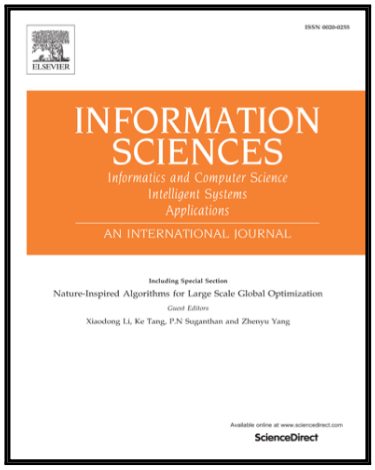
Journal Pre-proof



Online Collaborative Filtering with Local and Global Consistency

Xiao-Yu Huang, Bing Liang, Wubin Li

PII: S0020-0255(19)30743-1
DOI: https://doi.org/10.1016/j.ins.2019.08.009
Reference: INS 14758

To appear in: *Information Sciences*

Received date: 13 January 2019
Revised date: 1 August 2019
Accepted date: 3 August 2019

Please cite this article as: Xiao-Yu Huang, Bing Liang, Wubin Li, Online Collaborative Filtering with Local and Global Consistency, *Information Sciences* (2019), doi: https://doi.org/10.1016/j.ins.2019.08.009

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

© 2019 Published by Elsevier Inc.

JOURNAL PRE-PROOF

Online Collaborative Filtering with Local and Global Consistency*

Xiao-Yu Huang[a,*], Bing Liang[b], Wubin Li[c,d]

[a]*School of Economics and Commerce, South China University of Technology, Guangzhou 510006, China. Email: echxy@scut.edu.cn.*
[b]*Academy of Guangdong Telecom Co.Ltd, Guangzhou 510630, China.*
[c]*Ericsson Research, Ericsson Montréal, H4P 2N2, QC, Canada.*
[d]*Cloud Research Center, North China University of Technology, Beijing 100144, China.*

Abstract

Collaborative Filtering (CF) is one of the most popular technologies used in online recommendation systems. Most of the existing CF studies focus on the offline algorithms, a major drawback of these algorithms is the lack of ability to use the latest user feedbacks to update the learned model in realtime, due to the high cost of the offline training procedure. In this work, we propose *Logo*, an online CF algorithm. Our proposed method is based on a hierarchical generative model, with which, we derive a set of *local* and *global* consistency constraints for the prediction targets, and eventually obtain the design of the learning algorithm. We conduct comprehensive experiments to evaluate the proposed algorithm, the results show that: (1) Under the online setting, our algorithm achieves notably better prediction results than the benchmark algorithms; (2) Under the offline setting, our algorithm attains comparable accurate prediction results with the best performed competitors; (3) In all the experiments, our algorithm performs tens or even hundreds of times faster than the comparison algorithms.

Keywords: Artificial intelligence; Collaborative filtering; Online learning; Recommender system

*The corresponding author.

JOURNAL PRE-PROOF

JOURNAL PRE-PROOF

1. Introduction

A main concern of the online recommender systems is how to accurately predict the users preferences on the shopping items, so they can make appropriate recommendations to the users. Collaborative Filtering (CF) is one of the most popular technologies for this usage. A classic CF model consists of three components: users, items and ratings. Where each rating associates with an *user-item* pair, indicating the preference of the user to the item. In actual applications, every user often has rated only a small fraction of the items, so the CF model has to infer the unavailable (or *missing*) ratings based on the few ones at hand [35, 19]. A typical CF application example is shown in Figure. 1, where given an incomplete 4 users-by-4 items rating matrix[1], to advertise the products to the users precisely, the recommender system has to first use the CF algorithm to predict all the missing ratings, then based on the observed and predicted ratings, it recommends the products to the users accordingly.

Ever since the launch of the *Netflix* matrix completion competition in 2006 [27], the CF problem has attracted many research attentions [17], specifically, inspired by the recent emerging matrix imputation theory, the theory of matrix completion has been well established [4, 5]. However, concerning on the working scheme, almost all the proposed algorithms work in the offline mode, where in each training round, they always take all the available data samples as input to learn the model. When the amount of the data is large, the learning procedure will be very time consuming. For example, suppose there is a recommender system with n users and m items, to train an item-based collaborative filtering model [30] for it, we need to calculate all the pairwise inner products of the item based rating vectors. Noting that in the system there are altogether C_m^2 pairs of items and the length of the rating vector is n, so the time complexity of the computation is $\theta(nm^2)$. Specifically, given a trained item-based CF model, suppose that there have some changes happened to one of the rating vectors (e.g.,

[1] We assume here that greater rating value refers to higher user preference.

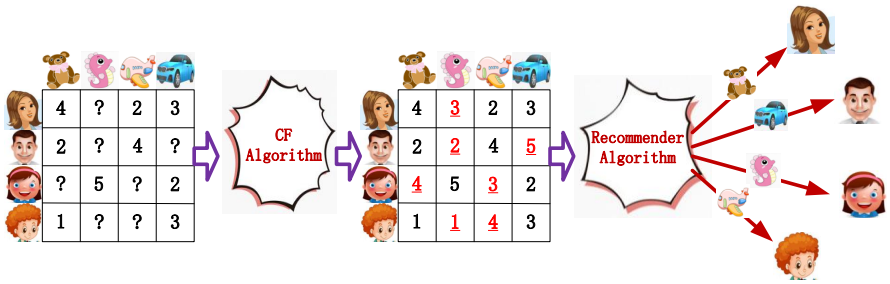


Figure 1: An example application of the CF algorithm, where the question marks correspond to the missing ratings, and the underlined red values correspond to the predicted ones. The CF algorithm is used to predict the missing ratings, and the recommender algorithm is used to advertise the products to the users according to the observed and predicted ratings. It's assumed here that greater rating value refers to higher user preference.

a new rating to some item is observed), to adjust the model with regard to the changes, we have to recalculate all the inner products between the changed rating vector and the other $(m-1)$ vectors, which needs $\theta(nm)$ computations. However, for many online business platforms, these computation costs are too high. For instance, it's reported that in Taobao[2], the largest online consumer-to-consumer platform in China, there're about one billion users and two billion items[34]. According to the above complexity results, every adjustment of the trained model needs $c \times 10^{18}$ computations, here c is constant. For a system with stream data in, this cost is hardly to be acceptable. While on the other side, in practical uses, the online recommender systems are always expected to adjust their predictions and improve the performances in realtime. Therefore, a CF algorithm that can predict online and adjust the model itself by learning incrementally, or the online CF algorithm, is in need.

In this work, we propose *Logo*, a novel online CF algorithm. Unlike the conventional offline CF models that perform in the "trained with the all (available ratings), and predict for the all (missing ratings)" scheme, *Logo* learns the

[2]http://www.taobao.com

JOURNAL PRE-PROOF

model and use it to predict incrementally. Specifically, in the *Logo* model, there consists of a set of predictors, whenever a rating prediction request arrives, the model will respond with an estimate produced by the predictors. In addition, it's assumed that upon the prediction is made, the true value of the queried target will be revealed, and the algorithm has to use the value to adjust the before-learned predictors immediately.

The term *Logo* stands for *learning with **Lo**cal and **glo**bal consistency*. As aftermentioned, *Logo* is based on a three-tiered statistical model on the rating data, where the first tier consists of the individual ratings, the second tier is on the distribution of the first tier, and the third tier is on the distribution of the second tier. With this representation, the estimates of the ratings in the first tier should be consistent with the second tiered model (*local consistency*). Meanwhile, the introduction of the estimated values will eventually lead to changes to the distribution of the second tier. These changes, should be consistent with the third tier model (*global consistency*).

In summary, the contributions of this paper are threefold: First, we propose a three-tiered generative model for the *user-item* ratings, based on the model, we recast the prediction problem as one that minimize a derived target function with a set of local and global constraints. Second, we present an efficient solution algorithm to the optimization target, which only needs $\theta(k)$ time to make a rating prediction, where k is the size of the rating domain. Third, for evaluation, we conduct comprehensive experiments with four real-world large scale data sets, all the results show that our algorithm is competitive to the state-of-the-art methods.

2. Related works

Many efforts have been devoted to the CF problem, for details we refer the readers to the comprehensive survey of Koren et al. [17]. According to the working scheme, all these works can be categorized into offline models and online models.

JOURNAL PRE-PROOF

JOURNAL PRE-PROOF

2.1. The offline CF models

Roughly, the offline models can be classified into three types: The neighborhood models, the factor models, and the deep models.

In principle, the neighborhood models are based on the manifold assumption [22], where similar users should have similar preferences (or similar items should have similar ratings) [38, 14]. Therefore, the algorithms usually work in two steps: First, they calculate the similarity between the target user (item) and the other users (items); Second, they produce the prediction via fusing all the observed ratings of the target item (user) as well as all the similarity values obtained in the first step.

The factor models usually adopt an $n \times m$ matrix R to store the ratings given by the n users to the m items, and assume that R can be approximated as the product of a set of low rank factor matrices, i.e., $R \approx Z_1 * Z_2 * \cdots * Z_l$ [24, 17, 13]. To predict the target rating (e.g., $R_{i,j}$), the algorithms first estimate the factor matrices $\hat{Z}_1, \hat{Z}_2, \ldots, \hat{Z}_l$ with the observed ratings, then produce the estimate as $\hat{R}_{i,j} = (\hat{Z}_1 * \hat{Z}_2 * \cdots * \hat{Z}_l)_{i,j}$.

The recent emerging deep models have attracted many research attentions [32, 36]. An importation distinction between the deep models and the conventional models is the representations of the objects: in conventional models, the representations are by empirical feature engineering, while in the deep models, the the representations are mainly by learning. Therefore, compared with the conventional models, the deep models need extra training to learn to represent.

In summary, all the offline models strictly follow the "trained offline - predict online" scheme, during the training procedure, they always exhaust all the data samples available to learn the predictors, so the time cost is very high, therefore, the models are not appropriate for the stream data processing applications where the learned parameters need adjusted in realtime upon the new samples arrive.

