A Fast Nonnegative Autoencoder-based Approach to Latent Feature Analysis on High-Dimensional and Incomplete Data

Fanghui Bi, Tiantian He, *Member, IEEE*, and Xin Luo, *Senior Member, IEEE*

Abstract—High-Dimensional and Incomplete (HDI) data are frequently encountered in various big data-related applications. Despite its incompleteness, an HDI data repository contains rich knowledge and patterns concerning the complex interactions among numerous nodes. Recently, a Neural Network (NN)-based approach to Latent Feature Analysis (LFA) model becomes popular owing to its strong representation learning ability to HDI data. Nevertheless, existing NN-based LFA models neglect the inherent nonnegativity in most HDI data, resulting in representation accuracy loss. Motivated by this discovery, this study innovatively proposes a **F**ast **N**onnegative **A**uto**E**ncoder (FNAE)-based approach to LFA on HDI data, whose ideas are three-fold: a) constructing a multilayered autoencoder subject to nonnegativity constraints for high representation learning ability; b) incorporating the data density-oriented modeling mechanism into FNAE's input and output layers for high computational and storage efficiency; and c) implementing an Adam-based single latent factor-dependent, nonnegative and multiplicative update algorithm for efficient model training as well as fulfilling the nonnegativity constraints. Experimental results on eight commonly-adopted HDI matrices from industrial applications demonstrate that the proposed FNAE significantly outperforms several state-of-the-art NN-based LFA models in both estimation accuracy for missing links of an HDI matrix and computational efficiency.

Index Terms—Knowledge Acquisition, Data Science, High-Dimensional and Incomplete Data, Neural Network, Fast Nonnegative AutoEncoder, Latent Feature Analysis, Link Prediction, Network Representation Learning.

1 Introduction

LARGE-SCALE interaction data among numerous nodes are commonly found in big data-related applications such as recommender systems [1], [2], [3], cloud services [4], [5], [6], and blockchain networks [7], [8]. As the number of involved nodes explosively increases, it becomes impossible to achieve the full interaction mapping among them, making the resultant interaction data highly incomplete. For instance, when performing service selection, it is unrealistic to achieve the full Quality-of-Service (QoS) records regarding various properties between numerous of users and services [9]. Hence, such interaction data are commonly modeled as a High-Dimensional and Incomplete (HDI) matrix [1], [2], [3], [4], [5], [6], [7], [8]. Despite its high incompleteness, an HDI matrix consists of massive valuable knowledge regarding various patterns, e.g., user preferences [1], [2], [3], [9], [10], [11], protein-protein interactions [12], [13], and potential communities [14], [15]. To acquire such knowledge from a given HDI matrix, building an efficient representation learning model is vital and thorny.

To tackle this critical challenge, several cutting-edge models for learning representations to HDI data are developed. Among them, Matrix Factorization (MF)-based models for nonnegative Latent Feature Analysis (LFA) [16], [17], [18], [19], [20], [21], [22], [23] have attracted much attention for their success in extracting accurate nonnegative latent features from HDI data. Given an HDI matrix, they map each node's features into a low dimensional nonnegative latent feature space with matrix factorization, and then estimate the missing data by calculating the inner product among nodes. To obtain desired latent features subject to nonnegativity constraints, a Single Latent Factor-dependent, Nonnegative and Multiplicative Update (SLF-NMU) algorithm and an Alternating-Direction-Method-of-Multipliers (ADMM)-based learning scheme are commonly adopted to optimize latent features on the known part of an HDI matrix [16], [21], [22], [23]. Due to the inherent nonnegativity of most industrial HDI data, a matrix factorization-based nonnegative LFA model enjoys an efficient representation learning process. However, up to now, all the models are linear, leading to limit their representation capacity and can hardly discover deep-level nonlinear latent features in HDI data.

On the other hand, Neural Network (NN)-based approaches have proven to be highly effective in extracting complicated nonlinear features and are widely introduced to LFA tasks for HDI data. An AutoEncoder (AE)-based LFA model takes interactions as input and reconstructs them to generate nonlinear latent features [24], [25], [26], [27], [28], [29], e.g., an AE-based recommendation model [24], a long short time memory-incorporated AE model [25], a stacked AE-based model [26], a collaborative denoising AE-based model [27], a hypergraph variational AE-based model [28], and a fast AE-based model [29]. Great research efforts [30], [31], [32], [33], [34] replace inner product in MF with neural network to perform nonlinear transformations thereby capturing complex interactions among nodes, e.g., a neural MF-based model [30], a collaborative memory network-

- *F. Bi is with the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, and also with the Chongqing School, University of Chinese Academy of Sciences, Chongqing 400714, China. E-mail: bifanghui@cigit.ac.cn.*
- *T. He is with the Center for Frontier AI Research, Institute of High Performance Computing, Agency for Science, Technology and Research (A*STAR), 699010, Singapore. E-mail: he_tiantian@cfar.a-star.edu.sg.*
- *X. Luo is with the College of Computer and Information Science, Southwest University, Chongqing 400715, China. E-mail: luoxin@swu.edu.cn.*
- *Corresponding Author: X. Luo.*

based model [31], a deep MF-based model [32], an attention-based model [33], and an outer product-based neural collaborative filtering model [34]. Recent Graph Neural Network (GNN)-based LFA models [35], [36], [37], [38], [39], [40], [41] aggregate high-order connectivity information of each node's multi-hop neighbors to enrich its representation by exploiting the interaction graph, e.g., a neural graph collaborative filtering model [35], a graph neural network-based MF model [36], a light graph convolutional network-based model [37], an ultra graph convolutional network-based model [38], a self-supervised graph learning model [39], a graph trend filtering network-based model [40], and a hypergraph contrastive collaborative filtering model [41]. Prior studies also resort to other sophisticated attempts to acquire nonlinear representations, e.g., a federated MF-based model [42] and a neural autoregressive distribution estimator-based model [43].

Existing neural network-based LFA models perform well in extracting nonlinear latent features and possess excellent adaptability to side information. Nevertheless, they invariably neglect the inherent nonnegativity in most HDI data and do not fulfill nonnegativity constraints, resulting in a lot of negative features. As unveiled in [19], [20], [21], [22], [23], [44], [45], given an HDI matrix, nonnegative latent features can describe its nonnegative nature more precisely and enhance the representation to its partial characteristics. Consequently, to further improve the representation learning ability, investigating an efficient nonlinear and nonnegative LFA model is necessary.

Given a full matrix, several AE-based Nonnegative Matrix Factorization (NMF) models have been proposed by pioneer researchers [46], [47], [48], [49], [50]. Ren *et al.* [46] construct a deep NMF model that adopts the neural networks to implement the Nonnegative and Multiplicative Update (NMU) process for desired latent features. Ye *et al.* [47] propose a deep linear AE-like framework for NMF, where the latent features are also updated by NMU. Yang *et al.* [48] extend the orthogonal NMF model with the AE-based deep-layered structure while reserving the orthogonality constraints. Huang *et al.* [49] implement a linear deep AE-based NMF method to multi-view data representation. Zhao *et al.* [50] present a multi-view graph regularized AE-based NMF model with self-updating weights that balance each view's effects. These existing models have proven to be effective in addressing a full matrix like images. However, given an HDI matrix, these models suffer from the following serious issues:

(1) First, existing AE-based NMF models are all designed for a full matrix. When handling an HDI matrix, they require the artificial imputations of its unknown entries for facilitating the NMU-based update as shown in Fig. 1. Considering the high incompleteness of an HDI matrix, such pre-imputation strategy will lead to unnecessarily high computational costs. Moreover, the artificially-imputed data may bend the learning process to the known part of the target HDI matrix, making the resultant latent features lose the precise representation to an HDI matrix and leading to probable accuracy loss.

(2) Second, existing AE-based NMF models mostly adopt AE's multilayered structure but weaken the nonlinearity [47], [48], [49], [50]. Therefore, they cannot well represent complex HDI matrices as shown in the experiments of this study.

Motivated by these discoveries, we have the following critical research question: *Is it possible to incorporate nonnegativity constraints into neural networks to perform nonlinear LFA tasks on HDI data, i.e., implement a neural network-based nonnegative LFA model with both high representation learning accuracy to HDI data and high efficiency?* To answer it, this paper proposes a **F**ast **N**onnegative **A**uto**E**ncoder-based latent feature analysis (FNAE) model with the following ideas:

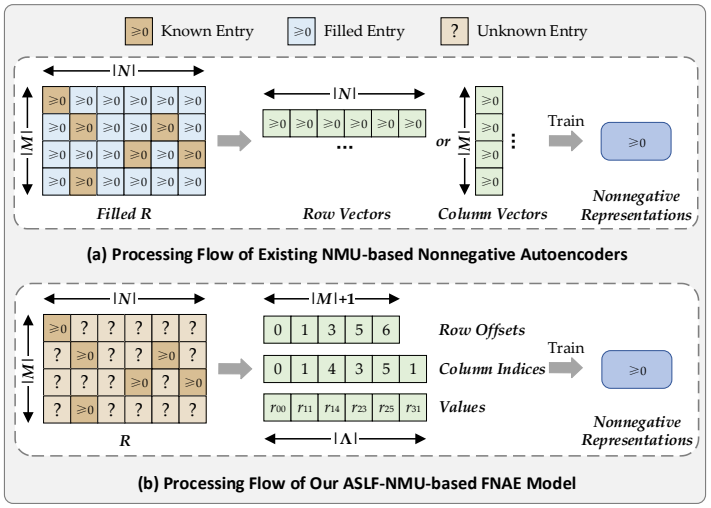


Fig. 1. An illustrative example for the processing flow of existing NMU-based nonnegative autoencoders for MF and the proposed ASLF-NMU-based FNAE model. The formers are designed for a full matrix, i.e., their NMU-based updates require prefilling the unknown part of an HDI matrix, but the latter directly handle an HDI matrix based on the known entries only.

(1) It follows the data density-oriented principle and activate the input layer nodes based only on the known data in an HDI matrix, which avoids accuracy and efficiency loss caused by the artificial value imputation. The output layer is similar as it only optimizes the reconstruction errors between the known entries and their estimations rather than between the whole input and output matrices as in prior AE-based models. Overall, the data density-oriented mechanism enables FNAE to obtain lower reconstruction errors and higher latent feature extraction efficiency.

(2) It completely reserves the nonlinearity of an AE-based network under the nonnegativity constraints via the presented **A**dam-based **S**ingle **L**atent **F**actor-dependent, **N**onnegative and **M**ultiplicate **U**pdate (ASLF-NMU) algorithm. Therefore, it is able to well handle complex HDI matrices filled with nonnegative data and nonlinear patterns.

By doing so, this work aims to make the following distinctive contributions:

(1) A novel multilayered autoencoder subject to the nonnegativity constraints is built to learn nonlinear and nonnegative latent features for performing highly accurate representation learning to an HDI matrix.

(2) The data density-oriented modeling principle is incorporated into the input and output layers of the FNAE model for high storage and computational efficiency. The CUDA-based approach to computation acceleration for HDI data is also implemented with the specifically optimized operations on incomplete data to reduce the training time.

(3) An ASLF-NMU algorithm is newly proposed to optimize the latent features for fast convergence under the nonnegativity constraints.

Extensive empirical studies have been conducted with care on eight large-scale real-word HDI matrices. The obtained results demonstrate that the above distinctive contributions enable the proposed FNAE model significantly outperforms existing AE-based NMF models and other state-of-the-art models in performing representation learning to HDI data.

The remainder of this paper is organized as follows. Section 2 provides the Preliminaries. Section 3 presents the FNAE model. Section 4 gives the Experimental Results and Analysis. And at last, Section 5 draws the Conclusions.

