A Nonlinear PID-Enhanced Adaptive Latent Factor Analysis Model

Jinli Li
School of Computer Science and Technology, Chongqing University of Posts and Telecommunications
Chongqing, 400065, China
appleli_li@163.com

Ye Yuan
College of Computer and Information Science
Southwest University
Chongqing, 400715, China
yuanyekl@gmail.com

Abstract—High-dimensional and incomplete (HDI) data holds tremendous interactive information in various industrial applications. A latent factor (LF) model is remarkably effective in extracting valuable information from HDI data with stochastic gradient decent (SGD) algorithm. However, an SGD-based LFA model suffers from slow convergence since it only considers the current learning error. To address this critical issue, this paper proposes a Nonlinear PID-enhanced Adaptive Latent Factor (NPALF) model with two-fold ideas: 1) rebuilding the learning error via considering the past learning errors following the principle of a nonlinear PID controller; 2) implementing all parameters adaptation effectively following the principle of a particle swarm optimization (PSO) algorithm. Experience results on four representative HDI datasets indicate that compared with five state-of-the-art LFA models, the NPALF model achieves better convergence rate and prediction accuracy for missing data of an HDI data.

Keywords—High-dimensional and incomplete Data, Latent Factor Analysis, Stochastic Gradient Descent, non-linear Proportional Integral Derivation, Particle Swarm Optimization, Parameter Adaptation.

I. INTRODUCTION

With the development of big data, the internet is releasing lots of data all the time, e.g., some big-data-related applications hold numerous users, items and massive amounts of interactive information [1-9]. However, as the number of users and items consistent growth, the proportion of interactive information is shrinking, e.g., each user only touches a small part of item, but fail to contact with richer items [3, 4]. Thus, it is highly incomplete of resultant interactions in many applications. In consequence High-Dimensional and Incomplete (HDI) matrix commonly used to characterize interactive information with a high deficiency rate [5, 6, 22].

Even though an HDI matrix exists a lot of blank terms, it still contains plentiful interactive results of users and items, which helps to calculate user preferences and attributes of items [26]. Hence, how to extract target information efficiently and accurately from known interactive data in HDI matrix becomes a focused issue and trend in research [14, 15, 17]. Much research indicated that latent factor analysis (LFA) model obtained great success in acquiring useful information from HDI data [17-21]. LFA model maps both users and items into identical low-dimensional LF space, and then calculate its low-rank approximation based on known data in HDI. In general, LFA has become the mainstream model for explaining HDI matrix by feat of its scalability and efficiency [20-22].

Stochastic gradient descent (SGD) method is commonly considered as one of the most suitable optimization algorithms for explaining LFA model [14, 26-29]. It utilizes negative gradient information of each point for helping objection convergence. However, a standard SGD algorithm only takes current gradient information into consideration instead of all known historical information of this point. Thus, SGD-based LF model consumes multiple iterations to converge and costs more training time and resources on large-scale datasets [26]. Therefore, many researchers are devoted to investigating how to accelerate the training speed based on SGD-based LFA without sacrificing the accuracy. To date, many studies have put forward a series of optimization algorithms based on SGD by incorporating past gradient information. For instance, a momentum-incorporated SGD algorithm, it considers both the past and present gradients to update the parameter [1, 26]. A Nesterov’s accelerated gradient (NAG)-incorporated SGD algorithm is a variant of momentum algorithm, but the gradient is estimated after the current velocity is applied [21]. In conclusion, above methods introduce history gradient information by adding momentum term.

It has been found that proportional-integral-derivative (PID) popularly employed in control field due to its simplicity, functionality and applicability [10-13]. It is a feedback-loop just with a single input, and it contains the proportion of current error, the integral of all history error, and the derivative of future trend (the difference between recent two errors) [16, 23, 31]. Inspired by PID method, a PID-incorporated SGD algorithm is proposed to improve the convergence rate without accuracy loss [25]. Specifically, it combines the current, past and future trend of error to rebuild a new error, and then replace current error of SGD with rebuild error. However, the above model cannot deal with nonlinear problems well. Thus, a Non-linear PID-incorporated SGD-based LF (NSLF) algorithm is proposed to solve a complex nonlinear problem [24]. However, there
are many gain parameters and hyper-parameters like lambda and learning rate requires grid search, this issue significantly affects model performance when processing an industrial HDI matrix.

Aiming at addressing the above issue, this paper innovatively proposes a Nonlinear PID-enhanced Adaptive Latent Factor (NPALF) algorithm with following two-fold ideas:

a) Replacing lambda times learning rate with a new constant $\phi$, then using NPID rebuilding the instant error, which replaces the original instant error in SGD-based LF model with rebuilt error to achieve fast convergence rate and competitive prediction accuracy for missing data of an HDI matrix; and;

b) Enabling $\phi$ and gain parameters of NPID adaptation in formula, thereby improving the efficiency of LFA model significantly.

