Effect of Linear Biases in Latent Factor Models on High-dimensional and Sparse Matrices from Recommender Systems

Ye Yuan, Xin Luo, Ming-Sheng Shang, Xin-Yi Cai

Chongqing Key Laboratory of Big Data and Intelligent Computing
Chongqing Institute of Green and Intelligent Technology
Chinese Academy of Sciences
Chongqing, China
{yuanye, luoxin21, msshang, caixinyi}@cigit.ac.cn

***Abstract*—Latent factor (LF)-based models have been proven to be efficient in implementing recommender systems, owing to their ability to well represent high-dimensional and sparse matrices. While prior works focus on boosting both the prediction accuracy and computation efficiency of original LF model by adding linear biases to it, the individual and combinational effects by linear biases in such performance gain remains unclear. To address this issue, this work thoroughly investigates the effect of prior linear biases and training linear biases. We have investigated the parameter update rules and training processes of an LF model with different combinations of linear biases. Empirical validations are conducted on a high dimensional and sparse matrix from industrial systems currently in use. The results show that each linear bias does have positive/negative effects in the performance of an LF model. Such effects are partially data dependent; however, some linear biases like the global average can bring stable performance gain into an LF model. The theoretical and empirical results along with analysis provide guidance in designing the bias scheme in an LF model for recommender systems.**

Keywords—Recommender Systems, Latent Factor Model, Linear Biases, High-dimensional and Sparse Matrices, Industrial Applications

I. INTRODUCTION

The rapid growth of the Internet has brought human society into a high-speed developing information age. Online consumption becomes indispensable for people's daily life. Thousands of products are provided online by numerous online retailers. However, such massive online information leads to information overload, which has seriously reduced the utilization ratio of information. It is highly demanded to develop robust [1] optimal [2, 3] adaptive [4] intelligent [5-7] and efficient strategies to amplify information utilization. Therefore, recommender systems, which can connect people with their potential preferences based on their historical behaviors, have attracted people's attention.

Generally speaking, existing recommender systems fall into two categories, respectively are the Content Based (CB) [8-11] and the Collaborative Filtering (CF) [12-14] recommenders.

Owing to its computational efficiency and convenience for implementation, CF-based recommenders have been very popular during the past decade.

A CF-based recommender usually models the user preferences (either explicit ratings or implicit behaviors such as purchase history, browsing history, search patterns, or even mouse movements) on involved items into a user-item rating matrix, where high ratings commonly indicate strong preferences. Naturally, with explosively growing user and item numbers, it becomes impossible for a user to review all items, or an item to be reached by all users. Hence, the resulting user-item rating matrix is a high-dimensional and sparse (HiDS) matrix with numerous missing data. On the other hand, if we can predict these missing data with high reliability, then it is probable for the system to identify a user's potential preferences which are not yet observed.

Hence, with a CF-based recommender the problem of personalized recommendation is transformed into missing data estimation, i.e., to estimate the unknown user-item pairs of the rating matrix based on the known ones. According to recent progress in recommender systems community [15], current CF-based recommenders generally can be divided into two branches, i.e., the neighborhood-based model (NBM) [16-18] and latent factor model (LFM) [19-24]. Specifically, an NBM-based recommender works by building the relationship among items or users, and estimates unknown ratings based on known ones attached to their K-nearest-neighbors [25, 26]. Different from NBM, an LFM-based recommender maps both users and items into the same latent factor space where a user/an item is connected with a latent factor (LF) vector inferred from the existing ratings. Predictions for unknown ratings are generated heavily relying on these obtained LFs. Compared with NBM-based recommenders, an LFM-based recommender [27, 28] usually is a more accurate in predicting missing data in matrix, and b) highly efficient with low computational and storage costs. Hence, LFM-based recommenders have been frequently mentioned and investigated in the recommender systems community [29], as well as been popular in industrial applications.

Corresponding Author: X. Luo (luoxin21@cigit.ac.cn).

This research is supported in part by the Pioneer Hundred Talents Program of Chinese Academy of Sciences, in part by the National Natural Science Foundation of China under Grant No. 91646114, No. 61402198 and No. 61370150, in part by the Young Scientist Foundation of Chongqing under Grant No. cstc2014kjrc-qnrc40005, and in part by the Fundamental Research Funds for the Central Universities under Grant No. 106112016CDJXY180005 and No. CDJZR12180012.

978-1- 5090-4429- 0/17/$31.00 ©2017 IEEE.

According to prior works, a frequently adopted strategy to extend an LFM-based recommender for improving its prediction accuracy for missing data along with convergence rate is by incorporating linear biases into it. Generally speaking, there are two kinds of linear biases often adopted in an LFM.

The first kind of linear biases act similarly to the principle factors in artificial neural networks (ANN) [30-33]. Such a linear bias is trained along with the desired latent factors, so we call them the training linear biases (TLBs). Representative LFMs adopting TLB include the biased matrix factorization model [34, 35], singular value decomposition plus-plus model, biased regularized incremental simultaneous matrix factorization model [36], and probabilistic matrix factorization model [37], and non-negative latent factor model [38-40] The second kind of linear biases are based on the statistics of the known data in the target matrix. Since they are pre-computed before the training process of an LFM, we call them prior linear biases (PLBs). Representative works adopting such kind of linear biases including the neighborhood-aware matrix factorization model [18], and the incremental regularized matrix factorization model [36].

Although current works have proven that to incorporate linear biases into an LF model is effective in improving its performance, most of them follow the empirical experiences to choose the incorporation scheme of linear biases. For instance, when adopting TLBs, most current works intends to adopt user and item biases simultaneously, without further study regarding the separate and combined effect of each bias. Meanwhile, to the authors' best knowledge, systematically empirical study regarding issue is missing in previously studies, either. Note that different linear biases actually reflect different user/item features in practice. For instance, a user's PLB usually reflects his/her rating patterns, while TLB is the principle LF hidden in his/her known ratings. Answers to the following questions: a) whether TLBs and PLBs can work compatibly to achieve a more powerful LF model? And b) what are the separate and combined effects of linear biases remain unknown.