2 Preliminaries

Table 1 summarizes the symbols used in this paper. An HDI matrix quantifying a certain kind of relationship among two large node sets is regarded as the fundamental input of an LFA model, which is defined as [7], [8], [19], [20], [21], [22], [23]:

***Definition* 1. (*An HDI matrix*).** Given two node sets M and N, $R^{|M|\times|N|}$ is a matrix in which each entry $r_{m,n}$ describes the interaction among $m\in M$ and $n\in N$. Let Λ and O denote R's known and unknown entry sets, R is an HDI matrix if $|\Lambda|\ll|\mathrm{O}|$.

Note that an LFA model learn the latent features of M and N by seeking for a given R's low-rank approximation $\hat{R}$, and then based on the obtained latent features to estimate the missing data. According to prior studies [7], [8], [16], [17], [18], [19], [20], [21], [22], [23], an LFA model can perform effective representation learning to an HDI matrix, which is defined as:

***Definition* 2. (*An LFA model*).** Given R, an LFA model builds the rank-D approximation $\hat{R}$, where $\hat{r}_{m,n}$ is the estimation for each $r_{m,n}\in R$ and the latent feature space dimension $D\ll\min\{|M|, |N|\}$.

To obtain the desired latent features and approximation, a commonly-adopted Euclidean distance-based objective function [16], [22], [23], [38], [39], [44], [45], [51] defined only on Λ to measure the difference between R and $\hat{R}$ is formulated as:

$$\varepsilon=\frac{1}{2}\sum_{r_{m,n}\in\Lambda}\left(r_{m,n}-\hat{r}_{m,n}\right)^2. \tag{1}$$

TABLE 1
Adopted Symbols and Descriptions.

Sym.	Description
M, N	Involved node sets of HDI data.
$R^{\|M\|\times\|N\|}$	An HDI matrix of interactions among M and N.
$\hat{R}$	R's low-rank approximation.
D	Rank of $\hat{R}$, and the dimensionality of hidden units.
$r_{m,n}$, $\hat{r}_{m,n}$	Single entries of R and $\hat{R}$.
Λ, O	Known and unknown entry sets in R.
R_B, $\hat{R}_B$	A batch-size HDI interaction submatrix of R and $\hat{R}$.
Λ_B	A batch-size subset of Λ regarding R_B.
S	Batch size of R_B, $\hat{R}_B$, and Λ_B.
K	Number of hidden layers.
F^k	Node count of k-th hidden layer.
H^k	Latent feature matrix of k-th hidden layer.
$h^k_{m,f}$	Single entries of H^k.
W^k, B^k	Weight matrix and bias vector of k-th layer.
$w^k_{d,f}$, b^k_f	Single variables in W^k and B^k.
V, C	Weight matrix and bias vector of output layer.
$v_{f,n}$, c_n	Single variables in V and C.
l, o	Estimates of the first moment (the mean) and the second moment (the uncentered variance).
$\hat{l}$, $\hat{o}$	Bias-corrected estimates of l and o.
e, p	Negative and nonnegative components of gradients.
$\hat{\lambda}$	Adaptive learning rate ensuring the nonnegativity.
$\lceil\cdot\rceil$	Ceiling function of an involved number.
$a_k(\cdot)$	Activation function of k-th hidden layer.
$i(\cdot)$	Identity function of output layer.
$\varepsilon(\cdot)$	Loss function.
$\Lambda(m)$, $\Lambda(n)$	Subsets of Λ related to $m\in M$ and $n\in N$.
H, Ψ, Φ	Training, validation, and testing datasets from Λ.
β_1, β_2	Coefficients controlling the decay rates of l and o.
τ	Smoothing term avoiding division by zero.
λ	Initial learning rate of hidden layers.
η	L_2 regularization coefficient.
T_1	Time cost of Algorithm 1.
S_1	Storage cost of Algorithm 1.
t	Current training epoch.
T	Maximum training epoch.

3 An FNAE Model

3.1 Structure and Learning Objective

Given two node sets M and N, and an HDI matrix R, an AE-based LFA model [26], [27], [28], [29] works with three-fold fundamental ideas: 1) encoding input vectors to acquire latent features of M and N by an encoder module; 2) reconstructing the approximation $\hat{R}$ based on the obtained latent features by a decoder module; and 3) reducing the reconstruction loss between R and $\hat{R}$ to obtain final optimal latent features. In light of its effectiveness in various data analysis tasks, we further extend it to extract nonlinear and nonnegative latent features to represent HDI data in this study. Following the data density-oriented mechanism and incorporating nonnegativity constraints into a nonlinear autoencoder, we redesign the structure of FNAE. It adopts m-oriented input data as shown in Fig. 2, and its details are as follows.

Input Layer. The input layer receives batch-size (e.g., batch size S=512) interaction data of an HDI matrix as presented in Fig. 2, where the known data are marked nonnegative to indicate the nonnegativity of most industrial data and the unknown ones are represented with interrogation marks. Note that existing AE-based models mostly fill the unknown inputs with artificial values to obtain a complete matrix as the input [26], [27], [46], [47], which inevitably causes accuracy loss and redundant calculations since such data imputation ignores the natural characteristics of an HDI matrix, i.e., high incompleteness and imbalanced distributions of its known entries. Therefore, we follow the data density-oriented principle and activate the input layer nodes based only on the known data in an HDI matrix, which avoids accuracy and efficiency loss caused by the artificial value imputation. The output layer is similar as it only optimizes the reconstruction errors between the known entries and their estimations rather than between the whole input and output matrices as in prior AE-based models. Overall, the data density-oriented mechanism enables FNAE to obtain lower reconstruction errors and higher latent feature extraction efficiency.

Hidden Layer. An FNAE model achieves multiple hidden layers to improve the representation learning to an HDI matrix. Specifically, for the nodes of different layers, the target propagation rules are designed. Let H^k denote the k-th layer latent feature matrix, W^k, B^k, and F^k denote the corresponding weight matrix, bias vector, and dimensionality. In the first layer, we map the input data to an F^1-dimensional latent feature space, where $F^1\ll\min\{|M|, |N|\}$. For simplicity, we set the node count uniformly in each layer, i.e., $F^1=F^2=\cdots=F^K=D$. Hence, we present the mapping formula of first hidden layer as:

$$h^1_{m,f}=a_1\left(\sum_{n\in\Lambda(m)}r_{m,n}w^1_{n,f}+b^1_f\right), \tag{2}$$

where $r_{m,n}$, $h^1_{m,f}$, $w^1_{n,f}$, and b^1_f are the single elements in R, H^1, W^1, and B^1, $\Lambda(m)$ represents the known entry subset related to m, and $a_1(\cdot)$ denotes the first-layer activation function, which can be sigmoid, hyperbolic tangent (tanh) or Rectified Linear Unit (ReLU), among others. Owing to the fact that $|\Lambda(m)|\ll|N|$ in HDI data,

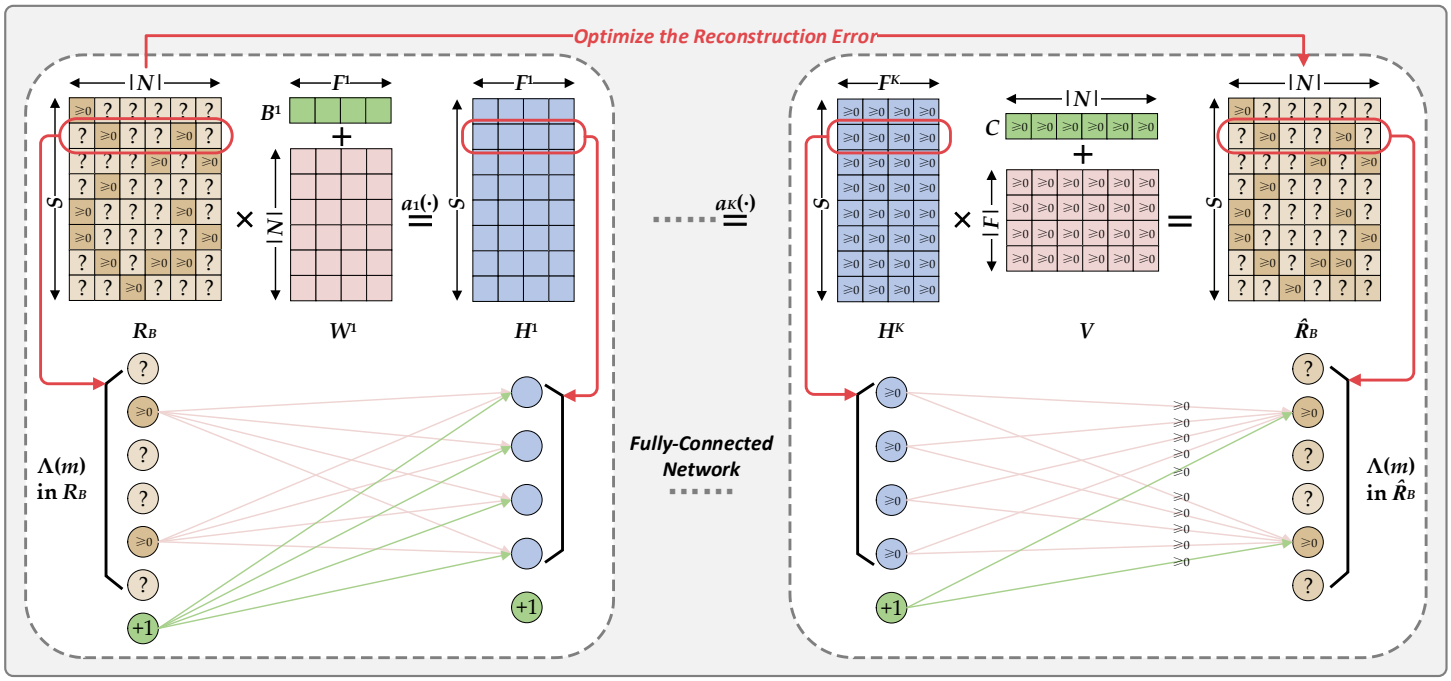


Fig. 2. An illustration of FNAE's overall structure. In the input layer, it receives a highly-incomplete nonnegative interaction matrix, where only the input nodes regarding known data are activated. Afterwards, it builds the nonlinear and nonnegative latent features H^K for each node with a multilayered fully-connected network. In the last layer, it rebuilds the known data in R on the nonnegative weight matrix V and bias vector C, which are learnt by our proposed ASLF-NMU algorithm. Best viewed in color.

Eq. (2) greatly reduces the storage and computational costs. For the (2~K−1)-th layers, we adopt a multilayered and fully-connected network:

$$h_{m,f}^{k} = a_k\left(\sum_{d\in F^{k-1}} h_{m,d}^{k-1} w_{d,f}^{k} + b_f^{k}\right), \tag{3}$$

where $h_{m,f}^{k}$, $h_{m,d}^{k-1}$, $w_{d,f}^{k}$, and b_f^{k} are the single elements in H^k, H^{k-1}, W^k, and B^k respectively, F^{k-1} is the node count of (k−1)-th hidden layer, and $a_k(\cdot)$ is an activation function. To acquire the resultant nonnegative latent features, the K-th layer is given by:

$$\begin{aligned} &h_{m,f}^{K} = a_K\left(\sum_{d\in F^{K-1}} h_{m,d}^{K-1} w_{d,f}^{K} + b_f^{K}\right), \\ &s.t.\ h_{m,f}^{K} \geq 0, \end{aligned} \tag{4}$$

where $h_{m,f}^{K}$, $h_{m,d}^{K-1}$, $w_{d,f}^{K}$, and b_f^{K} are the single elements in H^K, H^{K-1}, W^K, and B^K respectively, F^{K-1} is the node count of (K−1)-th hidden layer, and $a_K(\cdot)$ is a strictly nonnegative activation function to ensure the nonnegativity constrains of H^K. By summarizing the propagation rules of hidden layers, we have:

$$\begin{aligned} &\forall k \in \{1,2,\cdots,K\}, h_{m,f}^{k} \in H^{k}: \\ &h_{m,f}^{k} = \begin{cases} a_1\left(\sum_{n\in\Lambda(m)} r_{m,n} w_{n,f}^{1} + b_f^{1}\right) & if\ k=1, \\ a_k\left(\sum_{d\in F^{k-1}} h_{m,d}^{k-1} w_{d,f}^{k} + b_f^{k}\right) & if\ k=2\sim K-1, \\ a_K\left(\sum_{d\in F^{K-1}} h_{m,d}^{K-1} w_{d,f}^{K} + b_f^{K}\right) & if\ k=K, \end{cases} \\ &s.t.\ H^K \geq 0, \end{aligned} \tag{5}$$

where sigmoid is chosen as activation function in each layer uniformly for simplicity and the nonnegativity constraints of the K-th layer's latent feature matrix H^K is fulfilled naturally. In comparison with other nonlinear LFA models [30], [31], [32] like a neural matrix factorization model [30], FNAE's efficient multilayered design for hidden layers facilitates more efficient and effective representation learning to nonlinear patterns in an HDI matrix.

