Knowledge and Information Systems (2021) 63:105–124
https://doi.org/10.1007/s10115-020-01511-x

REGULAR PAPER

Incremental one-class collaborative filtering with co-evolving side networks

Chen Chen[1] · Yinglong Xia[2] · Hui Zang[1] · Jundong Li[3] · Huan Liu[4] · Hanghang Tong[5]

Received: 12 March 2019 / Revised: 26 August 2020 / Accepted: 30 August 2020 /
Published online: 17 September 2020

Abstract
One-class collaborative filtering (OCCF) is a fundamental research problem in a myriad of applications where the preferences of users can only be implicitly inferred from their one-class feedback (e.g., click an ad or purchase a product). The main challenges of OCCF lie in the sparsity of user feedback and the ambiguity of unobserved preferences. To effectively address the above two challenges, side networks from users and items are extensively exploited by state-of-the-art methods, which are predominantly focused on static settings. However, as real-world recommender systems evolve over time (where both the user–item ratings and user–user/item–item side networks will change), it is necessary to update OCCF results (e.g., the latent features of users and items) accordingly. The main obstacle for OCCF online update with co-evolving side networks lies in the fact that the coupled system is highly sensitive to local changes, which may cause massive perturbation on the latent features of a large number of users and items. In this paper, we propose a novel incremental one-class collaborative filtering (OCCF) method that can cope with co-evolving side networks efficiently. In particular, we model the evolution of latent features as a linear transformation process, which enables fast update of the latent features on the fly. Empirical experiments demonstrate that our method can provide high-quality recommendation results on real-world datasets.

Keywords Incremental algorithms · One-class collaborative filtering · Evolving networks

1 Introduction

The past decade has witnessed the prosperity of recommender systems in various applications, ranging from e-commerce platforms to online service providers. Among the numerous recommendation algorithms in the literature, collaborative filtering-based methods are widely adopted in many applications due to its superior effectiveness. Traditional collaborative filtering algorithms are typically designed to provide recommendations based on users' explicit, multi-scale feedback (e.g., rating 1–5). However, in many real applications, the preferences

The work was done while the first three authors were working at Futurewei Technologies, Inc.

Extended author information available on the last page of the article

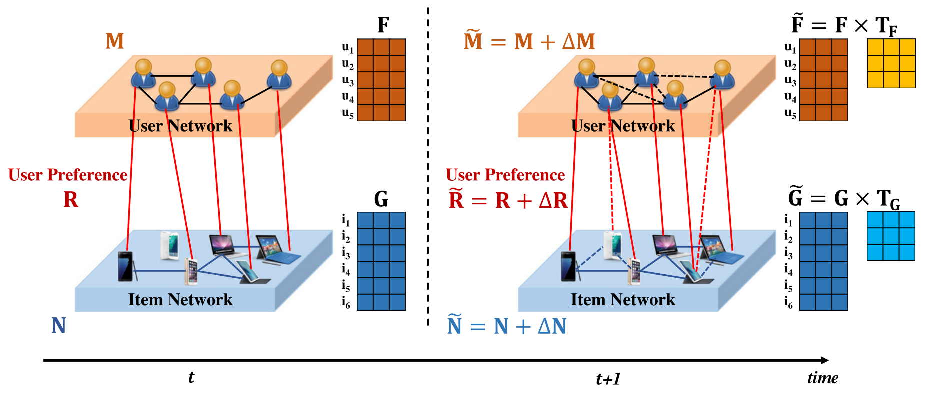

M
F
u_1
u_2
u_3
u_4
u_5
User Network
User Preference
R
G
i_1
i_2
i_3
i_4
i_5
i_6
Item Network
N
$\tilde{M} = M + \Delta M$
$\tilde{F} = F \times T_F$
User Preference
$\tilde{R} = R + \Delta R$
$\tilde{G} = G \times T_G$
$\tilde{N} = N + \Delta N$
t
t+1
time

Fig. 1 An illustration of online one-class recommendation problem with side networks. Solid lines represent the links in the original system, dashed lines represent the newly emerged links (best viewed in color)

might only be inferred from users' implicit, one-class feedback (e.g., actions or inactions). For example, it is reasonable to infer that a user likes a song if s/he listened to it from the beginning to the end; otherwise, s/he may not be into the song. Such applications are generally formulated as one-class collaborative filtering (OCCF) problems [24].

The key challenges for OCCF lie in the sparsity of positive feedback (preferences) and the ambiguity of missing preferences. A promising way to address those issues is to exploit side information from the social networks of users and/or similarity networks of items as in [23] and [42]. The key idea behind those methods is that socially connected users tend to share similar tastes on items, while similar items are more likely to impose similar impact to users.

In real applications, the user preferences in the systems often evolve over time, which would inevitably affect the preference inference results in the original static system. For example, in the e-commerce platform shown in Fig. 1, new friendship relations (black dashed lines) and user preferences (red dashed lines) are emerging over time. Meanwhile, as new products are being released to the market, similarity links between newly released products and existing products would appear in the system as well. In such a coupled system, the emergence of new connections and preferences may cause a ripple effect to the platform, hence affect the preferences of a large proportion of users. The dynamics of the system have spurred the research on collaborative filtering in different ways. One line of research is to treat the system dynamics as the temporal data input, and the goal is to learn a unified user model to predict the potential preference for the current step [15]. In such temporal recommender systems, the learnt user model would need to be re-calibrated when new links are established over the time, making it inefficient to maintain on the fly. Another line of research is to treat the emerging links at each step as the incremental changes to the original system snapshot. Thus, the representation for users and items can be efficiently updated with some evolutionary algorithms like incremental matrix factorization [38]. However, most of the evolutionary algorithms only exploit the newly observed preferences between users and items, with few attempts to incorporate social network between users and similarity network between items into the model.

In this paper, we propose an efficient algorithm to incrementally update one-class collaborative filtering results with co-evolving side networks. To efficiently accommodate the changes in the system, we propose to model the evolution of latent features based on the following observations: the system often evolves smoothly between two consecutive timestamps such that a large number of observed preferences and network links remain unchanged. Thus, we can view the new latent features as a subtle linear transformation from the previous features. This would in turn allows us to incrementally solve the OCCF problem in a timely manner without re-solving it from scratch.

The main contributions of the paper can be summarized as follows:

- *Problem Formulation.* We formally define the problem of incremental OCCF with co-evolving side networks.
- *Algorithms and Analysis*. We propose an incremental OCCF algorithm (ENCORE) that can efficiently accommodate system dynamics and analyze its optimality and complexity.
- *Evaluations*. We empirically evaluate the proposed method on real-world datasets to verify its effectiveness and efficiency.

The rest of the paper is organized as follows. In Sect. 2, we give the formal definition of the problem. Section 3 presents the proposed method and analysis. Section 4 shows the evaluation results on real datasets. Section 5 reviews the related work, and Sect. 6 concludes the paper.

2 Problem definition and preliminaries

2.1 Problem definition

In this section, we first give a formal definition of the studied problem of incremental OCCF with co-evolving side networks. After that, we provide the preliminaries to facilitate the understanding of the proposed algorithm.

The main symbols used in the paper are summarized in Table 1. We use bold uppercase for matrices (e.g., $\mathbf{A}$) and $\Delta\mathbf{A}$ for the perturbation matrix of $\mathbf{A}$. $\tilde{}$ sign denotes the notations after adding the perturbations into the system (i.e., $\tilde{\mathbf{A}} = \mathbf{A} + \Delta\mathbf{A}$). $'$ sign denotes the matrix transpose.

With the above notations, we first define the static OCCF problem with side networks as follows.

Definition 1 The problem of static OCCF problem with side networks.
Given: $\Gamma =< \mathbf{M}, \mathbf{N}, \mathbf{R} >$ where $\mathbf{M}$ is an $n_u \times n_u$ social network between users; $\mathbf{N}$ is an $n_i \times n_i$ similarity network between items; and $\mathbf{R}$ is an $n_u \times n_i$ user preference matrix, in which $\mathbf{R}(i, j) = 1$ if user i shows preference on item j, otherwise $\mathbf{R}(i, j) = 0$.
Output: The inferred preference between user u and item i in the original system $\Gamma =< \mathbf{M}, \mathbf{N}, \mathbf{R} >$.

Based on the above definition, we give the formal definition of incremental OCCF problem with co-evolving side networks.

