# Online Learning for Collaborative Filtering

Guang Ling*, Haiqin Yang*, Irwin King*†, Michael R. Lyu*
*Department of Computer Science and Engineering
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
{gling, hqyang, king, lyu}@cse.cuhk.edu.hk
†AT&T Labs Research, Forlham Park, NJ, USA,
irwin@research.att.com

***Abstract*—*Collaborative filtering* (CF), aiming at predicting users' unknown preferences based on observational preferences from some users, has become one of the most successful methods to building recommender systems. Various approaches to CF have been proposed in this area, but seldom do they consider the dynamic scenarios: 1) new items arriving in the system, 2) new users joining the system; or 3) new rating updating the system are all dynamically obtained with respect to time. To capture these changes, in this paper, we develop an online learning framework for collaborative filtering. Specifically, we construct this framework consisting of two state-of-the-art matrix factorization based CF methods: the probabilistic matrix factorization and the top-one probability based ranking matrix factorization. Moreover, we demonstrate that the proposed online algorithms bring several attractive advantages: 1) they scale linearly with the number of observed ratings and the size of latent features; 2) they obviate the need to load all ratings in memory; 3) they can adapt to new ratings easily. Finally, we conduct a series of detailed experiments on real-world datasets to demonstrate the merits of the proposed online learning algorithms under various settings.**

## I. Introduction

With the emergence of large-scale online user-contributed websites and online shopping websites, e.g., Amazon, IMDB, etc., users are presented with unprecedentedly large amount of items. On one hand, users can easily get stuck in the information-overloading problem, and how to select favorite items from millions of options becomes a major bottleneck. On the other hand, it is important for vendors to find out users' preferences so as to boost sales. Recommender systems, aiming at selecting attractive items for users, become one promising technology to resolve the aforementioned problems.

Collaborative filtering (CF) methods are one of the major approaches to recommender system. In CF, users are allowed to rate the items as an indicator of preference. These ratings, in which users' preference patterns and items' specific features embed, are collected to make predictions. Traditional collaborative filtering methods adopt batch-trained algorithms. These methods suffer from two major drawbacks. First, they scale poorly. The nature of many CF algorithms make them hard to be parallelized and recently there are a few works try to investigate this approach. Before training, they require that all data are available. During training, at each iteration, all ratings must be scanned through once to perform the algorithms. This is very expensive since real-world datasets cannot be loaded into the memory easily. The second drawback of batch-trained algorithms is that they are unsuitable for dynamic ratings. A recommender system may change in one of the following ways: 1) a new user may join the system and rate existing items; 2) a new item may appear in the system and existing users may purchase and rate it; and 3) existing users may purchase and rate existing items; in other words, ratings are collected over time. In these cases, to capture the change, the batch-trained CF methods have to rebuild the model, which is very expensive.

Online learning algorithms, as an alternative to parallel algorithm, emerge as a natural solution to attack the incremental rating problem. In the literature, although there are several tasks [1], [2], [3], [4], [5] investigating online learning for collaborative filtering, they did not explore the complete properties of online algorithms. In addition, none of previous work considers ranking-oriented collaborative filtering. Hence, in this paper, we study online algorithms from various aspects to solve the issues facing batch-trained CF algorithms and previously proposed online learning algorithms. Our contributions include:

- We apply the proposed online learning framework to both rating-oriented *matrix factorization* (MF) based methods (PMF) and ranking-oriented MF-based methods (RMF). Our proposed online learning algorithms can handle new rating incrementally without retraining the models. To our best knowledge, this is the first attempt to develop online algorithms for ranking-oriented CF methods.
- We develop the online learning algorithms employing two approaches, stochastic gradient descent and the dual-average method. We provide succinct and efficient solutions to both of them. All the online algorithms scale linearly with the number of observed ratings and memory consumption is linear in the number of users and the number of items.
- Finally, we conduct a series of detailed experiments on real-world datasets to demonstrate the merits and properties of the proposed online learning algorithms.

The remainder of this paper is organized as follows. In Section II, we briefly review some related work. Section III discusses in more detail regarding two models for which we develop online algorithms, namely probabilistic matrix factorization and top-one probability based ranking matrix factorization. We present our online algorithms in Section IV. Experimental results and analysis are shown in Section V and

978-1-4673-1490-9/12/$31.00 ©2012 IEEE

conclusions are drawn in Section VI.

## II. Related Work

In this section, we review related work on collaborative filtering and online learning.

### A. Collaborative Filtering

Collaborative filtering techniques for recommender systems are generally categorized into two types: memory-based methods and model-based methods. With different optimization objectives, collaborative filtering can be classified into rating-oriented methods, which try to minimize error between prediction and true rating, and ranking-oriented methods, which try to rank the items in a correct order.

