

Adaptive PID-Incorporated Nonnegative Latent Factor Analysis

Jinli Li, Ye Yuan, *Member, IEEE*, Tiantian He, *Member, IEEE*, and Xin Luo, *Fellow, IEEE*

Abstract—High-dimensional and incomplete (HDI) data are ubiquitous for representing the intricate interactions among a vast number of nodes arising from diverse real application scenarios. Non-negative latent factor (NLF) models have demonstrated their effectiveness in extracting critical latent features from HDI data by leveraging a single latent factor-dependent, non-negative, and multiplicative update (SLF-NMU) algorithm. However, the SLF-NMU algorithm always leads NLF models to converge sluggishly as it updates a latent factor based only on the current updated information. To address this critical issue, we present APNLF, which innovatively adopts an adaptive proportional-integral-derivative (PID) controller to enable the learning process of SLF-NMU to be more efficient. The proposed APNLF encompasses the following two-fold ideas: a) establishing a PID-increment-based SLF-NMU (PSN) algorithm, which updates a latent factor by comprehensively modeling the current, past, and future update increment information guided by the principle of a PID controller, and b) designing an effective fuzzy reasoning rule to implement all hyper-parameters adaptation, which boosts model's applicability in practical applications. Moreover, APNLF's convergence analysis is provided in theory. Experiments on five HDI datasets demonstrate that the APNLF model outperforms the state-of-the-art models in efficiency and accuracy on an HDI matrix. The source code is available at <https://github.com/Aaaaapple/APNLF-Development>.

Index Terms—High-Dimensional and Incomplete Data, Non-negative Latent Factor Analysis, Proportional Integral Derivative, Fuzzy Reasoning Rule.

I. INTRODUCTION

VARIOUS real data from diverse industrial domains related contain plentiful nodes and non-negative coactive relationships between them [1, 3, 14, 15, 18], e.g., the user ratings on items in a recommender system [5, 10], user invocations on services in cloud computing [38], protein interactions in the biological network [32, 50], and user trust relationships in the social network [49, 59]. Due to the explosive growth of data, acquiring all interactions among them has become an impossibility, which means a node only contacts a very tiny number of nodes. Consequently, the resulting non-negativity interactive relationships are mostly described by a high-dimensional and incomplete (HDI) matrix [7, 16], whose high dimension and high incomplete characteristics correspond to a vast number of nodes and the interactions between nodes.

This study was supported by the National Natural Science Foundation of China under grants 62372385, and 62272078, and the Chongqing Natural Science Foundation under Grant CSTB2023NSCQ-LZX0069. (*Corresponding Author: Ye Yuan.*)

J. L. Li is with the School of Computer and Software Engineering, Xihua University, Chengdu 610039, China (email: appleli_li@163.com).

Y. Yuan and X. Luo are with the College of Computer and Information Science, Southwest University, Chongqing 400715, China (email: yuanyekl@gmail.com, luoxin@ieee.org).

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 (email: he_tiantian@cfor.a-star.edu.sg).

A large proportion of the entries in the HDI matrix are undiscovered. However, it involves abundant study-worthy known entries involving user potential preferences [4] and community structure [59]. Hence, learning precise representation of HDI data contributes to various analysis tasks in real application scenarios [13, 32]. In recent years, latent factor (LF) models have exhibited significant potential in representing an HDI matrix efficiently [3, 6, 13, 19, 34–36] since they depend on the known terms in the HDI matrix only. Among the popular LF models, a non-negative latent factor (NLF) model is the representative one [13, 20, 41]. The main reason is that an NLF model utilizes the single latent factor-dependent non-negative and multiplicative update (SLF-NMU) algorithm to eliminate negative terms and obtains better performance. However, existing NLF models encounter the issue of slow convergence [41, 42] when dealing with an HDI matrix since the update process of SLF-NMU depends upon the current update increment information merely [3, 20]. Therefore, it is significant to accelerate the NLF model's convergence rate via integrating other information, i.e., past and future update increments, based on current update increments.

Previous studies [11, 21, 47, 51] have shown that a proportional-integral-derivative (PID) controller helps obtain an efficient and stable control of the dynamic process based on current, historical, and future trend information. Specifically, it effectively obtains a refinement state by combining the proportional, integral, and derivative terms to incorporate the current, historical, and future trend information. Due to its availability, simplicity, and stability [23, 24], a PID controller has started to be applied to machine learning models. For instance, Ma *et al.* [22] propose a meta PID attention network (MPA-Net), which facilitates network training and boosts denoising performance. Zhou *et al.* [33] propose a PID-based de-raining network, which efficiently explores and exploits the features of rain streaks recursively. Li *et al.* [43] propose a self-adaption non-linear PID incorporated LFA model, which obtains remarkable improvement.

Motivated by the previously mentioned discoveries, in this paper, we hypothesize that leveraging the PID controller to fuse the current, past, and future update increment information into SLF-NMU's learning process can improve the performance of the NLF model. To this end, we present APNLF, which contains the following two-fold steps:

- building a PID-increment-based SLF-NMU (PSN) algorithm, which obtains a refactored variable by comprehensively modeling the current, historical, and future trend update increment information guided by the mechanism of a PID to update target latent factor; and
- designing an effective fuzzy reasoning rule to implement the parameter's self-adaptation in the proposed above method, which boosts the model's applicability in prac-

tical applications.

By accomplishing the above points, the contributions of this research are provided below:

- a) **Proposal of an APNLF model for efficient HDI data processing.** APNLF innovatively introduces the current, historical, and future trend increments to refine the instant increment following the principle of a PID controller, thereby achieving high-efficiency non-negative latent factor analysis on diverse HDI data. In addition, it establishes a reasonable fuzzy reasoning rule to realize hyperparameter self-adaptation, which significantly enhances the APNLF's practicality and feasibility in real-world industrial scenarios;
- b) **Theoretical convergence analysis of an APNLF model.** It contains a rigorous theoretical convergence analysis, which provides a solid mathematical foundation for the proposed APNLF model;
- c) **Design of the APNLF algorithm and experimental validation.** It presents a specific implementation algorithm of APNLF and perform a quantitative analysis of its computational complexity. Moreover, it executes comprehensive experiments on real industrial HDI data validate that APNLF significantly outperforms existing baselines in both accuracy and efficiency.

Section II depicts the related works. Section III gives the preliminaries. Section IV presents the methodology. Section V provides the experimental results and analyses. Finally, Section VI concludes.

II. RELATED WORK

A. Latent Factor Models

The pursuit of valuable information from the HDI matrix has motivated the development of numerous LFA models [2, 4, 6, 8–10, 17, 25, 26, 28]. For instance, GNSLF [2] utilizes a generalized Nesterov's method to accelerate second-order model. ADLF [4] adds the exponentially decaying average of the previous state of gradients to improve the convergence rate. PPL [6] introduces the adaptive moment estimation in pseudo gradients and optimizes particle's positions based on pseudo gradients. IN-CoLFA [8] introduces co-factorization technique and neural network-inspired structure to improve analysis results. LFM [9] constructs multi-relation likelihoods estimation and signed pairwise ranking to efficiently spam detection tasks. MBLFE [10] builds a gating expert network to models all latent factors within entire scenario to enables a more accurate representation of user preferences. ANP [17] connects the feedback control and neural network to design a dynamic learning process. IPSLF [25] builds the relationship between training loss and learning error based on feedback control mechanism to obtain higher accuracy. EDG-PSO [26] updates the directions of particles by their gradient's changes and introduces Adam to enhance the search efficiency. PSGO [28] analyzes the relationship between the SGD-momentum and PID control, and updates the network parameters by PID, thereby reducing the overshoot and improving the convergence accuracy. Compared with the existing LFA models,

the proposed APNLF incorporates data non-negativity constraints during training process. It not only preserves the interpretability of feature representations but also aligns more closely with the non-negative nature of data in real-world scenarios. Therefore, it significantly enhances the model's representational capacity and generalization performance.

B. Nonnegative Latent Factor Models

The NLF models have garnered significant investigation due to its inherent data non-negativity [3, 11, 12, 18, 19, 31, 34, 41, 57]. For instance, SNLF [3] leverages L_2 -norm regularization and the Lagrangian multiplier to resolve overfitting and employs the extragradient-based alternating direction method to accelerate the model training. CB-NLFT [11] utilizes a novel convolution bias scheme to obtain more precious instantaneous data fluctuations. INLFM [12] proposes a target customer estimation model based on customer portrait label system and a dual-layer affinity propagation method to facilitate precise ERP recommendation. SNMCF [18] integrates students' knowledge states and their exercising process to more accurately predict their exercise performance. EBNL [19] integrates multi-linear biases into the NLF model and adopts PSO algorithm to achieve better estimation accuracy. RTOSNMF [31] separates noise from raw adjacency matrices by linear separation and utilizes L_2 norm constraint to achieve denoising. PINLF [34] aggregates historical information to accelerate the convergence process, thereby enhancing the model's prediction performance. FNLf [41] establishes a generalized momentum method with SLF-NMU for a superior convergence rate and prediction accuracy for missing data. RNMH-NMF [57] builds a hypergraph regularization to discover the higher-order relationship between instances and utilizes a pair-wise consistency learning term to mine consistency information in multi-view data. Compared with the above NLF models, the proposed APNLF proposes a novel training mechanism by obtaining the refinement increment following the principle of a PID controller and design an efficient parameter adaption strategy, which flexibly exploits the current, historical, and future trend of increments during the training process.

III. PRELIMINARIES

A. Problem Formulation

To appropriately formulate the concerned problem, we firstly present the following two definitions:

Definition 1 (An HDI Matrix): M and N are two large node sets denoting user and item sets, respectively. In the objective matrix $R \in \mathbf{R}^{|M| \times |N|}$, each entry $r_{m,n}$ represents the preference of node m on node n [4, 9]. Similarly, the subsets Λ and Γ represent the known and unknown entity sets of R , respectively. If $|\Lambda| \ll |\Gamma|$, R denotes an HDI matrix [16].

Definition 2 (An NLF Model): An NLF model builds a rank- f approximation $\hat{R} = XY^T$ to R relying on Λ , where latent feature matrices $X^{|M| \times f}$ and $Y^{|N| \times f} \geq 0$, and $f \ll \{|M|, |N|\}$. To achieve the target, the Euclidean distance is used to build the loss function [13, 20, 34, 42].

For solving the overfitting problem, the loss function described by Euclidean distance with the regularization terms is shown as Eq. (1):

$$\varepsilon(X, Y) \triangleq \sum_{r_{m,n} \in \Lambda} \left(\left(r_{m,n} - \sum_{d=1}^f x_{m,d} y_{n,d} \right)^2 + \lambda \left(\sum_{d=1}^f x_{m,d}^2 + \sum_{d=1}^f y_{n,d}^2 \right) \right),$$

$$s.t. \forall m \in M, n \in N, d \in \{1, 2, \dots, f\} : x_{m,d} \geq 0, y_{n,d} \geq 0, \quad (1)$$

where $\varepsilon(X, Y)$ denotes the objective function defined over the factor matrices X and Y , $x_{m,d}$, $y_{n,d}$ and $r_{m,n}$ are correspondingly single entries of X , Y and R , $\hat{r}_{m,n} = \sum_{d=1}^f x_{m,d} y_{n,d}$ is the prediction of the corresponding $r_{m,n}$, and λ is the coefficient of the regularization term, respectively.

B. An SLF-NMU-based NLF Model

As discussed in previous studies [13, 34], an SLF-NMU algorithm is commonly utilized to learn the desired non-negative LFs efficiently. To do so, given an optimization parameter θ , applying the optimization process with the additive gradient descent (AGD) and eliminating the negative terms via manipulating the learning rate, the Eq.(2) can be obtained:

$$\hat{\theta}_t = \underset{\theta}{\operatorname{argmin}} \quad \varepsilon(\theta_{t-1}), \quad (2)$$

where $\hat{\theta}_t$ is expected state of SLF-NMU at the t -th iteration.

According to Eq.(2), the X_t and Y_t are obtained as follows:

$$\arg \min \varepsilon(X, Y) \xrightarrow{SLF-NMU} \begin{cases} x_{m,d} \leftarrow x_{m,d} - \frac{\sum_{n \in \Lambda(m)} y_{n,d} r_{m,n}}{\sum_{m \in \Lambda(n)} y_{n,d} \hat{r}_{m,n} + \lambda |\Lambda(m)| x_{m,d}}, \\ y_{n,d} \leftarrow y_{n,d} - \frac{\sum_{m \in \Lambda(n)} x_{m,d} r_{m,n}}{\sum_{m \in \Lambda(n)} x_{m,d} \hat{r}_{m,n} + \lambda |\Lambda(n)| y_{n,d}}. \end{cases} \quad (3)$$

where $\Lambda(m)$ and $\Lambda(n)$ are the subsets of Λ corresponding to $m \in M$ and $n \in N$, respectively.