Problem 1 The problem of incremental OCCF with co-evolving side networks.
Given: (1) The original system $\Gamma =< \mathbf{M}, \mathbf{N}, \mathbf{R} >$; (2) the perturbation of the system $\Delta\Gamma =< \Delta\mathbf{M}, \Delta\mathbf{N}, \Delta\mathbf{R} >$; (3) the $n_u \times r$ latent feature matrix $\mathbf{F}$ for users in the original system Γ; and (4) the $n_i \times r$ latent feature matrix $\mathbf{G}$ for items in the original system Γ.
Output: The inferred preference between user u and item i in the updated system $\tilde{\Gamma} =< \tilde{\mathbf{M}}, \tilde{\mathbf{N}}, \tilde{\mathbf{R}} >$.

Table 1 Main symbols

Symbol	Definition and Description
$\mathbf{A}, \mathbf{B}$	Adjacency matrices
$\Delta\mathbf{A}$	Perturbation matrix of $\mathbf{A}$
$\tilde{\mathbf{A}}$	Updated matrix of $\mathbf{A}$
$\mathbf{A}(i, j)$	The element at ith row jth column in $\mathbf{A}$
$\mathbf{A}'$	Transpose of matrix $\mathbf{A}$
$\mathbf{M}$	The adjacency matrix of user network
$\mathbf{N}$	The adjacency matrix of item network
$\mathbf{D}_M, \mathbf{D}_N$	The diagonal degree matrices for $\mathbf{M}$ and $\mathbf{N}$
$\mathbf{R}$	The preference matrix for users w.r.t. items
Γ	The recommendation problem with side Networks $\Gamma =< \mathbf{M}, \mathbf{N}, \mathbf{R} >$
$\mathbf{W}$	The weight matrix for $\mathbf{R}$
$\mathbf{F}$	The latent feature matrix for users
$\mathbf{G}$	The latent feature matrix for items
$\mathbf{F}(u, :)$	The latent feature for use u
$\mathbf{G}(i, :)$	The latent feature for item i
n_u, n_i	Number of users and items
m_u, m_i	Number of edges in $\mathbf{M}$ and $\mathbf{N}$
m_r	Number of observed links in $\mathbf{R}$
r	The feature dimension for $\mathbf{F}$ and $\mathbf{G}$
t	The number of iterations

2.2 Preliminaries

Under static settings, OCCF problem with side networks can be solved with the following optimization problem [42]

$$\min_{\mathbf{F},\mathbf{G}\geq 0} \underbrace{\|\mathbf{W} \odot (\mathbf{R} - \mathbf{F}\mathbf{G}')\|_F^2}_{\text{Matching Observed Ratings}} + \underbrace{\beta(\|\mathbf{F}\|_F^2 + \|\mathbf{G}\|_F^2)}_{\text{Regularization}}$$
$$+ \underbrace{\alpha(\mathrm{tr}(\mathbf{F}'(\mathbf{D}_M - \mathbf{M})\mathbf{F}) + \mathrm{tr}(\mathbf{G}'(\mathbf{D}_N - \mathbf{N})\mathbf{G}))}_{\text{Node Homophily}} \tag{1}$$

where $\odot$ is the Hadamard product with $[\mathbf{A} \odot \mathbf{B}](i, j) = \mathbf{A}(i, j)\mathbf{B}(i, j)$, $\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2}$ is the Frobenius norm of matrix $\mathbf{A}$, and $tr(\mathbf{A}) = \sum_{i=1}^n a_{ii}$ is the trace of matrix $\mathbf{A}$. In the above objective function, $\mathbf{R}$ is the user preference matrix; $\mathbf{F}$ and $\mathbf{G}$ are the non-negative[1] latent feature matrices for users and items, respectively; $\mathbf{D}_M$ and $\mathbf{D}_N$ are the diagonal degree matrices for user network $\mathbf{M}$ and item network $\mathbf{N}$ (i.e., $\mathbf{D}_M(u, u) = \sum_k^{n_u} \mathbf{A}(u, k), \mathbf{D}_N(i, i) = \sum_k^{n_i} \mathbf{A}(i, k)$), respectively. $\mathbf{W}$ is an $n_u \times n_i$ weighting matrix, in which $\mathbf{W}(i, j) = 1$ if $\mathbf{R}(i, j) = 1$ (i.e., positive preference observed between user i and item j), otherwise $\mathbf{W}(i, j) \in [0, 1]$ if $\mathbf{R}(i, j) = 0$ (i.e., no preference observed between user i and item j). It is worth mentioning that the weight for unobserved links is

[1] The rationale behind the non-negative constraint is that non-negative matrix factorization is more capable of a representation for parts of data [16], making the factorization results more expressive for data reconstruction [8].

used to mitigate its uncertainty between potential positive preferences and negative examples. Consequently, different weighting strategies can be applied in different scenarios. In this paper, we follow [42] to set the weight of all unobserved entries to a global value w for the ease of computation.

In Eq. (1), the first term is used to match the preferences in matrix $\mathbf{R}$; the second term is to prevent overfitting of the model; and the third term is used to exploit node homophily in side networks. The intuition behind this term is that similar users would hold similar preferences to items (i.e., small $\|\mathbf{F}(u,:) - \mathbf{F}(v,:)\|_2^2$).[2] Correspondingly, similar items would possess similar attractiveness to users (i.e., small $\|\mathbf{G}(i,:) - \mathbf{G}(j,:)\|_2^2$). The entire optimization problem in Eq. (1) can be solved by non-negative matrix factorization techniques [17] with time complexity $O(((m_u + m_i + m_r)r + (n_u + n_i)r^2)t)$ (t denotes the number of iterations for the optimization algorithm). The inferred preference between user u and item i can be estimated by $\mathbf{F}(u,:)\mathbf{G}(i,:)'$, where $\mathbf{F}$ and $\mathbf{G}$ are the local optimal solutions of Eq. (1).

3 Proposed algorithm and analysis

In this section, we first introduce the proposed algorithm for incremental OCCF with co-evolving side networks. Then we analyze its effectiveness and efficiency.

3.1 The proposed algorithm

Given a static recommendation input $\Gamma =< \mathbf{M}, \mathbf{N}, \mathbf{R} >$, we can find its latent feature matrices $\mathbf{F}$ and $\mathbf{G}$ by solving Eq. (1) as shown in the previous section. However, in real applications, networks are evolving over time with perturbation $\Delta\Gamma =< \Delta\mathbf{M}, \Delta\mathbf{N}, \Delta\mathbf{R} >$ from the previous timestamp. Consequently, the latent feature matrices should be updated accordingly to provide a more accurate preference estimation. Additionally, as the whole systems are often changing smoothly [19]; hence, we can assume that the updated user features $\tilde{\mathbf{F}}$ and item features $\tilde{\mathbf{G}}$ still reside in the same feature space with $\mathbf{F}$ and $\mathbf{G}$, but are subtly transformed by the system perturbations. In this way, the updated feature matrices $\tilde{\mathbf{F}}$ and $\tilde{\mathbf{G}}$ can be viewed as a linear transformation from $\mathbf{F}$ and $\mathbf{G}$ as shown in the example in Fig. 1 (i.e. $\tilde{\mathbf{F}} = \mathbf{F}\mathbf{T}_F$, $\tilde{\mathbf{G}} = \mathbf{G}\mathbf{T}_G$). Therefore, Problem 1 is equivalent to finding the transformation matrices $\mathbf{T}_F$ and $\mathbf{T}_G$ for the new timestamp. Hence, the new objective function under perturbation $\Delta\Gamma$ can be written as

$$\begin{aligned} \min_{\mathbf{T}_F,\mathbf{T}_G} \ & \|\tilde{\mathbf{W}} \odot (\tilde{\mathbf{R}} - \mathbf{F}\mathbf{T}_F\mathbf{T}_G'\mathbf{G}')\|_F^2 \\ & + \alpha(\mathrm{tr}\mathbf{T}_F'\mathbf{F}'(\mathbf{D}_{\tilde{M}} - \tilde{\mathbf{M}})\mathbf{F}\mathbf{T}_F) \\ & + \alpha\mathrm{tr}(\mathbf{T}_G'\mathbf{G}'(\mathbf{D}_{\tilde{N}} - \tilde{\mathbf{N}})\mathbf{G}\mathbf{T}_G) + \beta(\|\mathbf{F}\mathbf{T}_F\|_F^2 + \|\mathbf{G}\mathbf{T}_G\|_F^2) \\ & \text{s.t.} \quad \mathbf{F}\mathbf{T}_F, \mathbf{G}\mathbf{T}_G \geq 0 \end{aligned} \tag{2}$$