Output Layer. As shown in Fig. 2, the output estimation matrix $\hat{R}_B$ of input data is obtained by summing a bias vector C with the product of two dense matrices H^K and V. Like the input layer, we only activate the output layer nodes corresponding to the known entries in R_B thereby achieving high computational and storage efficiency. Note that the weight matrix V and the bias vector C are defined as nonnegative to correctly represent the nonnegative weights of an HDI matrix. Therefore, the estimation value for each known entry of R_B is calculated as:

$$\begin{aligned} &\hat{r}_{m,n} = i\left(\sum_{f\in F^K} h_{m,f}^{K} v_{f,n} + c_n\right), \\ &s.t.\ v_{f,n} \geq 0, c_n \geq 0, \end{aligned} \tag{6}$$

where $\hat{r}_{m,n}$, $h_{m,f}^{K}$, $v_{f,n}$, and c_n are the single elements in $\hat{R}$, H^K, V, and C respectively, F^K is the node count of K-th hidden layer, and $i(\cdot)$ is an identity function. The nonnegative learning rules of $v_{f,n}$ and c_n are carefully designed and introduced in next part.

Learning Objective. To reduce the reconstruction error, FNAE minimizes the Euclidean distance between R_B and $\hat{R}_B$, i.e., the objective function of FNAE is given as:

$$\begin{aligned} &\varepsilon\left(W^1, W^2, \cdots, W^K, B^1, B^2, \cdots, B^K, V, C\right) \\ &= \frac{1}{2}\left(\sum_{r_{m,n}\in\Lambda_B} \left(r_{m,n} - \hat{r}_{m,n}\right)^2 + \eta\left(\begin{array}{l} \left\|W^1\right\|_F^2 + \left\|W^2\right\|_F^2 + \cdots + \left\|W^K\right\|_F^2 + \left\|V\right\|_F^2 \\ + \left\|B^1\right\|_2^2 + \left\|B^2\right\|_2^2 + \cdots + \left\|B^K\right\|_2^2 + \left\|C\right\|_2^2 \end{array}\right)\right), \\ &s.t.\ \mathrm{V} \geq 0, C \geq 0, \end{aligned} \tag{7}$$

where Λ_B denotes the known entry set in R_B and η is the unified L_2 regularization coefficient of trainable variables.

3.2 Learning Scheme

In this part, to efficiently learn the FNAE parameters, we present our carefully-designed learning scheme. We start with defining some important intermediate variables constituting the gradients of FNAE's parameters; and then we introduce the optimization rules for desired parameters $\{W^1, W^2, \cdots, W^K, B^1, B^2, \cdots, B^K\}$; at last, we propose our novel Adam-based Single Latent Factor-dependent, Nonnegative Multiplicate Update (ASLF-NMU) algorithm to efficiently update V and C with the fulfillment of the nonnegative constraints.

Intermediate Variables. Given (5)~(7), to obtain the gradients of parameters, we define the following intermediate variables:

$$\delta_{m,f}^{k}=\begin{cases}h_{m,f}^{K}\left(1-h_{m,f}^{K}\right)\sum\limits_{n\in\Lambda(m)}v_{f,n}\left(\hat{r}_{m,n}-r_{m,n}\right) & \text{if } k=K,\\ h_{m,f}^{k}\left(1-h_{m,f}^{k}\right)\sum\limits_{d=1}^{D}w_{f,d}^{k+1}\delta_{m,d}^{k+1} & \text{otherwise.}\end{cases}\tag{8}$$

Based on (8), we have:

$$\forall w_{n,f}^{1}\in W^{1},b_{f}^{1}\in B^{1}:\begin{cases}\dfrac{\partial\varepsilon}{\partial w_{n,f}^{1}}=\sum\limits_{m\in\Lambda(n)}r_{m,n}\delta_{m,f}^{1}+\eta w_{n,f}^{1},\\ \dfrac{\partial\varepsilon}{\partial b_{f}^{1}}=\sum\limits_{m=1}^{S}\delta_{m,f}^{1}+\eta b_{f}^{1};\end{cases}\tag{9}$$

$$\forall k\in\{2,\cdots,K\},w_{f,d}^{k}\in W^{k},b_{d}^{k}\in B^{k}:\begin{cases}\dfrac{\partial\varepsilon}{\partial w_{f,d}^{k}}=\sum\limits_{m=1}^{S}h_{m,f}^{k-1}\delta_{m,d}^{k}+\eta w_{f,d}^{k},\\ \dfrac{\partial\varepsilon}{\partial b_{d}^{k}}=\sum\limits_{m=1}^{S}\delta_{m,d}^{k}+\eta b_{d}^{k};\end{cases}\tag{10}$$

$$\forall v_{f,n},c_{n}\in C:\begin{cases}\dfrac{\partial\varepsilon}{\partial v_{f,n}}=\sum\limits_{m\in\Lambda(n)}h_{m,f}^{K}\left(\hat{r}_{m,n}-r_{m,n}\right)+\eta v_{f,n},\\ \dfrac{\partial\varepsilon}{\partial c_{n}}=\sum\limits_{m\in\Lambda(n)}\left(\hat{r}_{m,n}-r_{m,n}\right)+\eta c_{n}.\end{cases}\tag{11}$$

Optimization based on Adam. Considering the update for the weights and biases in hidden layers, i.e., $\{W^1, W^2, \cdots, W^K, B^1, B^2, \cdots, B^K\}$, they can be optimized by gradient descent directly. Herein, we choose the Adaptive Moment Estimation (Adam) algorithm as the optimizer [52], which is widely-adopted in NN-based models [29], [30], [35]. Based on (9) and (10), the update rules of the parameters are given as:

$$\forall\theta\ in\ \left\{W^{1},W^{2},\cdots,W^{K},B^{1},B^{2},\cdots,B^{K}\right\}:\begin{cases}g^{(t)}=\dfrac{\partial\varepsilon}{\partial\theta^{(t)}},\\ l^{(t)}\leftarrow\beta_{1}l^{(t-1)}+\left(1-\beta_{1}\right)g^{(t)},\hat{l}^{(t)}=\dfrac{l^{(t)}}{1-\beta_{1}^{t}},\\ o^{(t)}\leftarrow\beta_{2}o^{(t-1)}+\left(1-\beta_{2}\right)\left(g^{(t)}\right)^{2},\hat{o}^{(t)}=\dfrac{o^{(t)}}{1-\beta_{2}^{t}},\\ \theta^{(t+1)}\leftarrow\theta^{(t)}-\dfrac{\lambda}{\sqrt{\hat{o}^{(t)}}+\tau}\hat{l}^{(t)},\end{cases}\tag{12}$$

where $l^{(t)}$ and $o^{(t)}$ are the first moment (the mean) and the second moment (the uncentered variance) estimates corresponding to θ, $\hat{l}^{(t)}$ and $\hat{o}^{(t)}$ are the bias-corrected ones of them. Note that we adopt default values of 0.9 for β_1, 0.999 for β_2, 10^{-8} for τ, and carefully tune the initial learning rate λ [52].

Optimization based on ASLF-NMU. Due to the nonnegativity constraints of V and C, they cannot be optimized directly by Adam. Inspired by the SLF-NMU algorithm [16], [17], [18], [19], [20], [21], which has proven to be highly efficient in learning desired nonnegative features in MF, we propose a novel ASLF-NMU algorithm to efficiently learn them. Its core idea is to cancel out negative components in gradient terms in the whole process of optimizing V and C based on the Adam optimizer [52]. With nonnegative initializations, it is expected that FNAE can achieve the optimization of V and C on the nonnegative number field and the obtained nonnegative latent features can better represent the value field of nonnegative HDI data, thus improving its estimation accuracy for missing entries. Detailed inferences and explanations for ASLF-NMU are as below.

First of all, by adopting general Adam optimizer, the whole update rules of each variable in V and C are achieved as:

$$\begin{cases}v_{f,n}^{(t+1)}\leftarrow v_{f,n}^{(t)}-\dfrac{\lambda}{\left(\sqrt{\hat{o}_{v_{f,n}}^{(t)}}+\tau\right)\left(1-\beta_{1}^{t}\right)}\left(\beta_{1}l_{v_{f,n}}^{(t-1)}+\left(1-\beta_{1}\right)g_{v_{f,n}}^{(t)}\right),\\ c_{n}^{(t+1)}\leftarrow c_{n}^{(t)}-\dfrac{\lambda}{\left(\sqrt{\hat{o}_{c_{n}}^{(t)}}+\tau\right)\left(1-\beta_{1}^{t}\right)}\left(\beta_{1}l_{c_{n}}^{(t-1)}+\left(1-\beta_{1}\right)g_{c_{n}}^{(t)}\right).\end{cases}\tag{13}$$

With (11), i.e., the gradients of $v_{f,n}$ and c_n, (13) is given as:

$$\begin{cases}v_{f,n}^{(t+1)}\leftarrow v_{f,n}^{(t)}-\dfrac{\lambda\left(\beta_{1}l_{v_{f,n}}^{(t-1)}+\left(1-\beta_{1}\right)\left(\sum\limits_{m\in\Lambda(n)}h_{m,f}^{K}\left(\hat{r}_{m,n}-r_{m,n}\right)+\eta v_{f,n}^{(t)}\right)\right)}{\left(\sqrt{\hat{o}_{v_{f,n}}^{(t)}}+\tau\right)\left(1-\beta_{1}^{t}\right)},\\ c_{n}^{(t+1)}\leftarrow c_{n}^{(t)}-\dfrac{\lambda\left(\beta_{1}l_{c_{n}}^{(t-1)}+\left(1-\beta_{1}\right)\left(\sum\limits_{m\in\Lambda(n)}\left(\hat{r}_{m,n}-r_{m,n}\right)+\eta c_{n}^{(t)}\right)\right)}{\left(\sqrt{\hat{o}_{c_{n}}^{(t)}}+\tau\right)\left(1-\beta_{1}^{t}\right)}.\end{cases}\tag{14}$$