2.2. The online CF models

There have been few efforts devoted to the online CF models, most of these works can be recognized as the latent factor based representation models, and

JOURNAL PRE-PROOF

can be categorized into two types: The sparse models and the dense models.

The sparse models focus on the parsimony of the representations. A pioneer work is the online dictionary learning model [25], which assumes that there exists a (latent) dictionary matrix, and each (latent) feature vector (i.e., the user feature vector and the item feature vector) can be represented as a sparse combination of the column vectors of the dictionary. However, to use the new observed ratings to update the learned parameters, the model has to call a time consuming LARS [9] solver, which is very inefficient for large scale online learning applications. Lin et al. [21] explore the model sparsity from another perspective, they assume that the (latent) feature vectors are intrinsically sparse, and use the ℓ_1 regularization terms to enhance the sparsity. As to the time efficiency issue, the proposed algorithm is based on the coordinate Alternating Least Square (ALS) method, which exhibits faster convergence speed compared with Mairal et al. [25] in empirical studies.

The dense CF methods don't need to provide guarantees for the model sparsity, therefore, the stochastic gradient descent (SGD) method is popularly used . Ling et al. [23] and Wang et al. [33] have proposed similar models, they both adopt the same regularized loss as that used in the probabilistic matrix factorization model [29], and employ SGD as the solution algorithm. The major distinction between the methods is, in Ling et al. [23], the SGD step is performed on the individuals (i.e., the individual user feature vector and the individual item feature vector); while in Wang et al. [33], the SGD step is performed on the groups (i.e., the group of feature vectors that corresponds to a set of similar users, and the group of feature vectors that corresponds to a set of similar items). But we note that both of the approaches belong to the first order online optimization method, which may incur relatively slow convergence rate.

Ling et al. [23] and Wang et al. [33] only consider the explicit ratings, in Koren's famous SVD++ model [18], it's shown that the prediction model can also benefit from the implicit user feedbacks. He et al. [12] present an online CF model that takes into account the implicit factors. In the proposed model,

JOURNAL PRE-PROOF

the prediction step is based on a modified ALS method, compared with the conventional ALS method, the modified ALS method reduces the time complexity by an order of magnitude of the latent feature dimension d. However, in the update step, the model needs $\theta(d^2)$ time to update a learned feature vector, this cost is too high for a stream data processing system.

All the above dense models can be treated as the ***onlinization*** variants of the corresponding offline CF methods, besides of them, there're also some direct explorations on the online CF algorithms. Blondel et al. [3] propose an online nonnegative matrix factorization method, which is based on the online classification algorithm PA (i.e., the **P**assive-**A**ggressive (PA) learning model [8]), But we note that in this model, it needs to maintain two feature matrices (i.e., the user feature matrix and the item feature matrix), which is relatively space inefficient compared with our afterproposed *Logo* method.

3. The propose method

3.1. Preliminaries

From herein we use upper case letters (e.g., $X, Y, Z, \dots$) to denote the random variables, and lower cases to represent the instances.

Our work is mainly based on Information Theory. Below we introduce some definitions and preliminary results used in this paper. Most of them can be found in [7].

Let $\mathcal{P}$ be a distribution with $p(X)$ as the probability density function (p.d.f) for $X \sim \mathcal{P}$, then *entropy* of X is defined as

$$H(X) = -\int p(x) \ln p(x) dx. \tag{1}$$

Specifically, in the case that $p(X)$ is Gaussian, let $X \sim N(\mu, \Sigma)$, where μ is the mean and Σ is the covariance matrix, then $H(X) = \frac{1}{2} \ln(2\pi e)^d |\Sigma|$, where d is the dimension of X, $|\Sigma|$ is the determinant of Σ. In particular, if $d = 1$, then Σ degenerates to the variance of X, denote $\Sigma = \sigma^2$, then $H(X) = \frac{1}{2} \ln 2\pi e\sigma^2$.

In the context of Coding Theory, $H(X)$ is also the *minimum expected coding length* of X with base e. Hence as an alternative expression, below we denote

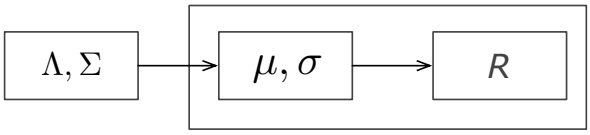


Figure 2: The rating generative model, where we assume that each rating of R is drawn from a Gaussian distribution $N(\mu, \sigma^2)$, and furthermore, each prior $[\mu, \sigma]^T$ is drawn from a hyper Gaussian distribution $N(\Lambda, \Sigma)$, too .

$Len(X) = H(X)$, term it *the minimum coding length of* X; and $Len(x) = -\ln p(x)$, term it *the optimum coding length of* x (or *the length of* x for short).

Let $x_1, x_2, \ldots, x_n \sim \mathcal{P}$, when n is large enough, according to the large number law, $Len(X)$ can be estimated by the mean coding length of x_is, i.e.,

$$Len(X) = \frac{\sum_{i=1}^{n} Len(x_i)}{n} = -\frac{\sum_{i=1}^{n} \ln p(x_i)}{n}. \tag{2}$$

In other words, we can approximate $H(X)$ with the sum $-\frac{\sum_{i=1}^{n} \ln p(x_i)}{n}$.

Some other adopted notations include the row rating sets R_is and column rating sets C_js, where R_i consists of the all non-zero values of the ith row[3], and C_j is composed of all nonzero values of the jth column.

3.2. The rating generation model

In this section we present the generative model for the *user-item* ratings.

Our proposed model is shown in Figure. 2, where given an n (users)-by-m (items) rating matrix R, we assume that R is generated in the *row by row* manner outlined in Procedure 1, showing that for each user, all her/his ratings are normally, identically and independently distributed (n.i.i.d), and in addition, for the all distributions, their parameters are n.i.i.d, too.

According to the symmetry, we can also assume that R is generated in the *column by column* manner. Similar to the above *row by row* manner, we

[3]Throughout this paper, unlike conventional definitions, we allow a set to contain duplicate values.

JOURNAL PRE-PROOF

Procedure 1: The ratting generation procedure.

Pick a pair of hyperprior parameters $\Lambda_1 = [\theta_{1,1}, \theta_{1,2}]^T$;

Pick a covariance matrix $\Sigma_1 = \begin{bmatrix} \kappa_{1,1}^2 & 0 \\ 0 & \kappa_{1,2}^2 \end{bmatrix}$;

for *each user* i **do**

  Draw a prior $(\mu_{1,i},\sigma_{1,i}^2) \sim N(\Lambda_1, \Sigma_1)$;

  for *each item* j **do**

    Draw $R_{i,j} \sim N(\mu_{1,i}, \sigma_{1,i}^2)$;

  end

end

use $\mu_{2,j}$ and $\sigma_{2,j}^2$ to denote the distribution parameters on the jth column, and denote the hyperprior parameters as $\Lambda_2 = [\theta_{2,1}, \theta_{2,2}]^T$ and $\Sigma_2 = \begin{bmatrix} \kappa_{2,1}^2 & 0 \\ 0 & \kappa_{2,2}^2 \end{bmatrix}$, respectively.

Our initial idea of the above model is inspired by the topic model [2]. For each user i, we treat the ratings given by i as a review article on the items, where the distribution over the the ratings is the article topic, and the detailed ratings correspond to the article words. Therefore, given n users, there're altogether n topics. In Procedure 1, we use the pair $(\mu_{1,i}, \sigma_{1,i})$ to characterize the topic of user i. In addition, noting that all the n topics are on the same item set, so there should be some inherent consistencies existing among these topics, to describe the consistency, we introduce the hyperdistribution over the topics, which is characterized with the pair (Λ_1, Σ_1) in the procedure.

It's noteworthy that the topic model belongs to the bag-of-word model, where the sequence of the words is not taken into account. But in the context of the CF applications, the rating sequence is unexchangable. For example, given two distinct items j_1 and j_2, in the bag-of-word model, the assignments $(R_{i,j_1} = 1, R_{i,j_2} = 5)$ and $(R_{i,j_1} = 5, R_{i,j_2} = 1)$ are identical, while in the CF applications, they are completely opposed!

JOURNAL PRE-PROOF

JOURNAL PRE-PROOF

The above observations suggest that we should introduce more constraints to keep the sequence information. Therefore, besides of the user based topics (i.e., the row-by-row rating generation manner), we also take into account the item based topics (i.e., the column-by-column rating generation manner). With both of the user based topics and the item based topics, every rating of R is located at the cross of an unique (user based topic, item based topic) pair, so the sequence information is retained.

3.3. The basic idea

To illustrate the basic idea of our proposed algorithm, let's first study the rating matrix imputation problem presented in Table. 1, where the task is to infer the true rating of the question mark based on some observations (colored pink in the background). According to the rating generation model, there're four statistical distributions related to the target estimate: The distribution over the target user's ratings, the distribution over the target item's ratings, the hyper distribution over the user rating distributions, and the hyper distribution over the item rating distributions. In accordance with the hierarchies of Figure. 2, introducing an estimate to the learned model will first lead to some changes to the pre-estimated distributions of the corresponding row and column, and then the changes of the row and column distributions will eventually lead to changes to the corresponding hyper distributions, too.

For the changes happened to the distribution of an individual row or column, for statistical consistency, we hope they can be as small as possible. Noting that here our concern is only with the individuals (i.e., the target user based row vector and the target item based column vector), we term it as the *local consistency*. As to the changes happened to the hyper distributions, as aforementioned, there exists some other intrinsic consistency underlying the data, which should be preserved after introducing the estimate. Since the scope of the hyper distributions is global, we term this constraint as the *global consistency*.

Our principle to make the predictions is via making use of the local and global consistency, therefore, we name the proposed model as *Logo*, namely,

3	5	2	...	
	?	2	...	1
	4		...	
	⋮		...	
			...	

Table 1: An illustration example, where the table stands for the rating matrix R, in which the rows are for the users, the columns are for the items, and the row-column cross entry is for the rating associated with the corresponding *user-item* pair.

*learning with **Lo**cal and **glo**bal consistency*. Given the prediction target, *Logo* will first exploit the consistent relations associated to the target rating, then explore the candidate rating set and produce the prediction which trades off the local and global consistency.