Notice that the above objective function imposes a linear constraint on $\mathbf{T}_F$ and $\mathbf{T}_G$ in $\mathbf{F}\mathbf{T}_F, \mathbf{G}\mathbf{T}_G \geq 0$, which would inevitably increase the computational complexity. We propose to simplify the constraint by replacing it with a non-negative constraint on $\mathbf{T}_F$ and $\mathbf{T}_G$. As $\mathbf{F}$ and $\mathbf{G}$ are non-negative in the first place, their non-negative linear combinations $\mathbf{F}\mathbf{T}_F$ and $\mathbf{G}\mathbf{T}_G$ are guaranteed to be non-negative as well. Therefore, we can rewrite the above objective

[2] $\|\mathbf{a}\|_2$ is the L2-norm of vector $\mathbf{a}$.

function as follows

$$\min_{\mathbf{T}_F,\mathbf{T}_G\geq 0} \|\tilde{\mathbf{W}} \odot (\tilde{\mathbf{R}} - \mathbf{F}\mathbf{T}_F\mathbf{T}'_G\mathbf{G}')\|_F^2 + \alpha\mathrm{tr}(\mathbf{T}'_F\mathbf{F}'(\mathbf{D}_{\tilde{M}} - \tilde{\mathbf{M}})\mathbf{F}\mathbf{T}_F) + \alpha\mathrm{tr}(\mathbf{T}'_G\mathbf{G}'(\mathbf{D}_{\tilde{N}} - \tilde{\mathbf{N}})\mathbf{G}\mathbf{T}_G) + \beta(\|\mathbf{T}_F\|_F^2 + \|\mathbf{T}_G\|_F^2) \tag{3}$$

As the objective function in Eq. (3) is not jointly convex w.r.t. $\mathbf{T}_F$ and $\mathbf{T}_G$ due to the term $\mathbf{F}\mathbf{T}_F\mathbf{T}'_G\mathbf{G}'$, it is hard to find the global optimal solution for the problem. Instead, we seek to obtain its local optimal solution by alternatively updating $\mathbf{T}_F$ and $\mathbf{T}_G$ while fixing the other one.

When $\mathbf{T}_G$ is fixed, the objective function w.r.t. $\mathbf{T}_F$ is reduced to

$$J_{\mathbf{T}_F} = \|\tilde{\mathbf{W}} \odot (\tilde{\mathbf{R}} - \mathbf{F}\mathbf{T}_F\mathbf{T}'_G\mathbf{G}')\|_F^2 + \alpha\mathrm{tr}(\mathbf{T}'_F\mathbf{F}'(\mathbf{D}_{\tilde{M}} - \tilde{\mathbf{M}})\mathbf{F}\mathbf{T}_F) + \beta\|\mathbf{T}_F\|_F^2 \tag{4}$$

Then the derivative of $J_{\mathbf{T}_F}$ w.r.t. $\mathbf{T}_F$ is

$$\frac{1}{2}\frac{\partial J_{\mathbf{T}_F}}{\partial \mathbf{T}_F} = \mathbf{F}'(\tilde{\mathbf{W}} \odot \tilde{\mathbf{W}} \odot (\mathbf{F}\mathbf{T}_F\mathbf{T}'_G\mathbf{G}'))\mathbf{G}\mathbf{T}_G - \mathbf{F}'(\tilde{\mathbf{W}} \odot \tilde{\mathbf{W}} \odot \tilde{\mathbf{R}})\mathbf{G}\mathbf{T}_G + \alpha\mathbf{F}'(\tilde{\mathbf{D}}_{\tilde{\mathbf{M}}} - \tilde{\mathbf{M}})\mathbf{F}\mathbf{T}_F + \beta\mathbf{T}_F \tag{5}$$

Therefore, we can update $\mathbf{T}_F$ with

$$\mathbf{T}_F(i,j) = \mathbf{T}_F(i,j)\sqrt{\frac{\mathbf{X}_F(i,j)}{\mathbf{Y}_F(i,j)}} \tag{6}$$

where

$$\mathbf{X}_F = \mathbf{F}'(\tilde{\mathbf{W}} \odot \tilde{\mathbf{W}} \odot \tilde{\mathbf{R}})\mathbf{G}\mathbf{T}_G + \alpha\mathbf{F}'\tilde{\mathbf{M}}\mathbf{F}\mathbf{T}_F \tag{7}$$

$$\mathbf{Y}_F = \mathbf{F}'(\tilde{\mathbf{W}} \odot \tilde{\mathbf{W}} \odot (\mathbf{F}\mathbf{T}_F\mathbf{T}'_G\mathbf{G}'))\mathbf{G}\mathbf{T}_G + \alpha\mathbf{F}'\tilde{\mathbf{D}}_{\tilde{\mathbf{M}}}\mathbf{F}\mathbf{T}_F + \beta\mathbf{T}_F \tag{8}$$

Note that the brute force way to update $\mathbf{Y}_F$ requires to calculate a large dense matrix $\tilde{\mathbf{W}} \odot \tilde{\mathbf{W}} \odot (\mathbf{F}\mathbf{T}_F\mathbf{T}'_G\mathbf{G}')$ (i.e. $\tilde{\mathbf{W}} \odot \tilde{\mathbf{W}} \odot (\tilde{\mathbf{F}}\tilde{\mathbf{G}}')$). This step will take $O(n_u n_i r)$ which is time-consuming in large-scale systems. Recall that we have set $\tilde{\mathbf{W}}(i,j) = 1$ if $\tilde{\mathbf{R}}(i,j) = 1$ and $\tilde{\mathbf{W}}(i,j) = w$ if $\tilde{\mathbf{R}}(i,j) = 0$, then the above term can be rewritten as $(1 - w^2)\tilde{\mathbf{R}}_e + w^2\tilde{\mathbf{F}}\tilde{\mathbf{G}}'$ where $\tilde{\mathbf{R}}_e = \tilde{\mathbf{R}} \odot (\tilde{\mathbf{F}}\tilde{\mathbf{G}})$. In other words, the entries in $\tilde{\mathbf{R}}_e$ are the reconstructed preferences of observed links in the updated preference matrix $\tilde{\mathbf{R}}$, which is very sparse in real applications. Moreover, the term $\tilde{\mathbf{W}} \odot \tilde{\mathbf{W}} \odot \tilde{\mathbf{R}}$ in Eq. (7) is equivalent to $\tilde{\mathbf{R}}$ itself. Therefore, the updating rule for $\mathbf{T}_F$ can be simplified as

$$\mathbf{X}_F = (\mathbf{F}'\tilde{\mathbf{R}}\mathbf{G})\mathbf{T}_G + \alpha(\mathbf{F}'\tilde{\mathbf{M}}\mathbf{F})\mathbf{T}_F \tag{9}$$

$$\mathbf{Y}_F = (1 - w^2)\mathbf{F}'\tilde{\mathbf{R}}_e\mathbf{G}\mathbf{T}_G + w^2(\mathbf{F}'\mathbf{F})\mathbf{T}_F\mathbf{T}'_G(\mathbf{G}'\mathbf{G})\mathbf{T}_G + \alpha(\mathbf{F}'\tilde{\mathbf{D}}_{\tilde{\mathbf{M}}}\mathbf{F})\mathbf{T}_F + \beta\mathbf{T}_F \tag{10}$$

Similarly, $\mathbf{T}_G$ can be updated with

$$\mathbf{T}_G(i,j) = \mathbf{T}_G(i,j)\sqrt{\frac{\mathbf{X}_G(i,j)}{\mathbf{Y}_G(i,j)}} \tag{11}$$

where

$$\mathbf{X}_G = (\mathbf{G}'\tilde{\mathbf{R}}'\mathbf{F})\mathbf{T}_F + \alpha(\mathbf{G}'\tilde{\mathbf{N}}\mathbf{G})\mathbf{T}_G \tag{12}$$

$$\begin{aligned}\mathbf{Y}_G =& (1-w^2)\mathbf{G}'\tilde{\mathbf{R}}_e'\mathbf{F}\mathbf{T}_F + w^2(\mathbf{G}'\mathbf{G})\mathbf{T}_G\mathbf{T}_F'(\mathbf{F}'\mathbf{F})\mathbf{T}_F\\ &+ \alpha(\mathbf{G}'\tilde{\mathbf{D}}_{\tilde{\mathbf{N}}}\mathbf{G})\mathbf{T}_G + \beta\mathbf{T}_G\end{aligned} \tag{13}$$

The proposed algorithm is summarized in Alg. 1. It first specifies r, the dimension of latent features $\mathbf{F}$ and $\mathbf{G}$ in step 1, and then initializes the transformation matrices $\mathbf{T}_F$ and $\mathbf{T}_G$ randomly in step 2 and 3. From step 4, the algorithm begins to update $\mathbf{T}_F$ (step 5) and $\mathbf{T}_G$ (step 6) alternatively until convergence.

