Incremental Matrix Factorization: A Linear Feature Transformation Perspective

Xunpeng Huang[†], **Le Wu**[‡], **Enhong Chen**[†*], **Hengshu Zhu**[♮], **Qi Liu**[†], **Yijun Wang**[†]
[†]Anhui Province Key Lab. of Big Data Analysis and Application, University of S&T of China
[‡]School of Computer and Information, HeFei University of Technology
[♮]Baidu Talent Intelligence Center
hxpsola@mail.ustc.edu.cn, lewu@hfut.edu.cn, cheneh@ustc.edu.cn,
zhuhengshu@baidu.com, qiliuql@ustc.edu.cn, wyjun@mail.ustc.edu.cn

Abstract

Matrix Factorization (MF) is among the most widely used techniques for collaborative filtering based recommendation. Along this line, a critical demand is to incrementally refine the MF models when new ratings come in an online scenario. However, most of existing incremental MF algorithms are limited by specific MF models or strict use restrictions. In this paper, we propose a general incremental MF framework by designing a linear transformation of user and item latent vectors over time. This framework shows a relatively high accuracy with a computation and space efficient training process in an online scenario. Meanwhile, we explain the framework with a low-rank approximation perspective, and give an upper bound on the training error when this framework is used for incremental learning in some special cases. Finally, extensive experimental results on two real-world datasets clearly validate the effectiveness, efficiency and storage performance of the proposed framework.

1 Introduction

Recent years, Matrix Factorization (MF) models have been successfully applied to various real-world recommender systems [Liu *et al.*, 2011; Wu *et al.*, 2012; Zhu *et al.*, 2015; Lin *et al.*, 2017; Wu *et al.*, 2017], due to the advantages of high accuracy, low computational cost and easy implementation [Salakhutdinov and Mnih, 2008; Bach *et al.*, 2008; Hazan, 2008; Zeng *et al.*, 2015; Chen *et al.*, 2016]. Indeed, most of MF models are batch-based, which usually have intensive computational cost when updating the latent feature vectors after the training phase [Rendle and Schmidt-Thieme, 2008; Luo *et al.*, 2012]. Therefore, a critical demand is to incrementally refine the MF models with new ratings for real-world online scenarios.

In the literature, to facilitate the computational overhead of batch-based MF models, a number of incremental MF studies have been made. However, there are still some major challenges to be addressed. First, most of the algorithms for incremental MF are not general, which are limited by specific batch-based MF models [Wang *et al.*, 2011; Luo *et al.*, 2012; Vinagre *et al.*, 2014], such as Non-negative Matrix Factorization (NMF) [Lee and Seung, 1999] or Probabilistic Matrix Factorization (PMF) [Salakhutdinov and Mnih, 2008], or some strict use restrictions, such as a pre-processed matrix without any missing elements [Brand, 2003; Sarwar *et al.*, 2002], the small-scale incremental data batch [Rendle and Schmidt-Thieme, 2008; Wang *et al.*, 2011; Luo *et al.*, 2012; Devooght *et al.*, 2015; Matuszyk *et al.*, 2015] and the scenarios of the new user/item problem [Song *et al.*, 2015]. Second, most existing incremental MF models have to store coming ratings nearly as much as that in batch-based MF models [Luo *et al.*, 2012; Devooght *et al.*, 2015], which is not truly incremental from a data storage perspective. Moreover, as most incremental MF models are non-convex and complex, the error bounds of them are hardly theoretical analyzed. To the best of our knowledge, there still lacks a holistic view for addressing all the above challenges at the same time.

To fill the above void, in this paper, we propose a general Linear Feature Transformation (LFT) framework for incremental MF. Specifically, the framework only needs to train a small-sized feature transformation matrix for capturing the changes of model parameters when new data stream comes. Furthermore, only a small constructed collection of samples needs to be saved to train the feature transformation matrix in our framework, and most coming ratings will be dropped in short periods of time. In particular, to demonstrate that the proposed framework is generally applicable for incrementally updating most of MF models, we select a representative batch-based MF model (i.e., a constrained MF model), and detail how to incrementally update it under the proposed LFT framework. We further theoretically explain the framework and analyze the training error bound from a low-rank approximation perspective. To be specific, the contributions of this paper can be summarized as follows:

- We propose a general LFT framework of incremental M-F for efficient online recommendation, which is generally applicable for incrementally updating most of MF models. In particular, we develop a concrete LFT implementation for a constrained MF model.
- We theoretically explain our proposed framework from a low-rank approximation perspective. In the case of using the framework for MF models with no constraints and

[*]Corresponding author.

no regularization terms, we prove the upper bound of the training error expectation in the incremental phase.

- We conduct extensive experiments on two real-world datasets. The results clearly validate the effectiveness, efficiency and storage performance of our concrete LFT implementation comparing with other state-of-the-art incremental MF models.

2 Related Work

In this section, we illustrate some existing incremental models, which are often designed for updating latent feature matrices in an online recommendation scenario. Up to now, incremental MF models can be mainly divided into the following three categories from a technical point of view.

SVD-based Model. This kind of incremental MF models is characterized by using SVD as its batch-based model (eg., [Brand, 2003; Sarwar *et al.*, 2002]). Incremental updating operations through the basic algebraic properties of SVD such as one-rank operations [Brand, 2003] leads to a high efficiency. However, SVD requires a matrix that has no missing elements to be decomposed. Therefore the performance of these models on the sparse datasets is limited.

Vector-retraining Model. Unlike the SVD-based Model, there are different kinds of batch-based MF approaches which can be applied into vector-retraining models [Rendle and Schmidt-Thieme, 2008; Devooght *et al.*, 2015]. Particularly, the key idea of this models is that, when a new rating arrives, the latent feature vectors used to estimate the rating need to be slightly updated [Vinagre *et al.*, 2014] or retrained [Rendle and Schmidt-Thieme, 2008; Luo *et al.*, 2012; Devooght *et al.*, 2015; Matuszyk *et al.*, 2015] in order to have a better performance in later time periods. Although some strategies to improve efficiency [Rendle and Schmidt-Thieme, 2008; Matuszyk *et al.*, 2015] and accuracy [Luo *et al.*, 2012] are introduced, this kind of models can only update coming ratings one by one. Besides, when the data comes incrementally, discarding some of the existing data will reduce the average number of elements corresponding to each user (item), which brings a quite negative impact on the effectiveness for such models. As a result, vector-retraining models have limited efficiency when large-scale incremental ratings arrive at the same time and relatively high storage consumption for saving ratings (in Section 4.2).

