Neurocomputing 427 (2021) 29–39

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A proportional-integral-derivative-incorporated stochastic gradient descent-based latent factor analysis model ☆

Jinli Li [a], Ye Yuan [b], Tao Ruan [c], Jia Chen [d], Xin Luo [a,e,*]

[a] School of Computer Science and Technology, Dongguan University of Technology, Dongguan, Guangdong 523808, China
[b] Chongqing Engineering Research Center of Big Data Application for Smart Cities, and Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China, and University of Chinese Academy of Sciences, Beijing 100049, China
[c] China Patent Information Center, Beijing 100088, China
[d] School of Cyber Science and Technology, Beihang University, Beijing, 100191 China
[e] Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China

ARTICLE INFO

Article history:
Received 30 September 2020
Revised 27 October 2020
Accepted 15 November 2020
Available online 28 November 2020
Communicated by Zidong Wang

Keywords:
Big data
Stochastic gradient descent
Proportional integral derivation
PID controller
High-dimensional and sparse matrix
Latent factor analysis

ABSTRACT

Large-scale relationships like user-item preferences in a recommender system are mostly described by a high-dimensional and sparse (HiDS) matrix. A latent factor analysis (LFA) model extracts useful knowledge from an HiDS matrix efficiently, where stochastic gradient descent (SGD) is frequently adopted as the learning algorithm. However, a standard SGD algorithm updates a decision parameter with the stochastic gradient on the instant loss only, without considering information described by prior updates. Hence, an SGD-based LFA model commonly consumes many iterations to converge, which greatly affects its practicability. On the other hand, a proportional-integral-derivative (PID) controller makes a learning model converge fast with the consideration of its historical errors from the initial state till the current moment. Motivated by this discovery, this paper proposes a PID-incorporated SGD-based LFA (PSL) model. Its main idea is to rebuild the instant error on a single instance following the principle of PID, and then substitute this rebuilt error into an SGD algorithm for accelerating model convergence. Empirical studies on six widely-accepted HiDS matrices indicate that compared with state-of-the-art LFA models, a PSL model achieves significantly higher computational efficiency as well as highly competitive prediction accuracy for missing data of an HiDS matrix.

1. Introduction

In the era of data explosion, it becomes very difficult for people to acquire truly desired information from billions of bytes. In such a context, recommender systems (RSs) have attracted great interests since they perform efficient information filtering. In an RS, a user-item rating matrix is commonly considered as the fundamental data source [3–5] that describes a user's preference on an item according to his/her historical experiences. Meanwhile, due to the exploding user and item counts, it is impossible to obtain the full interactions between users and items. Under such circumstances, a high-dimensional and sparse (HiDS) matrix [1–3] is frequently adopted to describe their connections. For instance, LibimSeTi collects the Dating Agency matrix [45], which contains 17,359,346 observed interactions between 135,359 users and 168,791 profiles, whose dimension comes to 135,359 × 168,791 while data density is 0.076% only.

In spite of its high sparsity, an HiDS matrix contains various desired knowledge like users' potential favorites [4–6,20–24]. Hence, many models are proposed for acquiring knowledge from an HiDS matrix [8,11,14,46–48]. Considering existing models, a latent factor analysis (LFA)-based model is highly popular owing to its high scalability and efficiency [4,5,9,15–19]. The essentially principle of an LFA model is to map the row and column entities into the same low-dimensional LF space, and build a learning objective on an HiDS matrix's observed data related to desired LFs. This objective is then minimized with a learning algorithm to build these LFs for correctly representing an HiDS matrix. So far, many sophisticated LFA models have been proposed, like a probabilistic LFA model [8], a biased combination LFA model [9],

☆ This research is supported in part by the National Natural Science Foundation of China under grants 61772493, in part by the Guangdong Province Universities and College Pearl River Scholar Funded Scheme (2019), and in part by the Natural Science Foundation of Chongqing (China) under grant cstc2019jcyjjqX0013.

* Corresponding author.
E-mail address: luoxin21@gmail.com (X. Luo).

https://doi.org/10.1016/j.neucom.2020.11.029
0925-2312/

a neighbor correction LFA model [10], a fast nonparametric LFA model [7,11], and an inherently non-negative LFA model [13,14].

An LFA model often adopts a standard stochastic gradient descent (SGD) algorithm as its learning algorithm to optimize its non-convex learning objective with desired LFs [49,59]. However, such learning algorithm updates an LF relying on the stochastic gradient defined on the instant loss only, without considering information described by historical updates. Hence, an SGD-based LFA model commonly takes many iterations to converge [4,8,9,12,37,38], which results in enormous time cost on large-scale datasets.

On the other hand, a proportional-integral-derivative (PID) controller [30–34], which is often adopted in feedback control system [34–36,50], exploits the past, current and future information of prediction error to control a feedback system. Owing to such design, a PID controller has achieved great success in industrial applications [50–52,56–58]. Is it possible to incorporate the principle of PID into an SGD-based LFA model, thereby making it achieve high performance on an HiDS matrix? To answer this critical question, this paper presents a PID-incorporated SGD-based LFA (PSL) model. Its main idea is to rebuild the instant error on a single instance following the principle of PID, and then substitute this rebuilt error into an SGD algorithm for accelerating an LFA model's convergence. The contributions of this paper are as follows:

a) A PSL model that incorporates the principle of PID into its SGD-based learning scheme, thereby successfully combines the information of past, current and future updates to achieve high performance;
b) Detailed algorithm design and analysis for a PSL model.

Extensive experiments on six widely-accepted HiDS matrices indicate that compared with state-of-the-art LFA models, a PSL model achieves significantly higher computational efficiency with highly competitive prediction accuracy for missing data of an HiDS matrix.

Section 2 gives the preliminaries. Section 3 presents the methods. Section 4 reports the experimental results. And finally, Section 5 draws the conclusions.

2. Preliminaries

2.1. Problem statement

An HiDS matrix is adopted as an LFA model's input, which is defined as:

***Definition* 1.** Let $R^{|M|\times|N|}$ quantify certain relationships among M and N, which are two large entity sets corresponding to row and column, respectively. Λ and Γ denote the known and unknown entity sets of R. R is an HiDS matrix if $|\Lambda|\ll|\Gamma|$.

***Definition* 2.** Give R, an LFA model is commonly expressed in the form of $\hat{R}=PQ^{\mathrm{T}}$ with $\hat{R}$ denoting R's rank-f approximation built on Λ only, $P^{|M|\times f}$ and $Q^{|N||\times f}$ denoting the LF matrices describing M and N, and f denoting the LF dimension as $f \ll \min\{|M|, |N|\}$.

In order to obtain P and Q, we commonly adopt the Euclidean distance to build a learning objective defined on Λ as follows:

$$\varepsilon(P,Q)=\sum_{r_{m,n}\in\Lambda}\left(r_{m,n}-\sum_{k=1}^{f}p_{m,k}q_{n,k}\right)^2. \tag{1}$$

Note that (1) is ill-posed on an imbalanced HiDS matrix. Therefore, regularization terms are incorporated into the learning objective (1) for avoiding overfitting [35,36,38]. In practice, Tikhonov regularization is commonly adopted. With it, the objective (1) is extended into:

$$\varepsilon(P,Q)=\sum_{r_{m,n}\in\Lambda}\left((r_{m,n}-\hat{r}_{m,n})^2+\lambda_P\sum_{k=1}^{f}p_{m,k}^2+\lambda_Q\sum_{k=1}^{f}q_{n,k}^2\right), \tag{2}$$

where $\hat{r}_{m,n}=\sum_{k=1}^{f}p_{m,k}q_{n,k}$, λ_P and λ_Q are regularization coefficients for LF matrices P and Q, respectively.

2.2. An SGD-based LFA model

As indicated by prior research [8,9,12,49], SGD is an efficient and commonly adopted learning algorithm for building an LFA model. It enables a simple and efficient learning scheme for desired LFs. With it, the learning scheme for desired LFs in P and Q in (2) is given as:

$$\operatorname*{argmin}_{P,Q}\varepsilon(P,Q)\overset{SGD}{\Rightarrow}\forall r_{m,n}\in\Lambda,k\in\{1,2,\cdots,f\}:\begin{cases}p_{m,k}^{t+1}=p_{m,k}^{t}-\eta\frac{\partial\varepsilon_{m,n}^{t}}{\partial p_{m,k}^{t}},\\ q_{n,k}^{t+1}=q_{n,k}^{t}-\eta\frac{\partial\varepsilon_{m,n}^{t}}{\partial q_{n,k}^{t}};\end{cases} \tag{3}$$

where η denotes the learning rate [25–30], t denotes the update point, and $\varepsilon_{m,n}^{t}$ denotes the partial objective on the training instance $r_{m,n}\in\Lambda$, respectively. Note that $\varepsilon_{m,n}^{t}$ is given as:

$$\varepsilon_{m,n}^{t}=\left(r_{m,n}-\hat{r}_{m,n}^{t}\right)^2+\lambda_P\sum_{k=1}^{f}\left(p_{m,k}^{t}\right)^2+\lambda_Q\sum_{k=1}^{f}\left(q_{n,k}^{t}\right)^2. \tag{4}$$

