A Nonnegative Latent Factor Model for Large-Scale Sparse Matrices in Recommender Systems via Alternating Direction Method

Xin Luo, *Member, IEEE*, MengChu Zhou, *Fellow, IEEE*, Shuai Li, *Member, IEEE*, Zhuhong You, *Member, IEEE*, Yunni Xia, *Member, IEEE*, and Qingsheng Zhu

***Abstract*— Nonnegative matrix factorization (NMF)-based models possess fine representativeness of a target matrix, which is critically important in collaborative filtering (CF)-based recommender systems. However, current NMF-based CF recommenders suffer from the problem of high computational and storage complexity, as well as slow convergence rate, which prevents them from industrial usage in context of big data. To address these issues, this paper proposes an alternating direction method (ADM)-based nonnegative latent factor (ANLF) model. The main idea is to implement the ADM-based optimization with regard to each single feature, to obtain high convergence rate as well as low complexity. Both computational and storage costs of ANLF are linear with the size of given data in the target matrix, which ensures high efficiency when dealing with extremely sparse matrices usually seen in CF problems. As demonstrated by the experiments on large, real data sets, ANLF also ensures fast convergence and high prediction accuracy, as well as the maintenance of nonnegativity constraints. Moreover, it is simple and easy to implement for real applications of learning systems.**

***Index Terms*— Alternating direction method, big data, collaborative filtering, recommender system, sparse matrices.**

I. Introduction

By EFFICIENTLY analyzing various historical data, recommender systems connect people with their potential favorites. Such systems can prevent people from being inundated by piles of information—where only tiny pieces are truly interesting. Since the early 1990s, research on recommender systems has shown high value in assisting people to locate preferences in the ocean of information according to their historical usage [1]–[5].

In general, a recommender system is a learning system consisting of three fundamental kinds of entities, i.e., users, items (e.g., commodities), and user-item usage data (e.g., scores). Such a system aims at learning useful patterns describing the hidden connections among users and items from the usage data, and tries to link users with items according to these patterns. To deal with such tasks, researchers proposed many models during the past 20 years [1], [2]. They can be further grouped into several categories, among which a very important one is collaborative filtering (CF) [1]–[8].

For a CF-based recommender system, each entry in a rating matrix models the usage history of an item by the user in correspondence. It is usually assigned with a high value to denote strong user-item preferences. Since only a finite item set can be touched by each user, this matrix is usually very sparse with a mass of missing values. On the other hand, if these missing values are estimated appropriately, it is feasible to link people with their potential favorites. Therefore, the main task of a CF-based recommender is to implement efficient missing-data-estimation, i.e., to estimate unknown ratings based on known ones subject to globally high accuracy and other requirements [1]–[8].

According to the up-to-date progress on CF-based recommender systems, matrix factorization (MF)-based models are highly accurate and scalable under many circumstances [2], [9]–[11]. Their principle is to build a low-rank estimate to the original rating-matrix based on its known entries. Such a model works by mapping both items and users into the same latent feature space, training desired user/item features on the existing ratings, and then predicting

Manuscript received June 18, 2014; revised October 9, 2014, January 2, 2015, and March 9, 2015; accepted March 16, 2015. This work was supported in part by the Hong Kong, Macao and Taiwan Science and Technology Cooperation Program of China under Grant No. 2013DFM10100, in part by the NSFC under Grants 61202347, 61272194, 61472051, 61401385, and 61373086, in part by the U.S. NSF under Grant CMMI-1162482, in part by the Young Scientist Foundation of Chongqing under Grant cstc2014kjrc-qnrc40005 and Grant cstc2013kjrc-qnrc0079, in part by the Post-Doctoral Science Funded Project of Chongqing under Grant Xm2014043, in part by the China Post-Doctoral Science Foundation under Grant 2014M562284, in part by the Fundamental Research Funds for the Central Universities under Grant 106112015CDJXY180005 and Grant 106112014CDJZR185503, and in part by the Specialized Research Fund for the Doctoral Program of Higher Education under Grant 20120191120030. *(Corresponding Author: Xin Luo.)*

X. Luo is with the Key Laboratory of Dependable Service Computing in Cyber Physical Society (Chongqing University), Ministry of Education, Chongqing 400044, China, and also with the Chongqing Key Laboratory of Software Theory and Technology, Chongqing University, Chongqing 400044, China (e-mail: luoxin21@cqu.edu.cn).

M. Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA, and the Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai 201804, China (e-mail: zhou@njit.edu).

S. Li and Z. You are with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong (e-mail: zhuhong.you@polyu.edu.hk; shuaili@polyu.edu.hk).

Y. Xia and Q. Zhu are with the College of Computer Science, Chongqing Key Laboratory of Software Theory and Technology, Chongqing University, Chongqing 400044, China (e-mail: xiayunni@hotmail.com; qszhu@cqu.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2015.2415257

unknown ratings heavily relying on the inner products of related user-item feature-vector pairs. Compared with other CF models, they take the following advantages.

1) The dimension of the latent feature space can be set low without impairing accuracy, thereby leading to small feature matrices [2], [9]–[16]. Hence, they possess fine scalability.
2) With a carefully designed loss function, their accuracy is very competitive [9]–[11], [13]–[16].

Sarwar *et al.* [12] proposed singular value decomposition (SVD)-based recommender. However, it requires prefilling unknown entries in the given matrix artificially for carrying out a standard SVD process, thereby leading to relatively low prediction accuracy and high complexity. The Netflix-prize competition has motivated several sophisticated and highly efficient MF-based models, e.g., the biased regularized incremental simultaneous MF model [13], SVD++ model [14], probabilistic MF model [15], and nonparametric MF model [16]. In recent years, further research yielded more sophisticated models, such as the multidimensional probabilistic model [17], weighted trace-norm regularization-based model [18], collaborative Gaussian process-based preference model [19], nonparametric Bayesian-based latent factor model [20], and probabilistic MF model with nonrandom missing data [21]. Their idea is also used to address other related issues, e.g., content-based image tagging [22] and recommendation [23], QoS prediction [24], [25], video reindexing [26], and mobile-user tracking [27].

The aforementioned MF-based CF models, in spite of their efficiency, do not fulfill nonnegativity constraints, i.e., the obtained features of users/items might be negative. Nonnegativity can enable the learnt features to represent the actual meanings, such as the user interests and community tendency more precisely [28]–[32], and make the obtained model more interpretable with fine representativeness of the target rating-matrix. Therefore, it is desired to design MF-based CF models subject to them.

With a complete target matrix, several approaches can be employed to conduct a nonnegative factorization process. Paatero and Tapper [33] applied alternating least squares to implementing the MF with restricting negative features at zero to maintain the nonnegativity. Lee and Seung [34] derived the nonnegative multiplicative updates for the desired feature matrices, which can maintain the nonnegativity of initially nonnegative features. Lin [35] used projected gradient decent to implement nonnegative MF (NMF). These methods and their extensions [33]–[38] can well factorize a complete matrix under nonnegativity constraints, but are not applicable for CF-based recommenders: a complete rating-matrix is never available, and prefilling the unknown entries artificially will inevitably impair the prediction accuracy [12], [28]–[32].

Different from problems in computer vision, our CF problems possess the nature of sparsity, and missing entries in a target rating-matrix are much more than given ones. Hence, the above mentioned NMF models should be modified fit for such condition. Zhang *et al.* [28] suggested integrating a binary weight matrix to represent the missing values. Xu *et al.* [39] proposed nonnegative matrix completion (NMC), where the principle is to introduce an intermediate, complete matrix to estimate a target rating-matrix, and take an NMF process on this matrix rather than on the incomplete rating-matrix to avoid addressing its missing entries. Although these models can deal with the incomplete one in a CF problem, they share the drawback of high computational and storage costs, which are linear with the size of the target matrix. Note that in industrial applications, the target rating-matrix can be extremely sparse and of high dimensions. For instance, the MovieLens 10M rating-matrix [40] has 69 878 rows and 10 677 columns. When processing such a huge matrix, the complexity linear with its size is hard to handle in practice.

Luo *et al.* [32] proposed the regularized single-element-based NMF (RSNMF) model. It builds a low-rank approximation to the target matrix under the nonnegativity constraint based on known entries only, thereby achieving the low computational and storage costs that are linearly related to the size of the known entry set in the target matrix. However, it depends on the multiplicative training process under nonnegativity constraints [32]–[34], and suffers slow convergence.

This work proposes an alternating direction method (ADM)-based approach to building a nonnegative latent factor (NLF) model for recommender systems. We use the term NLF to differ from NMF because we train the desired features through the single-element-dependent optimization instead of direct matrix manipulations, thereby drastically reducing complexity in computation and storage. The idea is to employ the principle of the ADM-based optimization, which decomposes the original optimization task into tiny subtasks, and then trains the involved parameters alternatively, to obtain high convergence rate. The objective is to implement a highly efficient nonnegative model enjoying which has high prediction accuracy and fast convergence, as well as low computational and storage costs.

Section II gives the preliminaries. Section III states our method. Section IV gives the experiments and discusses the results. Finally, Section V discusses some related issues and concludes this paper.

II. Preliminaries

CF-based recommender systems usually model historical user behaviors into a user-item rating-matrix as the fundamental data source, as defined in [1], [2], and [9]–[11].

Definition 1: Given an item set I and a user set U, a user-item rating-matrix R is a $|U| \times |I|$ matrix where each element $r_{u,i}$ is proportional to user u's preference on item i.

With this matrix, the problem of CF is equivalent to the following the problem of missing-data-estimation [1], [2], [9]–[11].

Definition 2: Let R_K and R_W denote the known and whole entry sets in R, respectively. A CF problem is to construct an estimator $\hat{R}$ based on R_K, to generate the prediction $\hat{r}_{u,i}$ for each entry $(u, i) \in R_W$ and the error $\sum_{(u,i)\in R_W} |r_{u,i} - \hat{r}_{u,i}|$ is minimized.

With an MF-based CF model, the given rating-matrix R is factorized into to two rank-f matrices P and Q, where P is $|U| \times f$, Q is $|I| \times f$, and $f \ll \min(|U|, |I|)$. Note that P and Q are interpreted as the user and item latent feature matrices that reflect the user and item characteristics hidden in the rating data. This factorization process is implemented by minimizing the objective function measuring the difference between R and the rank-f estimate PQ^T, e.g., the Euclidean distance or the Kullback–Leibler divergence [1], [2], [9]–[11], [28]–[32]. With Euclidean distance, such an objective function is given by

$$\arg\min_{P,Q} \varepsilon(P, Q) = \|R - PQ^T\|_F^2 \tag{1}$$

where $\|\cdot\|_F$ denotes the Frobenius norm of a given matrix. Note that we use the symbol $\varepsilon(P,Q)$ to denote the objective function regarding the desired parameters P and Q.