C. A PID Controller

The standard PID [25, 27] controller utilizes the current systematic error, historical systematic errors, and future trend systematic error (named as the proportional, integral and derivative terms, respectively) to compute a more consummate systematic feedback error [30]. It has been extensively applied in many real industries owing to its simplicity and reliability. The discrete PID is formulated as follows:

$$\tilde{e}^{(t)} = K_P e^{(t)} + K_I \sum_{k=0}^t e^{(k)} + K_D (e^{(t)} - e^{(t-1)}), \quad (4)$$

where t represents the time node in the control system; $e^{(t)}$ represents the systematic error in time node t ; $\tilde{e}^{(t)}$ represents the refined systematic feedback error corresponding to $e^{(t)}$; K_P , K_I , K_D represent the gain-parameters of proportional, integral and derivative terms, respectively.

IV. THE PROPOSED APNLF MODEL

A. The PSN Algorithm

According to prior studies [44, 52], the systematic control error stands for the value between the predicted and desired values, which can be efficiently eliminated by a PID controller. Analogously, an NLF model minimizes its objective function via the SLF-NMU algorithm [3, 13], where we can see the probability to associate a PID controller with its optimization process. To do so, Eqs. (2) and (3) illustrate that the SLF-NMU algorithm relies explicitly on the update increment rather than the gradient to update the optimization parameter in the update process since it eliminates the negative terms to ensure non-negative latent features [34]. Hence, the corresponding update increment based on Eq. (2) is formulated as:

$$\Delta_t = \hat{\theta}_t - \theta_{t-1}. \quad (5)$$

It should be pointed out that if the update increment is small enough, the optimization parameter is no longer updated. Therefore, it is reasonable to consider the update increment in SLF-NMU as the control error of a PID controller. Consequently, the following refined update increment can be obtained following the mechanism of a PID controller:

$$\tilde{\Delta}_t = K_P \Delta_t + K_I \sum_{j=0}^t \Delta_j + K_D (\Delta_t - \Delta_{t-1}), \quad (6)$$

where $\tilde{\Delta}_t$ represents the refined update increment related to Δ_t . From Eq. (6), we see that $\tilde{\Delta}_t$ represents the current, past, and future update increment information, and Δ_t models the current update increment information only. Therefore, the Eq. (7) is acquired by combining (5) and (6):

$$\theta_t = \theta_{t-1} + \left(K_P (\hat{\theta}_t - \theta_{t-1}) + K_I \sum_{j=0}^t (\hat{\theta}_j - \theta_{j-1}) + K_D ((\hat{\theta}_t - \theta_{t-1}) - (\hat{\theta}_{t-1} - \theta_{t-2})) \right). \quad (7)$$

In an NLF model, the optimization parameter consists of latent feature matrices X and Y . Hence, according to (2), the following equation can be obtained:

$$(\hat{X}_t, \hat{Y}_t) = \underset{X, Y}{\operatorname{argmin}} \quad \varepsilon(X_{t-1}, Y_{t-1}), \quad (8)$$

where X_{t-1} and Y_{t-1} represent the state of X and Y in the $(t-1)$ -th iteration, $\hat{X}_t$ and $\hat{Y}_t$ denote the expected states in the t -th iteration that are computed by SLF-NMU.

Therefore, the increment between t and $(t-1)$ -th iteration is computed with the following rule:

$$\Delta_t = \hat{\theta}_t - \theta_{t-1} = \begin{bmatrix} \hat{X}_t \\ \hat{Y}_t \end{bmatrix} - \begin{bmatrix} X_{t-1} \\ Y_{t-1} \end{bmatrix}. \quad (9)$$

As shown in (9), the increment is the differentiative. Afterwards, the equation for the PID with increment as the refined target is obtained by substituting (9) into (6):

$$\begin{aligned} \tilde{\Delta}_t = & K_P \left(\begin{bmatrix} \hat{X}_t \\ \hat{Y}_t \end{bmatrix} - \begin{bmatrix} X_{t-1} \\ Y_{t-1} \end{bmatrix} \right) + K_I \sum_{k=0}^t \left(\begin{bmatrix} \hat{X}_k \\ \hat{Y}_k \end{bmatrix} - \begin{bmatrix} X_{k-1} \\ Y_{k-1} \end{bmatrix} \right) \\ & + K_D \left(\left(\begin{bmatrix} \hat{X}_t \\ \hat{Y}_t \end{bmatrix} - \begin{bmatrix} X_{t-1} \\ Y_{t-1} \end{bmatrix} \right) - \left(\begin{bmatrix} \hat{X}_{t-1} \\ \hat{Y}_{t-1} \end{bmatrix} - \begin{bmatrix} X_{t-2} \\ Y_{t-2} \end{bmatrix} \right) \right). \end{aligned} \quad (10)$$

Note that if $t=1$, the special case that $t=-1$ will appear in Eq. (10). Since X_{-1} and Y_{-1} are equal to $\hat{X}_0$ and $\hat{Y}_0$, X_0 and Y_0 are actually the initial hypotheses of X and Y , so if $t=1$, (10) can be overwritten as follows:

$$\begin{aligned} \tilde{\Delta}_1 = & K_P \left(\begin{bmatrix} \hat{X}_1 \\ \hat{Y}_1 \end{bmatrix} - \begin{bmatrix} X_0 \\ Y_0 \end{bmatrix} \right) + K_I \left(\begin{bmatrix} \hat{X}_1 \\ \hat{Y}_1 \end{bmatrix} - \begin{bmatrix} X_0 \\ Y_0 \end{bmatrix} \right) \\ & + K_D \left(\begin{bmatrix} \hat{X}_1 \\ \hat{Y}_1 \end{bmatrix} - \begin{bmatrix} X_0 \\ Y_0 \end{bmatrix} \right). \end{aligned} \quad (11)$$

Evidently, the state $t = -1$ doesn't affect the update rule (12). Therefore, by combining (10) and (11), the increment refinement at the t -th iteration is given as:

$$\begin{aligned} \tilde{\Delta}_t = & K_P \left(\begin{bmatrix} \hat{X}_t \\ \hat{Y}_t \end{bmatrix} - \begin{bmatrix} X_{t-1} \\ Y_{t-1} \end{bmatrix} \right) + K_I \sum_{j=1}^t \left(\begin{bmatrix} \hat{X}_j \\ \hat{Y}_j \end{bmatrix} - \begin{bmatrix} X_{j-1} \\ Y_{j-1} \end{bmatrix} \right) \\ & + K_D \left(\begin{bmatrix} \hat{X}_t \\ \hat{Y}_t \end{bmatrix} - \begin{bmatrix} X_{t-1} \\ Y_{t-1} \end{bmatrix} - \begin{bmatrix} \hat{X}_{t-1} \\ \hat{Y}_{t-1} \end{bmatrix} - \begin{bmatrix} X_{t-2} \\ Y_{t-2} \end{bmatrix} \right). \end{aligned} \quad (12)$$

Combining (9) and (12), the Eq.(13) can be deduced:

$$\begin{aligned} \begin{bmatrix} X_t \\ Y_t \end{bmatrix} = & \begin{bmatrix} X_{t-1} \\ Y_{t-1} \end{bmatrix} + K_P \left(\begin{bmatrix} \hat{X}_t \\ \hat{Y}_t \end{bmatrix} - \begin{bmatrix} X_{t-1} \\ Y_{t-1} \end{bmatrix} \right) \\ & + K_I \sum_{j=1}^t \left(\begin{bmatrix} \hat{X}_j \\ \hat{Y}_j \end{bmatrix} - \begin{bmatrix} X_{j-1} \\ Y_{j-1} \end{bmatrix} \right) \\ & + K_D \left(\begin{bmatrix} \hat{X}_t \\ \hat{Y}_t \end{bmatrix} - \begin{bmatrix} X_{t-1} \\ Y_{t-1} \end{bmatrix} - \begin{bmatrix} \hat{X}_{t-1} \\ \hat{Y}_{t-1} \end{bmatrix} - \begin{bmatrix} X_{t-2} \\ Y_{t-2} \end{bmatrix} \right). \end{aligned} \quad (13)$$

The update scheme of X_t is computed as (14):

$$\begin{aligned} x_{m,d}^{(t)} = & x_{m,d}^{(t-1)} + K_P \left(\frac{x_{m,d}^{(t-1)} \sum_{n \in \Lambda(m)} y_{n,d}^{(t-1)} r_{m,n}}{\sum_{n \in \Lambda(m)} y_{n,d}^{(t-1)} \hat{r}_{m,n}^{(t-1)} + \lambda |\Lambda(m)| x_{m,d}^{(t-1)}} - x_{m,d}^{(t-1)} \right) \\ & + K_I \sum_{j=1}^t \left(\frac{x_{m,d}^{(j-1)} \sum_{n \in \Lambda(m)} y_{n,d}^{(j-1)} r_{m,n}}{\sum_{n \in \Lambda(m)} y_{n,d}^{(j-1)} \hat{r}_{m,n}^{(j-1)} + \lambda |\Lambda(m)| x_{m,d}^{(j-1)}} - x_{m,d}^{(j-1)} \right) \\ & + K_D \left\{ \left(\frac{x_{m,d}^{(t-1)} \sum_{n \in \Lambda(m)} y_{n,d}^{(t-1)} r_{m,n}}{\sum_{n \in \Lambda(m)} y_{n,d}^{(t-1)} \hat{r}_{m,n}^{(t-1)} + \lambda |\Lambda(m)| x_{m,d}^{(t-1)}} \right) - \right. \\ & \left. \left(\frac{x_{m,d}^{(t-2)} \sum_{n \in \Lambda(m)} y_{n,d}^{(t-2)} r_{m,n}}{\sum_{n \in \Lambda(m)} y_{n,d}^{(t-2)} \hat{r}_{m,n}^{(t-2)} + \lambda |\Lambda(m)| x_{m,d}^{(t-2)}} \right) \right\}. \end{aligned} \quad (14)$$

By comparing (3) with (14), we see that (3) is the special case of (14) when $K_P=1$ and $K_I=K_D=0$. This study centers on the effect of past and future updates, i.e., integral and derivative terms. To ensure the non-negativity, the integral and derivative in $x_{m,d}^{(t)}$ and $y_{n,d}^{(t)}$ are projected to be zero if they become negative. Thus, (14) are extended:

$$\begin{aligned} x_{m,d}^{(t)} = & \frac{x_{m,d}^{(t-1)} \sum_{n \in \Lambda(m)} y_{n,d}^{(t-1)} r_{m,n}}{\sum_{n \in \Lambda(m)} y_{n,d}^{(t-1)} \hat{r}_{m,n}^{(t-1)} + \lambda |\Lambda(m)| x_{m,d}^{(t-1)}} \\ & + \max \left\{ 0, K_I \sum_{j=1}^t \left(\frac{x_{m,d}^{(j-1)} \sum_{n \in \Lambda(m)} y_{n,d}^{(j-1)} r_{m,n}}{\sum_{n \in \Lambda(m)} y_{n,d}^{(j-1)} \hat{r}_{m,n}^{(j-1)} + \lambda |\Lambda(m)| x_{m,d}^{(j-1)}} - x_{m,d}^{(j-1)} \right) \right\} \\ & + \max \left\{ 0, K_D \left\{ \left(\frac{x_{m,d}^{(t-1)} \sum_{n \in \Lambda(m)} y_{n,d}^{(t-1)} r_{m,n}}{\sum_{n \in \Lambda(m)} y_{n,d}^{(t-1)} \hat{r}_{m,n}^{(t-1)} + \lambda |\Lambda(m)| x_{m,d}^{(t-1)}} \right) - \right. \right. \\ & \left. \left. \left(\frac{x_{m,d}^{(t-2)} \sum_{n \in \Lambda(m)} y_{n,d}^{(t-2)} r_{m,n}}{\sum_{n \in \Lambda(m)} y_{n,d}^{(t-2)} \hat{r}_{m,n}^{(t-2)} + \lambda |\Lambda(m)| x_{m,d}^{(t-2)}} \right) \right\} \right\}. \end{aligned} \quad (15)$$

Moreover, the Y_t can be derived in the same manner:

$$\begin{aligned} y_{n,d}^{(t)} = & \frac{y_{n,d}^{(t-1)} \sum_{m \in \Lambda(n)} x_{m,d}^{(t-1)} r_{m,n}}{\sum_{m \in \Lambda(n)} x_{m,d}^{(t-1)} \hat{r}_{m,n}^{(t-1)} + \lambda |\Lambda(n)| y_{n,d}^{(t-1)}} \\ & + \max \left\{ 0, K_I \sum_{j=1}^t \left(\frac{y_{n,d}^{(j-1)} \sum_{m \in \Lambda(n)} x_{m,d}^{(j-1)} r_{m,n}}{\sum_{m \in \Lambda(n)} x_{m,d}^{(j-1)} \hat{r}_{m,n}^{(j-1)} + \lambda |\Lambda(n)| y_{n,d}^{(j-1)}} - y_{n,d}^{(j-1)} \right) \right\} \\ & + \max \left\{ 0, K_D \left\{ \left(\frac{y_{n,d}^{(t-1)} \sum_{m \in \Lambda(n)} x_{m,d}^{(t-1)} r_{m,n}}{\sum_{m \in \Lambda(n)} x_{m,d}^{(t-1)} \hat{r}_{m,n}^{(t-1)} + \lambda |\Lambda(n)| y_{n,d}^{(t-1)}} \right) - \right. \right. \\ & \left. \left. \left(\frac{y_{n,d}^{(t-2)} \sum_{m \in \Lambda(n)} x_{m,d}^{(t-2)} r_{m,n}}{\sum_{m \in \Lambda(n)} x_{m,d}^{(t-2)} \hat{r}_{m,n}^{(t-2)} + \lambda |\Lambda(n)| y_{n,d}^{(t-2)}} \right) \right\} \right\}. \end{aligned} \quad (16)$$

Based on (15) and (16), the learning rule for the PID-increment-based SLF-NMU (PSN) algorithm is obtained.

