PMLF: Prediction-Sampling-based Multilayer-Structured Latent Factor Analysis

Di Wu[a], *Member, IEEE*, Long Jin[a], *Member, IEEE*, and Xin Luo[a,b], *Senior Member, IEEE*
[a] *Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, 400714, China*
[b] *School of Computer Science and Technology,Dongguan University of Technology, Dongguan, Guangdong 523808,China.*
{wudi, jinlong, luoxin21}@cigit.ac.cn

***Abstract*—A latent factor (LF) model can implement efficient analysis for a high-dimensional and sparse (HiDS) matrix from recommender systems (RSs). However, an LF model's representation learning ability to a targeted HiDS matrix is heavily proportional to its known data density. Unfortunately, an HiDS matrix's known data are limited due to users' activity limitations in RSs. Motivated by this observation, this paper proposes a Prediction-sampling-based Multilayer-structured Latent Factor (PMLF) model. Following the principle of *Deep Forest* [1], PMLF implements a loosely-connected multilayered LF structure, where each layer generates synthetic ratings to enrich the input for the next layer. Such an injection process is carefully monitored through a random sampling process and nonlinear activations to avoid overfitting. Thus, PMLF's representation learning ability to an HiDS matrix is significantly enhanced owing to the carefully injected estimates and its generalized multilayer-structure. Experimental results on four HiDS matrices from industrial RSs indicate that compared with six state-of-the-art LF-based and deep neural networks-based models, PMLF well balances the prediction accuracy and computational efficiency, making it satisfy demands of fast and accurate industrial applications.**

***Keywords*—Deep forest, High-dimensional and sparse data, Missing data estimation, Latent factor model, Deep learning, Generalized multilayer-structure, Recommender system.**

I. Introduction

A user-item rating matrix is commonly adopted to describe users' preferences on items in a recommender system (RS) [2, 3]. Since the number of items can be very large in an industrial RS like Amazon [4], a user can only touch a small part of all the items. As a result, such a matrix is usually high-dimensional and sparse (HiDS) [2, 3]. Hence, how to accurately and efficiently represent an HiDS matrix is a thorny issue. [5-7].

Up to now, a latent factor (LF) model is one of the most popular and successful approaches to address this issue [2, 3, 8]. Given an HiDS matrix, an LF model represents it by training a low-rank approximation based on its known data only [8]. Evidently, an LF model's representation learning ability to a targeted HiDS matrix is heavily proportional to its known data density. Fig. 1 gives an example to illustrate this.

This work is supported in part by the National Natural Science Foundation of China under grants 61702475, 61772493, and 61902370, in part by the Natural Science Foundation of Chongqing (China) under grants cstc2019jcyj-msxmX0578 and cstc2019jcyjjqX0013, in part by the CAS "Light of West China" Program, in part by the Guangdong Province Universities and College Pearl River Scholar Funded Scheme (2019), and in part by the Pioneer Hundred Talents Program of Chinese Academy of Sciences. D. Wu and L. Jin are co-first authors of this paper. (X. Luo is the Corresponding author).

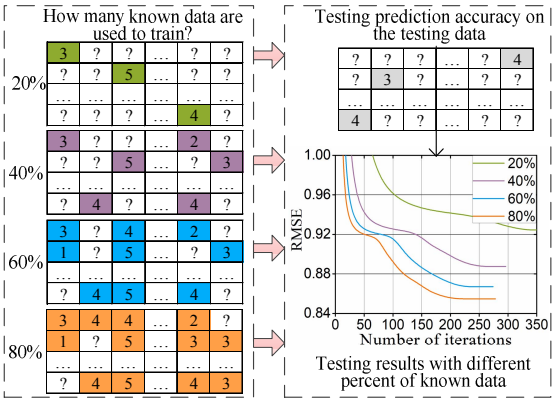

Fig. 1. An example: the known ratings of an HiDS matrix is extremely important for training an LF model.

Example. To illustrate a real case, we select the *MovieLens 1M* [1] as the example HiDS matrix. Its known data are randomly divided into five parts, where one part is the testing data and the other four parts are training data. Under the same conditions, we respectively choose one part (20%), two parts (40%), three parts (60%), and four parts (80%) from training data to train four different LF models first and then test their prediction accuracy (root mean squared error, RMSE) on the testing data. Fig.1 shows that when the percentage of training data increases from 20% to 80%, the achieved RMSE is reduced from 0.9245 to 0.8547, i.e, the prediction accuracy is improved by 7.55%.

Naturally, when we are asked to improve an LF model's representation learning ability, our first thought is to collect more known data for a targeted HiDS matrix. Such a thought, however, is not always reasonable in the area of RS because real rating collection is time-consuming, expensive, and sometimes impossible due to users' activity limitations [2, 9]. Alternatively, can we generate synthetic ratings instead of real ratings to enrich the input for an LF model?

As a matter of fact, synthetic ratings can also be generated by an LF model itself. However, such ratings are deduced from observed ratings in an HiDS matrix, making it risky to adopt them as the input for an LF model. On the other hand, as proposed by Zhou et al. [1, 10], *Deep Forest* can enhance a model gradually with a generalized multilayer-structure, where each layer is an individual model but it employs the features extracted from its preceding layer to enrich the information in its input. From this point of view, it becomes possible to generate synthetic ratings to enrich the input for an LF model

[1]https://grouplens.org/datasets/movielens/1m/

2374-8486/20/$31.00 ©2020 IEEE
DOI 10.1109/ICDM50108.2020.00076

via such an appropriately-designed multilayer-structure.

To do so, this paper proposes a Prediction-sampling-based Multilayer-structured Latent Factor (PMLF) model, which can accurately and efficiently predict the missing data of an HiDS matrix from RSs based on its known ones only. Its main idea is to generate synthetic ratings to enrich the input for an LF model layer by layer via an appropriately-designed multilayer-structure, which can enhance an LF model's representation learning ability. Main contributions of this work include:

a) We first propose to generate synthetic ratings as the input for an LF model to boost its representation learning ability via an appropriately-designed multilayer-structure.
b) We propose a PMLF model that can accurately and efficiently predict the missing data of an HiDS matrix from RSs.
c) We conduct extensive experiments on four HiDS matrices generated by industrial RSs to evaluate the PMLF model.

To the authors' best knowledge, this is the first study that implements a generalized multilayer-structure for an LF model through prediction-sampling. Note that in [11], a deep LF (DLF) model is proposed via aggregating a series of individual LF models. However, a PMLF model differs from a DLF model vastly because:

a) PMLF improves an LF model by enriching its input while DLF by enhancing its generalization ability.
b) PMLF's multilayer-structure is connected by prediction-sampling while DLF just sequentially connects its each layer.
c) PMLF achieves much higher prediction accuracy for the missing data of an HiDS matrix than DLF does.

II. Related Work

Up to now, many sophisticated approaches have been proposed to build an improved LF-based model, including a bias-based one [8], a dual-regularization-based one [12], a probabilistic one [13], a joint recommendation one [14], a neighborhood-and-location integrated one [15], a graph regularized one [16], a confidence-driven one [17], etc. Although these approaches are different from each other in model design, none of them considers improving an LF model by constructing a deep structure. Recently, a deep LF (DLF) model is proposed [11]. It sequentially aggregates a series of individual LF models to enhance their generalization ability. While following the principle of Deep Forest [1], PMLF constructs its multilayer-structure by prediction-sampling, aiming at enriching an LF model's input. As a result, a PMLF model possesses a better representation learning ability to an HiDS matrix than a DLF model does.

Since deep learning has powerful representation learning ability [18, 19], many recent studies have explored deep neural networks (DNNs) to build various RSs [20-24]. Representative models include an autoencoder-based one [24], a denoising autoencoder-based one [25], a stacked autoencoder-based one [26], a hybrid autoencoder-based one [27], a collaborative denoising autoencoder-based one [28], a recurrent neural network-based one [29], a neural collaborative filtering-based one [20], a hybrid deep structure-based one [30], a deep matrix factorization with neural network-based one [21], a multitask learning-oriented one [31], a neural factorization-based one [32], an attentional factorization-based one [33], a deep cooperative neural network model [34], and a convolutional matrix factorization model [35].