Let $\tau_{m,n}^{t}=r_{m,n}-\hat{r}_{m,n}^{t}$, the stochastic gradients in (3) are deduced as:

$$\begin{cases}\frac{\partial\varepsilon_{m,n}^{t}}{\partial p_{m,k}^{t-1}}=-\tau_{m,n}^{t}\cdot q_{n,k}^{t}+\lambda_P\cdot p_{m,k}^{t},\\ \frac{\partial\varepsilon_{m,n}^{t}}{\partial q_{n,k}^{t-1}}=-\tau_{m,n}^{t}\cdot p_{m,k}^{t}+\lambda_Q\cdot q_{n,k}^{t}.\end{cases} \tag{5}$$

By combining (5) and (3), we achieve the standard SGD-based learning scheme for LFs in P and Q:

$$\begin{cases}p_{m,k}^{t+1}=p_{m,k}^{t}+\eta\cdot\left(\tau_{m,n}^{t}\cdot q_{n,k}^{t}-\lambda_P\cdot p_{m,k}^{t}\right),\\ q_{n,k}^{t+1}=q_{n,k}^{t}+\eta\cdot\left(\tau_{m,n}^{t}\cdot p_{m,k}^{t}-\lambda_Q\cdot q_{n,k}^{t}\right).\end{cases} \tag{6}$$

2.3. A PID controller

The essential idea of a PID controller is to firstly calculate the instant error, i.e., the difference between the true and predicted values, and then applies a correction to this error based on propor-

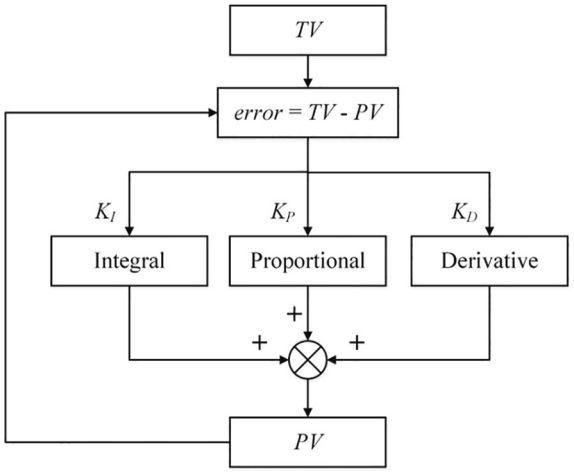


Fig. 1. Flowchart of PID.

tional (P), integral (I), and derivative (D) terms. Note that the update points of an SGD-based LFA model are discrete as shown in (6). Hence, a discrete PID controller is compatible with it. The flowchart of a discrete PID controller is shown in Fig. 1.

As shown in Fig. 1, *TV* and *PV* denote the true and predicted values, respectively. On arrival of *TV* and *PV*, a discrete PID controller calculates the instant error E_k, and then adjusts it as follows:

$$\tilde{E}_k = K_P E_k + K_I \sum_{i=0}^{k} E_i + K_D(E_k - E_{k-1}), \tag{7}$$

where K_P, K_I and K_D denote the controlling coefficients of proportional, integral and derivative terms, E_i denotes the instant error at the i-th update point, and $E\sim_k$ denotes the adjusted error, respectively. Based on the adjusted error, *PV* will be rebuilt and returned to the controller. This process continues till the termination condition is fulfilled, i.e., an LFA model converges in our context.

3. Methods

3.1. A PSL model

As unveiled by [4,9,12,37,38], an SGD-based LFA model takes many iterations to converge. Although the time cost per iteration of an SGD algorithm is low, its total time cost can be considerable due to its many iterations. Thus, it is essential to accelerate the convergent process of an SGD-based LFA model.

From (6), we see that at each update point of an SGD-based LFA model, the information of a training instance $r_{m,n}$ is passed into the model through the instant error, i.e., $\tau^t_{m,n} = r_{m,n} - \hat{r}^t_{m,n}$ only. Hence, this process can be described by a simplified PID controller where the integral and derivative terms are omitted by setting $K_I = K_D = 0$. From this point of view, this model can be probably enhanced by taking these two terms into consideration.

Based on the above inference, we extend the instant error with integral and derivative terms following the principle of PID given in (7) as follows:

$$\tilde{\tau}^t_{m,n} = K_P \tau^t_{m,n} + K_I \sum_{i=0}^{k} \tau^i_{m,n} + K_D\left(\tau^t_{m,n} - \tau^{t-1}_{m,n}\right). \tag{8}$$

The effect of (8) can be illustrated from the aspect of 'learning residual' by a learning algorithm:

a) $\tau^t_{m,n}$ is the current learning residual by the algorithm;
b) $\sum_{i=0}^{k} \tau^i_{m,n}$ describes the past learning residuals connected with the instance $r_{m,n}$. It adjusts the learning direction of the whole model to dampen oscillations for fast convergence, which functions similarly as a momentum method [54,55];
c) $\left(\tau^t_{m,n} - \tau^{t-1}_{m,n}\right)$ reflects the instant change of learning residual. It exploits future expectations of the model to avoid overshooting.

Note that the PID control in (8) is implemented with respect to each training instance and its corresponding instant error. By replacing the instant error $\tau_{m,n}$ in (6) with the adjusted error given in (7), the following learning scheme is achieved:

$$\begin{cases} p^{t+1}_{m,k} = p^t_{m,k} + \eta \cdot \left(\tilde{\tau}^t_{m,n} \cdot q^t_{n,k} - \lambda_P \cdot p^t_{m,k}\right), \\ q^{t+1}_{n,k} = q^t_{n,k} + \eta \cdot \left(\tilde{\tau}^t_{m,n} \cdot p^t_{m,k} - \lambda_Q \cdot q^t_{n,k}\right). \end{cases} \tag{9}$$

Note that the PID-incorporated learning scheme (9) enables a PSL model. Its flowchart is depicted in in Fig. 2.

3.2. Algorithm design and analysis

Based on the above section, we design the Algorithm PSL. In each iteration, it traverses Λ to access its each instance, and trains related LFs following the PID-incorporated learning scheme (9). As described in Algorithm PSL, the model's computational cost of a PSL model is:

$$\begin{aligned} T_{PSL} &= \Theta((|M| + |N|) \times f + (n \times |\Lambda| \times f)) \\ &\approx \Theta(n \times |\Lambda| \times f) \end{aligned} \tag{10}$$

Note that the condition of $|\Lambda| \gg \max\{|M|, |N|\}$ is commonly fulfilled even in an HiDS matrix. Hence, (10) reasonably omits the constant coefficients and lower-order terms to achieve the final result. It evidently indicates that the computational cost of a PSL model is linear with an HiDS matrix's known entry count, which is easy to resolve in practice.

Considering its storage complexity, it mainly depends on two factors:

a) LF matrices P and Q, whose storage cost sums to $\Theta((|M|+|N|) \times f)$; and
b) Auxiliary vectors Ψ and Yto implement the PID control of the instant error.

Thus, the storage cost of Algorithm PSL comes to:

$$S_{PSL} = ((|M| + |N|) \times f + 2 \times |\Lambda|). \tag{11}$$

which is linear with the size of involved entity sets and the target HiDS matrix's known entry count. Such storage burden is also easy to resolve for real applications.

Algorithm PSL

Input: M, N, Λ, f, λ_P, λ_Q, η, K_P, K_I, K_D, N	
Operation	**Cost**
init $P^{\|M\|\times f}$, $Q^{\|N\|\times f}$ with random numbers in [−0.01, 0.01]	$\Theta((\|M\|+\|N\|) \times f)$
init Ψ, Ywith size $\|\Lambda\|$	$\Theta(\|_\Lambda\|)$
for each $r_{m,n}$ **in** Λ	$\times\|\Lambda\|$
init $\psi_{m,n} = 0$ in Ψ, $\upsilon_{m,n} = 0$ in υ	$\Theta(\|_1\|)$
end for	–
while not converge **and** $n \le N$ **do**	$\times n$
for each $r_{m,n}$ **in** Λ	$\times\|\Lambda\|$
fetch $\psi_{m,n}$ from Ψ, $\upsilon_{m,n}$ from υ	
$\hat{r}_{m,n} = \sum_{k=1}^{f} p_{m,k} q_{n,k}$	$\Theta(f)$
$\tau_{m,n} = r_{m,n} - \hat{r}_{m,n}$	$\Theta(1)$
$\psi_{m,n} = \psi_{m,n} + \tau_{m,n}$	
$\tilde{\tau} = K_P \times \tau_{m,n} + K_I \times \psi_{m,n} + K_D \times (\tau_{m,n} - \upsilon_{m,n})$	$\Theta(1)$
$\upsilon_{m,n} = \tau_{m,n}$	$\Theta(1)$
for $k = 1$ **to** f	$\times f$
$p_{m,k} = p_{m,k} + \eta \times (\tilde{\tau} \times q_{n,k} - \lambda_P \times p_{m,k})$	$\Theta(1)$
$q_{n,k} = q_{n,k} + \eta \times (\tilde{\tau} \times p_{m,k} - \lambda_Q \times q_{n,k})$	$\Theta(1)$
end for	–
end for	–
$n = n+1$	$\Theta(1)$
end while	–
Output: P, Q	

4. Experimental results and analysis

4.1. General settings

Evaluation Metrics. For industrial applications [4,6,9,24], one major motivation to analyze an HiDS matrix is to recover the full connections among involved entities. Root mean squared error (RMSE) and mean absolute error (MAE) are adopted to evaluate the prediction accuracy by a tested model:

$$\begin{cases} RMSE = \sqrt{\left(\sum_{r_{u,v}\in\Gamma}(r_{u,v}-\hat{r}_{u,v})^2\right)/|\Gamma|}, \\ MAE = \left(\sum_{r_{u,v}\in\Gamma}|r_{u,v}-\hat{r}_{u,v}|_{abs}\right)/|\Gamma|; \end{cases} \tag{12}$$

where $\hat{r}_{u,v}$ is the prediction for $r_{u,v}$ generated by a tested model. Note that low RMSE and MAE mean high prediction accuracy of a tested model. Note that for acquiring objective results, all experiments are carried out on the same PC with a 3.2 GHz i5 CPU and 8 GB RAM. All compared models are implemented in JAVA SE 7U60.