This work investigates the effects by TLBs and PLBs in an LF model thoroughly and systematically. With such efforts, it aims at providing evidence and guideline for researchers and engineers who want to implement an LF model enhanced by linear biases. The contributions of this work include:

a) Theories. With the incorporation of different combinations of TLBs and PLBs, the parameter update rules and training processes of an LF model with Euclidean distance as the loss function is investigated;

b) Empirical studies. We have conducted thoroughly and systematically empirical studies regarding the effects by various combinations of TLBs and PLBs on a typical HiDS matrix generated by industrial applications currently in use.

The rest of this paper is organized as follows. Section II presents the LF model with different combinations of TLBs and PLBs. Section III gives and analyzes the experimental results. Finally, Section IV concludes this paper.

II. LF MODEL WITH TLB AND PLB

A. LF model with Linear Biases

An LF model maps both users and items into a unique LF space of dimension f, such that user-item interactions are modeled relying on user and item LFs. Given an item set I and a user set U, then involved users' ratings on involved items can be denoted by a matrix $R^{|U|\times|I|}$ where each row vector denotes a specified user (with the f-dimensional LF vector p_u), each column vector denotes a specified item (with the f-dimensional LF vector q_i), and each entry r_{ui} indicating user u's preference on item i (note that r_{ui} could depend on either explicit or implicit preferences of user u on item i, and is usually proportional to those preferences). Then the approximation of r_{ui} is given by [18-25]:

$$\hat{r}_{ui} = p_u \cdot q_i, \tag{1}$$

where $\cdot$ computes the inner product of two vectors.

Formula (1) approximates r_{ui} relying on the inner product of p_u and q_i, on the assumption that the ratings information can be equally represented by a set of LFs. However, in real applications, some properties of a recommender system are usually neither separately related to its users nor items, e.g., the global PLB μ; some user properties are hardly affected by the items, e.g., the user PLB b_u; and some item properties are hardly affected by the users, e.g., the item TLB b_i.

Moreover, according to prior works in the area of ANN [30-33], it is also helpful to assume that the desired approximation to the objective, i.e., R in our case, relies on a small part of LFs rather than the others. In other words, it is necessary to incorporate TLBs as the additional components to the LF space. Therefore, the rating value can be represented as the prior bias [36] a_{ui} relying on PLBs, training bias t_{ui} relying on TLBs [34, 38], and the inner product of p_u and q_i [18-24, 27-29]. In other words, (1) is reformulated into:

$$\hat{r}_{ui} = a_{ui} + t_{ui} + p_u \cdot q_i. \tag{2}$$

B. Prior Bias

As discussed in prior works [18, 36], the baseline commonly consists of global, user and item PLBs, given by:

$$a_{ui} = \mu + b_u + b_i. \tag{3}$$

Note that the global average μ represents the statistic characteristics of the given training dataset. Hence, it can be estimated by the average of the observed ratings [36] as follows:

$$\mu = \sum_{r_{ui}\in\kappa} r_{ui} \Big/ |\kappa|, \tag{4}$$

where κ denotes the known entry set of R. b_u and b_i can be simply estimated with the unbiased linear estimator, formulated by [18, 36]:

$$b_i = \sum_{u\in R(i)} (r_{ui} - \mu) \Big/ \left(\theta_1 + |R(i)|\right), \tag{5}$$

$$b_u = \sum_{i\in R(u)} (r_{ui} - \mu - b_i) \Big/ \left(\theta_2 + |R(u)|\right); \tag{6}$$

where $R(i)$ denotes the set of users who have rated item i, $R(u)$

denotes the set of items user u has rated, θ_1 and θ_2 are constants which is related to the average of $R(i)$ and $R(u)$, respectively. Note that (5) depends on μ, and (6) depends on μ and b_i. However, when we consider different combinations of PLBs, these formulas should be adjusted. In that case, we can simply set the absent PLBs at zero. For instance, when we consider a PLB combination consisting of b_i and b_u but without μ, by setting μ=0 we arrive at:

$$\begin{aligned} b_i &= \sum_{u \in R(i)} r_{ui} \Big/ \left(\theta_1 + \left|R(i)\right|\right), \\ b_u &= \sum_{i \in R(u)} \left(r_{ui} - b_i\right) \Big/ \left(\theta_2 + \left|R(u)\right|\right). \end{aligned} \tag{7}$$

TABLE I. DIFFERENT COMBINATIONS OF PLBS IN AN LF MODEL

Situation	S.1	S.2	S.3	S.4	S.5	S.6	S.7	S.8
Combination	0	$\mu+b_u+b_i$	b_u+b_i	$\mu+b_u$	$\mu+b_i$	b_u	b_i	μ
Results	a_{ui}							

TABLE II. COMPUTATIONAL RULES OF ALL COMBINATIONS OF PLBS

For S.1 (S.A~S.D)	$\mu = 0$ $b_i = 0$ $b_u = 0$
For S.2 (S.A~S.D)	$\mu = \sum_{r_{ui} \in \kappa} r_{ui} \Big/ \left\|\kappa\right\|$ $b_i = \sum_{u \in R(i)} \left(r_{ui} - \mu\right) \Big/ \left(\theta_1 + \left\|R(i)\right\|\right)$ $b_u = \sum_{i \in R(u)} \left(r_{ui} - \mu - b_i\right) \Big/ \left(\theta_2 + \left\|R(u)\right\|\right)$
For S.3 (S.A~S.D)	$\mu = 0$ $b_i = \sum_{u \in R(i)} r_{ui} \Big/ \left(\theta_1 + \left\|R(i)\right\|\right)$ $b_u = \sum_{i \in R(u)} \left(r_{ui} - b_i\right) \Big/ \left(\theta_2 + \left\|R(u)\right\|\right)$
For S.4 (S.A~S.D)	$\mu = \sum_{r_{ui} \in \kappa} r_{ui} \Big/ \left\|\kappa\right\|$ $b_i = 0$ $b_u = \sum_{i \in R(u)} \left(r_{ui} - \mu\right) \Big/ \left(\theta_2 + \left\|R(u)\right\|\right)$
For S.5 (S.A~S.D)	$\mu = \sum_{r_{ui} \in \kappa} r_{ui} \Big/ \left\|\kappa\right\|$ $b_i = \sum_{u \in R(i)} \left(r_{ui} - \mu\right) \Big/ \left(\theta_1 + \left\|R(i)\right\|\right)$ $b_u = 0$
For S.6 (S.A~S.D)	$\mu = 0$ $b_i = 0$ $b_u = \sum_{i \in R(u)} r_{ui} \Big/ \left(\theta_2 + \left\|R(u)\right\|\right)$
For S.7 (S.A~S.D)	$\mu = 0$ $b_i = \sum_{u \in R(i)} r_{ui} \Big/ \left(\theta_1 + \left\|R(i)\right\|\right)$ $b_u = 0$
For S.8 (S.A~S.D)	$\mu = \sum_{r_{ui} \in \kappa} r_{ui} \Big/ \left\|\kappa\right\|$ $b_i = 0$ $b_u = 0$