Moreover, we split the nonnegative and negative components of gradient terms, i.e., (14) is reformulated as:

$$\begin{cases}v_{f,n}^{(t+1)}\leftarrow v_{f,n}^{(t)}-\dfrac{\lambda\left(\overbrace{\beta_{1}l_{v_{f,n}}^{(t-1)}}^{\text{ambiguous}}\overbrace{-\left(1-\beta_{1}\right)\sum\limits_{m\in\Lambda(n)}h_{m,f}^{K}r_{m,n}}^{\text{nonnegative component}}\underbrace{+\left(1-\beta_{1}\right)\left(\sum\limits_{m\in\Lambda(n)}h_{m,f}^{K}\hat{r}_{m,n}+\eta v_{f,n}^{(t)}\right)}_{\text{negative component}}\right)}{\left(\sqrt{\hat{o}_{v_{f,n}}^{(t)}}+\tau\right)\left(1-\beta_{1}^{t}\right)},\\ c_{n}^{(t+1)}\leftarrow c_{n}^{(t)}-\dfrac{\lambda\left(\overbrace{\beta_{1}l_{c_{n}}^{(t-1)}}^{\text{ambiguous}}\overbrace{-\left(1-\beta_{1}\right)\sum\limits_{m\in\Lambda(n)}r_{m,n}}^{\text{nonnegative component}}\underbrace{+\left(1-\beta_{1}\right)\left(\sum\limits_{m\in\Lambda(n)}\hat{r}_{m,n}+\eta c_{n}^{(t)}\right)}_{\text{negative component}}\right)}{\left(\sqrt{\hat{o}_{c_{n}}^{(t)}}+\tau\right)\left(1-\beta_{1}^{t}\right)}.\end{cases}\tag{15}$$

Considering that the negative components in gradient terms of $v_{f,n}$ and c_n may make their values be updated into negative numbers. Hence, in a single element dependent way, we follow the principle of SLF-NMU [16], [17], [18], [19], [20], [21] to manipulate the learning rates $\lambda^{(t)}$ for each $v_{f,n}$ of V and c_n of C, for cancelling out the negative components to maintain the nonnegativity. Evidently, in (15), $(1-\beta_1)(\sum_{m\in\Lambda(n)}h_{m,f}^{K}\hat{r}_{m,n}+\eta v_{f,n}^{(t)})$ for $v_{f,n}$ and

$(1-\beta_1)(\sum_{m\in\Lambda(n)}\hat{r}_{m,n}+\eta c_n^{(t)})$ for c_n are negative components, while $(1-\beta_1)\sum_{m\in\Lambda(n)}h_{m,f}^K r_{m,n}$ for $v_{f,n}$ and $(1-\beta_1)\sum_{m\in\Lambda(n)}r_{m,n}$ for c_n are nonnegative ones. However, the (t−1)-th iteration values of $\beta_1 l_{v_{f,n}}$ for $v_{f,n}$ and $\beta_1 l_{c_n}$ for c_n are ambiguous, which can be clarified as:

$$\begin{cases}\hat{\lambda}_{v_{f,n}}^{(t)}=\begin{cases}\dfrac{v_{f,n}^{(t)}\left(\sqrt{\hat{o}_{v_{f,n}}^{(t)}}+\tau\right)\left(1-\beta_1^t\right)}{\underbrace{\beta_1 l_{v_{f,n}}^{(t-1)}+\left(1-\beta_1\right)\left(\sum_{m\in\Lambda(n)}h_{m,f}^K\hat{r}_{m,n}+\eta v_{f,n}^{(t)}\right)}_{\text{negative component corresponding to } v_{f,n}}} & \text{if } l_{v_{f,n}}^{(t-1)}>0,\\ \dfrac{v_{f,n}^{(t)}\left(\sqrt{\hat{o}_{v_{f,n}}^{(t)}}+\tau\right)\left(1-\beta_1^t\right)}{\underbrace{\left(1-\beta_1\right)\left(\sum_{m\in\Lambda(n)}h_{m,f}^K\hat{r}_{m,n}+\eta v_{f,n}^{(t)}\right)}_{\text{negative component corresponding to } v_{f,n}}} & \text{otherwise};\end{cases}\\ \hat{\lambda}_{c_n}^{(t)}=\begin{cases}\dfrac{c_n^{(t)}\left(\sqrt{\hat{o}_{c_n}^{(t)}}+\tau\right)\left(1-\beta_1^t\right)}{\underbrace{\beta_1 l_{c_n}^{(t-1)}+\left(1-\beta_1\right)\left(\sum_{m\in\Lambda(n)}\hat{r}_{m,n}+\eta c_n^{(t)}\right)}_{\text{negative component corresponding to } c_n}} & \text{if } l_{c_n}^{(t-1)}>0,\\ \dfrac{c_n^{(t)}\left(\sqrt{\hat{o}_{c_n}^{(t)}}+\tau\right)\left(1-\beta_1^t\right)}{\underbrace{\left(1-\beta_1\right)\left(\sum_{m\in\Lambda(n)}\hat{r}_{m,n}+\eta c_n^{(t)}\right)}_{\text{negative component corresponding to } c_n}} & \text{otherwise}.\end{cases}\end{cases} \tag{16}$$

With the above adaptive learning rates corresponding to $v_{f,n}$ and c_n, it is seen that when updating their values based on the rules of (15), the negative components are cancelled naturally and only the nonnegative components are reserved by directly multiplying the values in current iteration. In this way, the nonnegativity constraints of $v_{f,n}$ and c_n can be maintained in the whole optimization process by randomly initializing them with nonnegative values. Nevertheless, it is worth noting that $\hat{\lambda}^{(t)}$ may become too large owing to the change of gradients when cancelling the negative components. This cannot guarantee the convergence of $v_{f,n}$ and c_n, i.e., $\hat{\lambda}^{(t)}$ cannot be directly used to update parameters. To address this problem, we design the following adjustment strategy as:

$$\begin{cases}\lambda_{v_{f,n}}^{(t)}=\begin{cases}\lambda & \text{if } \hat{\lambda}_{v_{f,n}}^{(t)}>\lambda,\\ \hat{\lambda}_{v_{f,n}}^{(t)} & \text{otherwise};\end{cases}\\ \lambda_{c_n}^{(t)}=\begin{cases}\lambda & \text{if } \hat{\lambda}_{c_n}^{(t)}>\lambda,\\ \hat{\lambda}_{c_n}^{(t)} & \text{otherwise},\end{cases}\end{cases} \tag{17}$$

where λ is the initial value same as the one in hidden layers. Based on (17), when $\hat{\lambda}^{(t)}>\lambda$, λ is adopted to perform optimization of $v_{f,n}$ and c_n. With the update rules (15)~(17), all the negative components are cancelled out and the convergence are guaranteed by coupling the adjustment of learning rates with the hidden layers, i.e., we implement the NMU process for $v_{f,n}$ and c_n efficiently.

To summarize, with the novel ASLF-NMU algorithm, we achieve efficient optimization for FNAE while maintaining the nonnegativity constraints. And in the whole learning process, we do not introduce any new hyperparameters, i.e., we just need to tune the initialized learning rate λ in Adam.

ALGORITHM 1. FNAE

Input: Λ, M, N

	Operation	**Cost**
	/* **Initialization** */	
	Initialize W^1, W^2, …, W^K	
	Initialize B^1, B^2, …, B^K	
	Initialize V, C positively at random.	
1:	**Initialize** l, o, $\hat{l}$, $\hat{o}$, $\hat{\lambda}$, e, p	T_{11}
	Initialize β_1, β_2, τ	
	Initialize D, K, λ, η, S	
	Initialize t=1	
	/* **Adam-based Nonnegative Learning** */	
2:	**while not** converge and $t\le T$ **do**	
3:	**for** *batch*=1 to $\lceil\|M\|/batch\ size\rceil$	
4:	Sample a batch R_B from R;	
5:	**for** m=1 to S	
6:	**for** k=1 to K	
7:	**for** f=1 to D	
8:	Calculate $h_{m,f}^k$ according to (5);	
9:	**end for**	
10:	**end for**	
11:	**for** $n\in\Lambda(m)$	
12:	Calculate $\hat{r}_{m,n}$ according to (6);	
13:	**end for**	
14:	**end for**	
15:	**for** n=1 to $\|N\|$	
16:	**for** f=1 to D	
17:	Calculate the gradients of $v_{f,n}$ according to (11);	
18:	Calculate $e_{v_{f,n}}$ and $p_{v_{f,n}}$ according to (18);	
19:	Calculate $\lambda_{v_{f,n}}^{(t)}$ according to (16) and (17);	T_{12}
20:	Calculate $v_{f,n}$ according to (15);	
21:	**end for**	
22:	Calculate the gradients of c_n according to (11);	
23:	Calculate e_{c_n} and p_{c_n} according to (19);	
24:	Calculate $\lambda_{c_n}^{(t)}$ according to (16) and (17);	
25:	Calculate c_n according to (15);	
26:	**end for**	
27:	**for** k=K to 1	
28:	**for** θ in W^k, B^k	
29:	Calculate the gradients of θ according to (8)~(10);	
30:	Update θ according to (12);	
31:	**end for**	
32:	**end for**	
33:	**end for**	
34:	**end while**	
	/* **Operation Ending** */	

Output: W^1, W^2, …, W^K, B^1, B^2, …, B^K, nonnegative V and C

3.3 Algorithm Design and Analysis

Aiming at maintaining the nonnegativity, the ASLF-NMU algorithm adaptively calculates the learning rates of $v_{f,n}$ and c_n, whose gradients are decomposed to negative components and nonnegative components. For simplicity, we present the following intermediate variables corresponding to them:

$$e_{v_{f,n}}=\sum_{m\in\Lambda(n)}h_{m,f}^K\hat{r}_{m,n}+\eta v_{f,n}^{(t)}, \tag{18a}$$

$$p_{v_{f,n}}=\sum_{m\in\Lambda(n)}h_{m,f}^K r_{m,n}; \tag{18b}$$

$$e_{c_n}=\sum_{m\in\Lambda(n)}\hat{r}_{m,n}+\eta c_n^{(t)}, \tag{19a}$$

$$p_{c_n}=\sum_{m\in\Lambda(n)}r_{m,n}. \tag{19b}$$

Based on these inferences, we design Algorithm 1 for FNAE.

Algorithm 1's computational cost consists of two factors:

(a) Initialization: $T_{11}=\Theta(|N|\times D)$; (20a)

(b) Adam-based nonnegative learning: $T_{12}=\Theta(|\Lambda|\times D\times t)$. (20b)

Hence, the overall time cost of Algorithm 1 comes to:

$$T_1 = T_{11} + T_{12} \approx \Theta\left(|\Lambda|\times D\times t\right), \tag{21}$$

which is significantly lower than that of the state-of-the-art AE-based NMF or MF models like the AutoRec [24], Mult-VAE [53], DNMF [46], DANMF [47], and DAutoED-ONMF [48]. This is because these models require prefilling the unknown entries with nonnegative statistical values and activate all nodes of input and output layers, which makes their computational cost come to $\Theta(|M|\times|N|\times D\times t)$. Since the known entries in an HDI matrix are far less than the unknown ones, we have $|\Lambda|\ll\{|M|\times|N|\}$, making FNAE's computational cost lower than that of its peers.

Considering FNAE's storage cost, it involves three factors:

(a) Data cache: $S_{11}=\Theta(|\Lambda|)$; (22a)

(b) Weight and bias variable arrays: $S_{12}=\Theta(|N|\times D)$; (22b)

(c) Arrays caching intermediate variables: $S_{13}=\Theta(|N|\times D)$. (22c)

To sum up, the storage cost of FNAE is given as:

$$S_1 = S_{11} + S_{12} + S_{13} \approx \Theta\left(D\times\max\left\{|\Lambda|/D, |N|\right\}\right), \tag{23}$$

which is linear with the known entry count of the given HDI matrix and the number of latent features in the resultant FNAE model. Hence, FNAE's storage cost is also much lower than its peers' storage cost at $\Theta(|M|\times|N|)$ [46], [47], [48], [49], [50].

According to the above analysis, an FNAE model evidently possesses high computational and storage efficiency in addressing HDI data.

3.4 CUDA-based Sparse Matrix Computation

With the rapid evolution of computing hardware, most of existing AE-based LFA models can achieve GPU acceleration to extract nonlinear latent features efficiently by prefilling the unknown part of an HDI matrix. Owing to the high incompleteness of HDI data, FNAE follows the data density-oriented modeling principle to avoid the waste of computing resources and storage space caused by prefilling approaches. However, numerous sparse matrix operations involved in FNAE cannot be well supported for GPU acceleration by mainstream deep learning frameworks. Thus, based on the previous studies [54], [55], [56], we implement an efficient CUDA-based sparse matrix computation module for GPU acceleration of FNAE, whose detailed architecture is depicted in Fig. 3.