Note that these DNNs-based models have the limit caused by the defects of DNNs [1], i.e., they take complete data rather than known data of an HiDS matrix as input, resulting in extremely high computational cost [11]. For example, Epinion dataset (adopted in this paper) has a rating density of 0.02% only, where 13,668,321 ratings scatter in 755,760 rows and 120,492 columns [36]. If we take complete data as input, more than 91 billion entries need to be processed, which is greatly difficult in real applications. While a PMLF model constructs its multilayer-structure based on an LF model rather than the DNNs. It trains only on the known data of an HiDS matrix, thereby achieving highly computational efficiency.

III. Preliminaries

A. Symbols and Notations

TABLE I. Symbols and Notations.

Symbol	Explanation
U	Targeted user set.
u	A user of U.
I	Targeted item set.
i	An item of I.
Y	Targeted user-item rating matrix with $\|U\|$ rows and $\|I\|$ *columns*, which is an HiDS matrix.
$y_{u,i}$	Y's element at uth row and ith column denoting the rating experienced by a user $u \in U$ on an item $i \in I$.
S_K	Known ratings set of Y.
S_U	Unknown ratings set of Y.
d	Latent factor dimension.
W	Latent factor matrix of a latent factor model for users with $\|U\|$ rows and d columns.
$w_{u,\cdot}$	uth row-vector of W.
Z	Latent factor matrix of a latent factor model for items with $\|I\|$ rows and d columns.
$z_{i,\cdot}$	ith row-vector of Z.
$\hat{Y}$	Y's rank-d approximation built on S_K with $d \ll \min(\|U\|, \|I\|)$.
$\hat{y}_{u,i}$	$\hat{Y}$'s element at uth row and ith column denoting prediction for $y_{u,i}$.
Ω	A $\|U\| \times \|I\|$ binary index matrix, in which $\Omega_{u,i}$ is its element at uth row and ith column.
N	The maximun number of layers of PMLF.
n	Denoting the nth layer of PMLF, $n \in \{1, 2, \ldots, N\}$.
W_n	Latent factor matrix of PMLF for users at nth layer with $\|U\|$ rows and d columns.
$w^n_{u,\cdot}$	uth row-vector of W_n.
Z_n	Latent factor matrix of PMLF for items at nth layer with $\|I\|$ rows and d columns.
$z^n_{i,\cdot}$	ith row-vector of Z_n.
$\hat{Y}_n$	Y's rank-d approximation achieved by PMLF at nth layer.
$\hat{y}^n_{u,i}$	$\hat{Y}_n$'s element at uth row and ith column denoting prediction for $y_{u,i}$.
$\tilde{y}^n_{u,i}$	The output of the nonlinear activation function as $\hat{y}^n_{u,i}$ is the input.
$\bar{Y}_n$	An HiDS matrix consists of $\bar{S}_K^{n-1}$ and some randomly selected ratings from $\hat{Y}_n$. It is the input of n+1th layer of PMLF.
$\bar{y}^n_{u,i}$	$\bar{Y}_n$'s element at uth row and ith column.
$\bar{S}^n_K$	Known ratings set of $\bar{Y}_n$.
α	The ratio of selecting prediction ratings from $\hat{Y}_n$.
λ	The regularization parameter of l_2-norm-based regularization.
$\varepsilon(W, Z)$	Objective function with respect to W and Z.
$\varepsilon^n_{u,i}$	Instant loss on a single rating at nth layer.
T_{max}	Maximum training round count for each layer of PMLF.
η	Learning rate.

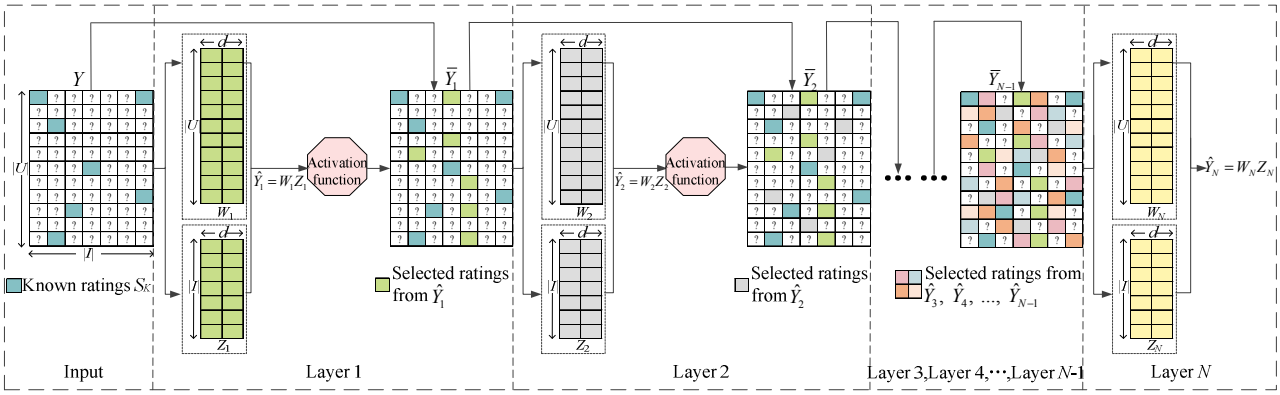

Fig. 2. The structure and processing flow of a PMLF model.

B. Latent Factor Model

***Definition* 1 (*An HiDS matrix*).** Given a user set U and an item set I, Y is a $|U|\times|I|$ matrix in which each element $y_{u,i}$ denotes user u's ($u\in U$) preference on item i ($i\in I$). S_K and S_U denote known and unknown rating sets of Y respectively. Y is an HiDS matrix with U and I being large and $|S_K|\ll|S_U|$.

***Definition* 2 (*A latent factor model*).** Given an HiDS matrix Y and the latent factor dimension d, a latent factor model is to train two corresponding latent factor matrices $W^{|U|\times d}$ and $Z^{|I|\times d}$ to achieve Y's rank-d approximation $\hat{Y}$ by minimizing the errors between Y and $\hat{Y}$ on S_K, where $\hat{Y}$ is given by $\hat{Y}=WZ^{\mathrm{T}}$ and its each element $\hat{y}_{u,i}$ is the prediction for Y's each corresponding element $y_{u,i}$.

From definition 2, an objective function that minimizes the errors between Y and $\hat{Y}$ is highly important. A Euclidean distance-based objective is a common choice [8]:

$$\arg\min_{W,Z}\varepsilon(W,Z)=\frac{1}{2}\left\|\Omega\odot\left(Y-\hat{Y}\right)\right\|_F^2=\frac{1}{2}\left\|\Omega\odot\left(Y-WZ^{\mathrm{T}}\right)\right\|_F^2, \quad (1)$$

where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix, $\odot$ denotes the Hadamard product (the component-wise multiplication), and Ω is a $|U|\times|I|$ binary index matrix given by

$$\Omega_{u,i}=\begin{cases}1 & \text{if } y_{u,i} \text{ is observed}\\ 0 & \text{otherwise}\end{cases}. \quad (2)$$

To avoid overfitting on S_K, regularization term like L_2-norm needs to be incorporated into (1) [2]:

$$\arg\min_{W,Z}\varepsilon(W,Z)=\frac{1}{2}\left\|\Omega\odot\left(Y-WZ^{\mathrm{T}}\right)\right\|_F^2+\frac{\lambda}{2}\left(\|W\|_F^2+\|Z\|_F^2\right), \quad (3)$$

where (3) can be minimized by an optimization algorithm, such as stochastic gradient descent (SGD).