Algorithm 1 ENCORE: The Incremental OCCF Algorithm with Co-Evolving Side Networks

```
Input: (1) the original recommendation input $\Gamma =< \mathbf{M}, \mathbf{N}, \mathbf{R} >$; (2) the perturbations on the system $\Delta\Gamma =< \Delta\mathbf{M}, \Delta\mathbf{N}, \Delta\mathbf{R} >$; (3) the original latent features $\mathbf{F}$ and $\mathbf{G}$; (4) weight $w$; and (5) regularized parameters $\alpha$ and $\beta$;
Output: (1) Transformation matrix for user latent features $\mathbf{T}_F$ and (2) transformation matrix for item latent features $\mathbf{T}_G$
1: $r \leftarrow$ latent feature dimension of $\mathbf{F}$ and $\mathbf{G}$
2: initialize $\mathbf{T}_F$ as $r \times r$ non-negative random matrix
3: initialize $\mathbf{T}_G$ as $r \times r$ non-negative random matrix
4: while not converge do
5:     update $\mathbf{T}_F(i,j) \leftarrow \mathbf{T}_F(i,j)\sqrt{\frac{\mathbf{X}_F(i,j)}{\mathbf{Y}_F(i,j)}}$ as Eq. (6)
6:     update $\mathbf{T}_G(i,j) \leftarrow \mathbf{T}_G(i,j)\sqrt{\frac{\mathbf{X}_G(i,j)}{\mathbf{Y}_G(i,j)}}$ as Eq. (11)
7: end while
8: return $\mathbf{T}_F, \mathbf{T}_G$
```

3.2 Algorithm analysis

We analyze the effectiveness and efficiency of Alg. 1. In terms of the effectiveness of the algorithm, we first show that the fixed point solutions of Eq. (6) and Eq. (11) satisfy the *KKT* (Karush–Kuhn–Tucker) condition [3].

Theorem 1 *The fixed point solutions of Eq.* (6) *and Eq.* (11) *satisfy the KKT condition.*

Proof As $\mathbf{T}_F$ and $\mathbf{T}_G$ are solved in the same way, we only need to show that the fixed point solution for $\mathbf{T}_F$ in Eq. (6) satisfies the *KKT* condition, the other one can be proved in the same procedure.

First, the Lagrangian function for Eq. (4) is

$$\begin{aligned}L_{J_F} =& \|\tilde{\mathbf{W}} \odot (\tilde{\mathbf{R}} - \mathbf{F}\mathbf{T}_F\mathbf{T}_G'\mathbf{G}')\|_F^2 + \mathrm{tr}(\mathbf{T}_F'\mathbf{F}'\mathbf{D}_{\tilde{M}}\mathbf{F}\mathbf{T}_F)\\ &- \alpha\mathrm{tr}(\mathbf{T}_F'\mathbf{F}'\tilde{\mathbf{M}}\mathbf{F}\mathbf{T}_F) + \beta\|\mathbf{T}_F\|_F^2 - \mathrm{tr}(\Lambda'\mathbf{T}_F)\end{aligned} \tag{14}$$

where Λ is the Lagrange multiplier. By setting the derivative of L_{J_F} w.r.t. $\mathbf{T}_F$ to 0, we get

$$2(\mathbf{F}'(\tilde{\mathbf{W}} \odot \tilde{\mathbf{W}} \odot (\mathbf{F}\mathbf{T}_F\mathbf{T}_G'\mathbf{G}'))\mathbf{G}\mathbf{T}_G \tag{15}$$

$$- \mathbf{F}'(\tilde{\mathbf{W}} \odot \tilde{\mathbf{W}} \odot \tilde{\mathbf{R}})\mathbf{G}\mathbf{T}_G$$
$$+ \alpha \mathbf{T}_F \mathbf{F}'\tilde{\mathbf{D}}_{\tilde{M}}\mathbf{F} - \alpha \mathbf{T}_F \mathbf{F}'\tilde{\mathbf{M}}\mathbf{F} + \beta \mathbf{T}_F) = \Lambda$$

By the KKT slackness condition, we have

$$\begin{aligned}
&[-\mathbf{F}'(\tilde{\mathbf{W}} \odot \tilde{\mathbf{W}} \odot \tilde{\mathbf{R}})\mathbf{G}\mathbf{T}_G - \alpha \mathbf{T}_F \mathbf{F}'\tilde{\mathbf{M}}\mathbf{F} \\
&+ \mathbf{F}'(\tilde{\mathbf{W}} \odot \tilde{\mathbf{W}} \odot (\mathbf{F}\mathbf{T}_F\mathbf{T}_G'\mathbf{G}'))\mathbf{G}\mathbf{T}_G \\
&+ \alpha \mathbf{T}_F \mathbf{F}'\tilde{\mathbf{D}}_{\tilde{M}}\mathbf{F} + \beta \mathbf{T}_F](i,j)\mathbf{T}_F(i,j) = 0
\end{aligned} \tag{16}$$

At the fixed point of Eq. (6), we have $\mathbf{X}_F(i,j) = \mathbf{Y}_F(i,j)$, which implies that

$$\begin{aligned}
&[\mathbf{F}'(\tilde{\mathbf{W}} \odot \tilde{\mathbf{W}} \odot \tilde{\mathbf{R}})\mathbf{G}\mathbf{T}_G + \alpha \mathbf{T}_F \mathbf{F}'\tilde{\mathbf{M}}\mathbf{F}](i,j) \\
&= [\mathbf{F}'(\tilde{\mathbf{W}} \odot \tilde{\mathbf{W}} \odot (\mathbf{F}\mathbf{T}_F\mathbf{T}_G'\mathbf{G}'))\mathbf{G}\mathbf{T}_G \\
&\quad + \alpha \mathbf{T}_F \mathbf{F}'\tilde{\mathbf{D}}_{\tilde{M}}\mathbf{F} + \beta \mathbf{T}_F](i,j)
\end{aligned} \tag{17}$$

Clearly, the above equation satisfies the KKT slackness condition in Eq. (16). Therefore, the fixed point solution of Eq. (6) satisfies the KKT condition.

Theorem 1 states that the updating rules in Eq. (6) lead to a local optimal solution to Eq. (4) at convergence. Also, it can be proved that by following the updating rule in Eq. (6), the objective function in Eq. (4) decreases monotonically.

Combining Theorem 1 with the monotonic decreasing property, we can conclude that Alg. 1 converges to the local optimal solution $\mathbf{T}_F$ for the objective function in Eq. (4). Similarly, we have the local optimal solution $\mathbf{T}_G$. The two matrices $\mathbf{T}_F$ and $\mathbf{T}_G$ form the local optimal solution for Eq. (3).

For efficiency of the algorithm, we analyze the time complexity and space complexity of Alg. 1 in Lemma 1 and Lemma 2, respectively.

Lemma 1 *The time complexity of proposed algorithm is* $O((\tilde{m}_u + \tilde{m}_i)r + ((n_u + n_i + r)r^2 + \tilde{m}_r r)t)$.

Proof See "Appendix". □

Compared with the complexity of static OCCF algorithm in the previous section ($O(((m_u + m_i + m_r)r + (n_u + n_i)r^2)t)$), the proposed ENCORE is more efficient, with an $O((\tilde{m}_u + \tilde{m}_i)r)$ reduction in the time complexity in *each* iteration.

Lemma 2 *The space complexity of proposed algorithm is* $O((n_u + n_i + r)r + \tilde{m}_u + \tilde{m}_i + \tilde{m}_r)$.

Proof See "Appendix". □

3.3 Variations

In this section, we discuss some of the variant of ENCORE. First, when the weighting matrix $\mathbf{W}$ is all-one matrix, ENCORE becomes a dynamic clustering algorithm, in which $\mathbf{T}_F$ and $\mathbf{T}_G$ can be viewed as the cluster membership transition matrix. Second, when one or both side networks are missing, the corresponding regularization term would be removed from the objective function. In particular, when both sides of the networks were removed, ENCORE is reduced to an incremental algorithm for the classic one class collaborative filtering problem.