Main contributions of this work as shown:

a) An NPALF-based LFA model. Using NPALF-based instant error control and PSO for adapting hyper-parameter and gain parameters of NPID, an NPALF-based LFA model achieves efficient convergence rate and reliable accuracy on HDI matrix.

b) Detail algorithm design and analysis for an NPALF-based LFA model. It explains that the NPALF algorithm makes LFA model obtain better results in terms of calculation and storage.

The rest part of this paper is organized as follows. Section II describes the preliminaries, Section III accounts for NPALF-based LFA model, Section IV reports the experiment and comparison of results, and finally, there are conclusions about this study in Section V.

II. PRELIMINARIES

A. Problem Statement

**Definition 1.** Given $M$ and $N$ denote the user and item sets respectively, $R^{M \times N}$ is rating matrix whose each value $r_{m,n}$ is user $m$‘s preference on item $n$. $A$ and $\Gamma$ denote the known and unknown entity sets of $R$, then $R$ is an HDI matrix if $A \cap \Gamma = \emptyset$.

**Definition 2.** Given $R$, an LFA model is its low-rank estimation $\hat{R} = XY^T$ built on $R$, where $X^{M \times f}$ and $Y^{N \times f}$ are the LF matrices corresponding to $M$ and $N$, and $f < \min \{ |M|, |N| \}$ denotes the rank of $R$.

Obviously, matrices $X$ and $Y$ are importance for an LFA model. In order to achieve them, an LFA model builds a learning objective to measure the difference between each node in $R_k$ and corresponding node in $\hat{R}$ [20, 21]. With Euclidean distance and regularization [15, 22], it is formulated as:

$$\mathcal{E} = \frac{1}{2} \sum_{m,n \in A} (r_{m,n} - \hat{r}_{m,n})^2 + \lambda \|x\|_2^2 + \lambda \|y\|_2^2,$$

where regularization constant $\lambda$ controls the regularization effect [22], and $\|\|_2$ computes the $l_2$ norm of a vector.

B. An SGD-based LF model

As indicated by prior studies [26], SGD algorithm relies it’s highly efficient when performing LFA of HDI matrices. Thus, with SGD method, the objective function (1) is shown as:

$$\arg \min_{X,Y} \mathcal{E} \Rightarrow \forall r_{m,n} \in A : \begin{bmatrix} x_m & \leftarrow & x_m - \eta \frac{\partial \mathcal{E}_{m,n}}{\partial x_m} \\ y_n & \leftarrow & y_n - \eta \frac{\partial \mathcal{E}_{m,n}}{\partial y_n} \end{bmatrix}$$

where $\eta$ denotes the learning rate; and the instant objective $\mathcal{E}_{m,n}$ is given as:

$$\mathcal{E}_{m,n} = \left( r_{m,n} - \langle x_m, y_n \rangle \right)^2 + \lambda \|x_m\|_2 + \lambda \|y_n\|_2. \tag{3}$$

By making $err_{m,n} = r_{m,n} - \langle x_m, y_n \rangle$, and folding (2)-(3), we achieve an SGD-based LFA model as follow:

$$\arg \min_{X,Y} \mathcal{E} \Rightarrow \forall r_{m,n} \in A : \begin{bmatrix} x_m & \leftarrow & x_m + \eta \langle \mathcal{E}_{m,n}, x_m \rangle - \lambda \cdot x_m \\ y_n & \leftarrow & y_n + \eta \langle \mathcal{E}_{m,n}, y_n \rangle - \lambda \cdot y_n \end{bmatrix} \tag{4}$$

C. An NPID Controller

**Fig. 1. Flowchart of NPID.**

The NPID controller exploits the present, past and future information of prediction error to calculate a correction and applies it to control a feedback system [32]. Unlike traditional PID, gain parameters are nonlinear functions with the instant error as the independent variable. Hence, NPID can adjust the control effect in real time according to the instant error, i.e., $K_p$, $K_i$, and $K_d$ in NPID is shown as:

$$K_p(x) = K_{p1} + K_{p2} \cdot \text{sech} \left( K_{p3} x \right),$$

$$K_i(x) = K_{i1} \cdot \text{sech} \left( K_{i2} x \right), \tag{5}$$

$$K_d(x) = K_{d1} + K_{d2} / \left( 1 + K_{d3} \exp \left( K_{d4} x \right) \right),$$

where $x$ represent the $i$-th instant error, $K_{pj} \in [K_{pj}, K_{pj}]$ and $K_{dj} \in [K_{dj}, K_{dj}]$ are proportional, integral and derivative parameters, nonlinear function $\text{sech}(x)$ is expressed as $2 / (\exp(x) + \exp(-x))$, respectively. With function (5), the NPID expression expands as follow:

$$\hat{y}^0 = \left( K_{p1} + K_{p2} \cdot \text{sech} \left( K_{p3} x^0 \right) \right) e^0 \tag{6}$$

$$+ K_{i1} \cdot \text{sech} \left( K_{i2} x^0 \right) \sum_{k=0}^{\infty} e^k$$

$$+ \left( K_{d1} + K_{d2} / \left( 1 + K_{d3} \exp \left( K_{d4} e^0 \right) \right) \right) (e^0 - e^{0+1})$$
D. A PSO Algorithm