We adopt two schedulers, i.e., SM and warp, in CUDA programming to implement two-level parallelism of sparse matrix computation, as shown in Fig. 3. Compressed Sparse Row (CSR) format is adopted to re-represent an HDI matrix in the form of compressed sparse matrix, which contains three arrays: 1) *ptr* array caching the row pointers, i.e., the row offsets; 2) *idx* array caching the column indexes; 3) *val* array caching the values of the known entries. Among them, given a target HDI matrix, the size of *ptr* array is on the same order of magnitude as the number of rows in it, while the size of *idx* and *val* arrays is equal to the total number of known entries in it. To efficiently perform GPU computation for such format, we adopt CSR-oriented Sparse Matrix Operators (SMOs) [54], [55], [56], [57], e.g., the row product one, the outer product one, and the transpose one.

Note that the size of block, i.e., the number of threads in a block, greatly influences the performance of CUDA-based parallel computing. Generally, a warp consists of 32 threads, which determines that the size of block is better to be set to a multiple of 32. Considering the different number of SM in GPUs with various architectures and the suggestions in prior studies, we set the block size to 32×32 in our implementation.

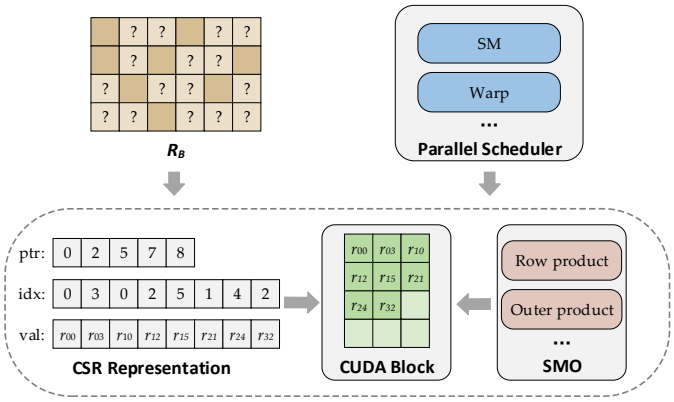


Fig. 3. CUDA-based Sparse Matrix Computation Module.

4. Experiments

To evaluate the performance of the proposed FNAE, this section presents detailed experimental results conducted on eight commonly-adopted industrial HDI datasets. Based on the obtained results, we aim at answering the following three key Research Questions (RQs):

- **RQ1.** How do individual components, e.g., the nonnegativity constraints (i.e., the ASLF-NMU algorithm) and multilayered design of hidden layers, affect FNAE?
- **RQ2.** How do different hyperparameter settings, e.g., the learning rate λ, the L_2 regularization coefficient η, and the latent feature space dimension D affect FNAE's performance?
- **RQ3.** How does FNAE perform when compared with state-of-the-art LFA methods?

4.1 General Settings

Evaluation Metrics. For a tested model, this work mainly concerns its representation learning ability to HDI data, which is commonly quantified by the estimation accuracy for the missing data in an HDI matrix. To measure this, we adopt the Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE), which are widely adopted evaluation metrics [16], [17], [18], [19], [20], [21], [22], [23]. They are computed as:

$$RMSE = \sqrt{\left(\sum_{r_{m,n}\in\Phi}\left(r_{m,n}-\hat{r}_{m,n}\right)^2\right)\Big/|\Phi|},$$

$$MAE = \left(\sum_{r_{m,n}\in\Phi}\left|r_{m,n}-\hat{r}_{m,n}\right|_{abs}\right)\Big/|\Phi|,$$

where Φ represents the testing set. Note that lower RMSE and MAE means that a tested model possesses higher estimation accuracy for the missing data of an HDI matrix, i.e., they can better represent the information hidden in its known data.

Besides, to evaluate the computational efficiency of a tested model, we carefully record its training time cost on each dataset. All experiments are implemented in Python 3.7, except that the sparse matrix computation module is programmed in CUDA C and compiled with CUDA 11.1. And all empirical studies are carried out on a server with 128-GB RAM, one 2.4-GHz Intel Xeon 4214R CPU, and four NVIDIA RTX 3090 GPUs.

Datasets. We adopt eight industrial HDI datasets in our experiments, whose properties are summarized in Table 2. Among them, D1-3 are from recommender systems MovieLens and Douban [58], [59]; D4-7 are collected by the protein interactome database STRING [60] for the species Pseudomonas aeruginosa, Emericella nidulans, Naumovozyma castellii, and Cellulophaga lytica DSM7489; D8 recording the throughput of user-service invocations are from the WS-Dream system [4], [5], [6]. The known data density of D1-8 is 0.54%, 0.22%, 0.21%, 5.04%, 1.77%, 4.38%, 5.95%, and 92.74%, respectively.

Comparison models. We compare the proposed FNAE with fourteen state-of-the-art models with high representation learning ability to an HDI matrix. Details of them are listed below.

M1. I-AutoRec [24]: An item-oriented AE-based LFA model. It builds an item-oriented autoencoder for extracting nonlinear latent features of an HDI matrix, which is efficient and widely adopted in many scenarios.

M2. Mult-VAE [53]: A variational AE-based LFA model. It is a generative model, which introduces a multinomial likelihood for the data distribution and achieves parameter estimation based on variational inference.

M3. NeuMF [30]: A neural collaborative filtering framework for LFA tasks. It incorporates a multilayered perception into MF and combines it with dot product to acquire nonlinear latent features.

M4. MetaMF [42]: A federated MF-based LFA model. It follows federated meta learning to generate private embeddings of an HDI matrix, which fully considers the limitations of mobile environments.

M5. GC-MC [61]: A graph convolutional matrix completion model. It applies graph neural networks to matrix completion and builds a graph AE framework to perform differentiable message passing on the interaction graph.

M6. NGCF [35]: A neural graph collaborative filtering model for LFA tasks. It builds a graph neural network with message and node dropout to integrate high-order connectivity from the bipartite graph structure of HDI data.

M7. LightGCN [37]: A light graph convolutional network-based model. It simplifies graph convolution operations by removing nonlinear activation and feature transformation to obtain better generation performance.

M8. LR-GCCF [62]: A linear residual graph convolutional collaborative filtering model. It omits unnecessary nonlinearities in message propagation and exploits residual connection to alleviate the over-smoothing problem.

M9. DGCN-HN [63]: A deep graph convolutional network-based LFA model. It constructs a hybrid normalization layer and a simplified attention network to adaptively choose different normalization rules.

M10. SGL-ED [64]: A self-supervised graph learning-based LFA model. It leverages node dropout to generate multiple views of a node and maximize the agreement between different views, thus reinforcing node representations.

M11. LightGCN$_{r\text{-}AdjNorm}$ [65]: A *r-AdjNorm* incorporated light graph convolutional network model. It controls the normalization strength to achieve an accuracy-novelty tradeoff in collaborative filtering.

M12. HMLET [66]: A hybrid method of linear and nonlinear collaborative filtering. It designs a gating module to determine whether each node performs linear or nonlinear message propagation in a layer.

TABLE 2
PROPERTIES OF TESTING CASES.

No.	$\lvert\Lambda\rvert$	$\lvert M\rvert$	$\lvert N\rvert$	Density
D1	20,000,263	138,493	26,744	0.54%
D2	16,830,839	129,460	58,541	0.22%
D3	10,000,054	71,567	65,133	0.21%
D4	1,559,616	5,565	5,565	5.04%
D5	1,120,030	7,963	7,963	1.77%
D6	1,182,124	5,194	5,194	4.38%
D7	638,904	3,277	3,277	5.95%
D8	1,831,253	339	5,825	92.74%

M13. HCCF [41]: A hypergraph contrastive collaborative filtering model. It jointly captures local and global collaborative relations in an interaction graph with a hypergraph-enhanced contrastive learning.

M14. GTN [40]: A graph trend filtering model. It introduces a principled graph trend filtering approach and proposes a graph trend network to capture the adaptive reliability of the interactions.

M15. DNMF [46]: A deep NMF model. It applies a nonlinear scheme to NMF via a deep autoencoder and uses neural networks to realize the NMU process for nonnegative latent features.

M16. FNAE: The model proposed in this study.

Training Settings. All involved models adopt the following settings uniformly in our experiments:

(1) The trainable weights and biases are initialized randomly with Xavier method [67], except that the variables of FNAE's last layer are initialized randomly between 0 and 0.004, and we optimize all involved models with Adam [52];

(2) For objective results, the 70%-10%-20% train-validation-test settings are applied to each dataset, i.e., 70% of Λ are divided into the training set H to train a target model, 10% into the validation set Ψ to monitor its training process, and the left 20% into the testing set Φ to conduct the performance test of the achieved model. Note that the division is disjoint and ten-fold cross validation settings are adopted to obtain the final average results;

(3) We carefully tune each model's hyperparameters to achieve its best performance, i.e., in individually-built cases H and Ψ, we tune all the hyperparameters for the lowest validation error on Ψ, and then fix the obtained settings to test and record the generalization performance on Φ;

(4) Each model adopts the same early stopping strategies. Specifically, its training process terminates if: a) its training epoch reaches a preset threshold, i.e., 1000; or b) the estimation error keeps increasing for 30 epochs.

4.2 Ablation and Effectiveness Analysis (RQ1)

Extensive ablation studies have been conducted for verifying the effectiveness of FNAE's essential components proposed in this paper, including the nonnegativity constraints and multilayered design of hidden layers. To illustrate the positive effects of FNAE's nonnegativity constraints, we implement a variant labeled FNAE-w/o-NC. Note that this variant omits the nonnegativity constraints imposed on V and C, i.e., performs model training with a normal Adam optimizer. The corresponding experimental results are summarized in Table S1 of the Supplementary File (SF)[1]. And to investigate whether FNAE can benefit from a multilayered structure, we vary the depth of the model

[1] https://github.com/Oak-B/FNAE/blob/main/FNAE-Supplementary%20File.pdf

to test its performance. Concretely, we search the hidden layer numbers, i.e., K, in the range of {1, 2, 3} on each testing case and record the results in Table S2 of SF. In Tables S1-2, FNAE's RMSE, MAE, training epochs to converge in RMSE and MAE are presented. Through analyzing them, we have the observations as below.

(1) **FNAE's estimation accuracy for missing data benefits from its nonnegativity constraints and the convergence rate is versa.** As recorded in Table S1, FNAE's RMSE/MAE is significantly lower than that of FNAE-w/o-NC on all our testing cases. For instance, as shown in Table S1, on D4, FNAE achieves the RMSE at 0.1088 and FNAE-w/o-NC achieves it at 0.1101, which means that FNAE's RMSE is 1.18% (i.e., $(Error_{high}–Error_{low})/Error_{high}$) lower than that of the latter. Considering the MAE on D4, FNAE's 0.0743 is 2.24% lower than 0.0760 achieved by removing the nonnegativity constraints. Similar outcomes are achieved on other testing cases. From these comparison results in Table S1, it is seen that FNAE's nonnegativity constraints enable it to better represent the value field of nonnegative HDI data and the proposed ASLF-NMU is effective for FNAE to learn desired nonnegative latent features [3], [13], [19], [21], [44], [68].