By integrating nonnegativity constraints into the optimization problem (1), we enable the involved parameters, i.e., P and Q, to be nonnegative during the training process. Formally, we have

$$\begin{aligned}&\arg\min_{P,Q} \varepsilon(P, Q) = \|R - PQ^T\|_F^2\\ &\text{s.t. } P, Q \geq 0.\end{aligned} \tag{2}$$

If R is complete, i.e., $R_K = R_W$, then this problem can be solved through several NMF-based approaches [32]–[38]. However, such a precondition can never be satisfied in a CF problem. Therefore, NMF models should be tuned to fit the incomplete target matrix. Several efforts have been made to address this issue [28]–[32], [39], e.g., the weighted NMF (WNMF) model [28], and the NMC model [39]. These models can deal with the nonnegative factorization task on incomplete matrices; however, they share the drawback of high complexity in both computation and storage when dealing with huge, sparse matrices commonly met in industrial applications.

To deal with such matrices, RSNMF [32] implements the nonnegative, single-element-dependent multiplicative update to obtain the desired feature matrices. RSNMF needs R_K only, thereby leading to its computational complexity at $\Theta(R_K \times f)$ for each training iteration, and storage at $\Theta(f \times \max\{(|R_K|/f), |U|, |I|\})$. However, due to the self-adaptive nature of multiplicative update under nonnegativity constraints, RSNMF suffers slow convergence and this becomes a barrier to real applications.

This paper aims at implementing an NLF model with fast convergence and high prediction accuracy while maintaining low computational and storage complexity. To do so, we introduce the ADM-based optimization framework [41] into a model-building process. Its principle is to decompose an original learning task into tiny subtasks by alternatively optimizing one involved parameter while fixing all the others. All desired parameters are sequentially updated, each of which is trained based on the updated optimal state. Such an optimization process results in a fast convergence, especially for long-tail convergence cases.

III. ADM-Based NLF Model for CF Problems

A. Feature Update Rules

When employing the ADM-based framework, the level-of-decomposition of an original learning task can affect the robustness of the resulting model. In general, it is preferred to decompose an optimization task into subtasks small-enough on the premise of convergence. Therefore, we first expand (2) into a single-feature-dependent form, given by

$$\begin{aligned}&\arg\min_{P,Q} \varepsilon(P, Q) = \sum_{(u,i)\in R_K} \left(r_{u,i} - \sum_{k=1}^{f} p_{u,k} q_{i,k} \right)^2\\ &\text{s.t. } P, Q \geq 0.\end{aligned} \tag{3}$$

Note that in (3), we consider the distance on the known entry set R_K only. It is reasonable since the other entries are missing and the corresponding part of distance information is unavailable. Due to the nonnegative constraints of P and Q, it is complex to conduct the optimization directly. Therefore, we introduce new parameters $X^{|U|\times f}$ and $Y^{|U|\times f}$ for separating the parameter update and nonnegative constraints, thereby expanding the original task (3) into

$$\begin{aligned}&\arg\min_{P,Q,X,Y} \varepsilon(P, Q, X, Y) = \sum_{(u,i)\in R_K} \left(r_{u,i} - \sum_{k=1}^{f} x_{u,k} y_{i,k} \right)^2\\ &\text{s.t. } P = X, P \geq 0\\ &\qquad Q = Y, Q \geq 0.\end{aligned} \tag{4}$$

Following the principle of ADM [41], we derive the augmented Lagrangian function for (4) as follows:

$$\begin{aligned}&L(P, Q, X, Y, \Gamma, \mathrm{K})\\ &= \frac{1}{2} \sum_{(u,i)\in R_K} \left(r_{u,i} - \sum_{k=1}^{f} x_{u,k} y_{i,k} \right)^2 + \Gamma \circ (X - P)\\ &\quad + \mathrm{K} \circ (Y - Q) + \frac{\rho}{2}\|X - P\|_F^2 + \frac{\tau}{2}\|Y - Q\|_F^2\end{aligned} \tag{5}$$

where $\Gamma^{|U|\times f}$ and $K^{|I|\times f}$ are the Lagrangian multiplier matrices corresponding to the constraints $P = X$ and $Q = Y$, respectively; operator $\circ$ denotes the Hadamard operation; $(\rho/2)\|X - P\|_F^2$ and $(\tau/2)\|Y - Q\|_F^2$ are the augmentation terms; and constants ρ and τ are augmenting parameters controlling the effect of the augmentation. Note that the nonnegative constraints, i.e., $P \geq 0$ and $Q \geq 0$ can be easily fulfilled by projecting the updated P and Q onto the nonnegative filed of real numbers.

We choose to expand the Lagrangian function (5) into a single-element-dependent form for fine-grained decomposition as follows:

$$\begin{aligned}&L(P, Q, X, Y, \Gamma, \mathrm{K})\\&= \frac{1}{2}\sum_{(u,i)\in R_K}\left(r_{u,i} - \sum_{k=1}^{f} x_{u,k}y_{i,k}\right)^2 + \sum_{(u,k)}\gamma_{u,k}(x_{u,k} - p_{u,k})\\&\quad + \sum_{(i,k)}\kappa_{i,k}(y_{i,k} - q_{i,k}) + \frac{\rho}{2}\sum_{(u,k)}(x_{u,k} - p_{u,k})^2\\&\quad + \frac{\tau}{2}\sum_{(i,k)}(y_{i,k} - q_{i,k})^2. \qquad (6)\end{aligned}$$

Note that in (6), parameters ρ and τ have crucial effect on the convergence and accuracy of the obtained model. It is usual to have ρ and τ depend on the norm of the target matrix to implement fine augmentation. However, in our problem the target matrix R can be extremely sparse and imbalanced. Therefore, we make a more careful choice of augmenting parameters to let them depend on each user and item involved. More specifically, we transform (6) into

$$\begin{aligned}&L(P, Q, X, Y, \Gamma, \mathrm{K})\\&\quad = \frac{1}{2}\sum_{(u,i)\in R_K}\left(r_{u,i} - \sum_{k=1}^{f} x_{u,k}y_{i,k}\right)^2\\&\qquad + \sum_{(u,k)}\gamma_{u,k}(x_{u,k} - p_{u,k}) + \sum_{(i,k)}\kappa_{i,k}(y_{i,k} - q_{i,k})\\&\qquad + \sum_{(u,k)}\frac{\rho_u}{2}(x_{u,k} - p_{u,k})^2 + \sum_{(i,k)}\frac{\tau_i}{2}(y_{i,k} - q_{i,k})^2 \qquad (7)\end{aligned}$$

where ρ_u and τ_i are given by

$$\begin{cases}\rho_u = \lambda|R_K(u)|\\ \tau_i = \lambda|R_K(i)|\end{cases} \qquad (8)$$

where $R_K(u)$ and $R_K(i)$ denote the known entries related to user u and item i, respectively; constant λ is a hyperparameter controlling the extent of augmentation. The physical meaning of (8) is to have the augmentation of each involved feature depend on the size of the related sample-subset, i.e., features corresponding to more ratings should be learnt more carefully.

Note that it is useful to combine the Lagrangian terms and augmentation ones to obtain a concise form, which can be achieved through a simple transformation of (8). For instance, considering two terms $\gamma_{u,k}(x_{u,k} - p_{u,k})$ and $(\rho_u/2)(x_{u,k} - p_{u,k})^2$, we have

$$\begin{aligned}&\frac{\rho_u}{2}(x_{u,k} - p_{u,k})^2 + \gamma_{u,k}(x_{u,k} - p_{u,k})\\&\quad = \frac{\rho_u}{2}\left((x_{u,k} - p_{u,k})^2 + 2(x_{u,k} - p_{u,k})\frac{\gamma_{u,k}}{\rho_u} + \left(\frac{\gamma_{u,k}}{\rho_u}\right)^2\right)\\&\qquad - \frac{\rho_u}{2}\left(\frac{\gamma_{u,k}}{\rho_u}\right)^2\\&\quad = \frac{\rho_u}{2}\left(x_{u,k} - p_{u,k} + \frac{\gamma_{u,k}}{\rho_u}\right)^2 - \frac{\gamma_{u,k}^2}{2\rho_u}. \qquad (9)\end{aligned}$$

By analogy, we transform (7) into a more concise form:

$$\begin{aligned}L(P, Q, X, Y, \Gamma, \mathrm{K}) &= \frac{1}{2}\sum_{(u,i)\in R_K}\left(r_{u,i} - \sum_{k=1}^{f} x_{u,k}y_{i,k}\right)^2\\&\quad + \sum_{(u,k)}\frac{\rho_u}{2}\left(x_{u,k} - p_{u,k} + \frac{\gamma_{u,k}}{\rho_u}\right)^2\\&\quad + \sum_{(i,k)}\frac{\tau_i}{2}\left(y_{i,k} - q_{i,k} + \frac{\kappa_{i,k}}{\tau_i}\right)^2 - \varphi \qquad (10)\end{aligned}$$

where $\varphi = \sum_{(u,k)}(\gamma_{u,k}^2/2\rho_u) + \sum_{(i,k)}(\kappa_{i,k}^2/2\tau_i)$. Note that (10) has the same objective as (7); it is a transformation of (7) to simplify the inference of update rules of involved parameters, since φ acts only when doing the dual gradient ascent on Γ and K.

In order to decompose the optimization task (10), we consider the optimization process with respect to the single latent feature $x_{u,k}$ by alternatively fixing the other parameters in (10). Hence, the Lagrangian function (10) is convex with $x_{u,k}$ and is analytically solvable. The optimal value of $x_{u,k}$ is derived as follows:

$$\begin{aligned}\frac{\partial L}{\partial x_{u,k}} &= -\sum_{i\in R_K(u)} y_{i,k}\left(r_{u,i} - \sum_{k=1}^{f} x_{u,k}y_{i,k}\right)\\&\quad + \rho_u\left(x_{u,k} - p_{u,k} + \frac{\gamma_{u,k}}{\rho_u}\right)\\&= -\sum_{i\in R_K(u)} y_{i,k}\left(r_{u,i} - \sum_{j=1,j\neq k}^{f} x_{u,j}y_{i,j}\right)\\&\quad - \rho_u p_{u,k} + \gamma_{u,k} + x_{u,k}\left(\sum_{i\in R_K(u)} y_{i,k}^2 + \rho_u\right). \qquad (11)\end{aligned}$$

By making $(\partial L/\partial x_{u,k}) = 0$, we have

$$x_{u,k} = \frac{\sum\limits_{i\in R_K(u)} y_{i,k}\left(r_{u,i} - \sum\limits_{j=1,j\neq k}^{f} x_{u,j}y_{i,j}\right) + \rho_u p_{u,k} - \gamma_{u,k}}{\sum\limits_{i\in R_K(u)} y_{i,k}^2 + \rho_u}. \qquad (12)$$

Note that (12) is obtained by fixing the other parameters to alternatively optimize $x_{u,k}$, to implement a one-pass training of a Gauss–Seidel process [41]. Let t and $t+1$, respectively, denote the initial and updated status of each parameter. We have the following process corresponding to (12):

$$x_{u,k}^{t+1} := \underset{x_{u,k}}{\arg\min}\, L\big(P^t, Q^t, x_{u,k}, (X - x_{u,k})^t, Y^t, \Gamma^t, \mathrm{K}^t\big). \qquad (13)$$

Note that in (13), we slightly abuse the notations using (X-$x_{u,k}$) to denote the parameters in X except for $x_{u,k}$.