B. Adaptation of Gain Parameters

As shown in (15) and (16), the PSN algorithm relies on the gain parameters K_I and K_D as well as the regularization coefficient λ , which are hard to adjust manually in diverse problems. To solve the above issue, section IV-B presents specially defined fuzzy reasoning rules to implement self-adaptation of gain parameters. Hence, this section presents an effective fuzzy reasoning rule to implement the hyper-parameter adaptation. As discussed in [29], designing a fuzzy reasoning rule consists of four steps, i.e., input, fuzzification, fuzzy reasoning, and defuzzification [30]. Next, a detailed introduction to the designed fuzzy reasoning rule is provided.

1) Input. When designing the fuzzy reasoning rule, the combination of the mean absolute error (MAE) and root mean squared error (RMSE) on the validation set (disjoint with the

TABLE I: FUZZY REASONING AND RELATIONSHIP.

$A^{(t)}$	A_1	A_2	A_3	A_4
λ	λ_1	λ_2	λ_3	λ_4
K_I	I_1	I_2	I_3	I_4
K_D	D_1	D_2	D_3	D_4

*The following inequalities hold: $A_1 < A_2 < A_3 < A_4$; $\lambda_1 < \lambda_2 < \lambda_3 < \lambda_4$; $I_1 > I_2 > I_3 > I_4$; $D_1 < D_2 < D_3 < D_4$.

training and testing sets) [4] is built as the input, targeting at achieving superior representation learning capacity to objective HDI data. Particularly, the input of the fuzzy reasoning rule at the t -th iteration is defined as:

$$\begin{aligned} \Delta^{(t)} &= \alpha \sqrt{\left(\sum_{r_{m,n} \in \Omega} (r_{m,n} - \hat{r}_{m,n})^2 \right) / |\Omega|} \\ &+ (1 - \alpha) \left(\sum_{r_{m,n} \in \Omega} |r_{m,n} - \hat{r}_{m,n}|_{abs} \right) / |\Omega|, \\ A^{(t)} &= \Delta^{(t-1)} - \Delta^{(t)}, \end{aligned} \quad (17)$$

where $A^{(t)}$ denotes the fuzzy input at the t -th iteration, $\hat{r}_{m,n}$ represents the prediction to $r_{m,n}$, t represents the iteration count, Ω represents the validation set, the coefficient α is 0.5 to balance the RMSE and MAE, $|\cdot|_{abs}$ denotes the operation to obtain the absolute value, respectively.

2) Fuzzification. The range of $A^{(t)}$ is divided into several subregions hinge on its empirical values. For instance, it is reasonable to divide the range of $A^{(t)}$ into five subintervals by defining four membership values (A_1, A_2, A_3, A_4). Subsequently, we can calculate the membership degree by triangular membership function. In detail, if $A^{(t)} \in [A_1, A_2]$, A_1 and A_2 are called the membership values, D_{A1} and D_{A2} are called the membership degrees belonging to A_1 and A_2 , respectively, as discussed in [53] and calculated as follows:

$$D_{A1} = \frac{A_2 - A^{(t)}}{A_2 - A_1}; D_{A2} = \frac{A^{(t)} - A_1}{A_2 - A_1}. \quad (18)$$

3) Fuzzy reasoning. From (17) and (18), the detailed fuzzy reasoning rule is obtained:

- To meet the target of LFA model, the value of x_m^{t-1} and y_n^{t-1} should be closer to the value of x_m^t and y_n^t . Hence, the fuzzy reasoning rule of regularization parameter λ is designed to be proportional to $A^{(t)}$ due to λ decreases as iteration count increases;
- As $A^{(t)}$ becomes smaller, the steady-state error leads to system oscillation. Moreover, the integral term helps to eliminate steady-state error. Hence, the integral parameter K_I is designed to be proportional to $A^{(t)}$ inversely;
- The derivative term reflects the future trend of the learning error and helps prevent overshooting. However, a large derivative decreases the convergence of the model and increases the time cost. Therefore, the absolute value of the derivative parameter K_D should be inversely proportional to $A^{(t)}$ for improving the robustness and efficiency of the proposed model.

Table I concludes the fuzzy reasoning rules according to the above inferences. It should be pointed out that the membership

values $A_1 \sim A_4$, $\lambda_1 \sim \lambda_4$, $I_1 \sim I_4$ are positive numbers, and the membership values $D_1 \sim D_4$ are negative.

4) Defuzzification. According to the membership degrees and fuzzing reasoning rules, the subintervals of the fuzzy outputs are acquired (e.g. if $A^{(t)} \in [A_1, A_2]$, then $\lambda \in [\lambda_1, \lambda_2]$, $K_I \in [I_1, I_2]$, and $K_D \in [D_1, D_2]$). Subsequently, the output of the fuzzy reasoning rules by the centroid method [53] is:

$$\begin{aligned} \lambda &= D_{A1} \cdot \lambda_1 + D_{A2} \cdot \lambda_2, \\ K_I &= D_{A1} \cdot I_1 + D_{A2} \cdot I_2, \\ K_D &= D_{A1} \cdot D_1 + D_{A2} \cdot D_2. \end{aligned} \quad (19)$$

The fuzzy results denote real-time changing parameters in the proposed model. Based on the above steps, an APNLF model is proposed completely.

C. Convergence Analysis for an APNLF Model

The convergence of APNLF is rigorously proved in this section. It is accelerated by a fuzzy PID, and its update scheme is achieved by the SLF-NMU. Hence, an APNLF's convergence proof can be separated into two steps as follows: **Step 1.** The APNLF converges with an SLF-NMU; and **Step 2.** The past and future update increments have no effect on its convergence.

As shown in (15) and (16), $x_{m,d}$ and $y_{n,d}$ are alternatively updated to implement the optimization process. Therefore, one of these two parameters is fixed to analyze the convergence with a specific LF alternatively, i.e., when $x_{m,d}$ is updated, $y_{n,d}$ is considered as a constant, and vice versa.

a) Implementation of Step 1.

The Eq. (1) should be proved that it is a non-increasing objective function with (3) and has a lower bound. To meet this goal, it is essential to build an auxiliary function [19].

Definition 3. Let $L(x, \bar{x})$ as an auxiliary function of $H(x)$ if the following conditions hold:

$$\begin{cases} L(x, \bar{x}) \geq H(x), \\ L(x, x) = H(x). \end{cases} \quad (20)$$

Firstly, provide the **Lemma 1** [19]:

Lemma 1. H is monotonically non-increasing if (21) holds:

$$x^{t+1} = \arg \min_x L(x, x^t). \quad (21)$$

Proof. Firstly, (22) holds based on the above **Definition 3**:

$$H(x^t) = L(x^t, x^t) \geq L(x^{t+1}, x^t) \geq H(x^{t+1}). \quad (22)$$

The remarkable thing is x^t denotes the optimal solution of $L(x, x^t)$ if $\nabla H(x^t) = 0$. That is $H(x^{t+1}) = H(x^t)$. Therefore, (22) is extended into the following converging sequence:

$$\begin{aligned} H(x_{\min}) &\leq \dots \leq H(x^{t+1}) \leq H(x^t) \\ &\leq H(x^{t-1}) \leq \dots \leq H(x_0). \end{aligned} \quad (23)$$

Afterward, we need to prove the SLF-NMU's learning rules (3) is consistent with (23).

Considering the single LF $x_{m,d}$ in X , its partial loss of objective function (1) is described as (24):

$$H_{x_{m,d}} = \sum_{r_{m,n} \in \Omega} \left(\lambda \left(\sum_{d=1}^f x_{m,d}^2 + \sum_{d=1}^f y_{n,d}^2 \right) \right), \quad (24)$$

where $\hat{r}_{m,n} = \sum_{d=1}^f x_{m,d} y_{n,d}$. After that, the first-order and second-order derivatives of $x_{m,d}$ are calculated as follows:

$$\frac{\partial H_{x_{m,d}}(x_{m,d})}{\partial x_{m,d}} = 2\lambda|\Lambda(m)|x_{m,d} + 2 \sum_{n \in \Lambda(m)} (y_{n,d} \hat{r}_{m,n} - y_{n,d} r_{m,n}), \quad (25)$$

$$\frac{\partial^2 H_{x_{m,d}}(x_{m,d})}{\partial (x_{m,d})^2} = 2\lambda|\Lambda(m)| + 2 \sum_{n \in \Lambda(m)} (y_{n,d})^2.$$

Thus, the auxiliary function is designed as follows:

$$L(x, x_{m,d}^{(t)}) = H_{x_{m,d}}(x_{m,d}^{(t)}) + \frac{\partial H_{x_{m,d}}(x_{m,d}^{(t)})}{\partial x_{m,d}^{(t)}} (x - x_{m,d}^{(t)}) + \left(\frac{\lambda|\Lambda(m)|x_{m,d}^{(t)} + \sum_{n \in \Lambda(m)} y_{n,d} \hat{r}_{m,n}}{x_{m,d}^{(t)}} \right) (x - x_{m,d}^{(t)})^2. \quad (26)$$

$$H_{x_{m,d}}(x) = H_{x_{m,d}}(x_{m,d}^{(t)}) + \frac{\partial H_{x_{m,d}}(x_{m,d}^{(t)})}{\partial x_{m,d}^{(t)}} (x - x_{m,d}^{(t)}) + \frac{1}{2} \frac{\partial^2 H_{x_{m,d}}(x_{m,d}^{(t)})}{\partial (x_{m,d}^{(t)})^2} (x - x_{m,d}^{(t)})^2. \quad (27)$$

Note that the condition $L(x, x) = H_{x_{m,d}}(x)$ holds if $x_{m,d}^{(t)} = x$. Recall **Definition 3**, $L(x, x_{m,d}^{(t)})$ is an auxiliary function of $H_{x_{m,d}}(z)$ if the following condition holds:

$$L(x, x_{m,d}^{(t)}) \geq H_{x_{m,d}}(x). \quad (28)$$

According to (26) and (27), to ensure (28) is equal to make the following inequality hold:

$$\frac{\lambda|\Lambda(m)|x_{m,d}^{(t)} + \sum_{n \in \Lambda(m)} y_{n,d} \hat{r}_{m,n}}{x_{m,d}^{(t)}} \geq \frac{1}{2} \frac{\partial^2 H_{x_{m,d}}(x_{m,d}^{(t)})}{\partial (x_{m,d}^{(t)})^2}. \quad (29)$$

With (25), Eq. (29) is rewritten as follows:

$$\frac{\lambda|\Lambda(m)|x_{m,d}^{(t)} + \sum_{n \in \Lambda(m)} y_{n,d} \hat{r}_{m,n}}{x_{m,d}^{(t)}} \geq \left(\frac{\lambda|\Lambda(m)| + \sum_{n \in \Lambda(m)} (y_{n,d})^2}{x_{m,d}^{(t)}} \right). \quad (30)$$

Considering $x_{m,d}$ and $y_{n,d}$ are non-negativity with SLF-NMU. Hence, based on (30), it is necessary to prove that:

$$\sum_{n \in \Lambda(m)} y_{n,d} \hat{r}_{m,n} \geq x_{m,d}^{(t)} \sum_{n \in \Lambda(m)} (y_{n,d})^2. \quad (31)$$

Moreover, $\hat{r}_{m,n} = \sum_{d=1}^f x_{m,d} y_{n,d}$ is reformulated as:

$$\hat{r}_{m,n} = \sum_{d=1}^f x_{m,d} y_{n,d} = x_{m,d}^{(t)} y_{n,d} + \sum_{l=1, l \neq d}^f x_{m,l} y_{n,l}. \quad (32)$$

