2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
October 11-14, 2020. Toronto, Canada

A Nonlinear Proportional Integral Derivative-Incorporated Stochastic Gradient Descent-based Latent Factor Model

Jinli Li
Computer School
China West Normal University
Nanchong, *Sichuan*,*China*
appleli_li@163.com

Ye Yuan
Chongqing Institute of Green and Intelligent Technology
Chongqing, China
yuanye@cigit.ac.cn

***Abstract*—Recommender system (RS) commonly describes its user-item preferences with a high-dimensional and sparse (HiDS) matrix. A latent factor (LF) model relying on stochastic gradient descent (SGD) is frequently adopted to extract useful information from such an HiDS matrix. In spite of its efficiency, an SGD-based LF model commonly takes many iterations to converge. When processing a large-scale HiDS matrix, its computational efficiency should be further improved by further accelerating its convergence rate as well as maintaining its learning ability. To address this issue, this paper innovatively proposes novel SGD algorithm which incorporates a nonlinear proportional integral derivative (NPID) controller into its learning scheme for building an LF model. The main idea is to adopt an NPID controller to model the learning residual achieved in the past iterations, thereby adjusting the learning direction and step size of the current iteration, thereby making a resultant model converge fast. With the NPID-incorporated SGD algorithm, this study proposes an NPID-SGD-based LF (NSLF) model. Experimental results on two HiDS matrices demonstrate that compared with a standard SGD-based LF model, the proposed model achieves higher computational efficiency and prediction accuracy for missing data of an HiDS matrix.**

Keywords—Recommender System, High-Dimensional and Sparse Matrix, Latent Factor Model, Stochastic Gradient Descent, Nonlinear Proportional Integral Derivative, Nonlinear Proportional Integral Derivative-Incorporated Stochastic Gradient Descent.

I. INTRODUCTION

Information overload has become a serious problem for ordinary people to extract desired information from enormous data scattering in the ever-exploding World-Wide-Web. How to develop an intelligent system to filter the useful information out of massive data has become a heated issue. Recommender systems (RSs) [1, 2, 33], being able to filter people's potential favorites out of massive data, are becoming increasingly important for various Web applications. A RS commonly takes a user-item rating matrix as its fundamental data source, where each user's preference on each item is modeled according to his/her user-item usage history.

With exponentially growing user and item counts, a user can only touch a tiny subset of items. Hence, a user-item rating matrix is frequently high-dimensional and sparse (HiDS) [3, 4, 35]. In spite of its extremely sparsity, an HiDS matrix involves a mountain of valuable information regarding various patterns, e.g., user community tendency in a social network service system, users' preferences on items in recommender systems.

In order to extract such desired knowledge from an HiDS matrix, great efforts have been made. Representative models include the biased combination LF model [5, 34], a probabilistic MF model [6], a singular value decomposition plus-plus model [7], a collaborative Gaussian process-based preference model [8], and a nonparametric MF model [9]. Considering existing models for analyzing HiDS matrices, latent factor (LF) models are highly popular owing to its high scalability and efficiency. Given an HiDS matrix, an LF model maps the row and column entities into a unique and low-dimensional LF space, and builds a cost function on an HiDS matrix' observed data related to desired LFs. Then it minimizes the objective function to train LFs and then predict the missing values in an HiDS matrix according to these obtained LFs.

As described in prior research [10, 11, 22, 31, 36], a stochastic gradient descent (SGD) optimizer is an efficiently and commonly used to build an LF model, since it can quickly acquire desired LFs with good representation from an HiDS matrix. However, an SGD-based LF model commonly takes numerous iterations to converge, thereby resulting in considerable time cost on large-scale datasets. To address this

*This research is supported in part by the National Natural Science Foundation of China under grants 61772493, 91646114 and 61602352, in part by the Natural Science Foundation of Chongqing (China) under grant cstc2019jcyjjqX0013, and in part by the Pioneer Hundred Talents Program of Chinese Academy of Sciences.