Space-retraining Model. In order to rule out the redundant calculation of the overhead which is caused by the "One by One" updating in the vector-retraining models, some space-retraining models are proposed. [Wang *et al.*, 2011] assumes the smooth evolution of the latent feature matrix, and optimizes the increment of the latent feature matrix by minimizing the upper bound of the loss function with a bunch of incremental ratings. However, the assumption is only established when the F-norm of the incremental rating matrix is small. Another space-retraining model [Song *et al.*, 2015] retrains the feature space via auxiliary feature learning and matrix sketching strategies [Liberty, 2013], while it is only designed for solving the new user/item problem.

Compared to vector-retraining models, the matrix we trained in this paper is to reflect the linear change of latent feature vectors over time, rather than latent feature vectors themselves. Therefore, algorithms based LFT framework have better time-space complexity when the incremental data scale reaches a certain size (in Section 4.2). Besides, the different assumption from [Wang *et al.*, 2011] makes the feasibility of our approach unaffected by the large-scale incremental rating matrix and the new user/item problem. Moreover, we illustrate the reasonableness of our assumption from a low rank approximation perspective and guarantee the quality of the solution when batch-based MF problems are formulated with no constraints and no regularization terms (in Section 5).

Table 1: Several important mathematical notations.

Notations	Description
$X_{i,:}$	the row vector in the i-th row of the matrix X
$X_{:,j}$	the column vector in the j-th column of the matrix X
Ω_X	the set of coordinates of the ratings that can be observed in matrix X
I_Ω	a binary matrix whose element is 1 if the corresponding coordinate is in the set Ω
$X_{\|t}$	the matrix X in the time period t
$\odot$	Hadamard product ($A_{i,j} = B_{i,j} \times C_{i,j},\ if\ A = B \odot C$)
K	the rank of latent feature matrices
η	the step size during iterations
n	the number of users
m	the number of items
$P \in \mathbb{R}^{n\times K}$	the latent feature matrix of users
$Q \in \mathbb{R}^{m\times K}$	the latent feature matrix of items
$R \in \mathbb{R}^{n\times m}$	the rating matrix

3 The Proposed LFT Framework

In this section, we introduce the overview of our incremental MF framework LFT (**L**inear **F**eature **T**ransformation). For better illustration, we first list some important notations and their descriptions in Table 1. Moreover, for a matrix A, we set $Sy(A) = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}$, where the 0 means the missing ratings rather than the exact values.

Indeed, most of incremental MF methods are made up of two steps. First, a batch-based MF method is used to train the initial latent feature matrices. Second, an incremental training process is developed for updating latent feature matrices when a bunch of new ratings come. As designing more sophisticated batch-based MF models is not the focus of our paper, here we pay close attention to the second step, and design an incremental MF framework with a linear transformation assumption to describe latent feature changes. Then we introduce the problem formulation, followed by the proposed LFT framework.

In a recommender system, there are n users and m items. In each time period $t+1$, incremental ratings are recorded as $\Delta R_{|t+1} \in \mathbb{R}^{n\times m}$. And $\Delta Y_{|t+1} = [\Delta P_{|t+1}, \Delta Q_{|t+1}] \in \mathbb{R}^{(n+m)\times K}$ is the latent feature matrix decomposed by the matrix $\Delta X_{|t+1} = Sy(\Delta R_{|t+1})$ with the batch-based MF model G (e.g., PMF or NMF). We concatenate all the ratings till time t as $R_{|t} \in \mathbb{R}^{n\times m}$ estimated by $P_{|t}Q_{|t}^T$, where matrix $Y_{|t} = [P_{|t}, Q_{|t}]$ has been learned before. Here, the time period is application-depended such as an hour, a day or a week. Given the above definition, the incremental MF problem turns into: *how to efficiently learn and update the new feature ma-*

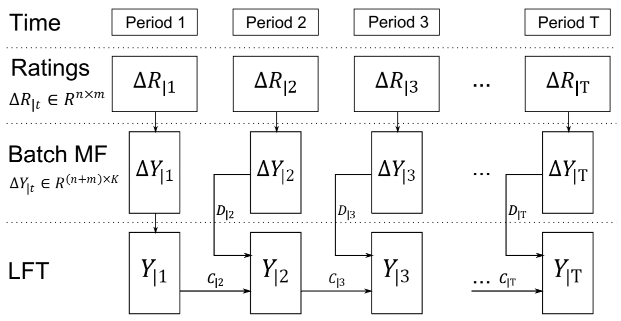

Figure 1: Work flow of the LFT framework.

trix $Y_{|t+1}$ from $Y_{|t}$ when a bunch of new ratings $\Delta R_{|t+1}$ come in time period $t+1$. We have the assumption:

Assumption 3.1. *The transformation of the latent feature matrix $Y_{|t}$ over the time period ($t = 1, 2, ...$) can be described as a linear function of its past value and the incremental latent feature matrix:*

$$Y_{|t+1} = Y_{|t}C_{|t+1} + \Delta Y_{|t+1}D_{|t+1} = O_{|t+1}L_{|t+1}, \tag{1}$$

where there are

$$L_{|t+1} = \begin{bmatrix} C_{|t+1} \\ D_{|t+1} \end{bmatrix} \in \mathbb{R}^{2K\times K}, O_{|t+1} = \left[Y_{|t}\ \Delta Y_{|t+1}\right] \in \mathbb{R}^{(n+m)\times 2K}. \tag{2}$$

It should be noted that $C_{|t+1}$ and $D_{|t+1}$ are K-dimensional matrices which project different latent feature vectors ($Y_{i,:|t+1}$ and $\Delta Y_{i,:|t+1}$) into a same linear space. In fact, the above linear transformation Eq.(1) presents a special form that resembles Vector Auto Regression (VAR) [Scott Hacker and Hatemi-J, 2008; Hatemi-J, 2004], which is an econometric model to capture the linear interdependencies among multiple time series. Besides, we will introduce this transformation with a low-rank approximation perspective in Section 5. In particular, we show the work flow of updating the latent feature matrix in Figure 1.