By combining (31) and (32), Eq. (33) is obtained:

$$\sum_{n \in \Lambda(m)} y_{n,d} \hat{r}_{m,n} = \left(\frac{x_{m,d}^{(t)} \sum_{n \in \Lambda(m)} (y_{n,d})^2 + \sum_{n \in \Lambda(m)} \left(y_{n,d} \sum_{l=1, l \neq d}^f x_{m,l} y_{n,l} \right)}{x_{m,d}^{(t)}} \right) \geq x_{m,d}^{(t)} \sum_{n \in \Lambda(m)} (y_{n,d})^2. \quad (33)$$

Therefore, (31) holds with (33), implying $L(x, x_{m,d}^{(t)})$ serves as an auxiliary function for $H_{x_{m,d}}(x)$. Thereupon, **Lemma 1** stands. On the other hand, based on (26) the following equation is deduced:

$$x_{m,d}^{(t+1)} = \arg \min_x L(x, x_{m,d}^{(t)}) \Rightarrow \frac{\partial H_{x_{m,d}}(x_{m,d}^{(t)})}{\partial x_{m,d}^{(t)}} + 2 \left(\frac{\lambda|\Lambda(m)|x_{m,d}^{(t)} + \sum_{n \in \Lambda(m)} y_{n,d} \hat{r}_{m,n}}{x_{m,d}^{(t)}} \right) \times (x - x_{m,d}^{(t)}) = 0. \quad (34)$$

Afterwards, we substitute (25) into (34) to obtain:

$$x_{m,d}^{(t+1)} = x_{m,d}^{(t)} \frac{\sum_{n \in \Lambda(m)} y_{n,d} r_{m,n}}{\sum_{n \in \Lambda(m)} y_{n,d} \hat{r}_{m,n} + \lambda|\Lambda(m)|x_{m,d}^{(t)}}. \quad (35)$$

Note that the SLF-NMU's learning rule (3) aligns with (35). Therefore, it is evident that $H_{x_{m,d}}$ is monotonically non-increasing with (3).

With a positive initialization $\forall m \in M, n \in N$ and $d \in \{1, 2, \dots, f\} : x_{m,d} \geq 0, y_{n,d} \geq 0$ objective function (1) is non-increasing with the SLF-NMU's learning rules (3). Additionally, Eq. (1) has a natural lower bound greater than 0. Hence, **Step 1** is completed.

b) Implementation of Step 2.

Firstly, assuming that $\hat{x}_{m,d}^{(j)}$ is the optimization parameter's expected state of SLF-NMU at the j -th iteration. With the monotone convergence theorem [42] and **Step 1**, we have:

$$j \in \{1, 2, \dots, t+1\} : H_{x_{m,d}}(x_{m,d}^{(j-1)}) \geq H_{x_{m,d}}(\hat{x}_{m,d}^{(j)}). \quad (36)$$

According to (36), since $x_{m,d} \geq 0$, we have:

$$j \in \{1, 2, \dots, t+1\} : x_{m,d}^{(j-1)} \geq \hat{x}_{m,d}^{(j)} \geq 0. \quad (37)$$

Thus, the Eq. (38) is obtained based on (15) and (25):

$$\begin{aligned} x_{m,d}^{(t+1)} &= \hat{x}_{m,d}^{(t+1)} + K_I \sum_{j=1}^{t+1} (\hat{x}_{m,d}^{(j)} - x_{m,d}^{(j-1)}) \\ &+ K_D \left((\hat{x}_{m,d}^{(j)} - x_{m,d}^{(j-1)}) - (\hat{x}_{m,d}^{(j-1)} - x_{m,d}^{(j-2)}) \right) \\ &\Rightarrow x_{m,d}^{(t+1)} - x_{m,d}^{(t)} = \hat{x}_{m,d}^{(t+1)} - x_{m,d}^{(t)} \\ &+ K_I \sum_{j=1}^{t+1} (\hat{x}_{m,d}^{(j)} - x_{m,d}^{(j-1)}) \\ &+ K_D \left((\hat{x}_{m,d}^{(j)} - x_{m,d}^{(j-1)}) - (\hat{x}_{m,d}^{(j-1)} - x_{m,d}^{(j-2)}) \right). \end{aligned} \quad (38)$$

So the following equation is obtained based on (37) and (38):

$$\begin{aligned}
 x_{m,d}^{(t+1)} - x_{m,d}^{(t)} &= \underbrace{\hat{x}_{m,d}^{(t+1)} - x_{m,d}^{(t)}}_{\leq 0} + K_I \underbrace{\sum_{j=1}^{t-1} (\hat{x}_{m,d}^{(j)} - x_{m,d}^{(j-1)})}_{\leq 0} \\
 &\quad + \underbrace{(K_I + K_D)}_{\geq 0} \underbrace{(\hat{x}_{m,d}^{(j)} - x_{m,d}^{(j-1)})}_{\leq 0} \\
 &\quad + \underbrace{(K_I - K_D)}_{\geq 0} \underbrace{(\hat{x}_{m,d}^{(j-1)} - x_{m,d}^{(j-2)})}_{\leq 0},
 \end{aligned} \tag{39}$$

where the integral parameter K_I is a non-negative constant, the derivative parameter K_D is a negative constant, and K_D 's absolute value is much less than K_I . Hence, the following recursion is established:

$$\begin{aligned}
 H_{x_{m,d}}(x_{m,d}^{(0)}) &\geq \dots \geq H_{x_{m,d}}(x_{m,d}^{(t-1)}) \\
 &\geq H_{x_{m,d}}(x_{m,d}^{(t)}) \geq H_{x_{m,d}}(x_{m,d}^{(t+1)}) \geq 0,
 \end{aligned} \tag{40}$$

which demonstrates that the sequence $\{H_{x_{m,d}}(x_{m,d}^{(t)})\}$ is bounded and non-increasing. Thus, **Step 2** is implemented. According to the aforementioned conclusions, the APNLF's convergence is accomplished.

D. Algorithm Design and Analysis

Algorithm 1 APNLF model

Input: M, N, Λ, f, Ω , fuzzy rules table

1 Operation

2 **while** not converging in t iterations **do**

3 $\forall r_{m,n} \in \Lambda$

4 $\forall d \in \{1, \dots, f\}$

5 Update x_m and y_n based on (3).

6 $\forall d \in \{1, \dots, f\}$

7 Calculate current increment $xpr_{m,d}$ with (9).

8 Accumulate past increment $xin_{m,d}$ with (13).

9 Calculate derivative increment $xde_{m,d}$ with (13).

10 Calculate refinement increment $xc_{m,d}$ with (13).

11 Update $x_{m,d}$ with (15).

12 $\forall d \in \{1, \dots, f\}$

13 Calculate current increment $ypr_{n,d}$ with (9).

14 Accumulate past increment $yin_{n,d}$ with (13).

15 Calculate derivative increment $yde_{n,d}$ with (13).

16 Calculate refinement increment $yc_{n,d}$ with (13).

17 Update $y_{n,d}$ with (16).

18 Compute A^t based on (17).

19 Update λ, K_I , and K_D based on (19).

20 $t = t + 1$.

Output: X, Y

21 **end**

The APNLF model's algorithm analysis is designed in **Algorithm 1**. As shown in Algorithm 1, the APNLF model's computation cost depends on two factors:

a) Initialization: $T_1 = \Theta((|M| + |N|) \times f)$.

b) Training: $T_2 = \Theta(|\Lambda| \times f + (|M| + |N|) \times f)$.

TABLE II: THE DETAILS OF DATASETS.

No.	Name	Row	Column	Known Entries	Density
D1	Hetrec-ML [54]	10,109	2,113	855,598	4.01%
D2	Delicious [54]	68,755	14,345	487,131	0.04%
D3	EachMovie [55]	72,916	1,648	2,811,718	2.34%
D4	Pro-Int1 [56]	3,277	3,277	638,904	5.95%
D5	Goodbooks [40]	53,424	10,000	5,976,480	1.12%

Conclusively, the computational cost of the APNLF model is as below:

$$\begin{aligned}
 T &= T_1 + T_2 \times t \\
 &= \Theta((|M| + |N|) \times f) \\
 &\quad + \Theta(|\Lambda| \times f \times t + (|M| + |N|) \times f \times t) \\
 &\approx \Theta(t \times |\Lambda| \times f).
 \end{aligned}$$

V. EXPERIMENTAL RESULTS AND ANALYSIS

To validate the superiority and efficacy of the proposed APNLF model, we conduct a comprehensive analysis of a series of experiments to address the four pertinent questions:

RQ1: How does APNLF perform on different industrial HDI datasets compared to various state-of-the-art models?

RQ2: How does the current, past, and future update increment information contribute to the performance of APNLF?

RQ3: How do the parameters of integral and derivative affect the convergence performance of APNLF?

RQ4: How APNLF can be applied to real-world scenario?

A. General Settings

Evaluation Metrics. To evaluate the converged result of the APNLF model on diverse industrial HDI matrices. The mean absolute error (MAE) and root mean squared error (RMSE) are regarded as the evaluation indicators [4]. The formulas of them are defined below:

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

where $\hat{r}_{m,n}$ represents the predicted value corresponding to $r_{m,n}$ produced by the tested model, Φ represents the testing dataset, $|\cdot|_{abs}$ represents the absolute value of the enclosed number, respectively. Particularly, the model possesses better-converged performance if the value of RMSE and MAE is even smaller. Meanwhile, the model's efficiency is reflected in the total time for model convergence.

Experimental Datasets. Table II gives the details of five different industrial datasets. To ensure the fairness of the experiment, all known entries of each dataset are equally divided into ten subsets. Subsequently, the ten subsets are divided into three parts in the ratio of seven to two to one, as the training set, testing set, and validation set respectively. Meanwhile, the process of dividing the data is cross-validated five times. Furthermore, the termination conditions are defined as below: a) the training iteration count more than 1000, and

TABLE III: COMPARED MODELS.

M1	The proposed APNLF model.
M2	A standard NLF model [20].
M3	A fast NLF model [41].
M4	An LFA model with a PID-based stochastic optimizer [28].
M5	A fuzzy PID-based LF model [53].
M6	An Adam-based LFA model [48].
M7	A RMSprop-based LFA model [4].
M8	An Ada-Delta-based LFA model [4].
M9	A variational autoencoder-based LFA model [46].
M10	A light graph convolution network-based LFA model [45].
M11	A Graph Diffusion Collaborative Filtering model [58].
M12	An incremental PID-enhanced SGD-based LFA model [25].
M13	A PI-controlled acceleration NLF model [34].

b) the difference of RMSE or MAE between two consecutive rounds is less than 10^{-5} . If either of the above conditions is satisfied during the convergence process, the tested model convergence terminates immediately.

To guarantee the uniformity of the tested models, the initial configuration of the involved model is unified as the following.

- According to prior studies [53], the LF matrices X and Y are initialized randomly with the same range of $(-0.005, 0.005)$ for all compared models;
- To ensure the fairness of compared models, regularization coefficient $\lambda=0.05$ and learning rate $\eta=0.04$ for all SGD-based LFA related models [53];
- To balance model efficiency and representation ability, the LF space dimension f is set as 20 [53]
- Based on the empirical obtained after training, we set membership value in fuzzy PID $A_1 \sim A_4$ is (0.001, 0.0015, 0.002, 0.0025), $\lambda_1 \sim \lambda_4, I_1 \sim I_4$, and $D_1 \sim D_4$ of D1-D2 is (0.078, 0.08, 0.082, 0.084), (0.039, 0.04, 0.041, 0.042), (-0.00052, -0.00051, -0.0005, -0.00049), D3 is (0.029, 0.03, 0.031, 0.032), (0.088, 0.089, 0.09, 0.091), (-0.0036, -0.0034, -0.0032, -0.003), D4 is (0.01, 0.011, 0.012, 0.013), (0.071, 0.072, 0.073, 0.074), (-0.00017, -0.00016, -0.00015, -0.00014), D5 is (0.077, 0.078, 0.079, 0.08), (0.037, 0.038, 0.039, 0.04), (-0.0065, -0.0062, -0.0059, -0.0056).

