An L_1-and-L_2-norm-oriented Latent Factor Model for Recommender Systems

Di Wu, *Member*, *IEEE*, Mingsheng Shang, Xin Luo, *Senior Member*, *IEEE*, and Zidong Wang, *Fellow*, *IEEE*

[1] ***Abstract***—A recommender system is highly efficient in filtering people's desired information from high-dimensional and sparse (HiDS) data. To date, a latent factor (LF)-based approach becomes highly popular when implementing a recommender system. However, current LF models mostly adopt single distance-oriented *Loss* like an L_2 norm-oriented one, which ignores target data's characteristics described by other metrics like an L_1 norm-oriented one. To investigate this issue, this paper proposes an L_1-and-L_2-norm-oriented Latent Factor (L^3F) model. It adopts two-fold ideas: a) aggregating L_1 norm's robustness and L_2 norm's stability to form its *Loss*, and b) adaptively adjusting weights of L_1 and L_2 norms in its *Loss*. By doing so, it achieves fine aggregation effects with L_1 norm-oriented *Loss*'s robustness and L_2 norm-oriented *Loss*'s stability to precisely describe HiDS data with outliers. Experimental results on nine HiDS datasets generated by real systems show that an L^3F model significantly outperforms state-of-the-art models in prediction accuracy for missing data of an HiDS dataset. Its computational efficiency is also comparable with the most efficient LF models. Hence, it has good potential for addressing HiDS data from real applications.

Keywords**—**Recommender System, High-dimensional and Sparse Matrix, Latent Factor Analysis, L_1 Norm, L_2 Norm.

I. Introduction

In this era of information explosion, people are inundated by big data [1]. For instance, Google's data come to PBs and Flickr generates TBs every day [2]. How to implement an intelligent system to filter desired information out of such big data is highly challenging [1, 3, 4]. Recommender system (RS) is highly useful in addressing this issue [5,6]. So far, various approaches are proposed to implement an RS, where collaborative filtering (CF) is highly popular [7-13].

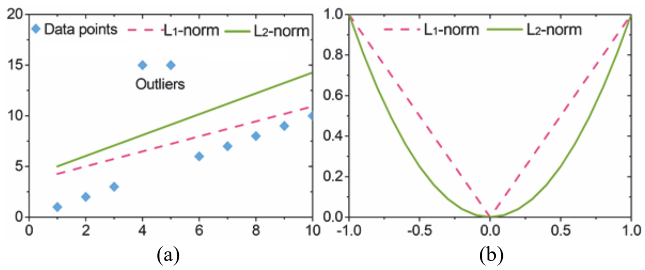


Fig. 1. Differences between L_1 and L_2 norms: (a) fitness on ten instances with two outliers, (b) loss functions with L_1 norm and L_2 norm respectively when the difference between predictions and ground truth is small (less than 1).

Great efforts have been made to achieve various CF-based RSs, where a latent factor (LF) model [14-16] is widely adopted owing to its high efficiency and scalability in industrial applications [17, 18]. Commonly, an LF model is developed based on a user-item rating matrix [3, 19], where each row denotes a specific user, each column denotes a specific item (e.g., movie, electronic product, and music), and each entry denotes a user's preference on an item. Note that a user cannot touch all items since the item count can be huge in an industrial RS like Amazon [20]. Thus, a user-item matrix is commonly high-dimensional and sparse (HiDS).

Given an HiDS matrix, an LF model maps both users and items into the same low-dimensional LF space to train the desired LFs based on the observed entries only, and then estimate its missing entries relying heavily on these trained LFs [9]. An LF model's objective function commonly has the form of *Loss*+*Penalty* [18, 21] that should be carefully designed. On an HiDS matrix, the *Loss* is the sum error between its observed entries and corresponding estimates generated by an LF model, and the *Penalty* commonly consists of regularization terms to prevent an LF model from overfitting.

Considering the *Loss* of an LF model, it usually depends on L_1 or L_2 norm defined on the known data of an HiDS matrix. Fig .1 illustrates the differences between L_1 norm-oriented and L_2 norm-oriented *Losses*:

a) The former is less sensitive to outliers than the latter, thereby enhancing the robustness of a resultant model [22-25] as shown in Fig. 1(a); and
b) The latter is smoother than the former when the predictions and ground truth data are close, thereby enhancing the stability of a resultant model [26] as shown in Fig. 1(b).

An HiDS matrix generated by real RSs are commonly filled with outliers by malicious users (e.g., a user always

This work is supported in part by the National Natural Science Foundation of China under Grants 61702475, 61772493, 62002337, and 61902370, in part by CAAI-Huawei MindSpore Open Fund under grant CAAIXSJLJJ-2020-004B, in part by the Natural Science Foundation of Chongqing (China) under grants cstc2019jcyj-msxmX0578 and cstc2019jcyjjqX0013, in part by the Chinese Academy of Sciences "Light of West China" Program, in part by the Technology Innovation and Application Development Project of Chongqing, China under grants cstc2018jszx-cyzdX0041and cstc2019jscx-fxydX0027, and in part by the Pioneer Hundred Talents Program of Chinese Academy of Sciences. D. Wu and X. Luo contributed equally to this study and are co-first authors of this paper (*Corresponding Author*: *X. Luo*).

D. Wu, and M. Shang are with the 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 (e-mail: {wudi, msshang}@cigit.ac.cn).

X. Luo is with the 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 also with the Department of Big Data Analyses Techniques, Cloudwalk, Chongqing 401331, China (e-mail: luoxin21@cigit.ac.cn).

Z. Wang is with the Department of Computer Science, Brunel University London, Uxbridge, Middlesex, UB8 3PH, United Kingdom. (Email: Zidong.Wang@brunel.ac.uk).

D. Wu and X. Luo are also with the Chongqing School, University of Chinese Academy of Sciences, Chongqing 400714, China.

assigns extremely high or low raring to an item) [27, 28], which can greatly impair the performance of an LF model relying on L_2 norm-oriented *Loss* only. On the other hand, an LF model relying on L_1 norm-oriented *Loss* solely suffers from unstable prediction performance, which can greatly harm its overall prediction accuracy for a user's unknown rating on an item. From this point of view, an LF model with a *Loss* relying on L_1 or L_2 norm only fails in well describing an HiDS matrix [25, 29].

To the authors' best knowledge, an existing LF model's *Loss* commonly depends solely on L_1 norm, L_2 norm, or other distance metrics like Kullback-Leibler Divergence [26, 27, 30-35]. Related studies [36-41] also propose to combine both L_1 and L_2 norms in the *Penalty* rather than *Loss* of an LF model. How will an LF model perform with its *Loss* relying on multiple norms which are efficiently aggregated? What is the theoretical evidence behind its performance? Motivated by these critical issues, this study proposes an L_1-and-L_2-norm-oriented latent factor (L^3F) model. It is significantly different from existing LF models owing to its L_1-and-L_2-norm-oriented *Loss* with an adaptive weighting strategy for well aggregating the effects of both L_1 and L_2 norms. It achieves both robustness and stability to well describe an HiDS matrix from a recommender system. Main contributions of this study include:

a) An L^3F model is proposed. It efficiently aggregates L_1 norm-oriented *Loss*'s robustness and L_2 norm-oriented *Loss*'s stability to well describe an HiDS rating matrix. Hence, it is robust to outliers in an HiDS rating matrix, as well as achieves high prediction accuracy for missing data of an HiDS rating matrix;
b) Theoretical analyses and proofs showing an L^3F model's ability to aggregate the effects of L_1 and L_2 norm-oriented *Losses* are presented; and
c) Algorithm design and analysis for an L^3F model.

Empirical studies on nine HiDS matrices generated by real RSs are carefully conducted to evaluate L^3F's performance. Results demonstrate that compared with state-of-the-art LF models, an L^3F model achieves significant accuracy gain when predicting missing data of an HiDS rating matrix. Its computational efficiency is also highly competitive when compared with the most efficient LF models.

Note that a recommender system is a learning system highly efficient in filtering people's desired information out from big data [3, 5, 6]. The L^3F proposed by this study can promote the development and applications of related recommender systems [14-16]. Moreover, this study shows that L^3F can well represent HiDS data with outliers by aggregating L_1 norm-oriented *Loss*'s robustness and L_2 norm-oriented *Loss*'s stability [22, 23, 26]. Its principle can be adopted to address similar issues raised by neural network-based learning systems [2, 51, 52, 59].

Section II gives the preliminaries. Section III proposes an L^3F model. Section IV presents empirical studies. Finally, Section V concludes this paper.

II. Preliminaries

A. Symbols and Notations

Please refer to Table S.I in the Supplementary File.

B. Related Work

An LF model is widely adopted to implement an RS [17, 18]. So far, various sophisticated LF models have been proposed, including a bias-based one [18], a nonparametric one [31], a non-negativity-constrained one [19], a probabilistic one [30], a dual-regularization-based one [33], a posterior-neighborhood-regularized one [42], a randomized one [43], a graph regularized one [44], a neighborhood-and-location integrated one [32], a confidence-driven one [34], and a data characteristic-aware one [35]. Although they are different from each other in objective functions or learning algorithms, they all adopt an L_2 norm-oriented *Loss* that is highly sensitive to outliers [25, 27, 28]. To make an LF model less sensitive to outlier data, Zhu *et al*. propose to adopt an L_1-norm-oriented *Loss* [24]. However, an LF model with an L_1 norm-oriented *Loss* has possibly multiple solution spaces because L_1 norm is less smooth than L_2 norm.

On the other hand, matrix completion [36-39] or feature representation [40, 41] models adopt both L_1 and L_2 norms to construct their *Penalty*, thereby achieving model sparsity [45] or generality [38]. Nonetheless, as mentioned before, *Penalty* and *Loss* are two different and critical components of an LF model's learning objective. *Penalty* affects a resultant model's own characteristics like model sparsity, while *Loss* mostly decides how an achieved model describes target data. But different from existing models, an L^3F model investigates the effects by multiple norms-oriented *Loss*. It implements an L_1-and-L_2-norm-oriented *Loss* where the effects of L_1 and L_2 norms are aggregated efficiently via an adaptive weighting strategy, thereby achieving both robustness to outliers and model stability simultaneously.

Recently, deep neural networks (DNN) [46]-based approaches to an RS attract researchers' attention [47-50]. Zhang *et al*. conduct a detailed review of DNN-based RSs [51]. Sophisticated DNN-based recommender models include an autoencoder-based one [52], a hybrid autoencoder-based one [53], a multitask learning-oriented one [54], a neural factorization-based one [55], an attentional factorization-based one [56], a deep cooperative neural network model [57], and a convolutional matrix factorization model [58]. However, when addressing HiDS data, a DNN-based model's performance is achieved with high computation burden [59-61], while an L^3F model does not suffer from such limitations. Note that in Section IV(E), some DNN-based models [55-58] mentioned above are not compared since they are defined on different data sources. More specifically, models proposed in [55-56] focus on implicit feedbacks like log information rather than explicit feedbacks like ratings concerned in this study. Models proposed in [57-58] rely on additional review information.

C. Problem Definition

In our context, an HiDS matrix is defined as in [9, 16, 18]:
***Definition* 1**. Given a user set U and an item set I, $Y^{|U|\times|I|}$ is a matrix where each element $y_{u,i}$ describes user $u\in U$'s preference on item $i\in I$. Let R_K and R_U denote its known and unknown entry sets, respectively. Y is HiDS if $|R_K|\ll|R_U|$.

Given Y, an LF model is built on R_K as defined in [17, 18]:
***Definition* 2.** Given Y and f, an LF model aims to achieve LF

matrices $P^{|U|\times f}$ and $Q^{|I|\times f}$ for Y's rank-f approximation $\hat{Y}=PQ^{\mathrm{T}}$ based on R_K only with $f\ll\min\{|U|, |I|\}$.