In order to get the updated feature matrix $Y_{|t+1}$, we need to figure out the matrix $L_{|t+1}$ when the matrix $X_{|t+1} = Sy(R_{|t+1})$. Specifically, the incremental training problem could be generally formulated as Eq.(3), where $f^G(.)$ is the objective function. n_1 and n_2 are the numbers of equality constraints $h^G(.)$ and inequality constraints $g^G(.)$ respectively. As shown in Eq.(3), the proposed incremental framework LFT is applicable to any batch-based MF model G with explicit objective functions (and explicit constrains).

$$\begin{aligned} \min_{L_{|t+1}} \quad & f^G\left(X_{|t+1}, O_{|t+1}L_{|t+1}\right) \\ s.t. \quad & h_i^G\left(X_{|t+1}, O_{|t+1}L_{|t+1}\right) = 0 \quad i = 1, 2, ..., n_1 \\ & g_i^G\left(X_{|t+1}, O_{|t+1}L_{|t+1}\right) \le 0 \quad i = 1, 2, ..., n_2 \end{aligned} \tag{3}$$

Before introducing the concrete algorithms to solve the objective function, we summarize the LFT framework in Algorithm 1. As can been seen from this algorithm, the key to LFT framework lies in Step 3. We leave the details for learning the matrix $L_{|t+1}$ in the next section.

Algorithm 1 The Linear Feature Transformation Process.

```
Input: Incremental matrix $\Delta R_{|t+1}$; Feature matrix $Y_{|t}$; The selected batch-based MF model $G$;
Output: Updated latent feature matrix $Y_{|t+1}$;
1: Decompose the incremental matrix $\Delta R_{|t+1}$ with the selected MF model $G$ and get a feature matrix $\Delta Y_{|t+1}$;
2: Combine the matrix $Y_{|t}$ and $\Delta Y_{|t+1}$ to get $O_{|t+1}$ by Eq.(2);
3: Calculate $L_{|t+1}$ until the objective function Eq.(3) converges;
4: return Updated feature matrix $Y_{|t+1} = O_{|t+1}L_{|t+1}$;
```

4 Implementation of LFT

In this section, we introduce a concrete LFT implementation for a constrained MF model named FAVA [Zeng *et al.*, 2015]. Compared to the choice of unconstrained batch-based MF models (e.g., PMF), choosing FAVA can be more representative to show that our framework is generally applicable to incrementally updating most of batch-based MF models by formalizing the corresponding incremental training problem as Eq.(3). After that, we compare the time-space complexity of our proposed model with some state-of-the-art methods.

4.1 The Incremental FAVA Model based on LFT

In this section, we develop a model named FAVA_LFT which incrementally update a batch-based MF model named FAVA as an example of applying LFT framework to batch-based constrained MF models.

Batch MF Phase. As with FAVA, FAVA_LFT uses the hyper-parameters written as $\{\sigma_j\}_{j=1}^K$, which can ensure that the superiority of hyper-parameters will not lose when the scale of the rating matrix is increasing [Zeng *et al.*, 2015]. The loss function is formulated in the following

$$f^{FAVA}(X, Y) = \frac{1}{2}\|I_{\Omega_X} \odot \left(X - YY^T\right)\|_F^2, \tag{4}$$

where $X = Sy(R)$ and Y denotes the combination of user and item feature matrix as shown in Section 3.

Through the FAVA algorithm, we can get the matrix $\Delta Y_{|t+1}$ from optimizing the problem [Zeng *et al.*, 2015]:

$$\begin{aligned} \min_{\Delta Y_{|t+1}} \quad & f^{FAVA}\left(\Delta X_{|t+1}, \Delta Y_{|t+1}\right) \\ s.t. \quad & \frac{1}{n+m}\sum_{i=1}^{n+m} \Delta Y_{i,k|t+1} \le \sigma_k^2 \quad (k = 1, 2, ..., K). \end{aligned} \tag{5}$$

LFT Phase. After getting the matrix $\Delta Y_{|t+1}$, we can establish $O_{|t+1}$ with the Assumption 3.1 and the known latent feature matrix $Y_{|t}$, then $Y_{|t+1}$ can be updated by Eq.(1) after getting the matrix $L_{|t+1}$ through optimizing the problem:

$$\begin{aligned} \min_{L_{|t+1}} \quad & f^{FAVA}\left(\overline{X}_{|t+1}, Y_{|t+1}\right) \\ s.t. \quad & L_{:,i|t+1}^T O_{|t+1}^T O_{|t+1} L_{:,i|t+1} \le (n+m)\sigma_i^2 \quad (i = 1, 2, ..., K), \end{aligned} \tag{6}$$

where $\overline{X}_{|t+1} = Sy\left(\overline{R}_{|t+1}\right)$, and $\overline{R}_{|t+1}$ consists of some original rating samples (a sub-matrix of $\overline{R}_{|t}$) and new coming ratings ($\Delta R_{|t+1}$). The scale of the matrix $\overline{X}_{|t+1}$ could be manually adjusted for balancing the effectiveness and efficiency of FAVA_LFT. Considering from the reasonableness,

Algorithm 2 FAVA_LFT Algorithm.

```
Input: Threshold of the absolute difference of the objective function $\varepsilon$; Sample rating matrix $\overline{R}_{|t+1}$; Factorization Variances $\{\sigma_i^2\}_{i=1}^K$; Latent feature matrix $Y_{|t}$; Incremental latent feature matrix $\Delta Y_{|t+1}$;
Output: Updated latent feature matrix $Y_{|t+1}$;
1: Initialize $L_{|t+1}$ and combined feature matrix $O_{|t+1}$ by Eq.(2);
2: do
3:     Calculate the gradient of $L_{|t+1}$ by Eq.(8);
4:     Use the steepest descent method to get the step size $\eta$;
5:     Update column vectors of $L_{|t+1}$ by Eq.(7);
6:     Calculate the absolute difference $\varepsilon_t$ of the objective function before and after the update of $L_{|t+1}$;
7: while $\varepsilon_t \geq \varepsilon$;
8: return $Y_{|t+1} = O_{|t+1}L_{|t+1}$
```

such process can also be regarded as a kind of approximation of original rating matrix by sparsification [Halko *et al.*, 2011].