B. Comparison against State-of-the-art Models (RQ1)

This part performs comparison experiments on five industrial HDI matrices. All configurations of the tested models are given in Table III. It should be pointed out that M1-M8, M12, and M13 are executed on a Tablet with a 3.2 GHz i5 CPU and 16 GB RAM with the program platform of JAVA SE 7U60. The deep learning models M9-M11 are tested on a GPU platform with two NVIDIA RTX 3090 GPU cards. Tables IV records the comparison results. Figs. 1 illustrates the training curves of M1-M13. Through the analysis of the above data, the following outcomes are drawn:

- APNLF possesses considerable excellent computational efficiency.** M1 obtains the highest convergence efficiency on all cases from Table IV. For instance, M1 spends 17.1 seconds to achieve the minimum RMSE on D1, 40% (i.e., $(\text{Cost}_{\text{high}} - \text{Cost}_{\text{low}}) / \text{Cost}_{\text{high}}$) improvement

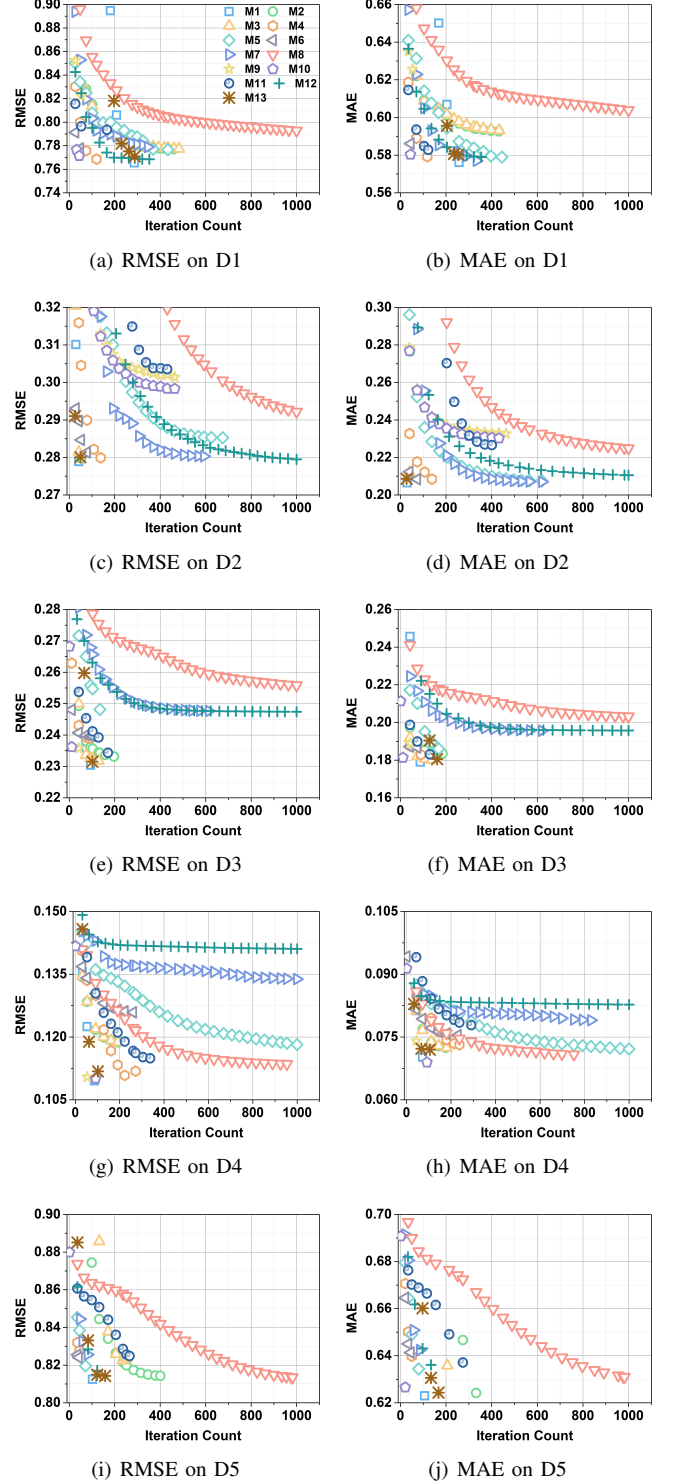


Fig. 1: Training curves of M1-13 on D1-5 in RMSE and MAE.

over M2's 28.6 seconds, 40% over M3's 28.9 seconds, 18% over M4's 21.1 seconds, 42% over M5's 29.8 seconds, 13% over M6's 19.7 seconds, 83% over M7's 103.9 seconds, 96% over M8's 486.1 seconds, 97% over M9's 632.2 seconds, 85% over M10's 118.9 seconds, 91% over M11's 209.1 seconds, 18% over M12's 21.1 seconds, and 3.9% over M13's 17.8 seconds, respectively.

TABLE IV: RMSE, MAE, AND CONSUMED TIME OF M1-13 ON D1-5, WHERE * INDICATES THAT THE COMPARED MODEL IS OUTPERFORMED BY M1.

Model	D1				D2			
	RMSE	Time-RMSE	MAE	Time-MAE	RMSE	Time-RMSE	MAE	Time-MAE
M1	0.7654 $\pm$ 9E-4	17.1 $\pm$ 3.4	0.5761 $\pm$ 6E-4	15.3 $\pm$ 2.3	0.2734 $\pm$ 1E-4	3.2 $\pm$ 1.6	0.2066 $\pm$ 4E-4	2.8 $\pm$ 2.5
M2	0.7716 $\pm$ 5E-4	28.6 $\pm$ 2.1	0.5894 $\pm$ 2E-4	26.7 $\pm$ 5.8	0.2736 $\pm$ 8E-4	4.1 $\pm$ 3.1	0.2068 $\pm$ 4E-4	4.1 $\pm$ 4.8
M3	0.7721 $\pm$ 2E-4	28.9 $\pm$ 2.4	0.5892 $\pm$ 1E-4	25.9 $\pm$ 4.1	0.2737 $\pm$ 5E-4	3.7 $\pm$ 4.2	0.2070 $\pm$ 3E-4	3.6 $\pm$ 8.8
M4	0.7686 $\pm$ 5E-4	21.1 $\pm$ 7.3	0.5791 $\pm$ 4E-4	21.0 $\pm$ 2.6	0.2804 $\pm$ 1E-4	24.1 $\pm$ 2.3	0.2091 $\pm$ 5E-4	24.2 $\pm$ 2.4
M5	0.7666 $\pm$ 2E-4	29.8 $\pm$ 6.4	0.5789 $\pm$ 5E-4	30.6 $\pm$ 5.8	0.2778 $\pm$ 3E-4	47.8 $\pm$ 3.1	0.2069 $\pm$ 8E-4	44.3 $\pm$ 5.2
M6	0.7831 $\pm$ 2E-4	19.7 $\pm$ 3.2	0.5907 $\pm$ 3E-4	19.4 $\pm$ 4.6	0.2738 $\pm$ 8E-4	28.1 $\pm$ 2.2	0.2075 $\pm$ 6E-4	27.3 $\pm$ 4.3
M7	0.7789 $\pm$ 6E-4	103.9 $\pm$ 5.8	0.5770 $\pm$ 3E-4	102.6 $\pm$ 4.3	0.2923 $\pm$ 5E-4	154.5 $\pm$ 2.3	0.2316 $\pm$ 6E-4	177.7 $\pm$ 2.4
M8	0.7948 $\pm$ 2E-4	486.1 $\pm$ 3.4	0.6010 $\pm$ 4E-4	470.7 $\pm$ 2.4	0.2812 $\pm$ 2E-4	471.8 $\pm$ 2.1	0.2157 $\pm$ 4E-4	468.5 $\pm$ 4.3
M9	0.8223 $\pm$ 7E-4	632.2 $\pm$ 32.4	0.6261 $\pm$ 3E-4	597.1 $\pm$ 28.6	0.3012 $\pm$ 3E-4	2372.0 $\pm$ 2.1	0.2326 $\pm$ 1E-4	2324.9 $\pm$ 3.2
M10	0.7714 $\pm$ 3E-4	118.9 $\pm$ 4.1	0.5803 $\pm$ 2E-4	120.5 $\pm$ 4.2	0.2980 $\pm$ 6E-4	2129.0 $\pm$ 2.4	0.2300 $\pm$ 2E-4	2129.0 $\pm$ 2.4
M11	0.7741 $\pm$ 2E-4	209.1 $\pm$ 3.4	0.5827 $\pm$ 5E-4	290.1 $\pm$ 3.1	0.3034 $\pm$ 2E-4	400.0 $\pm$ 6.4	0.2265 $\pm$ 2E-4	401.4 $\pm$ 3.4
M12	0.7684 $\pm$ 8E-4	21.1 $\pm$ 4.1	0.5899 $\pm$ 1E-4	22.1 $\pm$ 2.8	0.2805 $\pm$ 4E-4	35.8 $\pm$ 4.7	0.2113 $\pm$ 4E-4	36.1 $\pm$ 2.6
M13	0.7690 $\pm$ 5E-4	17.8 $\pm$ 2.5	0.5776 $\pm$ 2E-4	15.6 $\pm$ 4.2	0.2738 $\pm$ 3E-4	3.7 $\pm$ 5.3	0.2069 $\pm$ 5E-4	2.9 $\pm$ 3.9
Model	D3				D4			
	RMSE	Time-RMSE	MAE	Time-MAE	RMSE	Time-RMSE	MAE	Time-MAE
M1	0.2350 $\pm$ 2E-4	13.5 $\pm$ 4.2	0.1793 $\pm$ 3E-4	12.6 $\pm$ 4.5	0.1096 $\pm$ 8E-4	3.5 $\pm$ 2.2	0.0702 $\pm$ 2E-4	2.9 $\pm$ 7.9
M2	0.2375 $\pm$ 2E-4	30.5 $\pm$ 2.3	0.1856 $\pm$ 4E-4	31.8 $\pm$ 2.8	0.1165 $\pm$ 3E-4	7.6 $\pm$ 3.7	0.0714 $\pm$ 5E-4	7.0 $\pm$ 4.5
M3	0.2363 $\pm$ 2E-4	18.0 $\pm$ 4.5	0.1824 $\pm$ 3E-4	17.8 $\pm$ 9.3	0.1166 $\pm$ 2E-4	7.6 $\pm$ 1.2	0.0713 $\pm$ 5E-4	6.4 $\pm$ 5.3
M4	0.2362 $\pm$ 6E-4	42.5 $\pm$ 1.3	0.1803 $\pm$ 6E-4	45.5 $\pm$ 6.2	0.1118 $\pm$ 3E-4	34.4 $\pm$ 5.3	0.0741 $\pm$ 7E-4	30.7 $\pm$ 2.3
M5	0.2436 $\pm$ 4E-4	25.3 $\pm$ 2.4	0.1836 $\pm$ 6E-4	31.8 $\pm$ 3.2	0.1203 $\pm$ 3E-5	49.0 $\pm$ 4.1	0.0729 $\pm$ 7E-4	46.1 $\pm$ 3.9
M6	0.2357 $\pm$ 2E-4	126.9 $\pm$ 2.4	0.1845 $\pm$ 5E-4	89.3 $\pm$ 2.7	0.1280 $\pm$ 2E-4	87.7 $\pm$ 2.3	0.0762 $\pm$ 5E-4	77.0 $\pm$ 6.4
M7	0.2868 $\pm$ 2E-4	525.6 $\pm$ 4.2	0.2199 $\pm$ 4E-4	523.1 $\pm$ 3.2	0.2026 $\pm$ 6E-4	207.2 $\pm$ 1.3	0.1868 $\pm$ 3E-4	173.1 $\pm$ 3.4
M8	0.2490 $\pm$ 2E-4	1416.8 $\pm$ 4.2	0.1952 $\pm$ 4E-4	1425.2 $\pm$ 2.3	0.1171 $\pm$ 2E-4	336.1 $\pm$ 5.3	0.8030 $\pm$ 7E-4	268.1 $\pm$ 6.3
M9	0.2357 $\pm$ 2E-4	27.8 $\pm$ 2.2	0.1899 $\pm$ 1E-4	26.6 $\pm$ 2.2	0.1127 $\pm$ 2E-4	28.2 $\pm$ 0.6	0.0741 $\pm$ 4E-4	29.8 $\pm$ 1.1
M10	0.2362 $\pm$ 1E-4	173.4 $\pm$ 1.6	0.1813 $\pm$ 3E-4	173.3 $\pm$ 1.6	0.1101 $\pm$ 2E-4	149.1 $\pm$ 2.2	*0.0676 $\pm$ 2E-4	142.4 $\pm$ 3.6
M11	0.2392 $\pm$ 6E-4	2725.4 $\pm$ 5.4	0.1831 $\pm$ 5E-4	2831.0 $\pm$ 6.4	0.1149 $\pm$ 6E-4	167.5 $\pm$ 3.2	0.0778 $\pm$ 2E-4	144.8 $\pm$ 4.1
M12	0.2436 $\pm$ 6E-4	163.9 $\pm$ 3.3	0.1939 $\pm$ 2E-4	162.5 $\pm$ 4.2	0.1431 $\pm$ 6E-4	39.8 $\pm$ 3.4	0.0835 $\pm$ 7E-4	39.8 $\pm$ 2.3
M13	0.2353 $\pm$ 5E-4	16.3 $\pm$ 3.4	0.1795 $\pm$ 3E-4	15.1 $\pm$ 3.1	0.1098 $\pm$ 4E-4	4.1 $\pm$ 4.6	0.0710 $\pm$ 7E-4	4.3 $\pm$ 4.7
Model	D5				*Loss/Win of M1 compared with other models			
	RMSE	Time-RMSE	MAE	Time-MAE	Accuracy		Efficiency	
M1	0.8131 $\pm$ 2E-4	42.8 $\pm$ 3.1	0.6239 $\pm$ 2E-4	47.7 $\pm$ 2.6	-		-	
M2	0.8142 $\pm$ 4E-4	190.8 $\pm$ 0.4	0.6248 $\pm$ 4E-4	145.4 $\pm$ 0.2	0/10		0/10	
M3	0.8205 $\pm$ 2E-4	177.1 $\pm$ 3.2	0.6357 $\pm$ 5E-4	176.1 $\pm$ 5.1	0/10		0/10	
M4	0.8265 $\pm$ 3E-4	81.2 $\pm$ 3.3	0.6399 $\pm$ 2E-4	93.9 $\pm$ 2.7	0/10		0/10	
M5	0.8132 $\pm$ 3E-4	44.8 $\pm$ 4.4	0.6345 $\pm$ 5E-4	49.0 $\pm$ 3.6	0/10		0/10	
M6	0.8232 $\pm$ 6E-4	211.6 $\pm$ 2.1	0.6415 $\pm$ 5E-4	216.5 $\pm$ 1.6	0/10		0/10	
M7	0.8342 $\pm$ 5E-4	160.1 $\pm$ 3.0	0.6357 $\pm$ 5E-4	175.8 $\pm$ 2.8	0/10		0/10	
M8	0.8135 $\pm$ 6E-4	3311.0 $\pm$ 3.8	0.6309 $\pm$ 6E-4	3353.3 $\pm$ 3.0	0/10		0/10	
M9	0.9386 $\pm$ 5E-4	633.2 $\pm$ 2.6	0.7671 $\pm$ 5E-4	597.0 $\pm$ 3.0	0/10		0/10	
M10	0.8141 $\pm$ 2E-4	2940.0 $\pm$ 2.5	0.6265 $\pm$ 1E-4	2940.0 $\pm$ 2.5	1/9		0/10	
M11	0.8248 $\pm$ 5E-4	21712.1 $\pm$ 4.3	0.6370 $\pm$ 2E-4	22355.5 $\pm$ 4.2	0/10		0/10	
M12	0.8163 $\pm$ 5E-4	52.3 $\pm$ 5.4	0.6361 $\pm$ 3E-4	57.7 $\pm$ 2.4	0/10		0/10	
M13	0.8140 $\pm$ 6E-4	62.9 $\pm$ 3.7	0.6247 $\pm$ 5E-4	68.0 $\pm$ 4.8	0/10		0/10	