Note that a standard PSO algorithm originated from the foraging behavior of bird flocks [30, 33]. In general, PSO builds a swarm with J particles, and those particles search for the optimal solution in a D-dimensional space. Each particle j has two intrinsic properties: velocity \( v_j \) and position vectors \( s_j \). Thus, \( v_j \) and \( s_j \) at the t-th iteration are vectors \( v_j^{(t)} = [v_j^{(t)}(1), v_j^{(t)}(2), \ldots, v_j^{(t)}(D)] \) and \( s_j^{(t)} = [s_j^{(t)}(1), s_j^{(t)}(2), \ldots, s_j^{(t)}(D)] \) where \( 1 \leq d \leq D \). Thus, its evolution scheme of the j-th particle at the t-th iteration is as follows:

\[
\forall j \in \{1, \ldots, J\} : \\
\begin{align*}
\{v_j^{(t)}\} &= wv_j^{(t-1)} + c_1r_1(b_j^{(t-1)} - s_j^{(t-1)}) + c_2r_2(g^{(t-1)} - s_j^{(t-1)}), \\
\{s_j^{(t)}\} &= s_j^{(t-1)} + v_j^{(t)};
\end{align*}
\]

where \( b_j^{(t-1)} \) and \( g^{(t-1)} \) denote the j-th particle’s local optimal position and the global position after the (t-1)-th iteration, \( s_j^{(t-1)} \) and \( v_j^{(t-1)} \) denotes the state of position and velocity after the (t-1)-th iteration, respectively.

III. METHOD

A. NPID-based Error Refinement

A NPID controller will make calculate a correction based on the instant error \( e^{(t)} \) between the prediction value and the true value. Starting from our problem, we handle each single instance \( r_{m,n} \in \Lambda \) with such refinement, and consider each iteration is a time point in NPID controller given in (8). Actually, we implement a refinement sequence of \( \Lambda \) by execute NPID controller of each one. Following such principle, the SGD-based learning principle (4) as follow:

\[
\forall r_{m,n} \in \Lambda : \\
\left\{ \\
\begin{align*}
x_n^{(t)} &\leftarrow x_n^{(t)} + \eta \cdot \hat{e}_{m,n}^{(t)} \cdot y_n^{(t)} - \lambda \cdot x_n^{(t)}, \\
y_n^{(t)} &\leftarrow y_n^{(t)} + \eta \cdot \hat{e}_{m,n}^{(t)} \cdot x_n^{(t)} - \lambda \cdot y_n^{(t)};
\end{align*}
\right.
\]

where \( \hat{e}_{m,n}^{(t)} \) represent the estimation error of the instance \( r_{m,n} \) at the t-th training iteration, also the t-th time point for an NPID controller. Thus, the NPID based error refining rule is shown as below:

\[
\hat{e}_{m,n}^{(t)} = (K_{p} + K_{p2}(1 - sech(K_{s}e_{m,n}^{(t)}))) \hat{e}_{m,n}^{(t-1)} + K_{s}sech(K_{s}e_{m,n}^{(t)}))\xi_{m,n}^{(t-1)} \\
+ \left(K_{d} + K_{d2}/\{1 + K_{d2}\exp(K_{d2}\xi_{m,n}^{(t-1)})\}\right) \xi_{m,n}^{(t-1)} - \hat{e}_{m,n}^{(t-1)};
\]

Formula (9) can be interpreted as follows:

a) \( \hat{e}_{m,n}^{(t)} \) represents the current training residual of an LFA;

b) \( \xi_{m,n}^{(t-1)} \) represents the history residual of \( r_{m,n} \) from the beginning till current;

c) \( \xi_{m,n}^{(t-1)} \) represent future trend of \( r_{m,n} \), it helps avoid overshooting.