However, it should be noted that FNAE's convergence rate is slowed down by the nonnegativity constraints. As summarized in Table S1, FNAE-w/o-NC achieves significantly fewer training epochs than FNAE does on all testing cases. Taking the case D1 as an example, FNAE-w/o-NC takes 187 epochs to converge in RMSE and 193 epochs in MAE, which are all fewer than FNAE's 344 epochs in RMSE and 345 epochs in MAE. Note that the ASLF-NMU algorithm adopted by FNAE cancels out the negative components in gradient terms of some variables in V and C when performing optimization, i.e., some learnt information related to these variables are abandoned by FNAE to maintain the nonnegativity of V and C for better representations to the target nonnegative HDI data. Hence, FANE takes more epochs to converge than FNAE-w/o-NC does to make compensations to the aforementioned 'information dropout'. However, FANE converges much faster than its most state-of-the-art peers even with the nonnegativity constraints, which will be shown in Section 4.3.

(2) **Integrating deep structure improves FNAE's estimation accuracy for missing data in an HDI matrix.** On most testing cases, a multilayered FNAE model can better fit the hidden nonlinearity in HDI data compared with a single-layer one, thus enhancing its representation ability. For instance, as summarized in Table S2, on D8 with other hyperparameters being fixed, when K increases from 1 to 3, FNAE's RMSE reduces from 0.3791 to 0.3351 with the estimation error gap of 11.61%; and the MAE reduces from 0.1431 to 0.1062 with the estimation error gap of 25.79%. Similarly, on D3, when K is set to 1, FNAE achieves the RMSE and MAE at 0.7904 and 0.6052; as K=2, they both decrease to 0.7805 and 0.6015. Situations like this can be found on other testing cases. However, on some cases (e.g., on D4-6), stacking multiple hidden layers may lead to performance degradation due to the issues of overfitting/gradient vanishing just like other AE-based LFA models [24], [53], [61].

Note that increasing FNAE's depth also vastly influences its computational efficiency. Taking the case D1 as an example (Table S2), as K increases from 1 to 3, the converging epoch count in RMSE and MAE respectively increases from 344 to 676 and 345 to 677. Considering that FNAE takes more time per epoch as the number of hidden layers increases, the total time cost to converge increases as K increases on D1. And from Table S2, we see that the hidden layer numbers are also data-dependent in computational efficiency. Hence, to balance the estimation accuracy and computational efficiency thereby achieving FNAE's best performance, we set K=2 on D1-3 and D7, K=1 on D4-6, and K=3 on D8. In general, integrating deep structure can help FNAE better represent a target HDI matrix.

4.3 Hyperparameter Sensitivity Analysis (RQ2)

As described in Section 3, FNAE's performance is affected by learning rate λ, L_2 regularization coefficient η, and dimensionality of hidden units D. Therefore, it is an important issue to conduct hyperparameter sensitivity test with them. In this section, on D1-8, we test each hyperparameter's impact on FNAE's performance while others are being fixed. Figs. S1 and S2 of the SF illustrate the lowest errors and training epochs to converge as λ increases; Figs. S3 and S4 depict the lowest errors and converging epochs as η increases; Figs. S5 and S6 plot the lowest errors and training epochs as D increases. Through jointly analyzing these results, we have the following findings:

(1) **FNAE's RMSE, MAE, and convergence rate rely significantly on λ and η.** It is obvious that λ is important to FNAE's estimation accuracy and computational efficiency, as shown in Figs. S1 and S2. For instance, taking testing case D2 as an example (Fig. S1(b)), by setting η=0.5 and D=500, when λ increases from 2^{-14} to 2^{-11}, FNAE's RMSE increases from 0.7048 to 0.8279 where the RMSE increment comes to 14.87%; the MAE increases from 0.5499 to 0.5942 with the increment of 7.46%. Also, on D3 with other hyperparameters being fixed (Fig. S2(b), FNAE's training epochs to converge decrease from 161 to 123 in RMSE and 248 to 123 in MAE as λ increases from 2^{-14} to 2^{-11}. Similarly, on D8 (Figs. S1(h) and S2(h)), as η=0.5, D=500, and λ varies from 2^{-12} to 2^{-9}, FNAE's RMSE varies from 0.3364 to 0.3326 with the gap of 1.13% and its converging epochs decrease from 1000 to 358; the MAE varies from 0.1091 to 0.1069 with the gap of 2.02% and its converging epochs decrease from 1000 to 354.

Considering η, it is also important to FNAE's performance, as illustrated in Figs. S3 and S4. Taking testing case D1 as an example (Fig. S3(a)), with λ=2^{-14} and D=500, FNAE achieves the RMSE at 0.7860 and MAE at 0.5958 as η=10^{-4} but 0.7754 and 0.5898 as η=2, i.e., the estimation error gaps are at 1.35% in RMSE and 1.01% in MAE, respectively. And as plotted in Fig. S4(a), by fixing other hyperparameters on D1, FNAE's training epochs to converge vary from 250 to 956 in RMSE and 276 to 910 in MAE as η varies from 10^{-4} to 2. Similar situations are also found on other datasets, e.g., as presented in Fig. S3(b-h), on D2-8, as η varies while other hyperparameters are being fixed, the gaps of RMSE are achieved at 0.38%, 0.63%, 20.19%, 22.67%, 16.78%, 20.28%, and 1.42%; the gaps of MAE are achieved at 0.78%, 0.92%, 25.48%, 25.54%, 20.05%, 21.16%, and 5.87%.

It is worth noting that FNAE's converging epochs decrease as λ increases but increase as η increases. For instance, as shown in Figs. S2(c) and S4(c), on D3 with other hyperparameters being fixed, FNAE's converging epochs in RMSE decrease from 609 to 81 as λ varies from 2^{-16} to 2^{-11} but increase from 119 to 807 as η varies from 10^{-4} to 2, MAE is similar. Considering that λ is the learning rate which controls the step size of

FNAE's gradient descent and η is the L_2 regularization coefficient that controls the degree of smoothing parameters, this phenomenon is reasonable. Moreover, λ and η are data-dependent, e.g., as illustrated in Figs. S1 and S3, to achieve FNAE's best estimation accuracy for the missing data in an HDI matrix, we tune η in between 0.1 and 2 on D1-3 and D8, but in between 10^{-4} and 0.1 on D4-7. For λ, similar results are achieved. Hence, to balance FNAE's estimation accuracy and convergence rate for best performance, its hyperparameters should be tuned with care, which indicates their self-adaption mechanism is desired. Currently, we can adopt grid search to easily determine them.

(2) **Increasing *D*'s value greatly improves FNAE's estimation accuracy for an HDI matrix's missing data and its convergence rate.** Note that increasing the dimensionality of hidden units (D) expands FNAE's solution space thus learning more accurate representations. On all testing cases, the RMSE and MAE of FNAE substantially decrease with the increase of D. For instance, as shown in Fig. S5(a), by fixing $\lambda=2^{-14}$ and $\eta=0.5$ on D1, as D increases from 20 to 500, FNAE's RMSE decreases from 0.8325 to 0.7809 with the estimation error gap of 6.20%; its MAE decreases from 0.6399 to 0.5921 with the estimation error gap of 7.47%. Similar outcomes can be commonly obtained on D2-8 (Fig. S5(b-h)), as D varies from 20 to 500, FNAE's RMSE decreases by 5.42%, 7.74%, 18.79%, 13.09%, 14.74%, 12.62%, and 10.47%; the MAE decreases by 6.85%, 9.17%, 21.56%, 16.04%, 17.88%, 14.14%, and 15.62%.

Considering FNAE's convergence rate, which is also improved with D's increase. On all eight testing cases, the training epochs of FNAE decrease as D increases. For instance, as depicted in Fig. S6, on D1 with other hyperparameters being fixed, as D increases from 20 to 500, FNAE's training epochs to converge decrease from 1000 to 365 in RMSE and 1000 to 389 in MAE. Similar situations are also encountered on other testing cases. Generally, in our experiments, FNAE can have a satisfactory performance in both estimation accuracy and computational efficiency as D=500.

4.4 Comparison with State-of-the-Arts (RQ3)

Herein, we compare the proposed FNAE with fifteen state-of-the-art LFA approaches, including AE-based ones, neural MF-based ones, and GNN-based ones. Tables S3-6 of SF respectively present the RMSE, MAE, and converging time cost in RMSE and MAE achieved by M1-16 on D1-8.

To better understand the results obtained in Tables S3-6, the win/loss counts of M16 versus other comparison models are summarized in the last column [69], [70], [71]. And in Table S7, we have summarized all tested models' results of the Friedman test in estimation accuracy (RMSE and MAE) and computational efficiency (training time in RMSE and MAE). The hypothesis that all tested models are significantly different is accepted as significance level=0.05. Besides, in Tables S8-9, the Wilcoxon singed-ranks test results on each comparison model are presented to check whether M16's estimation accuracy for the missing data of an HDI matrix and computational efficiency are significantly better than that of its peers [69], [70], [71]. Note that as a widely-adopted nonparametric pairwise comparison procedure, Wilcoxon signed-ranks test has three important indicators, i.e., $R+$, $R-$, and p-value. Among them, higher $R+$ values denote higher accuracy (Table S8) and efficiency (Table S9) of FNAE, $R-$ is versa; and p-value stands for the significance level [69], [70], [71]. From the results, we achieve some important findings:

(1) **M16, i.e., FNAE's estimation accuracy for missing data of an HDI matrix significantly outperforms its peers.** As summarized in Tables S3-4, M16's RMSE/MAE is obviously lower than that of its peers on most of involved datasets owing to its nonlinear and nonnegative representation. For instance, as shown in Table S3, M16 achieves the RMSE on D4 at 0.1105, which is about 2.64% lower than M1's 0.1135, 10.74% lower than M2's 0.1238, 6.91% lower than M7's 0.1187, and 8.07% lower than M12's 0.1202. When the MAE obtained on D4 (Table S4) is considered, M16 achieves it at 0.0762, which is respectively 1.93%, 13.41%, 46.68%, 16.99%, 20.13%, 15.89%, 9.18%, 15.80%, 13.01%, 29.96%, 8.96%, 10.25%, 18.94%, 15.89%, and 18.94% lower than that of M1-15. Similar results are encountered on other testing cases. Meanwhile, M16 achieves the lowest F-rank value in estimation accuracy compared with its peers. As recorded in Table S7, M16's F-rank value in RMSE and MAE is 1.06, which is significantly lower than M1-15's 3.66, 9.81, 11.44, 10.81, 11.00, 8.63, 5.59, 5.94, 6.81, 11.69, 3.38, 7.56, 12.63, 10.81, and 15.19. And in our Wilcoxon signed-ranks tests (Table S8), M16's estimation accuracy is significantly higher than that of M1-15. And we see that the p-values of their comparisons are all within the accepted range of hypothesis with a significance level of 0.1. Thus, we conclude that FNAE achieves significantly higher estimation accuracy than its peers do.

(2) **FNAE achieves higher computational efficiency than that of its peers.** As shown in Tables S5-6, owing to its data density-oriented computational rules and ASLF-NMU algorithm, M16 achieves less time cost to converge in RMSE and MAE than its peers. For instance, as presented in Table S5, on D1, M16 consumes 5346 seconds to converge in RMSE, which is only 75.53% of 7078 seconds by M1, 7.45% of 71767 seconds by M4, 24.45% of 21863 seconds by M6, 7.39% of 72382 seconds by M13. Considering the MAE on D1 (Table S6), M15's converging time cost is 5285 seconds, which is about 74.81%, 78.31%, 4.76%, 7.65%, 12.02%, 23.93%, 15.84%, 33.00%, 6.67%, 14.76%, 20.91%, 6.27%, 7.30%, 6.49%, and 35.62% of that achieved by M1-15. On D2-8, M16 also achieves less converging time cost than most of its peers. By performing the Friedman tests (Table S7), we see that M16 possesses lower F-rank value in computational efficiency than M1-15. Concretely, M16's F-rank value on time cost to converge in RMSE and MAE is 1.31, which is vastly lower than 2.38, 2.94, 13.25, 12.56, 11.75, 8.25, 9.06, 5.81, 10.06, 9.88, 11.50, 11.63, 9.06, 12.50, and 4.06 of M1-15. Moreover, in our Wilcoxon signed-ranks test results (Table S9), the $R+$ values versus each comparison model are high and most of p-values are within the accepted range of hypothesis as significance level=0.1. Thus, compared with state-of-the-art models, M16's computational efficiency is highly competitive.

