, complex networks, cloud computing, and their applications.

Yi He received the B.E. degree in transportation engineering from the Harbin Institute of Technology, Harbin, China, in 2013, and the M.S. degree in civil engineering from the University of Louisiana at Lafayette, Lafayette, LA, USA, in 2017, where he is currently pursuing the Ph.D. degree in computer science.

His current research interests include optimization, machine learning, and data mining.

Guoyin Wang (M'98–SM'03) received the B.E. and M.S. degrees in computer software and the Ph.D. degree in computer organization and architecture from Xi'an Jiaotong University, Xi'an, China, in 1992, 1994, and 1996, respectively.

He was with the University of North Texas, Denton, TX, USA, and the University of Regina, Regina, SK, Canada, as a Visiting Scholar, from 1998 to 1999. Since 1996, he has been with the Chongqing University of Posts and Telecommunications, Chongqing, China, where he is currently a Professor and the Ph.D. Supervisor, the Director of the Chongqing Key Laboratory of Computational Intelligence, and the Dean of the College of Computer Science and Technology. His current research interests include data mining, machine learning, rough set, granular computing, and cognitive computing.

Prof. Wang is the President of International Rough Set Society, a Vice-President of the Chinese Association for Artificial Intelligence, and a Council Member of the China Computer Federation.

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

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

Prof. Zhou was a recipient of the Humboldt Research Award for U.S. Senior Scientists from Alexander von Humboldt Foundation, Franklin V. Taylor Memorial Award, and the Norbert Wiener Award from IEEE Systems, Man and Cybernetics Society for which he serves as VP for Conferences and Meetings. He is the Founding Editor of IEEE Press Book Series on Systems Science and Engineering and the Editor-in-Chief of the IEEE/CAA JOURNAL OF AUTOMATICA SINICA. He is a Life Member of the Chinese Association for Science and Technology, USA, and served as its President in 1999. He is a fellow of the International Federation of Automatic Control, the American Association for the Advancement of Science, and the Chinese Association of Automation.