Datasets. Six HiDS matrices are included in our experiments, whose details are provided below.

a) **D1: MovieLens 10 M**. The GroupLens research team collects this dataset from the MovieLens system. It contains 10,000,054 rating in the scale of [1, 5], 10,681 movies and 71,567 users. The density is 1.31%.
b) **D2: MovieLens 20 M**. The GroupLens research team collects this dataset from the MovieLens system. Its rating scale is [0.5, 5] and it contains 20,000,263 known ratings by 138,493 users on 26,744 movies. Its density is 0.54%.
c) **D3: Douban**. The Chinese largest online book, movie and music database Douban collect it. The rating scale of this dataset is [1, 5]. It contains 16,830,839 known ratings by 129,490 users and 58,541 items. Its density is only 0.22%.
d) **D4: Jester**. It has 6,500,000 anonymous ratings of jokes by users of the Jester Joke Recommender System. Its values from [-10, 10] of 100 jokes from 73,421 users.
e) **D5: Flixter**. The Flixter commercial website collects this dataset, which contains 8,196,077 ratings in [0.5, 5] from 147,612 users on 48,794 movies. Its data density is 0.11%.
f) **D6: Extended Epinion**. It is collected from Trustlet website. This dataset's rating scale is [1, 5] and contains 13,668,320 known entries by 120,492 users on 775,760 articles. Its density is only 0.015%.

Note that all datasets included in this paper are also extremely sparse and high-dimensional. Hence, the experimental results on them are highly representative.

All the datasets are randomly split into five disjoint and equally-sized subsets. In our experiments, we adopt the 80%-20% train-test settings. For obtaining objective results, five-fold cross-validations are applied, i.e., each time four subsets are selected as the training set Λ to train a model to predict the remaining one subset as the testing set Γ. This process is sequentially repeated for five times to obtain the final results. The training process of a compared model terminates if a) the number of consumed iterations reaches 1000, and b) the error difference between two

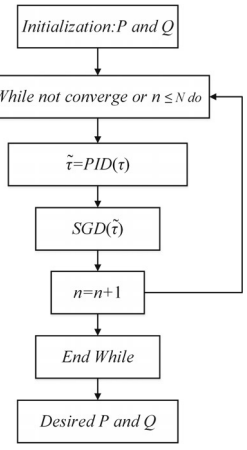


Fig. 2. Flowchart of PSL.

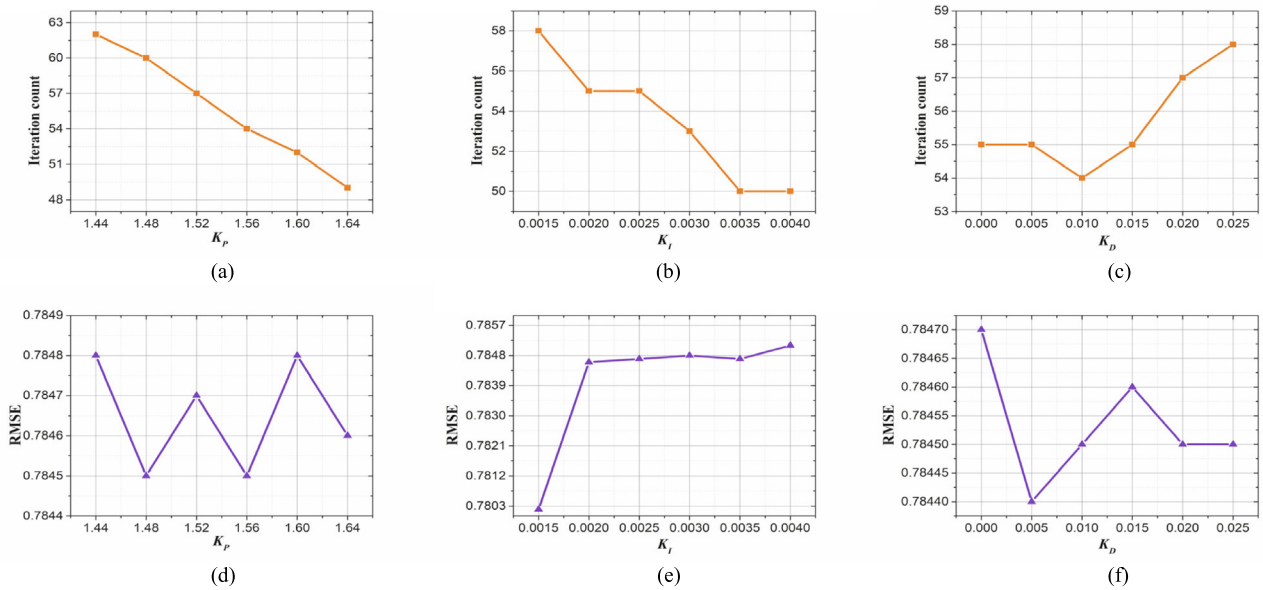


Fig. 3. A PSL model's parameter sensitivity test on D1.

consecutive iterations is smaller than 10^{-5}. Moreover, the following settings are adopted to ensure objective evaluations:

a) Regularization items are added to all models to avoid overfitting. We set $\lambda_P = \lambda_Q = 0.05$ uniformly.
b) The LF space dimension f is set at 20 uniformly.
c) For an SGD-based LFA model, the learning rate η is set at 0.04 following prior research [4,9,12,37,38,49].

4.2. Parameter sensitivity

In this set of experiments, we test the effects of K_P, K_I and K_D on the performance of a PSL model. During each test, we fix two parameters and make the remaining one vary to see its effects. The results are given in Figs. 3–8. From them, we have the following findings (Fig. 9):

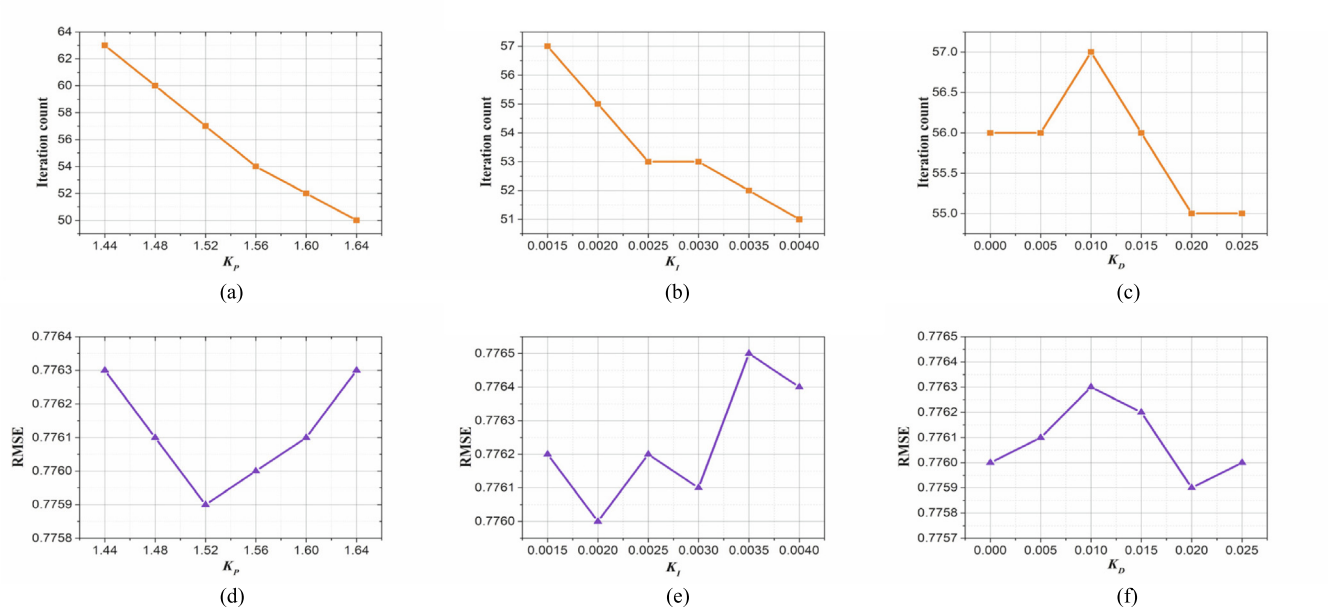


Fig. 4. A PSL model's parameter sensitivity testing on D2.