*the *Loss/Win of M1's accuracy (efficiency) compared to other models stems from their performance in RMSE and MAE (Time-RMSE and Time-MAE).

TABLE V: AVERAGE RANKS OF ALL COMPARED MODELS.

Rank	M1	M2	M3	M4	M5	M6	M7
Accuracy	1.1	5.8	6.3	6.4	5.6	8.4	10.5
Rank	M8	M9	M10	M11	M12	M13	-
Accuracy	9.6	10.4	5.7	9.0	9.0	3.2	-
Rank	M1	M2	M3	M4	M5	M6	M7
Efficiency	1.0	5.3	4.6	5.4	6.2	7.0	9.5
Rank	M8	M9	M10	M11	M12	M13	-
Efficiency	12.0	9.1	10.6	11.6	6.2	2.5	-

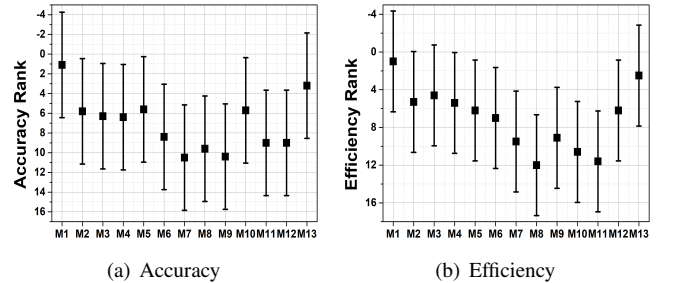


Fig. 2: Results of Nemenyi analysis.

Note that APNLF's time cost per iteration is much longer than the M2, M3, M5, and M12 model, but it takes

TABLE VI: ABLATION STUDY RESULTS ON D1-5.

Data	Method	RMSE	Iterations	*Per	**Total	MAE	Iterations	*Per	**Total
D1	APNLF w/o ID	0.7716	485	0.058	28.6	0.5894	452	0.059	26.7
	APNLF w/o D	0.7690	288	0.062	17.8	0.5776	256	0.061	15.6
	APNLF	0.7654	266	0.064	17.1	0.5761	244	0.062	15.3
D2	APNLF w/o ID	0.2736	43	0.095	4.1	0.2068	40	0.102	4.1
	APNLF w/o D	0.2738	38	0.097	3.7	0.2069	27	0.107	2.9
	APNLF	0.2734	32	0.1	3.2	0.2066	25	0.112	2.8
D3	APNLF w/o ID	0.2375	197	0.154	30.5	0.1856	198	0.16	31.8
	APNLF w/o D	0.2353	102	0.159	16.3	0.1795	92	0.164	15.1
	APNLF	0.2350	82	0.164	13.5	0.1793	73	0.172	12.6
D4	APNLF w/o ID	0.1165	192	0.039	7.6	0.0714	176	0.039	7.0
	APNLF w/o D	0.1098	102	0.04	4.1	0.0710	105	0.041	4.3
	APNLF	0.1096	83	0.042	3.5	0.0702	69	0.042	2.9
D5	APNLF w/o ID	0.8142	483	0.395	190.8	0.6248	373	0.389	145.4
	APNLF w/o D	0.8140	156	0.403	62.9	0.6247	165	0.412	68.0
	APNLF	0.8131	102	0.419	42.8	0.6239	106	0.45	47.7

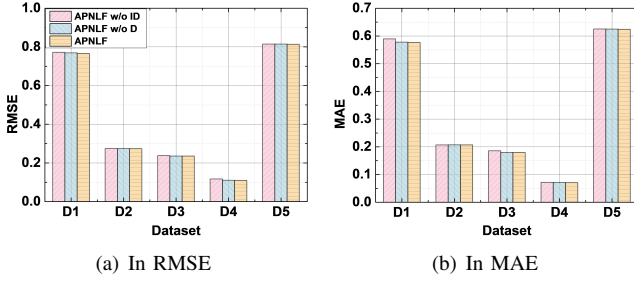


Fig. 3: Accuracy gain by incorporating integral and derivative terms on D1-5.

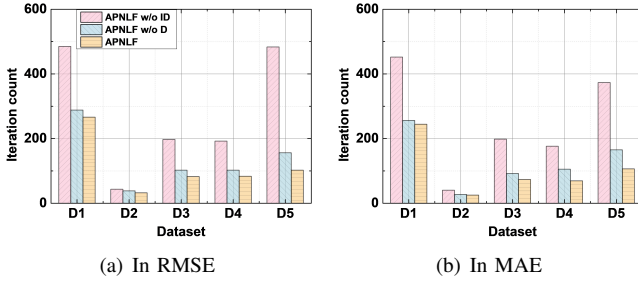


Fig. 4: Efficiency gain by incorporating integral and derivative terms on D1-5.

much less iterations to converge. Hence, the convergence efficiency of APNLF is much higher than the M2, M3 and M5, and M12. Besides, M4 and M6-M8 consume much time because they utilize past information to compute the current gradient per iteration. Moreover, M13 neglects the positive effects of future trend states on convergence efficiency. Meanwhile, the costs per iteration of M9-M11 are quite intensive because they are DNN-based models, thus, the convergence efficiency of M9-M11 are not a patch on M1. Thus, M1 achieves crucial efficiency gain.

- b) **APNLF acquires outstanding results in missing value prediction of an HDI matrix.** It has the best prediction accuracy in nine out of ten cases from Table IV. M1's RMSE converged with a result of 0.7654 as the best prediction accuracy, which is 0.8% (i.e., $(RMSE_{high} -$

$RMSE_{low})/RMSE_{high}$) improvement over the M2's 0.7716, 0.86% improvement over the M3's 0.7721, 0.41% improvement over the M4's 0.7686, 0.15% improvement over the M5's 0.7666, 2.26% improvement over the M6's 0.7831, 1.73% improvement over the M7's 0.7789, 3.69% improvement over the M8's 0.7948, 6.91% improvement over the M9's 0.8223, 0.6% improvement over the M10's 0.7714, 1.12% improvement over the M11's 0.7741, 0.39% improvement over the M12's 0.7684, and 0.46% improvement over the M13's 0.7690. It should be pointed out that M2-M8, M12 and M13 are efficient optimization algorithms for missing data prediction. Meanwhile, M9-M11 are representative and highly expressive DNN-based models. Thus, the comparative experimental results strongly prove that the APNLF model has excellent prediction accuracy. Moreover, the excellent representation learning ability of the APNLF model relies on the following two aspects: a) it utilizes learning error to compute past state and future trends to accelerate convergence; b) the fuzzy reasoning rule enables APNLF to have fine-grained hyper-parameters.

- c) **The APNLF model's performance shows vital statistical significance in our experiments.** To explain the above results better, we introduce the Friedman test [37] to ensure APNLF's statistical significance based on Tables IV. Foremost, the efficiency and accuracy's average rank of each model are computed as shown in Table V. Note that there are k compared models and N testing cases in total, r_i^j denotes the rank of the j -th model on the i -th testing cases, and the average rank of each model's performance is computed as $A_j = \sum_{i=1}^N r_i^j / N$. Thus, the following Friedman value is obtained:

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

The computation of the testing score is formulated by:

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

The testing score obeys the F -distribution with $k - 1$ and $(k - 1)(N - 1)$ degrees of freedom. If the F_F is higher than the corresponding value with critical level α , then all model-equivalent assumptions can be directly rejected. There are thirteen models are tested on ten testing cases. Thus, $k=13$, $N=10$, and critical value is 1.84, respectively. Simultaneously, the F_F s are computed as 9.98 for accuracy and 28.67 for efficiency based on Table V. Thus, at a confidence level of 95%, the models tested varied considerably in accuracy and efficiency. Furthermore, M1 achieves the lowest rank value both in accuracy and efficiency, which means that the proposed model is more outstanding than others in convergence effect. The Nemenyi test was executed to verify whether the experimental performance of the presented models outperforms the compared models. Specifically, the Nemenyi test considers that any two models have significant differences if the gap in their rank is larger than the critical value. The critical value is given as follows [37]:

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

where the constant q_α is determined by the studentized range statistics. The critical value $q_\alpha=3.07$ with the critical level $\alpha=0.1$ with $k=13$, and $N=10$ [10]. Thus, the $\delta=5.35$ based on the critical value calculation equation, which figures that the two models have a significant difference in terms of performance with a confidence level of 90% if the rank difference exceeds 5.35.

As shown in Fig. 2, it is evident that M1 significantly outperforms all compared models in convergence effect. Therefore, the APNLF model has considerable representation capacity for missing data of an HDI matrix.

C. Ablation Analyses (RQ2)

This study devotes to investigate the impact of past and future update increment information, i.e., integral and derivative terms. Hence, the experiment implements the comparison among APNLF w/o ID (such a model only maintains proportional term K_P is 1, thus such a model is equal to a standard NLF model), APNLF w/o D, and APNLF. Table VI and Fig. 3-4 record the comparison results. From them, we have:

- The APNLF w/o D obtains higher prediction accuracy and convergence rate than the APNLF w/o ID does.** For example, the APNLF w/o D takes 288 counts (17.8 seconds) to converge at the lowest RMSE of 0.7690 on D1 from Table VI, while the APNLF w/o ID takes 485 counts (28.6 seconds) to converge at the lowest RMSE of 0.7716. Thus, the APNLF w/o D acquires the accuracy gain and efficiency gain at 0.33% and 40% (37%). Similarly, the same goes for the other testing cases. It proves that historical information brings a yield both on the convergence efficiency and accuracy.
- With the derivative term, the APNLF obtains better convergence results.** For instance, the APNLF achieves the lowest RMSE at 0.7654 on D1 from Table VI, which is about 0.46% and 0.8% lower than 0.7690 by APNLF

w/o D and 0.7716 by APNLF w/o ID. Likewise, APNLF consumes 266 counts (17.1 seconds) to converge, which is 7.63% (3.9%) lower than APNLF w/o D's 288 counts (17.8 seconds), 45.15% (40.2%) lower than APNLF w/o ID's 485 counts (28.6 seconds). The same results are encountered in other testing cases, demonstrating that adding future update increment information to APNLF w/o D can help the model converge further.