IV. Proposed PMLF Model

A. Structure of PMLF

Following the generalized multilayer-structure of a Deep Forest proposed by Zhou et al. [1] and the principle that more known data of an HiDS matrix can improve an LF model's representation learning ability as illustrated in Fig.1, we design the PMLF model, as shown in Fig.2. PMLF sequentially connects N LF models and N-1 nonlinear activation functions to construct a prediction-sampling-based multilayer-structure. A PMLF model works as follows:

a) Input: inputting S_K into layer 1 as the initial inputs;
b) Layer 1: training LF matrices W_1 and Z_1 based on S_K only to obtain $\hat{Y}_1$, the training rules are analyzed in *Section IV.B*;
c) Layer 1: randomly selecting some prediction ratings from $\hat{Y}_1$ (the prediction data for the unknown data of Y) to input into a nonlinear activation function;
d) Layer 1: combining the outputs of the activation function and Y to form a new HiDS matrix $\bar{Y}_1$, which is taken as the input of layer 2;
e) Layer n: training LF matrices W_n and Z_n based on the known ratings of $\bar{Y}_{n-1}$ only to obtain $\hat{Y}_n$, the training rules are analyzed in *Section IV.C*, where $n\in\{2, \ldots, N\}$.
f) Layer n: randomly selecting some prediction ratings from $\hat{Y}_n$ (the prediction data for the unknown data of Y) to input into a nonlinear activation function, where $n\in\{2, \ldots, N\}$;
g) Layer n: combining the outputs of the activation function and $\bar{Y}_{n-1}$ to form a new HiDS matrix $\bar{Y}_n$, which is taken as the input of the next layer, where $n\in\{2, \ldots, N\}$;
h) Output: repeating steps e–g until last layer N to obtain $\hat{Y}_N$, then output $\hat{Y}_N$.

B. Training Layer 1 of PMLF with SGD

To efficiently train layer 1 of PMLF, we expand (3) into the following single LF-dependent form:

$$\arg\min_{W_1,Z_1}\varepsilon(W_1,Z_1)=\frac{1}{2}\sum_{y_{u,i}\in S_K}\left(y_{u,i}-\sum_{\tau=1}^{d}w_{u,\tau}^1 z_{i,\tau}^1\right)^2+\frac{\lambda}{2}\left(\|W_1\|_F^2+\|Z_1\|_F^2\right). \quad (4)$$

Then, we consider the instant of (4) on a single rating $\forall y_{u,i}\in S_K$:

$$\varepsilon_{u,i}^1=\frac{1}{2}\left(y_{u,i}-\sum_{\tau=1}^{d}w_{u,\tau}^1 z_{i,\tau}^1\right)^2+\frac{\lambda}{2}\left(\sum_{\tau=1}^{d}\left(w_{u,\tau}^1\right)^2+\sum_{\tau=1}^{d}\left(z_{i,\tau}^1\right)^2\right). \quad (5)$$

Next, we employ SGD to optimize (5). To do so, we move the LFs involved in (5) along the opposite direction against the stochastic gradient of (5) with respect to each single LF:

$$\text{on } y_{u,i}, \forall\tau\in\{1,2,\ldots,d\}:\begin{cases}w_{u,\tau}^1\leftarrow w_{u,\tau}^1-\eta\dfrac{\partial\varepsilon_{u,i}^1}{\partial w_{u,\tau}^1}\\ z_{i,\tau}^1\leftarrow z_{i,\tau}^1-\eta\dfrac{\partial\varepsilon_{u,i}^1}{\partial z_{i,\tau}^1}\end{cases}. \quad (6)$$

Further, we extend (6) to obtain the training rules of layer 1 of a PMLF model as follows:

$$\text{on } y_{u,i}, \forall\tau\in\{1,2,\ldots,d\}:\begin{cases}w_{u,\tau}^1\leftarrow w_{u,\tau}^1+\eta z_{i,\tau}^1\left(y_{u,i}-\sum_{\tau=1}^{d}w_{u,\tau}^1 z_{i,\tau}^1\right)-\eta\lambda w_{u,\tau}^1,\\ z_{i,\tau}^1\leftarrow z_{i,\tau}^1+\eta w_{u,\tau}^1\left(y_{u,i}-\sum_{\tau=1}^{d}w_{u,\tau}^1 z_{i,\tau}^1\right)-\eta\lambda z_{i,\tau}^1\end{cases}. \quad (7)$$

After each known rating in S_K is trained with (7), W_1 and Z_1 are extracted and can be used to predict the missing data of Y by computing $\hat{Y}_1$ as follows:

$$\hat{Y}_1 = W_1 Z_1^{\mathrm{T}}. \tag{8}$$

Next, we randomly select some prediction data from $\hat{Y}_1$ to input into a nonlinear activation function as follows:

$$\forall \hat{y}^1_{u,i} \in \hat{Y}_1 : \tilde{y}^1_{u,i} = \begin{cases} y_{\min} + 1/\left(1+e^{-\hat{y}^1_{u,i}}\right) & if\ \hat{y}^1_{u,i} < y_{\min} \\ y_{\max} / \left(1+e^{-\hat{y}^1_{u,i}}\right) & if\ \hat{y}^1_{u,i} > y_{\max} \\ \hat{y}^1_{u,i} & otherswise. \end{cases} \tag{9}$$

Note that if $\hat{y}^1_{u,i} < y_{\min}$ or $\hat{y}^1_{u,i} > y_{\max}$, these prediction data are evidently not correct. Thus, (9) has the physical meaning that it can reset the extremely unreasonable prediction ratings of $\hat{Y}_1$. Finally, the outputs of (9) plus the S_K forms a new HiDS matrix $\bar{Y}_1$, which is the input for the layer 2 of a PMLF model.

C. Training Layer n of PMLF with SGD

To explain how to train the other multilayered layers of PMLF, we give the training process of layer n as a general description, where $n \in \{2,\ldots, N\}$. Note that the synthetic HiDS matrix $\bar{Y}_{n-1}$, which comes from the layer n-1, is the input for training the layer n. Similar to (4), training the layer n is also formulated as the following single LF-dependent form:

$$\arg\min_{W_n,\, Z_n} \varepsilon(W_n,\, Z_n) = \frac{1}{2} \sum_{\bar{y}^{n-1}_{u,i} \in \bar{S}^{n-1}_K} \left(\bar{y}^{n-1}_{u,i} - \sum_{\tau=1}^{d} w^n_{u,\tau} z^n_{i,\tau} \right)^2 + \frac{\lambda}{2} \left(\|W_n\|_F^2 + \|Z_n\|_F^2 \right), \tag{10}$$

On a single rating $\forall \bar{y}^{n-1}_{u,i} \in \bar{S}^{n-1}_K$, the instant of (10) is:

$$\varepsilon^n_{u,i} = \frac{1}{2} \left(\bar{y}^{n-1}_{u,i} - \sum_{\tau=1}^{d} w^n_{u,\tau} z^n_{i,\tau} \right)^2 + \frac{\lambda}{2} \left(\sum_{\tau=1}^{d} \left(w^n_{u,\tau} \right)^2 + \sum_{\tau=1}^{d} \left(z^n_{i,\tau} \right)^2 \right). \tag{11}$$

To solve (11) with SGD, we obtain the training rules of layer n as follows:

$$\text{on } \bar{y}^{n-1}_{u,i}, \forall \tau \in \{1,2,\ldots,d\}: \quad \begin{cases} w^n_{u,\tau} \leftarrow w^n_{u,\tau} + \eta z^n_{i,\tau} \left(\bar{y}^{n-1}_{u,i} - \sum_{\tau=1}^{d} w^n_{u,\tau} z^n_{i,\tau} \right) - \eta \lambda w^n_{u,\tau} \\ z^n_{i,\tau} \leftarrow z^n_{i,\tau} + \eta w^n_{u,\tau} \left(\bar{y}^{n-1}_{u,i} - \sum_{\tau=1}^{d} w^n_{u,\tau} z^n_{i,\tau} \right) - \eta \lambda z^n_{i,\tau} \end{cases}. \tag{12}$$

Based on (12), W_n and Z_n can be obtained. Then, we employ them to predict the missing ratings of $\bar{Y}_{n-1}$ by $\hat{Y}_n = W_n Z_n^{\mathrm{T}}$. After that, we randomly select some prediction data from $\hat{Y}_n$ to input into a nonlinear activation function (13), aiming at resetting the extremely unreasonable prediction ratings of $\hat{Y}_n$:

$$\forall \hat{y}^n_{u,i} \in \hat{Y}_n : \tilde{y}^n_{u,i} = \begin{cases} y_{\min} + 1/\left(1+e^{-\hat{y}^n_{u,i}}\right) & if\ \hat{y}^n_{u,i} < y_{\min}, \\ y_{\max} / \left(1+e^{-\hat{y}^n_{u,i}}\right) & if\ \hat{y}^n_{u,i} > y_{\max}, \\ \hat{y}^n_{u,i} & otherswise. \end{cases} \tag{13}$$

Then, we combine the outputs of (13) and the $\bar{S}^{n-1}_K$ to form $\bar{Y}_n$, which is the input for layer n+1.