J.L Li, is with the Computer School of China West Normal University, Nanchong Sichuan 637002, China (e-mail: appleli_li@163.com)

Y. Yuan is with the Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China (e-mail: yuanye@cigit.ac.cn). (Corresponding authors: Y. Yuan)

Y. Yuan is also with the University of Chinese Academy of Sciences, Beijing 100049, China.

issue, the frequently adopted strategy is an SGD-Momentum optimizer [12, 13]. It considers both the current gradients and the latest update velocity to accelerate SGD. However, it suffers the overshoot problem [14], which refers to the phenomena that a parameter exceeds much its target value and does not change its update direction. Such a problem results an unstable convergence process, thereby maybe resulting in accuracy loss [15]. Hence, it is significantly important to investigate whether we can find a new approach to improve the convergence rate with accuracy loss.

Aiming at addressing the above issue, this paper innovatively proposes to implement SGD algorithm with a nonlinear proportional integral derivative (NPID) controller in LF, thereby achieving an NPID-SGD-based LF (NSLF) model. Main contributions of this paper include:

a) NPID-incorporated SGD algorithm design and analyzation.

b) An NSLF model, which incorporates the NPID controller into an SGD-based LF model to achieve fast convergence rate and competitive prediction accuracy for missing data of an HiDS matrix;

Experimental results are conducted on two HiDS matrices generated by real RSs to evaluate NSLF's performance.

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

II. PRELIMINARIES

A. Symbol Appointment

Necessary symbols adopted in this paper are summarized in Table I.

TABLE I. SYMBOLS.

Parameter	Description
M, N	Involved entity sets.
R	A $\|M\|\times\|N\|$ HiDS matrix as input.
$\hat{R}$	R's rank-f approximation.
f	LF dimension/Rank of $\hat{R}$.
$r_{m,n}$, $\hat{r}_{m,n}$	Each single element of R and $\hat{R}$.
$err_{m,n}$	Difference of $r_{m,n}$ and $\hat{r}_{m,n}$.
Λ, Γ	Known and unknown entry sets in R and $\|\Lambda\|\ll\|\Gamma\|$.
P	A $\|M\|\times f$ LF matrix corresponding to M.
Q	A $\|N\|\times f$ LF matrix corresponding to N.
$p_{m,k}$, $q_{n,k}$	Each single element of P and Q.
λ	Regularization coefficient.
η	Learning rate in SGD.
t	Iteration count of SGD.
$\varepsilon_{m,n}$	instant error on each training instance $r_{m,n}\in\Lambda$.
K_p	Proportional control coefficients of NPID/PID.
K_i	Integral control coefficients of NPID/PID.
K_d	Derivative control coefficients of NPID/PID.
TV	True value of users in NPID/PID.
PV	Prediction value of users in NPID/PID.
E_k	Difference of TV and PV.
err	Error for feedback control system.
z_{11}, z_{21}	The signal for tracking TV and PV.
z_{12}, z_{22}	The differential signal of TV and PV.
e_1	The deviation between z_{11} and z_{21}.
e_0	Integration of e_1.
e_2	Differentiation of e_1.
TD-I	First tracking differentiator.
TD-II	Second tracking differentiator.
$\|\cdot\|_F$	Frobenius norm of a given matrix.
$\|\cdot\|_2$	l_2 norm of a given matrix.

B. Problem Statement

In R, each element represents user m's preference on item n. Thus, the problem can be regarded as a process of missing-data-estimation. An LF model builds a low-rank approximation PQ to HiDS matrix. In order to get more precise P and Q, the LF model constructs a cost function that measures the difference between R and the rank-f approximation $\hat{R}$. In general, the function is explained by Euclidean distance, such an objective function is shown as below:

$$\begin{aligned}\arg\min_{P,Q}\varepsilon(P,Q) &= \| R-PQ^T \|_F^2 \\ &\approx \sum_{r_{m,n}\in\Lambda}\left(r_{m,n}-\sum_{k=1}^{f}p_{m,k}\cdot q_{n,k}\right)^2\end{aligned} \tag{1}$$

In addition, there is ill-posed question for LF models' method. Devoting to getting better results in the process of solving cost function may cause overfitting[32, 37]. Therefore, it is important to combine regularization terms with objective functions for avoiding overfitting. The most frequently used method is Tikhonov regularization. Hence, the function is change into:

$$\begin{aligned}&\arg\min_{P,Q}\varepsilon(P,Q) \\ &=\sum_{r_{m,n}\in\Lambda}\left((r_{m,n}-\hat{r}_{m,n})^2+\lambda_P\|p_m\|_2+\lambda_Q\|q_n\|_2\right) \\ &=\sum_{r_{m,n}\in\Lambda}\left((r_{m,n}-\hat{r}_{m,n})^2+\lambda_P\sum_{k=1}^{f}p_{m,k}^2+\lambda_Q\sum_{k=1}^{f}q_{n,k}^2\right)\end{aligned} \tag{2}$$