For the constrained nonconvex optimization problem Eq.(6), we use a non-standard projection gradient method to update the optimization variables while maintaining its feasibility as follows:

$$L_{:,i|t+1} \leftarrow \Pi\left(L_{:,i|t+1} - \eta\frac{\partial f^{FAVA}\left(\overline{X}_{|t+1}, Y_{|t+1}\right)}{\partial L_{:,i|t+1}}, i\right), \tag{7}$$

where the gradient of $L_{|t+1}$ and the projection function $\Pi(.)$ are respectively described as Eq.(8) and Eq.(9). Besides, the step size η can be calculated by the steepest descent method.

$$\begin{aligned}\nabla L_{|t+1} = &-2O_{|t+1}^T\left[I_{\Omega_{\overline{X}_{|t+1}}} \odot \left(\overline{X}_{|t+1}\right.\right.\\ &\left.\left.-O_{|t+1}L_{|t+1}L_{|t+1}^TO_{|t+1}^T\right)\right]O_{|t+1}L_{|t+1}\end{aligned} \tag{8}$$

$$\Pi(l,i) = \begin{cases}\dfrac{\sqrt{n+m}\sigma_i l}{\sqrt{l^TO_{|t+1}^TO_{|t+1}l}}, & l^TO_{|t+1}^TO_{|t+1}l \geq (n+m)\sigma_i^2\\ l, & else\end{cases} \tag{9}$$

It should be noted that we project the different column vectors $L_{:,i|t+1}$ of the intermediate solution to the hyper-ellipsoidal surface in direction of the connection between the $L_{:,i|t+1}$ and the origin, which can help us to approximate a feasible point without solving linear equations established by proximal methods. In addition, the selection of the initial values for $L_{|t+1}$ in the feasible set is also a critical step. A simple idea is that we set $L_{|t+1} = [I, 0]$ as the initial optimization variable, because the solution $Y_{|t}$ in the previous time period necessarily satisfies the constraints.

To summarize the process that we iteratively update $L_{|t+1}$ and η until Eq.(6) converges, the implementation of FAVA_LFT has been detailed in Algorithm 2.

4.2 Complexity Analysis

In this section, we compare FAVA_LFT with Batch MF and vector-retraining models (e.g., RMF_R [Rendle and Schmidt-Thieme, 2008] and IRMF [Luo *et al.*, 2012]) in terms of the time-space complexity. And the results are shown in Table 2, where D is the iteration number. To a certain extent, the result can show the high efficiency and low storage overhead of

Table 2: The complexity of algorithms in the incremental phase

Methods	Time Complexity
Batch MF	$O(D(\|\Omega_{R_t}\| + \|\Omega_{\Delta R_t}\|)K)$
RMF_R	$O(D\|\Omega_{\Delta R_t}\|(\|\Omega_{R_t}\|/n + \|\Omega_{R_t}\|/m)K)$
IRMF	$O(D\|\Omega_{\Delta R_t}\|(\|\Omega_{R_t}\|/n + \|\Omega_{R_t}\|/m)K)$
FAVA_LFT	$O(D(\|\Omega_{\overline{R}_t}\|K + (n+m)K^2))$
Methods	Space Complexity
Batch MF	$O(\|\Omega_{R_t}\| + (n+m)K)$
RMF_R	$O(\|\Omega_{R_t}\| + (n+m)K)$
IRMF	$O(\|\Omega_{R_t}\| + D(n+m)K)$
FAVA_LFT	$O(\|\Omega_{\overline{R}_t}\| + (n+m)K)$

our framework. And the same complexity can be extended to LFT implementations based on some other batch-based MF models (e.g., PMF or NMF).

Time Complexity. According to Table 2, Batch MF is usually slower than RMF_R and IRMF in the case of $|\Omega_{R_t}| \gg |\Omega_{\Delta R_t}|$. However, when $|\Omega_{\Delta R_t}| \geq min\{n, m\}$, RMF_R and IRMF would have lower efficiency than the Batch MF. As a result, most vector-retraining models (e.g., RMF_R and IRMF) cannot address the incremental MF problem well when the number of coming ratings in each time period is large. On the other hand, the efficiency of our framework in incremental phase can be easily controlled by tuning $|\Omega_{\overline{R}_t}|$ which is usually a constant multiple of $|\Omega_{\Delta R_t}|$. Besides, according to Table 2, the scale of the incremental rating matrix has limited impact on the efficiency of LFT based models, and we believe that LFT framework will have high efficiency in large-scale online recommender systems.

Space Complexity. Vector-retraining models usually require a large number of user's historical ratings $R_{i,:}$ when retraining the user's feature vector $P_{i,:}$, which leads a certain number of historical ratings to be stored for each user. And because of the sparseness of the rating matrix, most of coming ratings cannot be dropped in each time period. As a result, the overhead of saving ratings shown in Table 2 should not be ignored. On the other hand, the matrix trained in LFT framework is a linear change of latent feature vectors over time rather than specific feature vectors. Therefore, only a small-scale sample matrix $|\Omega_{\overline{R}_t}|$ is needed and the data storage overhead can be reduced.

5 Theoretical Analysis

In this section, we conduct theoretical analysis to further explain the LFT framework and discuss its properties. Specifically, we first introduce a kind of model based on LFT named extend_LFT (eLFT), and then analyze eLFT from the low-rank approximation perspective. It is worth mentioning that we prove an upper bound on the training error expectation of the model. The bound can give the theoretical basis to guarantee that incremental MF models using the LFT framework is reasonable and practical.