The above operations are a general training process for any layer of PMLF except for layer 1. Following these operations, we sequentially train the multilayered layers of PMLF layer by layer until to the last layer N. Finally, we obtain the $\hat{Y}_N$ by $\hat{Y}_N = W_N Z_N$, which is the final predictions of PMLF for the missing ratings of Y.

D. Method of Selecting Prediction Ratings from $\hat{Y}_n$

How to select prediction ratings from $\hat{Y}_n$ at PMLF's each layer n is crucial for PMLF. The specific method is a) randomly selecting an unknown entry between two known entries in each row of $\bar{Y}_{n-1}$ as the *blank entry*, b) selecting the entry that is the prediction for the *blank entry* from $\hat{Y}_n$ to input into the activation function, and c) combining the outputs of the activation function and $\bar{Y}_{n-1}$ to form $\bar{Y}_n$. To illustrate this method, an example is given in Fig. 3.

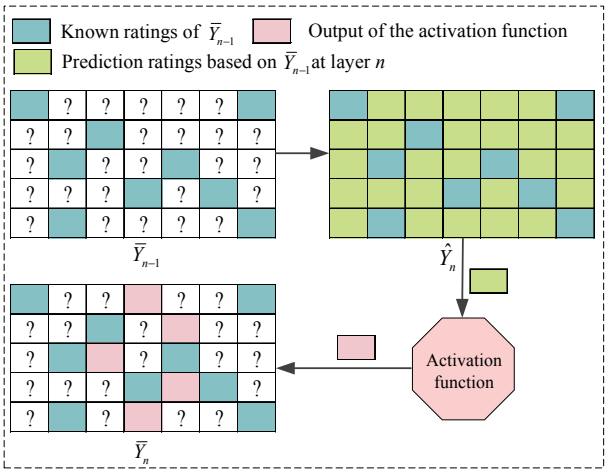

Fig. 3. An example of how to select prediction ratings from $\hat{Y}_n$ at PMLF's each layer n.

E. Algorithm Design and Analysis

Based on the above analyses, we design *Algorithm PMLF*. For PMLF with only one layer, it actually degenerates an original LF model. Hence, layer 1 of PMLF has the computational complexity of $O(T_{max} \times |S_K| \times d)$ [11]. For layer 2, some additional prediction ratings selected from $\hat{Y}_1$ are used as the input. Supposing the selecting ratio from $\hat{Y}_1$ is α, where $\alpha = (|\bar{S}^2_K| - |S_K|)/|S_K|$. Then, the computational complexity of layer 2 is $O(T_{max} \times |S_K| \times d \times (1+\alpha))$. Similarly for layer n, supposing the selecting ratio from $\hat{Y}_{n-1}$ is also α, where $\alpha = (|\bar{S}^n_K| - |\bar{S}^{n-1}_K|)/|\bar{S}^{n-1}_K|$, the computational complexity of layer n is $O(T_{max} \times |S_K| \times d \times (1+\alpha)^{n-1})$. Finally, supposing PMLF consists of N layers, its computational complexity is $O(T_{max} \times |S_K| \times d \times (1+\alpha)^N/\alpha)$, which has higher computational complexity than an LF model.

Evidently, the extra computing burden of PMLF is caused by its multilayer-structure and decided by its maximum number of layers and ratio of selecting prediction ratings from $\hat{Y}_n$. However, the computational complexity of PMLF is acceptable in practice because: a) an LF model, which is the basic component of PMLF, is highly efficient in addressing an HiDS matrix [2, 37-38]; b) PMLF can be parallelized to train because it belongs to the vanilla SGD-based matrix factorization [39-40], which can greatly reduce its computational cost. More discussions regarding parallel PMLF are provided in *Section VI.B*.

ALGORITHM PMLF

Steps	**Input:** S_K **Output:** $\hat{Y}_N$
1	initializing d, λ, η, T_{max}; W_1, Z_1
2	**while** $t \leq T_{max}$ && not converge
3	**for** $\forall y_{u,i} \in S_K$
4	**for** τ=1 to d
5	computing $w^1_{u,k}$ according to (7)
6	computing $z^1_{i,k}$ according to (7)
7	**end for**
8	**end for**
9	t=t+1
10	**end while**
11	randomly selecting some prediction ratings from $\hat{Y}_1$, $\hat{Y}_1 = W_1 Z_1^{\mathrm{T}}$
12	inputting the selected ratings into (9)
13	combining the outputs of (9) and S_K to form $\bar{Y}_1$
14	**for** n=2 to N
15	initializing W_n, and Z_n
16	**while** $t \leq T_{max}$ && not converge
17	**for** $\forall \bar{y}^{n-1}_{u,i} \in \bar{S}^{n-1}_K$
18	**for** τ=1 to d
19	computing $w^n_{u,\tau}$ according to (12)
20	computing $z^n_{i,\tau}$ according to (12)
21	**end for**
22	**end for**
23	t=t+1
24	**end while**
25	randomly selecting some prediction ratings from $\hat{Y}_n$, $\hat{Y}_n = W_n Z_n^{\mathrm{T}}$
26	inputting the selected ratings into (13)
27	combining the outputs of (13) and $\bar{S}^{n-1}_K$ to form $\bar{Y}_n$
28	**end for**

V. EXPERIMENTS

A. General Settings

Datasets. Four benchmark datasets are real HiDS datasets generated by industrial applications [36, 41]. Table II summarizes their properties. Dating is collected by an online dating website LibimSeTi [42], Epinion is collected by Trustlet website [36], Flixter is collected by the Flixter website [43], and MovieLens is collected by the MovieLens system [44].

TABLE II. PROPERTIES OF ALL THE DATASETS.

No.	Name	$\|U\|$	$\|I\|$	$\|Y_K\|$	Density*
D1	Dating	135,359	168,791	17,359,346	0.08%
D2	Epinion	755,760	120,492	13,668,321	0.02%
D3	Flixter	147,612	48,794	8,196,077	0.11%
D4	MovieLens	6040	3706	1,000,209	4.47%

*Density denotes the percentage of the known ratings in the dataset.

Evaluation Metrics. Rating prediction is a common recommendation task [19]. It aims to predict the missing data of a given HiDS user-item matrix. To evaluate rating prediction accuracy, mean absolute error (MAE) and root mean squared error (RMSE) are widely adopted as the evaluation metrics [19]. They are calculated as follows:

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

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

where Γ denotes the testing set and $|\cdot|_{abs}$ denotes the absolute value of a given number. Besides, we also evaluate ranking prediction performance. Normalized discounted cumulative gain (NDCG) is usually adopted as the evaluation metrics [19, 45, 46]. NDCG is calculated as follows:

$$DCG@K(u,\Gamma) = \sum_{k=1}^{K} \frac{2^{y_{u,k}} - 1}{\log_2(i+1)},$$

$$DCG@K(u,\hat{Y}) = \sum_{k=1}^{K} \frac{2^{\hat{y}_{u,k}} - 1}{\log_2(i+1)},$$

$$NDCG@K = \frac{1}{|U|} \sum_{u\in U} \frac{DCG@K(u,\hat{Y})}{DCG@K(u,\Gamma)},$$

where $y_{u,k}$ denotes the kth known rating regarding a user u in Γ in descending order, $\hat{y}_{u,k}$ denotes the kth prediction rating corresponding to each item in Γ in descending order, and K is a cutoff parameter determining how many items are considered in the ranked list. Finally, to evaluate computational efficiency, we test the CPU running time.

Baselines. We compare our proposed PMLF model with six related state-of-the-art models, including three LF-based models (BLF [8], FNLF [47], and DLF [11]) and three DNNs-based models (AutoRec [24], NRT [31], and DCCR [27]). Table III gives brief descriptions of all the involved models.