In this section, statistical overall average μ, statistical user bias b_u and statistical item bias b_i are called PLB. The possible combinations of PLBs are summarized in TABLE I. The computational rules of PLBs with all combinations are given in TABLE II.

Note that PLBs are not trained along with LFs but relying on the prior estimators as (4)-(6). Hence, by incorporating into an LF model we actually training the desired TLBs and LFs on the residual matrix of R by removing the involved PLBs from its each known entries:

$$r_{ui}^{res} = r_{ui} - a_{ui} = r_{ui} - \mu - b_u - b_i, \tag{8}$$

where r_{ui}^{res} denotes the residual by removing PLBs from r_{ui}. Note that (8) only considers the situation when μ, b_i and b_u are all involved. However, with (3) and TABLE II, it is convenient to infer the other situations with different PLB combinations.

C. Training Bias

Based on the inference above, eight PLB combinations, i.e., S.1-8, are obtained through TABLES I and II, along with (8). Each combination results in a residual set of κ, where each residual is approximated as follows:

$$r_{ui}^{res} = t_{ui} + p_u \cdot q_i, \tag{9}$$

As discussed in prior works, training bias t_{ui} mainly consists of user and item TLBs, leading to the following formulation of t_{ui}:

$$t_{ui} = c_u + c_i, \tag{10}$$

where c_u and c_i denote the user and item TLBs, respectively. With (2), (8), (9) and (10), we formulate the objective function of an LF model with Euclidean distance as follows:

$$\begin{aligned} RSE = \sum_{r_{ui} \in \kappa} &\left(r_{ui} - a_{ui} - c_u - c_i - p_u \cdot q_i\right)^2 \\ &+ \lambda\left(\left\|p_u\right\|^2 + \left\|q_i\right\|^2 + c_u^2 + c_i^2\right), \end{aligned} \tag{11}$$

where $\|\cdot\|$ computes the l_2 norm of a vector, the term behind λ is the Tikhonov regularizing term for avoiding over-fitting the generalized error [31-36], the constant λ controls the regularization effect, respectively.

Naturally, it is not necessary for an LF model to incorporate both c_u and c_i into its objective function. Different combinations of TLBs also lead to different effect. With c_u and c_i, we have the TLB combinations as in TABLE III.

TABLE III. DIFFERENT COMBINATIONS OF TLBS IN AN LF MODEL

Situation	S.A	S.B	S.C	S.D
Combination	$c_u + c_i$	0	c_u	c_i

Let err_{ui} denote the term $\left(r_{ui} - a_{ui} - c_u - c_i - p_u \cdot q_i\right)$. With (2), (8), (9) and (10), along with TABLES I-III, we derive the expressions of err_{ui} as in TABLE IV. With different combinations of linear biases, we derive the parameter training rules of an LF model with Euclidean distance as the loss function and stochastic gradient descent as the solver, as in TABLE V.

TABLE IV. EXPRESSIONS OF ERRUI WITH DIFFERENT COMBINATIONS OF PLBS AND TLBS

Situation		PLB							
		S.1	***S.2***	***S.3***	***S.4***	***S.5***	***S.6***	***S.7***	***S.8***
TLB	***S.A***	r_{ui}-c_u-c_i-$p_u{\cdot}q_i$	r_{ui}-μ-b_u-b_i-c_u-c_i-$p_u{\cdot}q_i$	r_{ui}-b_u-b_i-c_u-c_i-$p_u{\cdot}q_i$	r_{ui}-μ-b_u-c_u-c_i-$p_u{\cdot}q_i$	r_{ui}-μ-b_i-c_u-c_i-$p_u{\cdot}q_i$	r_{ui}-b_u-c_u-c_i-$p_u{\cdot}q_i$	r_{ui}-b_i-c_u-c_i-$p_u{\cdot}q_i$	r_{ui}-μ-c_u-c_i-$p_u{\cdot}q_i$
	S.B	r_{ui}-$p_u{\cdot}q_i$	r_{ui}-μ-b_u-b_i-$p_u{\cdot}q_i$	r_{ui}-b_u-b_i-$p_u{\cdot}q_i$	r_{ui}-μ-b_u-$p_u{\cdot}q_i$	r_{ui}-μ-b_i-$p_u{\cdot}q_i$	r_{ui}-b_u-$p_u{\cdot}q_i$	r_{ui}-b_i-$p_u{\cdot}q_i$	r_{ui}-μ-$p_u{\cdot}q_i$
	S.C	r_{ui}-c_u-$p_u{\cdot}q_i$	r_{ui}-μ-b_u-b_i-c_u-$p_u{\cdot}q_i$	r_{ui}-b_u-b_i-c_u-$p_u{\cdot}q_i$	r_{ui}-μ-b_u-c_u-$p_u{\cdot}q_i$	r_{ui}-μ-b_i-c_u-$p_u{\cdot}q_i$	r_{ui}-b_u-c_u-$p_u{\cdot}q_i$	r_{ui}-b_i-c_u-$p_u{\cdot}q_i$	r_{ui}-μ-c_u-$p_u{\cdot}q_i$
	S.D	r_{ui}-c_i-$p_u{\cdot}q_i$	r_{ui}-μ-b_u-b_i-c_i-$p_u{\cdot}q_i$	r_{ui}-b_u-b_i-c_i-$p_u{\cdot}q_i$	r_{ui}-μ-b_u-c_i-$p_u{\cdot}q_i$	r_{ui}-μ-b_i-c_i-$p_u{\cdot}q_i$	r_{ui}-b_u-c_i-$p_u{\cdot}q_i$	r_{ui}-b_i-c_i-$p_u{\cdot}q_i$	r_{ui}-μ-c_i-$p_u{\cdot}q_i$