3.4. The local consistency

Suppose the prediction target is the preference of user i on item j, i.e., $R_{i,j}$. Let $\mathcal{D}_i$ be the distribution estimated with R_i. Denote the estimated rating as $\hat{R}_{i,j}$, and the distribution estimated with $R_i \bigcup \{\hat{R}_{i,j}\}$ as $\hat{\mathcal{D}}_i$. According to the row based generation assumption, $\mathcal{D}_i$ and $\hat{\mathcal{D}}_i$ are identical, so we can constraint $\hat{R}_{i,j}$ with below *local consistency* condition:

$$\hat{R}^*_{i,j} \in \{\hat{R}_{i,j} | KL(\mathcal{D}_i || \hat{\mathcal{D}}_i) = 0\}, \tag{3}$$

where $KL(\cdot||\cdot)$ is the Kullback-Leibler divergence between the two distributions.

On Equation (3), however, we claim that for finite $|R_i|$, there exists no solutions except all members of R_i are identical. In fact, if $\mathcal{D}_i$ and $\hat{\mathcal{D}}_i$ are equivalent, then they must have the same mean and variance values, with simple calculations, we can verify the assertion.

As the result, we relax the equality constraint of Equation (3) to Equation (4), where $\hat{R}_{i,j}$ is defined to be the one that causes the slightest changes to $\mathcal{D}_i$:

$$\hat{R}^*_{i,j} = \arg\min_{\hat{R}_{i,j}} KL(\mathcal{D}_i || \hat{\mathcal{D}}_i). \tag{4}$$

JOURNAL PRE-PROOF

Below we calculate $KL(\mathcal{D}_i||\hat{\mathcal{D}}_i)$. Let $p_i(X)$ be the p.d.f of $\mathcal{D}_i$, and $\hat{p}_i(X)$ be the p.d.f of $\hat{\mathcal{D}}_i$. We have:

$$\begin{aligned} KL(\mathcal{D}_i||\hat{\mathcal{D}}_i) &= \int_x p_i(x) \ln \frac{p_i(x)}{\hat{p}_i(x)} dx \\ &= \int_x p_i(x) \ln p_i(x) dx - \int_x p_i(x) \ln \hat{p}_i(x) dx \\ &= -H_{p_i(x)}(x) - \int_x p_i(x) \ln \hat{p}_i(x) dx \end{aligned} \tag{5}$$

Without loss of generality, we assume both $\mathcal{D}_i$ and $\hat{\mathcal{D}}_i$ are Gaussian. Denote the mean of R_i as $\bar{\mu}_{1,i}$, the variance of R_i as $\bar{\sigma}^2_{1,i}$, and let $l = |R_i|$. Then the mean of $R_i \bigcup \{\hat{R}_{i,j}\}$ is

$$\hat{\mu}_{1,i} = \frac{l\bar{\mu}_{1,i} + \hat{R}_{i,j}}{l+1} \tag{6}$$

and the variance is

$$\hat{\sigma}^2_{1,i} = \frac{l}{l+1}\bar{\sigma}^2_{1,i} + \frac{(\hat{R}_{i,j} - \bar{\mu}_{1,i})^2 + 2(\bar{\mu}_{1,i} - \hat{\mu}_{1,i})(\hat{R}_{i,j} - \bar{\mu}_{1,i})}{l+1} + \frac{(\hat{R}_{i,j} - \bar{\mu}_{1,i})^2}{(l+1)^2}. \tag{7}$$

Noting that in actual applications, the rating value is bounded, hence as $l \to \infty$, from Equation (6), we have

$$\hat{\mu}_{1,i} \approx \bar{\mu}_{1,i}. \tag{6'}$$

Take a step further, with Equations (6') and (7), we can also approximate $\hat{\sigma}^2_{1,i}$ as

$$\hat{\sigma}^2_{1,i} \approx \bar{\sigma}^2_{1,i}. \tag{7'}$$

Now with Equations (6'), (7') and the Gaussian assumption, we get

$$\hat{p}_i(x) \approx p_i(x) \tag{8}$$

for all $x \in Dom(R)$, where $Dom(R)$ is the domain of the ratings in R. Therefore, Equation (5) can be rewritten as

$$\begin{aligned} KL(\mathcal{D}_i||\hat{\mathcal{D}}_i) &= -H_{p_i(X)}(X) - \int_x p_i(x) \ln \hat{p}_i(x) dx \\ &\approx -H_{p_i(X)}(X) - \int_x \hat{p}_i(x) \ln \hat{p}_i(x) dx \\ &= -H_{p_i(X)}(X) + H_{\hat{p}_i(X)}(X). \end{aligned} \tag{9}$$

JOURNAL PRE-PROOF

In Equation (9), R_i is fixed, so $H_{p_i(X)}(X)$ is constant, therefore, with Equations (4) and (9) we have:

$$\hat{R}^*_{i,j} = \underset{\hat{R}_{i,j}}{argmin} H_{\hat{p}_i(x)}(X) \approx \underset{\hat{R}_{i,j}}{argmin} \ln \hat{\sigma}^2_{1,i}. \tag{10}$$

In other words, Equation (10) indicates that the optimum estimate of $R_{i,j}$ is the one that minimizes the variance estimate of $\hat{\mathcal{D}}_i$.

From another side, we note that the estimate of $\hat{R}_{i,j}$ can also be obtained from the *column based* generating perspective: Let $\hat{\mathcal{F}}_j$ be the distribution on $C_j \bigcup \{R_{i,j}\}$ with $\hat{q}_j(x)$ as the p.d.f, then similar to the above derivations, we attain

$$\hat{R}^*_{i,j} = \underset{\hat{R}_{i,j}}{argmin} H_{\hat{q}_j(y)}(Y) \approx \underset{\hat{R}_{i,j}}{argmin} \ln \hat{\sigma}^2_{2,j}. \tag{11}$$

Combining with Equation (10) and (11), we eventually obtain the local consistency based optimization target, as follow:

$$\hat{R}^*_{i,j} = \underset{\hat{R}_{i,j}}{argmin} \ln \hat{\sigma}^2_{1,i} + \alpha \ln \hat{\sigma}^2_{2,j}, \tag{12}$$

where $\alpha > 0$ is the trade-off parameter.

3.5. The global consistency

Following many other well established matrix completion results [4, 5, 28], we assume that the rating matrix R holds the below *uniform assumption*: For $(i, j) \in [1, n] \times [1, m]$, all $R_{i,j}$s take empty value (i.e., *missing*) with equal probability. Below we derive the global consistency.

We first calculate the total coding length of the non-zero entries in R. For clarity, when the context is clear, we also use $Len(S)$ to denote the sum of the coding lengths of the variables in the set S, i.e., $Len(S) = \sum_{e \in S} Len(e)$.

From the row based generation perspective, we have

$$\sum_{R_{i,j} \neq' \Lambda'} Len(R_{i,j}) = \sum_{i=1}^{n} Len(R_i), \tag{13}$$

where $'\Lambda'$ means the value is missing.

JOURNAL PRE-PROOF

Since $Len(R_i) = |R_i|(\frac{\sum_{R_{i,j}\neq 0} Len(R_{i,j})}{|R_i|})$, when $|R_i|$ is large enough, we have

$$\frac{\sum_{R_{i,j}\neq '\Lambda'} Len(R_{i,j})}{|R_i|} \approx H_{N(\mu_{1,i},\sigma^2_{1,i})}(X),$$

and hence Equation (13) can be reformulated as follow:

$$\sum_{R_{i,j}\neq '\Lambda'} Len(R_{i,j}) = \sum_{i=1}^{n} |R_i| H_{N(\mu_{1,i},\sigma^2_{1,i})}(X) = \sum_{i=1}^{n} \frac{1}{2}|R_i| \ln 2\pi e\sigma^2_{1,i}. \tag{14}$$

On the other side, by the uniform assumption, $|R_1| \approx |R_2| \approx \ldots \approx |R_n|$, and thus

$$\sum_{R_{i,j}\neq '\Lambda'} Len(R_{i,j}) = \frac{1}{2}|R_1| \sum_{i=1}^{n} \ln 2\pi e\sigma^2_{1,i} = \frac{n|R_1|}{2} \frac{\sum_{i=1}^{n} \ln 2\pi e\sigma^2_{1,i}}{n}. \tag{15}$$

When n is large enough,

$$\frac{\sum_{i=1}^{n} \ln 2\pi e\sigma^2_{1,i}}{n} \approx H_{N(\theta_{1,2},\kappa^2_{1,2})}(X) = \frac{1}{2} \ln 2\pi e\kappa^2_{1,2},$$

and hence from Equation (15), we achieve

$$\sum_{R_{i,j}\neq '\Lambda'} Len(R_{i,j}) = \frac{n|R_1|}{2} \ln 2\pi e\kappa^2_{1,2}. \tag{16}$$

Similar to the above process, from the column based generation perspective, we have

$$\sum_{R_{i,j}\neq '\Lambda'} Len(R_{i,j}) = \frac{m|C_1|}{2} \ln 2\pi e\kappa^2_{2,2}. \tag{17}$$

It is notable that $n|R_1| = m|C_1|$, so with Equations (16) and (17), we have the following *global consistency constraint*

$$\kappa^2_{1,2} = \kappa^2_{2,2}. \tag{18}$$

3.6. The optimization target

Combining with Equation (12) and (18), we achieve the optimization target for the prediction task, as follow:

$$\hat{R}^*_{i,j} = \underset{\hat{R}_{i,j}}{\arg min} \ln \hat{\sigma}^2_{1,i} + \alpha \ln \hat{\sigma}^2_{2,j} + \beta |\ln \hat{\kappa}^2_{1,2} - \ln \hat{\kappa}^2_{2,2}|, \tag{19}$$

JOURNAL PRE-PROOF

where α and β are trade-off parameters,

$$\hat{\sigma}_{1,i}^2 = \frac{\sum_{e \in R_i \bigcup \{\hat{R}_{i,j}\}} (e - \hat{\mu}_{1,i})^2}{|R_i| + 1} \tag{20}$$

with $\hat{\mu}_{1,i} = \frac{\sum_{e \in R_i \bigcup \{\hat{R}_{i,j}\}} e}{|R_i| + 1}$, and

$$\hat{\sigma}_{2,j}^2 = \frac{\sum_{e \in C_j \bigcup \{\hat{R}_{i,j}\}} (e - \hat{\mu}_{2,j})^2}{|C_j| + 1} \tag{21}$$

with $\hat{\mu}_{2,j} = \frac{\sum_{e \in C_j \bigcup \{\hat{R}_{i,j}\}} e}{|C_j| + 1}$.