TABLE III. DESCRIPTIONS OF ALL THE INVOLVED MODELS.

Model	Description
BLF	The basic LF model proposed in 2009 [8] and has been extensively used in RSs.
FNLF	A fast non-negative LF model based on generalized momentum proposed in 2018 [47]. It improves a common non-negative LF model by considering the momentum effects.
DLF	A deep LF model proposed in 2019 [11]. Its deep structure plays a role like regularization.
AutoRec	A DNNs-based model proposed in 2015 [24]. It is the representative model in DNNs-based RSs.
NRT	A DNNs-based model proposed in 2017 [31]. It is a multitask learning framework developed by combining multilayer perceptron and recurrent neural networks.
DCCR	A DNNs-based model proposed in 2019 [27]. It improves AutoRec by using two different neural networks.
PMLF	The proposed prediction-sampling-based multilayer-structured latent factor model in this paper.

B. Experimental Designs

For each dataset, its 80% know ratings are used as the training dataset and the remaining 20% ones are used as the testing dataset. The training process of a tested model terminates if the number of consumed iterations reaches a preset threshold (500) or the error difference between two consecutive iterations is smaller than 10^{-6}. All the experiments are run on a computer server that has a 2.1 GHz E52620 CPU with 32 cores

and a 256 GB RAM. In the next experiments, we aim at answering the following research questions (RQs):

RQ. 1. Does the proposed PMLF model outperform state-of-the-art LF-based and DNNs-based models?

RQ. 2. Is PMLF's prediction-sampling-based multilayer-structure helpful for improving an LF model's prediction accuracy?

RQ. 3. How does the latent factor dimension *d* influence the performance of a PMLF model?

C. Performance Comparison (RQ.1)

We adopt the following model settings: a) setting latent factor dimension *d*=10 for the three LF-based models (BLF, FNLF, and DLF) and the proposed PMLF model, b) for the three DNNs-based models (AutoRec, NRT, and DCCR), setting their number of layers and dimension of hidden units according to their original papers, c) on rating prediction, tuning the other hyper-parameters of all the models on one fold of each dataset to achieve the best performance of each model and then adopting the same values on the remaining four folds, and d) on ranking prediction, the other hyper-parameters of all the models are set same as that in rating prediction.

1) Comparison of rating prediction accuracy

Table IV presents the comparison results. To better understand these comparison results, we conduct statistical analysis. First, the win/loss counts of PMLF versus other models are summarized in the second-to-last row of Table IV. Second, we conduct the Friedman test [48] because it is effective in validating the performance of multiple models on multiple datasets. The Friedman statistical result is recorded in the last row of Table IV, where it accepts the hypothesis that these comparison models have significant differences with a significance level of 0.05. From Table IV, we find that: a) PMLF evidently wins the three LF-based models on RMSE/MAE comparison, b) compared with the three DNNs-based models, PMLF achieves a lower RMSE/MAE in most cases, and c) PMLF achieves the lowest F-rank value. Therefore, these findings verify that PMLF achieves the highest rating prediction accuracy among all the models.

In addition, to check whether PMLF has significantly better prediction accuracy than each single model, we also conduct the Wilcoxon signed-ranks test [48] on the comparison results of Table IV. Wilcoxon signed-ranks test has three indicators—*R*+, *R*-, and *p*-value. The larger *R*+ value denotes a higher prediction accuracy and the *p*-value denotes the significance level. Table V records the results. First, we see that PMLF has a significantly higher rating prediction accuracy than the three LF-based models. Second, when comparing with the three DNNs-based models, the hypothesis is not accepted. One reason is that their hidden units (which is like the latent factor dimension *d* of an LF-based model, AutoRec=500, NRT=400, and DCCR=500) is much larger than PMLF's *d* (set as 10). However, PMLF still achieves a much larger *R*+ than the three DNNs-based models do, which means that PMLF has slightly higher rating prediction accuracy than them.

2) Comparison of ranking prediction accuracy

Table VI records the detailed comparison results, where the statistical analyses of win/loss counts and Friedman test are also presented. Besides, we also conduct the Wilcoxon signed-ranks test on the comparison results of Table VI and the results are recorded in Table VII. From Tables VI and VII, we find that: a) PMLF has the highest ranking prediction accuracy among all the models because it has the highest F-rank value, b) PMLF has significantly higher ranking prediction accuracy than BLF, DLF, and AutoRec, c) PMLF has a slightly better ranking prediction accuracy than FNLF and DCCR, and d) PMLF and NRT have the comparable ranking prediction accuracy.

TABLE IV. THE COMPARISON RESULTS ON RATING PREDICTION ACCURACY, INCLUDING WIN/LOSS COUNTS AND FRIEDMAN TEST, WHERE • INDICATES PMLF HAS A LOWER RMSE/MAE THAN THE COMPARISON MODELS.

Dataset	Metric	BLF	FNLF	DLF	AutoRec	NRT	DCCR	PMLF
D1	RMSE	2.0616•	2.1028•	2.0502•	2.2863•	2.3033•	2.2712•	2.0271
	MAE	1.4269•	1.4964•	1.4258•	1.8429•	1.7948•	1.8358•	1.4093
D2	RMSE	1.2679•	1.2566•	1.2579•	1.2297	1.2268	1.2298	1.2416
	MAE	0.5960•	0.5801	0.5958•	0.5943•	0.5899•	0.5960•	0.5880
D3	RMSE	1.0706•	1.0895•	1.0712•	1.0603	1.0609	1.0630	1.0652
	MAE	0.7431•	0.7555•	0.7465•	0.7325	0.7332	0.7331	0.7422
D4	RMSE	0.9168•	0.9278•	0.9160•	0.9187•	0.9176•	0.9169•	0.9138
	MAE	0.7238•	0.7271•	0.7234•	0.7285•	0.7273•	0.7265•	0.7212
Statistic	Win/Loss	8/0	7/1	8/0	5/3	5/3	5/3	—
	F-rank*	4.44	5.00	3.88	4.25	4.00	4.19	**2.25**

* A lower F-rank value indicates a higher rating prediction accuracy.

TABLE V. RESULTS OF THE WILCOXON SIGNED-RANKS TEST ON RMSE/MAE OF TABLES IV.

Comparison	*R*+	*R*-	*p*-value*
PMLF vs. BLF	36	0	**0.0039**
PMLF vs. FNLF	34	2	**0.0117**
PMLF vs. DLF	36	0	**0.0039**
PMLF vs. AutoRec	23.5	12.5	0.2422
PMLF vs. NRT	22	14	0.3203
PMLF vs. DCCR	24	12	0.2305

* The accepted hypotheses with a significance level of 0.05 are highlighted.

TABLE VI. THE COMPARISON RESULTS ON RANKING PREDICTION ACCURACY, INCLUDING WIN/LOSS COUNTS AND FRIEDMAN TEST, WHERE • INDICATES PMLF HAS A HIGHER NDCG THAN THE COMPARISON MODELS.

Dataset	Metric	BLF	FNLF	DLF	AutoRec	NRT	DCCR	PMLF
D1	NDCG@5	0.9161•	0.9212	0.9162•	0.9163•	0.9227	0.9166•	0.9172
	NDCG@10	0.8995•	0.9033	0.9005	0.8987•	0.9030	0.8994•	0.9002
D2	NDCG@5	0.9481	0.9465•	0.9493	0.9438•	0.9469•	0.9453•	0.9474
	NDCG@10	0.9601	0.9589•	0.9609	0.9560•	0.9589•	0.9549•	0.9596
D3	NDCG@5	0.8553•	0.8583•	0.8536•	0.8612	0.8625	0.8628	0.8605
	NDCG@10	0.8713•	0.8747•	0.8697•	0.8775	0.8785	0.8798	0.8767
D4	NDCG@5	0.7512•	0.7457•	0.7517•	0.7566•	0.7503•	0.7536•	0.7617
	NDCG@10	0.7480•	0.7452•	0.7496•	0.7484•	0.7476•	0.7493•	0.7566
Statistic	Win/Loss	6/2	6/2	5/3	6/2	4/4	6/2	—
	F-rank*	3.25	3.44	4.13	3.38	4.56	4.13	**5.13**

* A higher F-rank value indicates a higher ranking prediction accuracy.