By combining (8)-(9), we obtain the NPID incorporated SGD-based learning scheme for an LFA model as (10):

\[
\arg \min_{x,y} e^{(t)} \Rightarrow \\
\text{NPID-SGD:}
\]

During the t-th training iteration, \( \forall r_{m,n} \in \Lambda : \\
\hat{e}_{m,n}^{(t)} = \left\{ \left( K_{p} + K_{p2}(1 - sech(K_{s}e_{m,n}^{(t)}))\right) \hat{e}_{m,n}^{(t-1)} + K_{s}sech(K_{s}e_{m,n}^{(t)}))\xi_{m,n}^{(t-1)} + \left(K_{d} + K_{d2}/\{1 + K_{d2}\exp(K_{d2}\xi_{m,n}^{(t-1)})\}\right) \xi_{m,n}^{(t-1)} - \hat{e}_{m,n}^{(t-1)} \right\}.
\]

B. Self-adaptation of All Parameters

NPID-incorporated SGD-based LFA model (10) has nine gain parameters and two hyper-parameters need to adjust, which is time-consuming. This will limit the use of NPID-incorporated SGD learning scheme in large industrial application. Therefore, we introduce the PSO algorithm to NPID-incorporated SGD method for implementing the self-adaptation of all parameters.

We combine regularization coefficient \( \lambda \) times learning rate \( \eta \) into a new constant term \( \varphi \), then, we expand \( \hat{e}_{m,n}^{(t)} \) in (10) with NPID method, and multiply \( \eta \) into NPID method, equation (10) is modified as (11):

\[
\hat{e}_{m,n}^{(t)} = \left\{ \left( K_{p} + K_{p2}(1 - sech(K_{s}e_{m,n}^{(t)})))\hat{e}_{m,n}^{(t-1)} + K_{s}sech(K_{s}e_{m,n}^{(t)}))\xi_{m,n}^{(t-1)} + \left(K_{d} + K_{d2}/\{1 + K_{d2}\exp(K_{d2}\xi_{m,n}^{(t-1)})\}\right) \xi_{m,n}^{(t-1)} - \hat{e}_{m,n}^{(t-1)} \right\}.
\]

After that, we multiply the learning rate into the corresponding term. Meanwhile, we represent the parameters that need to be adapted by a set of unified symbols as (12):

\[
\hat{e}_{m,n}^{(t)} = \left\{ \left( \xi_{m,n}^{(t-1)} \right) \left( K_{p} + K_{p2}(1 - sech(K_{s}e_{m,n}^{(t-1)})))\hat{e}_{m,n}^{(t-1)} + K_{s}sech(K_{s}e_{m,n}^{(t-1)}))\xi_{m,n}^{(t-1)} + \left(K_{d} + K_{d2}/\{1 + K_{d2}\exp(K_{d2}\xi_{m,n}^{(t-1)})\}\right) \xi_{m,n}^{(t-1)} - \hat{e}_{m,n}^{(t-1)} \right\}.
\]

Formula (12) gives the complete NPID model with constant \( \varphi \). Thus, the NPID-incorporated SGD-based learning scheme depends on ten parameters, e.g. \( \hat{e}_{m,n}^{(t)}, \xi_{m,n}^{(t-1)}, \xi_{m,n}^{(t-1)}, \hat{e}_{m,n}^{(t-1)}, \hat{e}_{m,n}^{(t-1)}, \hat{e}_{m,n}^{(t-1)}, \hat{e}_{m,n}^{(t-1)}, \hat{e}_{m,n}^{(t-1)}, \hat{e}_{m,n}^{(t-1)} \) . Thus, we construct a
swarm with $J$ particles, which search for the optimal solution in $D$-dimensional space ($D=10$). And $j$-th particle maintains a set of parameters in an NPID controller applied to the same group of LFs. Then, the $j$-th particle’s position and velocity vectors are given as follow:

$$
s_j = \frac{s_{j,1}, s_{j,2}, s_{j,3}, s_{j,4}, s_{j,5}, s_{j,6}, s_{j,7}, s_{j,8}, s_{j,9}, s_{j,10}}{\hat{b}^{(t)}}
$$

$$
\dot{s}_j = \frac{\dot{s}_{j,1}, \dot{s}_{j,2}, \dot{s}_{j,3}, \dot{s}_{j,4}, \dot{s}_{j,5}, \dot{s}_{j,6}, \dot{s}_{j,7}, \dot{s}_{j,8}, \dot{s}_{j,9}, \dot{s}_{j,10}}{\hat{v}^{(t)}}
$$

By combining (7) with (13), we obtain the evolution process of parameters with PSO algorithm. As shown in (7), PSO changes the state of $j$-th particles based on $b_i$ and $g$ in each iteration, whose update process is as follow:

$$
\begin{align*}
\forall j \in \{1,..., J\}:
\begin{cases}
\dot{b}^{(t)}_j = \frac{b^{(t-1)}_j, F(b^{(t-1)}_j)}{\leq F(s^{(t)}_j)};
\dot{s}^{(t)}_j, F(s^{(t-1)}_j) > F(s^{(t)}_j),
\dot{g}^{(t)}_j = \frac{g^{(t-1)}_j, F(g^{(t-1)}_j)}{\leq F(s^{(t)}_j)};
\dot{s}^{(t)}_j, F(g^{(t-1)}_j) > F(s^{(t)}_j).
\end{cases}
\end{align*}
$$

In order to fitting the known set $A$ better, we adopt following two fitness function $F(.)$ for the $j$-th particle:

$$
\begin{align*}
F_1 (j) = \left( \sum_{m,n} |\gamma_{m,n} - \hat{\gamma}_{(j,m,n)}| \right) / |\Omega|,
F_2 (j) = \left( \sum_{m,n} |\gamma_{m,n} - \hat{\gamma}_{(j,m,n)}| \right) / |\Omega|,
\end{align*}
$$

where $|.|$ denote the absolute value of the data, $\Omega$ represents the prediction data, $\hat{\gamma}_{(j,m,n)}$ represents the $\gamma_{m,n}$ of $\gamma_{m,n}$.