With (13), we actually decompose the optimization of X in (10) into |U| $\times$ f subtasks, each of which is to optimize a single parameter $x_{u,k}$. Through a similar induction, we obtain the optimal parameter $y_{i,k}$ as follows:

$$y_{i,k}^{t+1} := \arg\min_{y_{i,k}} L\big(P^t, Q^t, X^t, y_{i,k}, (Y - y_{i,k})^t, \Gamma^t, \mathrm{K}^t\big)$$

$$\Rightarrow y_{i,k} = \frac{\sum_{u\in R_K(i)} x_{u,k}\left(r_{u,i} - \sum_{j=1,j\neq k}^{f} x_{u,j}y_{i,j}\right) + \tau_i q_{k,i} - \kappa_{i,k}}{\sum_{u\in R_K(i)} x_{u,k}^2 + \tau_i}. \tag{14}$$

Note that the optimization of X and Y is bilinear. Therefore, we can further implement the alterative-direction-based optimization into the training process of $x_{u,k}$ and $y_{i,k}$, by making $y_{i,k}$ depend on the latest state of $x_{u,k}$

$$y_{i,k}^{t+1} := \arg\min_{y_{i,k}} L\big(P^t, Q^t, x_{u,k}^{t+1}, (X - x_{u,k})^t, y_{i,k}, (Y - y_{i,k})^t, \Gamma^t, \mathrm{K}^t\big). \tag{15}$$

However, (13) and (15) perplex the optimization process and raise its computational complexity to $\Theta(|R_K| \times f^2)$ for each iteration. To balance the convergence rate and computational cost, we train the features of the kth dimension in X and Y alternatively as follows:

$$\text{for } k = 1 \sim f$$
$$\begin{cases} X_{.,k}^{t+1} := \arg\min_{X_{.,k}} L\big(P^t, Q^t, X_{.,1\sim k-1}^{t+1}, X_{.,k}, X_{.,k+1\sim f}^t, Y_{.,1\sim k-1}^{t+1}, Y_{.,k\sim f}^t, \Gamma^t, \mathrm{K}^t\big) \\ Y_{.,k}^{t+1} := \arg\min_{Y_{.,k}} L\big(P^t, Q^t, X_{.,1\sim k-1}^{t+1}, X_{.,k\sim f}^t, Y_{.,1\sim k-1}^{t+1}, Y_{.,k}, Y_{.,k+1\sim f}^t, \Gamma^t, \mathrm{K}^t\big) \end{cases} \tag{16}$$

where $X_{.,*}$ and $Y_{.,*}$ denote the corresponding columns of X and Y, respectively. The training of $X_{.,k}$ and $Y_{.,k}$ depends on the single-element-dependent update rule (12) and (14). With (16), we simultaneously train each column of X and Y based on the optimized values of the other columns trained before, as shown in Fig. 1.

In terms of the other parameters in (5), i.e., P, Q, Γ, and K, we employ the general process of ADM to sequentially update P and Q

$$\begin{cases} P^{t+1} = \arg\min_{P} L(P, Q^t, X^{t+1}, Y^{t+1}, \Gamma^t, \mathrm{K}^t) \\ Q^{t+1} = \arg\min_{Q} L(P^t, Q, X^{t+1}, Y^{t+1}, \Gamma^t, \mathrm{K}^t) \end{cases} \tag{17}$$

and then take dual gradient ascent on Γ and K

$$\begin{cases} \Gamma^{t+1} = \Gamma^t + \eta\nabla_\Gamma L(P^{t+1}, Q^{t+1}, X^{t+1}, Y^{t+1}, \Gamma, \mathrm{K}^t) \\ \mathrm{K}^{t+1} = \mathrm{K}^t + \eta\nabla_{\mathrm{K}} L(P^{t+1}, Q^{t+1}, X^{t+1}, Y^{t+1}, \Gamma^t, \mathrm{K}). \end{cases} \tag{18}$$

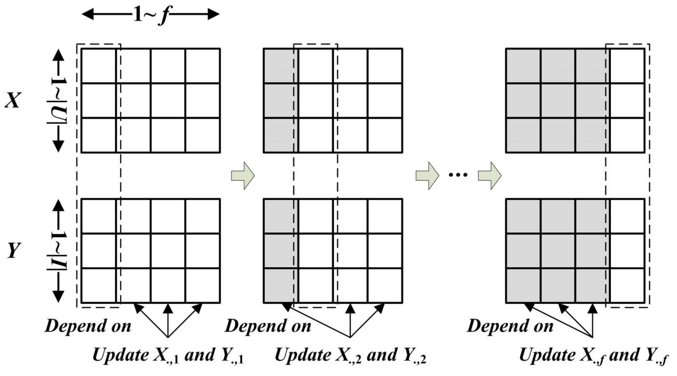

1~f
X
1~|U|
Y
1~|I|
Depend on
Update $X_{.,1}$ and $Y_{.,1}$
Depend on
Update $X_{.,2}$ and $Y_{.,2}$
Depend on
Update $X_{.,f}$ and $Y_{.,f}$

Fig. 1. Alternative optimization process of X and Y in one iteration.

Note that constant η in (18) denotes the step size for the dual gradient ascent. To keep the optimization restricted in R_K, we take the update for P, Q, Γ, and K in a single-element-dependent form, given by

$$\begin{cases} p_{u,k} = \max\left(0, x_{u,k} + \frac{\gamma_{u,k}}{\rho_u}\right) \\ q_{i,k} = \max\left(0, y_{i,k} + \frac{\kappa_{i,k}}{\tau_i}\right) \\ \gamma_{u,k} = \gamma_{u,k} + \eta\rho_u(x_{u,k} - p_{u,k}) \\ \kappa_{i,k} = \kappa_{i,k} + \eta\tau_i(y_{i,k} - q_{i,k}) \end{cases} \tag{19}$$

where the constant η is the learning rate for the dual gradient ascent on Γ and K. Note that in (19) the nonnegativity of P and Q is maintained by projecting $p_{u,k}$ and $q_{i,k}$ onto the nonnegative field of real numbers.

Based on the above induction, we derive the optimization process of the proposed ADM-based NLF (ANLF) model by combining (16)–(18), given by

$$\text{for } k = 1 \sim f$$
$$\begin{cases} X_{.,k}^{t+1} := \arg\min_{X_{.,k}} L\big(P^t, Q^t, X_{.,1\sim k-1}^{t+1}, X_{.,k}, X_{.,k+1\sim f}^t, Y_{.,1\sim k-1}^{t+1}, Y_{.,k\sim f}^t, \Gamma^t, \mathrm{K}^t\big) \\ Y_{.,k}^{t+1} := \arg\min_{Y_{.,k}} L\big(P^t, Q^t, X_{.,1\sim k-1}^{t+1}, X_{.,k\sim f}^t, Y_{.,1\sim k-1}^{t+1}, Y_{.,k}, Y_{.,k+1\sim f}^t, \Gamma^t, \mathrm{K}^t\big) \end{cases} \tag{20}$$
$$\begin{cases} P^{t+1} = \arg\min_{P} L(P, Q^t, X^{t+1}, Y^{t+1}, \Gamma^t, \mathrm{K}^t) \\ Q^{t+1} = \arg\min_{Q} L(P^t, Q, X^{t+1}, Y^{t+1}, \Gamma^t, \mathrm{K}^t) \\ \Gamma^{t+1} = \Gamma^t + \eta\nabla_\Gamma L(P^{t+1}, Q^{t+1}, X^{t+1}, Y^{t+1}, \Gamma, \mathrm{K}^t) \\ \mathrm{K}^{t+1} = \mathrm{K}^t + \eta\nabla_{\mathrm{K}} L(P^{t+1}, Q^{t+1}, X^{t+1}, Y^{t+1}, \Gamma^t, \mathrm{K}). \end{cases}$$

The single-element-dependent update rules of each parameter involved in this optimization process are derived by combining (12), (14), and (19)

$$\rho_u = \lambda|R_K(u)|, \quad \tau_i = \lambda|R_K(i)|$$
$$\begin{cases} x_{u,k} = \dfrac{\sum_{i\in R_K(u)} y_{i,k}\left(r_{u,i} - \sum_{j=1,j\neq k}^{f} x_{u,j}y_{i,j}\right) + \rho_u p_{u,k} - \gamma_{u,k}}{\sum_{i\in R_K(u)} y_{i,k}^2 + \rho_u} \\ y_{i,k} = \dfrac{\sum_{u\in R_K(i)} x_{u,k}\left(r_{u,i} - \sum_{j=1,j\neq k}^{f} x_{u,j}y_{i,j}\right) + \tau_i q_{k,i} - \kappa_{i,k}}{\sum_{u\in R_K(i)} x_{u,k}^2 + \tau_i} \\ p_{u,k} = \max\left(0, x_{u,k} + \dfrac{\gamma_{u,k}}{\rho_u}\right) \\ q_{i,k} = \max\left(0, y_{i,k} + \dfrac{\kappa_{i,k}}{\tau_i}\right) \\ \gamma_{u,k} = \gamma_{u,k} + \eta\rho_u(x_{u,k} - p_{u,k}) \\ \kappa_{i,k} = \kappa_{i,k} + \eta\tau_i(y_{i,k} - q_{i,k}). \end{cases} \tag{21}$$

From (21), we see that the update of each parameter in ANLF is confined into R_K. Therefore, its computational and storage costs are linear with $|R_K|$. In Section III-B, we give the algorithm for ANLF.

B. Algorithm Design and Analysis for ANLF

According to (21), the main difficulty in designing a highly efficient algorithm is to deal with the sum $\Delta(u,i,k) = \sum_{j=1,j\neq k}^{f} x_{u,j}y_{i,j}$ for updating $x_{u,k}$ and $y_{i,k}$, which costs $\Theta(f)$ to resolve, and according to (16), $\Delta(u,i,k)$ varies with u, i or k. Therefore, a straight-forward approach needs to compute $\Delta(u,i,k)$ for $x_{u,k}$ and $y_{i,k}$ corresponding to each $r_{u,i} \in R_K$ during one iteration, and has the computational complexity $\Theta(|R_K| \times f^2)$. Fortunately, this can be avoided by a novel programming trick.