TABLE VII. RESULTS OF THE WILCOXON SIGNED-RANKS TEST ON NDCG OF TABLE VI.

Comparison	*R*+	*R*-	*p*-value*
PMLF vs. BLF	32.5	3.5	**0.0234**
PMLF vs. FNLF	25	11	0.1914
PMLF vs. DLF	28	8	**0.0938**
PMLF vs. AutoRec	33	3	**0.0195**
PMLF vs. NRT	18	18	0.5273
PMLF vs. DCCR	27	9	0.1250

* The accepted hypotheses with a significance level of 0.1 are highlighted.

3) Comparison of computational efficiency

Fig. 4 records the CPU running time of each model tested on all the datasets. First, we see that PMLF costs more CPU running time than the three LF-based models. One reason is that PMLF needs to train some extra synthetic ratings due to its prediction-sampling-based multilayer-structure. Second, we find that PMLF costs much less CPU running time than the three DNNs-based models, especially on the large-scale datasets, like D1–D3. The main reason is that PMLF only takes the known ratings of an HiDS matrix as input while the three DNNs-based models take the complete data to do that.

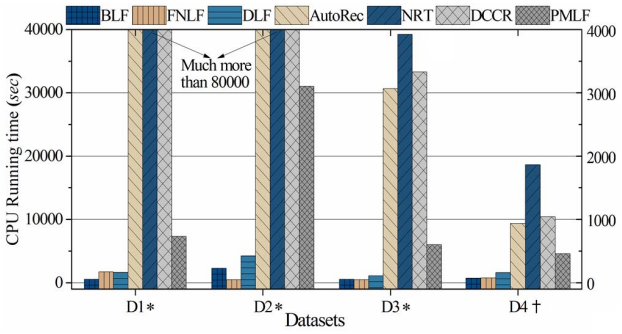

Fig. 4. The comparison CPU running time of involved models, where ∗ denotes that D1–D3 is left tick label and † denotes that D4 is right tick label.

D. Influence of Multilayer-structure in PMLF (RQ.2)

In this set of experiments, we test PMLF's rating prediction accuracy as its layer count increases. The hyper-parameters are set as λ=0.01, η=0.001, and d=10, uniformly. Figs. 5 and 6 present the training process of PMLF at different layers, where we have the following important findings:

a) PMLF's prediction-sampling-based multilayer-structure does not affect its each basic LF model's convergence. At each layer, RMSE/MAE keeps decreasing with more training rounds until reaching convergence,
b) PMLF's prediction-sampling-based multilayer-structure helps improve an LF model's prediction accuracy. As can be seen, PMLF has a lower RMSE/MAE as the layer count increases,
c) Note that, with only one layer, PMLF degenerates into an original LF model, i.e., BLF. Comparing with BLF, the RMSE and MAE reduced by PMLF on D1–4 are 2.63% and 2.38%, 2.70% and 2.20%, 1.64% and 0.99%, and 0.86% and 0.59%, respectively, and
d)PMLF's prediction accuracy is not always higher with a deeper layer. For example, the MAE on D4 decreases first and then increases as the layer becomes deeper. One possible reason is that when the layer count grows over its optimal threshold, some unreliable prediction-sampling ratings are input into the next layer.

E. Influence of Latent Factor Dimension d (RQ.3)

This set of experiments increases d from 10 to 320 for PMLF. The hyper-parameters are set as λ=0.01 and η=0.001, uniformly. Figs. 7 and 8 record the results, where we find that the larger d makes PMLF achieve a higher rating prediction accuracy, which means that PMLF has a better representation learning ability with a larger d. Such a finding is consistent with the prior studies [8, 37]. Note that in *Section V.C*, we only set d=10 for PMLF to compare with the DNNs-based models. Based on the analyses of this section, we know that PMLF with a larger d (such as 40) actually can achieve a better comparison results when comparing with AutoRec, NRT, and DCCR.

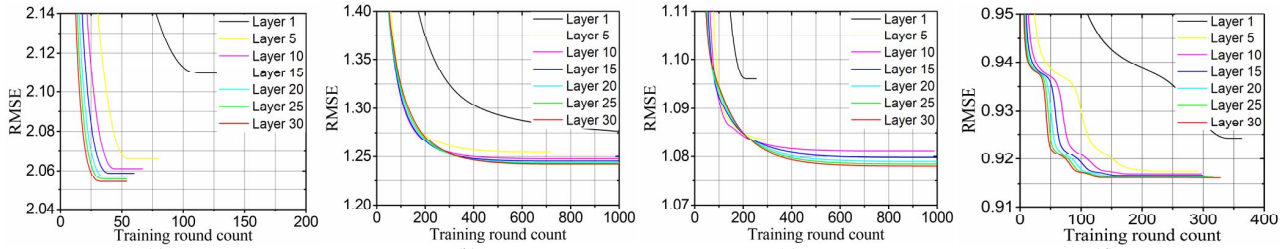

Fig. 5. The training process of PMLF regarding RMSE at different layers on all the datasets.

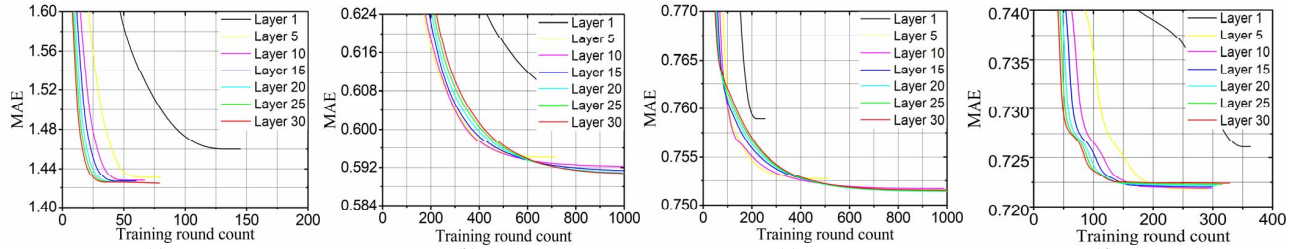

Fig. 6. The training process of PMLF regarding MAE at different layers on all the datasets.

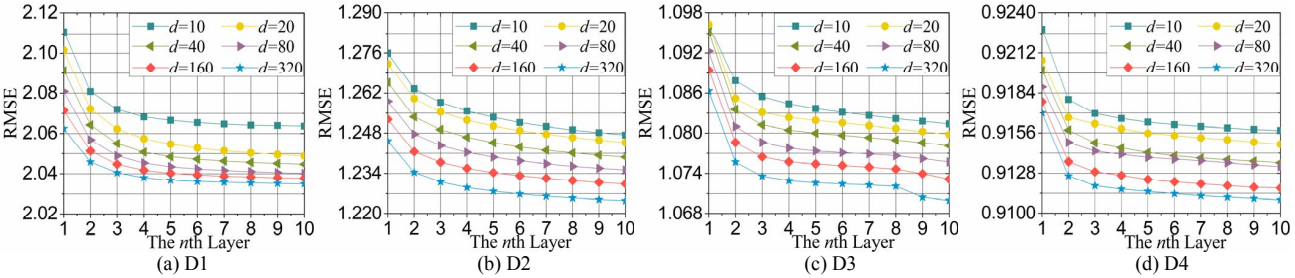

Fig. 7. The lowest RMSE of PMLF at different layers as d increases on all the datasets.