Problem Formulation. Similar to Eq.(3), we formulate the eLFT problem with no constraints and no regularization terms as follows:

$$\underset{L_{|t+1}}{min}\ \frac{1}{2}\left\|I_{\Omega_{\widetilde{X}_{|t+1}}} \odot \left(\widetilde{X}_{|t+1} - O_{|t+1}L_{|t+1}L_{|t+1}^TO_{|t+1}^T\right)\right\|_F^2, \tag{10}$$

where $\widetilde{X}_{|t+1}$ is defined as

$$\begin{aligned}\widetilde{X}_{|t+1} =&Sy\left(I_{\Omega_{R_{|t}}} \odot \left(R_{|t} + \Delta P_{|t+1}\Delta Q^T_{|t+1}\right)\right)\\ &+ Sy\left(I_{\Omega_{\Delta R_{|t+1}}} \odot \left(\Delta R_{|t+1} + P_{|t}Q^T_{|t}\right)\right).\end{aligned} \tag{11}$$

It should be noted that since users cannot repeat rating, we assume $\Omega_{R_{|t}} \bigcap \Omega_{\Delta R_{|t+1}} = \emptyset$ in all subsequent analysis. Through Eq.(11), the estimated rating matrix has an offset which is related to the latent feature matrix. Therefore, we need some other methods to predict the missing ratings (e.g., the average of the ratings which are predicted as exact values in previous time periods).

Equivalence and Approximation. In order to understand eLFT from the perspective of low rank approximation, we consider the following joint optimization problem of existing ratings and incremental ratings in time period $t+1$

$$\begin{aligned}\min_{P,Q,\Delta P,\Delta Q}\ & \frac{1}{2}\left\|I_{\Omega_{Sy(\Delta R_{|t+1})}} \odot Sy\left(\Delta R_{|t+1} - \Delta P\Delta Q^T\right)\right\|_F^2\\ &+ \frac{1}{2}\left\|I_{\Omega_{Sy(R_{|t})}} \odot Sy\left(R_{|t} - PQ^T\right)\right\|_F^2.\end{aligned} \tag{12}$$

We replace the variable with $O \in \mathbb{R}^{(n+m)\times 2K}$, and $O = \begin{bmatrix} P & \Delta P \\ Q & \Delta Q \end{bmatrix}$. The optimization problem Eq.(12) can be equally expressed as

$$\begin{aligned}\min_{O}\ & \frac{1}{2}\left\|I_{\Omega_{Sy(\Delta R_{|t+1})}} \odot \left(Sy\left(\Delta R_{|t} + PQ^T\right) - OO^T\right)\right\|_F^2\\ &+ \frac{1}{2}\left\|I_{\Omega_{Sy(R_{|t})}} \odot \left(Sy\left(R_{|t} + \Delta P\Delta Q^T\right) - OO^T\right)\right\|_F^2,\end{aligned} \tag{13}$$

where $P = \begin{bmatrix} I_n & 0 \end{bmatrix} O \begin{bmatrix} I_K \\ 0 \end{bmatrix}$. Besides, $Q, \Delta P, \Delta Q$ can be obtained in a similar way.

We know that a local optimal solution $O_{|t+1}$ of Eq.(13) can be easily found when $O_{|t+1} = \begin{bmatrix} P_{|t}\ \Delta P_{|t+1} \\ Q_{|t}\ \Delta Q_{|t+1} \end{bmatrix}$. And the distinction between the local optimal and the global optimal solutions only depends on the batch-based model which is used to decompose $R_{|t}$ and $\Delta R_{|t+1}$ into $P_{|t}Q^T_{|t}$ and $\Delta P_{|t+1}\Delta Q^T_{|t+1}$. To solve the incremental problem Eq.(13) in different time periods, an intuitive idea is that when incremental data arrives, the columns of the matrix O are expanded with the incremental latent feature matrix to obtain the local optimal solution of the joint optimization problem Eq.(13). However, as time goes by, we cannot tolerate the continuous growing of the rank of the latent feature matrix. Therefore, we should find a K-rank approximation of $O_{|t+1}$.

According to [Halko *et al.*, 2011], the approximation can be set as $O_{|t+1}MM^T$, where the column vectors of matrix $M \in \mathbb{R}^{2K\times K}$ are orthonormal. The problem Eq.(13) can be relaxed as follows:

$$\begin{aligned}\min_{M}\ & \frac{1}{2}\left\|\left(I_{\Omega_{Sy(\Delta R_{|t+1})}} + I_{\Omega_{Sy(R_{|t})}}\right) \odot\right.\\ &\left.\left(\widetilde{X}_{|t+1} - O_{|t+1}MM^TO^T_{|t+1}\right)\right\|_F^2,\end{aligned} \tag{14}$$

which is equivalent to Eq.(10) when columns of $L_{|t+1}$ are orthonormal vectors.

In summary, the eLFT can be considered as to find a special K-rank approximation of a local optimal solution which can be got through optimizing a joint optimization problem of existing ratings and incremental ratings in time period $t+1$. In particular, we have the following theorem.

Theorem 5.1. In any time period $t+1$, we can obtain a solution of the optimization problem Eq.(10) to make the expectation of the training error $e_{|t+1}$ satisfies Eq.(15) when $K \geq 2$.

$$\mathbb{E}\sqrt{e_{|t+1}} \leq \sqrt{e_{|t} + \Delta e_{|t+1}} + \left(\frac{2K-1}{\sqrt{2}K - \sqrt{2}}\right)\left(\sum_{2K\geq j>K} \lambda_j^2\right). \tag{15}$$

In Eq.(15), λ_j means the j-th largest singular value of matrix $O_{|t+1}$, and $\Delta e_{|t}$ means the residual sum of squares (RSS) of the matrix $\Delta R_{|t}$ and $\Delta P_{|t}\Delta Q^T_{|t}$.

Proof. According to the discussion above, we know that the optimization problem Eq.(14) is equal to the low-rank approximation of a local optimal solution of problem Eq.(12).