Note that parameter matrices X and Y are employed to approximate each entry in R_K, and each entry $e_{u,i} = \sum_{k=1}^{f} x_{u,k}y_{i,k}$ in $E = XY^T$ is the estimation of the corresponding entry in R_K. Since we have $\Delta(u,i,k) = \sum_{j=1,j\neq k}^{f} x_{u,j}y_{i,j} = e_{u,i} - x_{u,k}y_{i,k}$ (naturally, $x_{u,k}$ and $y_{i,k}$ in this formula are initial values in the current iteration), we simply use an auxiliary array to cache the entry set $\{e_{u,i}|(u,i) \in R_K\}$, and update each $e_{u,i}$ after we update the kth column of X and Y. To initialize this auxiliary array costs $\Theta(|R_K| \times f)$ to compute the estimations. During each training iteration, a single traverse on it is required to update each estimation corresponding to the feature update of each dimension, which costs $\Theta(|R_K| \times f)$ in total. Thus, the computational complexity of the optimization process is linear with $|R_K|$.

Based on this design, we give the pseudocode of Algorithm ANLF. In this algorithm, we employ seven auxiliary arrays for high computational efficiency, which are X_U, X_D, and X_C with dimension $|U|$, Y_U, Y_D, and Y_C with dimension $|I|$, and E_K with dimension $|R_K|$, respectively. Note that for parameters involved in the optimization process, i.e., X, Y, P, Q, Γ, and K, only the dominant parameters X and Y should be initialized with randomly generated positive-real-numbers for performing effective update [and note that it is preferred to restrict their initial values in a particular interval,

Algorithm 1 Algorithm ANLF

```
Input: $R_K$, $U$, $I$, $f$
Operation                                                          Cost
Initialize $X$ positively at random                                $\Theta(|U| \times f)$
Initialize $Y$ positively at random                                $\Theta(|I| \times f)$
Initialize $P$, $\Gamma$                                           $\Theta(|U| \times f)$
Initialize $Q$, $K$                                                $\Theta(|I| \times f)$
Initialize $X\_U$, $X\_D$, $X\_C$, $\rho$                          $\Theta(|U|)$
Initialize $Y\_U$, $Y\_D$, $Y\_C$, $\tau$                          $\Theta(|I|)$
Initialize $\eta$, $\lambda$, $E_K$.                               $\Theta(1)$
Initialize $t = 0$, $n$ = max_training_round                       $\Theta(1)$
for $r_{u,i} \in R_K$                                              $\times|R_K|$
    $\rho_u = \rho_u + \lambda$                                    $\Theta(1)$
    $\tau_i = \tau_i + \lambda$                                    $\Theta(1)$
    $e_{u,i} = \sum_{k=1}^{f} x_{u,k}y_{k,i}$                      $\Theta(f)$
    append $e_{u,i}$ to $E_K$                                      $\Theta(1)$
end for                                                            –
while not converge && $t \leq n$ do                                $\times t$
    for $k = 1$ to $f$                                             $\times f$
        set $X\_U$, $X\_D = 0$                                     $\Theta(|U|)$
        set $Y\_U$, $Y\_D = 0$                                     $\Theta(|I|)$
        for $r_{u,i} \in R_K$                                      $\times|R_K|$
            $X\_U_u = X\_U_u + y_{i,k}\left(r_{u,i} - e_{u,i} + x_{u,k}y_{i,k}\right)$
                                                                   $\Theta(1)$
            $X\_D_u = X\_D_u + y_{i,k}^2$                          $\Theta(1)$
            $Y\_U_i = Y\_U_i + x_{u,k}\left(r_{u,i} - e_{u,i} + x_{u,k}y_{i,k}\right)$
                                                                   $\Theta(1)$
            $Y\_D_i = Y\_D_i + x_{u,k}^2$                          $\Theta(1)$
        end for                                                    –
        for $u \in U$                                              $\times|U|$
            $x_{u,k} = \left(X\_U_u + \rho_u p_{u,k} - \gamma_{u,k}\right)/(X\_D_u + \rho_u)$
                                                                   $\Theta(1)$
            $p_{u,k} = \max\left(0, x_{u,k} + \gamma_{u,k}/\rho_u\right)$   $\Theta(1)$
            $\gamma_{u,k} = \gamma_{u,k} + \eta\rho_u\left(x_{u,k} - p_{u,k}\right)$   $\Theta(1)$
        end for                                                    –
        for $i \in I$                                              $\times|I|$
            $y_{i,k} = \left(Y\_U_i + \tau_i q_{k,i} - \kappa_{i,k}\right)/(Y\_D_i + \tau_i)$
                                                                   $\Theta(1)$
            $q_{i,k} = \max\left(0, y_{i,k} + \kappa_{i,k}/\tau_i\right)$   $\Theta(1)$
            $\kappa_{i,k} = \kappa_{i,k} + \eta\tau_i\left(y_{i,k} - q_{i,k}\right)$   $\Theta(1)$
        end for                                                    –
        for $e_{u,i} \in E_K$                                      $\times|R_K|$
            $e_{u,i} = e_{u,i} + x_{u,k} \cdot y_{i,k} - X\_C_u \cdot Y\_C_i$.
                                                                   $\Theta(1)$
        end for                                                    –
    end for                                                        –
    $t = t + 1$.                                                   $\Theta(1)$
end while                                                          –
Output: non-negative features $P$ and $Q$ built on $R_K$
```

e.g., (0, 0.5) to obtain fine initialization], while P, Q, Γ, and K can be set to zeroes due to their dependence on X and Y.

After initialization, we fill auxiliary array E_K with initial estimation corresponding to each known entry $r_{u,i}$. In each iteration, we follow (16) to sequentially update each column of X and Y, corresponding to the alternative optimization on each dimension of involved latent features. Before training the features of the kth dimension, their initial values, i.e., $X_{\cdot,k}$ and $Y_{\cdot,k}$, are cached in auxiliary arrays X_C and Y_C for updating estimations in E_K later on.

Then, a simple traverse on R_K is conducted for recording training variations related to user u and item i on the entry $r_{u,i}$. Subsequently, the kth column of X and Y is updated depending on variations recorded in auxiliary arrays X_U, X_D, Y_U, and Y_D. Note that we make the update of P, Q, Γ, and K once we update X and Y. This is equivalent to (16). And from (17), we observe that the update of the kth column in P and Γ depends on the kth column of X only, and that of the kth column in Q and K depends on the kth column of Y only. After updating all parameters, we update the estimations recorded in E_K according to the latest state of X and Y. By analyzing Algorithm ANLF, we obtain the following results.

Theorem 1: Let t and f denote the number of training iterations and the dimension of the latent feature space, respectively. With $|R_K| \gg \max\{|U|, |I|\}$, Algorithm ANLF has the computational complexity $\Theta(|R_K| \times t \times f)$.

Proof: First of all, we summarize the computational cost of each step in Algorithm ANLF. From it, we see that its computational complexity consists of three parts.

1) The initialization of feature matrices and auxiliary arrays, where the dominant term costs $\Theta(\max\{|U|, |I|\} \times f)$.
2) The initialization of parameter arrays ρ, τ, and E_K, which costs $\Theta(|R_K| \times f)$.
3) The optimization process has the computational cost

$$\begin{aligned} T &= t \times (f \times (|U| + |I| + 4|R_K| + 3|U| \\ &\qquad + 3|I| + |R_K|) + 1) \\ &= t \times f \times (5|R_K| + 4|U| + 4|I|) + t \\ &= \Theta(|R_K| \times t \times f) \end{aligned} \tag{22}$$

where the last step is obtained by reasonably omitting the lower order terms with the condition $|R_K| \gg \max\{|U|, |I|\}$. Hence, its computational complexity is given by $\Theta(|R_K| \times t \times f)$. ◇

In terms of storage complexity, ANLF requires the following data structures.

1) Two arrays of length $|R_K|$ to cache the given entry and corresponding estimations.
2) Three matrices of size $|U| \times f$ to cache X, P, and Γ.
3) Three matrices of size $|I| \times f$ to cache Y, Q, and K.
4) Four arrays of length $|U|$ to implement the update of X.
5) Four arrays of length $|I|$ to implement the update of Y.

Thus, its storage complexity is given by

$$\begin{aligned} S &= 2|R_K| + 3|U| \times f + 3|I| \times f + 4|U| + 4|I| \\ &= \Theta\left(f \times \max\left\{\frac{|R_K|}{f}, |U|, |I|\right\}\right). \end{aligned} \tag{23}$$

Therefore, if the target rating-matrix is extremely sparse, i.e., $|R_K| \ll |U| \times |I|$, then Algorithm ANLF is highly efficient in both computation and storage. This conclusion will be verified next via experiments.

IV. Experiments and Results

A. General Settings

1) Evaluation Metrics: The main concern of our experiments is to validate the accuracy and efficiency of the proposed ANLF. For accuracy, we mainly care about the closeness of estimations to the actual entries, since it directly reflects whether or not the model has captured the essential characteristics of given data [44], [45]. Hence, we choose the root mean squared error (RMSE) and mean absolute error (MAE) [44], [45] as the accuracy metrics, both of which are widely used for evaluating the statistical accuracy of CF models. Formally, the RMSE of a CF model is given by

$$\mathrm{RMSE} = \sqrt{\left(\sum_{(u,i)\in \mathrm{T}} (r_{u,i} - \hat{r}_{u,i})^2\right) \Big/ |\mathrm{T}|} \tag{24}$$

where T denotes the validation data set. Similarly, the MAE of a CF model is given by

$$\mathrm{MAE} = \left(\sum_{(u,i)\in \mathrm{T}} |r_{u,i} - \hat{r}_{u,i}|_{\mathrm{abs}}\right) \Big/ |\mathrm{T}| \tag{25}$$

where $|\cdot|_{\mathrm{abs}}$ denotes the absolute value of a given number.

In terms of efficiency, we care about: 1) the computational efficiency of each iteration; 2) the convergence rate; and 3) the storage complexity. Thus, we have recorded: 1) the average time consumption by one iteration of each tested model; 2) the necessary number of iterations to make the model converge; and 3) the memory consumption when running each model on experimental data sets. All experiments are conducted on a Tablet with a 2.5-GHz i5 CPU and 32-GB RAM. All models are implemented in JAVA SE 7U60 to test their capability for industrial usage.

2) Data Sets: Two data sets are employed in our experiments.

1) *D1 (Jester 1.1 M Data Set):* Collected through the joke-recommender Jester [46], and contains 1 186 324 continuous ratings in the scale of [−10, 10] from 24 983 users on 100 jokes. Its density is 47.32%, which is relatively high in CF problems.
2) *D2 (MovieLens 10 M Data Set):* It is also collected by the MovieLens system [40], and contains 10 000 054 ratings in the range of [0, 5], by 69 878 users on 10 677 movies. Its rating density is 1.34% only. We employ D2 to simulate large-scale sparse-matrices in real applications.

To keep the experimental results comparable on both data sets, we have mapped the data of D1 into the scale of [0, 5]. To obtain objective and unbiased results, we have employed the 80%–20% train-test settings and five-fold cross-validation.