Table 2 Statistics of datasets

Dataset	Ciao	Epinions
# of users	6102	33,725
# of items	12,082	43,542
# of user links	75,861	328,455
# of items links	283,284	249,397
# of preferences	117,731	500,478
Mean degree of users	24.86	19.48
Cluster coefficient of users	0.13	0.10
Mean degree of items	23.45	11.46
Cluster coefficient of items	0.72	0.35

4 Evaluation

In this section, we evaluate the proposed ENCORE algorithm on two real datasets. The experiments are designed to answer the following two questions.

- *Effectiveness*. How effective is ENCORE for OCCF problem with co-evolving side networks?
- *Efficiency*. How fast is ENCORE compared with batch-mode static counterpart?

4.1 Experimental setup

We first introduce the datasets used, comparing methods, evaluation metrics and experimental settings before presenting the details of the experiments.

4.1.1 Datasets description

We use two real datasets Ciao [31] and Epinions [32] to evaluate the proposed ENCORE method. Ciao and Epinions are two popular online product review websites in which users are allowed to build connections and share experiences on the products with each other. To fit the one-class collaborative filtering problem, all missing links and ratings that are no greater than 3 are viewed as negative examples (i.e., labeled as 0), while ratings that are greater or equal to 4 are marked as positive examples (i.e., labeled as 1). The user side network contains the trust relations between users, while the item side network describes the similarity between items based on their reviews.[3] Both datasets have been preprocessed and used in [42] and are publicly available. The statistics of the datasets are summarized in Table 2.

In our evaluation, the datasets are partitioned into three different groups. The first group is the original training system which contains 50% of the ratings and the corresponding side network links; the second group is the incremental system, which adds 1% links to the rating matrix and the side network connections at each timestamp; the last group is the testing system, which contains the rest of the data.

[3] Similarity between items is calculated by the cosine similarity between TF-IDF (Term Frequency-Inverse Document Frequency) [27] word vectors constructed from item reviews.

4.1.2 Comparing methods

We compare ENCORE with the following baseline methods to demonstrate its effectiveness.

– **ReRun**. *ReRun* is the batch-mode static counterpart for ENCORE. At each timestamp, it takes the current system snapshot as input networks and solves the optimization problem in Eq. (1) from scratch.[4] As *ReRun* does not impose any transformation constraints on the latent features in two consecutive timestamps, it can be used to validate the effectiveness of the transformation model in ENCORE.
– **M+R**. *M+R* is a variant of ENCORE, which only contains user side network and preference matrix in the system.
– **N+R**. *N+R* is another variant of ENCORE, which only contains item side network and preference matrix.
– **R-MF**. *R-MF* is a simple method for OCCF proposed in [24], which only utilizes the preference matrix in the system for the recommendation.
– **CLiMF**. *CLiMF* is also a matrix factorization-based method proposed in [30] that is designed to improve the performance of top-k recommendations on binary relevance data.
– **R-SGD**. *R-SGD* shares the same objective function with *R-MF*. Instead of calculating the latent features at each timestamp, *R-SGD* modifies related latent features with newly emerged ratings by stochastic gradient descent method.
– **eNMF**. *eNMF* is an incremental matrix factorization algorithm proposed in [38]. Here we apply this algorithm to perform incremental update on the factorization results of the preference matrix.

4.1.3 Evaluation metrics

In our experiments, we assess the effectiveness of ENCORE with *MAP* and *R-MPR* as evaluation metrics.

– **MAP**. *MAP* (Mean Average Precision) is originally used to evaluate ranked documents over a set of queries. Here it computes the mean average precision over all users in the test set [24]. The larger the *MAP* is, the better the performance is.
– **R-MPR**. *R-MPR* (Reverse Mean Percentage Ranking) is a variation of *MPR*, which is originally used to evaluate users' satisfaction of items by a ranked list. A randomly generated item list can achieve a *MPR* of 50% [22]. The smaller the *MPR* is, the better the performance is. Here we set *R-MPR* to be 0.5-*MPR*, thus a larger *R-MPR* indicates a better performance.

4.1.4 Machine

The experiments are performed on a machine with 2 Intel Xeon 3.5 GHz processors and 256 GB of RAM. The algorithms are implemented with MATLAB using a single thread. We will release the code when the paper is published.

[4] Eq. (1) is derived from wiZAN-Dual in [42]. Therefore, ReRun is equivalent to wiZan-Dual here.

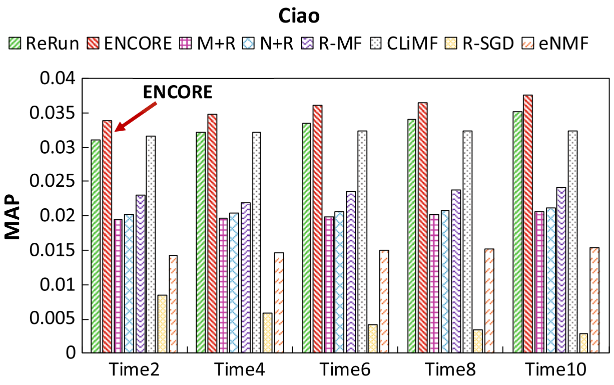


Fig. 2 MAP comparison on Ciao. Higher is better (best viewed in color)

4.2 Effectiveness results

We compare the proposed algorithm with other methods on both Ciao and Epinions datasets. The results are shown from Figs. 2, 3, 4 and 5. We make the following observations from these two figures.

- In both datasets, ENCORE achieves close performance with *ReRun*. Such results demonstrate that ENCORE can effectively accommodate newly emerged links in the dynamic system for the recommendation. It should be noted that here our goal is not to develop a better recommendation algorithm that outperforms *ReRun*. Instead, our goal is to ensure that the incrementally updated model with linear transformation at each time stamp can closely approximate the performance of *ReRun*.
- Side networks between users and items are both important for improving the quality of recommendation results. As we can see from Figs. 2, 3, 4 and 5, user network or item network alone with the preference matrix cannot boost the recommendation quality compared with the methods that use preference matrix only. However, when both networks are added to the model, the performance can be improved significantly.
- For the two incremental update algorithms (ENCORE and eNMF), our proposed algorithm ENCORE is consistently better than eNMF due to its effectiveness on exploiting the information from side networks.
- The CLiMF method is designed to maximize the Mean Reciprocal Rank (MRR) for top-k recommendations. Therefore, its MAP score is comparable to ENCORE on both datasets as in Figs. 2 and 4. However, such scheme would ignore the recall on the recommended items, which results in lower R-MPR scores as indicated in Figs. 3 and 5.

4.3 Efficiency results

Next, we evaluate the efficiency of ENCORE on both Ciao and Epinions. As the results are similar, we only report the one on Ciao for brevity. In the experiment, we have varied the dimension of latent features r with the same input data and evaluated the running time ENCORE under different settings. It can be seen from Fig. 6 that the average running time of ENCORE for a single iteration is shorter than the *ReRun* method. Specifically, as the latent feature dimension r increases, the speed-up of ENCORE compared to *ReRun* becomes larger accordingly. This observation is consistent with our time complexity analysis in the

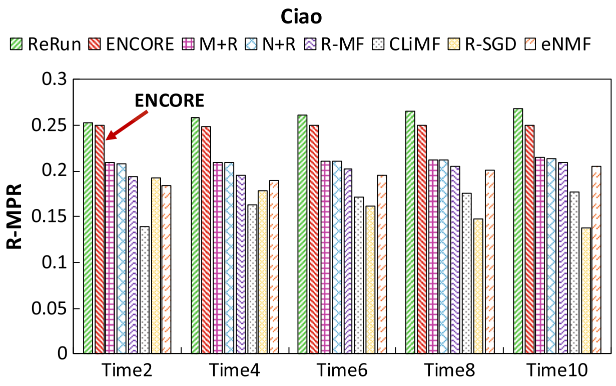


Fig. 3 R-MPR comparison on Ciao. Higher is better (best viewed in color)