TABLE V. TRAINING RULES OF INVOLVED PARAMETERS CORRESPONDING TO DIFFERENT COMBINATIONS OF LINEAR BIASES

For S.A (S.1~S.8)*	$c_u = c_u + \eta(err_{ui} - \lambda c_u)$ ** $c_i = c_i + \eta(err_{ui} - \lambda c_i)$ $p_u = p_u + \eta(q_i \cdot err_{ui} - \lambda p_u)$ $q_i = q_i + \eta(p_u \cdot err_{ui} - \lambda q_i)$
For S.B (S.1~S.8)	$c_u = 0$ $c_i = 0$ $p_u = p_u + \eta(q_i \cdot err_{ui} - \lambda p_u)$ $q_i = q_i + \eta(p_u \cdot err_{ui} - \lambda q_i)$
For S.C (S.1~S.8)	$c_u = c_u + \eta(err_{ui} - \lambda c_u)$ $c_i = 0$ $p_u = p_u + \eta(q_i \cdot err_{ui} - \lambda p_u)$ $q_i = q_i + \eta(p_u \cdot err_{ui} - \lambda q_i)$
For S.D (S.1~S.8)	$c_u = 0$ $c_i = c_i + \eta(err_{ui} - \lambda c_i)$ $p_u = p_u + \eta(q_i \cdot err_{ui} - \lambda p_u)$ $q_i = q_i + \eta(p_u \cdot err_{ui} - \lambda q_i)$

*Note that the Expressions of err_{ui} are given in Table 4, and different combinations of PLBs will affect the expressions of err_{ui} but not the parameter update rules.

**The positive constant η is the learning rate.

III. EXPERIMENTAL RESULTS AND ANALYSIS

Evaluation Protocol. For industrial applications, one major motivation to factorize a HiDS matrix is to estimate its missing entries. Owing to its popularity and usefulness, we adopt it as the evaluation protocol to validate involved models' performance. For a tested model, its prediction accuracy for missing entries in a HiDS matrix can be measured by root mean squared error (RMSE) [41] and mean absolute error (MAE):

$$RMSE = \sqrt{\sum_{u,i \in T} \left(r_{ui} - \hat{r}_{ui} \right) \Big/ |T|}, \tag{12}$$

$$MAE = \left(\sum_{u,i \in T} \left| r_{ui} - \hat{r}_{ui} \right|_{abs} \right) \Big/ |T|; \tag{13}$$

where T denotes the testing set and $|\cdot|_{abs}$ denotes the absolute value of a given number. For a given recommender, low RMSE and MAE means high prediction accuracy.

Dataset Description. The empirical validations were conducted on MovieLens 20M dataset. This dataset is collected by the GroupLens Research Project at the University of Minnesota through the MovieLens web site. It is a well-known and widely used non-negative dating HiDS matrix in the community of recommender systems. All the known dating entries are in the range of [0, 5]. TABLE VI shows the dataset details and D.1 represents MovieLens 20M.

TABLE VI. DATASET DETAILS

Dataset	Num.user	Num.item	Num.rating
D.1	13,849,3	26,744	20,000,263

Model Setting. In this part of experiments, we aim to compare the performance of different biases combination. The hyper parameters are set as η=0.01 and λ=0.01 [42]. To minimize the impact caused by an initial guess, the latent factor of each model is initialized with the same randomly-generated arrays. Meanwhile, to obtain objective results, five-fold cross validation is employed on dataset.

Results. TABLE VII records the RMSE and MAE of different linear biases combination on D.1 when the LF dimension is set as f=5, f=10, f=20, and f=40, respectively. TABLE VIII records the total consumed time. Fig. 1 depicts typical training process when f=10 and f=20. From these results, we have the following findings:

a) From TABLE VII, we can find that when f=5 and f=10, S.2 has the better performance than others. But it is different when f=20 and f=40. S.5 can achieve the highest prediction accuracy. The lowest RMSE is 0.8018, 0.7872, 0.7771, 0.7725 when the LF dimension set to f=5, f=10, f=20, f=40 respectively. S.2 and S.5 achieve the lowest RMSE and S.2 is better in the vast majority of cases. Moreover, we see that the RMSE reduces as f increases. Together all the results on the dataset, S.2, S.5, S.8 achieve better prediction accuracy for missing data than other combination. An obvious factor for this phenomenon is the incorporation of the global average μ into the PLBs of an LF model. Note that S.2, S.5 and S.8 all contain μ, and it brings stable performance gain into the LF model.