As to $\hat{\kappa}_{1,2}^2$ and $\hat{\kappa}_{2,2}^2$, first, for $k \neq i$, we use $\sigma_{1,k}^2$ to denote the variance estimate of R_k, let $\gamma_1 = \frac{\hat{\sigma}_{1,i}^2 + \sum_{k \neq i} \sigma_{1,k}^2}{n}$, then

$$\hat{\kappa}_{1,2}^2 = \frac{(\hat{\sigma}_{1,i}^2 - \gamma_1)^2 + \sum_{k \neq i} (\sigma_{1,k}^2 - \gamma_1)^2}{n}. \tag{22}$$

And similarly, for $k \neq j$, we use $\sigma_{2,k}^2$ to denote the variance estimate of C_k, let $\gamma_2 = \frac{\hat{\sigma}_{2,j}^2 + \sum_{k \neq j} \sigma_{2,k}^2}{m}$, then

$$\hat{\kappa}_{2,2}^2 = \frac{(\hat{\sigma}_{2,j}^2 - \gamma_2)^2 + \sum_{k \neq j} (\sigma_{2,k}^2 - \gamma_2)^2}{m}. \tag{23}$$

We still have some more comments on Equation 19: The equation is a trade-off between the local and global consistency. Specifically, $\ln \hat{\sigma}_{1,i}^2$ and $\ln \hat{\sigma}_{2,j}^2$ are for the local consistency, where $\ln \hat{\sigma}_{1,i}^2$ is for the ratings given by user i, and $\ln \hat{\sigma}_{2,j}^2$ is for those targeted at item j. By minimizing the (weighted) sum of the two factors, we can prevent the estimate from being too far away from the distributions where they are drawn from. The term $|\ln \hat{\kappa}_{1,2}^2 - \ln \hat{\kappa}_{2,2}^2|$ corresponds to the global consistency. We note that according to the rating generation procedure described in Section 3.2, we have two perspectives on the ratings: The user-based perspective and the item-based perspective. To bridge the gap between the above perspectives, in Section 3.5 we derive the global consistency condition, which claims that the average rating coding lengthes of the two coding schemes are identical. Therefore, by minimizing the term $|\ln \hat{\kappa}_{1,2}^2 - \ln \hat{\kappa}_{2,2}^2|$, we obtain the guarantee that the new introduced estimate rating is the one that leads to the slightest violations to the global consistency.

JOURNAL PRE-PROOF

3.7. The adjustments

As shown in previous section, all the predictors used in the optimization target are from the distributions estimated with the observed data, so upon the true value of the predicted target (denoted as $R_{i,j}$) is revealed, the algorithm needs to update related estimates accordingly with the new observation. Noting that introducing $R_{i,j}$ will lead to changes to the four Gaussian distributions: The distribution on R_i, the distribution on C_j and the two hyper-distributions. So we have to update all the mean and variance estimates of all the four distributions.

With simple calculations, we attain the below incremental adjustments of the estimates:

$$\mu_{1,i}^{(new)} = \frac{R_{i,j} + |R_i|\mu_{1,i}^{(old)}}{|R_i|+1}, \sigma_{1,i}^{2(new)} = \frac{|R_i|(\mu_{1,i}^{(old)} - R_{i,j})^2}{(|R_i|+1)^2} + \frac{|R_i|\sigma_{1,i}^{2(old)}}{|R_i|+1}. \quad (24)$$

$$\mu_{2,j}^{(new)} = \frac{R_{i,j} + |C_j|\mu_{2,j}^{(old)}}{|C_j|+1}, \sigma_{2,j}^{2(new)} = \frac{|C_j|(\mu_{2,j}^{(old)} - R_{i,j})^2}{(|C_j|+1)^2} + \frac{|C_j|\sigma_{2,j}^{2(old)}}{|C_j|+1}. \quad (25)$$

As to the update of γ and κ, the computations are more complicated, below we only give the conclusions, and leave the details in the appendix section.

$$\gamma_1^{(new)} = \gamma_1^{(old)} + \frac{\sigma_{1,i}^{2(new)} - \sigma_{1,i}^{2(old)}}{n}, \gamma_2^{(new)} = \gamma_2^{(old)} + \frac{\sigma_{2,j}^{2(new)} - \sigma_{2,j}^{2(old)}}{m}, \quad (26)$$

$$\kappa_{1,2}^{2(new)} = \kappa_{1,2}^{2(old)} + (\gamma_1^{(new)} - \gamma_1^{(old)})(2\sigma_{1,i}^{2(new)} + (n-1)\gamma_1^{(new)} - (n+1)\gamma_1^{(old)}), \quad (27)$$

$$\kappa_{2,2}^{2(new)} = \kappa_{2,2}^{2(old)} + (\gamma_2^{(new)} - \gamma_2^{(old)})(2\sigma_{2,j}^{2(new)} + (m-1)\gamma_2^{(new)} - (m+1)\gamma_2^{(old)}). \quad (28)$$

4. The algorithm

4.1. The proposed Logo

In this section, we present *Logo*, the online CF algorithm that learning with local and global consistency.

JOURNAL PRE-PROOF

The details of *Logo* is in presented in Algorithm. 1∼ 3, where Algorithm. 1 is for predictions and model adjustments, Algorithm. 2 is used to calculate the evaluation value of Equation (19), and Algorithm. 3 is used to compute the mediate predictors according to Equation (24)∼ (28). Below we mainly focus on the first algorithm.

Algorithm 1: The proposed ***Logo*** algorithm.

Input: $[n, m, \alpha, \beta, T, r_1, r_2, \ldots, r_k]$: The number of users ($n$), the number of items ($m$), the model parameters (α and β), the max round number (T) and the candidate rating set ($\{r_1, r_2, \ldots, r_k\}$).

`/* Initialization,` μ `is for the mean,` σ^2 `is for the variance, and` c `counts the number of revealed ratings. */`

1 Use the available ratings of $R_1, R_2, \ldots, R_n$ to initialize the n row based triples $z_1^r = \langle \mu_{1,1}, \sigma_{1,1}^2, c_{1,1} \rangle, \ldots, z_n^r = \langle \mu_{1,n}, \sigma_{1,n}^2, c_{1,n} \rangle$;

2 Use the available ratings of $R_1, R_2, \ldots, R_n$ to initialize the m column based triples $z_1^c = \langle \mu_{2,1}, \sigma_{2,1}^2, c_{2,1} \rangle, \ldots, z_m^c = \langle \mu_{2,m}, \sigma_{2,m}^2, c_{2,m} \rangle$;

3 Use the z's obtained in Step 1 and 2 to initialize the hyper parameters $[h^r, h^c] = [\langle \gamma_1, \kappa_{1,2}^2 \rangle, \langle \gamma_2, \kappa_{2,2}^2 \rangle]$;

`/* Online prediction and model adjustment. */`

4 **for** $t = 1 : T$ **do**

5 **if** *Receive a query on* $R_{i,j}$ **then**

6 **for** $l = 1 : k$ **do**

7 $[\widetilde{v}_l, \widetilde{z}_l^r, \widetilde{z}_l^c, \widetilde{h}_l^r, \widetilde{h}_l^c]$=Evaluate($\alpha$, β, r_l, z_i^r, z_j^c, h^r, h^c);

8 **end**

9 Let $p = \underset{q}{argmin}\{\widetilde{v}_q | q = 1, \ldots, k\}$;

10 Response the rating requester with r_p;

11 Receive the revealed true $R_{i,j}$;

12 Let $p' = \sum_{q=1}^{k} q\mathbb{I}(R_{i,j} = r_q)$;

13 Let $z_i^r = \widetilde{z}_{p'}^r$, $z_j^c = \widetilde{z}_{p'}^c$, $h^r = \widetilde{h}_{p'}^r$, $h^c = \widetilde{h}_{p'}^c$;

14 **end**

15 **end**

In Algorithm. 1, line 1∼ 3 are for initializations, we note that for the fully

JOURNAL PRE-PROOF

cold start scheme, there aren't any ratings available, hence all the initial mean and variance estimates equal to zero. To overcome this problem, for each empty row/column, we initialize its mean to be the median value of the rating set $\{r_1, r_2, \ldots, r_k\}$, and replace all the zero-valued variance with constant 0.001. Line 6$\sim$ 10 are for predictions, in the implementation, we adopt the enumeration policy to seek the solution of Equation (19): As shown in the algorithm, we replace $\hat{R}_{i,j}$ of the equation with $r_1, r_2, \ldots, r_k$ in sequence and call the **Evaluate** function to calculate the result in accordance, and take the r that with the minimum function value (i.e., the $\widetilde{v}$ value) as the predicted estimate. Line 12 and line 13 are for model adjustments, where based on the revealed true rating, we use the corresponding predictors calculated in the prediction stage to replace their current values, respectively.

Algorithm 2: The **Evaluate** function.

```
/* Description: This function is used to calculate the value of Eq. 19
   with the input parameters, the result is stored in ṽ. After the
   calculation, it returns ṽ as well as all the mediated adjusted
   predictors.                                                    */
```

Input: $[\alpha, \beta, r, z_i^r, z_j^c, h^r, h^c]$: The model parameters ($\alpha$ and β), the rating to be evaluated (r), the z triple of the target user (z_i^r), the z triple of the target item (z_j^c) and the hyper parameters (h^r and h^c).