Thus, an objective function defined on R_K is highly desired to achieve P and Q. It commonly has the form of *Loss*+*Penalty* with P and Q [18], where *Loss* quantifies distance between Y and $\hat{Y}$ and *Penalty* generalizes the achieved model. So the learning objective of an LF model is given as:

$$\mathop{\arg\min}_{P,Q} \varepsilon(P,Q)= L\left(P,Q\right)+\lambda F\left(P,Q\right), \tag{1}$$

where $L(P, Q)$ is the *Loss*, $F(P, Q)$ is the *Penalty*, and λ is the constant adjusting the *Penalty* effects, respectively. Most existing LF models adopt a unique norm-oriented *Loss* like an L_2 norm-oriented one with linear bias [18]:

$$L_2\left(P,Q\right)=\left\|\Omega\odot\left(Y-W-\hat{Y}\right)\right\|_{L_2}^2=\left\|\Omega\odot\left(Y-W-PQ^T\right)\right\|_{L_2}^2, \tag{2}$$

where $\odot$ denotes the Hadamard product performing the element-wise multiplication between two matrices, W is a linear bias matrix consisting of linear bias related with $\forall y_{u,i}\in R_K$, and Ω is a $|U|\times|I|$ binary indexing matrix given as

$$\Omega_{u,i}=\begin{cases}1 & \text{if } y_{u,i}\in R_K,\\ 0 & \text{otherwise.}\end{cases} \tag{3}$$

Note that L_2 norms in (2) can be replaced with L_1 norms as:

$$L_1\left(P,Q\right)=\left\|\Omega\odot\left(Y-W-\hat{Y}\right)\right\|_{L_1}=\left\|\Omega\odot\left(Y-W-PQ^T\right)\right\|_{L_1}. \tag{4}$$

Considering $F(P, Q)$, it often depends on the L_2 norm of P and Q to enhance an LF model's generality:

$$F\left(P,Q\right)=\left\|P\right\|_{L_2}^2+\left\|Q\right\|_{L_2}^2. \tag{5}$$

III. An L_1-and-L_2-norm-oriented Latent Factor Model

A. Objective Formulation

As shown above, either L_1 or L_2 norm-oriented *Loss* has its own advantage in describing HiDS data [25, 29]. To aggregate their effects, an L_1-and-L_2-norm-oriented *Loss* is built as:

$$\begin{aligned}\mathop{\arg\min}_{P,Q} \varepsilon(P,Q)&=\alpha_1\left\|\Omega\odot\left(Y-W-PQ^{\mathrm{T}}\right)\right\|_{L_1}\\ &+\alpha_2\left\|\Omega\odot\left(Y-W-PQ^{\mathrm{T}}\right)\right\|_{L_2}^2+\lambda\left(\left\|P\right\|_{L_2}^2+\left\|Q\right\|_{L_2}^2\right),\end{aligned} \tag{6}$$

where α_1 and α_2 are aggregation weights controlling the effects of L_1 and L_2 norms, respectively. Note that we make $\alpha_1+\alpha_2=1$ and α_1, $\alpha_2\geq0$ to maintain numerical magnitude of the *Loss*. Moreover, due to Y's extreme sparsity and R_K's imbalance (e.g., some users are related with many ratings and the others are related with few), it is vital to expand (6) into a density-oriented form [17, 19]:

$$\begin{aligned}&\mathop{\arg\min}_{P,\,Q} \varepsilon(P,\,Q)=\\ &\underbrace{\underbrace{\alpha_1\sum_{(u,i)\in R_K}\left|y_{u,i}-w_{u,i}-\sum_{d=1}^{f}p_{u,d}q_{i,d}\right|}_{L_1\text{-norm-oriented Loss}}+\underbrace{\alpha_2\sum_{(u,i)\in R_K}\left(y_{u,i}-w_{u,i}-\sum_{d=1}^{f}p_{u,d}q_{i,d}\right)^2}_{L_2\text{-norm-oriented Loss}}}_{L_1\text{-and-}L_2\text{-norm-oriented Loss}}\\ &\underbrace{+\lambda\sum_{(u,i)\in R_K}\left(\sum_{d=1}^{f}\left(p_{u,d}\right)^2+\sum_{d=1}^{f}\left(q_{i,d}\right)^2\right)}_{L_2\text{-norm-oriented Penalty}},\end{aligned} \tag{7}$$

where $p_{u,d}$ and $q_{i,d}$ denote specific entries in P and Q, $w_{u,i}$ is given by $w_{u,i}=\mu+b_u+b_i$ as μ denotes the global-average of R_K, b_u denotes the observed deviations on user u, and b_i denotes the observed deviations on item I, respectively. Note that in (7), the regularization effect on each LF is specified with its relevant known rating count [17, 19], thereby implementing a finely-grained control of the regularization effects.

With (7), an L^{3}F model reasonably aggregates the merits of L_1 norm-based and L_2 norm-oriented *Losses*, i.e., model robustness and stability. Note that when $w_{u,i}=0$, (7) involves no linear bias, which can be considered as a special case of an L^{3}F model. Hereafter we mark models with and without linear bias as L^{3}F$_{\bar{b}}$ and L^{3}F$_b$.

B. Model Optimization

The optimization of (7) with P and Q can be achieved by various learning algorithms. As discussed in [17, 19], a stochastic gradient descent (SGD) algorithm is efficient to do so. It considers the instant loss on a single rating $y_{u,i}$ in (7) as:

$$\begin{aligned}\varepsilon_{u,i}&=\alpha_1\left|y_{u,i}-w_{u,i}-\sum_{d=1}^{f}p_{u,d}q_{i,d}\right|+\alpha_2\left(y_{u,i}-w_{u,i}-\sum_{d=1}^{f}p_{u,d}q_{i,d}\right)^2\\ &+\lambda\left(\sum_{d=1}^{f}\left(p_{u,d}\right)^2+\sum_{d=1}^{f}\left(q_{i,d}\right)^2\right).\end{aligned} \tag{8}$$

Then, in the nth iteration, it moves each single LF along the opposite direction of the stochastic gradient of (8) with it as:

$$\forall d\in\{1,2,...,f\}:\begin{cases}p_{u,d}^{n}=p_{u,d}^{n-1}-\eta\dfrac{\partial\varepsilon_{u,i}^{n-1}}{\partial p_{u,d}^{n-1}},\\ q_{i,d}^{n}=q_{i,d}^{n-1}-\eta\dfrac{\partial\varepsilon_{u,i}^{n-1}}{\partial q_{i,d}^{n-1}};\end{cases} \tag{9}$$

where $\varepsilon_{u,i}^{n\text{-}1}$, $p_{u,d}^{n\text{-}1}$, and $q_{i,d}^{n\text{-}1}$ denote the states of ε, $p_{u,d}$, and $q_{i,d}$ during the (n-1)th iteration, and η denotes the learning rate, respectively. Let $\Delta_{u,i}=y_{u,i}-w_{u,i}-\sum_{d=1}^{f}p_{u,d}q_{i,d}$, then the expression of (8) actually depends on the sign of $\Delta_{u,i}$:

$$\begin{cases}\Delta_{u,i}^{n-1}\geq0:\begin{cases}\varepsilon_{u,i}^{n-1}=\alpha_1^{n-1}\Delta_{u,i}^{n-1}+\alpha_2^{n-1}\left(\Delta_{u,i}^{n-1}\right)^2\\ +\lambda\left(\sum_{d=1}^{f}\left(p_{u,d}^{n-1}\right)^2+\sum_{d=1}^{f}\left(q_{i,d}^{n-1}\right)^2\right)\end{cases}\\ \Delta_{u,i}^{n-1}<0:\begin{cases}\varepsilon_{u,i}^{n-1}=-\alpha_1^{n-1}\Delta_{u,i}^{n-1}+\alpha_2^{n-1}\left(\Delta_{u,i}^{n-1}\right)^2\\ +\lambda\left(\sum_{d=1}^{f}\left(p_{u,d}^{n-1}\right)^2+\sum_{d=1}^{f}\left(q_{i,d}^{n-1}\right)^2\right)\end{cases}\end{cases}, \tag{10}$$

where $\alpha_1^{n\text{-}1}$, $\alpha_2^{n\text{-}1}$ and $\Delta_{u,i}^{n\text{-}1}$ are the states of α_1, α_2 and $\Delta_{u,i}$ during the (n-1)th iteration, respectively.

By combining (9) and (10), we have the following scheme:

On $\mathrm{y}_{u,i}, \forall d\in\{1,2,...,f\}$:

$$\begin{cases}\Delta_{u,i}^{n-1}>0:\begin{cases}p_{u,d}^{n}=p_{u,d}^{n-1}+\alpha_1^{n-1}\eta q_{i,d}^{n-1}+\alpha_2^{n-1}\eta q_{i,d}^{n-1}\Delta_{u,i}^{n-1}-\eta\lambda p_{u,d}^{n-1}\\ q_{i,d}^{n}=q_{i,d}^{n-1}+\alpha_1^{n-1}\eta p_{u,d}^{n-1}+\alpha_2^{n-1}\eta p_{u,d}^{n-1}\Delta_{u,i}^{n-1}-\eta\lambda q_{i,d}^{n-1}\end{cases}\\ \Delta_{u,i}^{n-1}<0:\begin{cases}p_{u,d}^{n}=p_{u,d}^{n-1}-\alpha_1^{n-1}\eta q_{i,d}^{n-1}+\alpha_2^{n-1}\eta q_{i,d}^{n-1}\Delta_{u,i}^{n-1}-\eta\lambda p_{u,d}^{n-1}\\ q_{i,d}^{n}=q_{i,d}^{n-1}-\alpha_1^{n-1}\eta p_{u,d}^{n-1}+\alpha_2^{n-1}\eta p_{u,d}^{n-1}\Delta_{u,i}^{n-1}-\eta\lambda q_{i,d}^{n-1}\end{cases}\end{cases}. \tag{11}$$

C. Self-adaptive Aggregation and Proof

To finely aggregate the effects of L_1 and L_2 norm-oriented *Losses*, we make α_1 and α_2 adaptive according to the training error. Let L_1^n and L_2^n denote the partial loss depending on L_1 and L_2 norms in (7) at the nth iteration, respectively. The main idea is to increase α_1 and decrease α_2 if $L_1^n < L_2^n$, and decrease α_1 and increase α_2 otherwise. For theoretically validating the effectiveness of this strategy, we firstly present the following definitions:

***Definition* 3.** Let L_1^n and L_2^n be the states of partial *Losses* separately depending on L_1 and L_2 norms in (7) at the nth training iteration, then L_1^n and L_2^n are given as:

$$\forall u \in \{1,\ 2,\ \ldots,|U|\},\ \forall i \in \{1,\ 2,\ \ldots,|I|\}:$$
$$L_1^n = \sum_{(u,i)\in R_K} \left|\Delta_{u,i}^n\right|,\ L_2^n = \sum_{(u,i)\in R_K} \left(\Delta_{u,i}^n\right)^2. \tag{12}$$

***Definition* 4.** Let L_{12}^n be the state of L^{3}F's *Loss* based on aggregating L_1^n and L_2^n in the nth training iteration, then L_{12}^n is formulated as:

$$\forall u \in \{1,\ 2,\ \ldots,|U|\},\ \forall i \in \{1,\ 2,\ \ldots,|I|\}:$$
$$L_{12}^n = \alpha_1^n \sum_{(u,i)\in R_K} \left|\Delta_{u,i}^n\right| + \alpha_2^n \sum_{(u,i)\in R_K} \left(\Delta_{u,i}^n\right)^2 = \alpha_1^n L_1^n + \alpha_2^n L_2^n. \tag{13}$$

***Definition* 5.** Given L_1^n, L_2^n, and L_{12}^n, let C_{L1}^n, C_{L2}^n, and C_{L12}^n be the cumulative *Loss* corresponding to L_1^n, L_2^n, and L_{12}^n till the nth iteration, respectively. Then C_{L1}^n, C_{L2}^n, and C_{L12}^n are given as:

$$C_{L_1}^n = \sum_{j=1}^n L_1^j,\ C_{L_2}^n = \sum_{j=1}^n L_2^j,\ C_{L_{12}}^n = \sum_{j=1}^n L_{12}^j\ . \tag{14}$$

Based on ***Definitions*** 3–5, we present *Theorem* 1:

***Theorem* 1**. Considering an L^{3}F model, assuming that its L_1^n and L_2^n lie in the scale of [0,1]. If α_1^n and α_2^n fulfill the following condition:

$$\alpha_1^n = \frac{e^{-\gamma C_{L_1}^{n-1}}}{e^{-\gamma C_{L_1}^{n-1}} + e^{-\gamma C_{L_2}^{n-1}}},\ \alpha_2^n = \frac{e^{-\gamma C_{L_2}^{n-1}}}{e^{-\gamma C_{L_1}^{n-1}} + e^{-\gamma C_{L_2}^{n-1}}}, \tag{15}$$

then the following equality holds:

$$C_{L_{12}}^N \le min\left\{C_{L_1}^N, C_{L_2}^N\right\} + \frac{\ln 2}{\gamma} + \frac{\gamma N}{8}. \tag{16}$$

Note that in ***Theorem* 1**, α_1^n and α_2^n denote the states of α_1 and α_2 in the nth training iteration, and γ denotes a hyper-parameter controlling their learning rates. More specifically, with $\gamma = \sqrt{1/\ln N}$, the upper bound becomes $min\{C_{L1}^N,\ C_{L2}^N\}$+ln2$\sqrt{\ln N}$+$N$/ (8$\sqrt{\ln N}$). Note that the term of ln2$\sqrt{\ln N}$ + N/ (8$\sqrt{\ln N}$) is linearly bounded by the number of iterations. Hence, C_{L12}^N is comparable to the minimum of C_{L1}^N and C_{L2}^N after N training iterations. Based on *Theorem* 1, we present the following proposition:

***Proposition* 1.** Given that $\gamma = \sqrt{1/\ln N}$, if $C_{L1}^N > C_{L2}^N$, the following inequality holds:

$$C_{L_{12}}^N \le C_{L_2}^N + const < C_{L_1}^N + const, \tag{17}$$

otherwise if $C_{L1}^N < C_{L2}^N$, the following inequality holds:

$$C_{L_{12}}^N \le C_{L_1}^N + const < C_{L_2}^N + const, \tag{18}$$

where $\lim_{N\to\infty} const$=19.45.