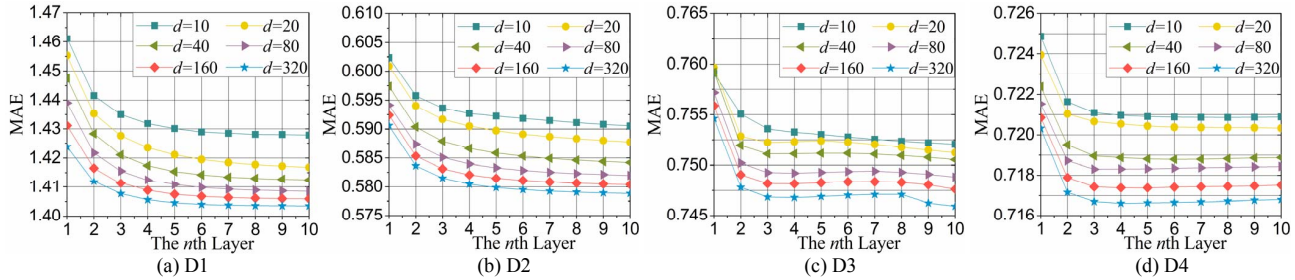

Fig. 8. The lowest MAE of PMLF at different layers as d increases on all the datasets.

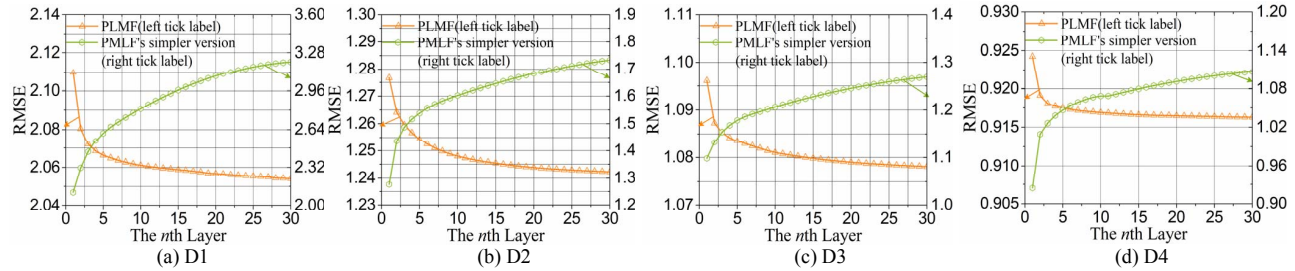

Fig. 9. The lowest RMSE of PMLF and its simpler version at different layers on all the datasets.

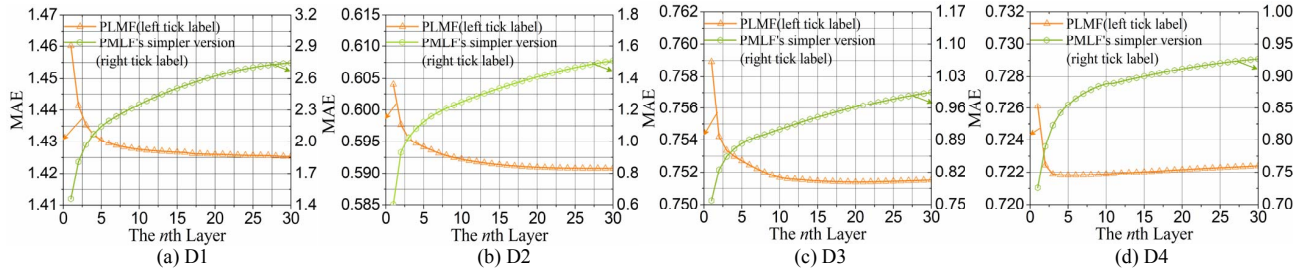

Fig. 10. The lowest MAE of PMLF and its simpler version at different layers on all the datasets.

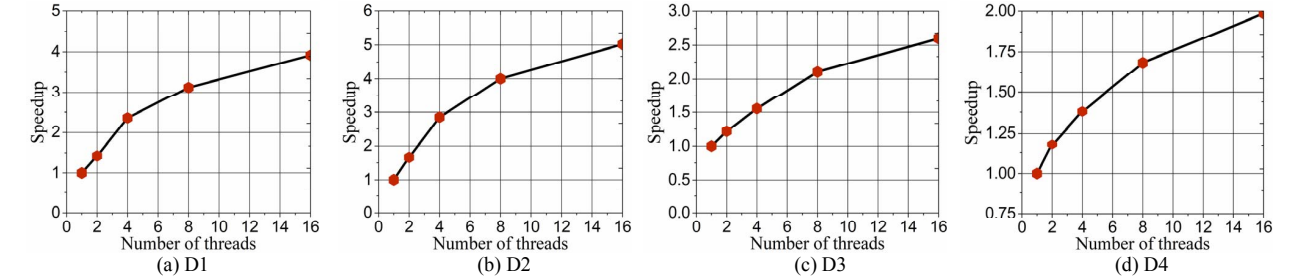

Fig. 11. The speedup results of parallel PMLF as the number of threads increases on all the different datasets.

VI. Discussion

A. Why can PMLF Improve an LF Model's Representation Learning Ability?

There are some other cascade multilayered model designs [1, 49, 50]. They have a common principle that the output of the preceding model is used as the additional input for the next model to help training. Similarly, each layer of PMLF receives not only the targeted matrix Y, but also some prediction-sampling data come from its preceding layer. Such prediction-sampling data are generated by fully training the desired LF matrices to well approximate the input ratings. They contain some valuable high-order information of the targeted matrix Y. Next, we conduct experiments to verify the above analyses. To this end, we construct a simpler version of PMLF, i.e., using random data (same value range as the original ratings) instead of prediction sampling data as the additional input for each layer of PMLF. Then, we compare PMLF with its simpler version and their hyper-parameters are set the same. Figs. 9 and 10 record the comparison results, where we see that as the layer count increases, PMLF has the lower RMSE/MAE in general while its simpler version has contrary results. These results verify that the prediction-sampling data are the helpful additional input for an LF mode while the random data are not in such a multilayer-structured PMLF.

B. How to Improve PMLF's Computational Efficiency?

According to [39-40], the vanilla SGD-based matrix factorization for RSs can be efficiently implemented in parallel. On this basis, we can greatly improve PMLF's computational efficiency through parallelization. For example, we develop PMLF to a parallel version according to Hogwild! [40]. Specifically, at each layer, we randomly sample the training ratings first and then employ them to respectively update the LF matrices of each layer through different threads simultaneously. Please refer to [40] for details. Then, we test the speedup of parallel PMLF with different numbers of threads. Fig. 11 records the results, where we see that PMLF's computational efficiency has been significantly improved with a nearly linear speedup as the number of threads increases. Note that there are no significant differences in prediction accuracy between original PMLF and its parallel version.

C. How to Make PMLF's Layer Count Self-adaptive?

Similar to [1], PMLF's layer count can also be determined adaptively when training on a specific dataset through the following operations. First, we split the training set into two parts i.e., growing set and validation set. Second, we employ the growing set to train a deeper cascade layer first and then the validation set to evaluate its performance. If a deeper layer does not improve PMLF's performance, the current layer count is adopted. Finally, we retrain PMLF on the whole training set.

VII. Conclusion

This paper proposes a Prediction-sampling-based Multilayer-structured Latent Factor (PMLF) model to accurately and efficiently representing a high-dimensional and sparse (HiDS) user-item rating matrix from recommender systems (RSs). Similar to the principle of Deep Forest [1], a PMLF model is to generate synthetic ratings to enrich the input for a latent factor (LF) model layer by layer via an appropriately-designed generalized multilayer-structure, which can greatly enhance an LF model's representation learning ability. Extensive experiments on four HiDS matrices generated by industrial RSs are conducted to evaluate the proposed PMLF model. The results verify that, when predicting the missing data of an HiDS matrix, i) PMLF achieves much higher prediction accuracy than three state-of-the-art LF-based models do, and ii) PMLF has not only slightly higher prediction accuracy but also much higher computational efficiency than three state-of-the-art deep neural networks (DNNs)-based models. Besides, we also show that PMLF's computational efficiency can be greatly improved through parallelization. In the future, we plan to make PMLF's hyper-parameters self-adaptive based on intelligent algorithm [51-52].

References

[1] Z.-H. Zhou, and J. Feng, "Deep forest: towards an alternative to deep neural networks," *In Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI* 2017, pp. 3553-3559.

[2] X. Luo, M. Zhou, S. Li, D. Wu, Z. Liu and M. Shang, "Algorithms of unconstrained non-negative latent factor analysis for recommender systems," *IEEE Transactions on Big Data*, pp. 1-1, 2019, doi: 10.1109/TBDATA.2019.2916868.