Output: $[\widetilde{v}, \widetilde{z}^r, \widetilde{z}^c, \widetilde{h}^r, \widetilde{h}^c]$: The evaluation result ($\widetilde{v}$) and the adjusted predictors ($\widetilde{z}^r$, $\widetilde{z}^c$, $\widetilde{h}^r$ and $\widetilde{h}^c$) based on r.

```
/* Call the Adjust function to compute the mediate adjusted predictors.  */
```

1 Let $[\widetilde{z}^r, \widetilde{h}^r]$=Adjust($r$, z_i^r, h^r, n);

2 Let $[\widetilde{z}^c, \widetilde{h}^c]$=Adjust($r$, z_j^c, h^c, m);

```
/* Calculate the evaluation value of Eq. 19 with r and the medium
   predictors.                                                    */
```

3 Let $\widetilde{v} = \ln \widetilde{z}^r.\sigma^2 + \alpha \ln \widetilde{z}^c.\sigma^2 + \beta |\ln \widetilde{h}^r.\kappa^2 - \ln \widetilde{h}^c.\kappa^2|$;

4 Return $[\widetilde{v}, \widetilde{z}^r, \widetilde{z}^c, \widetilde{h}^r, \widetilde{h}^c]$;

JOURNAL PRE-PROOF

Algorithm 3: The **Adjust** function.

```
/* Description: This function is used to calculate the mediate adjusted
   predictors with the input parameters, all are according to Eq. 24∼ 28.
   After the computations, it returns all the calculated predictors.    */
```

Input: $[r, z, h, w]$: The rating (r), the z triple (z), the hyper parameters (h) on z, and the amount of the objects (w).

Output: $[\widetilde{z}, \widetilde{h}]$: The adjusted predictors ($\widetilde{z}$ and $\widetilde{h}$) based on r.

```
/* Calculate the adjusted predictors with the input parameters according to
   Eq. 24∼ 28.                                                          */
```

Let $\widetilde{z}.c = z.c + 1$;

Let $\widetilde{z}.\mu = \dfrac{r + z.c * z.\mu}{\widetilde{z}.c}$;

Let $\widetilde{z}.\sigma^2 = \dfrac{z.c * (z.\mu - r)^2}{\widetilde{z}.c^2} + \dfrac{z.c * z.\sigma^2}{\widetilde{z}.c}$;

Let $\widetilde{h}.\gamma = h.\gamma + \dfrac{\widetilde{z}.\sigma^2 - z.\sigma^2}{w}$;

Let $\widetilde{h}.\kappa^2 = h.\kappa^2 + (\widetilde{h}.\gamma - h.\gamma)(2r + (w-1) * \widetilde{h}.\gamma - (w+1) * h.\gamma)$;

Return $[\widetilde{z}, \widetilde{h}]$;

4.2. Parameterization

There are two tuning parameters in our algorithm: α and β. Among which α trades off the row and column consistency, and β trades off the local and global consistency.

We observe that if the spatial distributions of the available sample data are imbalanced, then the effects of a rating are spatial imbalanced, too. For example, suppose $|R_i| = 1$ and $|C_j| \to \infty$, then introducing an estimate $\hat{R}_{i,j}$ may lead to significant changes to $\mathcal{D}_i$, but it can hardly have any effects on $\mathcal{F}_j$ (Noting that here we adopt the notations of Section.3.4, where $\mathcal{D}_i$ is the estimated distribution of R_i, and $\mathcal{F}_j$ is the estimated distribution of C_j). Inspired by the observations, for each target estimate $R_{i,j}$, we define the corresponding $\alpha_{i,j}$ instance as follow:

$$\alpha_{i,j} = \begin{cases} 1 & \text{if } |R_i| = 0, \\ \frac{|C_j|}{|R_i|} & \text{if } |R_i| > 0. \end{cases}$$

JOURNAL PRE-PROOF

JOURNAL PRE-PROOF

Intuitively, α can be treated as a compensate factor of the side (i.e., R_i or C_j) that with more observations. If $|R_i| < |C_j|$, then $a_{i,j} > 1$ and the impact of C_j is enhanced. Otherwise, the effects of C_j is weakened.

As to β, we don't have quantitative principles. Therefore, we explore the suitable β value via empirical tuning. As will be shown in Section 5.4, slight changes of β only lead to light variations to the algorithm performance, hence it is possible to achieve good performance with the manual tuning method.

4.3. Complexity analysis

For an n users-by m items CF systems, Algorithm. 1 needs to maintain $(n+m)$ triples(namely, $\langle \mu_{1,1}, \sigma^2_{1,1}, c_{1,1} \rangle$, ..., $\langle \mu_{1,n}, \sigma^2_{1,n}, c_{1,n} \rangle$and $\langle \mu_{2,1}, \sigma^2_{2,1}, c_{2,1} \rangle$, ..., $\langle \mu_{2,m}, \sigma^2_{2,m}, c_{2,m} \rangle$) and 2 tuples($\langle \gamma_1, \kappa^2_{1,2} \rangle$and $\langle \gamma_2, \kappa^2_{2,2} \rangle$), hence the space complexity is $\theta(n+m)$. As to the time complexity, Algorithm. 1 needs $\theta(n+m)$ computations for initialization. Then for each rating query, it needs k trials to produce the answer rating, Besides, after the true rating is revealed, the algorithm needs constant steps to adjust the learned model.

5. Experiments and discussions

All our experiments are conducted on a desktop with an Intel 6 cores CPU (Intel i7 5930K) and 32G RAM. Below we will first introduce the settings of the experiments, including the data, the benchmark algorithms and the evaluation metrics, then we will present the results and related discussions.

5.1. The data

We evaluate the algorithms with four real data sets: Amazon[4][20], Douban[5][39], Jester[6][11] and Movielens[7]. Among all these four sets, Amazon, Douban and

[4]http://konect.uni-koblenz.de/networks/amazon-ratings
[5]http://socialcomputing.asu.edu/datasets/Douban
[6]http://eigentaste.berkeley.edu/dataset/
[7]https://grouplens.org/datasets/movielens/

Movielens take discrete valued ratings, while Jester takes continuous valued ratings from the interval [-10, 10]. We discretize Jester's ratings by dividing the interval [-10, 10] into 10 disjoint equal length bins [-10, -8), [-8, -6), ..., [6, 8) and [8, 10], and indexing the bins with 1, 2, ..., 10, respectively. Therefore, each rating of the Jester set is mapped to the value which indexes the bin it falls in.

In Table. 2, we summarize some statistics of the experimental data.

Table 2: Data characteristics.

Data set	# Users	# Items	# Observations	Density	Domain
Amazon	19,293	20,000	570,318	$1.5 \times 10^{-3}\%$	$\{1, 2, 3, 4, 5\}$
Douban	29,999	20,000	1,140,521	$1.9 \times 10^{-3}\%$	$\{1, 2, 3, 4, 5\}$
MovieLens	2,113	10,109	855,598	$4.01 \times 10^{-2}\%$	$\{0.5, 1, 1.5, \ldots, 5\}$
Jester	24,983	100	1,810,455	77.04%	$\{1, 2, \ldots, 10\}$

From Table.2, we see that both Amazon and Douban have similar characteristics: They have the same item amount and rating domain, and their user amounts and data densities are in the same magnitude, respectively. We use these two sets to evaluate the algorithms' stability of prediction performance. As to MovieLens and Jester, we see they're much more denser: MovieLens is about 10 times denser than Douban, while Jester is nearly 200 times denser than MovieLens. Therefore, with the prediction results obtained on the all sets, we can analyze the algorithms' performances under different data density settings.

5.2. The comparison algorithms

We compare *Logo* with five state-of-the-art CF algorithms, among which three are online algorithms, and two others are offline algorithms.

The online algorithms are the Sparse Online Collaborative Filtering (SOCF) algorithm [21], the online Probabilistic Matrix Factorization algorithm (OLPMF) [23] and the Online Passive-Aggressive Non-Negative Matrix Factorization algorithm (OLPA) [3]. As discussed in Section. 2.2, SOCF is a sparse model, which pays special attention to the sparseness of the object representations. OLPMF

JOURNAL PRE-PROOF

is a dense model, it mainly stems from the classic offline algorithm PMF [29]. OLPA also belongs to the dense model, but it's based on the online algorithm PA [8].

The offline algorithms are the similarity-based collaborative filtering model (Sim) [1] and the Probabilistic Matrix Factorization model (PMF) [29]. Both of the algorithms are among the most classic CF algorithms, Sim focus on making use of the explicit rating information and the other related side information, while PMF tries to exploit the implicit latent feature representations.

We determine all the comparison algorithms' parameter settings via coordinately changing their values with exponential incremental steps on the experiment data sets. For example, in PMF there're two tuning parameters: the learning rate τ and the regularized coefficient λ. To exploit the suitable configurations of PMF on Amazon, we first randomly sample 10% ratings from Amazon to constitute a smaller set A, then we sequentially apply the configurations $(\tau, \lambda) = (0.01, 0.01 \times 2^{n-1})$ $(n = 1, 2, \dots)$ to PMF to perform predictions on A, when the best result (i.e., the minimum prediction error) is attained, we let $\dot{\lambda} = \lambda$, and apply the configurations $(\tau, \lambda) = (0.01 \times 2^{n-1}, \dot{\lambda})$ $(n = 1, 2, \dots)$ sequentially to PMF and perform predictions until the minimum error occurs, denote the corresponding τ as $\dot{\tau}$, then we adopt $(\dot{\tau}, \dot{\lambda})$ as the parameter setting of PMF in the comparison experiments on Amazon.

5.3. The evaluation metrics

The Root Mean Square Error (RMSE) and the Mean Absolute Error (MAE) are the two often used error metrics for CF algorithm evaluations [16, 29, 31, 37]. Many existing studies have shown that these two metrics behave similar to each other in the applications, therefore, we only report the RMSE results, where denote $S = \{s_1, s_2, \dots s_n\}$ as the true data, and $\hat{S} = \{\hat{s_1}, \hat{s_2}, \dots \hat{s_n}\}$ the corresponding estimates, then the RMSE is given by

$$\sqrt{\frac{1}{n}\sum_{k=1}^{n}(s_k - \hat{s_k})^2}. \tag{29}$$

JOURNAL PRE-PROOF

An intrinsic limitation of the conventional RMSE metric lies in its batch processing oriented nature, even the model performance has been boosted greatly after a long time training, the error may be still dominated by the historic accumulated not-so-good predictions. Aiming at this limitation, we propose the below Up-to-Date Error (UtDE) metric: Denote $s_1, s_2, \ldots$ the ratings to be predicted and $\hat{s}_1, \hat{s}_2, \ldots$ the corresponding estimates, we divide the data into equal size blocks $B_1, B_2, \ldots$, where $|B_1| = |B_2| = \cdots = l$ and the ith element of B_j is $(s_{(j-1)*l+i}, \hat{s}_{(j-1)*l+i})$. Denote the RMSE on B_j as z_j, then z_j the jth l-UtDE of the predictions.