C. An SGD-based LF model

As indicated by prior research [10, 11, 22], SGD is an efficient optimizer when performing LF of HiDS matrices. With SGD, the objective function (2) is minimized as follows:

$$p_{m,k}^t=p_{m,k}^{t-1}-\eta\frac{\partial\varepsilon_{m,n}^{t-1}}{\partial p_{m,k}^{t-1}},q_{n,k}^t=q_{n,k}^{t-1}-\eta\frac{\partial\varepsilon_{m,n}^{t-1}}{\partial q_{n,k}^{t-1}} \tag{3}$$

where $err_{m,n}=r_{m,n}-\hat{r}_{m,n}$, the functions above-mentioned is simplified as below:

$$\begin{aligned}\frac{\partial\varepsilon_{m,n}^{t-1}}{\partial p_{m,k}^{t-1}}&=-err_{m,n}^{t-1}\cdot q_{n,k}^{t-1}+\lambda_P\cdot p_{m,k}^{t-1} \\ \frac{\partial\varepsilon_{m,n}^{t-1}}{\partial q_{n,k}^{t-1}}&=-err_{m,n}^{t-1}\cdot p_{m,k}^{t-1}+\lambda_Q\cdot q_{n,k}^{t-1}\end{aligned} \tag{4}$$

Finally, we combine (4) with (2), the functions change into:

$$\begin{aligned}p_{m,k}^t&=p_{m,k}^{t-1}+\eta\cdot(err_{m,n}^{t-1}\cdot q_{n,k}^{t-1}-\lambda_P\cdot p_{m,k}^{t-1}) \\ q_{n,k}^t&=q_{n,k}^{t-1}+\eta\cdot(err_{m,n}^{t-1}\cdot p_{m,k}^{t-1}-\lambda_Q\cdot q_{n,k}^{t-1})\end{aligned} \tag{5}$$

With (5), an SGD-based LF model is achieved, which is frequently adopted in various HiDS matrices analysis tasks.

III. METHOD

A. A Nonlinear PID Controller

The PID controller is one of the most popular classical control algorithms since its invention in 1910 [20, 21]. It is a feedback system and the feedback-loop just with a single input, and its parameters of control is easy to adjust [26]. Thus, PID controller has some excellent features:

a) PID controller takes present, past and nearest conditions into account;

b) It has a straightforward process, clear physical meaning and low computational complexity [20-23]. Therefore, PID controller doesn't increase much burden of time in each iteration.

PID controller gets result by combining three kinds of errors with parameters, and called them proportional, integral and derivative respectively. In general, discrete PID is usually used by replacing the integral and derivative terms with summation and subtraction. This measure reduce the complexity of calculation and more applicable. Mathematically, the discrete PID is shown as follow:

$$PID = K_P \times E_k + K_I \times \sum_{i=0}^{k} E_i + K_D \times (E_k - E_{k-1}) \quad (6)$$

The nonlinear PID control is an improvement of PID control with nonlinear characteristics [23-25, 28-30]. As we known, traditional PID controller has been widely used in industrial applications, but it is powerless in face of some complex problems, especially nonlinear object. To solve this problem, nonlinear PID introduces nonlinear factors to the traditional PID for better performance.