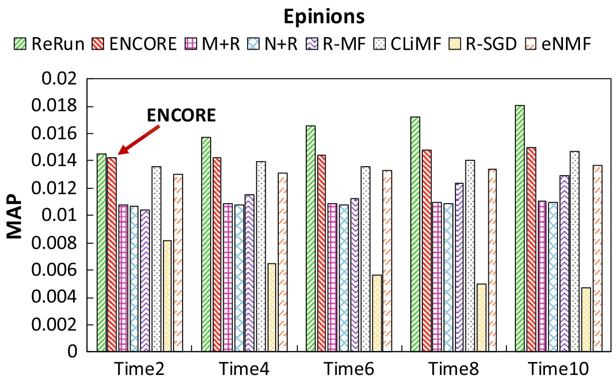


Fig. 4 MAP comparison on Epinions. Higher is better (best viewed in color)

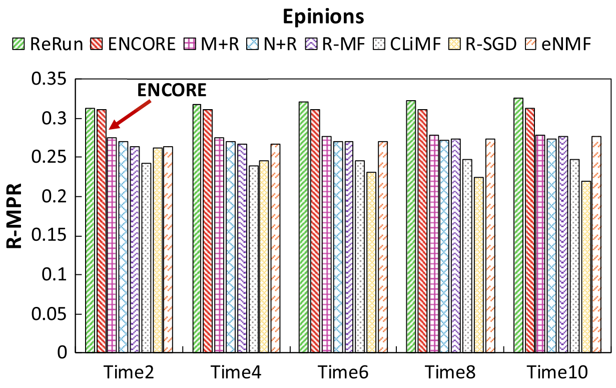


Fig. 5 R-MPR comparison on Epinions. Higher is better (best viewed in color)

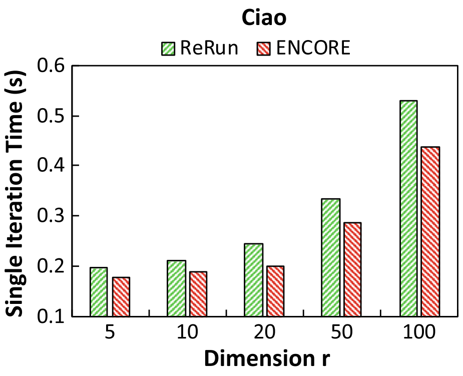


Fig. 6 The running time of *ReRun* versus ENCORE for a single iteration on the Ciao dataset

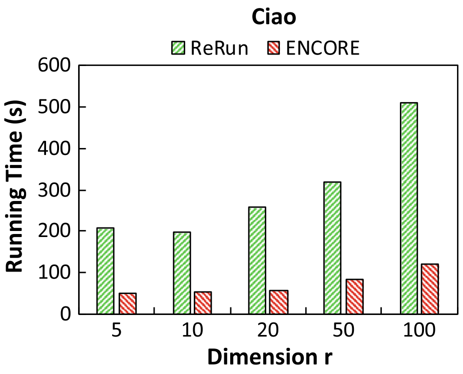


Fig. 7 The running time of *ReRun* versus ENCORE for one timestamp (many iterations until convergence) on the Ciao dataset

previous section that the proposed ENCORE algorithm has an $O(r)$ factor speed-up over its static counterpart (*ReRun*). Moreover, as shown in Fig. 7, the average running time of ENCORE for one timestamp is much shorter than *ReRun* (with around 75% improvement). This is mainly due to the fact that ENCORE has much fewer variables to optimize at each timestamp compared to *ReRun* (i.e., $2r \times r$ vs. $(m+n) \times r$), which makes it converge faster in a small number of iterations.

5 Related work

We review the related work from two parts: (1) collaborative filtering; and (2) dynamic network analysis.

5.1 Collaborative filtering

Collaborative filtering is originally designed to provide recommendations to users where their preferences to items are explicitly given in a multi-scale (e.g., score 1–5 ratings). Generally speaking, existing collaborative filtering methods can be classified into three categories, including memory-based methods [9], model-based methods [10] and the mixture of the above two methods [14].

Despite the success of collaborative filtering in a large portion of explicit, multi-scale recommender systems, most real-world applications only have binary preference classes where users' preferences are implicitly expressed (e.g., click of webpages, browse of news and purchase of products). In response, one-class collaborative filtering has received a surge of research interests in recent years. Existing one-class collaborative filtering (OCCF) algorithms can be broadly classified into two categories: (1) pointwise regression methods; and (2) pairwise ranking methods. First, pointwise regression methods attempt to learn latent factors for users and items to approximate the original ratings by minimizing a well-designed loss function. The very first few attempts are credited to [11,24]. On top of that, Li et al. [22] proposed to incorporate rich user side information to improve OCCF. Yao et al. [42] further extended to propose a novel approach wiZAN-Dual by exploiting the user–user similarity and item–item similarity. The proposed dual-regularized OCCF framework can also be extended in a multi-layered scenario [5]. On the other hand, pairwise ranking methods such as BPR [28] resorts to maximizing the pairwise ranking results on observed and unobserved user actions. Akin to pointwise-based methods, there are some attempts to consider side information such as social relationships for ranking [25,43] and recommendation [13,39]. A more detailed review of social recommendation can be referred to [33].

5.2 Dynamic network analysis

Many real networks are constantly evolving over time. In such dynamic systems, we not only know structure of the network but also have the establishing time information associated to the edges. Such temporal information has been extensively explored to improve the recommendation algorithms on static systems [7,15,26,29,40,41] and is often well-known as temporal/sequential recommendation. However, it should be noted our proposed framework using incremental matrix factorization is orthogonal to the aforementioned works as our goal is not to improve the performance of recommendation with temporal information, but to develop an efficient solution that can be used to maintain the freshness of the system from previous timestamp.

Similarly, as the network structures of many real-world networks are continuously changing, some key network properties and learning models trained from historical data would become stale over time and need to be updated to reflect such dynamics. Many research efforts have been devoted along this line. For instance, Tong et al. [36] provided an efficient algorithm to dynamically track node proximity and centrality on bipartite graphs. Chen et al. [4] tracked the eigen-systems of dynamic networks which is important to many graph mining problems. In addition, various dynamic algorithms are proposed for different applications, such as low-rank approximation [6,35], node classification [2], community detection [34], feature selection [21], network embedding [20] and incremental matrix factorization [12,37,38]. Another line of work focuses on monitoring the properties of dynamic networks. Leskovec et al. [19] discovered that dynamic networks densify over time, and the network diameter shrinks. While in [18], they studied dynamic networks from a microscopic perspective and developed a network model to simulate the evolution process. A more detailed review of dynamic network analysis can be found in [1].

6 Conclusions

In this paper, we propose a novel algorithm (ENCORE) to address the one-class collaborative filtering problem with co-evolving networks. To effectively model the evolution of latent features, we propose to solve the OCCF problem incrementally since most recommender systems are changing in a smooth way. In detail, by assuming that the updated latent features and the original features reside in the same latent feature space, we model the updated latent features of users and items as linear transformed vectors from the original features. The experimental results on two real datasets show that the proposed method can achieve similar recommendation quality to the batch-mode static OCCF method while taking much less running time.

Acknowledgements This work is supported by National Science Foundation under Grant Nos. 1947135, and 2003924 by the NSF Program on Fairness in AI in collaboration with Amazon under Award No. 1939725. The content of the information in this document does not necessarily reflect the position or the policy of the Government or Amazon, and no official endorsement should be inferred. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

Appendix

6.1 Proof for Lemma 1

Proof In Alg. 1, as term $\tilde{\mathbf{M}}$, $\tilde{\mathbf{N}}$, $\tilde{\mathbf{R}}$, $\mathbf{F}$ and $\mathbf{G}$ remain the same during the iterations, we can pre-compute related constant terms to avoid redundant computations. The complexities of computing constant terms in Eqs. (9)–(13) are $O(n_i r^2 + \tilde{m}_r r)$ for $\mathbf{F}'\tilde{\mathbf{R}}\mathbf{G}$; $O(n_u r^2 + \tilde{m}_u r)$ for $\mathbf{F}'\tilde{\mathbf{M}}\mathbf{F}$; $O(n_u r^2)$ for $\mathbf{F}'\mathbf{D}_{\tilde{M}}\mathbf{F}$; $O(n_u r^2)$ for $\mathbf{F}'\mathbf{F}$; $O(n_i r^2)$ for $\mathbf{G}'\mathbf{G}$; $O(n_i r^2 + \tilde{m}_i r)$ for $\mathbf{G}'\tilde{\mathbf{N}}\mathbf{G}$; and $O(n_i r^2)$ for $\mathbf{G}'\mathbf{D}_{\tilde{N}}\mathbf{G}$. Thus, the complexity for pre-computing is $O((n_u + n_i)r^2 + (\tilde{m}_u + \tilde{m}_i + \tilde{m}_r)r)$. In each iteration, it takes $O(n_u r^2)$ and $O(n_i r^2)$ to compute $\mathbf{F}\mathbf{T}_F$ (i.e., $\tilde{\mathbf{F}}$) and $\mathbf{G}\mathbf{T}_G$ (i.e., $\tilde{\mathbf{G}}$), respectively. The complexity of computing $\mathbf{F}'\tilde{\mathbf{R}}_e\mathbf{G}\mathbf{T}_G$ is $O(n_i r^2 + \tilde{m}_r r)$, the rest of the computations for updating $\mathbf{T}_F$ and $\mathbf{T}_G$ are both $O(r^3)$. Therefore, the overall complexity for Alg. 1 is $O((\tilde{m}_u + \tilde{m}_i)r + ((n_u + n_i + r)r^2 + \tilde{m}_r r)t)$, where t is the number of iterations in the algorithm. □