According to the definition, it's clear that the UtDE only reflects the current prediction performance of the learned model, specifically, for $j = 1, 2, \ldots$, the jth UtDE only takes the prediction errors of $s_{(j-1)*l+1}, s_{(j-1)*l+2}, \ldots, s_{j*l}$ into account, therefore, compared with the conventional RMSE, UtDE is more suitable to describe the *evolution* of the online model.

5.4. Parameter sensitivity analysis

To analyze the parameter sensitivity of *Logo*, we first constitute four evaluation sets from the experimental sets (i.e., Amazon, Douban, MovieLens and Jester), where each evaluation set consists of 10% randomly selected samples from the corresponding experimental set[8]. Then in Algorithm.1, we set $\beta = 0.01 \times 2^i$ for $i = 0, 2, \ldots, 10$ in sequence and use the algorithm to perform predictions, respectively. We summarize all the RMSE results in Figure. 3.

From Figure. 3, we see that the algorithm performance is very stable with respect to β: Even the β value varies in a very wide range (noting that in the interval [0.01, 10.24], the right end value is 1,000 times larger than the left end), the corresponding RMSE value just fluctuates in a narrow interval.

In addition, we also study the results quantitatively. On each evaluation set

[8]For convenience, in this subsection, we refer to each evaluation set with the same name of the experimental set where it's sampled from.

JOURNAL PRE-PROOF

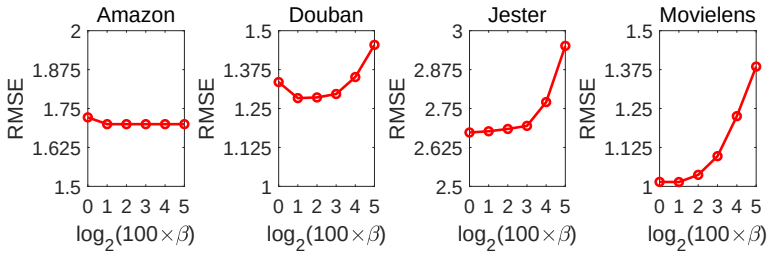


Figure 3: Results of parameter sensitivity analysis, where all experiments are conducted on the parameter evaluation sets.

D, we define the Max Relative Error Ratio (MRER) as follow:

$$MRER_D = \frac{error^D_{max} - error^D_{min}}{error^D_{min}}. \tag{30}$$

Where $error^D_{max}$ and $error^D_{min}$ are the max and min errors obtained by *Logo* with varied β's on D, respectively.

Table 3: The MRER values of *Logo* on the parameter evaluation data sets (smaller means more stable).

	Amazon	Douban	Jester	MovieLens
MRER	0.0128	0.1333	0.1041	0.3657

Table.3 reports the MRER values of *Logo* on the four parameter evaluation sets, where we see that among all the experiments, the largest MRER value is not to 0.4. This value, taking the the changes of the magnitude of the β value into account, is so small that can be even neglected. Therefore, the empirical studies demonstrate the stability of *Logo*.

5.5. Comparisons with the online algorithms

In this section, we report the online prediction performance of *Logo* and the other three comparison algorithms, i.e., OLPMF, OLPA and SOCF. We note that all the experiments are conducted 3 times, and all the results presented here are the mean values.

JOURNAL PRE-PROOF

JOURNAL PRE-PROOF

We first compare the algorithms on the prediction accuracy. Table. 4 reports the RMSEs obtained by *Logo* and the comparison algorithms. We see that *Logo* maintains superior performances compared to other algorithms, on each data set, the RMSE of *Logo* is just nearly half of the smallest correspondence of the comparison algorithms. Our interpretation of the distinction is as follow: For the comparison algorithms, noting that they are all factor models, where every (latent) user feature factor has equal effects on all the ratings given by the user, and every (latent) item feature factor has equal effects on all the ratings of the item too, hence all the factors are *global*. While on the other side, in the online learning setting, the learned model is adjusted in the round-by-round manner, where in each round the adjustment is only with respect to the newly arrived individual sample, which mainly carries the *local* information. Therefore, it's unreasonable to expect that the global model can be well trained with only the local data. For our proposed *Logo*, as implied in its name, the algorithm uses both of the local and global information to train the model, so even an individual rating can help to improve the prediction performance.

From Table. 4, we also find that on the similar data sets Amazon and Douban, all the four algorithms achieve smaller errors on the latter. This observation can be easily explained with the statistics in Table. 2, where it's shown that the density of Douban is about 30% larger than that of Amazon. What's more, it's worthy to note that from Amazon to Douban, *Logo* has reduced its prediction error by more than 25%, while for the other algorithms, none of them reduces the error by more than 10%. Therefore, the comparisons provide some evidences that compared with the benchmark algorithms, *Logo* works better in exploiting the potential of the data.

Figure. 4 shows the UtDEs of all the algorithms, where we set the block size $l = 10,000$. From Figure. 4, we see that *Logo* achieves the best performance on all the four data sets. Besides, Figure. 4 also shows that the convergence speed of *Logo* is very fast: In all the experiments, the algorithm converges in the first 20% blocks.

In addition to the above, we also study the composition of the prediction

Table 4: The algorithms' prediction errors in the online settings (all errors are measured in RMSE, smaller is better).

Algorithm	Amazon	Douban	Jester	MovieLens
Logo	**1.2060**	**0.8849**	**2.4071**	**0.9113**
OLPMF	2.9648	2.8149	5.4074	2.2688
OLPA	2.1674	1.9751	6.4360	2.0407
SOCF	3.1737	3.0497	5.3308	3.3786

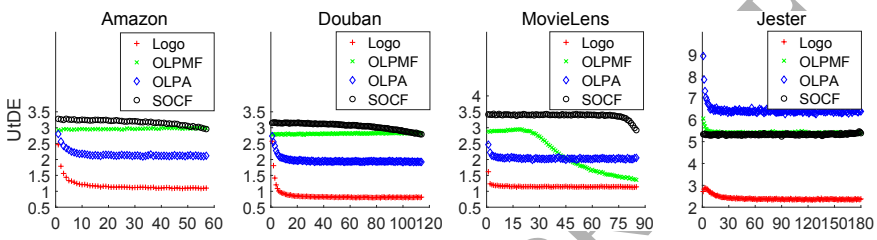


Figure 4: The UtDEs results of the experiments, where we take $l = 10,000$.

results. Denote the true rating as r^{true}, and the corresponding prediction as r^{pred}, we categorize the predicted results into the below three classes:

1. The exact predictions, where $r^{pred} = r^{true}$;
2. The close predictions, where $|r^{pred} - r^{true}| = 1$ (For MovieLens, $|r^{pred} - r^{true}| = 0.5$);
3. Others.

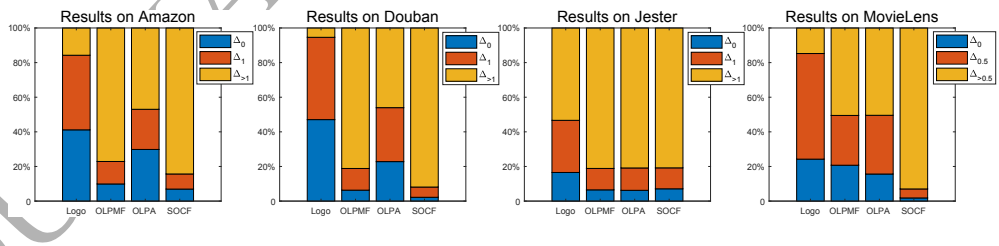


Figure 5: The compositions of the prediction results in the online settings, where Δ_x is for $|r^{pred} - r^{true}| = x$ ($x \in \{0, 0.5, 1\}$), and $\Delta_{>y}$ is for $|r^{pred} - r^{true}| > y$ ($y \in \{0.5, 1\}$).

Figure. 5 reports the compositions of the predictions. As shown in the figure, on Amazon, Douban and MovieLens, more than 80% of *Logo*'s predictions are

JOURNAL PRE-PROOF

very close to the true ratings, specifically, on Amazon and Douban, *Logo* has more than 40% exact predictions. We also find that on Jester, *Logo* just has not to 50% exact or close predictions of the all, but it's worthy to note that in Jester, every rating can take 10 possible values, therefore, the greater prediction bias could be due to the smaller granularity of ratings.

Our last comparison is on the running time of the algorithms. Table. 5 reports the time that the algorithms used in the experiments, we see that *Logo* is nearly two order of magnitude faster than the others. The speed performance of our algorithm can be due to the efficient implementation. As shown in Algorithm.1 and related discussions therein, a fully "predicting-updating" round of the algorithm only needs of $\theta(k)$ computations, where k is the size of the candidate rating set, in the experiments, $k = 5$ for Amazon and Douban, and $k = 10$ for MovieLens and Jester. Hence to apply our algorithm to predict $\mathcal{N}$ ratings, it just needs $\theta(k\mathcal{N})$ computations. In addition, noting that k is constant for specific applications, so the time complexity can also be treated as $\theta(\mathcal{N})$. That is, to produce a rating prediction, our algorithm only needs constant time.

Table 5: The running time of the algorithms in the online prediction experiments (in seconds).

	Amazon	Douban	MovieLens	Jester
Logo	**13**	**24**	**30**	**69**
OLPMF	1440	3276	1908	4500
OLPA	2160	4536	3204	7056
SOCF	1620	3420	2016	5076

5.6. Comparisons with the offline algorithms

In this section we report the experiment results under the offline settings. As mentioned above, we adopt PMF and Sim as the competitive algorithms of *Logo*.