***Remark* 1**. *Proposition* 1 states that C_{L12}^N is bounded by C_{L1}^N +*const* and C_{L2}^N+*const* with the condition of $\gamma = \sqrt{1/\ln N}$. Hence, an L^{3}F model's cumulative prediction error is always comparable to (or not larger than) that of an LF model solely built on an L_1 or L_2 norm-oriented *Loss* during the training process.

Note that to enable an L^{3}F model's practicability, the balancing coefficient γ is set self-adaptive as follows:

$$\gamma^n = \frac{1}{C_{L_1}^n + C_{L_2}^n}, \tag{19}$$

where γ^n indicates the state of γ in the nth training iteration.

The next section gives the sketch proofs of *Theorem* 1 and *Proposition* 1 for conciseness. Note that their complete proofs are provided in the Supplementary File of this paper.

1) Proof of Theorem 1

Firstly, recall the *Hoeffding Inequality* [62]:

***Lemma* 1**. Let X be a random variable fulfilling $a \le X \le b$, $\forall s\in\mathbb{R}$ the following inequality holds:

$$\ln \mathbb{E}\left[e^{sX}\right] \le s\mathbb{E}X + \frac{s^2(b-a)^2}{8}. \tag{20}$$

Note that the detailed proof of *Lemma* 1 is given in [63].□

Then note that C_{L1}^N and C_{L2}^N are bounded by:

$$A_n = e^{-\gamma C_{L_1}^n} + e^{-\gamma C_{L_2}^n}, \tag{21}$$

and A_0=1+1=2 since there is no *Loss* when n=0. Based on (21), we have the following inference:

$$\ln \frac{A_N}{A_0} \ge -\gamma \min\left\{C_{L_1}^N, C_{L_2}^N\right\} - \ln 2. \tag{22}$$

Besides, $\forall n\in\{1, \ldots, N\}$, based on (15) and (21), we have:

$$\ln \frac{A_n}{A_{n-1}} = \ln\left(\alpha_1^n e^{-\gamma L_1^n} + \alpha_2^n e^{-\gamma L_2^n}\right). \tag{23}$$

Let s=-γ and $X\in\{L_1^n, L_2^n\}$ with L_1^n and L_2^n in the scale of [0,1]. Note that α_1^n and α_2^n are interpreted as the probabilities for L_1^n and L_2^n, respectively. Thus, we have:

$$s\mathrm{E}X = -\gamma\left(\alpha_1^n L_1^n + \alpha_2^n L_2^n\right). \tag{24}$$

Then based on *Lemma* 1, (13), (23), and (24), we have

$$\ln \frac{A_n}{A_{n-1}} \le -\gamma\left(\alpha_1^n L_1^n + \alpha_2^n L_2^n\right) + \frac{\gamma^2}{8} = -\gamma L_{12}^n + \frac{\gamma^2}{8}. \tag{25}$$

Considering the accumulation of (25) as n increases from 1 to N, we have the following inferences:

$$\ln \frac{A_N}{A_0} \le -\gamma C_{L_{12}}^N + \frac{\gamma^2}{8} N. \tag{26}$$

By combining (22) and (26), the following inequality is achieved:

$$C_{L_{12}}^N \le \min\left\{C_{L_1}^N, C_{L_2}^N\right\} + \frac{\ln 2}{\gamma} + \frac{\gamma N}{8}. \tag{27}$$

Based on (27), *Theorem* 1 holds.□

2) Proof of Proposition 1

Firstly, recall the *Bernstein Inequality* [63]:

***Lemma* 2**. Let X be a random variable in the scale of [0,1], then $\forall s\in\mathbb{R}$ the following inequality holds:

$$\ln \mathrm{E}\left[e^{sX}\right] \le \left(e^s - 1\right)\mathrm{E}X. \tag{28}$$

Note that the detailed proof of *Lemma* 2 is given in [63].□

When $C^N_{L1}>C^N_{L2}$, we reduce (22) into:

$$\ln\frac{A_N}{A_0}\ge -\gamma C^N_{L_2}-\ln 2. \tag{29}$$

By combining *Lemma* 2**,** (13), (23), and (24), the following inequality is achieved:

$$\ln\frac{A_n}{A_{n-1}}\le \left(e^{-\gamma}-1\right)L^n_{12}. \tag{30}$$

Considering the accumulation of (30) according to (27) as n increases from 1 to N, we have the following inferences:

$$\ln\frac{A_N}{A_0}\le \left(e^{-\gamma}-1\right)C^N_{L_{12}}. \tag{31}$$

By combining formulas (29) and (31), we achieve:

$$C^N_{L_{12}}\le \frac{\gamma}{1-e^{-\gamma}}C^N_{L_2}+\frac{\ln 2}{1-e^{-\gamma}}. \tag{32}$$

Let $\psi=\gamma/(1\text{-}e^{-\gamma})$ and $\Phi=\ln 2/(1\text{-}e^{-\gamma})$, then we rewrite (32) as $C^N_{L12}\le\psi C^N_{L2}+\Phi$. With $\gamma=\sqrt{1/\ln N}$, we have $\lim_{N\to\infty}\psi=1$ following L'Hopital's Rule [64]. Moreover, considering the partial derivative of Φ with N, we have the inference that $\lim_{N\to\infty}(\mathrm{d}\Phi/\mathrm{d}N)\to 0$. Thus, Φ becomes a constant (i.e., 19.45 according to its curve) when $N\to\infty$. Finally, (32) can be reduced to:

$$C^N_{L_{12}}\le\left[C^N_{L_2}\right]^+ +\left[19.45\right]^- \le C^N_{L_2}+\underset{<19.45}{const}. \tag{33}$$

Hence, we obtain that $C^N_{L12}\le C^N_{L1}+const<C^N_{L2}+const$ under the condition of setting $\gamma=\sqrt{1/\ln N}$ when $C^N_{L1}>C^N_{L2}$.

Analogously, when $C^N_{L1}<C^N_{L2}$, the following result can be also achieved: $C^N_{L12}\le C^N_{L1}+const<C^N_{L2}+const$ under the condition of setting $\gamma=\sqrt{1/\ln N}$. Hence, *Proposition* 1 holds.□

D. Algorithm Design and Analysis

Based on the above inferences, we design the algorithm of L^{3}F as in Algorithm 1. As shown in Algorithm 1, its computational and storage costs are respectively $\Theta(N\times|R_K|\times f)$ and $\Theta(f\times\max\{|R_K|/f,\ |U|,\ |I|\})$, where the detailed complexity analyses are given in the Supplementary File of this paper. Note that both N and f are positive constants. Hence, an L^{3}F model's computational and storage costs are both linear with $|R_K|$ and easy to resolve in real applications.

Algorithm1. L^{3}F

Input: R_K

```
Operation                                              Cost
Initializing $f$, $\lambda$, $\eta$, $\alpha_1$=0.5, $\alpha_2$=0.5, $N$=1000   $\Theta(1)$
Initializing $P$ randomly                               $\Theta(|U|\times f)$
Initializing Q randomly                                 $\Theta(|I|\times f)$
while $n\le N$ && not converge                          $\times N$
  for each rating $y_{u,i}$ in $R_K$                    $\times|R_K|$
    for $d$=1 to $f$                                    $\times f$
      update $p^n_{u,d}$ according to (11)              $\Theta(1)$
      update $q^n_{i,d}$ according to (11)              $\Theta(1)$
    end for                                             --
  end for                                               --
  update $\alpha^n_1$ according to (15) and (19)        $\Theta(1)$
  update $\alpha^n_2$ according to (15) and (19)        $\Theta(1)$
  $n$=$n$+1                                             $\Theta(1)$
end while                                               --
```

Output: ***P***, ***Q***

IV. Experiments and Results

A. General Settings

1) Datasets

Nine HiDS datasets are adopted in the experiments [17, 65], whose details are summarized in Table I.

TABLE I
SUMMARY OF EXPERIMENTAL DATASETS

No.	Name	$\lvert U\rvert$	$\lvert I\rvert$	$\lvert R_K\rvert$	Density
D1	Dating [66]	135,359	168,791	17,359,346	0.08%
D2	Douban [17]	129,490	58,541	16,830,839	0.22%
D3	Eachmovie [3]	72,916	1,628	2,811,718	2.37%
D4	Epinion [65]	755,760	120,492	13,668,321	0.02%
D5	Flixter [67]	147,612	48,794	8,196,077	0.11%
D6	Jester [67]	24,983	100	1,186,324	47.49%
D7	MovieLens_10M [68]	71,567	65,133	10,000,054	0.21%
D8	MovieLens_20M [68]	138,493	26,744	20,000,263	0.54%
D9	*Yahoo-R2 [69]	200,000	136,736	76,344,627	0.28%

*We select its first 200,000 users with more than 76 million ratings to build D9 for conducting the experiments.

2) Evaluation Metrics

In an RS, missing data prediction and ranking prediction (top-K recommendation) are commonly adopted to evaluate its performance [51]. Considering missing data prediction, mean absolute error (MAE) and root mean squared error (RMSE) are widely adopted as the evaluation metrics [9, 19]:

$$MAE=\left(\sum_{(u,i)\in\Gamma}\left|y_{u,i}-\hat{y}_{u,i}\right|_{abs}\right)\Big/|\Gamma|,$$

$$RMSE=\sqrt{\left(\sum_{(u,i)\in\Gamma}\left(y_{u,i}-\hat{y}_{u,i}\right)^2\right)\Big/|\Gamma|},$$

where Γ denotes the testing set and $|\cdot|_{abs}$ calculates the absolute value of a given number. Note that a model's low MAE and RMSE indicate its high prediction accuracy.

On the other hand, the task of ranking prediction produces a ranked list with K items recommended to each user, where K is a cutoff parameter. Precision and normalized discounted cumulative gain (NDCG) is adopted to evaluate the ranking prediction accuracy [51, 70] of a model. The precision of a tested model is given as:

$$Precision@K=\frac{1}{|U|}\sum_{u\in U}\frac{|\Lambda_K(u)|}{K},$$

where $\Lambda_K(u)$ denotes the intersection between the top-K set generated by the model and the testing set by user u. NDCG of a tested model is given as:

$$DCG@K(u,\hat{Y})=\sum_{k=1}^{K}\frac{2^{\hat{y}_{u,k}}-1}{\log_2(i+1)},$$

$$DCG@K(u,\Gamma)=\sum_{k=1}^{K}\frac{2^{y_{u,k}}-1}{\log_2(i+1)},$$

$$NDCG@K=\frac{1}{|U|}\sum_{u\in U}\left(\frac{DCG@K(u,\hat{Y})}{DCG@K(u,\Gamma)}\right).$$

where $\hat{y}_{u,k}$ denotes the kth prediction in $\Lambda_K(u)$ in descending order and $y_{u,i}$ denotes the actual kth rating by user u in Γ in descending order. Note that both Precision and NDCG lie in the scale of [0, 1], where high Precision and NDCG of a tested model indicate its high ranking prediction accuracy.

3) Baselines

We compare L^{3}F with nine related models with different characteristics, including six LF-based models (L$_1$-LF$_{\bar{b}}$, L$_1$-LF$_b$, L$_2$-LF$_{\bar{b}}$, L$_2$-LF$_b$, NLF, and FNLF) and three DNN-based models (AutoRec, NRT, and DCCR). Note that we also test two versions of L^{3}F, i.e., L^{3}F$_{\bar{b}}$ (without linear bias) and L^{3}F$_b$ (with linear bias). Table II gives their summary.