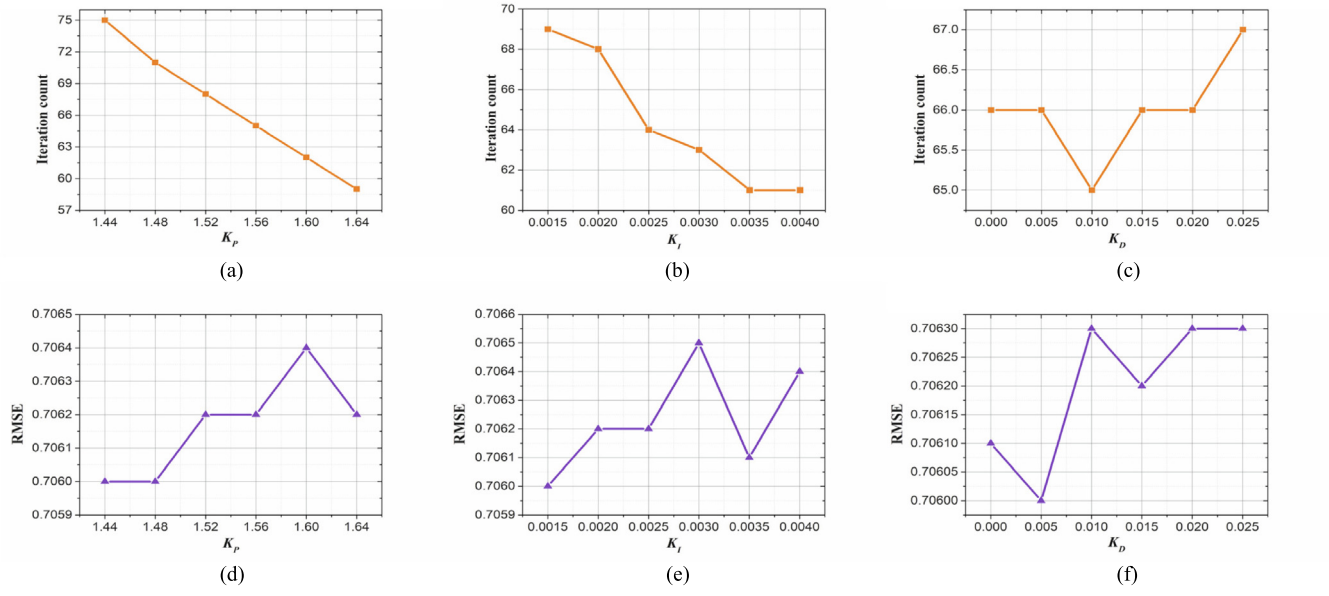


Fig. 5. A PSL model's parameter sensitivity testing on D3.

a) **K_P affects the convergence rate of a PSL model.** For instance, as shown in Fig. 3(a), on D1 a PSL model converges faster as K_P increases. However, it has little influence on prediction accuracy according to Fig. 3(d). Similar phenomena also occur on D2-6 as in Figs. 4-8.

b) **K_I affects the convergence rate and prediction accuracy of a PSL model.** For instance, as shown in Fig. 3(b), on D1 a PSL model consumes less iterations to converge as K_I increases. However, from Fig. 3(e) we further see that K_I affects the prediction accuracy of a PSL model. As K_I increases, a PSL model generally achieves its lowest RMSE, and then its RMSE increases again. Similar phenomena are also encountered on D2-6 as shown in Figs. 4–8.

c) **K_D's effect on a PSL model's performance is data-dependent.** As shown in Figs. 3–6 and 8, K_D has slight effects on PSL's convergence rate and prediction accuracy. Nonetheless, as shown in Fig. 7, on D5 it has significant effect on a PSL model's convergence rate without impairing its accuracy.

d) To summarize, a PSL model's performance is sensitive with K_P, K_I and K_D. They should be chosen with care.

4.3. Comparison against state-of-the-art LFA models

In this set of the experiments, the proposed PSL is proposed with several state-of-the-art LFA models. The details of compared models are summarized in Table 1. To derive objective and

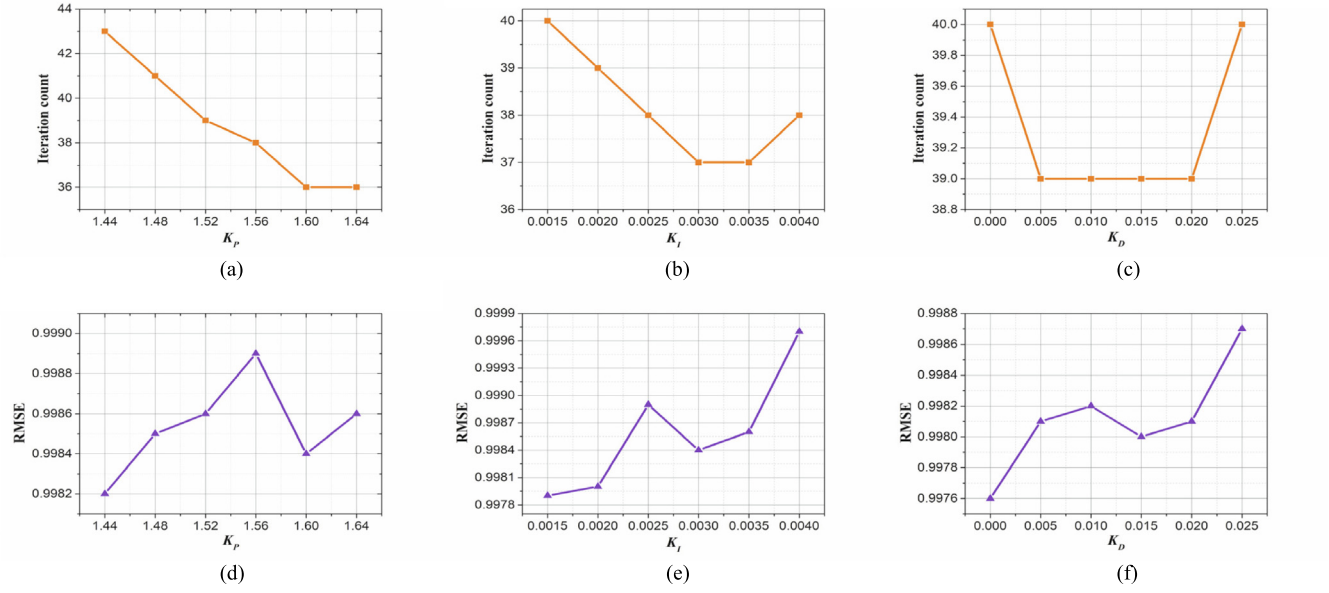


Fig. 6. A PSL model's parameter sensitivity testing on D4.

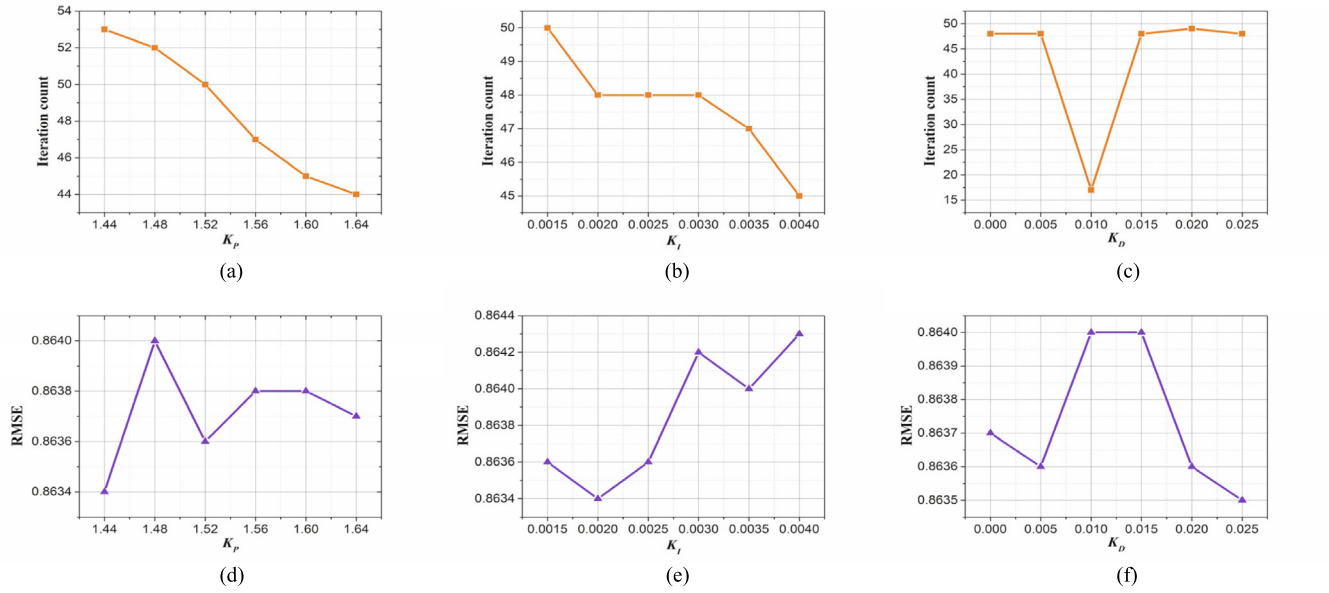


Fig. 7. A PSL model's parameter sensitivity testing on D5.