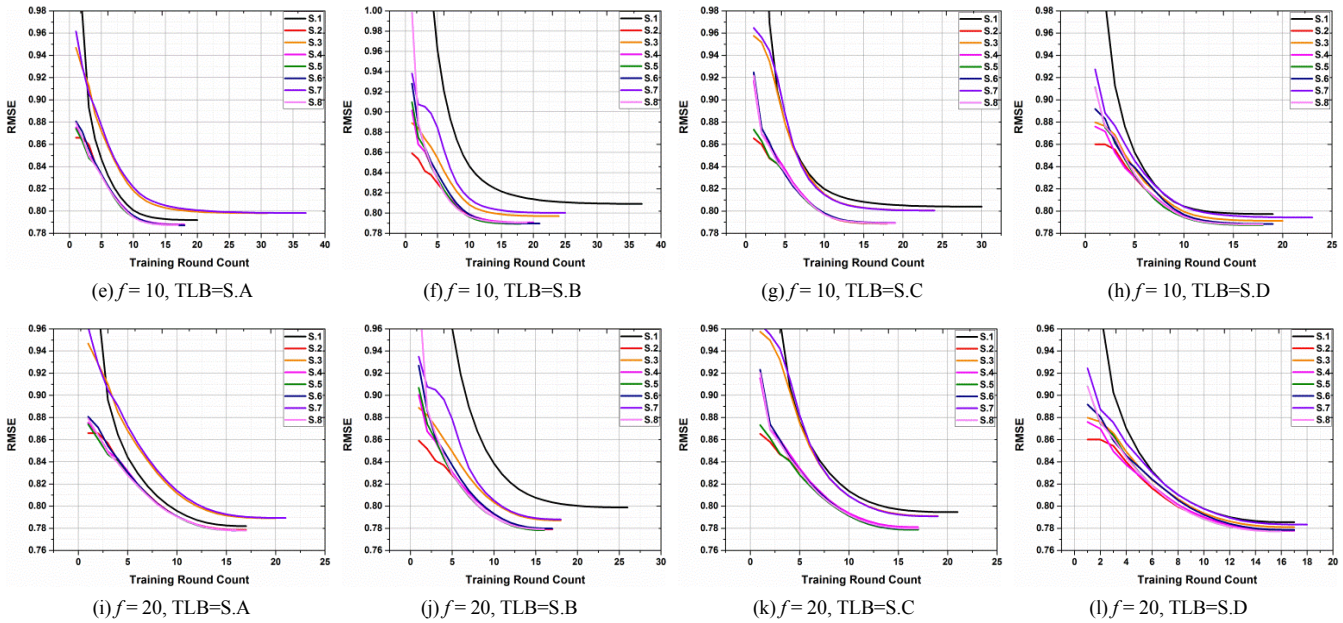
Meanwhile, we observe that S.3 and S.7 result in the least accurate LF models. S.4 and S.6 have almost the same performance as S.2, S5, and S.8. If we only consider user bias b_u in the model like S.6, an LF model can achieve the same performance as that with the global average μ. Some PLBs like the b_u and b_i do have obviously positive/negative effect as f varies.

TABLE VII. RMSE AND MAE OF DIFFERENT COMBINATIONS

D.1		RMSE								MAE							
		S.1	*S.2*	*S.3*	*S.4*	*S.5*	*S.6*	*S.7*	*S.8*	*S.1*	*S.2*	*S.3*	*S.4*	*S.5*	*S.6*	*S.7*	*S.8*
f=5	***S.A***	0.8075	**0.8018**	0.8096	0.8036	0.8029	0.8019	0.8095	0.8019	0.6178	**0.6123**	0.6165	0.6129	0.6120	0.6124	0.6164	0.6129
	S.B	0.8234	**0.8067**	0.8142	0.8070	0.8069	0.8070	0.8217	0.8074	0.6300	**0.6129**	0.6211	0.6150	0.6162	0.6156	0.6254	0.6161
	S.C	0.8178	**0.8040**	0.8181	0.8048	0.8041	0.8043	0.8175	0.8051	0.6241	**0.6128**	0.6242	0.6136	0.6131	0.6134	0.6221	0.6137
	S.D	0.8173	**0.8034**	0.8081	0.8056	0.8052	0.8053	0.8101	0.8053	0.6243	**0.6127**	0.6158	0.6145	0.6149	0.6157	0.6175	0.6154
f=10	***S.A***	0.7918	**0.7872**	0.7981	0.7875	0.7873	0.7878	0.7984	0.7880	0.6051	**0.5996**	0.6059	0.6003	0.5997	0.5997	0.6061	0.6001
	S.B	0.8077	**0.7896**	0.7968	0.7898	0.7903	0.7898	0.7990	0.7897	0.6176	**0.6002**	0.6067	0.6009	0.6013	0.6015	0.6075	0.6015
	S.C	0.8040	**0.7889**	0.8004	0.7893	0.7887	0.7906	0.8006	0.7889	0.6139	**0.6004**	0.6080	0.6014	0.6006	0.6009	0.6077	0.6010
	S.D	0.7992	**0.7880**	0.7910	0.7882	0.7885	0.7893	0.7934	0.7873	0.6091	**0.5993**	0.6015	0.6001	0.6010	0.6006	0.6038	0.6003
f=20	***S.A***	0.7822	0.7783	0.7891	0.7785	**0.7780**	0.7781	0.7894	0.7781	0.5958	0.5919	0.5983	0.5923	**0.5917**	0.5920	0.5987	0.5922
	S.B	0.7990	0.7795	0.7870	0.7798	**0.7783**	0.7798	0.7876	0.7794	0.6092	0.5917	0.5969	0.5919	**0.5916**	0.5929	0.5974	0.5927
	S.C	0.7938	0.7789	0.7911	0.7806	**0.7785**	0.7796	0.7908	0.7799	0.6055	0.5920	0.5994	0.5932	**0.5919**	0.5931	0.5992	0.5932
	S.D	0.7855	0.7779	0.7809	0.7778	**0.7771**	0.7784	0.7834	0.7773	0.5980	0.5914	0.5934	0.5915	**0.5911**	0.5921	0.5946	0.5915
f=40	***S.A***	0.7767	0.7730	0.7834	0.7734	**0.7727**	0.7729	0.7840	0.7728	0.5909	0.5874	0.5939	0.5878	**0.5873**	0.5876	0.5943	0.5879
	S.B	0.7961	0.7738	0.7811	0.7745	**0.7725**	0.7746	0.7821	0.7737	0.6058	0.5865	0.5918	0.5873	**0.5865**	0.5882	0.5923	0.5880
	S.C	0.7890	0.7735	0.7859	0.7753	**0.7733**	0.7745	0.7857	0.7747	0.6016	0.5873	0.5947	0.5892	**0.5871**	0.5885	0.5945	0.5890
	S.D	0.7890	0.7735	0.7859	0.7753	**0.7733**	0.7745	0.7857	0.7747	0.5929	0.5868	0.5882	0.5863	**0.5867**	0.5874	0.5902	0.5867