6.2 Proof for Lemma 2

Proof The algorithm requires a space of $O(n_u r + n_i r)$ to store $\mathbf{F}$ and $\mathbf{G}$, $O(r^2)$ to store the transformation matrices $\mathbf{T}_F$ and $\mathbf{T}_G$, and $O(\tilde{m}_u + \tilde{m}_i + \tilde{m}_r)$ to store the updated rating matrix and side networks. The space needed to compute and store the constant terms are $O(n_i r + r^2)$ for $\mathbf{F}'\tilde{\mathbf{R}}\mathbf{G}$; $O(n_u r + r^2)$ for $\mathbf{F}'\tilde{\mathbf{M}}\mathbf{F}$ and $\mathbf{F}'\mathbf{D}_{\tilde{M}}\mathbf{F}$; $O(r^2)$ for $\mathbf{F}'\mathbf{F}$ and $\mathbf{G}'\mathbf{G}$; $O(n_i r + r^2)$ for $\mathbf{G}'\tilde{\mathbf{N}}\mathbf{G}$ and $\mathbf{G}'\mathbf{D}_{\tilde{N}}\mathbf{G}$, respectively. Therefore, the space costs for computing constant terms are $O((n_u + n_i)r + r^2)$. In each iteration, it takes a space of $O((n_u + n_i)r)$ to compute $\tilde{\mathbf{F}}$ and $\tilde{\mathbf{G}}$, $O(\tilde{m}_r)$ to compute $\tilde{\mathbf{R}}_e$, $O(n_i r + r^2)$ to compute $\mathbf{F}'\tilde{\mathbf{R}}_e\mathbf{G}\mathbf{T}_G$, $O(r^2)$ for the rest of the matrix multiplications to update $\mathbf{T}_F$ and $\mathbf{T}_G$. Putting all these terms together, the overall space complexity for Alg. 1 is $O((n_u + n_i + r)r + \tilde{m}_u + \tilde{m}_i + \tilde{m}_r)$. □

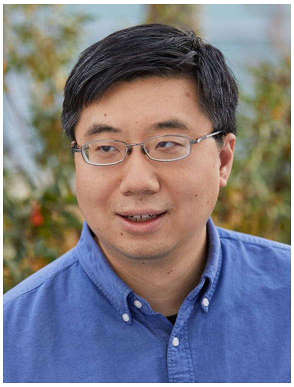
References

1. Aggarwal C, Subbian K (2014) Evolutionary network analysis: a survey. ACM Comput Surv 47(1):10
2. Aggarwal, CC, Li N (2011) On node classification in dynamic content-based networks. In: Proceedings of the 2011 SIAM international conference on data mining, pp 355–366
3. Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge University Press, Cambridge
4. Chen C, Tong H (2015) Fast eigen-functions tracking on dynamic graphs. In: Proceedings of the 2015 SIAM international conference on data mining, pp 559–567

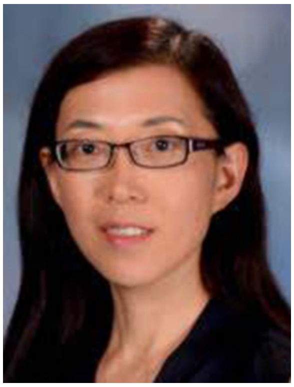
5. Chen C, Tong H, Xie L, Ying L, He Q (2016) Fascinate: fast cross-layer dependency inference on multi-layered networks. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp 765–774
6. Chen X, Candan KS (2014) LWI-SVD: low-rank, windowed, incremental singular value decompositions on time-evolving data sets. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 987–996
7. Ding Y, Li X (2005) Time weight collaborative filtering. In: Proceedings of the 14th ACM international conference on information and knowledge management, pp 485–492
8. Field DJ (1994) What is the goal of sensory coding? Neural Comput 6(4):559–601
9. Herlocker JL, Konstan JA, Borchers A, Riedl J (1999) An algorithmic framework for performing collaborative filtering. In: Proceedings of the 22nd annual international ACM SIGIR conference on research and development in information retrieval, pp 230–237
10. Hofmann T (2004) Latent semantic models for collaborative filtering. ACM Trans Inf Syst 22(1):89–115
11. Hu Y, Koren Y, Volinsky C (2008) Collaborative filtering for implicit feedback datasets. In: Proceedings of the 8th IEEE international conference on data mining, pp 263–272
12. Huang X, Wu L, Chen E, Zhu H, Liu Q, Wang Y, Center BTI (2017) Incremental matrix factorization: a linear feature transformation perspective. In: Proceedings of the 26th international joint conference on artificial intelligence, pp 1901–1908
13. Jiang M, Cui P, Wang F, Zhu W, Yang S (2014) Scalable recommendation with social contextual information. IEEE Trans Knowl Data Eng 26(11):2789–2802
14. Koren Y (2008) Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, pp 426–434
15. Koren Y (2009) Collaborative filtering with temporal dynamics. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining, pp 447–456
16. Lee DD, Seung HS (1999) Learning the parts of objects by non-negative matrix factorization. Nature 401(6755):788–791
17. Lee DD, Seung HS (2001) Algorithms for non-negative matrix factorization. In: Advances in neural information processing systems, pp 556–562
18. Leskovec J, Backstrom L, Kumar R, Tomkins A (2008) Microscopic evolution of social networks. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, pp 462–470
19. Leskovec J, Kleinberg J, Faloutsos C (2005) Graphs over time: densification laws, shrinking diameters and possible explanations. In: Proceedings of the 11th ACM SIGKDD international conference on knowledge discovery in data mining, pp 177–187
20. Li J, Dani H, Hu X, Tang J, Chang Y, Liu H (2017) Attributed network embedding for learning in a dynamic environment. In: Proceedings of the 26th ACM international conference on conference on information and knowledge management
21. Li J, Hu X, Jian L, Liu H (2016) Toward time-evolving feature selection on dynamic networks. In: Proceedings of the 2016 IEEE international conference on data mining, pp 1003–1008
22. Li Y, Hu J, Zhai C, Chen Y (2010) Improving one-class collaborative filtering by incorporating rich user information. In: Proceedings of the 19th ACM international conference on information and knowledge management, pp 959–968
23. Ma H, Zhou D, Liu C, Lyu MR, King I (2011) Recommender systems with social regularization. In: Proceedings of the 4th ACM international conference on web search and data mining, pp 287–296
24. Pan R, Zhou Y, Cao B, Liu NN, Lukose R, Scholz M, Yang Q (2008) One-class collaborative filtering. In: Proceedings of the 8th IEEE international conference on data mining, pp 502–511
25. Pan W, Chen L (2013) Gbpr: group preference based bayesian personalized ranking for one-class collaborative filtering. In: Proceedings of the 23rd international joint conference on artificial intelligence, vol 13, pp 2691–2697
26. Qin J, Ren K, Fang Y, Zhang W, Yu Y (2020) Sequential recommendation with dual side neighbor-based collaborative relation modeling. In: Proceedings of the 13th international conference on web search and data mining, pp 465–473
27. Rajaraman A, Ullman JD (2011) Mining of massive datasets. Cambridge University Press, Cambridge
28. Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L (2009) Bpr: Bayesian personalized ranking from implicit feedback. In: Proceedings of the 25th conference on uncertainty in artificial intelligence, pp 452–461
29. Rendle S, Freudenthaler C, Schmidt-Thieme L (2010) Factorizing personalized markov chains for next-basket recommendation. In: Proceedings of the 19th international conference on world wide web, pp 811–820