For any known M, we have

$$\begin{aligned}&\left\|\left(I_{\Omega_{Sy(\Delta R_{|t+1})}} + I_{\Omega_{Sy(R_{|t})}}\right) \odot \left(\widetilde{X}_{|t+1} - O_{|t+1}MM^TO^T_{|t+1}\right)\right\|_F\\ \leq &\left\|\left(I_{\Omega_{Sy(\Delta R_{|t+1})}} + I_{\Omega_{Sy(R_{|t})}}\right) \odot \left(O_{|t+1}\left(I - MM^T\right)O^T_{|t+1}\right)\right\|_F\\ &+ \left\|\left(I_{\Omega_{Sy(\Delta R_{|t+1})}} + I_{\Omega_{Sy(R_{|t})}}\right) \odot \left(\widetilde{X}_{|t+1} - O_{|t+1}O^T_{|t+1}\right)\right\|_F\\ \leq &\left\|\left(I_{\Omega_{Sy(\Delta R_{|t+1})}} + I_{\Omega_{Sy(R_{|t})}}\right) \odot \left(\widetilde{X}_{|t+1} - O_{|t+1}O^T_{|t+1}\right)\right\|_F\\ &+ \left\|O_{|t+1}\left(I - MM^T\right)O^T_{|t+1}\right\|_F\\ = &\left\|O_{|t+1}\left(I - MM^T\right)^2O^T_{|t+1}\right\|_F + \sqrt{2e_{|t} + 2\Delta e_{|t+1}}.\end{aligned} \tag{16}$$

According to the *Average Frobenius errors* mentioned in [Eckart and Young, 1936], in Eq.(16), if we initialize the M with $\overline{M}$ which can be got by *Randomized Range Finder* algorithm [Halko *et al.*, 2011], we have

$$\mathbb{E}\left\|\left(I - \overline{MM}^T\right)O^T_{|t+1}\right\|_F^2 \leq \left(1 + \frac{K}{K-1}\right)\sum_{2K\geq j>K} \lambda_j^2, \tag{17}$$

when $K \geq 2$. And we can get the following inequality for any $\overline{M}$ through the Cauchy-Schwarz inequality.

$$\left\|O_{|t+1}\left(I - \overline{MM}^T\right)^2O^T_{|t+1}\right\|_F \leq \left\|\left(I - \overline{MM}^T\right)O^T_{|t+1}\right\|_F^2 \tag{18}$$

Through Eq.(17) and Eq.(18), we can easily get

$$\mathbb{E}\left\|O_{|t+1}\left(I - \overline{MM}^T\right)^2O^T_{|t+1}\right\|_F \leq \left(\frac{2K-1}{K-1}\right)\left(\sum_{2K\geq j>K} \lambda_j^2\right). \tag{19}$$

Then we substitute Eq.(19) into Eq.(16).

As the problem Eq.(10) and Eq.(14) are equivalent when matrix of $L_{|t+1}$ is orthonormal, we can apparently obtain a solution that meets Eq.(15) when $L_{|t+1}$ is initialized with $\overline{M}$. Besides, the error bounds will be maintained after the gradient descent process of optimizing Eq.(10). Therefore, the theorem has been proved.

According to the theorem and the proof, regardless of the number of incremental ratings, the total error will always be a function of the rank of latent feature matrices K and the singular values $\{\lambda_j\}_{j=K+1}^{2K}$ of matrix $O_{|t+1}$. Thus, finding the optimal K-rank approximation of the joint optimization problem is a good way to solve the incremental problem.

6 Experiments

In this section, we mainly introduce four experiments, which evaluate our method and baselines from four aspects: effectiveness, efficiency, the storage and use rate of rating samples.

Dataset. The experiments were conducted on two real-world datasets MovieLens1M (ML1M) and MovieLens10M (ML10M), which are standard datasets in rating prediction. The data is collected by the GroupLens Research from the MovieLens web site [1].

Measurements. The four metrics, which are designed to evaluate the effectiveness, efficiency, storage and use rate of rating samples are the RMSE of all predicted ratings [Luo *et al.*, 2012], the cumulative runtime, the number and the use rate of rating samples in each time period, respectively.

Baselines. We chose two widely used incremental M-F methods, namely RMF_R [Rendle and Schmidt-Thieme, 2008] and IRMF [Luo *et al.*, 2012], for comparison on a small dataset. Besides, we accelerated the above two models as new baselines, namely QRMF_R and QIRMF, on both datasets. Otherwise, the efficiency of original vector-retraining models on ML10M is intolerable because of the high time complexity. For QRMF_R and QIRMF, the main idea is that we update a latent feature vector only when the corresponding user (item) last appears in the incremental ratings. Therefore, we can effectively reduce the number of gradient calculation to improve the efficiency. Of course, it will make algorithms less effective because of incomplete updates, and specific performance gaps will be shown on the ML1M dataset.

Experimental Process. The specific steps of experimental process is as follows:

- First, the experimental data sorted by timestamp was divided into T disjoint continuous parts with a similar scale. We call the i-th incremental data part i-ID.
- Second, 1-ID was used as the batch training set of incremental recommenders.
- After that, we applied i-ID to simulate the real-world rating batch, and evaluated each model by calculating the four metrics mentioned above. Note that, we only predicted the ratings of whom the user and item have appeared in former ID.
- Then, vector-retraining models (RMF_R, IRMF, QRMF_R and QIRMF) need to deal with the coming ratings one by one, and FAVA_LFT can update the whole feature matrix with batch production (e.g., Algorithm 1).
- We repeated steps 3 and 4 from $i = 2$ until $i = T$

Through such experimental setup demonstrated in Figure 2, we can observe that the incremental training process is shown as the arrow from testing 2-ID to training 2-ID.

[1] http://movielens.org

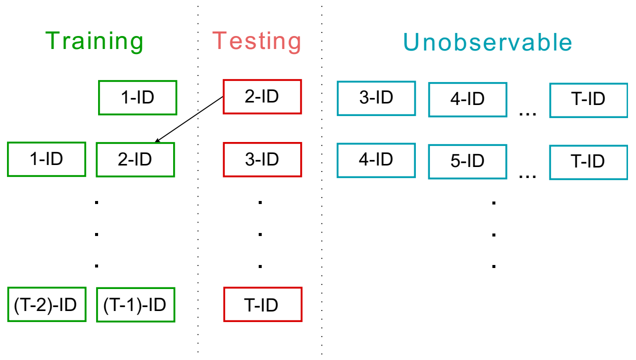

Figure 2: The Change of the State of Each i-ID.