TABLE VIII. CONSUMED TIME OF DIFFERENT COMBINATIONS

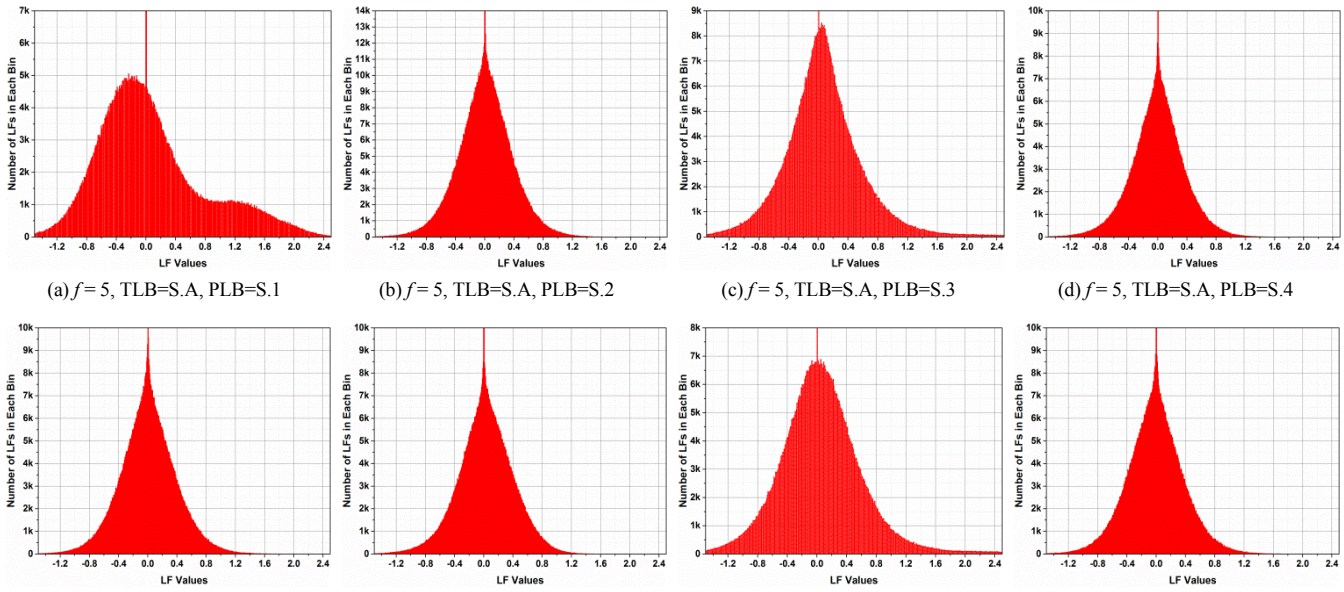
D.1		Total Consumed Time (Seconds)							
		S.1	*S.2*	*S.3*	*S.4*	*S.5*	*S.6*	*S.7*	*S.8*
f=5	***S.A***	65.43	**35.00**	193.91	37.33	36.02	35.61	190.57	36.82
	S.B	82.81	**29.22**	60.60	39.86	37.35	48.04	48.74	39.26
	S.C	704.01	33.47	57.02	41.70	**30.84**	37.74	52.91	40.78
	S.D	34.23	**28.58**	36.26	32.41	34.32	38.69	58.11	36.06
f=10	***S.A***	34.23	**28.58**	36.26	32.41	34.32	38.69	58.11	36.06
	S.B	55.59	33.46	37.80	33.54	**31.16**	34.59	38.73	34.39
	S.C	52.58	34.86	43.94	37.43	**34.37**	36.64	41.57	35.69
	S.D	36.43	**32.29**	37.53	35.69	36.20	37.86	42.05	36.09
f=20	***S.A***	43.74	44.07	48.74	44.16	43.83	44.12	48.98	**43.57**
	S.B	55.67	**42.26**	46.97	43.76	43.58	45.24	46.79	43.70
	S.C	51.45	42.48	47.56	44.46	**42.23**	42.88	45.74	44.24
	S.D	**43.10**	44.88	45.75	43.61	43.51	43.72	45.91	44.01
f=40	***S.A***	**62.13**	62.28	66.13	63.04	63.79	63.06	65.30	63.10
	S.B	70.12	**63.02**	65.92	64.06	64.71	64.01	66.07	63.82
	S.C	67.75	**61.14**	65.21	62.37	63.45	62.80	67.69	70.38
	S.D	74.56	76.24	78.91	76.28	69.17	66.06	66.81	**66.66**



(e) f = 10, TLB=S.A (f) f = 10, TLB=S.B (g) f = 10, TLB=S.C (h) f = 10, TLB=S.D

(i) f = 20, TLB=S.A (j) f = 20, TLB=S.B (k) f = 20, TLB=S.C (l) f = 20, TLB=S.D

Fig. 1. Training process in RMSE



(a) $f = 5$, TLB=S.A, PLB=S.1 (b) $f = 5$, TLB=S.A, PLB=S.2 (c) $f = 5$, TLB=S.A, PLB=S.3 (d) $f = 5$, TLB=S.A, PLB=S.4

(e) $f = 5$, TLB=S.A, PLB=S.5 (f) $f = 5$, TLB=S.A, PLB=S.6 (g) $f = 5$, TLB=S.A, PLB=S.7 (h) $f = 5$, TLB=S.A, PLB=S.8

Fig. 2. Distributions of LFs in different biases

b) Different combinations of linear biases also affect the computational efficiency of an LF model. First, we clearly see that different linear bias combinations lead to different training curves of an LF model from Fig. 1. In general, PLBs mainly affect the initial training error of an LF model, and TLBs can accelerate the training process, as depicted in Fig. 1. However, we do not observe significant connections between the linear bias combinations and the time cost in an LF model, as depicted in TABLE VIII.

c) When considering TLBs, we find that S.A has the highest prediction accuracy but not the lowest time cost. For example, when f=20 in TABLE VII and VIII, the RMSE of S.1 is 0.7822, 0.7990, 0.7938 and 07855. The best is S.A and the worst is S.B. This proves that an LF model with TLBs can better present the given HiDS matrix than those without.