4) Experimental Designs

In the next experiments, we aim at answering the following research questions (RQs):

a) RQ.1. Will L^{3}F achieve balanced aggregation effects with its L_1-and-L_2 norm-oriented *Loss*?
b) RQ.2. How do outlier data affect L^{3}F?
c) RQ.3. How do hyper-parameters affect L^{3}F?
d) RQ.4. Can L^{3}F outperform its peers?

In detail, we adopt 80%–20% train-test settings. The training process of a tested model terminates if the number of consumed iterations reaches a preset threshold (i.e., 1000) or the error difference between two consecutive iterations is smaller than 10^{-6}. All experiments are run on a PC with 3.4 GHz i7 CPU and 64 GB RAM. The source code of this paper is available on:
https://github.com/Wuziqiao/Resource-code.git.

TABLE II
SUMMARY OF COMPARED MODELS.

Model	Description
L$_1$-LF$_{\bar{b}}$	L$_1$-LF is a basic LF model built on an L_1-norm-oriented *Loss*. L$_1$-LF$_{\bar{b}}$ stands for an L$_1$-LF model without linear biases.
L$_1$-LF$_b$	An L$_1$-LF model with linear biases.
L$_2$-LF$_{\bar{b}}$	A sohpostictaed LF model proposed in 2009 [18]. It is built on an L_2-norm-oriented *Loss* solely. L$_2$-LF$_{\bar{b}}$ stands for an L$_2$-LF model without linear biases..
L$_2$-LF$_b$	An L$_2$-LF model with linear biases.
NLF	A sophisticated LF model proposed in 2016 [19]. It improves an L$_2$-LF model by introducing the non-negative constraints into it for efficiently describing non-negative data.
FNLF	A fast non-negative LF model based on a generalized momentum method [9]. It is proposed in 2018 [9] and also relies on an L_2-norm-oriented *Loss* solely.
AutoRec	A autoencoder-based model proposed in 2015 [52]. It is a representative DNN-based recommender model.
NRT	A DNN-based model proposed in 2017 [54]. It adopts a multi-task learning-incorporated framework based on multi-layered perceptron and recurrent neural networks.
DCCR	A DNN-based model proposed in 2019 [53]. It improves AutoRec with the consideration of two different networks.
L^{3}F$_{\bar{b}}$	A proposed L^{3}F model without linear biases.
L^{3}F$_b$	A proposed L^{3}F model with linear biases.

B. L^{3}F's Aggregation Effects (RQ.1)

In this section, we empirically study that how L^{3}F achieve balanced aggregation effects between L_1 and L_2 norm-oriented *Losses* during its training process from two aspects, i.e., monitoring the changes of α_1 and α_2 and testing L^{3}F's rating prediction accuracy.

Monitoring the changes of α_1 and α_2. The results on D1 are presented in Fig. 2. The complete results on all the datasets are recorded in Figs. S1 and S2 in the Supplementary File. From them, we find that during the training process of L^{3}F$_{\bar{b}}$/L^{3}F$_b$, both α_1 and α_2 adaptively change first and then converge to the different constants on different datasets. This phenomenon indicates that L^{3}F adaptively controls the aggregation effects of L_1 and L_2 norm-oriented *Losses* according to the data characteristics during the training process.

Testing L^{3}F's rating prediction accuracy. We compare L^{3}F with LF models built on an L_1 or L_2 norm-oriented *Loss* solely. Specifically, we conduct two sets of experiments: a) comparing L^{3}F$_{\bar{b}}$ with L$_1$-LF$_{\bar{b}}$ and L$_2$-LF$_{\bar{b}}$ in MAE and RMSE, and b) comparing L^{3}F$_b$ with L$_1$-LF$_b$ and L$_2$-LF$_b$ in MAE and RMSE. Fig. 3 presents the comparison results of models without linear biases on D1. The complete results on all the datasets are recorded in Figs. S3–S6 in the Supplementary File. Considering models without linear biases, from Figs 3, S3, and S4, we see that a) when evaluated by MAE, L^{3}F$_{\bar{b}}$ and L$_1$-LF$_{\bar{b}}$ achieve very close results and they both outperform L$_2$-LF$_{\bar{b}}$ greatly, and b) when evaluated by RMSE, L^{3}F$_{\bar{b}}$ and L$_2$-LF$_{\bar{b}}$ achieve very close results and they both outperform L$_1$-LF$_{\bar{b}}$ greatly. Considering models with linear biases, we arrive at highly similar results from Figs S5 and S6 that L^{3}F$_b$ and L$_1$-LF$_b$ achieve much lower MAE than L$_2$-LF$_b$ does, while L^{3}F$_b$ and L$_2$-LF$_b$ achieve much lower RMSE than L$_1$-LF$_b$ does.

Summary. Based on the above results, we see that by selecting α_1 and α_2 with (15), L^{3}F adaptively controls the aggregation effects of L_1 and L_2 norm-oriented *Losses* during its training process, making it achieve low MAE and RMSE simultaneously than LF models built on an L_1 or L_2 norm-oriented *Loss* solely. Hence, we conclude that L^{3}F finely aggregates the effects of L_1 and L_2 norm-oriented *Losses* to achieve such balanced performance.

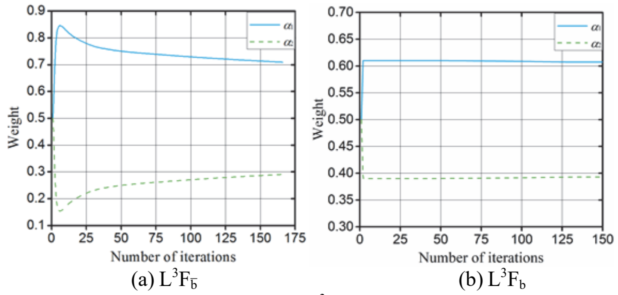


Fig. 2. The changes of α_1 and α_2 of L^{3}F during the training process on D1.

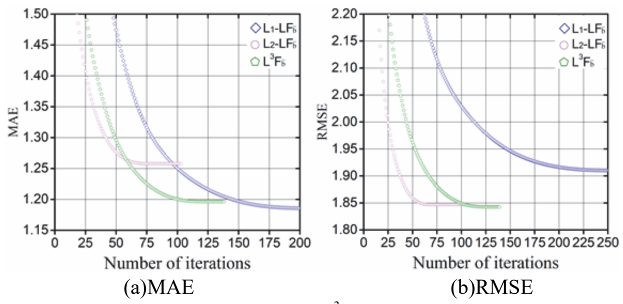


Fig. 3. The training process of L^{3}F$_{\bar{b}}$, L$_1$-LF$_{\bar{b}}$, and L$_2$-LF$_{\bar{b}}$ on D1.

C. L^{3}F's Robustness to Outlier Data (RQ.3)

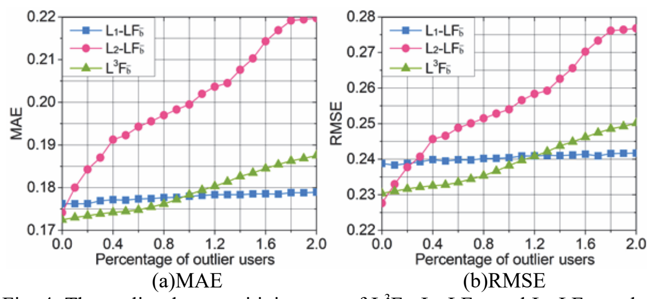
Artificial dataset. To test L^{3}F's sensitivity to outlier data, we create an artificial HiDS matrix with the following characteristics: a) it has 5000 users (rows) and 1000 items (columns), b) its known entries take 2% of the whole entry set only, which are generated at random in the range of [0, 1]. Afterwards, outlier users who randomly pick a subset of items to assign them with the same maximum or minimum rating are gradually added into this dataset. The percentage of outlier users increases from 0% to 200% with an interval of 10% to test L^3F's sensitivity to outlier data.

Note that the outlier users are added to the training set only, since in real applications we do not want a recommender to fit malicious users' attacks. Fig. S7 in the Supplementary File provides an example to further illustrate this set of experiments. The generated artificial dataset is available on https://github.com/Wuziqiao/Resource-code.git.

Analysis. We compare $L^3F_{\bar{b}}$/L^3F_b with L_1-$LF_{\bar{b}}$/L_1-LF_b and L_2-$LF_{\bar{b}}$/L_2-LF_b. Fig. 4 depicts the comparison results of models without linear biases. Figs. S8 in the Supplementary File depicts the comparison results of models with linear biases. From them, we find that outlier data have different impacts on compared models. In general, $L^3F_{\bar{b}}$/L^3F_b and L_2-$LF_{\bar{b}}$/L_2-LF_b outperform L_1-$LF_{\bar{b}}$/L_1-LF_b at initial steps. However, since L_1 norm is less sensitive to outlier data than L_2 norm, L_2-$LF_{\bar{b}}$/L_2-LF_b has the lowest prediction accuracy while L_1-$LF_{\bar{b}}$/L_1-LF_b has the highest when the percentage of outlier users grows over 100%.

Note that in several testing cases, $L^3F_{\bar{b}}$/L^3F_b outperforms L_1-$LF_{\bar{b}}$/L_1-LF_b in MAE and L_2-$LF_{\bar{b}}$/L_2-LF_b in RMSE. The reason is that $L^3F_{\bar{b}}$/L^3F_b aggregates the merits of both L_1 and L_2 norm-oriented *Losses* with a carefully-designed self-adaptive weighting strategy, which is consistent with the proof given in Section III(C). In detail, with few outlier users, $L^3F_{\bar{b}}$/L^3F_b increases the weight of L_2 norm, making it better describe such an HiDS matrix. Hence, $L^3F_{\bar{b}}$/L^3F_b's MAE is lower than that of L_1-$LF_{\bar{b}}$/L_1-LF_b. On the other hand, since L_1 norm is more robust to outlier data, $L^3F_{\bar{b}}$/L^3F_b increases the weight of L_1 norm as outlier data increase, thereby achieving lower RMSE than L_2-$LF_{\bar{b}}$/L_2-LF_b does. For example, MAE of $L^3F_{\bar{b}}$/L^3F_b (0.1725/0.1718) is lower than that of L_1-$LF_{\bar{b}}$/L_1-LF_b (0.1762/0.1737) when there are no outlier users. RMSE of $L^3F_{\bar{b}}$/L^3F_b (0.2328/0.2304) is also lower than that of L_2-$LF_{\bar{b}}$/L_2-LF_b (0.2466/0.2698) when the percentage of outlier users is 50%.

Summary. This set of experiments shows that L^3F finely aggregates the merits of L_1 and L_2 norm-oriented *Losses* to well balance the model robustness (depending on L_1 norm) and stability (depending on L_2 norm). Hence, it achieves robustness to outlier data as well as precisely describes the known data of an HiDS matrix.



(a)MAE (b)RMSE

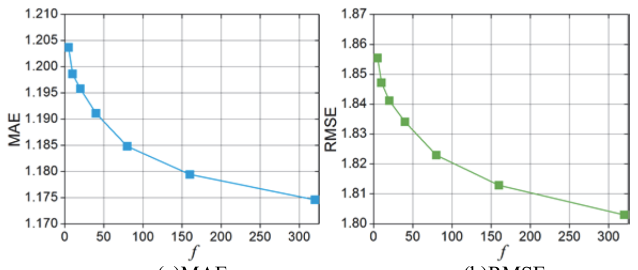
Fig. 4. The outlier data sensitivity tests of $L^3F_{\bar{b}}$, L_1-$LF_{\bar{b}}$, and L_2-$LF_{\bar{b}}$ on the artificial dataset.

D. L^3F's Sensitivity to its Hyper-parameters (RQ.3)

In this section, we analyze L^3F's behaviors with f, λ, and η.

1) With f

According to prior research [18], a large f enables an LF model to better describe an HiDS matrix unless f increases over its actual rank, which is reflexed by the increasing prediction accuracy of an LF model as f increases. Considering $L^3F_{\bar{b}}$/L^3F_b, this conclusion is also true. Fig.5 shows the prediction error of $L^3F_{\bar{b}}$ on D1 as f increases. The complete results of $L^3F_{\bar{b}}$/ L^3F_b on all the datasets are drawn in Figs. S9–S12 in the Supplementary File. From these results, we see that in most testing cases, $L^3F_{\bar{b}}$/L^3F_b's RMSE and MAE decrease as f increases (besides the testing cases on D4 where $L^3F_{\bar{b}}$/L^3F_b's MAE increases as f increases). However, this accuracy gain becomes less significant as f increases over a certain threshold, e.g., 20 on most testing cases. Besides, according to Section III(D) we see that $L^3F_{\bar{b}}$/L^3F_b's time cost increases linearly with f. Hence, f should be appropriately chosen to carefully balance prediction accuracy and computational cost. Commonly, it is chosen from the scale of [10, 20] [18, 19, 35].