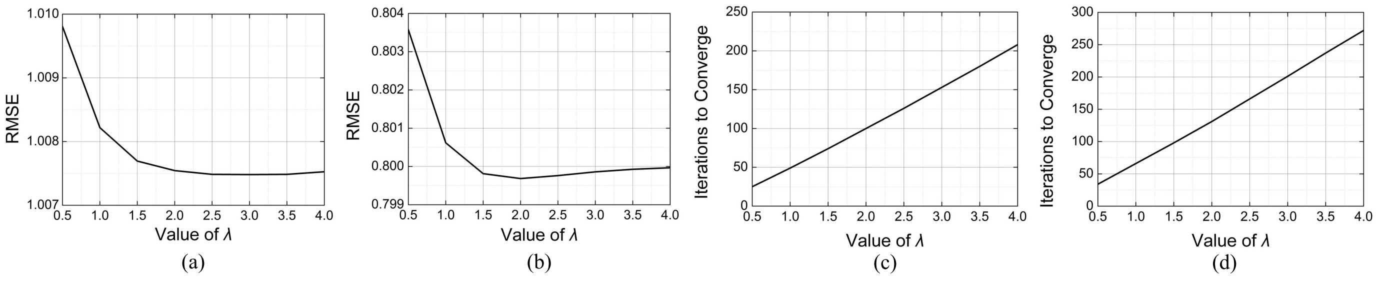


Fig. 2. Performance of ANLF measured by RMSE as λ increases. (a) RMSE on D1. (b) RMSE on D2. (c) Iterations to converge on D1. (d) Iterations to converge on D2.

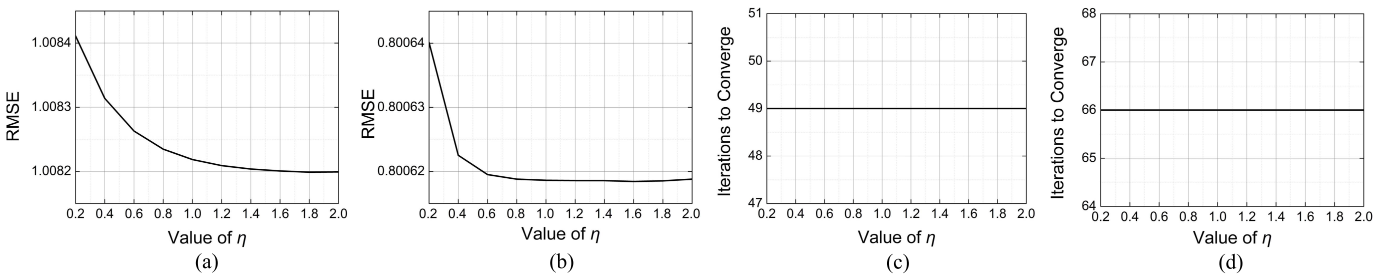


Fig. 3. Performance of ANLF measured by RMSE as η increases. (a) RMSE on D1. (b) RMSE on D2. (c) Iterations to converge on D1. (d) Iterations to converge on D2.

B. Parameter-Sensitive Tests

As shown in (15), the training process of ANLF depends on the hyperparameters η and λ, where η is the learning rate for the dual gradient ascent of Γ and K, and λ is the coefficient controlling the augmentation effect. Thus, it is necessary to conduct parameter-sensitive tests with respect to them to show their impact on the performance of ANLF. In this part, the dimension of the latent space f is set at 20. Here we present the results measured by RMSE; however, the results measured by MAE are very similar.

First, we fix the value of η at 1, and test ANLF with λ increasing from 0.5 to 4 on all four data sets, to show the effect of this parameter. During the test, the final RMSE of the converged model, along with the necessary number of training iterations corresponding to different λ values are recorded, as shown in Fig. 2. Based on these results, we have the following findings.

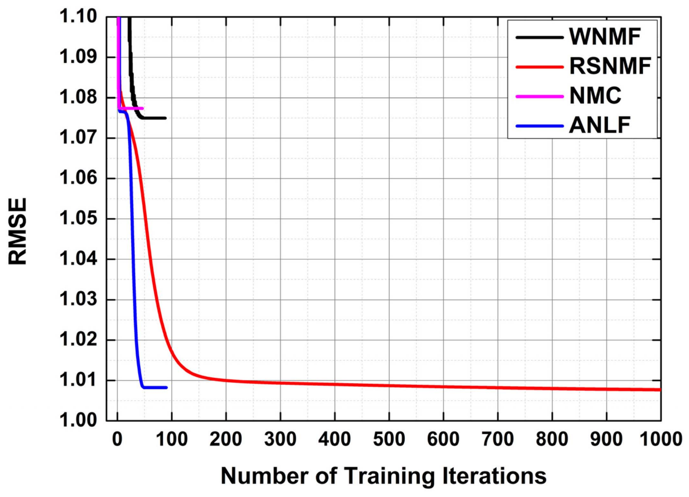
1) By carefully manipulating the augmentation terms, ANLF converge very fast while maintaining the nonnegativity. As shown in Figs. 2(c) and (d), ANLF generally requires only 30 or less iterations to converge with $\lambda = 0.5$ on all data sets under nonnegativity constraints.
2) Obviously, the prediction accuracy of ANLF is connected with the value of λ, as shown in Figs. 2(a) and (b). In general, as λ increases, its RMSE on all four data sets arrives at some minimum points, and then increases slowly, as shown in Fig. 2(a) and (b). Meanwhile, we have also noticed that the difference in RMSE with different λ values is not very significant, e.g., on D1, the highest RMSE with $\lambda = 0.5$ is 1.0098, and the lowest one 1.0075 appears with $\lambda = 3$, leading to the difference at 0.0023, or 0.22% only.
3) We find a rather strict linear relationship between the convergence rate of ANLF and the value of λ. This is clearly seen from Figs. 2(c) and (d).

Based on these results, we have subsequently tested its performance with $\lambda = 1.0$ and η varying in the range of [0.2, 2]. The results of this part are shown in Fig. 3. From them, we have the following findings.

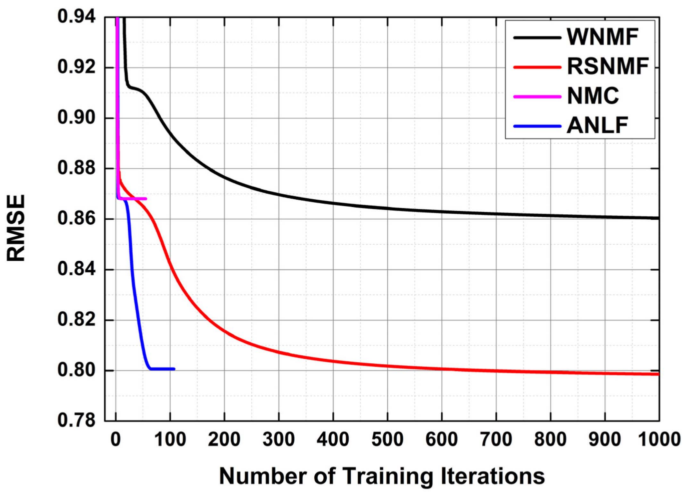
1) In the tested range, the value of η has slight impact on its prediction accuracy. As shown in Fig. 3(a) and (b), its RMSE tends to decrease as η increases, but the effect is tiny. For instance, on D1, its RMSE is 1.00841 with $\eta = 0.2$, and 1.00819 with $\eta = 2$, leading to the difference at 0.00022, or 0.02% only. Such a small gap can be nearly ignored in practice.
2) When λ is fixed, the value of η in the tested range does not affect its convergence rate. As shown in panels (c) and (d) of Figs. 3, the required number of iterations never changes with λ varying.

Based on the results of this part, we summarize that the hyperparameter λ decides the convergence rate of ANLF, while λ and η in their proper intervals have a very small effect on the prediction accuracy. In other words, it is rather robust to the variation of hyperparameters in such cases. However, we also find that when λ decreases behind a certain threshold (~0.3 on the experimental data sets), ANLF cannot converge well. Note that λ actually regularizes the augmented Langrangian (7). Hence, when λ is too small, the ADM-based training process may overshoot P and Q, thereby making the model unable to converge to a solution.

In general, we can employ relatively small λ from [0.5, 1], and simply fix η at 1 to obtain fine accuracy and convergence.



(a)



(b)

Fig. 4. RMSE comparison of WNMF, RSNMF, and ANLF. (a) D1. (b) D2.

C. Comparison Against Other Nonnegative CF Models

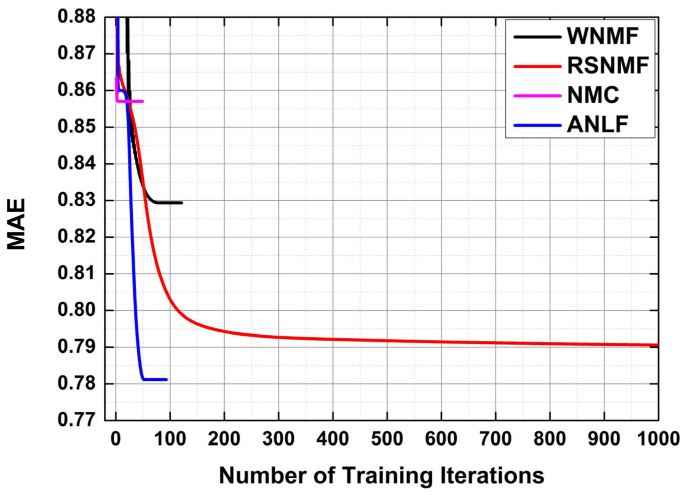
Four nonnegative models designed for CF problems, i.e., WNMF [28], RSNMF [32], NMC [39], and ANLF, are compared in this part. WNMF is a widely adopted model that can perform the NMF process on a sparse matrix in a CF problem [28]–[31], [42], [43], and is chosen as the benchmark. NMC is a recent model, which can handle the CF task on incomplete matrices under the nonnegativity constraints. RSNMF is a model based on the single-element-dependent form of multiplicative update under nonnegativity constraints. It has high accuracy and efficiency when addressing CF problems, but suffers from slow convergence [32]. We employ it as the competitor to see if the proposed ANLF model can outperform it in convergence rate, while maintaining the same-level accuracy and efficiency.

For all compared models, the common parameter—the dimension of the latent feature space f—is set at 20 uniformly. WNMF do not depend on any other hyperparameters. For RSNMF, based on a tuning process [32], the regularizing coefficient is set at 0.07 and 0.15 on D1 and D2, respectively. For NMC, the hyperparameters are tuned according to [39]. Since the performance of ANLF is rather steady with different settings of hyperparameters in the proper interval, we adopt a uniform setting of $\lambda = 1$ and $\eta = 1$ on both data sets.