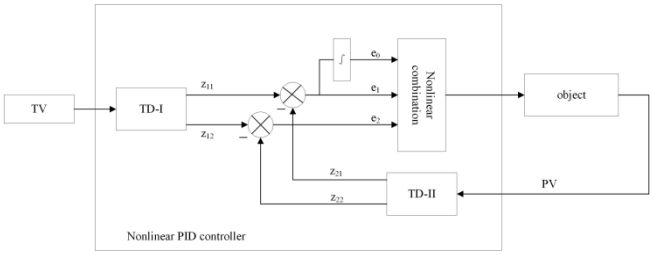


Fig. 1: The implement of nonlinear PID

Formulas about those parameters are shown as follow:

$$K_p(e(t)) = k_{p_1} + k_{p_2} \times \left(1 - \operatorname{sech}(k_{p_3} \times e(t))\right) \quad (7)$$

$$K_i(e(t)) = k_i \times \operatorname{sech}(k_{i_2} \times e(t)) \quad (8)$$

$$K_d(e(t)) = k_{d_1} + \frac{k_{d_2}}{1 + k_{d_3} \times e^{k_{d_4} \times e(t)}} \quad (9)$$

Note that for a nonlinear PID control, three parameters of proportion, integration and differentiation are all nonlinear functions of error. According to the classical control theory, the purpose of K_P is to improve the response time and reduce the transition time. Thus, the value of K_P is in direct proportion to the absolute value of error. Integral term's aim at decreasing steady-state error, therefore, K_I is inversely proportional to the absolute value of error. The third one is K_D, the main task of it is to prevent overshoot. Meanwhile, its value shouldn't have negative influence on response time. Also note that values of those parameters don't equal zero in order to avoiding system out of control [17].

B. A NSLF Model

In this part, we discuss how to incorporate nonlinear PID into SGD for accelerating the training process. The flowchart of our method is as Fig. 2.

As shown above, this system calculates the error firstly, which measures difference between true value and predictive value for each instance of Λ, and it expressed as $err_{m,n} = r_{m,n} - \hat{r}_{m,n}$. Then, nonlinear PID receives the error and use it calculate parameters of proportional, derivative and integral terms by formulas (7-9). After that, NSLF model calculates value of those three terms and combines these three things to get appropriate output. Then nonlinear PID applies the output to target object and help to change the state of object. At last, nonlinear PID receives the updated status value of target to compute feedback error like last round for next training process. The calculation is as below:

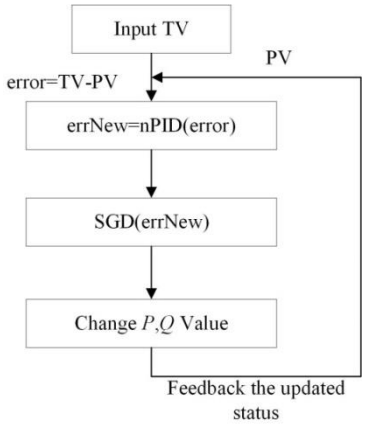


Fig. 2: Flowchart of NSLF

$$nPID(err_{m,n}^t) = K_P err_{m,n}^t + K_i \sum_{i=0}^{t} err_{m,n}^i + K_d (err_{m,n}^t - err_{m,n}^{t-1}) \quad (10)$$

And then, we achieve the expression by (7-9):

$$\begin{aligned} nPID(err_{m,n}^t) = & \left\{k_{p1p1} + k_{p2} \times \left[1 - \operatorname{sech}\left(k_{p3} \times e(t)\right)\right]\right\} \times err_{m,n}^t \\ & + \left\{k_{i_1} \times \operatorname{sech}\left(k_{i2} \times e(t)\right)\right\} \times \sum_{i=0}^{t} err_{m,n}^i \\ & + \left\{k_{d_1} + \frac{k_{d2}}{1 + k_{d3} \times e^{k_{d4} \times e(t)}}\right\} \times (err_{m,n}^t - err_{m,n}^{t-1}) \end{aligned} \quad (11)$$

Afterwards, we use (11) to instead of the $err_{m,n}$ in (5) as below:

$$errNew^t = nPID(err_{m,n}^t) \quad (12)$$

Then, we obtain the expression of the update rule for LF matrices P and Q by combining (5) and (12), it is shown as below:

$$p_{m,k}^{t} = p_{m,k}^{t-1} + \eta \times (errNew^{t-1} \cdot q_{n,k}^{t-1} - \lambda_P \cdot p_{m,k}^{t-1})$$
$$q_{n,k}^{t} = q_{n,k}^{t-1} + \eta \times (errNew^{t-1} \cdot p_{m,k}^{t-1} - \lambda_Q \cdot q_{n,k}^{t-1}) \quad (13)$$

Finally, we implement the NSLF model based on (11)-(13).