(a)MAE (b)RMSE

Fig. 5. The prediction error of $L^3F_{\bar{b}}$ on D1 as f increases.

2) With λ and η

In this set of experiments, we perform a grid-based search to test the performance of $L^3F_{\bar{b}}$/ L^3F_b when λ increases from 0.01 to 0.1 and η increases from 0.0001 to 0.01. Fig.6 presents the results of $L^3F_{\bar{b}}$ on D1. The complete results of $L^3F_{\bar{b}}$/ L^3F_b on all the datasets are recorded in Figs. S13–S16 in the Supplementary File. From them, we see that:

a) Both λ and η affect the rating prediction accuracy of $L^3F_{\bar{b}}$/ L^3F_b. As λ and η increase, MAE/RMSE decreases till λ and η reach their optimal thresholds, and then increase again. For example, on D8, the RMSE of $L^3F_{\bar{b}}$ decreases from 0.7866 to 0.7745 initially, and then it increases to 0.8285 as λ and η keep increasing.

b) Generally speaking, η should be set relatively small to well balance the prediction accuracy and convergence rate, and λ should be tuned carefully since it is domain-specific [18, 31]. Similar results are also achieved by $L^3F_{\bar{b}}$/L^3F_b: On different datasets, optimal values of η are close while optimal values of λ are different. More specifically, the optimal η is around 0.001 on each dataset, while the optimal λ lies in the range of [0.02, 0.09] on D1-9 for $L^3F_{\bar{b}}$/ L^3F_b.

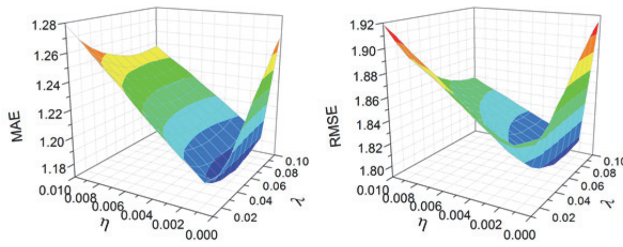
Summary. We see that L^3F's performance is closely connected with f, λ, and η. Empirically, it is suggested to set f=10–20 and η=0.001 in real applications. For λ, it should be carefully tuned on the target dataset.

TABLE III

THE COMPARISON RESULTS ON RATING PREDICTION ACCURACY, INCLUDING WIN/TIE/LOSS COUNTS STATISTIC AND FRIEDMAN TEST, WHERE ○ AND ● INDICATE THAT THE RATING PREDICTION ACCURACY OF $L^3F_{\bar{b}}$ AND L^3F_b IS HIGHER THAN OR SAME TO THAT OF COMPARISON MODELS RESPECTIVELY.

Dataset	Metric	L_1-$LF_{\bar{b}}$	L_1-LF_b	L_2-$LF_{\bar{b}}$	L_2-LF_b	NLF	FNLF	AutoRec	NRT	DCCR	$L^3F_{\bar{b}}$	L^3F_b
D1	MAE	1.1606	1.1492	1.2392○●	1.2328○●	1.2617○●	1.2588○●	1.2610○●	1.2303○●	1.2574○●	1.1740	1.1781
	RMSE	1.8672○●	1.8025○●	1.8066○●	1.7809●	1.8245○●	1.8215○●	1.8027○●	1.8101○●	1.8013○●	1.7987	1.7704
D2	MAE	0.5394●	0.5245	0.5537○●	0.5516○●	0.5590○●	0.5592○●	0.5606○●	0.5675○●	0.5581○●	0.5412	0.5347
	RMSE	0.7344○●	0.7222○●	0.7139●	0.6993	0.7150○●	0.7139●	0.7080●	0.7106○●	0.7074●	0.7145	0.7000
D3	MAE	0.1713○●	0.1705○●	0.1732○●	0.1730○●	0.1767○●	0.1763○●	0.1784○●	0.1774○●	0.1775○●	0.1700	0.1698
	RMSE	0.2290○●	0.2278○●	0.2251○	0.2262○●	0.2264○●	0.2259○●	0.2305○●	0.2301○●	0.2289○●	0.2249	0.2256
D4	MAE	0.2517○●	0.2223	0.3011○●	0.2938○●	0.3047○●	0.3036○●	0.3014○●	0.3002○●	0.3036○●	0.2512	0.2422
	RMSE	0.6100○●	0.4875●	0.5958●	0.4821●	0.5994○●	0.5977○●	0.5946●	0.5926●	0.5952●	0.5969	0.4775
D5	MAE	0.6319○●	0.6042	0.6447○●	0.6207●	0.6550○●	0.6520○●	0.6295●	0.6337○●	0.6308●	0.6318	0.6050
	RMSE	0.9098○●	0.8450●	0.8961○●	0.8383●	0.9056○●	0.9038○●	0.8682●	0.8677●	0.8792●	0.8960	0.8326
D6	MAE	0.7580○●	0.7600○●	0.7664○●	0.7676○●	0.7769○●	0.7778○●	0.7905○●	0.7878○●	0.7883○●	0.7562	0.7578
	RMSE	1.0206○●	1.0177○●	0.9957●	0.9982○●	1.0049○●	1.0003○●	1.0078○●	1.0056○●	1.0042○●	0.9978	0.9935
D7	MAE	0.5986○●	0.6038○●	0.5999○●	0.6067○●	0.6080○●	0.6068○●	0.6048○●	0.6035○●	0.6002○●	0.5956	0.5981
	RMSE	0.7953○●	0.8048○●	0.7819	0.7936○●	0.7893○	0.7881○	0.7865○	0.7834○	0.7847○	0.7821	0.7899
D8	MAE	0.5877○●	0.5927○●	0.5886○●	0.5955○●	0.5977○●	0.5961○●	0.5947○●	0.6018○●	0.5902○●	0.5848	0.5863
	RMSE	0.7873○●	0.7960○●	0.7737	0.7848○●	0.7819○●	0.7798○	0.7802○	0.7798○	0.7789○	0.7743	0.7806
D9	MAE	0.7661	0.7916○●	0.8037○●	0.8475○●	0.8342○●	0.8228○●	0.8029○●	0.8221○●	0.8058○●	0.7742	0.7840
	RMSE	1.1129○●	1.1460○●	1.0596	1.0968○●	1.0746○	1.0704○	1.0629○	1.1150○●	1.0931○●	1.0601	1.0776
Statistical Analysis	○win/tie/loss	15/0/3	12/0/6	12/0/6	13/0/5	18/0/0	17/0/1	14/0/4	16/0/2	14/0/4	—	—
	●win/tie/loss	16/0/2	14/0/4	14/0/4	17/0/1	16/0/2	15/0/3	15/0/3	16/0/2	16/0/2	—	—
	F-rank*	6.83	5.50	4.97	5.56	8.83	7.64	7.11	7.19	6.31	3.28	**2.78**

* The smaller F-rank value denotes a higher rating prediction accuracy.



(a)MAE (b)RMSE

Fig. 6. The performance of $L^3F_{\bar{b}}$ with respect to λ and η on D1.

E. Comparison between L^3F and Baselines (RQ.4)

In the comparisons, we evaluate rating prediction accuracy, ranking prediction accuracy, and computational efficiency. To draw fair comparisons, we adopt the following settings: a) setting f=20 for all the LF-based models, b) adopting five-fold cross-validations and report the average results, c) tuning the other hyper-parameters on one fold of each dataset to achieve the best performance of each model and then adopting the same values on the remaining four folds, d) adopting same random initialization method for initial solution of each model, e) adopting SGD to optimize each model with same order of training samples, f) on rating prediction comparison, respectively tuning hyper-parameters for achieving lowest MAE and RSME, g) on ranking prediction comparison, hyper-parameters are the same as that set in rating prediction comparison with lowest RMSE.

1) Comparison of rating prediction accuracy

Table III presents the detailed comparison results, where we see that $L^3F_{\bar{b}}$/L^3F_b and L_1-$LF_{\bar{b}}$/L_1-LF_b have lower MAE than the other models. On RMSE, $L^3F_{\bar{b}}$/L^3F_b, L_2-$LF_{\bar{b}}$/L_2-LF_b, AutoRec, and DCCR achieve better performance than the other models. In particular, we see that $L^3F_{\bar{b}}$/L^3F_b outperforms L_1-$LF_{\bar{b}}$/L_1-LF_b in MAE and L_2-$LF_{\bar{b}}$/L_2-LF_b in RMSE on some datasets. The reason is same as analyzed in Section IV(C), i.e., $L^3F_{\bar{b}}$/L^3F_b aggregates the merits of both robustness (depending on L_1 norm) and stability (depending on L_2 norm) to well describe an HiDS matrix with different situations of outliers. By comparing $L^3F_{\bar{b}}$ and L^3F_b, we find that linear bias can improve L^3F's rating prediction accuracy on several cases but not on all. For example, RMSE is reduced from 0.5969 to 0.4775 on D4 while increased from 1.0601 to 1.0776 on D9.

To check whether $L^3F_{\bar{b}}$/L^3F_b has a higher rating prediction accuracy than the other models, we conduct statistical analysis on Table III. First, the win/tie/loss counts of $L^3F_{\bar{b}}$/L^3F_b versus other models one by one are summarized in the third/second-to-last row of Table III, which indicates that $L^3F_{\bar{b}}$/L^3F_b achieves a higher rating prediction accuracy than other models on most datasets. Second, we perform Friedman test [71] because it is an effective statistical test method in validating the performance of multiple models on multiple datasets. Friedman test results on the MAE/RMSE of Table III are recorded in the last row of Table III, where it accepts the hypothesis that these comparison models have significant differences with a significance level of 0.05. Friedman test results show that $L^3F_{\bar{b}}$/L^3F_b has a higher rating prediction accuracy than the other models. Further, L^3F_b has a smaller F-rank value than $L^3F_{\bar{b}}$, which means that L^3F_b outperforms $L^3F_{\bar{b}}$ in general.

Besides, to check whether L^3F_b achieves significantly higher rating prediction accuracy than the other models, we conduct the Wilcoxon signed-ranks test [72, 73] on the results of Table III. In detail, we compare MAE/RMSE of L^3F_b with that of the other models one by one. Wilcoxon signed-ranks test is a nonparametric pairwise comparison procedure and has three indicators—R+, R-, and p-value. The larger R+ value indicates higher performance and the p-value indicates the significance level. Table S.II in the Supplementary File record the test results, where we see that L^3F_b has a significantly higher rating prediction accuracy than all

TABLE IV

THE COMPARISON RESULTS ON NDCG UNDER SITUATION 1, INCLUDING WIN/TIE/LOSS COUNTS STATISTIC AND FRIEDMAN TEST, WHERE ○ AND ● INDICATE THAT THE NDCG OF $L^3F_{\bar{b}}$ AND L^3F_b IS HIGHER THAN OR SAME TO THAT OF COMPARISON MODELS RESPECTIVELY.