After that, we set a range for position and velocity of each particle to ensure them is constrained in a certain range.

$$
\begin{align*}
\forall j \in \{1,..., J\}:
\begin{cases}
\dot{s}^{(t)}_j = \min \left\{ \hat{\dot{s}}, \max \left( \hat{\dot{s}}, s^{(t)}_j \right) \right\},
\dot{v}^{(t)}_j = \min \left\{ \hat{\dot{v}}, \max \left( \hat{\dot{v}}, v^{(t)}_j \right) \right\};
\forall d \in \{1,..., D\}
\end{cases}
\end{align*}
$$

where $\hat{\dot{s}}$ and $\hat{\dot{v}}$ denote upper and lower bounds of the particle’s velocity, $\hat{s}$ and $\hat{v}$ denote upper and lower bounds of the particle’s position. In general, we commonly have $\hat{v} = 0.01 (2^{-1} - 2^{-2})$, and $\hat{v} = \hat{v}$.

In addition, for $\forall j \in \{1,..., J\}$, $s_j$ is linked with the same group of LF matrices, i.e., $X$ and $Y$. Thus, each iteration contains $J$ sub-iterations. And in $j$-th sub-iteration of the $t$-th iteration, $X$ and $Y$ are trained as:

$$
\begin{align*}
\begin{cases}
\forall j \in \{1,..., J\}:
\begin{cases}
s_{x,j} \leftarrow (1 - K_{d,x}^{(t)}) s_{x,j} + \frac{K_{d,x}^{(t)} \sum_{j=1}^{J} \gamma_{m,n} - \hat{\gamma}_{(j,m,n)} ||}{1 + \sum_{j=1}^{J} \gamma_{m,n} - \hat{\gamma}_{(j,m,n)} ||},
\end{cases}
\end{cases}
\end{align*}
$$

where the subscript $(j)$ on $X$ and $Y$ represents their current states are linked with the $j$-th particle. According to formula (8)-(17), we obtain an NPALF-based LFA model.

C. NPALF Algorithm Design and Analysis

We design the algorithm for an NPALF-based LFA model as in Algorithm 1. According to Algorithm 1, it doesn’t bring much burden whether in time complexity or space complexity. And its time complexity is shown as follows:

$$
T = \Theta \left( \times J \times (|A| + |\Omega|) \times f \right).
$$

Note that the condition of (14) is $|A| > \max (|M|, |N|)$, and this condition is used in many applications to realize dimension reduction. Thus, the time complex of an NPALF-based LFA model is linear with $|A|$.

The storage complexity of Algorithm 1 depends on two important parts: a) the caches of LF matrices $X$ and $Y$ in SGD storage cost comes to $\Theta (|M| + |N|) \times f$; b) Auxiliary vector $\Psi$ and $v$ for caching the historical correction errors, whose storage cost comes to $\Theta (|A|)$; c) PSO’s auxiliary arrays $b, S, V$ and $F$, whose storage cost are far less than min $\{ |M|, |N| \}$. Thus, the storage complexity is:

$$
S = \left( (|M| + |N|) \times f + 2 \times |A| \right).
$$

Algorithm 1. NPALF-based LFA

| Operation | Cost |
|-----------|------|
| Input: $M$, $N$, $A$, $f$, $\hat{s}$, $\eta$, $\beta$, $\phi$, $c_{x, y}$, $c_{\theta}$, $J$ | |
| while not converge in $t$ iterations do | |
| for $j=1$ to $J$ | $\times J$ |
| for each $\gamma_{m,n}$ in $A$ | $\times |A|$ |
| fetch $y_{m,n}$ from $Y$, $b_{m,n}$ from $U$ | $\Theta (1)$ |
| $e_{m,n} = \gamma_{m,n} - y_{m,n} < c_{x, y} >$ | $\Theta (1)$ |
| $y_{m,n} = \min \{ y_{m,n}, e_{m,n} \}$ | $\Theta (1)$ |
| update $s_{x,j}$ and $y_{x,j}$ based on (17) | $\Theta (f)$ |
| $b_{m,n} = e_{m,n}$ | $\Theta (1)$ |
| end for | |
| end for | |
| for $j=1$ to $J$ | $\times J$ |
| compute $F_j$ based on (15) | $\Theta (|\Omega| \times f)$ |
| update $b_j$ and $g_j$ based on (14) | $\Theta (1)$ |
| evolve $S_j$ with (7) | $\Theta (1)$ |
| bound $S_j$ with (16) | $\Theta (1)$ |
| end for | |
| $t = t+1$ | $\Theta (1)$ |
| end while | |

Output: $X, Y$. 