30. Shi Y, Karatzoglou A, Baltrunas L, Larson M, Oliver N, Hanjalic A (2012) Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering. In: Proceedings of the sixth ACM conference on recommender systems, pp 139–146
31. Tang J, Gao H, Liu H (2012) mtrust: discerning multi-faceted trust in a connected world. In: Proceedings of the 5th ACM international conference on web search and data mining, pp 93–102
32. Tang J, Gao H, Liu H, Das Sarma A (2012) etrust: understanding trust evolution in an online world. In: Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining, pp 253–261
33. Tang J, Hu X, Liu H (2013) Social recommendation: a review. Soc Netw Anal Min 3(4):1113–1133
34. Tang L, Liu H, Zhang J, Nazeri Z (2008) Community evolution in dynamic multi-mode networks. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, pp 677–685
35. Tong H, Papadimitriou S, Sun J, Yu PS, Faloutsos C (2008) Colibri: fast mining of large static and dynamic graphs. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, pp 686–694
36. Tong H, Papadimitriou S, Yu PS, Faloutsos C (2008) Fast monitoring proximity and centrality on time-evolving bipartite graphs. Stat Anal Data Min 1(3):142–156
37. Vinagre J, Jorge AM, Gama J (2014) Fast incremental matrix factorization for recommendation with positive-only feedback. In: International conference on user modeling, adaptation, and personalization. Springer, pp 459–470
38. Wang F, Tong H, Lin CY (2011) Towards evolutionary nonnegative matrix factorization. In: Twenty-fifth AAAI conference on artificial intelligence
39. Wang X, Lu W, Ester M, Wang C, Chen C (2016) Social recommendation with strong and weak ties. In: Proceedings of the 25th ACM international on conference on information and knowledge management, pp 5–14
40. Wu L, Ge Y, Liu Q, Chen E, Hong R, Du J, Wang M (2017) Modeling the evolution of users preferences and social links in social networking services. IEEE Trans Knowl Data Eng 29(6):1240–1253
41. Xiong L, Chen X, Huang TK, Schneider J, Carbonell JG (2010) Temporal collaborative filtering with bayesian probabilistic tensor factorization. In: Proceedings of the 2010 SIAM international conference on data mining. SIAM, pp 211–222
42. Yao Y, Tong H, Yan G, Xu F, Zhang X, Szymanski BK, Lu J (2014) Dual-regularized one-class collaborative filtering. In: Proceedings of the 23rd ACM international conference on conference on information and knowledge management, pp 759–768
43. Zhao T, McAuley J, King I (2014) Leveraging social connections to improve personalized ranking for collaborative filtering. In: Proceedings of the 23rd ACM international conference on conference on information and knowledge management, pp 261–270

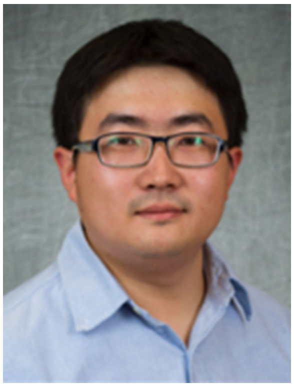
Chen Chen is currently a software engineer at Google. Before joining Google, Chen got her Ph.D. degree from Arizona State University. Her research has been focusing on the connectivity of complex networks, which has been applied to address pressing challenges in various high-impact domains, including social media, bioinformatics, recommendation, and critical infrastructure systems. Her research has appeared in top-tier conferences (including KDD, ICDM, SDM, and DASFAA), and prestigious journals (including IEEE TKDE, ACM TKDD, and SIAM SAM). Chen has received several awards, including "Bests of KDD", "Bests of SDM" and Rising Star in EECS.



Yinglong Xia is an Applied Research Scientist in Facebook, working on Graph Learning platforms and applications. Prior to that, he was a chief architect in Futurewei Technologies on Cloud AI platforms and a research staff member at IBM Research on graph database and reasoning frameworks. He received his PhD at USC in 2010 and MS from Tsinghua University in 2006. He has published 70+ technical papers and filed 30+ patents. He is an associate editor of IEEE trans. on Knowledge and Data Engineering (TKDE), and IEEE trans. on Big Data (TBD); he was a general co-chair of IEEE HiPC'19, a vice co-chair of IEEE BigData'19, a SPC/TPC member of IJCAI'20, KDD'20, CIKM'20, VLDB'20, and ICDE'20, etc.



Hui Zang is currently a tech lead and software engineer at Google. Previously, she was a distinguished data scientist at Futurewei Technologies, a lead data scientist at Guavus Inc., and a research scientist at Sprint Labs. She received her B.S. degree in computer science from Tsinghua University, China, and the M.S. and Ph.D. degrees in computer science from the University of California, Davis. Her research has focused on applying AI and machine learning techniques to build data-driven products and to optimize the performance of computing systems. She is the author of the book "WDM Mesh Networks - Management and Survivability" (Kluwer Academic, 2002). She has published over seventy conference papers and journal articles and currently has over thirty US patents granted and many more pending. Dr. Zang is a senior member of IEEE.



Jundong Li is an assistant professor of the Department of Electrical and Computer Engineering at the University of Virginia, with a joint appointment in the Department of Computer Science and the School of Data Science. He received his Ph.D. degree in Computer Science at Arizona State University in 2019, M.Sc. degree from the Department of Computing Science at the University of Alberta in 2014, and B.Eng. degree from the College of Computer Science and Technology at Zhejiang University in 2012. His research interests include data mining, machine learning, and social computing.

Huan Liu is a professor of Computer Science and Engineering at Arizona State University. He obtained his Ph.D. in Computer Science at University of Southern California and B.Eng. in Computer Science and Electrical Engineering at Shanghai JiaoTong University. Before he joined ASU, he worked at Telecom Australia Research Labs and was on the faculty at National University of Singapore. At Arizona State University, he was recognized for excellence in teaching and research in Computer Science and Engineering and received the 2014 President's Award for Innovation. His research interests are in data mining, machine learning, social computing, and artificial intelligence, investigating interdisciplinary problems that arise in many real-world, data-intensive applications with high-dimensional data of disparate forms such as social media. His well-cited publications include books, book chapters, encyclopedia entries as well as conference and journal papers. He is a co-author of a text, Social Media Mining: An Introduction, Cambridge University Press. He is a founding organizer of the International Conference Series on Social Computing, Behavioral-Cultural Modeling, and Prediction, and Field Chief Editor of Frontiers in Big Data and its Specialty Chief Editor of Data Mining and Management. He is a Fellow of ACM, AAAI, AAAS, and IEEE. More can be found at http://www.public.asu.edu/~huanliu.

Hanghang Tong is currently an associate professor at Department of Computer Science at University of Illinois at Urbana-Champaign. Before that he was an associate professor at School of Computing, Informatics, and Decision Systems Engineering (CIDSE), Arizona State University. He received his M.Sc. and Ph.D. degrees from Carnegie Mellon University in 2008 and 2009, both in Machine Learning. His research interest is in large scale data mining for graphs and multimedia. He has received several awards, including SDM/IBM Early Career Data Mining Research award (2018), NSF CAREER award (2017), ICDM 10- Year Highest Impact Paper award (2015), four best paper awards (TUP'14, CIKM'12, SDM'08, ICDM'06), seven 'bests of conference', 1 best demo, honorable mention (SIGMOD'17), and 1 best demo candidate, second place (CIKM'17). He has published over 100 refereed articles. He is the Editor-in-Chief of SIGKDD Explorations (ACM), an action editor of Data Mining and Knowledge Discovery (Springer), and an associate editor of Knowledge and Information Systems (Springer) and Neurocomputing Journal (Elsevier); and has served as a program committee member in multiple data mining, database and artificial intelligence venues (e.g., SIGKDD, SIGMOD, AAAI, WWW, CIKM, etc.).

Affiliations

Chen Chen[1] · Yinglong Xia[2] · Hui Zang[1] · Jundong Li[3] · Huan Liu[4] · Hanghang Tong[5]

✉ Chen Chen
chenannie45@gmail.com

Yinglong Xia
yxia@fb.com

Hui Zang
huizang@gmail.com

Jundong Li
jl6qk@virginia.edu

Huan Liu
huanliu@asu.edu

Hanghang Tong
htong@illinois.edu

[1] Google, Inc., Mountain View, USA

[2] Facebook, Inc., Menlo Park, USA

[3] University of Virginia, Charlottesville, USA

[4] Arizona State University, Tempe, USA

[5] University of Illinois at Urbana-Champaign, Champaign, USA