Dataset	Metric	L_1-$LF_{\bar{b}}$	L_1-LF_b	L_2-$LF_{\bar{b}}$	L_2-LF_b	NLF	FNLF	AutoRec	NRT	DCCR	$L^3F_{\bar{b}}$	L^3F_b
D1	NDCG@5	0.9030○●	0.9058○●	0.9056○●	0.9040○●	0.9020○●	0.9025○●	0.8963○●	0.8951○●	0.8982○●	0.9068	0.9061
	NDCG @10	0.9110○●	0.9137○●	0.9137○●	0.9123○●	0.9106○●	0.9109○●	0.9052○●	0.9061○●	0.9092○●	0.9148	0.9142
D2	NDCG@5	0.8588○●	0.8600○●	0.8701	0.8703	0.8647○●	0.8643○●	0.8608○●	0.8565○●	0.8631○●	0.8696	0.8700
	NDCG @10	0.8777○●	0.8794○●	0.8869○●	0.8879	0.8817○●	0.8822○●	0.8855○●	0.8758○●	0.8862○●	0.8874	0.8877
D3	NDCG@5	0.9060○●	0.9062○●	0.9076○●	0.9059○●	0.9088	0.9090	0.9064○●	0.9071○●	0.9081	0.9077	0.9077
	NDCG @10	0.9254○●	0.9251○●	0.9267●	0.9253○●	0.9275	0.9276	0.9273	0.9288	0.9286	0.9268	0.9262
D4	NDCG@5	0.9787○	0.9786○●	0.9807	0.9795○	0.9808	0.9809	0.9731○●	0.9783○●	0.9723○●	0.9800	0.9786
	NDCG @10	0.9825○	0.9821○●	0.9835○	0.9828○	0.9850	0.9856	0.9776○●	0.9819○●	0.9762○●	0.9835	0.9821
D5	NDCG@5	0.8970○●	0.8977○●	0.8985○●	0.8982○●	0.8974○●	0.8976○●	0.8770○●	0.8820○●	0.8792○●	0.8986	0.8993
	NDCG @10	0.9113○●	0.9119○●	0.9124○●	0.9125○●	0.9114○●	0.9116○●	0.8934○●	0.8980○●	0.8952○●	0.9126	0.9132
D6	NDCG@5	0.7431○●	0.7433○●	0.7489●	0.7447○●	0.7483○●	0.7482○●	0.7294○●	0.7288○●	0.7311○●	0.7483	0.7494
	NDCG @10	0.8180○●	0.8177○●	0.8217●	0.8204○●	0.8208○●	0.8211○●	0.8071○●	0.7970○●	0.8096○●	0.8214	0.8223
D7	NDCG@5	0.8247○	0.8202○●	0.8298○	0.8231○●	0.8288○	0.8290○	0.8167○●	0.8020○●	0.8181○●	0.8299	0.8245
	NDCG @10	0.8586○●	0.8550○●	0.8627○	0.8577○●	0.8615○	0.8618○	0.8444○●	0.8348○●	0.8502○●	0.8627	0.8588
D8	NDCG@5	0.8257○	0.8213○●	0.8299	0.8235○●	0.8287○	0.8290○	0.8110○●	0.8117○●	0.8184○●	0.8297	0.8253
	NDCG @10	0.8596○	0.8561○●	0.86311	0.8582○●	0.8619○	0.8623○	0.8472○●	0.8427○●	0.8487○●	0.8630	0.8593
D9	NDCG@5	0.8020○	0.7890○●	0.8193	0.8046○	0.8103○	0.8116○	0.7785○●	0.7748○●	0.7815○●	0.8154	0.8016
	NDCG @10	0.8217○	0.8106○●	0.8338○	0.8243○	0.8293○	0.8299○	0.7985○●	0.7997○●	0.8005○●	0.8339	0.8214
Statistical Analysis	○win/tie/loss	18/0/0	18/0/0	9/2/7	16/0/2	13/1/4	14/0/4	17/0/1	17/0/1	16/0/2	—	—
	●win/tie/loss	11/0/7	16/2/0	9/0/9	12/0/6	8/0/10	8/0/10	17/0/1	17/0/1	16/0/2	—	—
	F-rank*	5.06	4.75	9.14	6.44	7.31	8.06	1.72	2.67	3.72	**9.44**	7.69

* The larger F-rank value denotes better performance on NDCG.

TABLE V

THE COMPARISON RESULTS ON PRECISION UNDER SITUATION 2, INCLUDING WIN/TIE/LOSS COUNTS STATISTIC AND FRIEDMAN TEST, WHERE ○ AND ● INDICATE THAT THE PRECISION OF $L^3F_{\bar{b}}$ AND L^3F_b IS HIGHER THAN OR SAME TO THAT OF COMPARISON MODELS RESPECTIVELY.

Dataset	Metric	L_1-$LF_{\bar{b}}$	L_1-LF_b	L_2-$LF_{\bar{b}}$	L_2-LF_b	NLF	FNLF	AutoRec	NRT	DCCR	$L^3F_{\bar{b}}$	L^3F_b
D1	Precision@5	0.0044	0.0010○●	0.0030○	0.0011○	0.0019○	0.0018○	0.0019○	0.0027○	0.0022○	0.0042	0.0010
	Precision@10	0.0031	0.0007○●	0.0022○	0.0007○●	0.0013○	0.0012○	0.0016○	0.0021○	0.0017○	0.0029	0.0007
D2	Precision@5	0.0217	0.0038○●	0.0187○	0.0038○●	0.0094○	0.0102○	0.0048○	0.0073○	0.0046○	0.0216	0.0038
	Precision@10	0.0155	0.0033○●	0.0134○	0.0034○●	0.0079○	0.0082○	0.0033○●	0.0063○	0.0031○●	0.0153	0.0034
D3	Precision@5	0.1493○	0.0273○●	0.1526○	0.0271○●	0.0881○	0.0974○	0.0492○	0.0677○	0.0523○	0.1544	0.0273
	Precision@10	0.1220○	0.0251○	0.1257○	0.0220○●	0.0779○	0.0845○	0.0310○	0.0587○	0.0351○	0.1260	0.0221
D4	Precision@5	0.0036○	0.0018○●	0.0036○	0.0018○●	0.0018○●	0.0016○●	0.0011○●	0.0015○●	0.0013○●	0.0036	0.0018
	Precision@10	0.0031○	0.0018○●	0.0031○	0.0018○●	0.0018○●	0.0017○●	0.0010○●	0.0012○●	0.0011○●	0.0031	0.0018
D5	Precision@5	0.0242○	0.0057○	0.0224○	0.0057○	0.0116○	0.0112○	0.0071○	0.0186○	0.0068○	0.0247	0.0055
	Precision@10	0.0177○	0.0055○●	0.0174○	0.0055○●	0.0106○	0.0102○	0.0067○	0.0148○	0.0065○	0.0179	0.0057
D6	Precision@5	0.9568○	0.6957○	0.9272○	0.6812○●	0.7270○	0.7150○	0.8172○	0.8732○	0.8241○	0.9577	0.6906
	Precision@10	0.8824	0.5925○	0.7683○	0.5778○●	0.6217○	0.6450○	0.7283○	0.7726○	0.7369○	0.8181	0.5812
D7	Precision@5	0.0251○	0.0105○	0.0238○	0.0099○●	0.0099○●	0.0104○	0.0084○●	0.0118○	0.0092○●	0.0254	0.0099
	Precision@10	0.0167○	0.0084○	0.0163○	0.0074○●	0.0065○●	0.0067○●	0.0055○●	0.0079○	0.0061○●	0.0168	0.0074
D8	Precision@5	0.0403○	0.0187○	0.0359○	0.0187○	0.0118○●	0.0122○●	0.0075○●	0.0174○●	0.0093○●	0.0408	0.0176
	Precision@10	0.0289○	0.0137○	0.0264○	0.0129○●	0.0084○●	0.0096○●	0.0066○●	0.0124○●	0.0075○●	0.0291	0.0130
D9	Precision@5	0.0101	0.0006○	0.0092○	0.0006○	0.0014○	0.0015○	0.0054○	0.0067○	0.0058○	0.0100	0.0005
	Precision@10	0.0077○	0.0007○	0.0068○	0.0006○●	0.0011○	0.0011○	0.0044○	0.0058○	0.0042○	0.0077	0.0006
Statistical Analysis	○win/tie/loss	9/3/6	18/0/0	16/2/0	18/0/0	18/0/0	18/0/0	18/0/0	18/0/0	18/0/0	—	—
	●win/tie/loss	0/0/18	2/6/10	0/0/18	6/8/4	3/3/12	5/0/13	7/0/11	4/0/14	7/0/11	—	—
	F-rank*	10.31	4.14	8.22	3.42	5.44	5.47	3.89	6.78	4.33	**10.53**	3.47

* The larger F-rank value denotes better performance on Precision.

the comparison models with a significance level of 0.05 except for $L^3F_{\bar{b}}$. However, we see that L^3F_b achieves a larger *R*+ when comparing with $L^3F_{\bar{b}}$, which means that linear bias can slightly boost L^3F's rating prediction accuracy in general.

2) Comparison on ranking prediction accuracy

We consider two situations. First is to check whether the models can correctly rank the items of testing set. Second is to simulate a real recommendation scenario. ***Situation* 1**: predicting ratings for each item of testing set first and then calculating NDCG. Note that the Precision of ***Situation* 1** is 1 because all the items of testing set are predicted and ranked. ***Situation* 2**: predicting ratings for the items that are not rated by the users of training set first and then calculating Precision and NDCG. Since it is extremely time-consuming to predict and rank all items under ***Situation* 2,** we followed the common strategy [47] that randomly samples 500 items.

***Situation* 1**. Table IV presents the detailed comparison results. Meanwhile, we also conduct two statistical analyses on these comparison results, i.e., win/tie/loss counts of $L^3F_{\bar{b}}$/L^3F_b versus other models and Friedman test among all the models. From Table IV, we find that a) $L^3F_{\bar{b}}$ and L_2-$LF_{\bar{b}}$ achieve a much higher NDCG than the other models, b) $L^3F_{\bar{b}}$ has a slightly better performance than L_2-$LF_{\bar{b}}$, and c) $L^3F_{\bar{b}}$ performs better than L^3F_b.

TABLE VI
THE COMPARISON RESULTS ON NDCG UNDER SITUATION 2, INCLUDING WIN/TIE/LOSS COUNTS STATISTIC AND FRIEDMAN TEST, WHERE ○ AND ● INDICATE THAT THE NDCG OF $L^3F_{\bar{b}}$ AND L^3F_b IS HIGHER THAN OR SAME TO THAT OF COMPARISON MODELS RESPECTIVELY.

Dataset	Metric	L_1-LF$_{\bar{b}}$	L_1-LF$_b$	L_2-LF$_{\bar{b}}$	L_2-LF$_b$	NLF	FNLF	AutoRec	NRT	DCCR	$L^3F_{\bar{b}}$	L^3F_b
D1	NDCG@5	0.0033○	0.0019○●	0.0030○	0.0019○●	0.0016○●	0.0015○●	0.0017○●	0.0023○	0.0019○●	0.0035	0.0019
	NDCG @10	0.0032○	0.0018○●	0.0029○	0.0018○●	0.0013○●	0.0010○●	0.0020○	0.0019○	0.0021○	0.0033	0.0018
D2	NDCG@5	0.0203	0.0048○	0.0183○	0.0048○	0.0083○	0.0093○	0.0042○●	0.0085○	0.0040○●	0.0200	0.0047
	NDCG @10	0.0188	0.0054○●	0.0173○	0.0054○●	0.0076○	0.0088○	0.0030○●	0.0096○	0.0028○●	0.0180	0.0055
D3	NDCG@5	0.1586○	0.0379○●	0.1644○	0.0378○●	0.0953○	0.1088○	0.0514○	0.0585○	0.0547○	0.1662	0.0380
	NDCG @10	0.1569○	0.0442○	0.1638○	0.0389○●	0.0917○	0.0974○	0.0342○●	0.0696○	0.0371○●	0.1639	0.0392
D4	NDCG@5	0.0028○	0.0001○●	0.0028○	0.0001○●	0.0013○	0.0012○	0.0009○	0.0013○	0.0011○	0.0028	0.0001
	NDCG @10	0.0025○	0.0001○●	0.0024○	0.0002○●	0.0012○	0.0011○	0.0009○	0.0011○	0.0010○	0.0025	0.0002
D5	NDCG@5	0.0164○	0.0013○●	0.0149○	0.0013○●	0.0088○	0.0083○	0.0075○	0.0119○	0.0071○	0.0167	0.0014
	NDCG @10	0.0142○	0.0013○●	0.0139○	0.0014○	0.0076○	0.0078○	0.0070○	0.0113○	0.0068○	0.0142	0.0013
D6	NDCG@5	0.6711	0.5610○	0.6635○	0.5495○●	0.7449	0.7396	0.7148	0.7196	0.7264	0.6704	0.5581
	NDCG @10	0.6904○	0.5863○	0.6798○	0.5786○●	0.8199	0.8220	0.7427	0.7620	0.7592	0.6923	0.5804
D7	NDCG@5	0.0212○	0.0115○	0.0213○	0.0109○●	0.0099○●	0.0102○●	0.0094○●	0.0128○	0.0099○●	0.0217	0.0109
	NDCG @10	0.0190○	0.0121○	0.0191○	0.0107○	0.0075○●	0.0087○●	0.0085○●	0.0120○	0.0087○●	0.0195	0.0105
D8	NDCG@5	0.0361○	0.0214○	0.0318○	0.0215○	0.0115○●	0.0118○●	0.0063○●	0.0198○●	0.0075○●	0.0375	0.0202
	NDCG @10	0.0336○	0.0206○	0.0312○	0.0196○●	0.0180○●	0.0182○●	0.0056○●	0.0199○	0.0063○●	0.0352	0.0196
D9	NDCG@5	0.0089	0.0007○●	0.0078○	0.0007○●	0.0014○	0.0016○	0.0066○	0.0072○	0.0069○	0.0088	0.0007
	NDCG @10	0.0088○	0.0011○	0.0075○	0.0009○●	0.0017○	0.0018○	0.0067○	0.0071○	0.0066○	0.0088	0.0009
Statistical Analysis	○win/tie/loss	10/4/4	18/0/0	17/1/0	18/0/0	16/0/2	16/0/2	16/0/2	16/0/2	16/0/2	—	—
	●win/tie/loss	0/0/18	4/5/9	0/0/18	6/8/4	6/0/12	6/0/12	8/0/10	1/0/17	7/1/10	—	—
	F-rank*	9.53	3.97	8.72	3.39	5.39	5.83	3.83	7.28	4.47	**10.08**	3.50

* The larger F-rank value denotes better performance on NDCG.