There are two questions should be mentioned. Firstly, the NSLF model doesn't use tracking-differentiator. Because the NSLF model can extract input value preciously, and we use discrete formula in algorithm, this measure avoids signal cannot differential. Second, the nonlinearity of NSLF model reflected in parameters of three terms. In other words, formula (7-9) represents nonlinear characteristic of NSLF model. In our study, this choice is proved efficient.

C. NSLF Algorithm Design and Analysis

NPID-incorporated SGD algorithm

Input: M, N, Λ, d, λ_P, λ_Q, h, K_P, K_I, K_D, *totalEntry*	
Operation	**Cost**
initialize $P^{|M|\times f}$, $Q^{|N|\times f}$	$\Theta((\|M\|+\|N\|)\times f)$
initialize λ_P, λ_Q, h, K_P, K_I, K_D , *totalEntry*	$\Theta(1)$
initialize *Count* = 0, n = 0, *Max-training-round* = N	$\Theta(1)$
initialize *sumArray*[*totalEntry*][2]	$\Theta(1)$
initialize *subArray*[*totalEntry*][2]	$\Theta(1)$
while not converge **and** $n \le N$ **do**	$\times n$
for each $r_{m,n}$ **in** Λ	$\times\|\Lambda\|$
$\hat{r}_{u,i} = \sum_{m=1}^{f} x_{u,m} y_{m,i}$	$\Theta(f)$
$err = r_{m,n} - \hat{r}_{m,n}$	$\Theta(1)$
$K_p = k_{p1} + k_{p2} \times [1 - \mathrm{sech}(k_{p3} \times err)]$	$\Theta(1)$
$K_i = k_{i1} + \mathrm{sech}(k_{i2} \times err)$	$\Theta(1)$
$K_d = k_{d1} + [k_{d2} \div (1 + k_{d3} \times e^{kd4 \times err})]$	$\Theta(1)$
$errSum = sumArray[Count][1] + err$	$\Theta(1)$
$errSub = err - subArray[Count][1]$	$\Theta(1)$
$subArray[Count][1] = err$	$\Theta(1)$
$errNew = K_P \times err + K_I \times errSum + K_D \times errSub$	$\Theta(1)$
$Count = Count + 1$	$\Theta(1)$
for d=1 **to** f	$\times f$
$p_{m,k} = p_{m,k} + \eta \times (errNew \times q_{n,k} - \lambda_P \times p_{m,k})$	$\Theta(1)$
$q_{n,k} = q_{n,k} + \eta \times (errNew \times p_{m,k} - \lambda_Q \times q_{n,k})$	$\Theta(1)$
end for	-
end for	-
$n = n + 1$	$\Theta(1)$
end while	-
Output: P, Q	

As shown above, NSLF model has change a little in algorithm. Thus, this improvement doesn't bring too much of burden whether in time complexity or space complexity. And its time complexity is shown as follows:

$$T_{NSLF} = \Theta((|M| + |N|) \times f + (n \times |\Lambda| \times f)) \approx \Theta(n \times |\Lambda| \times f) \quad (14)$$

The condition of (14) is that $|\Lambda| \gg \max\{|M|, |N|\}$, and this condition is used in many applications to realized dimension reduction.

There are two important parts on space complexity, a) the caches of P and Q in SGD cost comes to $\Theta((|M|, |N|) \times f)$; b) nonlinear PID's arrays cache cost comes to (4×count), thus, the space complexity of NSLF is:

$$S_{NSLF} = \left[(|M| + |N|) \times f + Count \right] \quad (15)$$

In all, NSLF model has little impact on both time and space complexity, and it is most related to the count of training set.

IV. Experimental Results and Analysis

A. General Settings

Evaluation Metrics. There are many half-baked datasets in most industrial applications, and these datasets meet the feature of HiDS. Thus, in our experiment, our evaluation metrics is missing data estimation, we use root mean squared error (RMSE) and mean absolute error (MAE) to measure the results. The form of it is shown as below:

$$RMSE = \sqrt{\left(\sum_{r_{u,v} \in \Gamma} (r_{u,v} - \hat{r}_{u,v})^2 \right) / |\Gamma|}, \qquad MAE = \left(\sum_{r_{u,v} \in \Gamma} |r_{u,v} - \hat{r}_{u,v}|_{abs} \right) / |\Gamma|; \quad (16)$$

As mentioned in Table I, Γ represents unknown entry set in R, thus, $|\Gamma|$ is entry-number of unknown 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 8 GB RAM. All tested models are implemented in JAVA SE 7U60.