4.5 Summary

Pros. Based on the above results and analysis, we summarize that an FNAE model has several important virtues:

(1) FNAE is able to represent the nonlinearity and nonnegativity of a nonnegative HDI matrix in a precise way.

(2) FNAE has high estimation accuracy for the missing data of an HDI matrix, indicating that it is able to accurately represent the information hidden in the known data of an HDI matrix.

(3) FNAE has competitively high computational efficiency when performing representation learning to HDI data, which

should be owing to its data density-oriented modeling mechanism and ASLF-NMU-based learning algorithm.

Cons. Considering the limitations of FNAE, similar to the other neural networks-based models [24], [27], [30], [40], [41], its hyperparameters have to be tuned with care, and its optimization is GPU-dependent.

5 Conclusions

Aiming at performing highly accurate and efficient representation learning to an HDI matrix, this paper presents an FNAE model. It incorporates nonnegativity constraints into its data density-oriented autoencoder structure for acquiring nonlinear and nonnegative latent features, thereby highly improving its representation ability to an HDI matrix while enjoying high computational efficiency. Meanwhile, an ASLF-NMU algorithm is designed for achieving FNAE's fast convergence under nonnegativity constraints. Detailed experiments are carefully conducted on eight large-scale industrial HDI matrices, which validate that FNAE significantly outperforms its state-of-the-art peers. Considering the future studies, the following issues are of high interests:

(1) FNAE's excellent performance relies on nonlinear and nonnegative representation learning, whose convergence rate can be further improved with other advanced optimization strategies [72], [73], [74], e.g., Alternating-Direction-Method-of-Multipliers [72].
(2) For the convergence of the ASLF-NMU-based learning algorithm, theoretical proof is deserved.
(3) Can we explore the deep relationships among users and cloud services with the interaction graphs [75], [76]?
(4) It is attractive to apply the proposed FNAE model and ASLF-NMU algorithm to addressing other thorny issues that involve HDI data, e.g., community detection [46], [47], [77], top-K recommendation [78], [79], high-dimensional expensive problems [80], [81], and graph classification [82], [83].

We will address the above issues in the future.

Acknowledgments

The authors would like to show their gratitude to Mr. Fei Luo for his assistance in the programming of FNAE's sparse matrix computation. This research is supported by the National Natural Science Foundation of China under Grant 62272078.

References

[1] M. Fu, L. Huang, A. Rao, A. A. Irissappane, J. Zhang, and H. Qu, "A deep reinforcement learning recommender system with multiple policies for recommendations," *IEEE Trans. on Industrial Informatics*, vol. 19, no. 2, pp. 2049-2061, 2023.

[2] W. Yue, Z. Wang, J. Zhang, and X. Liu, "An overview of recommendation techniques and their applications in healthcare," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 4, pp. 701-717, 2021.

[3] 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. on Neural Networks and Learning Systems*, vol. 27, no. 3, pp. 579-92, 2016.

[4] X. Zhu, X. Jing, D. Wu, Z. He, J. Cao, D. Yue, and L. Wang, "Similarity-maintaining privacy preservation and location-aware low-rank matrix factorization for QoS prediction based web service recommendation," *IEEE Trans. on Services Computing*, vol. 14, no. 3, pp. 889-902, 2021.

[5] Y. Yang, Z. Zheng, X. Niu, M. Tang, Y. Lu, and X. Liao, "A location-based factorization machine model for web service QoS prediction," *IEEE Trans. on Services Computing*, vol. 14, no. 5, pp. 1264-1277, 2021.

[6] D. Wu, X. Luo, M. Shang, Y. He, G. Wang, and X. Wu, "A data-characteristic-aware latent factor model for web services QoS prediction," *IEEE Trans. on Knowledge and Data Engineering*, vol. 34, no. 6, pp. 2525-2538, 2022.

[7] J. Wu, J. Liu, W. Chen, H. Huang, Z. Zheng, and Y. Zhang, "Detecting mixing services via mining bitcoin transaction network with hybrid motifs," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 4, pp. 2237-2249, 2022.

[8] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, and C. Leiserson, "Evolvegcn: evolving graph convolutional networks for dynamic graphs," in *Proc. of the 34th AAAI Conf. on Artificial Intelligence*, New York, USA, 2020, pp. 5363-5370.

[9] J. Li, H. Wu, J. Chen, Q. He, and C.-H. Hsu, "Topology-aware neural model for highly accurate QoS prediction," *IEEE Trans. on Parallel and Distributed Systems*, vol. 33, no. 7, pp. 1538-1552, 2022.

[10] M. Aliannejadi, D. Rafailidis, and F. Crestani, "A joint two-phase time-sensitive regularized collaborative ranking model for point of interest recommendation," *IEEE Trans. on Knowledge and Data Engineering*, vol. 32, no. 6, pp. 1050-1063, 2020.

[11] J. Tang, X. Du, X. He, F. Yuan, Q. Tian, and T.-S. Chua, "Adversarial training towards robust multimedia recommender system," *IEEE Trans. on Knowledge and Data Engineering*, vol. 32, no. 5, pp. 855-867, 2020.

[12] P. Yang, J. T. Ormerod, W. Liu, C. Ma, A. Y. Zomaya, and J. Y. H. Yang, "AdaSampling for positive-unlabeled and label noise learning with bioinformatics applications," *IEEE Trans. on Cybernetics*, vol. 49, no. 5, pp. 1932-1943, 2018.

[13] Y. Zhong, L. Jin, M. Shang, and X. Luo, "Momentum-incorporated symmetric non-negative latent factor models," *IEEE Trans. on Big Data*, vol. 8, no. 4, pp. 1096-1106, 2022.

[14] Z. Liu, G. Yuan, and X. Luo, "Symmetry and nonnegativity-constrained matrix factorization for community detection," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 9, pp. 1691-1693, 2022.

[15] Y. Su, C. Liu, Y. Niu, F. Cheng and X. Zhang, "A community structure enhancement-based community detection algorithm for complex networks," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 5, pp. 2833-2846, 2021.

[16] X. Luo, Z. Liu, S. Li, M. Shang and Z. Wang, "A fast nonnegative latent factor model based on generalized momentum method," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 1, pp. 610-620, 2021.

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

[18] G. E. Moon, J. A. Ellis, A. Sukumaran-Rajam, S. Parthasarathy, and P. Sadayappan, "ALO-NMF: accelerated locality-optimized non-negative matrix factorization," in *Proc. of the 26th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Virtual Event, USA, 2020, pp. 1758-1767.

[19] Y. Qian, C. Tan, D. Ding, H. Li, and N. Mamoulis, "Fast and secure distributed nonnegative matrix factorization," *IEEE Trans. on Knowledge and Data Engineering*, vol. 34, no. 2, pp. 653-666, 2022.

[20] V. Leplat, N. Gillis, and A. M. S. Ang, "Blind audio source separation with minimum-volume beta-divergence NMF," *IEEE Trans. on Signal Processing*, vol. 68, pp. 3400-3410, 2020.

[21] D. Hajinezhad, T. Chang, X. Wang, Q. Shi, and M. Hong, "Nonnegative matrix factorization using ADMM: algorithm and convergence analysis," in *Proc. of the 41th IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Shanghai, China, 2016, pp. 4742-4746.

[22] X. Luo, M. Zhou, S. Li, L. Hu, and M. Shang, "Non-negativity constrained missing data estimation for high-dimensional and sparse matrices from industrial applications," *IEEE Trans. on Cybernetics*, vol. 50, no. 5, pp. 1844-1855, 2020.

[23] D. D. Lee, and H. S. Seung, "Algorithms for non-negative matrix factorization," *Advances in Neural Information Processing Systems*, vol. 13, no. 2000, pp. 556-562, 2001.

[24] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "AutoRec: autoencoders meet collaborative filtering," in *Proc. of the 24th Int. Conf. on World Wide Web*, Florence, Italy, 2015, pp. 111–112.

[25] C. Lin, Z. Cao, and M. Zhou, "Autoencoder-embedded iterated local search for energy-minimized task schedules of human–cyber–physical systems," *IEEE Trans. on Automation Science and Engineering*, DOI: 10.1109/TASE.2023.3267714, 2023.

[26] H. Yuan, Q. Hu, J. Bi, J. Lü, J. Zhang, and M. Zhou, "Profit-optimized computation offloading with autoencoder-assisted evolution in large-scale mobile edge computing," *IEEE Internet of Things Journal*, DOI: 10.1109/JIOT.2023.3244665, 2023.

[27] Y. Wu, C. DuBois, A. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-N recommender systems," in *Proc. of the 9th ACM Int. Conf. on Web Search and Data Mining*, San Francisco, USA, 2016, pp. 153-162.

[28] H. Fan, F. Zhang, Y. Wei, Z. Li, C. Zou, Y. Gao, and Q. Dai, "Heterogeneous hypergraph variational autoencoder for link prediction," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 44, no. 8, pp. 4125-4138, 2022.

[29] J. Jiang, W. Li, A. Dong, Q. Gou, and X. Luo, "A fast deep autoencoder for high-dimensional and sparse matrices in recommender systems," *Neurocomputing*, vol. 412, pp. 381-391, 2020.

[30] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. of the 26th Int. Conf. on World Wide Web*, Perth, Australia, 2017, pp. 173-182.

[31] T. Ebesu, B. Shen, and Y. Fang, "Collaborative memory network for recommendation systems," in *Proc. of the 41st Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Ann Arbor, USA, 2018, pp. 515-524.

[32] H. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems," in *Proc. of the 26th Int. Joint Conf. on Artificial Intelligence*, Melbourne, Australia, 2017, pp. 3203-3209.

[33] Z. Cheng, Y. Ding, X. He, L. Zhu, X. Song, and M. Kankanhalli, "A3NCF: an adaptive aspect attention model for rating prediction," in *Proc. of the 27th Int. Joint Conf. on Artificial Intelligence*, Stockholm, Sweden, 2018, pp. 3748-3754.

[34] X. He, X. Du, X. Wang, F. Tian, J. Tang, and T.-S. Chua, "Outer product-based neural collaborative filtering," in *Proc. of the 27th Int. Joint Conf. on Artificial Intelligence*, Stockholm, Sweden, 2018, pp. 2227-2233.

[35] X. Wang, X. He, M. Wang, F. Feng, T.-S. Chua, "Neural graph collaborative filtering," in *Proc. of the 42nd Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Paris, France, 2019, pp. 165-174.

[36] P. Han, P. Yang, P. Zhao, S. Shang, Y. Liu, J. Zhou, X. Gao, and P. Kalnis, "GCN-MF: disease-gene association identification by graph convolutional networks and matrix factorization," in *Proc. of the 25th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Anchorage, USA, 2019, pp. 705-713.

[37] X. He, K. Deng, X. Wang, Y. li, Y. Zhang, and M. Wang, "LightGCN-simplifying and powering graph convolution network for recommendation," in *Proc. of the 43rd Int. ACM SIGIR conf. on research and development in Information Retrieval*, Xi'an, China, 2020, pp. 639-648.

[38] K. Mao, J. Zhu, X. Xiao, B. Lu, Z. Wang, and X. He, "UltraGCN: ultra simplification of graph convolutional networks for recommendation," in *Proc. of the 30th ACM Int. Conf. on Information and Knowledge Management*, Queensland, Australia, 2021, pp. 1253-1262.

[39] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie, "Self-supervised graph learning for recommendation," in *Proc. of the 44th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Virtual Event, Canada, 2021, pp. 726-735.

[40] W. Fan, X. Liu, W. Jin, X. Zhao, J. Tang, and Q. Li, "Graph trend filtering networks for recommendation," in *Proc. of the 45th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Madrid, Spain, 2022, pp. 112-121.