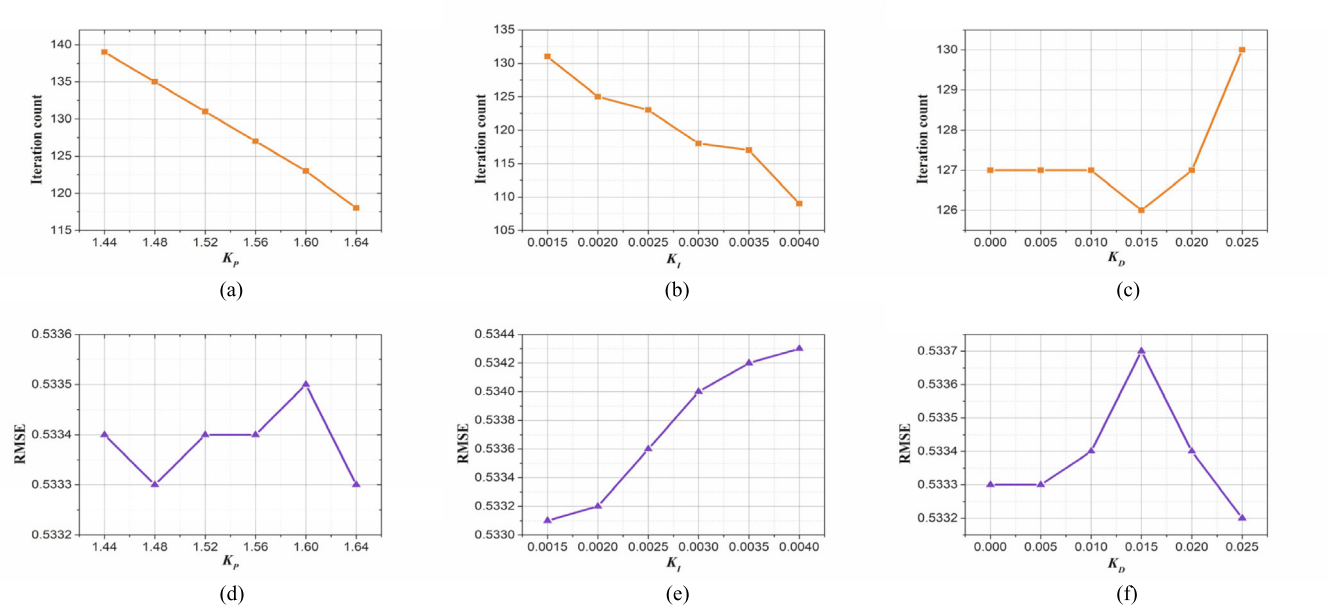


Fig. 8. A PSL model's parameter sensitivity testing on D6.

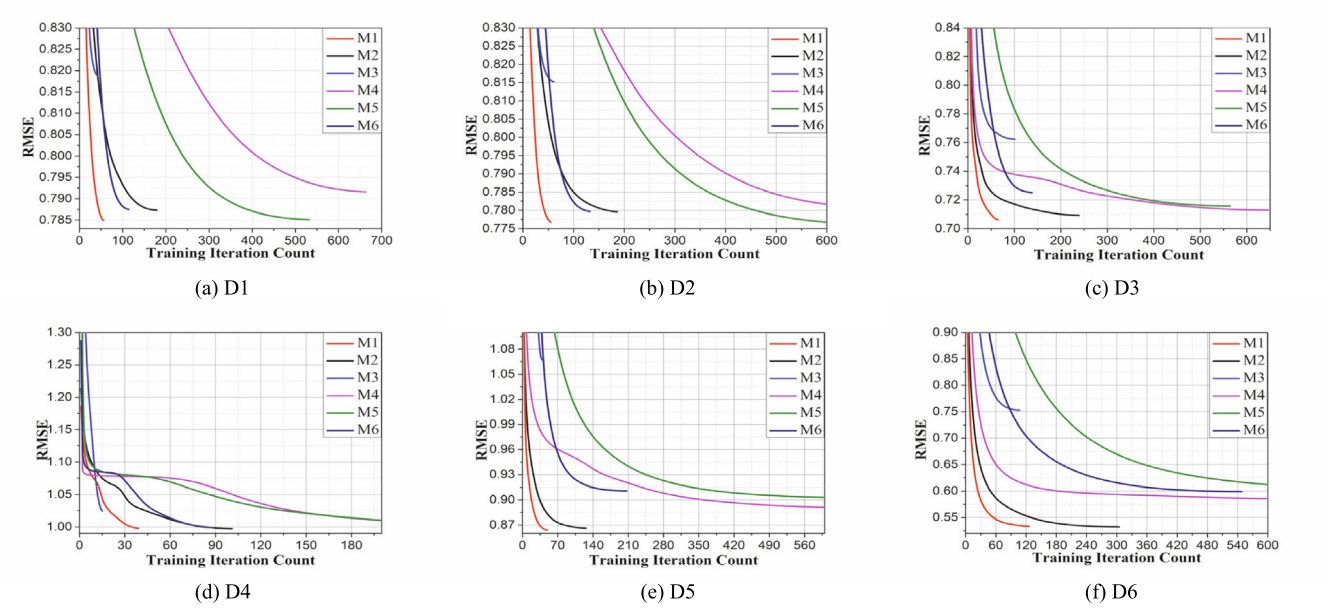


Fig. 9. Training curves of M1-M6 on D1-6 in RMSE.

unbiased results, all compared models are initialized with the same randomly generated arrays for eliminating the performance bias. Table 2 summarizes the time costs of compare models. Table 2 summarizes their RMSE, MAE and iteration counts. Fig. 10 depicts their training curves. From them, we conclude:

a) **A PSL model's computational efficiency is much higher than its peers.** For instance, as depicted in Table 2, M1 consumes 52 s to converge in RMSE on D1. Its time cost is 39.09% of 133 s by M2, 9.88% of 526 s by M3, 2.16% of 2405 s by M4, 2.74% of 1899 s by M5, and 16.83% of 309 s by M6. The same results are also found on the other testing cases.

b) **A PSL model's prediction accuracy is highly competitive.** As shown in Table 3, on D1, D2, D3 and D5, M1's RMSE is the lowest among that of its peers. On the other testing cases, it also ranks top three among its peers in terms of prediction accuracy.

4.4. Significance analysis

To better understand the comparison results, we perform the significance analysis based on Tables 2 and 3. As unveiled by prior work [39], Friedman test is efficient in testing multiple models's performance on multiple datasets. Assuming r_i^j is the rank of the jth of k compared models on the ith of N testing cases. The Fried-

Table 1
rime costs of tested models on d1-6.

Case		M1	M2	M3	M4	M5	M6
D1	RMSE-Time	52 ± 1.56 56 ± 1.58	133 ± 0.91	526 ± 2.73	2405 ± 12.93	1899 ± 12.63	309 ± 5.01 599 ± 8.33
	MAE-Time		148 ± 0.94	637 ± 6.88	2726 ± 13.82	3495 ± 31.71	
D2	RMSE-Time	93 ± 1.92 105 ± 1.96	260 ± 1.26	1426 ± 9.63	4754 ± 16.87	4678 ± 36.42	710 ± 9.05 1431 ± 26.58
	MAE-Time		334 ± 1.58	1631 ± 12.06	8453 ± 45.32	9190 ± 75.18	
D3	RMSE-Time	120 ± 2.03	333 ± 1.58	1683 ± 12.94	3733 ± 25.27	3540 ± 32.64	657 ± 8.87 1194 ± 22.43
	MAE-Time	127 ± 2.27	372 ± 1.62	1858 ± 15.26	4225 ± 28.56	6271 ± 69.55	
D4	RMSE-Time	3 ± 0.22 3 ± 0.22	7 ± 0.17 9 ± 0.28	27 ± 0.32	118 ± 2.95	201 ± 4.63	26 ± 0.97 58 ± 1.52
	MAE-Time			30 ± 0.54	144 ± 3.68	227 ± 5.01	
D5	RMSE-Time	35 ± 1.02 44 ± 1.15	79 ± 0.49 108 ± 0.83	385 ± 1.68	1932 ± 16.75	2355 ± 25.45	442 ± 5.72 814 ± 11.22
	MAE-Time			735 ± 6.99	2083 ± 25.46	3998 ± 30.36	
D6	RMSE-Time	185 ± 2.96	375 ± 1.63	1469 ± 12.73	3985 ± 27.88	9006 ± 64.58	3929 ± 86.53
	MAE-Time	245 ± 3.31	653 ± 3.76	1344 ± 11.86	5071 ± 36.75	9597 ± 68.72	3859 ± 76.66

Table 2
rhe lowest rmse/mae and converging iteration count.