***Situation* 2**. Tables V and VI present the detailed comparison results, where win/tie/loss counts and Friedman test results are also recorded. From them, we observe that a) $L^3F_{\bar{b}}$ and L_1-LF$_{\bar{b}}$ achieve a much higher Precision and NDCG than the other models, b) $L^3F_{\bar{b}}$ has a slightly better performance than L_1-LF$_{\bar{b}}$, and c) $L^3F_{\bar{b}}$ performs better than L^3F_b.

***Analysis*.** Under ***Situation* 1,** $L^3F_{\bar{b}}$ and L_2-LF$_{\bar{b}}$ perform better than the other models, which coincide with the result obtained from rating prediction comparison. While under ***Situation* 2,** $L^3F_{\bar{b}}$ and L_1-LF$_{\bar{b}}$ do better. One reason is that both rating prediction comparison and ***Situation* 1** are tested on each item of testing set, where L_2 norm-oriented *Loss* can search for a better solution. Under ***Situation* 2,** only a small part of randomly sampled 500 items is included in the testing set. These randomly sampled items that are not included in the testing set maybe like outliers in training a model. As a result, L_1 norm-oriented *Loss* can search for a better solution. Since $L^3F_{\bar{b}}$ has the merits of both L_1 and L_2 norm-oriented *Loss*, it performs well under both situations. Note that $L^3F_{\bar{b}}$ outperforms L^3F_b under both situations, which means that linear bias tends to degrade instead of improving L^3F's ranking prediction accuracy.

3) Comparison of computational efficiency

Fig. 7 presents the CPU running time of all the models when training for rating prediction, where we observe that a) DNN-based models (AutoRec, NRT, and DCCR) cost much more CPU running time than the other models, which is caused by their DNN-based learning strategy [59], b) L^3F costs a little more CPU running time than L_1-LF and L_2-LF because it ensembles L_1 and L_2 norms, and c) L^3F cost less or more CPU running time than NLF and FNLF on the different datasets.

Note that vanilla SGD-based matrix factorization for RSs can be efficiently computed in parallel [74]. Since L^3F also belongs to the family of SGD-based matrix factorization, we can improve its computational efficiency through parallelization. On this basis, we develop L^3F to a parallel version according to Hogwild! [74]. Specifically, we randomly sample the known ratings from R_K first and then employ them to respectively update L^3F's each LF through different threads simultaneously. Please refer to [74] for details.

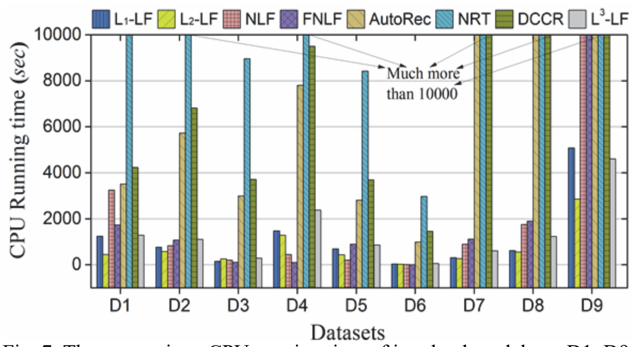
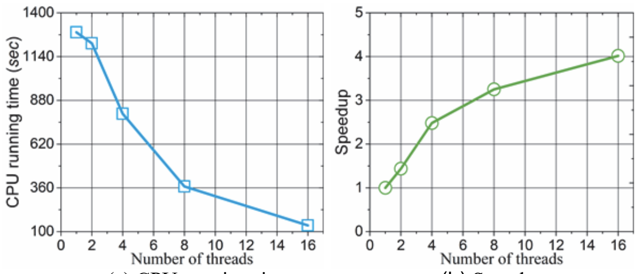


Fig. 7. The comparison CPU running time of involved models on D1–D9.



(a) CPU running time (b) Speedup

Fig. 8. The computational efficiency of parallel L^3F on D1 as the number of threads increases.

Next, we test the computational efficiency of parallel L^3F with different numbers of threads. Fig. 8 records the results on D1. The complete results on all datasets are recorded in Fig. S17 in the Supplementary File. From them, we see that L^3F's computational efficiency has been significantly improved with a nearly linear speedup as the number of threads increases. For example, the CPU running time with 16 threads (137.04 seconds) is much less than that with only one thread (1284.69 seconds, original L^3F without parallelization) on D1. Note that there are no significant differences in prediction accuracy between original L^3F and its parallel version.

4) Summary of comparison results

We compare $L^3F_{\bar{b}}/L^3F_b$ with nine related state-of-the-art models on rating prediction, ranking prediction, and computational efficiency. The comparison results verify that a) L^3F_b has a significantly higher rating prediction accuracy than the nine models, b) $L^3F_{\bar{b}}$ performs best on both situations of ranking prediction among all the models, c) linear bias has positive effects in improving L^3F's rating prediction accuracy, while it can hardly improve L^3F's ranking optimization performance, d) L^3F's computational efficiency is higher than that of DNN-based models and comparable to that of LF-based models, and e) L^3F's computational efficiency can be significantly improved through parallelization.

According to recent LF analysis on an HiDS matrix, a linear LF model like L_2-$LF_{\bar{b}}$/ L_2-LF_b may outperform DNN-based models with careful model settings. In this study, each involved model's hyper parameters are carefully tuned on D1–9 to achieve its best performance. Therefore, we find that L_1-$LF_{\bar{b}}$/L_1-LF_b, L_2-$LF_{\bar{b}}$/L_2-LF_b, NLF, and FNLF outperform AutoRec, NRT, and DCCR in terms of prediction accuracy for missing data of an HiDS matrix on several testing cases. This phenomenon is also highly consistent with the study presented in [60-61]. On the other hand, this accuracy gain is also data-dependent since the involved DNN-based models can outperform these linear models on the other testing cases. However, a proposed L^3F outperforms its peers in terms of prediction accuracy for missing data on most testing cases.

Note that a DNN-based model costs much time to build, while an LF model enjoys its high computational efficiency on HiDS data. From this point of view, a proposed L3F model can better satisfy the demands of industrial RSs for fast and accurate recommendations than its peers do.

V. Conclusions

This study proposes an L^3F model for efficiently addressing HiDS matrices arising from RSs. It adopts two-fold ideas: a) Aggregating the robustness depending on L_1 norm and stability depending on L_2 norm to form its L_1-and-L_2-norm-oriented *Loss*, and b) Adaptively adjusting weights of L_1 and L_2 norms in its *Loss*. Theoretical and empirical studies show that an L^3F model obtains fine aggregation effects with L_1 norm-oriented *Loss*'s robustness and L_2 norm-oriented *Loss*'s stability when handling an HiDS matrix with outliers.

According to Section IV(D), an L^3F model's performance is sensitive to its regularization coefficient λ, which is data-dependent and should be tuned carefully. Therefore, its self-adaptation is desired to enhance L^3F's practicability. On the other hand, will L^3F achieve further performance gain with the incorporation of more distance metrics into its *Loss*, e.g., $L_{2,1}$ norm? Naturally, more complex weighting strategies are needed to do so. We plan to address these open issues in the future.

Acknowledgments

The authors highly appreciate Dr. Yi He working at the University of Louisiana at Lafayette, Lafayette, 70503, USA, for his efforts in the theoretical studies presented in Section III(C), and Mr. Jiajia Jiang for his efforts in the empirical studies of this paper.

References

[1] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, “Data mining with big data,” *IEEE Transactions on Knowledge and Data Engineering,* vol. 26, no. 1, pp. 97-107, 2014.

[2] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, “A survey on deep learning for big data,” *Information Fusion,* vol. 42, pp. 146-157, 2018.

[3] Y. Shi, M. Larson, and A. Hanjalic, “Collaborative filtering beyond the user-item matrix: a survey of the state of the art and future challenges,” *Acm Computing Surveys,* vol. 47, no. 1, pp. 1-45, 2014.

[4] R. Lu, X. Jin, S. Zhang, M. Qiu, and X. Wu, “A study on big knowledge and its engineering issues,” *IEEE Transactions on Knowledge and Data Engineering,* vol. 31, no. 9, pp. 1630 - 1644, 2019.

[5] Y. Koren, and R. Bell, "Advances in collaborative filtering," *Recommender Systems Handbook*, pp. 77-118: Springer, 2015.

[6] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," *In Proceedings of the 2018 World Wide Web Conference*, International World Wide Web Conferences Steering Committee, 2018, pp. 689-698.

[7] A. Liu, X. Shen, Z. Li, G. Liu, J. Xu, L. Zhao, K. Zheng, and S. Shang, “Differential private collaborative Web services QoS prediction,” *World Wide Web*, pp. 1-24, 2018, DOI:10.1007/s11280-018-0544-7.

[8] D. Ryu, K. Lee, and J. Baik, “Location-based Web Service QoS Prediction via Preference Propagation to address Cold Start Problem,” *IEEE Transactions on Services Computing*, 2018, DOI: 10.1109/TSC.2018.2821686.

[9] X. Luo, Z. Liu, S. Li, M. Shang, and Z. Wang, “A fast non-negative latent factor model based on generalized momentum method,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1 - 11, 2018.

[10] L. Wu, P. Sun, R. Hong, Y. Ge, and M. Wang, “Collaborative neural social recommendation,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1-13, 2018.

[11] J.-D. Zhang, C.-Y. Chow, and J. Xu, “Enabling kernel-based attribute-aware matrix factorization for rating prediction,” *IEEE Transactions on Knowledge and Data Engineering,* vol. 29, no. 4, pp. 798-812, 2017.

[12] J. Castro, J. Lu, G. Zhang, Y. Dong, and L. Martínez, “Opinion dynamics-based group recommender systems,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* vol. 48, no. 12, pp. 2394 - 2406, 2018.

[13] W. Luan, G. Liu, C. Jiang, and L. Qi, “Partition-based collaborative tensor factorization for POI recommendation,” *IEEE/CAA Journal of Automatica Sinica,* vol. 4, no. 3, pp. 437-446, 2017.

[14] M. Gong, X. Jiang, H. Li, and K. C. Tan, “Multiobjective sparse non-negative matrix factorization,” *IEEE Transactions on Cybernetics,* vol. 49, no. 99, pp. 1-14, 2018.

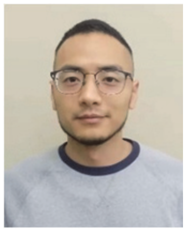
[15] X. He, J. Tang, X. Du, R. Hong, T. Ren, and T. Chua, “Fast matrix factorization with nonuniform weights on missing data,” *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1-14, 2019.

[16] Z. Li, J. Tang, and X. He, “Robust structured nonnegative matrix factorization for image representation,” *IEEE Transactions on Neural Networks and Learning Systems,* vol. 29, no. 5, pp. 1947-1960, 2018.