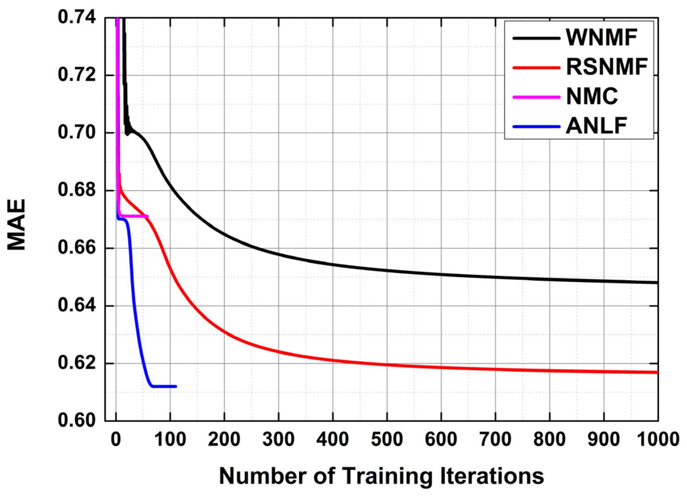
For all tested models, the termination criteria of a training process include: 1) the training iteration count reaches a preset threshold, i.e., 1000 and 2) the model converges, i.e., the training error start to increase. To avoid misjudging caused by training fluctuations, for each tested model, we make its training process to last 40 more iterations after it seems to converge at a certain point. Its prediction error at the possible converging point is continuously recorded to illustrate the model convergence performance well.

The comparison in RMSE and MAE of all tested models is depicted in Figs. 4 and 5. Their lowest RMSE and MAE, along with the iteration count are recorded in Table I. Their average time consumption in one training iteration and the total time required to converge are recorded in Table II. Their memory consumption during the experiments is recorded in Table III. From these results, we have the following findings.

1) From Figs. 4 and 5, ANLF is able to converge very fast. In general, it requires only dozens of training iterations to achieve the lowest RMSE/MAE on both data sets. Due to the multiplicative updates under nonnegativity constraints, WNMF encounters long-tail convergence on D2. It can converge quite fast on D1. However, this fast convergence is achieved at the expense of low prediction accuracy. For example, as shown in Table I, on D1, WNMF converges after 48 training iterations to achieve the lowest RMSE at 1.0749, which is ∼6.68% higher than that achieved by RSNMF, and 6.47% higher than that achieved by ANLF. Meanwhile, RSNMF has the lowest convergence rate. On all four data sets, it consumes 1000—which is the maximum iteration count in our settings—to obtain its lowest RMSE and MAE. NMC can converge even faster than ANLF; usually, it requires only a few training iterations to achieve the lowest MAE/RMSE, e.g., on D1 it converges with only five training iterations. However, in spite of its fast convergence, its prediction accuracy is lower than that of ANLF or RSNMF. For instance, on D1, its lowest RMSE at 1.0773 is ∼6.68% and 6.92% higher than those achieved by ANLF and RSNMF, respectively.
2) The prediction accuracy of ANLF is very competent. For example, on D1, ANLF can achieve the RMSE at 1.0098 with 49 iterations, which is 0.22% higher than that achieved by RSNMF with 1000 iterations, 6.45% lower than that by WNMF with 48 iterations, and 6.68% lower than that achieved by NMC with five iterations. Similar situations can be found from the results on D2, as shown in Fig. 4(b) and Table I. Moreover, ANLF achieves the lowest MAE among all tested models on all four data sets. For instance, on D1 it obtains the lowest MAE at 0.7812, ∼1.19% lower than 0.7905 achieved



(a)



(b)

Fig. 5. MAE comparison of WNMF, RSNMF, and ANLF. (a) D1. (b) D2.

TABLE I
RMSE, MAE, AND CONSUMED ITERATIONS OF COMPARED NONNEGATIVE MODELS

	D1		D2	
	RMSE	MAE	RMSE	MAE
WNMF	1.0749/48	0.8293/81	0.8603/1000	0.6481/1000
RSNMF	1.0076/1000	0.7905/1000	0.7986/1000	0.6168/1000
NMC	1.0773/5	0.8572/8	0.8682/14	0.6711/18
ANLF	**1.0098/49**	**0.7812/52**	**0.8006/66**	**0.6119/68**

TABLE II
TIME CONSUMPTION OF COMPARED NONNEGATIVE MODELS

	D1		D2	
	Average (ms)	Total (seconds)	Average (ms)	Total (seconds)
WNMF	1,325	64/107	14,671	14,671/14,671
RSNMF	159	159/159	1,779	1,779/1,779
NMC	4,602	23/37	3145,127	44,031/56,612
ANLF	**475**	**23/25**	**5,831**	**385/397**

by RSNMF, 6.12% lower than 0.8293 by WNMF, and 9.73% lower than 0.8572 by NMC, as recorded in Fig. 5(a) and Table I. Similar conclusions can be drawn on the other data set.

3) ANLF can be put into industrial use. On D2, where the matrix is 69 878 × 10 677 with less than 10 million given entries, the average time cost by a training iteration of ANLF is ∼5.83 s, as shown in Table II. When measured by MAE, ANLF needs 68 training iterations to converge on this data set, which means consuming 397 s to deal with such a huge, sparse matrix under nonnegativity constraints. RSNMF requires much less time for one iteration than ANLF. The reason is that the optimization process of ANLF is more complicated than that of RSNMF. According to (17), six matrices, i.e., X, Y, P, Q, Γ, and K, are involved in its training, while RSNMF has to train P and Q only [32]. However, RSNMF requires much more training iterations to achieve the prediction accuracy close to that of ANLF. Hence, RSNMF actually consumes more time than ANLF does. For instance, on D2, RSNMF consumes 1779 s or ∼29.7 min to achieve the lowest RMSE/MAE, which is much longer than that consumed by ANLF, as shown in Table II. We also notice that the time needed by an iteration of WNMF and NMC is much more than that by ANLF and RSNMF. This is due to their dependence on the matrix-manipulation-based nonnegative update; both need to carry out matrix multiplications between the target matrix and feature matrices, which leads to the computational complexity at $\Theta(|U| \times |I| \times f)$ in theory. In contrast, RSNMF and ANLF only require traversing on the known entry set R_K due to their single-element-dependent nonnegative update, and thus achieve the computational complexity at $\Theta(|R_K| \times f)$. Since in CF problems, the target matrix is very sparse, we have $|R_K| \ll |U| \times |I|$. Therefore, WNMF and NMC require much more time than RSNMF and ANLF do. Moreover, as recorded in Table II, we find that NMC consumes even much more time than WNMF does. Although the computational complexity of both models is $\Theta(|U| \times |I| \times f)$, WNMF introduces the binary weight matrix to implement the matrix multiplications on sparse matrices [28]. Such operation can be accelerated by ignoring the operations corresponding to zero entries. On the other hand, NMC employs a full matrix corresponding to the target matrix to carry out the MF process. Hence, its each operation is performed on this $|U| \times |I|$ matrix, thereby resulting in considerable consumption of time. For instance, on D2, NMC consumes 15.73 h for the initial 18 iterations to achieve the lowest MAE, and another 34.95 h for the next 40 iterations to check its convergence. Such huge consumption of time makes its practical usage unlikely.

4) ANLF also possesses fair scalability because of its low memory consumption. As shown in Table III, among all

TABLE III

MEMORY CONSUMPTION OF COMPARED NONNEGATIVE MODELS

	Space Complexity	Memory Consumption (GB)	
		D1	D2
WNMF	$Q(\|U\|\times\|I\|)$	0.32	9.76
RSNMF	$Q(f\times\max\{\|R_K\|/f,\|U\|,\|I\|\})$	0.15	0.83
NMC	$Q(\|U\|\times\|I\|)$	0.47	18.95
ANLF	$\mathbf{Q(f\times\max\{\|R_K\|/f,\|U\|,\|I\|\})}$	**0.17**	**1.24**

four tested models, RSNMF requires the least memory to carry out its single-element-dependent multiplicative update. The memory consumption of ANLF is slightly higher than that of RSNMF, due to its usage of more arrays for accelerating the training process. However, the difference is rather small. On the other hand, the memory consumption of WNMF and NMC is significantly higher than that of RSNMF and ANLF. This should be due to their usage of full matrices corresponding to the target matrix. WNMF employs a weight matrix and NMC employs two mapping matrices. They are of size $|U| \times |I|$. Hence, when dealing with extremely sparse matrices, the scalability of WNMF and NMC is not as good as that of RSNMF and NMC.

5) We conclude that ANLF is a rather balanced model with: 1) fast convergence; 2) high prediction accuracy; 3) high computational efficiency; 4) low memory consumption; and 5) the fulfillment of nonnegativity constraints. Therefore, it is a competent model for large scale CF problems, where the nonnegativity constraints should be fulfilled.

D. Comparison Against Sophisticated Models Without Nonnegativity Constraints

In Section IV-C, we have compared ANLF against several sophisticated nonnegative models designed for the problem of CF. However, is ANLF still competent when compared with state-of-the-art models without the nonnegativity constraints? To answer this question, we have compared ANLF against three up-to-date MF-based CF models: 1) the SVD++ model proposed in [11]–[14]; 2) the weighted alternating least squares (WALS) model [50]; and 3) the weighted-trace-norm regularized MF (WRMF) [18]. Among all tested models, only the proposed ANLF fulfills nonnegativity constraints.

For all tested models, f is set at 20 uniformly. The hyperparameters of SVD++, WALS, and WRMF are tuned according to instructions in [11], [14], [18], and [50]. For ANLF, λ and η are set at 1. The training of each model terminates when: 1) the iteration count reaches a preset threshold, i.e., 1000 and 2) the model converges, i.e., the training error starts to increase. For each tested model, 40 more iterations are taken after it seems to converge to avoid misjudging caused by fluctuations.

For all tested models, the lowest RMSE and MAE, along with the iteration count, are recorded in Table IV. Their average time consumption in one iteration and the total time consumption are given in Table V. Their memory consumption during the experiments is shown in Table VI. From these results, we conclude that the following.