IV. EXPERIMENTAL RESULTS AND ANALYSIS

A. General Settings

Evaluation Metrics. The root mean squared error (RMSE) is widely-adopted as evaluation metrics, the lower RMSE represent higher prediction accuracy:

\[
RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (r_{i} - \hat{r}_{i})^2}
\]

where \( r_{i,n} \) represents the prediction to \( r_{i,n} \) generated on test data, \( \Phi \) represents the testing set, \(|\cdot|\) calculates the cardinality of an enclosed set. At the same time, we record the convergent rounds and time-consuming in each descend. Note that all experiments are carried out on the same PC with a 3.2 GHz i5 CPU and 16 GB RAM. All tested models are implemented in JAVA SE 7U60.

| No. | Name                  | Row       | Column | Known Entries | Density |
|-----|-----------------------|-----------|--------|---------------|---------|
| D1  | MovieLens10M [24]     | 71,567    | 10,681 | 10,000,054    | 1.31%   |
| D2  | MovieLens20M [24]     | 138,493   | 26,744 | 20,000,263    | 0.54%   |
| D3  | Douban [25]           | 129,490   | 58,541 | 16,830,839    | 0.22%   |
| D4  | Jester [25]           | 124,113   | 150    | 5,865,235     | 31%     |

In order to acquire objective results, all known entry set of each HDI date is split into 10 disjoint and equally-sized subsets randomly, and choose seven subsets as the training set, one set as the validation set, remaining two sets as the testing set. And above process is repeated five times for five-fold cross-validation. The conditions of termination are: a) the number of training iterations exceeds preset threshold, i.e., 1000; b) the model converges, i.e., the error difference between two consecutive iterations is smaller than \( 10^{-5} \).

Moreover, there are several settings in our experiment for more compelling results:

a) The LF matrices \( X \) and \( Y \) are initialized with the same randomly generated arrays for all compared LFA models;

b) The regularization coefficients are set as 0.05 for SGD-based LFA model [2];

c) For balance the computational efficiency and representative ability of learning models, the dimension of LF space \( f \) is set as \( f=20 \) uniformly [8].

B. Comparison against State-of-the-art Models

| Model | Name                  | Description                           |
|-------|-----------------------|---------------------------------------|
| M1    | NPALF                 | The proposed NPALF -based LFA model.  |
| M2    | PSIL                  | A PID-incorporated SGD-based LFA model [25]. |
| M3    | SGD-LF                | A standard SGD-based LFA model [34].   |
| M4    | Adam                  | An Adam-based LFA model [34].          |
| M5    | AdaDelta              | An Ada-Delta-based LFA model [34].     |
| M6    | RMProp                | An RMProp-based LFA model [34].        |

| TABLE II. COMPARED MODELS IN OUR EXPERIMENTS |

In this section, we give the comparative experimental results of NPALF-based LFA model with several state-of-art LFA models. Table II records the details of all compared models. Table III records the detailed performance of M1-M6 on D1-D4. Figs. 2 depict the training curves of M1-M6. From those results, we have following findings:

a) M1’s prediction efficiency is significantly higher than other models. For instance, as depicted in Table III and Fig. 2, M1, i.e., a proposed NPALF model, just consumes 109.5 seconds to converge in RMSE on D2, compared with others, M1’s computational efficiency is improved about 26% (i.e., \((\text{Cost}_{\text{high}} - \text{Cost}_{\text{low}}) / \text{Cost}_{\text{high}}\)) than M2’s 149.4 seconds, 64% than M3’s 305.2 seconds, 93% than M4’s 1645.8 seconds, 98% than M5’s 6250.8 seconds, and 91% than M6’s 1265.6 seconds. However, we find that M1’s total time-consumption is higher than M2 in terms of RMSE on D1, this means that the NPALF model is data-dependent. Same outcomes are also seen on other datasets as depicted in Table III and Fig.2.

b) M1 also achieve quite satisfactory prediction accuracy for missing data of an HDI matrix when compared with other models. According to Table III, M1 implement the lowest prediction error on three testing cases out of four in total, and small gap in predictions in remaining test cases. For instance, M2’s RMSE is 0.7837 on D1, which is about 0.16% lower than M3’s 0.7929, 0.3% lower than M4’s 0.7939, 0.7% lower than M5’s 0.7939, 0.9% lower than M6’s 0.7939, and 3% lower than M1’s 0.7914. Moreover, similar results are proved on the other testing cases as shown in Table III.
V. CONCLUSIONS

This paper innovatively proposes an NPF model, which incorporates the NPID controller into an SGD-based LF model to achieve fast convergence rate and satisfactory prediction accuracy for missing data of an HDI matrix. Meanwhile, we adopt PSO method into NPF-based LFA model for implementing all parameter adaptation. Experimental results show that it improves prediction accuracy and computational efficiency significantly.