[17] X. Luo, Z. Wang, and M. Shang, “An instance-frequency-weighted regularization scheme for non-negative latent factor analysis on high-dimensional and sparse data,” *IEEE Transactions on Systems,*

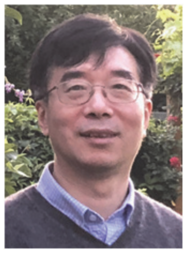
Man, and Cybernetics: Systems, pp. 1 - 11, 2019.
[18] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer,* vol. 42, no. 8, pp. 30-37, 2009.
[19] X. Luo, M. Zhou, S. Li, Z. You, Y. Xia, and Q. Zhu, "A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 27, no. 3, pp. 579-592, 2016.
[20] B. Smith, and G. Linden, "Two decades of recommender systems at Amazon. com," *Ieee internet computing,* vol. 21, no. 3, pp. 12-18, 2017.
[21] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua, "Fast matrix factorization for online recommendation with implicit feedback," *In Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, ACM, 2016, pp. 549-558.
[22] W. Ma, N. Li, Y. Li, J. Duan, and B. Chen, "Sparse Normalized Least Mean Absolute Deviation Algorithm Based on Unbiasedness Criterion for System Identification With Noisy Input," *IEEE Access,* vol. 6, pp. 14379-14388, 2018.
[23] Q. Ke, and T. Kanade, *Robust subspace computation using L1 norm*: School of Computer Science, Carnegie Mellon University Pittsburgh, PA, 2003.
[24] X. Zhu, X.-Y. Jing, D. Wu, Z. He, J. Cao, D. Yue, and L. Wang, "Similarity-maintaining privacy preservation and location-aware low-rank matrix factorization for QoS prediction based web service recommendation," *IEEE Transactions on Services Computing*, 2018.
[25] L. Wang, M. D. Gordon, and J. Zhu, "Regularized least absolute deviations regression and an efficient algorithm for parameter tuning," *In Proceedings of the 6th IEEE International Conference on Data Mining, ICDM*, 2006, pp. 690-700.
[26] R. Koenker, and K. F. Hallock, "Quantile regression," *Journal of economic perspectives,* vol. 15, no. 4, pp. 143-156, 2001.
[27] B. Lakshminarayanan, G. Bouchard, and C. Archambeau, "Robust Bayesian matrix factorisation," *In Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, 2011, pp. 425-433.
[28] C. Wu, W. Qiu, Z. Zheng, X. Wang, and X. Yang, "QoS prediction of web services based on two-phase K-means clustering," *In Proceeding of 2015 IEEE International Conference on Web Services, ICWS*, 2015, pp. 161-168.
[29] T. Goldstein, and S. Osher, "The split Bregman method for L1-regularized problems," *SIAM journal on imaging sciences,* vol. 2, no. 2, pp. 323-343, 2009.
[30] X. Ren, M. Song, E. Haihong, and J. Song, "Context-aware probabilistic matrix factorization modeling for point-of-interest recommendation," *Neurocomputing,* vol. 241, pp. 38-55, 2017.
[31] K. Yu, S. Zhu, J. Lafferty, and Y. Gong, "Fast nonparametric matrix factorization for large-scale collaborative filtering," in Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, Boston, MA, USA, 2009, pp. 211-218.
[32] D. Ryu, K. Lee, and J. Baik, "Location-based Web Service QoS Prediction via Preference Propagation to address Cold Start Problem," *IEEE Transactions on Services Computing*, 2018.
[33] H. Wu, Z. Zhang, K. Yue, B. Zhang, J. He, and L. Sun, "Dual-regularized matrix factorization with deep neural networks for recommender systems," *Knowledge-Based Systems,* vol. 145, pp. 46-58, 2018.
[34] C. Wang, Q. Liu, R. Wu, E. Chen, C. Liu, X. Huang, and Z. Huang, "Confidence-aware matrix factorization for recommender systems," *In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018, pp. 434-442.
[35] D. Wu, X. Luo, M. Shang, Y. He, G. Wang, and X. Wu, "A data-characteristic-aware latent factor model for web service QoS prediction," *IEEE Trans. on Knowledge and Data Engineering,* 2020, doi: 10.1109/TKDE.2020.3014302.
[36] Y. Chen, H. Xu, C. Caramanis, and S. Sanghavi, "Robust matrix completion and corrupted columns," *In Proceedings of the 28th International Conference on Machine Learning, ICML*, 2011, pp. 873-880.
[37] Y. Cherapanamjeri, K. Gupta, and P. Jain, "Nearly optimal robust matrix completion," *In Proceedings of the 34th International Conference on Machine Learning*, JMLR. org, 2017, pp. 797-805.
[38] O. Klopp, K. Lounici, and A. B. Tsybakov, "Robust matrix completion," *Probability Theory and Related Fields,* vol. 169, no. 1-2, pp. 523-564, 2017.
[39] H. Mansour, D. Tian, and A. Vetro, "Factorized robust matrix completion," *Handbook of Robust Low-Rank and Sparse Matrix Decomposition: Applications in Image and Video Processing,* vol. 5, 2016.
[40] A. Argyriou, T. Evgeniou, and M. Pontil, "Multi-task feature learning," *Advances in neural information processing systems*, 2007, pp. 41-48.
[41] Y. Han, Y. Yang, and X. Zhou, "Co-regularized ensemble for feature selection," *In Proceedings of 23rd International Joint Conference on Artificial Intelligence, IJCAI*, 2013, pp. 1380-1386.
[42] D. Wu, Q. He, X. Luo, M. Shang, Y. He, and G. Wang, "A Posterior-neighborhood-regularized latent factor model for highly accurate web service QoS prediction," *IEEE Transactions on Services Computing*, pp. 1-1, 2019.
[43] M. Shang, X. Luo, Z. Liu, J. Chen, Y. Yuan, and M. Zhou, "Randomized latent factor model for high-dimensional and sparse matrices from industrial applications," *IEEE/CAA Journal of Automatica Sinica,* vol. 6, no. 1, pp. 131-141, /, 2019.
[44] C. Leng, H. Zhang, G. Cai, I. Cheng, and A. Basu, "Graph regularized Lp smooth non-negative matrix factorization for data representation," *IEEE/CAA Journal of Automatica Sinica,* vol. 6, no. 2, pp. 584-595, /, 2019.
[45] Q. Shi, H. Lu, and Y.-M. Cheung, "Rank-one matrix completion with automatic rank estimation via L1-norm regularization," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 29, no. 10, pp. 4744-4757, 2017.
[46] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature,* vol. 521, no. 7553, pp. 436, 2015.
[47] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," *In Proceedings of the 26th International World Wide Web Conference*, 2017, pp. 173-182.
[48] H.-J. Xue, X.-Y. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems," *In Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI*, 2017, pp. 3203-3209.
[49] S. Zhang, Y. Tay, L. Yao, B. Wu, and A. Sun, "DeepRec: An open-source toolkit for deep learning based recommendation," *In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI*, 2019, pp. 6581-6583.
[50] S. Zhang, L. Yao, A. Sun, S. Wang, G. Long, and M. Dong, "NeuRec: on nonlinear transformation for personalized ranking," *In Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI*, 2018, pp. 3669-3675.
[51] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Computing Surveys,* vol. 52, no. 1, pp. 5:1-5:38, 2019.
[52] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "AutoRec: Autoencoders meet collaborative filtering," *In Proceedings of the 24th International Conference on World Wide Web*, ACM, 2015, pp. 111-112.
[53] Q. Wang, B. Peng, X. Shi, T. Shang, and M. Shang, "DCCR: deep collaborative conjunctive recommender for rating prediction," *IEEE Access,* vol. 7, pp. 60186-60198, 2019.
[54] P. Li, Z. Wang, Z. Ren, L. Bing, and W. Lam, "Neural rating regression with abstractive tips generation for recommendation," *In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* 2017, pp. 345-354.
[55] X. He, and T.-S. Chua, "Neural factorization machines for sparse predictive analytics," *In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017, pp. 355-364.
[56] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, and T.-S. Chua, "Attentional factorization machines: learning the weight of feature interactions via attention networks," *In Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI*, 2017, pp. 3119-3125.
[57] L. Zheng, V. Noroozi, and P. S. Yu, "Joint deep modeling of users and items using reviews for recommendation," *In Proceedings of the 10th ACM International Conference on Web Search and Data Mining,*

WSDM 2017, pp. 425-434.
[58] D. H. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware rcommendation," *In Proceedings of the 10th ACM Conference on Recommender Systems*, 2016, pp. 233-240.
[59] Z.-H. Zhou, and J. Feng, "Deep forest: Towards an alternative to deep neural networks," *In Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI* 2017, pp. 3553-3559.
[60] S. Rendle, L. Zhang, and Y. Koren, "On the difficulty of evaluating baselines: a study on recommender systems," *CoRR,* vol. abs/1905.01395, pp. 1-19, 2019.
[61] S. Rendle, W. Krichene, Li Zhang, and J. Anderson, "Neural collaborative filtering vs. matrix factorization revisited", *In Proceedings of the 14th ACM Conference on Recommender Systems, RecSys 2020,* pp. 240-248.
[62] N. I. Fisher, and P. K. Sen, "Probability Inequalities for Sums of Bounded Random Variables," *Publications of the American Statistical Association,* vol. 58, no. 301, pp. 13-30, 1963.
[63] Nicolo, and CesaBianchi, *Prediction, Learning, and Games*, 2006.
[64] W. Rudin, *Principles of mathematical analysis*: McGraw-hill New York, 1964.
[65] P. Massa, and P. Avesani, "Trust-aware recommender systems," *In Proceedings of the 2007 ACM conference on Recommender systems*, ACM, 2007, pp. 17-24.
[66] L. Brozovsky, and V. Petricek, "Recommender system for online dating service," *arXiv preprint cs/0703042*, 2007.
[67] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *information retrieval,* vol. 4, no. 2, pp. 133-151, 2001.
[68] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, "GroupLens: applying collaborative filtering to Usenet news," *Communications of the ACM,* vol. 40, no. 3, pp. 77-87, 1997.
[69] Yahoo!, "R2-Yahoo! Music User Ratings of Songs with Artist, Album, and Genre Meta Information, v. 1.0 (1.4 Gbyte & 1.1 Gbyte)," https://webscope.sandbox.yahoo.com/catalog.php?datatype=r.
[70] X. He, T. Chen, M.-Y. Kan, and X. Chen, "TriRank: review-aware explainable recommendation by modeling aspects," *In Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, 2015, pp. 1661-1670.
[71] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research,* vol. 7, no. Jan, pp. 1-30, 2006.
[72] B. Rosner, R. J. Glynn, and M. L. T. Lee, "The Wilcoxon signed rank test for paired comparisons of clustered data," *Biometrics,* vol. 62, no. 1, pp. 185-192, 2006.
[73] D. Wu, X. Luo, G. Wang, M. Shang, Y. Yuan, and H. Yan, "A highly accurate framework for self-labeled semisupervised classification in industrial applications," *IEEE Transactions on Industrial Informatics,* vol. 14, no. 3, pp. 909-920, 2018.
[74] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," *Advances in Neural Information Processing Systems*, 2011, pp. 693-701.


Di Wu (M'19) received the Ph.D. degree in computer application technology from Chongqing Institute of Green and Intelligent Technology (CIGIT), Chinese Academy of Sciences (CAS), Chongqing, China in 2019. Currently, he is an Associate Professor with the CIGIT, CAS. He was a visiting scholar during the time from April 2018 to April 2019 at the University of Louisiana, Lafayette, USA. His research interests include machine learning and data mining. He has published over 40 papers in *IEEE Trans. Knowl. Data Eng., IEEE Trans. Syst, Man, Cybern.: Syst., IEEE Trans. Ind. Inform., IEEE Trans. Serv. Comput., IEEE International Conference on Data Mining, ACM International World Wide Web Conferences, International Joint Conference on Artificial Intelligence*, etc. He is currently serving as an Associate Editor for *Frontiers in Neurorobotics*.

Mingsheng Shang received his Ph.D Degree in Computer Science from University of Electronic Science and Technology of China (UESTC). He joined the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China, in 2015, and is currently a Professor of computer science and engineering. His research interests include data mining, complex networks, cloud computing and their applications.

Xin Luo (M'14–SM'17) received the B.S. degree in computer science from 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. He is currently also a Distinguished Professor of computer science with the Dongguan University of Technology, Dongguan, China. His current research interests include big data analysis and intelligent control. He has published over 100 papers (including over 60 IEEE TRANSACTIONS papers). 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, IEEE Access,* and Neurocomputing. He has received the Outstanding Associate Editor reward from *IEEE/CAA Journal of Automatica Sinica* in 2020 and from *IEEE Access* in 2018.


Zidong Wang (SM'03-F'14) was born in Jiangsu, China, in 1966. He received the B.Sc. degree in mathematics in 1986 from Suzhou University, Suzhou, China, and the M.Sc. degree in applied mathematics in 1990 and the Ph.D. degree in electrical engineering in 1994, both from Nanjing University of Science and Technology, Nanjing, China.

He is currently a Professor of Dynamical Systems and Computing in the Department of Computer Science, Brunel University London, U.K. Prof. Wang's research interests include dynamical systems, signal processing, bioinformatics, control theory and their applications. He has published more than 600 papers in international journals. He is a holder of the Alexander von Humboldt Research Fellowship of Germany, the JSPS Research Fellowship of Japan, and the William Mong Visiting Research Fellowship of Hong Kong.

Prof. Wang serves (or has served) as the Editor-in-Chief for International Journal of Systems Science, the Editor-in-Chief for Neurocomputing, and an Associate Editor for 12 international journals including IEEE Transactions on Automatic Control, IEEE Transactions on Control Systems Technology, and IEEE Transactions on Signal Processing. He is a Member of the Academia Europaea, a Fellow of the IEEE, a Fellow of the Royal Statistical Society and a member of program committee for many international conferences.