TABLE IV

RMSE, MAE, AND CONSUMED ITERATIONS OF COMPARED MODELS

	D1		D2	
	RMSE	MAE	RMSE	MAE
SVD++	1.0152/42	0.8034/46	0.7980/71	0.6223/89
WALS	1.0294/9	0.8105/11	0.8078/12	0.6308/14
WRMF	1.0121/37	0.7992/41	0.7948/46	0.6184/49
ANLF	**1.0098/49**	**0.7812/52**	**0.8006/66**	**0.6119/68**

TABLE V

TIME CONSUMPTION OF COMPARED MODELS

	D1		D2	
	Average (ms)	Total (seconds)	Average (ms)	Total (seconds)
SVD++	177	7/8	1,578	112/140
WALS	3,688	33/41	1,608,878	19,306/22,524
WRMF	192	7/8	1,612	74/79
ANLF	**475**	**23/25**	**5,831**	**385/397**

TABLE VI

MEMORY CONSUMPTION OF COMPARED MODELS

	Space Complexity	Memory Consumption (GB)	
		D1	D2
SVD++	$Q(f\times\max\{\|R_K\|/f,\|U\|,\|I\|\})$	0.14	1.07
WALS	$Q(f\times\max\{\|R_K\|/f,\|U\|,\|I\|\})$	0.22	1.55
WRMF	$Q(f\times\max\{\|R_K\|/f,\|U\|,\|I\|\})$	0.16	1.12
ANLF	$\mathbf{Q(f\times\max\{\|R_K\|/f,\|U\|,\|I\|\})}$	**0.17**	**1.24**

1) The prediction accuracy and convergence rate of ANLF are competent when compared with these models. As shown in Table IV, on D1, ANLF's RMSE is slightly lower than the other tested models. On D2, the situation is a bit different; ANLF's RMSE is slightly higher than that by SVD++ or WRMF. However, when measured by MAE, ANLF outperforms the other tested models on both data sets.
2) The computational efficiency of ANLF is comparable with that of sophisticated models. As shown in Table V, the time consumed by ANLF is higher than that by SVD++ or WRMF. This efficiency gap is caused by its maintenance of nonnegativity constraints. Note that, it does not prevent ANLF from its industrial usage as we analyzed in Section IV-C. We also notice that the time consumed by WALS is much higher than that by the other three involved models. This is because its training requires multiplying the user/item matrices with their transposes; when the user/item counts is very large, this operation costs much time. Hence, although the converging rate of WALS is faster than the other tested models as shown in Table IV, it costs much more time than the others do, as given in Table V.

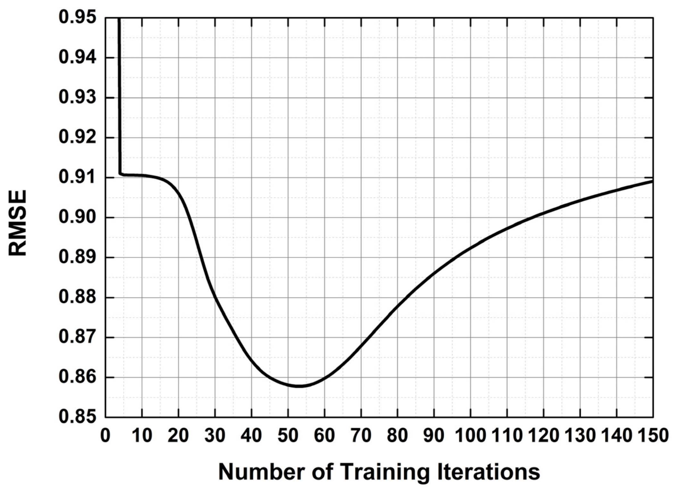


Fig. 6. RMSE of ANLF with overtraining on D1.

3) In terms of space complexity, all tested models are highly scalable and have closely low memory cost, which is easily to resolve in real applications. Hence, the scalability of ANLF is as good as that of the up-to-date models.
4) To summarize, when compared to state-of-the-art MF-based CF models without nonnegativity constraints, ANLF is able to achieve highly competent prediction accuracy and scalability while fulfilling the nonnegativity constraints. Nonetheless, its computational efficiency is lower than SVD++ and WRMF but much higher than WALS. This issue are worthy of further investigations.

V. Conclusion

A. Summary of Experimental Results

This paper focuses on implementing an NLF model that is able to address the large-scale sparse matrices with high performance. Through integrating the principle of the ADM optimization framework, it derives ANLF that is a rather balanced model. Compared with several state-of-the-art nonnegative models able to handle the incomplete matrices in CF problems, ANLF is advantageous in efficiency, scalability, and accuracy while maintaining the nonnegativity. Note that our experiments do not involve CF models without the nonnegativity constraints. Actually, with such constraints, the learning objective of a model is restricted. As demonstrated by pioneering research [28]–[32], such restrictions help the model better capture the actual meanings, such as the community tendency, and make it more interpretable. However, they also perplex and bias a training process. Hence, it is reasonable to compare the proposed ANLF with the models that also fulfill the nonnegativity constraints.

B. Industrial Usage

As described in Algorithm ANLF, its single-element-dependent optimization process not only ensures the low computational and storage complexity, but also make the algorithm easy to implement in any programming language, because: 1) it does not require any specific data structures like sparse matrices and 2) instead of matrix operations, only simple operations among single parameters are involved. Hence, ANLF is easy to implement. In addition, it is highly robust to different settings of hyperparameters within the proper scale, i.e., λ can be a value in [0.3, 2] and η in [0.2, 2] based on our experimental trials. As a result, it can suit real, big-data applications of learning systems with few tuning-processes. Such characteristics of ANLF can surely enhance its capability in fulfilling industrial requirements.

C. Convergence of ANLF

The single-element-dependent ADM-based optimization of ANLF makes it complicated to offer a theoretical proof of the model convergence. First of all, (4) is bilinear and nonconvex. Moreover, its single-element-dependent training process actually does the one-pass of Gauss-Seidel. Hence, several approximations should be made to fulfill the converging conditions of an ADM-based model [41]. The performance of ANLF is well supported by the experimental results. However, the theoretical analysis regarding its convergence remains to be performed.

D. Implementation Details

When coming to real applications, two issues regarding ANLF are remarked to achieve its high performance.

1) Similar to other MF-based CF models [9]–[16], ANLF applies randomized initialization to involved parameters. With such an initialization, randomness is injected into the obtained model. As mentioned above, (4) handled by ANLF is nonconvex. With the randomized initialization, different runs of the training process on the same data set might start at initial points in the regions of attraction of different local optima, thereby resulting in models with small gaps in prediction accuracy. To ensure the prediction accuracy, a simple but effective strategy is via ensemble [47], [48], i.e., using multiple models that are randomly initialized, and then averaging their prediction results. Usually, three to five individual models can enable their ensemble to achieve steadily high accuracy.
2) The convergence rate of ANLF is fast. Redundant training after the model converges may lead to overfitting, thereby resulting in the loss of prediction accuracy. For instance, we have trained ANLF on D1 without any stopping criterion, and recorded its RMSE, as shown in Fig. 6. To prevent such overfitting in real applications, it is necessary to employ early stopping settings. As suggested by the studies in [9]–[16], a straightforward way to achieve so is by comparing the errors of consecutive training iterations, and stopping the training process when their difference meets a certain threshold. More specifically, the training process stops when the sum error increases, or the decrease of the sum error is less than a positive constant σ, which is chosen according to training data. In CF problems, a feasible choice for σ is $\sim 1 \times 10^{-5}$ [9]–[16].

E. Possible Extensions

This work uses ANLF to handle incomplete matrices in CF problems. However, as mentioned before, the idea of

MF-based CF has been employed in many related areas [22]–[27]. Since the requirements for latent feature analysis on large-scale sparse matrices under nonnegativity constraints can rise in these learning-system-related areas, ANLF can also be extended to handle them. In particular, it is suitable for applications requiring large-scale and sparse network analysis [49]–[59], which needs future work.

REFERENCES

[1] P. Resnick and H. R. Varian, "Recommender systems," *Commun. ACM*, vol. 40, no. 3, pp. 56–58, 1997.

[2] 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. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, Jun. 2005.

[3] F. Dinuzzo, G. Pillonetto, and G. De Nicolao, "Client–server multitask learning from distributed datasets," *IEEE Trans. Neural Netw.*, vol. 22, no. 2, pp. 290–303, Feb. 2011.

[4] J. L. De La Rosa, N. Hormazabal, S. Aciar, G. Lopardo, A. Trias, and M. Montaner, "A negotiation-style recommender based on computational ecology in open negotiation environments," *IEEE Trans. Ind. Electron.*, vol. 58, no. 6, pp. 2073–2085, Jun. 2011.

[5] Y. Li *et al.*, "An efficient recommendation method for improving business process modeling," *IEEE Trans. Ind. Informat.*, vol. 10, no. 1, pp. 502–513, Feb. 2014.

[6] Y. Cai, H.-F. Leung, Q. Li, H. Min, J. Tang, and J. Li, "Typicality-based collaborative filtering recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 3, pp. 766–779, Mar. 2014.

[7] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proc. 14th Conf. Uncertainty Artif. Intell.*, Madison, WI, USA, Jul. 1998, pp. 43–52.

[8] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. World Wide Web Conf.*, Hong Kong, May 2001, pp. 285–295.

[9] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Adv. Artif. Intell.*, vol. 2009, pp. 1–19, Jan. 2009.

[10] Y. Koren, "Collaborative filtering with temporal dynamics," *Commun. ACM*, vol. 53, no. 4, pp. 89–97, 2010.

[11] Y. Koren and R. Bell, "Advances in collaborative filtering," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. New York, NY, USA: Springer-Verlag, 2011.

[12] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Application of dimensionality reduction in recommender systems—A case study," in *Proc. ACM WebKDD Web Mining E-Commerce Workshop*, Boston, MA, USA, Aug. 2000, pp. 285–295.

[13] G. Takács, I. Pilászy, B. Németh, and D. Tikk, "Scalable collaborative filtering approaches for large recommender systems," *J. Mach. Learn. Res.*, vol. 10, pp. 623–656, Mar. 2009.

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

[15] R. R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Advances in Neural Information Processing Systems*, vol. 20. Red Hook, NY, USA: Curran Associates, 2008, pp. 1257–1264.

[16] K. Yu, S. Zhu, J. Lafferty, and Y. Gong, "Fast nonparametric matrix factorization for large-scale collaborative filtering," in *Proc. 32nd Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Boston, MA, USA, Jul. 2009, pp. 211–218.

[17] W. Chu and Z. Ghahramani, "Probabilistic models for incomplete multi-dimensional arrays," in *Proc. 12th Int. Conf. Artif. Intell. Statist.*, Clearwater Beach, FL, USA, Apr. 2009, pp. 89–96.

[18] N. Srebro and R. R. Salakhutdinov, "Collaborative filtering in a non-uniform world: Learning with the weighted trace norm," in *Advances in Neural Information Processing Systems*, vol. 23. Vancouver, BC, Canada, Dec. 2010, pp. 2056–2064.

[19] N. Houlsby, F. Huszar, Z. Ghahramani, and J. M. Hernández-Lobato, "Collaborative Gaussian processes for preference learning," in *Advances in Neural Information Processing Systems*, vol. 25. Lake Tahoe, NV, USA, Dec. 2012, pp. 2105–2113.

[20] S. Chatzis, "Nonparametric Bayesian multitask collaborative filtering," in *Proc. 22nd ACM Int. Conf. Inf. Knowl. Manage.*, San Francisco, CA, USA, Oct. 2013, pp. 2149–2158.

[21] J. M. Hernández-Lobato, N. Houlsby, and Z. Ghahramani, "Probabilistic matrix factorization with non-random missing data," in *Proc. 31st Int. Conf. Mach. Learn.*, Beijing, China, Jun. 2014, pp. 1512–1520.