Case		M1	M2	M3	M4	M5	M6
D1	RMSE	**0.7847 ± 5.64E-4**	0.7874 ± 5.78E-4	0.8186 ± 6.42E-4	0.7915 ± 4.23E-4	0.7850 ± 5.72E-4	0.7874 ± 5.72E-4
	Round	56 ± 3.56	169 ± 2.13	**43 ± 2.12**	663 ± 1.56	532 ± 2.96	114 ± 1.70
	MAE	0.6042 ± 2.32E-4	0.6084 ± 4.27E-4	0.6245 ± 4.96E-4	0.6075 ± 3.87E-4	**0.6006 ± 4.93E-4**	0.6031 ± 4.83E-4
	Round	56 ± 3.56	148 ± 2.04	53 ± 2.83	700 ± 1.85	591 ± 3.12	121 ± 1.76
D2	RMSE	**0.7760 ± 5.16E-4**	0.7797 ± 5.56E-4	0.8151 ± 6.67E-4	0.7808 ± 4.16E-4	0.7760 ± 5.42E-4	**0.7796 ± 5.61E-4**
	Round	**55 ± 3.47**	189 ± 3.16	61 ± 3.01	684 ± 1.62	692 ± 3.36	132 ± 1.82
	MAE	0.5935 ± 9.72E-5	0.5986 ± 8.32E-5	0.6165 ± 4.68E-4	0.5947 ± 6.92E-5	0.5886 ± 6.38E-5	0.5919 ± 7.34E-5
	Round	105 ± 5.21	211 ± 3.33	69 ± 3.13	708 ± 1.91	763 ± 3.68	143 ± 1.86
D3	RMSE	**0.7061 ± 3.89E-4**	0.7092 ± 4.98E-4	0.7622 ± 5.49E-4	0.7127 ± 3.65E-4	0.7156 ± 5.14E-4	0.7251 ± 5.16E-4
	Round	**66 ± 3.96**	244 ± 3.52	101 ± 5.16	646 ± 1.46	564 ± 3.05	138 ± 1.85
	MAE	0.5555 ± 8.76E-5	0.5599 ± 7.22E-5	0.5849 ± 7.35E-5	0.5559 ± 6.33E-5	**0.5549 ± 6.12E-5**	0.5604 ± 7.12E-5
	Round	**68 ± 3.99**	261 ± 3.71	90 ± 5.02	674 ± 1.68	584 ± 3.16	137 ± 1.85
D4	RMSE	0.9980 ± 7.35E-4	**0.9966 ± 5.95E-4**	1.0242 ± 6.87E-4	1.0002 ± 5.62E-4	1.0003 ± 6.23E-4	0.9994 ± 6.83E-4
	Round	39 ± 2.21	101 ± 2.94	15 ± 0.68	313 ± 0.87	520 ± 2.85	86 ± 1.42
	MAE	0.7719 ± 5.02E-4	0.7757 ± 5.46E-4	0.7926 ± 5.86E-4	0.7725 ± 4.01E-4	**0.7668 ± 5.98E-4**	0.7679 ± 5.36E-4
	Round	42 ± 3.17	116 ± 3.05	16 ± 0.69	356 ± 0.92	357 ± 1.54	95 ± 1.47
D5	RMSE	**0.8634 ± 6.78E-4**	0.8662 ± 5.82E-4	1.0668 ± 6.92E-4	0.8907 ± 5.14E-4	0.9018 ± 6.02E-4	0.9107 ± 6.15E-4
	Round	48 ± 3.25	127 ± 3.12	**40 ± 1.96**	650 ± 1.54	785 ± 3.72	207 ± 2.33
	MAE	0.6442 ± 3.67E-4	0.6460 ± 4.68E-4	0.6838 ± 5.01E-4	**0.6411 ± 3.93E-4**	0.6465 ± 5.09E-4	0.6536 ± 4.96E-4
	Round	52 ± 3.31	153 ± 3.28	74 ± 3.35	717 ± 1.72	821 ± 3.93	210 ± 2.39
D6	RMSE	0.5333 ± 7.45E-5	**0.5323 ± 6.93E-5**	0.7528 ± 5.26E-4	0.5837 ± 6.86E-5	0.5971 ± 6.82E-5	0.5990 ± 7.56E-5
	Round	130 ± 5.68	320 ± 4.65	**106 ± 5.23**	810 ± 1.96	995 ± 4.16	548 ± 3.72
	MAE	0.3022 ± 3.66E-5	0.3024 ± 2.72E-5	0.4021 ± 5.31E-5	**0.2966 ± 2.78E-5**	0.3107 ± 2.36E-5	0.3129 ± 4.28E-5
	Round	173 ± 5.99	653 ± 6.82	**86 ± 3.12**	924 ± 2.03	1000 ± 4.19	587 ± 3.96

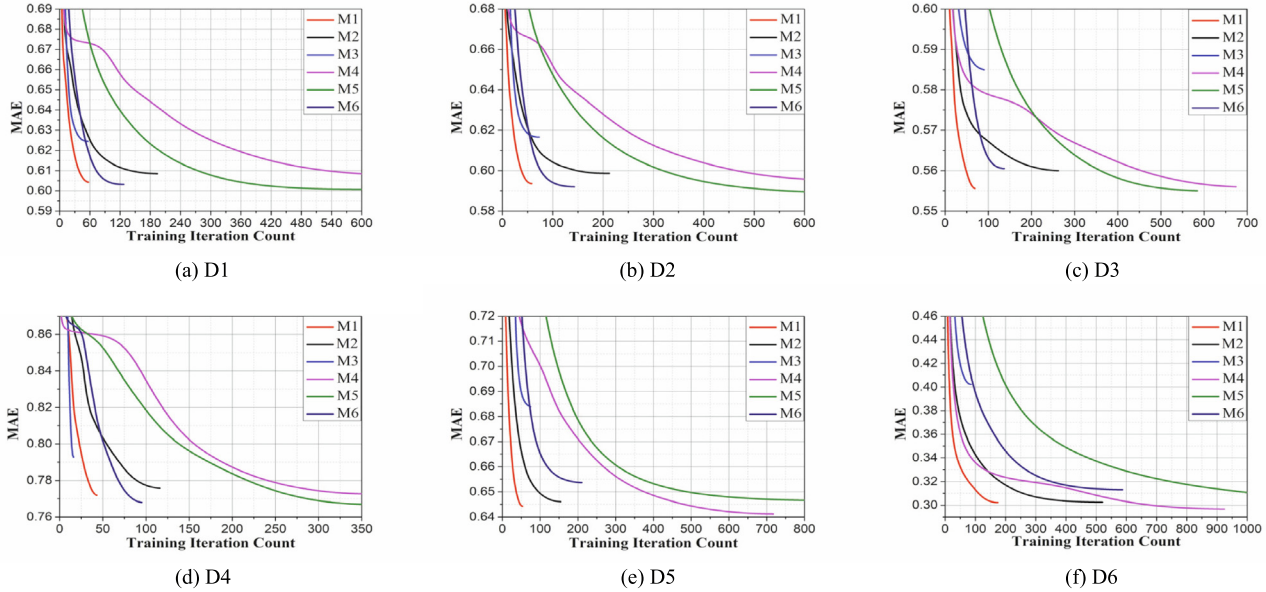


Fig. 10. Training curves of M1-M6 on D1-6 in MAE.

Table 3
Compared models.

No.	Name	Description
M1	PSL	A PSL model.
M2	LFA	An SGD-based LFA model.
M3	ALFA	An Adam-based LFA model.
M4	AGLFA	An Ada-grad-based LFA model.
M5	ADLFA	An Ada-Delta-based LFA model.
M6	RMSLFA	An RMSprop-based LFA model.

man test compares each average rank of compared models, i.e., $A_j = \sum_{j=1}^{N} r_i^j / N$. Table 4 records each model's average rank in terms of accuracy and efficiency.

Then the Friedman value should be computed as:

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j A_j^2 - \frac{k(k+1)^2}{4}\right]. \tag{13}$$

And the testing score is computed as:

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2}. \tag{14}$$

Note that a Friedman test's testing score is distributed based on the F-distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom. Hence, the null hypothesis that states all the models being equivalent can be rejected with the critical level α if F_F is greater than the corresponding critical value.

Note that we have six compared models on six datasets. Considers the prediction accuracy, we have two testing case on each dataset. Therefore, k = 6 and N = 12, and F_F with (5, 55) degrees of freedom is involved in the experiments. Similarly, for efficiency, we also have k = 6 and N = 12, and F_F with (5, 55) degrees of freedom. The critical value of $F(5, 55)$ for α = 0.05 is 2.382. Hence, if the testing scores are higher than 2.382, the null hypothesis is rejected. Following Table 4, the testing scores are 12.92 for accuracy and 212.55 for efficiency, respectively. Therefore, we assert that M1-6 are significantly different in performance with a confidence level at 95%.