We're first concerned on the testing protocol. There're many testing protocols for offline algorithm evaluations, such as the k-fold cross-validation protocol, the leave-one-out cross-validation protocol [6], and the *A-Test* protocol [10]. In

this work, we adopt the "*Given x*" ($0 < x < 1$) protocol, which is widely used to test the performance of the offline CF algorithms [26, 29, 15, 18, 24, 32, 36]. Given a rating set R, *Given x* means only x ratio of R are used to train the model, while the remained data are concealed to evaluate the learned result. In our experiments, we adopt "*Given x*" as the testing protocol. Specifically, for *Logo*, "*Given x*" is similar to the warm start setting, where the predictors of *Logo* will be first initialized with x ratio of R before being put into use, but after initialization, the learned model will just be used to perform predictions, while can't be updated anymore.

We first study the prediction accuracy of the algorithms.

Table. 6 and Table. 7 reports the RMSE results of the offline prediction experiments. In summary, on the all data sets, *Logo* performs almost as well as the best comparison algorithms. Specifically, it's worthy to note that on Amazon and Douban, *Logo* achieves the best performances under the *Given 30%* setting.

Table 6: The offline prediction errors on Amazon and Douban (all errors are measured in RMSE, smaller is better).

	Amazon			Douban		
Algorithm	Given 30%	Given 60%	Given 90%	Given 30%	Given 60%	Given 90%
Logo	**1.1538**	1.1089	1.0978	**0.8247**	0.8153	0.8098
PMF	1.1779	1.1748	1.1728	0.8459	0.8454	0.8465
Sim	1.2027	**1.0798**	**1.0229**	0.8379	**0.7843**	**0.7426**

Table 7: The offline prediction errors on MovieLens and Jester (all errors are measured in RMSE, smaller is better).

	Jester			MovieLens		
Algorithm	Given 30%	Given 60%	Given 90%	Given 30%	Given 60%	Given 90%
Logo	2.3759	2.3597	2.3590	0.8992	0.8884	0.8829
PMF	2.3598	**2.1210**	**2.1091**	0.9282	0.9083	0.9173
Sim	**2.2166**	2.1822	2.1746	**0.8320**	**0.8205**	**0.8112**

JOURNAL PRE-PROOF

In addition to the above observations, we also study the result compositions of the experiments. Noting that the experiments are conducted under the *Given* x protocol with different x settings, for brevity, we only report the results of *Given 90%*, as shown in Figure. 6.

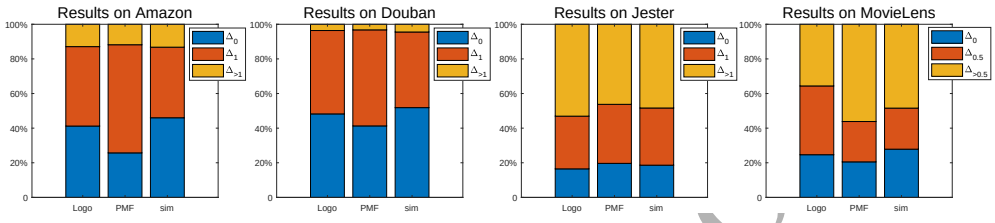


Figure 6: The compositions of the prediction results in the offline settings with the *Given 90%* protocol, where Δ_x is for $|r^{pred} - r^{true}| = x$ ($x \in \{0, 0.5, 1\}$), and $\Delta_{>y}$ is for $|r^{pred} - r^{true}| > y$ ($y \in \{0.5, 1\}$).

In Figure. 6 we adopt the same categories used in Figure. 5. We see that *Logo* produces about 40% exact predictions on Amazon and Douban, and about another 40% close predictions on Amazon, Douban and MovieLens. As to Jester, all the competing algorithms can't perform as well as they do on the other three data sets, however, *Logo* still achieves more than 40% exact or close predictions.

Our next concern is with the running time of the algorithms. Again for brevity, we only present the results of the *Given 90%* setting.

Table 8: The running time of the algorithms in the *Given 90%* offline prediction experiments (in seconds).

	Amazon	Douban	MovieLens	Jester
Logo	**6**	**12**	**9**	**19**
PMF	92	197	132	300
Sim	2998	6287	1245	2929

The running time statistics are presented in Table. 8, where we see that *Logo* is faster than PMF by about an order of magnitude, and faster than Sim by about two order of magnitude. Besides, it's noteworthy that compared with

JOURNAL PRE-PROOF

JOURNAL PRE-PROOF

Table. 5 and Table. 8, the time that *Logo* used in the offline settings is far less than its correspondences in the online settings. As to the reason, we see that when Algorithm. 1 runs in the online setting, in each "prediction-adjustment" round, it needs k trials to produce a prediction, and uses one extra step to update the model. When the algorithm runs under the offline setting, since "x" ratio of the samples are given, it can use the given data to initialize the model directly, therefore, the corresponding k-trial prediction cost is saved. Specifically, as our issue is concerned, noting that the *Given 90%* protocol means that only 10% of the data needs the k-trial computations, so we have the observations.

In summary, we conclude that in terms of the prediction accuracy, our proposed algorithm doesn't show overwhelming advantages over the others. However, we emphasize that in the actual CF applications, both of the prediction requests and the new revealed truths are streamed in, so it's crucial to response to the requests and truths in realtime. Taking both of the accuracy and efficiency issues into account, *Logo* can also be an appropriate alternative for the offline CF applications.

6. Conclusion

The collaborative filtering algorithms are widely deployed in various recommender systems. However, most of the existing CF studies are on offline algorithms, where before the models are put into providing prediction services online, they have to be trained with a large volume of historic data offline. In addition to the time consuming training process, the algorithms also suffer from the model update problem, due to the high time complexity to use the newly user feedbacks to adjust the learned models.

In this work, we focus in developing the online CF algorithm, our major contributions are summarized as follow:

Firstly, we present a generative model for the *user-item* ratings, where we assume that given a specific object, all its ratings are n.i.i.d. Besides, we also assumes that given a set of homogeneous objects, all their corresponding dis-

JOURNAL PRE-PROOF

tributions are from the same hyper distribution. With the assumptions, we derive the local and global consistency constraints, respectively, and formulate the prediction task as an optimization problem.

Secondly, we propose *Logo*, the solution algorithm to the above optimization problem. As discussed in the text, the time complexity of our algorithm is $\theta(k\mathcal{N})$, where k is the size of the candidate rating set, and $\mathcal{N}$ is the amount of prediction requests. Therefore, to make a prediction, the algorithm only needs $\theta(k)$ computations. What's more, it's noteworthy that as a by-product, during the predicting procedure we also obtain the updated predictors, which can be used to adjust the learned model directly.

Thirdly, we conduct comprehensive experiments to evaluate the proposed method. The results show that our algorithm outperforms the benchmark online algorithms significantly in both of prediction accuracy and running efficiency. Besides, even compared with the offline algorithms, our algorithm also obtains comparable accurate results, while with tens or even hundreds times faster running speeds.

As to the future works, we note the presented work is an initial step toward the generative online CF models, there're still many extensions to explore, among which, we believe that the below two directions are worthy of special attentions:

First, it's shown by Koren et.al. [18] that the CF predictions can benefit from the implicit user feedbacks and explicit side information (e.g., the rating date information), however, in the present model, only the rating data are used. Therefore, it's interesting to study how to incorporate more richer structures into the models, such as the user side information, the item side information and the date information, etc.

Second, in the proposed generative model, all the rating vectors are assumed to be n.i.i.d. This assumption may have weaken the data diversity more or less, for each user can have hybrid interests: She may have strong preferences on some kinds of items, while doesn't like some other kinds very much. Hence a more reasonable assumption may be that each rating vector is drawn from the

JOURNAL PRE-PROOF

mixture of some distributions, and as a result, the solution algorithms for this setting are worthy to explore.

Compliance with Ethical Standards

Funding

This work is supported by the National Key R&D Program of China (2017YFB1401100), the Welfare Research and Capacity Building Special Foundation of Guangdong Province(2015A030402003), the Science and Technology Foundation of Guangdong Province (2016ZC0039, 2017ZC0117), and the Philosophy and Social Science Foundation of Guangdong Province (GD15CGL05).

Conflict of Interest

The authors declares that they have no conflict of interest.

Ethical approval

This article does not contain any studies with human participants or animals performed by any of the authors.

References

[1] Charu C Aggarwal. Neighborhood-based collaborative filtering. In *Recommender Systems*, pages 29–70. Springer, 2016.

[2] Murugan Anandarajan, Chelsey Hill, and Thomas Nolan. Probabilistic topic models. In *Practical Text Analytics*, pages 117–130. Springer, 2019.

[3] Mathieu Blondel, Yotaro Kubo, and Ueda Naonori. Online Passive-Aggressive Algorithms for Non-Negative Matrix Factorization and Com-

JOURNAL PRE-PROOF

JOURNAL PRE-PROOF

pletion. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, pages 96–104. PMLR, 2014.

[4] Emmanuel J Candès and Benjamin Recht. Exact Matrix Completion via Convex Optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

[5] Emmanuel J Candès and Terence Tao. The Power of Convex Relaxation: Near-optimal Matrix Completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080, 2010.

[6] Gavin C Cawley and Nicola LC Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11(Jul):2079–2107, 2010.

[7] Thomas M Cover and Joy A Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012.

[8] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar):551–585, 2006.

[9] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[10] Arash Gharehbaghi and Maria Lindén. A deep machine learning method for classifying cyclic time series of biological signals using time-growing neural network. *IEEE transactions on neural networks and learning systems*, 29(9):4102–4115, 2018.

[11] Ken Goldberg, Theresa M Roeder, Dhruv Gupta, and C Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[12] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. Fast matrix factorization for online recommendation with implicit feedback. In

JOURNAL PRE-PROOF

Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, pages 549–558. ACM, 2016.

[13] Antonio Hernando, Jesús Bobadilla, and Fernando Ortega. A non negative matrix factorization for collaborative filtering recommender systems based on a bayesian probabilistic model. *Knowledge-Based Systems*, 97(4):188–202, 2016.