[22] N. Zhou, W. K. Cheung, G. Qiu, and X. Xue, "A hybrid probabilistic model for unified collaborative and content-based image tagging," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 7, pp. 1281–1294, Jul. 2011.

[23] S. Boutemedjet and D. Ziou, "Predictive approach for user long-term needs in content-based image suggestion," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 8, pp. 1242–1253, Aug. 2012.

[24] J. Wu, L. Chen, Y. Feng, Z. Zheng, M. C. Zhou, and Z. Wu, "Predicting quality of service for selection by neighborhood-based collaborative filtering," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 2, pp. 428–439, Mar. 2013.

[25] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Collaborative web service QoS prediction via neighborhood integrated matrix factorization," *IEEE Trans. Services Comput.*, vol. 6, no. 3, pp. 289–299, Jul./Sep. 2012.

[26] M.-F. Weng and Y.-Y. Chuang, "Collaborative video reindexing via matrix factorization," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 8, no. 2, 2012, Art. ID 23.

[27] J. J. Pan, S. J. Pan, J. Yin, L. M. Ni, and Q. Yang, "Tracking mobile users in wireless networks via semi-supervised colocalization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 3, pp. 587–600, Mar. 2012.

[28] S. Zhang, W. Wang, J. Ford, and F. Makedon, "Learning from incomplete ratings using non-negative matrix factorization," in *Proc. 6th SIAM Int. Conf. Data Mining*, Bethesda, MD, USA, Apr. 2006, pp. 549–553.

[29] M. Wu, "Collaborative filtering via ensembles of matrix factorizations," in *Proc. KDD Cup Workshop*, San Jose, CA, USA, Aug. 2007, pp. 43–47.

[30] G. Chen, F. Wang, and C. Zhang, "Collaborative filtering using orthogonal nonnegative matrix tri-factorization," *Inf. Process. Manage.*, vol. 45, no. 3, pp. 368–379, 2009.

[31] Q. Gu, J. Zhou, and C. Ding, "Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs," in *Proc. SIAM Int. Conf. Data Mining*, Columbus, OH, USA, Apr. 2010, pp. 199–210.

[32] 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. Ind. Informat.*, vol. 10, no. 2, pp. 1273–1284, May 2014.

[33] P. Paatero and U. Tapper, "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values," *Environmetrics*, vol. 5, no. 2, pp. 111–126, 1994.

[34] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788–791, Oct. 1999.

[35] C.-J. Lin, "Projected gradient methods for nonnegative matrix factorization," *Neural Comput.*, vol. 19, no. 10, pp. 2756–2779, 2007.

[36] C. Ding, T. Li, and M. I. Jordan, "Convex and semi-nonnegative matrix factorizations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 1, pp. 45–55, Jan. 2010.

[37] N. Guan, D. Tao, Z. Luo, and B. Yuan, "Online nonnegative matrix factorization with robust stochastic approximation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 7, pp. 1087–1099, Jul. 2012.

[38] A. Cichocki, R. Zdunek, A. H. Phan, and S.-I. Amari, *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-Way Data Analysis and Blind Source Separation*. Natick, MA, USA: Wiley, 2009.

[39] Y. Xu, W. Yin, Z. Wen, and Y. Zhang, "An alternating direction algorithm for matrix completion with nonnegative factors," *Frontiers Math. China*, vol. 7, no. 2, pp. 365–384, 2012.

[40] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, "GroupLens: Applying collaborative filtering to Usenet news," *Commun. ACM*, vol. 40, no. 3, pp. 77–87, 1997.

[41] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[42] N. Zheng, Q. Li, S. Liao, and L. Zhang, "Which photo groups should I choose? A comparative study of recommendation algorithms in Flickr," *J. Inf. Sci.*, vol. 36, no. 6, pp. 733–750, 2010.

[43] H. Ma, I. King, and M. R. Lyu, "Learning to recommend with explicit and implicit social relations," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–19, 2011.

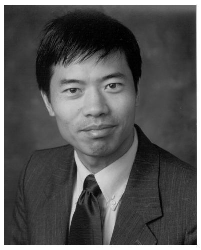
[44] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 5–53, 2004.

[45] G. Shani and A. Gunawardana, "Evaluating recommendation systems," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. New York, NY, USA: Springer-Verlag, 2011.

[46] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, “Eigentaste: A constant time collaborative filtering algorithm,” *Inf. Retr.*, vol. 4, no. 2, pp. 133–151, 2001.
[47] T. G. Dietterich, “Ensemble methods in machine learning,” in *Proc. 1st Int. Workshop Multiple Classifier Syst.*, Cagliari, Italy, Jun. 2000, pp. 1–15.
[48] L. Xin, Y. Ouyang, and X. Zhang, “Improving latent factor model based collaborative filtering via integrated folksonomy factors,” *Int. J. Uncertainty, Fuzziness, Knowl.-Based Syst.*, vol. 19, no. 2, pp. 307–327, 2011.
[49] X. Cao, X. Wang, D. Jin, Y. Cao, and D. He, “Identifying overlapping communities as well as hubs and outliers via nonnegative matrix factorization,” *Sci. Rep.*, vol. 3, pp. 1–8, Oct. 2013.
[50] Y. Hu, Y. Koren, and C. Volinsky, “Collaborative filtering for implicit feedback datasets,” in *Proc. 8th IEEE Int. Conf. Data Mining*, Pisa, Italy, Dec. 2008, pp. 263–272.
[51] C. Chen, D. Neal, and M. Zhou, “Understanding the evolution of a disaster—A framework for assessing crisis in a system environment (FACSE),” *Natural Hazards*, vol. 65, no. 1, pp. 407–422, Jan. 2013.
[52] W. Guo, W. M. Healy, and M. C. Zhou, “Impacts of 2.4 GHz ISM band interference on IEEE 802.15.4 wireless sensor network reliability in buildings,” *IEEE Trans. Instrum. Meas.*, vol. 61, no. 9, pp. 2533–2544, Sep. 2012.
[53] C. Jiang, H. Sun, Z. Ding, P. Wang, and M. C. Zhou, “An indexing network: Model and applications,” *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 44, no. 12, pp. 1584–1597, Dec. 2014.
[54] S. Liu, H. Cao, L. Li, and M. C. Zhou, “Predicting stay time of mobile users with contextual information,” *IEEE Trans. Autom. Sci. Eng.*, vol. 10, no. 4, pp. 1026–1036, Oct. 2013.
[55] C. Shang, M. C. Zhou, and C. Chen, “Cellphone Data and Applications,” *Int. J. Intell. Control Syst.*, vol. 19, no. 1, pp. 35–45, Mar. 2014.
[56] W. Tan and M. Zhou, “Service Oriented Workflow Systems,” in *Contemporary Issues in Systems Science and Engineering*, M. C. Zhou, H.-X. Li, and M. Weijnen, Eds. Hoboken, NJ, USA: Wiley, 2015, pp. 645–660.
[57] Y. Wu *et al.*, “A novel method for calculating service reputation,” *IEEE Trans. Autom. Sci. Eng.*, vol. 10, no. 3, pp. 634–642, Jul. 2013.
[58] M. Yu and M. C. Zhou, “A performance modeling scheme for multistage switch networks with phase-type and bursty traffic,” *IEEE/ACM Trans. Netw.*, vol. 18, no. 4, pp. 1091–1104, Aug. 2010.
[59] M. Yu, X. Ma, and M. C Zhou, “Radio channel allocations with global optimality and bounded computational scale,” *IEEE Trans. Veh. Technol.*, vol. 63, no. 9, pp. 4670–4680, Nov. 2014.

Xin Luo (M’14) received the B.S. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2005, and the Ph.D. degree in computer science from Beihang University, Beijing, China, in 2011.

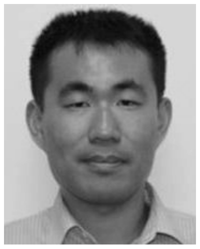
He joined the College of Computer Science, Chongqing University, Chongqing, China, in 2011, where he is currently an Associate Professor of Computer Science and Engineering. His current research interests include recommender systems, network analysis, service computing, and bioinformatics.



MengChu Zhou (S’88–M’90–SM’93–F’03) received the B.S. degree in control engineering from the Nanjing University of Science and Technology, Nanjing, China, in 1983, the M.S. degree in automatic control from the Beijing Institute of Technology, Beijing, China in 1986, and the Ph.D. degree in computer and systems engineering from the Rensselaer Polytechnic Institute, Troy, NY, USA, in 1990.

He joined the New Jersey Institute of Technology, Newark, NJ, USA, in 1990, where he is currently a Distinguished Professor of Electrical and Computer Engineering. His interests include Petri nets, Internet of Things, big data, and automation. He has authored over 600 publications, including 12 books, and over 290 journal papers (majority in IEEE TRANSACTIONS).

Prof. Zhou is a fellow of the International Federation of Automatic Control and the American Association for the Advancement of Science. He is the Editor of IEEE Press Book Series on *Systems Science and Engineering*.

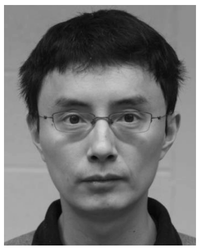


Shuai Li (M’14) received the B.E. degree in precision mechanical engineering from the Hefei University of Technology, Hefei, China, the M.E. degree in automatic control engineering from the University of Science and Technology of China, Hefei, and the Ph.D. degree in electrical engineering from the Stevens Institute of Technology, Hoboken, NJ, USA.

He is currently a Research Assistant Professor with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. His current research interests include distributed estimation and control, dynamic neural networks, robotic networks, and dynamic problems defined on a graph.

Zhuhong You (M’14) received the B.E. degree in electronic information science and engineering from Hunan Normal University, Changsha, China, in 2005, and the Ph.D. degree in pattern recognition and intelligent system from the University of Science and Technology of China, Hefei, China, in 2010.

He is currently a Post-Doctoral Fellow with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. His current research interests include pattern recognition, machine learning, and bioinformatics.



Yunni Xia (M’12) received the B.S. degree in computer science from Chongqing University, Chongqing, China, in 2003, and the Ph.D. degree in computer science from Peking University, Beijing, China, in 2008.

He joined the College of Computer Science, Chongqing University, in 2008, where he is currently an Associate Professor of Computer Science and Engineering. His current research interests include Petri nets, service computing, cloud computing, and Web services.

Qingsheng Zhu received the B.S., M.S., and Ph.D. degrees in computer science from Chongqing University, Chongqing, China, in 1982, 1985, and 2000, respectively.

He was a Visiting Scholar with Oxford University, Oxford, U.K., and the University of London, London, U.K., from 1993 to 1994, and a Visiting Professor with the University of Illinois at Chicago, Chicago, IL, USA, from 2001 to 2002. He is currently a Professor of Computer Science and Engineering with Chongqing University, and the Director of the Key Laboratory of Software Theory and Technology. His current research interests include service-oriented software engineering, data mining, and outlier detection.