[3] Y. Koren, and R. Bell, "Advances in collaborative filtering," *Recommender Systems Handbook*, pp. 77-118: Springer, 2015.

[4] B. Smith, and G. Linden, "Two decades of recommender systems at Amazon. com," *IEEE Internet Computing,* vol. 21, no. 3, pp. 12-18, 2017.

[5] J. Han, L. Zheng, Y. Xu, B. Zhang, F. Zhuang, P. S. Yu, and W. Zuo, "Adaptive deep modeling of users and items using side information for recommendation," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 31, no. 3, pp. 737-748, 2020.

[6] D. Wu, Q. He, X. Luo, M. Shang, Y. He, and G. Wang, "A posterior-neighborhood-regularized latent factor model for highly accurate web service QoS prediction," *IEEE Transactions on Services Computing*, pp. 1-1, 2019. doi: 10.1109/TSC.2019.2961895

[7] X. He, J. Tang, X. Du, R. Hong, T. Ren, and T. Chua, "Fast matrix factorization with nonuniform weights on missing data," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1-14, 2019.

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

[9] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Qos-aware web service recommendation by collaborative filtering," *IEEE Transactions on services computing,* vol. 4, no. 2, pp. 140-152, 2015.

[10] M. Pang, K.-M. Ting, P. Zhao, and Z.-H. Zhou, "Improving deep forest by confidence screening," *In Proceedings of the 2018 IEEE International Conference on Data Mining*, 2018, pp. 1194-1199.

[11] D. Wu, X. Luo, M. Shang, Y. He, G. Wang, and M. Zhou, "A deep latent factor model for high-dimensional and sparse matrices in recommender systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1-12, 2019. doi: 10.1109/TSMC.2019.2931393.

[12] H. Wu, Z. Zhang, K. Yue, B. Zhang, J. He, and L. Sun, "Dual-regularized matrix factorization with deep neural networks for recommender systems," *Knowledge-Based Systems,* vol. 145, pp. 46-58, 2018.

[13] 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, 2017.

[14] H. Liu, L. Jing, J. Yu, and M. K. Ng, "Social recommendation with learning personal and social latent factors," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1-1, 2019.

[15] D. Ryu, K. Lee, and J. Baik, "Location-based web service QoS prediction via preference propagation to address cold start problem," *IEEE Transactions on Services Computing*, 2018.

[16] C. Leng, H. Zhang, G. Cai, I. Cheng, and A. Basu, "Graph regularized Lp smooth non-negative matrix factorization for data representation," *IEEE/CAA Journal of Automatica Sinica,* vol. 6, no. 2, pp. 584-595, 2019.

[17] C. Wang, Q. Liu, R. Wu, E. Chen, C. Liu, X. Huang, and Z. Huang, "Confidence-aware matrix factorization for recommender systems," *In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018, pp. 434-442.

[18] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature,* vol. 521, no. 7553, pp. 436, 2015.

[19] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Computing Surveys,* vol. 52, no. 1, pp. 5:1-5:38, 2019.

[20] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," *In Proceedings of the 26th International World Wide Web Conference*, 2017, pp. 173-182.

[21] H.-J. Xue, X.-Y. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems," *In Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI*, 2017, pp. 3203-3209.

[22] S. Zhang, Y. Tay, L. Yao, B. Wu, and A. Sun, "DeepRec: An open-source toolkit for deep learning based recommendation," *In Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI*, 2019, pp. 6581-6583.

[23] S. Zhang, L. Yao, A. Sun, S. Wang, G. Long, and M. Dong, "NeuRec: on nonlinear transformation for personalized ranking," *In Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI*, 2018, pp. 3669-3675.

[24] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "AutoRec: Autoencoders meet collaborative filtering," *In Proceedings of the 24th International Conference on World Wide Web*, ACM, 2015, pp. 111-112.

[25] S. Li, J. Kawale, and Y. Fu, "Deep collaborative filtering via marginalized denoising autoencoder," *In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, ACM, 2015, pp. 811-820.

[26] S. Cao, N. Yang, and Z. Liu, "Online news recommender based on stacked auto-encoder," *2017 IEEE/ACIS 16th International Conference on Computer and Information Science*, 2017, pp. 721-726.

[27] 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.

[28] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-n recommender systems," *In: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining,* 2016, pp. 153-162.

[29] O. Shumpei, T. Yukihiro, O. Shingo, and T. Akira, "Embedding-based News Recommendation for Millions of Users," *In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1933-1942.

[30] 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 proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2017.

[31] P. Li, Z. Wang, Z. Ren, L. Bing, and W. Lam, "Neural rating regression with abstractive tips generation for recommendation," *In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* 2017, pp. 345-354.

[32] X. He, and T.-S. Chua, "Neural factorization machines for sparse predictive analytics," *In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017, pp. 355-364.

[33] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, and T.-S. Chua, "Attentional factorization machines: learning the weight of feature interactions via attention networks," *In Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI*, 2017, pp. 3119-3125.

[34] L. Zheng, V. Noroozi, and P. S. Yu, "Joint deep modeling of users and items using reviews for recommendation," *In Proceedings of the 10th ACM International Conference on Web Search and Data Mining, WSDM* 2017, pp. 425-434.

[35] D. H. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware rcommendation," *In Proceedings of the 10th ACM Conference on Recommender Systems*, 2016, pp. 233-240.

[36] P. Massa, and P. Avesani, "Trust-aware recommender systems," *In Proceedings of the 2007 ACM conference on Recommender systems*, 2007, pp. 17-24.

[37] 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 proceeding of the 23rd Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD*, 2019, pp. 384-399.

[38] D. Wu, X. Luo, M. Shang, Y. He, G. Wang, and X. Wu, "A data-characteristic-aware latent factor model for web service QoS prediction," *IEEE Trans. on Knowledge and Data Engineering*, pp. 1-1, 2020, doi: 10.1109/TKDE.2020.3014302.

[39] X. Shi, Q. He, X. Luo, Y. Bai and M. Shang, "Large-scale and Scalable Latent Factor Analysis via Distributed Alternative Stochastic Gradient Descent for Recommender Systems," *IEEE Transactions on Big Data*, 2020, doi: 10.1109/TBDATA.2020.2973141.

[40] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," *Advances in Neural Information Processing Systems*, 2011, pp. 693-701.

[41] Q. Gu, J. Zhou,and C. Ding, Collaborative filtering: weighted nonnegative matrix factorization incorporating user and item graphs, *In Proceedings of the the 2010 SIAM International Conference on Data Mining*, 2010, PP.199-210**.**

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

[43] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *Information Retrieval,* vol. 4, no. 2, pp. 133-151, 2001.

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

[45] X. Wang, Y. Xu, X. He, Y. Cao, M. Wang, and T.-S. Chua, "Reinforced negative sampling over knowledge graph for recommendation," *In Proceeding of The Web Conference 2020*, 2020, pp. 99-109.

[46] J.-w. Lee, M. Choi, J. Lee, and H. Shim, "Collaborative distillation for top-N recommendation," *In Proceeding of 2019 IEEE International Conference on Data Mining*, 2019, pp. 369-378.

[47] X. Luo, Z. Liu, S. Li, M. Shang, and Z. Wang, "A fast non-negative latent factor model based on generalized momentum method," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1 - 11, 2018. doi: 10.1109/TSMC.2018.2875452.

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

[49] J. Gama, and P. Brazdil, "Cascade Generalization," *Machine Learning,* vol. 41, no. 3, pp. 315-343, 2000.

[50] H. Zhao, and S. Ram, "Constrained cascade generalization of decision trees," *IEEE Transactions on Knowledge and Data Engineering,* vol. 16, no. 6, pp. 727-739, 2004.

[51] B. Chopard, and M. Tomassini, "Particle Swarm Optimization," *An Introduction to Metaheuristics for Optimization*, pp. 97-102: Springer, 2018.

[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 Transactions on Industrial Informatics*, vol. 14, no. 3, pp. 909- 920, 2018.