d) In terms of LF distributions, we find that they mainly depend on the PLBs but not the TLBs. Fig. 2 depicts the LF distributions of an LF model with PLB combinations S.1-S.8, TLB combination S.A and f=5. However, similar situations can also be found with TLB combinations S.B-S.C and f=10, 20, and 40. From Fig. 2, we see that with the incorporation of PLBs, the LF distributions tend to concentrate more obviously around a central number. For instance, without any PLBs, the LF distribution is obviously asymmetric as shown in Fig. 2(a). Nonetheless, the symmetry of LF distributions is obvious after the integration of PLBs into an LF model, as shown in Fig. 2(b)-(h).

e) **Summery.** Based on detailed empirical studies, we conclude that: a) the global average μ in PLBs brings stable performance gain into an LF model, while user PLB b_u and item PLB b_i have data-dependently positive/negative effect; b) to incorporate TLBs into an LF model is helpful in accelerate the convergence rate of an LF model; c) the time cost of an LF model is not closely connected with its linear bias scheme; d) certain PLB and TLB combinations, i.e., S.2 and S.5 result in significant advantage in prediction accuracy than the other linear bias schemes do in an LF model; and e) PLBs enable more symmetric and concentrative LF distributions in an LF model.

IV. Conclusions

In this work, we aim at validating the effects of linear biases, including the prior linear biases (PLBs) and training linear biases (TLBs) [34, 35], in the performance of a latent factor (LF) model. We first analyze the integration strategy of PLBs and TLBs into an LF model, along with the possible combinations of PLBs and TLBs. After that, we derive the different parameter update rules corresponding to different PLB and TLB combinations in an LF model relying on the loss function and stochastic gradient descent as the training scheme. Afterwards, detailed empirical studies are conducted on a high-dimensional and sparse matrix generated by industrial applications currently in use, and we do identify different effects by different combinations of linear biases. By doing so, we aim at providing empirical evidence for researchers and engineers who aim at implementing a biased LF model. This work did not consider the non-negativity constraints in an LF model. However, non-negativity is a vital characteristic most industrial data [30, 43, 44]. Hence, it is necessary to identify the effects of linear biases in non-negative LF models. This issue will be in our future plan.

References

[1] Y.-J. Liu, W. Wang, S.-C. Tong, and Y.-S. Liu, "Robust Adaptive Tracking Control for Nonlinear Systems Based on Bounds of Fuzzy Approximation Parameters," *IEEE Trans. on Systems, Man, and Cybernetics Part A: Systems and Humans*, vol. 40, no. 1, pp. 170-184, Jan. 2010.

[2] Y.-J. Liu, and S.-C. Tong, "Optimal Control-Based Adaptive NN Design for a Class of Nonlinear Discrete-Time Block-Triangular Systems," *IEEE Trans. on Cybernetics*, vol. 3, no. 9, pp. 1-11, Nov. 2016.

[3] Y.-Y. Xia, M.-C. Zhou, X. Luo, S.-C. Pang, and Q.-S. Zhu, "A Stochastic Approach to Analysis of Energy-Aware DVS-Enabled Cloud Datacenters," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 1, pp. 1-1, Jan. 2014.

[4] Y.-J. Liu, J. Li, S.-C. Tong, and C.-L. Chen, "Neural Network Control-Based Adaptive Learning Design for Nonlinear Systems With Full-State Constraints," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 27, no. 7, pp. 1-10, Mar. 2016.

[5] Y.-J. Liu, Y. Gao, S.-C. Tong, and Y.-M. Li, "Fuzzy Approximation-Based Adaptive Backstepping Optimal Control for a Class of Nonlinear Discrete-Time Systems With Dead-Zone," *IEEE Trans. on Fuzzy Systems*, vol. 24, no. 1, pp. 16-28, Feb. 2016.

[6] C.-J. Jiang, H.-C. Sun, Z.-J. Ding, P.-W. Wang, and M.-C. Zhou, "An Indexing Network: Model and Applications," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 12, pp. 1633-1648, Dec. 2014.

[7] Y.-J. Liu, S.-C. Tong, C.-L. Chen, and D.-J. Li, "Neural Controller Design-Based Adaptive Control for Nonlinear MIMO Systems With Unknown Hysteresis Inputs," *IEEE Trans. on Cybernetics*, vol. 46, no. 1, pp. 1, Jan. 2015.

[8] M. Balabanovic, "Content-Based, Collaborative Recommendation," *Communication of the ACM*, vol. 40, no. 3, pp. 66-72, 1997.

[9] H.-F. Liu, X.-J. Kong, X.-M. Bai, and W. Wang, "Context-Based Collaborative Filtering for Citation Recommendation," *IEEE Access*, vol. 3, pp. 1-1, Sep. 2015.

[10] J. Xuan, X. Luo, G. Zhang, and J. Lu, "Uncertainty Analysis for the Keyword System of Web Events," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol.46, pp. 1-1, Jun. 2015.

[11] D. Rafailidis, and A. Nanopoulos, "Modeling Users Preference Dynamics and Side Information in Recommender Systems," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol.46, pp. 1-1, Jun. 2016.

[12] G. Adomavicius, and A. Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," *IEEE Trans. on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734-749, Jun. 2005.

[13] B. Prasad, "HYREC: A Hybrid Recommendation System for E-Commerce," *in Proc. of the 6th Int. Conf. Case-Based Reasoning*, Chicago, USA, pp. 23-26, Aug. 2005.

[14] I. Im, and B. H. Kim, "Personalizing the settings for CF-based recommender systems," *in Proc. of the 4th ACM Conf. on Recommender Systems*, Barcelona, Spain, pp. 245-248, Sep. 2010.

[15] K. Junzo, A. Tomofumi, S. Shinji, and M. Hideo, "A Community-Based Recommendation System to Reveal Unexpected Interests," *in Proc. of the 11th Int. Multimedia Modelling Conf.*, Washington, DC, USA, pp. 433-438, Jan. 2005.

[16] X. Luo, Y. Ou, and Z. Xiong, "Improving Neighborhood Based Collaborative Filtering Via Integrated Folksonomy Information," *Pattern Recognition Letters*, vol. 33, no. 3, pp. 263-270, Feb. 2012.

[17] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," *in Proc. of the 10th int. conf. on World Wide Web*, Hong Kong, pp. 285-295, May. 2001.