Datasets. Two HiDS matrices adopted in our experiments are collected by real RSs.

a) D1: MovieLens 10M. This dataset is collected from the MovieLens system maintained by the GroupLens research team. There are 10,000,054 ratings in the scale of [1, 5] come from 71,567 users and 10,681 movies.

b) D2: Douban. This dataset is come from Douban, which is one of the China largest online book, movie and music database. However, this dataset's density is 0.22% only, it contains 16,830,839 ratings in the scale of [1, 5] come from 129,490 users and 58,541 items.

Both of two datasets are meet features as below: a) high-dimensional; b) extremely sparse; c) all data are collected by industrial applications in use. Thus, our results are highly convenience and representative.

Note that divide 80% of the dataset into training set and the rest into testing set, and five-fold cross-validations is adopted. And the conditions of termination are: a) the number of consumed iterations reaches a preset threshold; b) the model converges, i.e., the error difference is smaller than 10^{-6} between two consecutive iterations.

Model Settings. In order to validate NSLF model's performance, we compare the proposed NSLF with another two LF models.

TABLE II. COMPARED MODELS IN OUR EXPEROMENTS

Model	Name	Description
M1	NSLF	An NPID-SGD-based LF model
M2	PLF	An PID-SGD-based LF model
M3	LF	An SGD-based LF model

There are several standards in our experiment for more compellent results.

a) In order to avoiding overfitting, we add regularization term in both models, and set same regularization coefficient for P and Q. Finally, we determine $\lambda_P=\lambda_Q=0.05$.

b) One of the most important parameter in training process is learning rate η, and we set it equals 0.004 in both three models.

c) The dimension of LF space f has been seted 20 uniformly.

In M1, it has exponent function in training process, in order to saving the time cost, we import an approximate function of exponent function. The form of it is $e_y \approx a \times y + (b-c)$, a=220,b=1023×220, c=6801.

B. Performance Comparsion

In this part, we compare the performance of M1, M2 and M3 both in D1 and D2. All results are recorded in Table III and Table IV. Table III records their lowest RMSE, converging iteration count and total time cost. Table IV records their lowest MAE, converging iteration count and total time cost. And the training process of compared models in D1 and D2 are shown in Figs.3 and 4. From those results, we have following findings:

a) **NSLF can significantly reduce the converging iterations.** For instance, as depicted in Table III and Fig. 3(a), M1, i.e., a proposed NSLF model, just consumes 27 iterations to converge in RMSE on D1. In contrast, M2 converge iteration count comes to 55, and M3' iterations is 174. Thus, compared with M2 and M3, the converging iteration count decreases at 50.90% and 84.48% by M1, respectively. The same outcomes are also seen on D2 as depicted in Table III and Fig. 4(a).

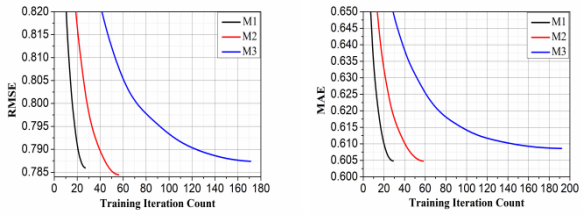
b) NSLF's computational efficiency is significantly higher than its peers. For instance, as recorded in Table III, on D1, M1, i.e., a proposed NSLF model, takes 22 seconds to achieve the lowest RMSE. Yet, M2 takes 39 seconds and M3 takes 129 seconds, respectively. Thus, its time cost reduces at 43.58% and 82.94%, respectively. The similar conclusions can be made on D2.