REFERENCES

[1] 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 System Man Cybernetics: Systems, vol. 51, no. 1, pp. 610-620, 2021.

[2] X. Luo, M-C. Zhou, S. Li, Y-N. Xia, Z-H. You, Q-S. Zhu, and H. Leung, “Incorporation of efficient second-order solvers into latent factor models for accurate prediction of missing QoS data,” IEEE Trans. on Cybernetics, vol. 48, no.4, pp. 1216-1228, 2018.

[3] X. Luo, Ye, Yuan, S-L. Chen, N-Y. Zeng, and Z-D. Wang, “Position-Transitional Particle Swarm Optimization-incorporated Latent Factor Analysis,” IEEE Trans. on Knowledge and Data Engineering, DOI: 10.1109/TKDE.2020.3033324.

[4] D. Wu, X. Luo, M-S. Shang, Y. He, G-Y. Wang, and X-D. Wu, “A Data-Characteristic-Aware Latent Factor Model for Web Services QoS Prediction,” IEEE Trans. on Knowledge and Data Engineering, DOI: 10.1109/TKDE.2020.3014302, 2020.

[5] D. Wu, and X. Luo, “Robust Latent Factor Analysis for Precise Representation of High-dimensional and Sparse Data,” IEEE/CAA Journal of Automatica Sinica, vol. 8, no. 4, pp. 796-805, 2021.

[6] X. Luo, H. Wu, and Z-C. Li, “NeuLFT: A Novel Approach to Nonlinear Canonical Polyadic Decomposition on High-Dimensional Infinite Tensors,” IEEE Trans. on Knowledge and Data Engineering, DOI: 10.1109/TKDE.2022.3174666, 2022.

[7] X. Luo, Y. Zhou, Z-G. Liu, and M-C. Zhou, “Fast and Accurate Non-negative Latent Factor Analysis on High-dimensional and Sparse Matrices in Recommender Systems,” IEEE Trans. on Knowledge and Data Engineering, DOI: 10.1109/TKDE.2021.3125252, 2021.

[8] X. Luo, H. Wu, Z. Wang, J-J. Wang, and D-Y. Meng, “A Novel Approach to Large-Scale Dynamically Weighted Directed Network Representation,” IEEE Trans. on Pattern Analysis and Machine Intelligence, DOI: 10.1109/TPAMI.2021.3132503, 2021.

[9] X. Luo, H. Wu, H. Yuan, M-C. Zhou, “Temporal pattern-aware QoS prediction via biased non-negative latent factorization of tensors,” IEEE trans. on cybernetics, vol. 50, no. 5, pp. 1798-1809, 2019.

[10] Q. Shi, H-K. Lam, and C-B. Xuan, “Adaptive neuro-fuzzy PID controller based on twin delayed deep deterministic policy gradient algorithm,” neurocomputing, vol. 402, pp. 183-194, 2020.

[11] J. Cuellar, A. Y; de Jesus, R. Troncoso, and M. Velazquez, “PID-controller tuning optimization with genetic algorithms in servo system,” Int. J. Adv. Robot. Syst., vol. 10, no. 9, pp. 324, 2013.

[12] J-D. Zhang, Z-J. Li, W-B. Meng, “The PID algorithm for multi-vehicles cruise control in platoon,” Journal of Information and Computational Science, vol. 12, no. 10, pp. 3899-3906, 2015.

[13] M. Li, Y. Zhang, D-Z. You, “Design of fuzzy PID steering motor controller based on particle swarm optimization,” in Proc. of the 3rd Conf. on Mechanical Engineering and Intelligent Manufacturing, pp. 449-453, 2020.

[14] X. Y. Shi, Q. He, X. Luo, Y. N. Bai, and M. S. Shang, “Large-scale and Scalable Latent Factor Analysis via Distributed Alternative Stochastic Gradient Descent for Recommender Systems,” IEEE Trans. on Big Data, DOI: 10.1109/TBDATA.2020.2973141, 2020.

[15] H. Wu, X. Luo, and M. C. Zhou, “Advancing Non-negative Latent Factorization of Tensors with Diversified Regularizations,” IEEE Trans. on Services Computing, DOI: 10.1109/TCST.2020.2989760, 2020.

[16] A-L. Salih, M. Moghavvemi, H-A. Mohamed, and K-S. Gaeid, “Modelling and pid controller design for a quadrotor unmanned vehicle,” in Proc. of the IEEE Int. Conf. on Automation, Quality and Testing, Robotics, pp. 1-5, 2010.