[41] L. Xia, C. Huang, Y. Xu, J. Zhao, D. Yin, and J. X. Huang, "Hypergraph contrastive collaborative filtering," in *Proc. of the 45th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Madrid, Spain, 2022, pp. 70-79.

[42] Y. Lin, P. Ren, Z. Chen, Z. Ren, D. Yu, J. Ma, M. Rijke, and X. Cheng, "Meta matrix factorization for federated rating predictions," in *Proc. of the 43rd Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Xi'an, China, 2020, pp. 981-990.

[43] Y. Zheng, B. Tang, W. Ding, H. Zhou, "A neural autoregressive approach to collaborative filtering," in *Proc. of the 33rd Int. Conf. on Machine Learning*, New York, USA, 2016, pp. 764-773.

[44] X. Luo, H. Wu, Z. Wang, J. Wang, and D. Meng, "A novel approach to large-scale dynamically weighted directed network representation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 44, no. 12, pp. 9756-9773, 2022.

[45] Y. E. Salehani, E. Arabnejad, A. Rahiche, A. Bakhta, and M. Cheriet, "MSdB-NMF: multispectral document image binarization framework via non-negative matrix factorization approach," *IEEE Trans. on Image Processing*, vol. 29, pp. 9099-9112, 2020.

[46] Z. Ren, W. Zhang, and Z. Zhang, "A deep nonnegative matrix factorization approach via autoencoder for nonlinear fault detection," *IEEE Trans. on Industrial Informatics*, vol. 16, no. 8, pp. 5042-5052, 2020.

[47] F. Ye, C. Chen, and Z. Zheng, "Deep autoencoder-like nonnegative matrix factorization for community detection," in *Proc. of the 27th ACM Int. Conf. on Information and Knowledge Management*, Torino, Italy, pp. 1393-1402, 2018.

[48] M. Yang, and S. Xu, "Orthogonal nonnegative matrix factorization using a novel deep autoencoder network," *Knowledge-Based Systems*, vol. 227, no. 107236, 2021.

[49] H. Huang, Y. Luo, G. Zhou, and Q. Zhao, "Multi-view data representation via deep autoencoder-like nonnegative matrix factorization," in *Proc. of the 2022 IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Singapore, Singapore, 2022, pp. 3338-3342.

[50] L. Zhao, Z. Wang, Z. Wang, and Z. Chen, "Multi-view graph regularized deep autoencoder-like NMF framework," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Rhodes Island, Greece, 2023, pp. 1-5.

[51] L. Dong, Y. Yuan, and X. Luxs, "Spectral–spatial joint sparse NMF for hyperspectral unmixing," *IEEE Trans. on Geoscience and Remote Sensing*, vol. 59, no. 3, pp. 2391-2402, 2021.

[52] D. P. Kingma, and J. Ba, "Adam: a method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[53] D. Liang, R. G. Krishnan, M. D. Hoffman, T. Jebara, "Variational autoencoders for collaborative filtering," in *Proc. of the 27th Int. Conf. on World Wide Web*, Lyon, France, 2018, pp. 689-698.

[54] M. Lee, S. Song, J. Moon, J. Kim, W. Seo, Y. Cho, and S. Ryu, "Improving GPGPU resource utilization through alternative thread block scheduling," in *Proc. of the 20th IEEE Int. Symposium on High Performance Computer Architecture*, Orlando, USA, 2014, pp. 260-271.

[55] Y.-Y. Jo, S.-W. Kim, and D.-H. Bae, "Efficient sparse matrix multiplication on GPU for large social network analysis," in *Proc. of the 24th ACM Int. on Conf. on Information and Knowledge Management*, Melbourne, Australia, 2015, pp. 1261-1270.

[56] J. Lee, S. Kang, Y. Yu, Y.-Y. Jo, S.-W. Kim, and Y. Park, "Optimization of GPU-based sparse matrix multiplication for large sparse networks," in *Proc. of the 36th IEEE Int. Conf. on Data Engineering*, Dallas, USA, 2020, pp. 925-936.

[57] S. Pal, J. Beaumont, D.-H. Park, A. Amarnath, S. Feng, C. Chakrabarti, H.-S. Kim, D. Blaauw, T. N. Mudge, and R. G. Dreslinski, "Outerspace: an outer product based sparse matrix multiplication accelerator," in *Proc. of the 24th IEEE Int. Symposium on High Performance Computer Architecture*, Vienna, Austria, 2018, pp. 724-736.

[58] 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.

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

[60] D. Szklarczyk, A. L. Gable, D. Lyon, A. Junge, S. Wyder, J. HuertaCepas, M. Simonovic, N. T. Doncheva, J. H. Morris, P. Bork, L. J. Jensen, and C. Mering, "STRING v11: protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets," *Nucleic Acids Research*, vol. 47, pp. D607-D613, 2019.

[61] R. Berg, T. N. Kipf, and W. Max, "Graph convolutional matrix completion," in *Proc. of the 24th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, London, UK, 2018.

[62] L. Chen, L. Wu, R. Hong, K. Zhang, and M. Wang, "Revisiting graph based collaborative filtering: a linear residual graph convolutional network approach," in *Proc. of the 34th AAAI Conf. on Artificial Intelligence*, New York, USA, 2020.

[63] W. Guo, Y. Yang, Y. Hu, C. Wang, H. Guo, Y. Zhang, R. Tang, W. Zhang, and X. He, "Deep graph convolutional networks with hybrid normalization for accurate and diverse recommendation", in *Proc. of 3rd Workshop on Deep Learning Practice for High-Dimensional Sparse Data with KDD*, Virtual Event, China, 2021.

[64] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie, "Self-supervised graph learning for recommendation," in *Proc. of the 44th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Virtual Event, Canada, 2021, pp. 726-735.

[65] M. Zhao, L. Wu, Y. Liang, L. Chen, J. Zhang, Q. Deng, K. Wang, X. Shen, T. Lv, and R. Wu, "Investigating accuracy-novelty performance for graph-based collaborative filtering," in *Proc. of the 45th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Madrid, Spain, 2022, pp. 50-59.

[66] T. Kong, T. Kim, J. Jeon, J. Choi, Y.-C. Lee, N. Park, and S.-W. Kim, "Linear, or non-linear, that is the question," in *Proc. of the 15th ACM Int. Conf. on Web Search and Data Mining*, Tempe, USA, 2022, pp. 517-525.

[67] X. Glorot, and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. of the 13th Int. Conf. on Artificial Intelligence and Statistics,* Sardinia, Italy, 2010, pp. 249-256.

[68] H. Wu, X. Luo, M. Zhou, M. J. Rawa, K. Sedraoui, and A. Albeshri, "A PID-incorporated latent factorization of tensors approach to dynamically weighted directed network analysis," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 3, pp. 533-546, 2022.

[69] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1-30, 2006.

[70] X. Luo, Z. Liu, M. Shang, J. Lou, and M. Zhou, "Highly-accurate community detection via pointwise mutual information-incorporated symmetric non-negative matrix factorization," *IEEE Trans. on Network Science and Engineering*, vol. 8, no. 1, pp. 463-476, 2021.

[71] D. Wu, X. Luo, Y. He, and M. Zhou, "A prediction-sampling-based multilayer-structured latent factor model for accurate representation of high-dimensional and sparse data," *IEEE Trans. on Neural Networks and Learning Systems*, 10.1109/TNNLS.2022.3200009, 2022.

[72] S. Boyd, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1-122, 2010.

[73] X. Luo, Y. Yuan, S. Chen, N. Zeng, and Z. Wang, "Position-transitional particle swarm optimization-incorporated latent factor analysis," *IEEE Trans. on Knowledge and Data Engineering*, vol. 34, no. 8, pp. 3958-3970, 2022.

[74] Q. Ge, X. Hu, Y. Li, H. He, and Z. Song, "A novel adaptive Kalman filter based on credibility measure," *IEEE/CAA Journal of Automatica Sinica*, vol. 10, no. 1, pp. 103-120, 2023.

[75] P. Zhang, W. Huang, Y. Chen, and M. Zhou, "Predicting quality of services based on a two-stream deep learning model with user and service graphs," *IEEE Trans. on Services Computing*, DOI: 10.1109/TSC.2023.3303191, 2023.

[76] F. Bi, T. He, Y. Xie, and X. Luo, "Two-stream graph convolutional network-incorporated latent feature analysis," *IEEE Trans. on Services Computing*, vol. 16, no. 4, pp. 3027-3042, 2023.

[77] C. He, Y. Zheng, X. Fei, H. Li, Z. Hu, and Y. Tang, "Boosting nonnegative matrix factorization based community detection with graph attention auto-encoder," *IEEE Trans. on Big Data*, vol. 8, no. 4, pp. 968-981, 2022.

[78] Y. Chen, L. Huang, C. Wang, and J. Lai, "Hybrid-order gated graph neural network for session-based recommendation," *IEEE Trans. on Industrial Informatics*, vol. 18, no. 3, pp. 1458-1467, 2022.

[79] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: a survey," *IEEE Trans. on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 249-270, 2022.

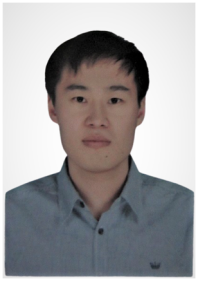
[80] M. Cui, L. Li, M. Zhou, and A. Abusorrah, "Surrogate-assisted autoencoder-embedded evolutionary optimization algorithm to solve high-dimensional expensive problems," *IEEE Trans. on Evolutionary Computation*, vol. 26, no. 4, pp. 676-689, 2022.

[81] M. Cui, L. Li, M. Zhou, J. Li, A. Abusorrah, and K. Sedraoui, "A bi-population cooperative optimization algorithm assisted by an autoencoder for medium-scale expensive problems," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 11, pp. 1952-1966, 2022.

[82] J. Gao, J. Gao, X. Ying, M. Lu, and J. Wang, "Higher-order interaction goes neural: a substructure assembling graph attention network for graph classification," *IEEE Trans. on Knowledge and Data Engineering*, vol. 35, no. 2, pp. 1594-1608, 2023.

[83] Y. Xie, Z. Xu, J. Zhang, Z. Wang, and S. Ji, "Self-supervised learning of graph neural networks: a unified review," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 45, no. 2, pp. 2412-2429, 2023.

Fanghui Bi received the B.S. degree in Software Engineering from the Northeastern University, Shenyang, China, in 2020. He is currently pursuing his Ph.D. degree in computer science at the University of Chinese Academy of Sciences, Beijing, China. His research interests include Big Data Analytics and Graph Neural Networks.



Tiantian He (*Member*, IEEE) received the Ph.D. degree in Computer Science from the Hong Kong Polytechnic University in 2017. Currently He is a Research Scientist in Center for Frontier AI Research (CFAR), Institute of High Performance Computing (IHPC), Agency for Science, Technology and Research (A*STAR). His research interests include artificial intelligence, machine learning, data-centric transfer optimization, and data mining.

Xin Luo (*Senior Member*, IEEE) 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 the Beihang University, Beijing, China, in 2011. He is currently a Professor of Data Science and Computational Intelligence with the College of Computer and Information Science, Southwest University, Chongqing, China. He has authored or coauthored over 200 papers (including over 110 IEEE Transactions papers) in the areas of his interests. His research interests focus on big data analysis and graph learning. Dr. Luo was the recipient of the Outstanding Associate Editor Award from IEEE/CAA JOURNAL OF AUTOMATICA SINICA in 2020. He is currently serving as a Deputy-Editor-in-Chief for IEEE/CAA JOURNAL OF AUTOMATICA SINICA, and an Associate Editor for IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS. His Google page: https://scholar.google.com/citations?user=hyGIDs4AAAAJ&hl=zh-TW.