TABLE III. PERFORMANCE OF M1-M3 ON D1-D2 (RMSE)

Dataset	Model	RMSE	Iteration	Time cost (s)
D1	M1	0.7854	**27**	**22**
	M2	**0.7844**	55	39
	M3	0.7874	174	129
D2	M1	0.7079	**34**	**49**
	M2	**0.7062**	67	86
	M3	0.7093	235	260

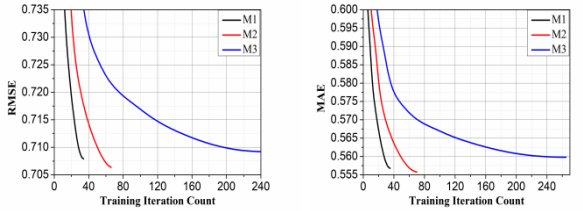
TABLE IV. PERFORMANCE OF M1-M3 ON D1-D2 (MAE)

Dataset	Model	RMSE	Iteration	Time cost (s)
D1	M1	**0.6048**	**29**	**23**
	M2	**0.6048**	58	42
	M3	0.6084	194	117
D2	M1	0.5567	**35**	**50**
	M2	**0.5557**	70	94
	M3	0.5599	252	262



(a) RMSE (b) MAE

Fig. 3. Training process of compared models on D1.



(a) RMSE (b) MAE

Fig. 4. Training process of compared models on D2.

c) **NSLF is able to achieve satisfactory prediction accuracy for missing data of an HiDS matrix.** According to Table III, M1, i.e., a proposed NSLF model outperforms M3 in terms of prediction accuracy on D1 and D2. For instance, as shown in Fig. 3(a), on D1, the lowest RMSE of M1 is 0.7854, which is about 0.25% lower than 0.7874 by M3. Moreover, M1's prediction accuracy is comparable with M2. For instance, on D1, M1 and M2's RMSE is 0.7854 and 0.7844, respectively. The gap is only 0.12%. Similar results on are obtained on MAE.

V. CONCLUSIONS

This paper innovatively proposes an NSLF 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 HiDS matrix.

Although NSLF has shown superior performance in addressing HiDS matrices, further issues still remain open. As shown in Section 3(B), many hyperparameters need to be careful manual tuning. The process is time-consuming. Hence, it seems highly promising to implement self-adaptation of hyperparameters for NSLF. According to prior research [27], an evolutionary-computation-based algorithm like particle swarm optimization can be useful in implementing efficient adaptations of hyper-parameters for LF models. We plan to address this challenging issue as future work.

REFERENCES

[1] X. Luo, M.-C. Zhou, S. Li, Z.-H. You, Y.-N. Xia, and Q.-S. Zhu, "A non-negative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 27, no. 3, pp. 579–592, 2016.

[2] 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.

[3] M.-S. Shang, X. Luo, Z.-G. Liu, J. Chen, Y. Yuan, and M.-C. Zhou, "Randomized latent factor model for high-dimensional and sparse matrices from industrial applications," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 1, pp. 131-141, 2019.

[4] X. Luo, M.-C. Zhou, S. Li, L. Hu, and M.-S. Shang, "Non-negativity Constrained Missing Data Estimation for High-dimensional and Sparse Matrices from Industrial Applications". *IEEE Trans. on Cybernetics,* DOI: 10.1109/TSMC.2019.2931468, 2019.

[5] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Networks*, vol. 12, no. 1, pp. 145-151, 1999.

[6] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," *Advances in Neural Information Processing Systems*, vol. 20, pp. 1257-1264, 2008.

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

[8] G. Takács, I. Pilászy, Bottyán Németh, and D. Tikky, "Scalable collaborative filtering approaches for large recommender systems," *Journal of Machine Learning Research*, vol. 10, pp. 623-656, 2009.

[9] 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.

[10] X. Luo, J.-P Sun, Z.-D Wang, S. Li, and M.-S Shang, "Symmetric and Non-negative Latent Factor Models for Undirected, High Dimensional and Sparse Networks in Industrial Applications". *IEEE Trans. on Industrial Informatics,* DOI: 10.1109/TII.2017.2724769, 2017.

[11] 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.

[12] H. Shao, G. Zheng, "Convergence analysis of a back-propagation algorithm with adaptive momentum," *Neurocomputing*, vol. 74, no. 5, pp. 749-752, 2011.

[13] X. Luo, M.-C. Zhou, S. Li, and M.-S. Shang, "An Inherently Non-negative Latent Factor Model for High-dimensional and Sparse Matrices from Industrial Applications," *IEEE Trans. on Industrial Informatics,* vol. 14, no. 5, pp. 2011-2022, 2018.