For further identifying compared models' performance, we adopt the Nemenyi test [39]. It states the significant difference between two models once if their performance rank difference is greater than the critical value [39]:

$$\delta = q_\alpha \sqrt{\frac{k(k+1)}{6N}}, \tag{15}$$

where q_α is based on the studentized range statistic [39]. With six compared models in the experiment, the critical value q_α = 2.59 with the critical level α = 0.1 [39]. By substituting k = 6, N = 12, and q_α = 2.59 into (15), we obtain that δ = 1.98. It indicates that any two models with a rank difference higher than 1.98 have significant difference in performance with a confidence level at 90%.

The results of Nemenyi analysis are depicted in Fig. 11. As depicted in Fig. 11(a), we see that M1 outperforms M2-M6 in terms of prediction accuracy. Especially, it significantly outperforms M3. Considering computational efficiency, M1 is always steadily consumes the least total time cost among all compared models. Meanwhile, as illustrated in Fig. 11(b), the computational efficiency of M1 is significantly higher than M3-6.

5. Conclusions

This paper proposes a PSL model. It incorporates the principle of a PID controller into an SGD-based LFA model, thereby making its learning scheme consider the historical information hidden in its past update points. Experimental results on six HiDS matrices demonstrate that its computational efficiency and prediction accuracy are both satisfactory.

A PSL model utilizes a standard PID controller to enhance its learning scheme. However, its performance is sensitive with its controlling coefficients. The tuning of these coefficients requires a three-fold grid-search that costs much time. Hence, we propose to make them self-adaptation with intelligent optimization algorithms [53,56]. Moreover, a nonlinear PID [40–44], an improvement of PID control with nonlinear characteristics, has shown excellent performance in real industrial application. Is it compatible with our problem, and if so, can we achieve further performance gain by incorporating its principle into an LFA model?

Table 4
Average ranks of all compared models.

Average rank	M1	M2	M3	M4	M5	M6
Accuracy	1.96	3.21	6	3.33	2.71	3.79
Efficiency	1	2	3.58	5.25	5.75	3.42

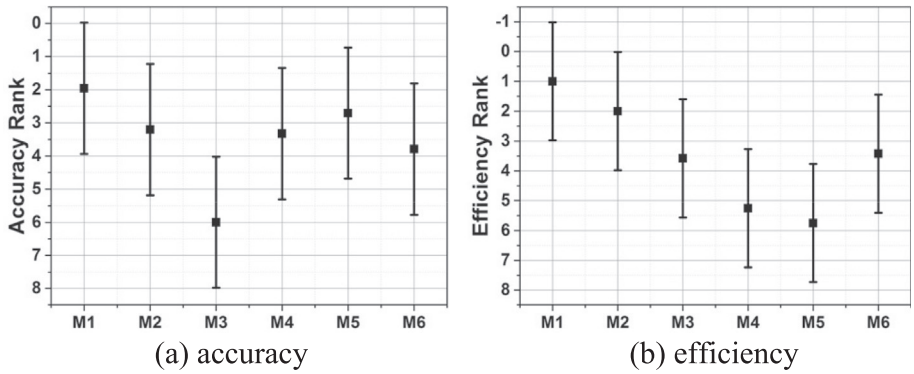


Fig. 11. Results of Nemenyi analysis.

This question remains open. We plan to address these issues in the future.

CRediT authorship contribution statement

Jinli Li: Investigation, Methodology, Software. **Ye Yuan:** Investigation, Methodology, Software. **Tao Ruan:** Original draft preparation, Visualization. **Jia Chen:** Original draft preparation, Visualization. **Xin Luo:** Conceptualization, Methodology, Reviewing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] X. Luo, M.-C. Zhou, S. Li, Y.-N. Xia, Z.-H. You, Q.-S. Zhu, H. Leung, Incorporation of efficient second-order solvers into latent factor models for accurate prediction of missing QoS data, IEEE Trans. Cybernetics 48 (4) (2018) 1216–1228.

[2] P. Massa, and P. Avesani, "Trust-aware recommender systems," in Proc. of the First ACM Conf. on Recommender Systems, Minneapolis, MN, USA, pp. 17-24, 2007.

[3] D. Rafael, M. Bonifacio, M. Nicolas, F. Julian, Computational intelligence tools for next generation quality of service management, Neurocomputing 72 (16–18) (2009) 3631–3639.

[4] X. Luo, Z.-D. Wang, M.-C. Zhou, H. Yuan, Latent factor-based recommenders relying on extended stochastic gradient descent algorithms, IEEE Trans. Syst. Man Cyber: Syst. (2018), https://doi.org/10.1109/TSMC.2018.2884191.

[5] X. Luo, M.-C. Zhou, S. Li, Z.-H. You, Y.-N. Xia, Q.-S. Zhu, A non-negative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method, IEEE Trans. Neural Networks Learn. Syst. 27 (3) (2016) 579–592.

[6] X. Luo, M.-C. Zhou, S. Li, Z.-H. You, Y.-N. Xia, Q.-S. Zhu, H. Leung, An efficient second-order approach to factorizing sparse matrices in recommender systems, IEEE Trans. Industrial Inform. 11 (4) (2015) 946–956.

[7] N. Qian, On the momentum term in gradient descent learning algorithms, Neural Networks 12 (1) (1999) 145–151.

[8] R. Salakhutdinov, A. Mnih, Probabilistic matrix factorization, Adv. Neural Inform. Process. Syst. 20 (2008) 1257–1264.

[9] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, IEEE Computer 42 (8) (2009) 30–37.

[10] G. Takács, I. Pilászy, B. Németh, D. Tikky, Scalable collaborative filtering approaches for large recommender systems, J. Mach. Learn. Res. 10 (2009) 623–656.

[11] K. Yu, S. Zhu, J. Lafferty, and Y. Gong, "Fast nonparametric matrix factorization for large-scale collaborative-filtering," in Proc. of the 32nd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, pp. 211-218, 2009.

[12] H. Shao, G. Zheng, Convergence analysis of a back-propagation algorithm with adaptive momentum, Neurocomputing 74 (5) (2011) 749–752.

[13] X. Luo, M.-C. Zhou, S. Li, M.-S. Shang, An inherently non-negative latent factor model for high-dimensional and sparse matrices from industrial applications, IEEE Trans. Industrial Inform. 14 (5) (2018) 2011–2022.

[14] J. Mohsen, and E. Martin, "A matrix factorization technique with trust propagation for recommendation in social networks," in Proc. of the Fourth ACM Conf. on Recommender Systems, Barcelona, Spain, pp. 135-142, 2010.

[15] B. Recht, C. Ré, S.-J. Wright, F. Niu, Hogwild: a lock-free approach to parallelizing stochastic gradient descent, Adv. Neural Inform. Process. Syst. 24 (2011) 693–701.

[16] X. Luo, J.-P. Sun, Z.-D. Wang, S. Li, M.-S. Shang, Symmetric and non-negative latent factor models for undirected, high dimensional and sparse networks in industrial applications, IEEE Trans. Industrial Inform. (2017), https://doi.org/10.1109/TII.2017.2724769.

[17] Y. Liu, J. Zhang, S. Wang, "Optimization design based on PSO algorithm for PID controller," in Proc. of the Fifth World Congress on Intelligent Control and Automation, pp. 2419-2422, 2004.

[18] Y. Song, M. Li, X. Luo, G. Yang, C. Wang, Improved symmetric and nonnegative matrix factorization models for undirected, sparse and large-scaled networks: a triple factorization-based approach, IEEE Trans. Industrial Inform. (2019), https://doi.org/10.1109/TII.2019.2908958.

[19] A. Shahzad, M. Lee, Y.K. Lee, S. Kim, N. Xiong, J.Y. Choi, Y. Cho, Real time MODBUS transmissions and cryptography security designs and enhancements of protocol sensitive information, Symmetry 7 (3) (2015) 1176–1210.

[20] X. Luo, M.-C. Zhou, S. Li, L. Hu, M.-S. Shang, Non-negativity constrained missing data estimation for high-dimensional and sparse matrices from industrial applications, IEEE Trans. Cyber. 50 (5) (2018) 1844–1855.

[21] X. Luo, M.-C. Zhou, Y.-N. Xia, Q.-S. Zhu, A.-C. Ammari, A. Alabdulwahab, Generating highly accurate predictions for missing QoS data via aggregating nonnegative latent factor models, IEEE Trans. Neural Networks Learn. Syst. 27 (3) (2016) 524–537.

[22] R. Gemulla, E. Nijkamp, P-J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in Proc. of the 17th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, San Diego, California, USA, pp. 69-77, 2011.

[23] X. Luo, H.-J. Liu, G.-P. Gou, Y.-N. Xia, Q.-S. Zhu, A parallel matrix factorization based recommender by alternating stochastic gradient decent, Eng. Appl. Artif. Intel. 25 (7) (2012) 1403–1412.

[24] X. Luo, Y.-N. Xia, Q.-S. Zhu, Applying the learning rate adaptation to the matrix factorization based collaborative filtering, Knowledge-Based Systems 37 (2013) 154–164.