D. Parameter Sensitivity (RQ3)

The gain parameters of integral and derivative exert a critical influence on the convergence performance of APNLF. Therefore, this subsection investigates the impact of the parameter searching ranges for K_I and K_D on APNLF's performance. To verify the selected ranges ($K_I \in [0.039, 0.042]$, $K_D \in [-0.00052, -0.00049]$) are optimal, the validated searching ranges are extended equally up and down. More specifically, the searching ranges of K_I are $[0.035, 0.038]$, $[0.039, 0.042]$ and $[0.043, 0.046]$, the searching ranges of K_D are $[-0.00056, -0.00053]$, $[-0.00052, -0.00049]$ and $[-0.00048, -0.00045]$. Table VII illustrates the accuracy and efficiency results of APNLF under different searching ranges on D1 and D2. Based on these results, the following finding are obtained:

- To validate the effects of K_I , the searching range of K_D is fixed at $[-0.00052, -0.00049]$. As shown in Table VII, the searching range of K_I adopted in this study, i.e., $[0.039, 0.042]$, achieves the best prediction accuracy on all testing cases. The main reason lies in the fact that an appropriate K_I eliminates steady-state error by incorporating historical information. Consequently, an overly small K_I struggles to eliminate the steady-state error, while an excessively large one leads to overshoot.
- Regarding the effect of derivative parameter K_D , the searching range of K_I is fixed at $[0.039, 0.042]$. For instance, the searching range of K_D adopted in this study, i.e., $[-0.00052, -0.00049]$, obtains a markedly superior convergence performance. The primary reason is that an appropriate search range for K_D is critical for suppressing overshoot and oscillation.

E. Application of Real Scenarios (RQ4)

The APNLF shows considerable potential value in practical applications. For instance, the scenario of Water Resources Research (WRR): Water quality data obtained through automatic high-frequency monitoring (AHFM) contains a variety of valuable information, which can be used for hydrological analysis and mathematical modeling. Nevertheless, substantial missing data caused by sensor failures and routine maintenance, resulting in the AHFM water quality data having an HDI characteristic that significantly hinder downstream applications. Advantageously, the APNLF has been proven to be able significantly enhance the SGD-LFA's processing capability for HDI data via incremental control acceleration. Consequently, it can achieve efficient and accurate imputation of AHFM water quality data, thereby reliably facilitating intelligent decision-making for eutrophication management

TABLE VII: PERFORMANCE COMPARISON BETWEEN DIFFERENT SEARCHING RANGES IN RMSE AND MAE.

Data	Evaluation	K_I			K_D		
		[0.035, 0.038]	[0.039, 0.042]	[0.043, 0.046]	[-0.00056, -0.00053]	[-0.00052, -0.00049]	[-0.00048, -0.00045]
D1	RMSE	0.7710	0.7654	0.7698	0.7695	0.7654	0.7687
	*Time-RMSE	17.2	17.1	26.4	17.3	17.1	21.5
	MAE	0.5812	0.5761	0.5768	0.5790	0.5761	0.5775
	*Time-MAE	15.1	15.3	23.6	15.5	15.3	20.3
D2	RMSE	0.2753	0.2734	0.2751	0.2753	0.2734	0.2744
	*Time-RMSE	3.4	3.2	3.9	3.5	3.2	4.1
	MAE	0.2088	0.2066	0.2082	0.2085	0.2066	0.2076
	*Time-MAE	3.0	2.8	3.6	3.1	2.8	3.3

* Total time (Secs).

and maintenance. Meanwhile, the APNLF also demonstrate potential in Intelligent Transportation Systems (ITS): A large amount of spatio-temporal traffic data is generated from human travel activities. However, due to faulty sensors and extreme weather, the traffic data exhibits extremely HDI characteristics, making it difficult to conduct in-depth statistical analysis and pattern mining. Accordingly, the proposed acceleration strategy can be jointly modeled with tensor factorization, leveraging a feedback mechanism to reconstruct learning error of real-time states to achieve accurate and efficient imputation of spatio-temporal traffic data, thereby predicting traffic flow, formulating real-time travel strategies, and promoting the development of intelligent transportation systems.

To demonstrate the application of the APNLF in real-world scenarios, we provide the following two cases. Moreover, the related experiment results are record in Table VIII.

- a) **Protein-protein interaction (PPI) network.** In the field of biology and medicine, elucidating the interactions between proteins provides critical theoretical foundations for medicine discovery and disease diagnosis. A PPI network is composed of a large number of protein nodes and a relatively limited set of known interactions. However, a large number of unobserved interactions limits the application of PPI in practical research. Therefore, the proposed APNLF can be used to accurately predict the interactions between proteins. The specific implementation step is as follows: a) proteins are abstracted as nodes, and their known interactions are quantified as weights of the edges connecting the nodes. Hence, an HDI interaction matrix is constructed based on the nodes and weights, where missing interactions correspond to unknown entries; b) APNLF generates LF matrices and is trained based on the HDI matrix to obtain the ultimate low-dimensional feature vector for each protein; and c) the interaction weights between any pair of proteins can be predicted based on these feature vectors. Consequently, APNLF refines the interaction information of PPI, which provides a basis for identification of potential medicine targets and disease diagnosis.
- b) **Recommender Systems.** Douban rating system, as a typical user-item interaction systems, involves large-scale user and movie entities. However, each user ratings only a small subset of movies, resulting in severely sparse rating data that hampers further accurate personalized

TABLE VIII: THE REAL-WORLD PERFORMANCE OF APNLF.

Dataset	RMSE	*Time-RMSE	MAE	*Time-MAE
Protein [32]	0.1224	15.8	0.0821	5.74
Douban [53]	0.7247	405.8	0.5621	540.2

* Total time (Secs).

recommendations. The proposed APNLF is able to mine users' potential preferences and items' potential attributes from the sparse historical interactive ratings between users and items. Specifically, APNLF construct a HDI matrix by representing users and items as nodes, and observed interactions as known entries. Specifically, APNLF construct a HDI rating matrix by representing users and items as nodes, and observed ratings as known entries. Subsequently, the constructed LF matrices are trained on the HDI matrix to derive user and item feature vectors. Finally, the obtained user and item feature vectors are utilized to predict ratings for items that users have not interacted with each other, which contributes to building accurate personalized movie recommendation systems.

VI. CONCLUSIONS

To accelerate the convergence of the NLF model, this study designs an APNLF model innovatively, which proposes a PSN algorithm by incorporating the integral and derivative terms of the PID mechanism into a standard NLF algorithm to obtain better convergence results; and a specific defined fuzzy reasoning rule to implement parameters adaptation. Rigorous experiment results and statistical analysis demonstrate that APNLF has achieved significant accuracy gain over other state-of-the-art models when addressing the down-stream task of missing data estimation on an HDI matrix. However, the proposed work still has several aspects for improvement:

- a) APNLF constructs a one-dimensional fuzzy table for parameter real-time adaptation. However, the one-dimensional fuzzy table demonstrates limited efficacy in adequately fitting nonlinear data, thereby restricting the model's ability to tackle complex problems. Therefore, a two-dimensional fuzzy table regarding to learning error and its variation will be exploring in the future.
- b) Outlier data caused by various uncertain disturbances are prevalent in real-world industrial datasets, these outliers can adversely affect model's convergence ability to a

certain extent. Thus, we will be devoted to developing enhanced robustness strategies in future work.

- c) APNLF obtains outstanding performance by reconstructing single node's learning error. However, nodes exhibit strong correlations derived from entity interactions, rather than being mutually independent. Hence, the future work will focus on jointly modeling relevant nodes to develop more systematic optimization strategies.