Parameters Settings. The initial values of the feature matrices of RMF_R, IRMF, QRMF_R and QIRMF were randomly drawn from a normal distribution ($\mathcal{N}(0, 0.01)$). The main parameters listed in Table 3 are: the regularization term coefficient λ, the step size η, the number of iteration in incremental training phase D, the rank of latent feature matrices K, the size of sampled matrix $|\Omega|$, the threshold of the absolute difference of the objective function in FAVA_LFT ε, and the ID number for each dataset T.

It should be noted that all the MF parameters (λ, η, D) are obtained by five folds cross tuning on original datasets with corresponding batch-based MF models.

Experimental Results. The experimental results on ML1M and ML10M are depicted in Figure 3 and Figure 4, and the accurate values at last time period are shown in Table 4.

Effectiveness and Efficiency. We can see that original vector-retraining models (RMF_R and IRMF) are usually more effective, while the efficiency of them is much lower than the accelerated baselines (QRMF_R and QIRMF) in Figure 3. Especially for IRMF, it is difficult to find a balance between effectiveness and efficiency, because the negative effect of speeding up the algorithm cannot be accepted. And we think the main reason is that the non-standard descent method in IRMF will expand the error in the case of incomplete updates. For RMF_R, although speeding up the algorithm has only a limited effect on the precision, its performance in terms of effectiveness is also less satisfactory on ML1M dataset. As a contrast to IRMF, the FAVA_LFT algorithm based LFT framework has increased the speed almost 174 times in the case of losing about 2% accuracy on ML1M. Meanwhile, compared to RMF_R, FAVA_LFT in terms of accuracy and efficiency can achieve more outstanding performance. And a similar result also appears on ML10M dataset, which is shown by Figure 4.

Table 3: Parameters in the experiments

	λ	η	D	K	$\|\Omega\|$	ε	T
ML1M							50
(Q)RMF_R	0.02	0.0002	600	70			
(Q)IRMF	0.02	0.0002	600	70			
FAVA_LFT				70	10^5	1.0	
ML10M							200
QRMF_R	0.01	0.00018	1200	70			
QIRMF	0.01	0.00018	1200	70			
FAVA_LFT				70	3×10^5	1.0	

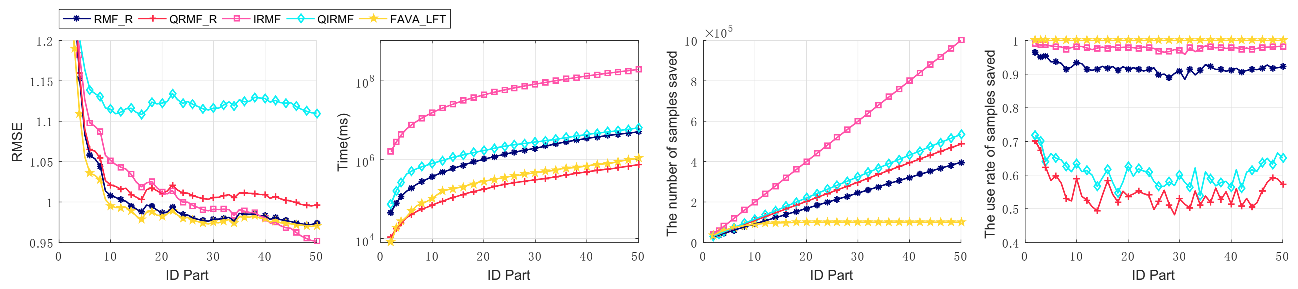

Figure 3: The experimental results on ML1M.

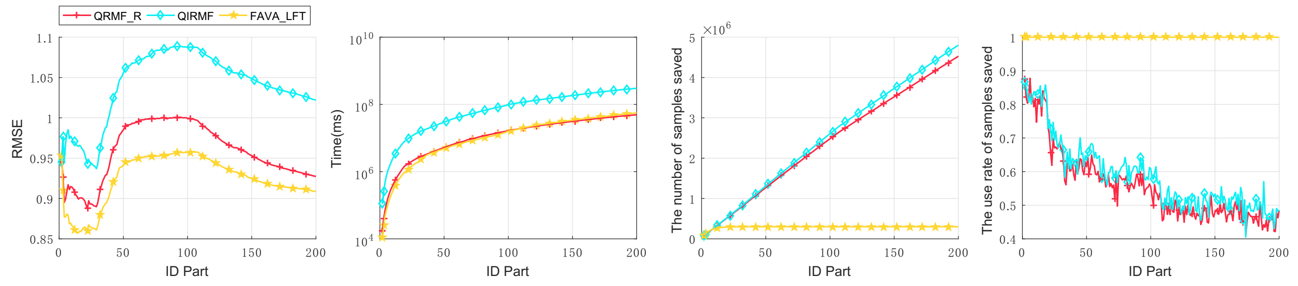

Figure 4: The experimental results on ML10M.

Table 4: Comparison on incremental learning performance

Dataset	ML1M					ML10M		
Models / Metrics	RMF_R	QRMF_R	IRMF	QIRMF	FAVA_LFT	QRMF_R	QIRMF	FAVA_LFT
RMSE	0.97373878	0.99632469	0.95135957	1.10995587	0.97089095	0.92735359	1.02200030	0.90891774
Runtime(ms)	4.90×10^6	7.29×10^5	1.86×10^8	6.19×10^6	1.07×10^6	4.92×10^7	3.02×10^8	5.72×10^7
Samples saved	396620	486695	1000000	534826	99914	4527003	4802356	299107

Storage Overhead and Sample Use Rate. In Figure 3 and Figure 4, RMF_R, QRMF_R, IRMF and QIRMF have to store much more rating samples than FAVA_LFT, which is caused by the uncertainty of using historical data. While consuming a large amount of storage resources, the low sample use rate causes the bad impacts on the effectiveness of QRMF_R and QIRMF. As for FAVA_LFT, with the high sample use rate, we are able to calculate the feature transformation matrix ($L_{|t+1}$) while saving a small amount of rating samples.