[18] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," *IEEE Computer Society*, vol. 42, no. 8, pp. 30-37, Aug. 2009.

[19] T. Hofmann, "Latent semantic models for collaborative filtering," *ACM Trans. on Information Systems*, vol. 22, no. 1, pp. 89-115, Jan. 2004.

[20] X. Luo, Y. Ou, and Z. Xiong, "Improving Latent Factor Model Based Collaborative Filtering Via Integrated Folksonomy Factors," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 19, no. 2, pp. 307-327, Apr.2011.

[21] N. Srebro, J. D. M. Rennie, and T. Jaakkola, "Maximum-Margin Matrix Factorization," *Advances in Neural Information Processing Systems*, vol. 37, no. 2, pp. 1329--1336, Nov. 2004.

[22] G. Takács, I. Pilászy, B. Németh, and D. Tikk, "Scalable Collaborative Filtering Approaches for Large Recommender Systems," *Journal of Machine Learning Research*, vol. 10, no. 10, pp. 623-656, Dec. 2009.

[23] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," *in Proc. of the 14th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Las Vegas, Nevada, USA, pp. 24-27, Aug. 2008.

[24] X. Luo, M. Zhou, M. Shang, S. Li, and Y. Xia, "A Novel Approach to Extracting Non-Negative Latent Factors From Non-Negative Big Sparse Matrices," *IEEE Access*, vol. 4, pp. 1-1, Jan. 2016.

[25] K. Fukunaga, and L. Hostetler, "k-nearest-neighbor Bayes-risk estimation," *IEEE Trans. on Information Theory*, vol. 21, no. 3, pp. 285-293, May. 1975.

[26] X. Luo, Y. Xia, Q. Zhu, and Y. Li, "Boosting the K-nearest-neighborhood Based Incremental Collaborative Filtering," *Knowledge-Based Systems*, vol. 53, pp. 90-99, Aug. 2013.

[27] X. Luo, M. Zhou, S. Li, and Z. You, "A Nonnegative 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, Mar. 2016.

[28] X. Luo, M. Zhou, S. Li, Y. Xia, Z. You, Q. Zhu, and H. Leung, "An Efficient Second-Order Approach to Factorize Sparse Matrices in Recommender Systems," *IEEE Trans. on Industrial Informatics*, vol. 11, no. 4, pp. 1-1, Aug. 2015.

[29] M. Rossetti, F. Stella, and M. Zanker, "Towards Explaining Latent Factors with Topic Models in Collaborative Recommender Systems," *in Proc. of the 24th Int. Workshop Conf. on Database and Expert Systems Applications*, pp. 26-30, Sep. 2013.

[30] D. P. Bertsekas, Nonlinear Programming, Belmont, Massachusetts: *Athena Scientific*, 1999.

[31] S. Boyd, and L. Vandenberghe, Convex Optimization, Cambridge: *Cambridge University Press*, 2009.

[32] S. Li, M. Zhou, X. Luo, and Z. You, "Distributed Winner-take-all in Dynamic Networks," *IEEE Trans. on Automatic Control*, vol. 62, no. 2, pp. 577-589, Jun. 2016.

[33] X. Luo, Z. You, S. Li, Y. Xia, and H. Leung, "Improving Network Topology-based Protein Interactome Mapping via Collaborative Filtering," *Knowledge-Based Systems*, vol. 90, pp. 23-32, Dec. 2015.

[34] T. Gábor, P. István, N. Bottyán, and T. Domonkos, "Investigation of Various Matrix Factorization Methods for Large Recommender Systems," *in Proc. of the 8th IEEE Int. Conf. on Data Mining Workshops*, Las Vegas, Nevada, pp. 553-562, Dec. 2008.

[35] X. Luo, H. Liu, G. Gou, Y. Xia, and Q. Zhu, "A Parallel Matrix Factorization Based Recommender by Alternating Stochastic Gradient Decent," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 7, pp. 1403-1414, Oct. 2012.

[36] X. Luo, Y. Xia, and Q. Zhu, "Incremental Collaborative Filtering recommender based on Regularized Matrix Factorization," *Knowledge-Based Systems*, vol. 27, no. 3, pp. 271-280, Mar. 2011.

[37] A. Mnih, and R. Salakhutdinov, "Probabilistic matrix factorization," *in Proc. of the 22nd Int. Conf. on Machine Learning*, pp. 880-887, 2012.

[38] X. Luo, M. Zhou, Y. Xia, Q. Zhu, A. C. Ammari, and A. Alabdulwahab, "Generating Highly Accurate Predictions for Missing QoS Data via Aggregating Nonnegative Latent Factor Models," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 27, no. 3, pp. 524-537, Apr. 2015.

[39] X. Luo, M. Zhou, H. Leung, and Y. Xia, "An Incremental-and-Static-Combined Scheme for Matrix-Factorization-Based Collaborative Filtering," *IEEE Trans. on Automation Science and Engineering*, vol. 13, no. 1, pp. 1-11, Aug. 2014.

[40] X. Luo, M. Zhou, Y. Xia, and Q. Zhu, "An Efficient Non-Negative Matrix-Factorization-Based Approach to Collaborative Filtering for Recommender Systems," *IEEE Trans. on Industrial Informatics*, vol. 10, no. 2, pp. 1273-1284, May. 2014.

[41] J. L. Herlocker, "Evaluating collaborative filtering recommender systems," *ACM Trans. Information Systems*, vol. 22, no. 1, pp. 5-53, Jan. 2004.

[42] X. Luo, Y. Xia, and Q. Zhu, "Applying the Learning Rate Adaptation to the Matrix Factorization Based Collaborative Filtering,". *Knowledge-Based Systems*, vol. 37, pp. 154-164, Aug. 2013.

[43] T. Wang, W. Zhang, C. Ye, and J. Wei, "FD4C: Automatic Fault Diagnosis Framework for Web Applications in Cloud Computing," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 1, pp. 61-75, Jan. 2016.

[44] Y. Xun, J. Zhang, and X. Qin, "FiDoop: Parallel Mining of Frequent Itemsets Using MapReduce," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 3, pp. 313-325, Mar. 2016.