[14] Daniel Kluver, Michael D Ekstrand, and Joseph A Konstan. Rating-based collaborative filtering: algorithms and evaluation. In *Social Information Access*, number 5, pages 344–390. Springer, 2018.

[15] Yehuda Koren. Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 426–434. ACM, 2008.

[16] Yehuda Koren. Collaborative Filtering with Temporal Dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[17] Yehuda Koren and Robert Bell. Advances in collaborative filtering. In *Recommender systems handbook*, pages 77–118. Springer, 2015.

[18] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix Factorization Techniques for Rcommender Systems. *Computer*, 42(8):30–37, 2009.

[19] Vikas Kumar, Arun K Pujari, Sandeep Kumar Sahu, Venkateswara Rao Kagita, and Vineet Padmanabhan. Collaborative filtering using multiple binary maximum margin matrix factorizations. *Information Sciences*, 380(2):1–11, 2017.

[20] Jérôme Kunegis. Konect: the koblenz network collection. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 1343–1350. ACM, 2013.

JOURNAL PRE-PROOF

JOURNAL PRE-PROOF

[21] Fan Lin, Xiuze Zhou, and Wenhua Zeng. Sparse online learning for collaborative filtering. *International Journal of Computers Communications & Control*, 11(2):248–258, 2016.

[22] Tong Lin and Hongbin Zha. Riemannian manifold learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5):796–809, 2008.

[23] Guang Ling, Haiqin Yang, Irwin King, and Michael R Lyu. Online Learning for Collaborative Filtering. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2012.

[24] Xin Luo, Mengchu Zhou, Yunni Xia, and Qingsheng Zhu. An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems. *IEEE Transactions on Industrial Informatics*, 10(2):1273–1284, 2014.

[25] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11(1):19–60, 2010.

[26] Benjamin Marlin. *Collaborative Filtering: a Machine Learning Perspective*. PhD thesis, University of Toronto, 2004.

[27] Netflix. Netflix Prize, 2018. `http://www.netflixprize.com`, visited December, 2018.

[28] Benjamin Recht, Maryam Fazel, and Pablo A Parrilo. Guaranteed Minimum-rank Solutions of Linear Matrix Equations via Nuclear Norm Minimization. *SIAM review*, 52(3):471–501, 2010.

[29] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic Matrix Factorization. *Advances in Neural Information Processing Systems*, 20:1257–1264, 2008.

[30] Badrul Munir Sarwar, George Karypis, Joseph A Konstan, John Riedl, et al. Item-based collaborative filtering recommendation algorithms. In

Proceedings of the 10th International ACM World Wide Web conference, pages 285–295. ACM, 2001.

[31] Nathan Srebro, Jason Rennie, and Tommi S. Jaakkola. Maximum-margin matrix factorization. In *Proceedings of the 17th Conference on Neural Information Processing Systems*, pages 1329–1336. MIT Press, 2005.

[32] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1235–1244. ACM, 2015.

[33] Jialei Wang, Steven CH Hoi, Peilin Zhao, and Zhi-Yong Liu. Online multi-task collaborative filtering for on-the-fly recommender systems. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 237–244. ACM, 2013.

[34] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 839–848. ACM, 2018.

[35] Yong Wang, Jiangzhou Deng, Jerry Gao, and Pu Zhang. A hybrid user similarity model for collaborative filtering. *Information Sciences*, 418(12):102–118, 2017.

[36] Jian Wei, Jianhua He, Kai Chen, Yi Zhou, and Zuoyin Tang. Collaborative filtering and deep learning based recommendation system for cold start items. *Expert Systems with Applications*, 69(3):29–39, 2017.

[37] Minjie Xu, Jun Zhu, and Bo Zhang. Fast max-margin matrix factorization with data augmentation. In *Proceedings of the 30th International Conference on Machine Learning*, pages 978–986. PMLR, 2013.

JOURNAL PRE-PROOF

[38] Xiwang Yang, Yang Guo, Yong Liu, and Harald Steck. A survey of collaborative filtering based social recommender systems. *Computer Communications*, 41(3):1–10, 2014.

[39] R. Zafarani and H. Liu. Social computing data repository at ASU, 2009.

Appendix. Detailed derivations of the κ adjustment equations(Equation (26) and Equation (27))

In this appendix, we present the detailed derivations of Equation (26) and Equation (27). Noting that Equation (27) and Equation (28) are of almost the same, therefore, all the below derivations are also applicable for Equation (28) with minor notation modifications.

Adopting the same notations used in Sec.3.5 and Sec.3.7, we first state the detailed results of Equation (26) and Equation (27) as bellow:

Given the incomplete rating matrix $R \in \mathbb{R}^{n\times m}$, for $k = 1, 2, \ldots, n$, let R_k be the kth row of R, denote the mean and variance of the non-zero ratings of R_k as $\mu_{1,k}$ and $\sigma^2_{1,k}$, and the mean and variance of all the σ^2_ks as γ and κ^2, respectively. For two matrices R and $R\bigcup\{R_{i,j}\}$, we use $t^{(old)}$ to denote an estimate t from R, use $t^{(new)}$ to denote the corresponding estimate from $R\bigcup\{R_{i,j}\}$, then

$$\gamma^{(new)} = \gamma^{(old)} + \frac{\sigma^{2(new)}_{1,i} - \sigma^{2(old)}_{1,i}}{n}, \tag{31}$$

and

$$\kappa^{2(new)} = \kappa^{2(old)} + (\gamma^{(new)} - \gamma^{(old)})(2\sigma^{2(new)}_{1,i} + (n-1)\gamma^{(new)} - (n+1)\gamma^{(old)}). \tag{32}$$

Proof. First, according to the definition of γ,

$$\begin{aligned}
\gamma^{(new)} &= \frac{1}{n}(\sigma^{2(new)}_{1,i} + \sum_{l\neq i}\sigma^{2(old)}_{1,l}) \\
&= \frac{1}{n}(\sigma^{2(new)}_{1,i} + (n\gamma^{(old)} - \sigma^{2(old)}_{1,i})) \\
&= \gamma^{(old)} + \frac{\sigma^{2(new)}_{1,i} - \sigma^{2(old)}_{1,i}}{n},
\end{aligned}$$

JOURNAL PRE-PROOF

hence Equation (6) is achieved.

As to Equation (32), we have

$$
\begin{aligned}
\kappa^{2(new)} &= \frac{1}{n}\sum(\sigma_{1,k}^{2(new)} - \gamma^{(new)})^2 \\
&= \frac{1}{n}\sum[(\sigma_{1,k}^{2(new)} - \gamma^{(old)}) + (\gamma^{(old)} - \gamma^{(new)})]^2 \\
&= \frac{1}{n}\sum[(\sigma_{1,k}^{2(new)} - \gamma^{(old)})^2 + (\gamma^{(old)} - \gamma^{(new)})^2 \\
&\qquad + 2(\gamma^{(old)} - \gamma^{(new)})(\sigma_{1,k}^{2(new)} - \gamma^{(old)})] \\
&= (\gamma^{(old)} - \gamma^{(new)})^2 + \frac{1}{n}[(\sigma_{1,i}^{2(new)} - \gamma^{(old)})^2 + \sum_{k\neq i}(\sigma_{1,k}^{2(new)} - \gamma^{(old)})^2] \\
&\quad + \frac{2}{n}(\gamma^{(old)} - \gamma^{(new)})[(\sigma_{1,i}^{2(new)} - \gamma^{(old)}) + \sum_{k\neq i}(\sigma_{1,k}^{2(new)} - \gamma^{(old)})] \\
&= (\gamma^{(old)} - \gamma^{(new)})^2 + \frac{1}{n}[(\sigma_{1,i}^{2(new)} - \gamma^{(old)})^2 + \sum_{k\neq i}(\sigma_{1,k}^{2(old)} - \gamma^{(old)})^2] \\
&\quad + \frac{2}{n}(\gamma^{(old)} - \gamma^{(new)})[(\sigma_{1,i}^{2(new)} - \gamma^{(old)}) + \sum_{k\neq i}(\sigma_{1,k}^{2(old)} - \gamma^{(old)})] \\
&= (\gamma^{(old)} - \gamma^{(new)})^2 + \frac{1}{n}[(\sigma_{1,i}^{2(new)} - \gamma^{(old)})^2 + n\kappa^{2(old)} - (\sigma_{1,i}^{2(old)} - \gamma^{(old)})^2] \\
&\quad + \frac{2}{n}(\gamma^{(old)} - \gamma^{(new)})[(\sigma_{1,i}^{2(new)} - \gamma^{(old)}) + (\gamma^{(old)} - \sigma_{1,i}^{2(old)})] \\
&= \kappa^{2(old)} + (\gamma^{(old)} - \gamma^{(new)})^2 + \frac{1}{n}[(\sigma_{1,i}^{2(new)} - \gamma^{(old)})^2 - (\sigma_{1,i}^{2(old)} - \gamma^{(old)})^2] \\
&\quad + \frac{2}{n}(\gamma^{(old)} - \gamma^{(new)})[(\sigma_{1,i}^{2(new)} - \sigma_{1,i}^{2(old)})] \\
&= \kappa^{2(old)} + (\gamma^{(old)} - \gamma^{(new)})^2 + \frac{1}{n}[(\sigma_{1,i}^{2(new)} - \gamma^{(old)})^2 - (\sigma_{1,i}^{2(old)} - \gamma^{(old)})^2] \\
&\quad - 2(\gamma^{(old)} - \gamma^{(new)})^2 \\
&= \kappa^{2(old)} - (\gamma^{(old)} - \gamma^{(new)})^2 + \frac{1}{n}[(\sigma_{1,i}^{2(new)} + \sigma_{1,i}^{2(old)} - 2\gamma^{(old)}) \\
&\quad \times (\sigma_{1,i}^{2(new)} - \sigma_{1,i}^{2(old)})]. \qquad (33)
\end{aligned}
$$

Noting that with Equation (6), we have

$$n(\gamma^{(new)} - \gamma^{(old)}) = \sigma_{1,i}^{2(new)} - \sigma_{1,i}^{2(old)},$$

we apply the above relation to Equation (33) and rearrange the items, then Equation (32) is attained.

JOURNAL PRE-PROOF