[17] X. Luo, Z. Ming, Z. You, S. Li, Y. Xia, and H. Leung, “Improving Network Topology-based Protein Interface Mapping via Collaborative Filtering,” Knowledge-Based Systems, pp. 23-32, 2015.

[18] D. Wu, Y. He, X. Luo, and M-C. Zhou, “A Latent Factor Analysis-based Approach to Online Sparse Streaming Feature Selection,” IEEE Trans. on System, Man, and Cybernetics: Systems, DOI: 10.1109/TSMC.2021.3096065, 2021.

[19] X. Luo, M. Shang, and S. Li, “Efficient Extraction of Non-negative Latent Factors from High-dimensional and Sparse Matrices in Industrial Applications,” in Proc. of the 16th ICDM, pp. 311-319, 2016.

[20] D. Wu, M. Shang, X. Luo, and Z. Wang, “An L1 and L2 Norm Oriented Factor Model for Recommender Systems,” IEEE Trans. on Neural Networks and Learning Systems, DOI: 10.1109/TNNLS.2021.3071392, 2020.

[21] X. Luo, Y. Zhou, Z. G. Liu, L. Hu, and M. C. Zhou, “Generalized Nesterov’s Acceleration- incorporated, Non-negative and Adaptive Latent Factor Analysis,” IEEE Trans. on Services Computing, DOI: 10.1109/TSC.2021.3069108, 2021.

[22] X. Luo, Z. D. Wang, and M. S. Shang, “An Instance-frequency-weighted Regularization Scheme for Non-negative Latent Factor Analysis on High Dimensional and Sparse Data,” IEEE Trans. on System Man Cybernetics: Systems, vol. 51, no. 6, pp. 3522-3532, 2021.

[23] L. Fan, E-M. Joo, “Design for auto-tuning PID controller based on genetic algorithms,” in Proc. of the 4th IEEE Conf. on Industrial Electronics and Applications, pp. 1924-1928, 2009.

[24] J-L. Li, Y. Yuan, “A Nonlinear Proportional Integral Derivative- Incorporated Stochastic Gradient Descent-based Latent Factor Model,” IEEE Int. Conf. on Systems, Man and Cybernetics, DOI: 10.1109/SMC2020.2020.9283344.

[25] J-L. Li, X-K. Wu, Y. Yuan, Y-J. Wu, K-K. M, Y. Zhou, “Accelerated Latent Factor Analysis for Recommender Systems via PID Controller,” in Proc. of the 31th ICNSC, DOI: 10.1109/ICNSC48988.2020.9238055, 2020.

[26] X. Luo, W. Qin, A. Dong, K. Sedraoui, and M. C. Zhou, “Efficient and High-quality Recommendations via Momentum- incorporated Parallel Stochastic Gradient Descent-based Learning,” IEEE/CAA Journal of Automatica Sinica, vol. 8, no. 2, pp. 402-411, 2021.

[27] X. Luo, Z. Liu, M. Shang, J Lou, and M-C. 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, 2020.

[28] X. Luo, Z. Liu, L. Jin, and M-C. Zhou, “Symmetric Non-negative Matrix Factorization-based Community Detection Models and Their Convergence Analysis,” IEEE Trans. on Neural Networks and Learning Systems, vol. 33, no. 3, pp. 1203-1215, 2022.

[29] S. Yao, Q. Kang, M. Zhou, M. Rawa, and A. Alabshri, “Discriminative Manifold Distribution Alignment for Domain Adaptation,” IEEE Trans. on Systems, Man, and Cybernetics: Systems, DOI: 10.1109/TSMC.2022.3195239, 2022.

[30] R. C. Eberhart and Y. H. Shi, “Particle swarm optimization: developments, applications and resources,” in Proc. of the Congress on Evolutionary Computation, pp. 81-86, 2001.

[31] Z. Li, S. Li, and X. Luo, “An Overview of Calibration Technology of Industrial Robots,” IEEE/CAA Journal of Automatica Sinica, vol. 8, no. 1, pp. 23-36, 2021.

[32] P. Zhao, J-J. Chen, Y. Song, X. Tao, T-J. Xu and T. Mei, “Design of a control system for an autonomous vehicle based on adaptive-pid,” International Journal of Advanced Robotic Systems, vol. 9, no. 2, pp. 44, 2012.

[33] Q. Deng, Q Kang, L. Zhang, M. Zhou, and J. An, “Objective Space-based Population Generation to Accelerate Evolutionary Algorithms for Large-scale Many-Objective Optimization,” IEEE Trans. on Evolutionary Computation, DOI: 10.1109/TEVC.2022.3168815, 2022.

[34] X. Luo, D-X 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, DOI: 10.1109/TSMC, 2018.