In summary, experimental results on two real-world datasets clearly validate the excellent effectiveness, efficiency and storage performance of the proposed framework.

7 Conclusions

In this paper, we designed a framework of incremental M-F for enhancing the performance of online recommendation. Specifically, we demonstrated that the LFT framework is generally applicable for incrementally updating most of M-F models. Besides, This framework shows a relatively high accuracy with a computation and space efficient training process in a online scenario. Furthermore, we explained the LFT framework in a low-rank approximation perspective, and proved an upper bound on the training error expectation of the model. Finally, extensive experimental results on two real-world datasets clearly validated the superiority of the proposed framework.

Acknowledgements

This research was partially supported by grants from the National Key Research and Development Program of China (Grant No. 2016YFB1000904), the National Science Foundation for Distinguished Young Scholars of China (Grant No. 61325010), the National Natural Science Foundation of China (Grants No. 61403358, No. 61602147 and U1605251), and the Anhui Provincial Natural Science Foundation(Grant No. 1708085QF155). Qi Liu gratefully acknowledges the support of the Youth Innovation Promotion Association of CAS (No. 2014299).

References

[Bach *et al.*, 2008] Francis Bach, Julien Mairal, and Jean Ponce. Convex sparse matrix factorizations. *arXiv preprint arXiv:0812.1869*, 2008.

[Brand, 2003] Matthew Brand. Fast online svd revisions for lightweight recommender systems. In *SDM*, pages 37–46. SIAM, 2003.

[Chen *et al.*, 2016] Chao Chen, Dongsheng Li, Qin Lv, Junchi Yan, Stephen M Chu, and Li Shang. Mpma: mixture probabilistic matrix approximation for collaborative filtering. In *IJCAI*, pages 1382–1388, 2016.

[Devooght *et al.*, 2015] Robin Devooght, Nicolas Kourtellis, and Amin Mantrach. Dynamic matrix factorization with priors on unknown values. In *SIGKDD*, pages 189–198. ACM, 2015.

[Eckart and Young, 1936] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[Halko *et al.*, 2011] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[Hatemi-J, 2004] Abdulnasser Hatemi-J. Multivariate tests for autocorrelation in the stable and unstable var models. *Economic Modelling*, 21(4):661–683, 2004.

[Hazan, 2008] Elad Hazan. Sparse approximate solutions to semidefinite programs. In *Latin American Symposium on Theoretical Informatics*, pages 306–316. Springer, 2008.

[Lee and Seung, 1999] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[Liberty, 2013] Edo Liberty. Simple and deterministic matrix sketching. In *SIGKDD*, pages 581–588. ACM, 2013.

[Lin *et al.*, 2017] Hao Lin, Hengshu Zhu, Yuan Zuo, Chen Zhu, Junjie Wu, and Hui Xiong. Collaborative company profiling: insights from an employees perspective. AAAI, 2017.

[Liu *et al.*, 2011] Qi Liu, Yong Ge, Zhongmou Li, Enhong Chen, and Hui Xiong. Personalized travel package recommendation. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 407–416. IEEE, 2011.

[Luo *et al.*, 2012] Xin Luo, Yunni Xia, and Qingsheng Zhu. Incremental collaborative filtering recommender based on regularized matrix factorization. *Knowledge-Based Systems*, 27:271–280, 2012.

[Matuszyk *et al.*, 2015] Pawel Matuszyk, João Vinagre, Myra Spiliopoulou, Alípio Mário Jorge, and João Gama. Forgetting methods for incremental matrix factorization in recommender systems. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 947–953. ACM, 2015.

[Rendle and Schmidt-Thieme, 2008] Steffen Rendle and Lars Schmidt-Thieme. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *RecSys*, pages 251–258. ACM, 2008.

[Salakhutdinov and Mnih, 2008] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *ICML*, pages 880–887. ACM, 2008.

[Sarwar *et al.*, 2002] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Fifth International Conference on Computer and Information Science*, pages 27–28. Citeseer, 2002.

[Scott Hacker and Hatemi-J, 2008] R Scott Hacker and Abdulnasser Hatemi-J. Optimal lag-length choice in stable and unstable var models under situations of homoscedasticity and arch. *Journal of Applied Statistics*, 35(6):601–615, 2008.

[Song *et al.*, 2015] Qiang Song, Jian Cheng, and Hanqing Lu. Incremental matrix factorization via feature space relearning for recommender system. In *RecSys*, pages 277–280. ACM, 2015.

[Vinagre *et al.*, 2014] João Vinagre, Alípio Mário Jorge, and João Gama. Fast incremental matrix factorization for recommendation with positive-only feedback. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 459–470. Springer, 2014.

[Wang *et al.*, 2011] Fei Wang, Hanghang Tong, and Ching-Yung Lin. Towards evolutionary nonnegative matrix factorization. In *AAAI*, volume 11, pages 501–506, 2011.

[Wu *et al.*, 2012] Le Wu, Enhong Chen, Qi Liu, Linli Xu, Tengfei Bao, and Lei Zhang. Leveraging tagging for neighborhood-aware probabilistic matrix factorization. In *CIKM*, pages 1854–1858. ACM, 2012.

[Wu *et al.*, 2017] Le Wu, Yong Ge, Qi Liu, Enhong Chen, Richang Hong, Junping Du, and Meng Wang. Modeling the evolution of users preferences and social links in social networking services. *IEEE Transactions on Knowledge and Data Engineering*, 29(6):1240–1253, 2017.

[Zeng *et al.*, 2015] Guangxiang Zeng, Hengshu Zhu, Qi Liu, Ping Luo, Enhong Chen, and Tong Zhang. Matrix factorization with scale-invariant parameters. In *IJCAI*, pages 4017–4024. AAAI Press, 2015.

[Zhu *et al.*, 2015] Hengshu Zhu, Enhong Chen, Hui Xiong, Kuifei Yu, Huanhuan Cao, and Jilei Tian. Mining mobile user preferences for personalized context-aware recommendation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(4):58, 2015.