[25] R-B. Zadeh, X-R. Meng, A. Ulanov, B. Yavuz, L. Pu, S. Venkataraman, E. Sparks, A. Staple, and M. Zaharia, "Matrix computations and optimization in apache spark," in Proc. of the 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, San Francisco, California, USA, pp. 31-38, 2016.

[26] K. Huang, N.-D. Sidiropoulos, A. Swami, Non-negative matrix factorization revisited: uniqueness and algorithm for symmetric decomposition, IEEE Trans. Signal Process. 62 (1) (2014) 211–224.

[27] C.-J. Lin, Projected gradient methods for nonnegative matrix factorization, Neural Comput. 19 (10) (2007) 2756–2779.

[28] X. Luo, M-S. 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 IEEE Int. Conf. on Data Mining, Barcelona, Spain, pp. 311-319, 2016.

[29] Y. Yuan, X. Luo, M.-S. Shang, Effects of preprocessing and training biases in latent factor models for recommender systems, Neurocomputing 275 (2018) 2019–2030.

[30] C. Jiang, Y. Ma, C. Wang, PID controller parameters optimization of hydro-turbine governing systems using deterministic-chaotic-mutation evplutionary programming(DCMEP), Energy Convers. Manage. 47 (9–10) (2006) 1222–1230.

[31] 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, 2009, pp. 1924–1928.

[32] S.-D. Hanwate, Y.-V. Hole, Optimal PID design for Load frequency control using QRAWCP approach, IFAC-PapersOnLine 51 (4) (2018) 651–656.

[33] J. Cuellar, A. Y; de Jesus, Romero-Troncoso R; Morales-Velazquez, L, "PID-controller tuning optimization with genetic algorithms in servo system," Int. J. Adv. Robot. Syst. vol. 10, no. 9, pp. 324, 2013.

[34] J.-A. Konstan, B.-N. Miller, D. Maltz, J.-L. Herlocker, L.-R. Gordon, J. Riedl, GroupLens: applying collaborative filtering to usenet news, Commun. ACM 40 (1997) 77–87.

[35] V.-M. Janakiraman, X.-L. Nguyen, D. Assanis, Stochastic gradient based extreme learning machines for stable online learning of advanced combustion engines, Neurocomputing 177 (2016) 304–316.

[36] D. Xu, H. Shao, H. Zhang, A new adaptive momentum algorithm for split-complex recurrent neural networks, Neurocomputing 93 (2012) 133–136.

[37] H. Ma, D. Zhou, C. Liu, M-R. Lyu, and I. King, "Recommender systems with social regularization," in Proc. of the 4th ACM Int. Conf. on Web Search and Data Mining, pp. 287-296, 2014.

[38] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (1) (2006) 1–30.

[39] Y-X. Su, B-Y Duan, "Anovel nonlinear PID Controller," Control and Decision, vol.35, no.4, 2003.

[40] H. Hua, Y-C. Fang, X-T. Zhang, C. Qian, "Auto-tuning nonlinear PID-type controller for rotorcraft-based aggressive transportation," mechanical systems and signal processing, DOI: 10.1016/j,ymssp.2020.106858, 2020.

[41] Q. Shi, H.-K. Lam, C.-B. Xuan, Adaptive neuro-fuzzy PID controller based on twin delayed deep deterministic policy gradient algorithm, Neurocomputing 402 (2020) 183–194.

[42] Y. Yang, N.X. Xiong, N.Y. Chong, X. Défago, A decentralized and adaptive flocking algorithm for autonomous mobile robots, in: Proc of the 3rd Int. Conf. on Grid and Pervasive Computing, 2008, pp. 262–268.

[43] G.-M. Riduwan, A.-M. Ashraf, I.-R. Mohd, T. Raja, A Multiple-node Hormone Regulation of Neuroendocrine-PID(MnHR-NEPID) Control for Nonlinear MIMO Systems, IETE J. Res. (2020).

[44] L. Brozovsky, V. Petricek, Recommender system for online dating service, eprint arXiv:cs/0703042 (2007).

[45] Y. Koren and R. Bell, "Advances in collaborative-filtering," in Recommender Systems Handbook, pp. 145–186, 2011.

[46] W.L. Wu, N.X. Xiong, C.X. Wu, Improved clustering algorithm based on energy consumption in wireless sensor networks, IET Networks 6 (3) (2017) 47–53.

[47] W. Chu, Z. Ghahramani, Probabilistic models for incomplete multi-dimensional arrays, in: Proc of the 12th International Conference on Artificial Intelligence and Statistics, 2009, pp. 89–96.

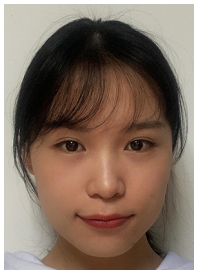
[48] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, Cambridge, 2009.

[49] K. Ogata, Discrete-time control systems, Prentice-Hall, 1995.

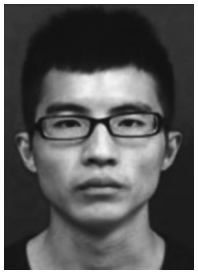
[50] T. Zhao, Y. Chen, S. Dian, R. Guo, S. Li, General type-2 fuzzy gain scheduling PID controller with application to power-line inspection robots, Int. J. Fuzzy Syst. 22 (1) (2020) 181–200.

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

[52] Q. Wang, S. Chen, X. Luo, An adaptive latent factor model via particle swarm optimization, Neurocomputing 369 (2019) 176–184.
[53] I. Sutskever, J. Martens, G.-E. Dahl, G.-E. Hinton, On the importance of initialization and momentum in deep learning, in: Proc of the 30th Int. Conf. on Machine Learning, 2013, pp. 1139–1147.
[54] X. Luo, Z.-G. Liu, S. Li, M.-S. Shang, Z.-D. Wang, A fast non-negative latent factor model based on generalized momentum method, IEEE Trans. Syst. Man Cyber.: Syst. (2019), https://doi.org/10.1109/TSMC.2018.2875452.
[55] S.-C. Gao, M.-C. Zhou, Y.-R. Wang, J.-J. Cheng, H.-N.-K. Yachi, J.-H. Wang, Dendritic neuron model with effective learning algorithms for classification, approximation and prediction, IEEE Trans. Neural Networks Learn. Syst. (2019) 601–614.
[56] Q. Zhang, C.J. Zhou, N.X. Xiong, Y.Q. Qin, X. Li, S. Huang, Multimodel-based incident prediction and risk assessment in dynamic cybersecurity protection for industrial control systems, IEEE Trans. Syst. Man Cyber.: Syst. 46 (10) (2016) 1429–1444.
[57] K.X. Huang, Q. Zhang, C.J. Zhou, N.X. Xiong, Y.Q. Qin, An efficient intrusion detection approach for visual sensor networks based on traffic pattern learning, IEEE Trans. Syst. Man Cyber. Syst. 47 (10) (2017) 2704–2713.
[58] 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, DOI: 10.1109/JAS.2020.1003321.



Jinli Li received the B.S. degree in computer science and technology from China West Normal University, Nanchong, China, in 2018. She is currently pursuing her M.S. degree at China West Normal University, Nanchong, Sichuan, China. She is also a visiting student at Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing China. Her research interests focus on Data Analysis and Control Theory.



Ye Yuan 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, respectively. He is currently working toward the Ph.D. degree in computer science from Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China. His research interests include data mining and intelligent computing.

Tao Ruan received the M. A. degree from the China University of Geosciences, Beijing, China, in 2013. Then she joined the China Patent Information Center, is now an assistant researcher mainly focusing on patent processing in the field of computer, and is also responsible for processing of PCT and PRI, processing of applicators and inventors and the like. Her research interests include big data analysis, network security and domestic and foreign patent translation in the fields of computer, electricity, machinery and the like.

Jia Chen received the B.S. degree in electronic circuit and system from Beihang University, Beijing, China, in 2004, and the Ph.D. degree in computer science from Beihang University, Beijing, China, in 2019. Since July 2020, she has been a faculty with the School of Cyber Science and Technology, Beihang University. Her research interests are in cyberspace security, recommender system and latent factor model.

Xin Luo received the B.S. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2005 and the Ph.D. degree in computer science from Beihang University, Beijing, China, in 2011. In 2016, he joined the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China, as a Professor of computer science and engineering. He is currently also a Distinguished Professor of computer science with the Dongguan University of Technology, Dongguan, China. His research interests include big data analysis and intelligent robotics. He has published over 100 papers (including over 50 IEEE TRANSACTIONS papers) in the above areas. Dr. Luo was a recipient of the Hong Kong Scholar Program jointly by the Society of Hong Kong Scholars and China Post-Doctoral Science Foundation in 2014, the Pioneer Hundred Talents Program of Chinese Academy of Sciences in 2016, and the Advanced Support of the Pioneer Hundred Talents Program of Chinese Academy of Sciences in 2018. He is currently serving as an Associate Editor for the IEEE/CAA JOURNAL OF AUTOMATICA SINICA and Neurocomputing.