REFERENCES

- [1] X. Liao, H. Wu, T-T. He, and X. Luo, "A Proximal-ADMM-Incorporated Nonnegative Latent-Factorization-of-Tensors Model for Representing Dynamic Cryptocurrency Transaction Network," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, DOI: 10.1109/TSMC.2025.3605054, 2025.
- [2] W-L. Li, R-F. Wang, and X. Luo, "A Generalized Nesterov-Accelerated Second-Order Latent Factor Model for High-Dimensional and Incomplete Data," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 36, no. 1, pp. 1518-1532, 2025.
- [3] M. Li, and Y. Song, "An Improved Non-Negative Latent Factor Model for Missing Data Estimation via Extragradient-Based Alternating Direction Method," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 34, no. 9, pp. 5640-5653, 2023.
- [4] X. Luo, Z-D. Wang, M-C. Zhou, and H. Yuan, "Latent Factor-Based Recommenders Relying on Extended Stochastic Gradient Descent Algorithms," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 2, pp. 916-926, 2021.
- [5] H-F. Liu, L-P. Jing, J. Yu, and M-K. Ng, "Social Recommendation with Learning Personal and Social Latent Factors," *IEEE Trans. on Knowledge and Data Engineering*, vol. 33, no. 7, pp. 2956-2970, 2021.
- [6] X. Luo, J-F. Chen, Y. Yuan and Z-D. Wang, "Pseudo Gradient-Adjusted Particle Swarm Optimization for Accurate Adaptive Latent Factor Analysis," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 54, no. 4, pp. 2213-2226, 2024.
- [7] M-W. Lin, X-Y. Lin, X-Q. Xu, Z-S. Xu, and X. Luo, "Neural Networks-Incorporated Latent Factor Analysis for High-Dimensional and Incomplete Data," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, DOI: 10.1109/TSMC.2025.3683919, 2025.
- [8] X-Y. Lin, S-B. Yu, M-W. Lin, X-Q. Xu, J-Y. Lin, and Z-S. Xu, "An Incremental Nonlinear Co-Latent Factor Analysis Model for Large-Scale Student Performance Prediction," *IEEE Trans. on Services Computing*, vol. 18, no. 6, pp. 3463-3476, 2025.
- [9] Y-L. Liu, "Signed Latent Factors for Spamming Activity Detection," *IEEE Trans. on Information Forensics and Security*, vol. 20, pp. 651-664, 2024.
- [10] M-S. Yan, Z-Y. Cheng, Y-H. Han, and M. Wang, "Latent Factor Modeling With Expert Network for Multi-Behavior Recommendation," *IEEE Trans. on Knowledge and Data Engineering*, vol. 37, no. 10, pp. 6020-6031, 2025.
- [11] Q. Wang, H. Wu, and X. Luo, "A Convolution Bias-Incorporated Non-negative Latent Factorization of Tensors Model for Accurate Representation Learning to Dynamic Directed Graphs," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, DOI: 10.1109/TSMC.2025.3611792, 2025.
- [12] Y-Q. Ma, R-N. Zheng, Y-H. Lu, Z. Zhang, Y-C. Wang, and Z-Z. Lin, "Electricity Retail Plan Recommendation Method Based on Multigranular Hesitant Fuzzy Sets and an Improved Non-Negative Latent Factor Model," *IEEE Trans. on Energy, Policy and Regulation*, vol. 2, no. 2, pp. 146-161, 2024.
- [13] X. Luo, M-C. Zhou, Y-N. Xia, and Q-X. Zhu, "An efficient non negative matrix-factorization-based approach to collaborative filtering for recommender systems," *IEEE Trans. on Industrial Informatics*, vol. 10, no. 2, pp. 1273-1284, 2014.
- [14] C. Wang, Q. Liu, R. Wu, and E. Chen, "Confidence-aware matrix factorization for recommender systems," in *Proc. of the 32nd AAAI Conf. on Artificial Intelligence*, 2018, pp. 434-442.
- [15] P. Liu, L. Zhang, and J-A. Gulla, "Multilingual Review-aware Deep Recommender System via Aspect-based Sentiment Analysis," *ACM Trans. on Information Systems*, vol. 39, no. 2, pp. 1-33, 2021.
- [16] D. Wu, Y-P. Hu, K-C. Liu, J. Li, X-M. Wang, S. Deng, N-G. Zheng, and X. Luo, "An Outlier-Resilient Autoencoder for Representing High-Dimensional and Incomplete Data," *IEEE Trans. on Emerging Topics in Computational Intelligence*, vol. 9, no. 2, pp. 1379-1391, 2025.
- [17] R-J. Ma, S-Y. Li, B. Zhang, and Z-M. Li, "Towards Fast and Robust Real Image Denoising with Attentive Neural Network and PID Controller," *IEEE Trans. on Multimedia*, vol. 24, pp. 2366-2377, 2021.
- [18] S-B. Yu, Y-F. Zeng, Y-H. Pan, and F. Yang, "SNMCF: A Scalable Non-Negative Matrix Co-Factorization for Student Cognitive Modeling," *IEEE Trans. on Knowledge and Data Engineering*, vol. 36, no. 7, pp. 3486-3500, 2024.
- [19] X-Q. Xu, M-W. Lin, W-Q. Li, J-P. Zhang, and H. Wu, "Time-varying QoS Estimation via Non-negative Latent Factorization of Tensors with Extended Linear Biases," in *Proc. of the IEEE Int. Conf. on Big Data*, DOI: 10.1109/BigData59044.2023.10386709, 2023.
- [20] Z-G. Liu, X. Luo, and Z-D. Wang, "Convergence analysis of single latent factor-dependent, nonnegative, and multiplicative update-based nonnegative latent factor models," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 32, no. 4, pp. 1737-1749, 2021.
- [21] D. Acharya, and D. Kumar, "Human Conception Optimizer-Based Optimal Type-2 Fuzzy PID Controller Design for Artificial Respiratory System," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 54, no. 9, pp. 5388-5399, 2024.
- [22] R-J. Ma, S-Y. Li, B. Zhang, and H-F. Hu, "Meta PID attention network for flexible and efficient real-world noisy image denoising," *IEEE Trans. on Image Process*, DOI: 10.1109/TIP.2022.3150294, 2022.
- [23] S-R. Lin, and X-W. Liu, "Output Synchronization and PID Control for Directed Networks with Multiple Communications," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 55, no. 3, pp. 1620-1633, 2025.
- [24] J. Cuellar, A-Y de Jesus, R. Troncoso, M. Velazquez, "PID-controller tuning optimization with genetic algorithms in servo system," *Int. J. Adv. Robot. Syst.*, vol. 10, no. 9, p. 324, 2013.
- [25] Z-H. Wang, and J. Zhang, "Incremental PID Controller-Based Learning Rate Scheduler for Stochastic Gradient Descent," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 35, no. 5, pp. 7060-7071, 2024.
- [26] Z-W. Ma, J-N. Cheng, and C. Lyu, "A Dynamic Gradient Enhanced Particle Swarm Optimization-Incorporated Adaptive Latent Factor Analysis Model," in *Proc. of the Int. Conf. on Systems, Man, and Cybernetics*, DOI: 10.1109/SMC58881.2025.11342693, 2025.
- [27] J-F. Zhang, J-W. Wang, H-K. Lam and H-X. Li, "Boundary Output Tracking of Nonlinear Parabolic Differential Systems via Fuzzy PID Control," *IEEE Trans. on Fuzzy Systems*, vol. 32, no. 12, pp. 6863-6877, 2024.
- [28] W-P. An, H-Q. Wang, Q-Y. Sun, J. Xu, Q-H. Dai, and L. Zhang, "A PID controller approach for stochastic optimization of deep networks," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2018, pp. 8522-8531.
- [29] Z-F. Wang, X-Y. Wang, G-L. Yang, Q-Q. Sun, and Y-Z. Ma, "Robust bounded control for fuzzy mechanical systems: Fuzzy optimal design and inequality constraint reorganize," *IEEE Trans. on Fuzzy Systems*, vol. 31, no. 8, pp. 2605-2616, 2023.
- [30] Y-Z. Wang, Z-D. Wang, L. Zhou, and H-L. Dong, "Observer-based fuzzy PID tracking control under try-once-discard communication protocol: An affine fuzzy model approach," *IEEE Trans. on Fuzzy Systems*, vol. 32, no. 4, pp. 2352-2365, 2024.
- [31] Q-L. Zhou, H-J. Che, W. Guo, X. He, M-F. Leung, and S-P. Wen, "Robust Low-Rank Tensor Constrained Orthogonal Symmetric Non-Negative Matrix Factorization for Multi-Layer Networks Community Detection," *IEEE Trans. on Emerging Topics in Computational Intelligence*, vol. 9, no. 6, pp. 4145-4160, 2025.
- [32] D. Mrozek, J. Kwiendacz, and B-M. Mrozek, "Protein Construction based Data Partitioning Scheme for alignment of Protein macromolecular structures though distributed querying in federated databases," *IEEE Trans. on NanoBioscience*, vol. 19, no. 1, pp. 102-116, 2020.
- [33] M. Zhou, F. Wang, R-J. Wang, and X. Wang, "PID controller-inspired model design for single image De-Raining," *IEEE Trans. on Circuits and System*, vol. 69, no. 4, pp. 2351-2355, 2022.
- [34] Y. Yuan, S-Y. Lu, and X. Luo, "A Proportional Integral Controller-Enhanced Non-Negative Latent Factor Analysis Model," *IEEE/CAA Journal of Automatic Sinica*, vol. 12, no. 6, pp. 1246-1259, 2025.
- [35] L. Dong, X-Q. Lu, G-C. Liu, and Y. Yuan, "A novel NMF guided for hyperspectral unmixing from incomplete and noisy data," *IEEE Trans. on Geoscience and Remote Sensing*, DOI: 10.1109/TGRS.2021.3101504, 2021.
- [36] Y-N. Li, R-L. Wang, Y-Q. Fang, M. Sun, and Z-K. Luo, "Alternating direction method of multipliers for convolutive non-negative matrix factorization," *IEEE Trans. on Cybernetics*, vol. 55, no. 12, pp. 7735-7748, 2023.
- [37] J. Dem˘sar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. 1, pp. 1-30,

- 2006.
- [38] L-T. Wu, P. Sun, Z-B. Wang, Y-J. Li, and Y. Yang, "Computation Off-loading in Multi-Cell Networks with Collaborative Edge-Cloud Computing: A Game Theoretic Approach," *IEEE Trans. on Mobile Computing*, vol. 23, no. 3, pp. 2093-2106, 2024.
 - [39] X-K. Wu, K. Shan, L. Wang, J-K. Wang, and M-S. Shang, "Spatiotemporal water quality data reconstruction: A tensor factorization framework," *Ecological Informatics*, vol. 90, p. 103283, 2025.
 - [40] S. Zhang, L. Yao, B. Wu, X. Xu, X. Zhang, and L. Zhu, "Unraveling metric vector spaces with factorization for recommendation," *IEEE Trans. on Industrial Informatics*, vol. 16, no. 2, pp. 732-742, 2020.
 - [41] X. Luo, Z-G. Liu, S. Li, M-S. Shang, and Z-D. Wang, "A fast non-negative latent factor model based on generalized momentum method," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 1, pp. 610-620, 2021.
 - [42] D. Hajinezhad, T-H. Chang, X-F. Wang, Q-J. Shi, and M-Y. Hong, "Non negative matrix factorization using ADMM: Algorithm and convergence analysis," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2016, pp. 4742-4746.
 - [43] J-L. Li, and Y. Yuan, "A nonlinear proportional integral derivative incorporated stochastic gradient descent-based latent factor model," in *Proc. of the Int. Conf. on Systems, Man, and Cybernetics*, 2020, pp. 2371-2376.
 - [44] L. Shi, Y-F. Zhang, W-G. Wang, J. Cheng, and H-Q. Lu, "Rethinking the PID optimizer for stochastic optimization of deep networks," in *Proc. of the Int. Conf. on Multimedia and Expo*, 2020, pp. 1-6.
 - [45] X-N. He, K. Deng, X. Wang, Y. Li, Y-D. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph convolution network for recommendation," in *Proc. of the Int. ACM SIGIR Conf. on research and development in Information Retrieval*, 2020, pp. 639-648.
 - [46] D. Liang, R-G. Krishnan, M-D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *Proc. of Int. Conf. on World Wide Web*, 2018, pp. 689-698.
 - [47] B-X. Weng, J. Sun, A. Sadeghi, and G. Wang, "ADAPID: An adaptive PID optimizer for training deep neural networks," in *Proc. of the Int. Conf. on Acoustics, Speech and Signal Processing*, DOI: 10.1109/ICASSP43922.2022.9746279, 2022.
 - [48] D-P. Kingma and J. Ba, "ADAM: A Method for stochastic optimization," in *Proc. of the Int. Conf. on learning representations*, 2015, pp. 303-309.
 - [49] X-Y. Guo, Y. Liu, D-F. Gong, F-L. Liu, "Dual Graph Convolutional Networks for Social Network Alignment," *IEEE Trans. on Big Data*, vol. 11, no. 2, pp. 684-695, 2025.
 - [50] J. Yang, Y-P. Li, G-Y. Wang, Z. Chen, and D. Wu, "An End-to-End Knowledge Graph Fused Graph Neural Network for Accurate Protein-Protein Interactions Prediction," *IEEE/ ACM Transactions on Computational Biology and Bioinformatics*, vol. 21, no. 6, pp. 2518-2530, 2024.
 - [51] J-L. Li, Y. Yuan, R. Tao, J. Chen, and X. Luo, "A proportional-integral derivative-incorporated stochastic gradient descent-based latent factor analysis model," *Neurocomputing*, vol. 427, pp. 29-39, 2020.
 - [52] Y-Z. Wang, Z-D. Wang, L. Zhou, L-F. Ma, and H-L. Dong, "Ultimately bounded PID control for T-S fuzzy systems under flexRay communication protocol," *IEEE Trans. on Fuzzy Systems*, vol. 31, no. 12, pp. 4308-4320, 2023.
 - [53] Y. Yuan, J-L. Li, and X. Luo, "A fuzzy PID-incorporated stochastic gradient descent algorithm for fast and accurate latent factor analysis," *IEEE Trans. on Fuzzy Systems*, vol. 32, no. 7, pp. 4049-4061, 2024.
 - [54] I. Cantador, P. Brusilovsky, and T. Kuflik, "Second workshop on information heterogeneity and fusion in recommender systems," in *Proc. of the ACM conf. on Recommender systems*, 2011, pp. 387-388.
 - [55] Y. Shi, M. Larson, and A. Hanjalic, "Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges," *ACM Computing Surveys*, vol. 47, no. 1, pp. 1-45, 2014.
 - [56] D. Szklarczyk, A-L. Gable, D. Lyon, A. Junge, S. Wyder, J-H. Cepas, M. Simonovic, N-T. Doncheva, and J-H. Morris, "STRING v11: Protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets," *Nucleic Acids Res.*, vol. 47, no. D1, pp. 607-613, 2019.
 - [57] H-J. Che, C-L. Li, M-F. Leung, D-Q. Ouyang, X-G. Dai, and S-P. Wen, "Robust Hypergraph Regularized Deep Non-Negative Matrix Factorization for Multi-View Clustering," *IEEE Trans. on Emerging Topics in Computational Intelligence*, vol. 9, no. 2, pp. 1817-1829, 2025.
 - [58] J. Hu, B. Hooi, S-S. Qian, Q. Fang, and C-S. Xu, "MGDCF: Distance learning via markov graph diffusion for neural collaborative filtering," *IEEE Trans. on Knowledge and Data Engineering*, DOI: 10.1109/TKDE.2023.3348537, 2024.
 - [59] W-Q. Fan, Y. Ma, Q. Li, J-P. Wang, G-Y. Cai, and J-L. Tang, "A graph

neural network framework for social recommendations," *IEEE Trans. on Knowledge and Data Engineering*, vol. 34, no. 5, pp. 2023-2047, 2022.



Jinli Li received the B.S. and M. S. degree in computer science and technology from China West Normal University, Nanchong, China, in 2021, and the Ph.D degree in the School of Computer Science and Technology from Chongqing University of Posts and Telecommunications, Chongqing, China, in 2025. She is currently a lecturer with the Xihua University, Chengdu, China. Her research interests focus on data analysis and control technology.



Ye Yuan (Member, IEEE) received the B.S. degree in electronic information engineering and the M.S. degree in signal processing from the University of Electronic Science and technology, Chengdu, China, in 2010 and 2013, and the Ph.D. degree in computer science from the University of Chinese Academy of Sciences, Beijing, China, in 2022. He is currently an Associate Professor with the College of Computer and Information Science, Southwest University, Chongqing, China. His research interests include data mining and machine learning.



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 (Fellow, 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 Distinguished Professor of Data Science and Computational Intelligence, and serving as the Dean of the College of Computer and Information Science, and School of Software, Southwest University, Chongqing, China. He has authored or coauthored over 400 papers (including over 190 *IEEE Transactions/Journal* papers) in the areas of Artificial Intelligence and Data Science, receiving 20,000+ Google Scholar citations with the H-Index of 87. Dr. Luo was the recipient of the Outstanding Associate Editor Award from *IEEE Access* in 2018, *IEEE/CAA Journal of Automatica Sinica* in 2020, and from *IEEE Transactions on Neural Networks and Learning Systems* in 2022-2024. He is currently serving as an Associate Editor for *IEEE Transactions on Neural Networks and Learning Systems*, and *IEEE/CAA Journal of Automatica Sinica*. His Google Scholar page is given at the link <https://scholar.google.com/citations?user=hyGIDs4AAAAJ&hl=zh-CN>.