[14] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. of the International Conference on Machine Learning,* DOI: 10.1007/s00287-015-0911-z, 2013.

[15] Y.-P. Dong et al., "Boosting Adversarial Attacks with Momentum," in *Proc. of the CVF Conference on Computer Vision and Pattern Recognition,* pp. 9185-9193, 2018.

[16] X. Luo, H.-J. Liu, G.-P. Gou, Y.-N. Xia, and Q.-S. Zhu, "A parallel matrix factorization based recommender by alternating stochastic gradient decent," *Engineering Applications of Artificial Intelligence,* vol. 25, no. 7, pp. 1403-1412, 2012.

[17] X. Luo, Y.-N. Xia, and Q.-S. Zhu, "Applying the learning rate adaptation to the matrix factorization based collaborative filtering," *Knowledge-Based Systems,* 2013, vol. 37, pp. 154-164, 2013.

[18] Y. Yuan, X. Luo, and M.-S. Shang, "Effects of preprocessing and training biases in latent factor models for recommender systems," *Neurocomputing*, vol. 275, pp. 2019-2030, 2018.

[19] 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.

[20] Jaen-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.

[21] C. Jiang, Y. Ma, C. Wang, "PID controller parameters optimization of hydro-turbine governing systems using deterministic-chaotic-mutation evplutionary programming (DCMEP)," *Energy Convers. Manag.*, vol. 47, no. 9, pp. 1222-1230, 2006.

[22] 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.

[23] J.-Q Han, "Nonlinear PID Controller," *Journal of automation*, vol. 20, no. 4, 1994.

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

[25] S.-D Hanwate, Y.-V Hole, "Optimal PID design for Load frequency control using QRAWCP approach," in *Proc. of the International Federation of Automatic Control,* pp. 651-656, 2018.

[26] L. Fan, E.-M Joo, "Design for auto-tuning PID controller based on genetic algorithms," *In Proc. of the Industrial Electronics and Applications*, pp. 1924-1928, 2009.

[27] Q.-X. Wang, S.-L Chen, X. Luo, "An adaptive latent factor model via particle swarm optimization," *Neurocompting,* vol. 369, no. 5, pp. 176-184, 2019.

[28] X. Yang and B. Zhao, "Optimal neuro-control strategy for nonlinear systems with asymmetric input constraints," *IEEE/CAA J. Autom. Sinica*, vol. 7, no. 2, pp. 575–583, 2020.

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

[30] R.-Z. Song and L. Zhu, "Optimal fixed-point tracking control for discrete-time nonlinear systems via ADP," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 3, pp. 657-666, 2019.

[31] X. Luo, M.-C. Zhou, S. Li, and M-S. Shang, "Algorithms of Unconstrained Non-negative Latent Factor Analysis for Recommender Systems", *IEEE Transactions on Big Data*, DOI: 10.1109/TBDATA.2019.2916868, 2019.

[32] 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 Transactions on System Man Cybernetics: Systems*, DOI: 10.1109/TSMC.2019.2930525, 2019.

[33] X. Luo, Y. Yuan, M-C. Zhou, Z-G. Liu, and M-S. Shang, "Non-negative Latent Factor Model based on β-divergence for Recommender Systems", *IEEE Transactions on System Man Cybernetics: Systems*, DOI: 10.1109/TSMC.2019.2931468, 2019.

[34] 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 Transactions on Knowledge and Data Engineering*, DOI: 10.1109/TKDE.2020.3014302, 2020.

[35] D. Wu, X. Luo, M-S Shang, Y. He, G-Y. Wang, and M-C. Zhou, "A Deep Latent Factor Model for High-Dimensional and Sparse Matrices in Recommender Systems", *IEEE Transactions on System Man Cybernetics: Systems*, DOI: 10.1109/TSMC.2019.2931393, 2019.

[36] 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 Transactions on Big Data*, DOI: 10.1109/TBDATA.2020.2973141, 2020.

[37] H. Wu, X. Luo, and M-C. Zhou, "Advancing Non-negative Latent Factorization of Tensors with Diversified Regularizations", *IEEE Transactions on Services Computing*, DOI: 10.1109/TSC.2020.2988760, 2020.