Memory-based methods manipulate the ratings assigned by users directly in making predictions. For rating-oriented approaches, user-based methods [6] and item-based methods [7], [8], [9] are mostly studied. Besides rating-oriented methods, Liu et al. proposed [10] a pairwise ranking-oriented method called EigenRank, and reported it attains more credible ranking scores.

Memory-based methods are easy to implement and understand, they are used in a lot of real-world systems [8]. However, they impose several limitations. First, they are more susceptible to the data sparsity problem because in order to measure the similarities, two users need to rate at least some items in common. Furthermore, as they manipulate the ratings directly, the time complexity and memory consumption can be potentially very expensive.

Model-based approaches, on the other hand, provide a systematic way to train a predefined compact model in the training phase that explains observed ratings, which is then used to make predictions. Usually, model-based CF methods can achieve better performance [11]. Various approaches, including rating-oriented methods [11], [12], [13], [14], [15] and ranking-oriented methods [16], [17], have been proposed.

Model-based methods hold several appealing properties. First, they often have a clear interpretation. Secondly, once trained, they can produce predictions much more efficiently compared with memory-based methods. Finally, we are able to gain specific insights concerning the community structure in several latent feature-based methods. Hence, we focus on model-based approaches, specifically PMF and RMF, in this paper.

### B. Online Learning

Online learning algorithms have been extensively studied in content-based filtering [18], [19], [20], which is another major approach to recommender system. However, in collaborative filtering, there are only limited investigations. In [4], an online algorithm is developed for memory-based collaborative filtering. In [2], [21], online algorithms for Non-negative Matrix Factorization (NMF) are considered. In [3], an online algorithm is conducted on a mixture of memory-based and model-based algorithms, where the data are dynamically clustered. The involved models, however, are different from what we consider as the matrix factorization models. Another work related to our work is [5], which applies online algorithms on sparse models in computer vision area. In [1], a gradient descent method on matrix factorization with (or without) features by directly minimizing the square loss is proposed to convert a batch-trained algorithm into an online version. It ignores regularization effects and may be suboptimal under certain conditions.

Although adapting the model-based collaborative filtering methods by online algorithms, e.g., stochastic gradient descent method [22], [23], can involve substantial implementation efforts, the real properties such as efficiency, convergence etc. of the various algorithms are still not well-investigated. This unexplored territory motivates us to study the online learning algorithms for PMF and RMF thoroughly. To be more specific, we investigate both stochastic gradient descent and dual averaging methods on both rating-oriented and ranking-oriented MF methods.

## III. Model-based Matrix Factorization

In this section we present PMF, RMF and their batch-trained algorithms. Suppose that we are given a set of $N$ users $\mathcal{U} = \{u_1, u_2, \cdots, u_N\}$ and a set of $M$ items $\mathcal{I} = \{i_1, i_2, \cdots, i_M\}$. Users' rating on the items forms an $N \times M$ matrix $R$, where the element $r_{ui}$ denotes user $u$'s rating on item $i$. Alternatively, we denote all observed ratings in a set of triplets as $(u, i, r) \in \mathcal{Q}$, where $u$ is the user id, $i$ is the item id, and $r$ is the rating given by $u$ to $i$. Usually, the rating $r$ is a value in the range $[R_{\min}, R_{\max}]$. Often, we map it to $[0, 1]$ by $(r - R_{\min})/(R_{\max} - R_{\min})$. In the following, we assume that $r$ have all been mapped to $[0, 1]$. To avoid clutter notations, we use $g_{ij}$ to denote $g(U_i^T V_j)$ and $g'_{ij}$ to denote the derivative of the logistic function $g'(U_i^T V_j)$, where $g(x)$ is the logistic function to be defined in Eq. (2).

The problem of matrix factorization collaborative filtering is to learn two low-rank feature matrices, $U$ and $V$, where $U^T V$ fits $R$ based on the given $M, N, R$ or $M, N, Q$. The user feature matrix $U$ is a $K \times N$ matrix where the column $U_u$ is user $u$'s feature vector. $V$ is a $K \times M$ matrix where the column $V_i$ is item $i$'s feature vector. Generally, the latent feature size $K$ is much smaller than $N$ and $M$.

For rating-oriented methods, the objective is to find $U$ and $V$ to best fit $R$ so as to correctly predict $\hat{r}_{ui}$, the rating user $u$ would assign to item $i$. Whereas in ranking-oriented methods, the target is to correctly output a ranking $\pi$ of the items in decreasing order of preference for an active user.

### A. Probabilistic Matrix Factorization

Probabilistic Matrix Factorization (PMF) adopts a probabilistic linear model with Gaussian observation noise [11]. Maximizing the posterior probability of $p(U, V | R, \sigma^2, \sigma_U^2, \sigma_V^2)$ is equivalent to minimizing a squared loss with regularization defined as:

$$\mathcal{L} = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{M} I_{ij} (r_{ij} - g_{ij})^2 + \frac{\lambda_U}{2} \|U\|_F^2 + \frac{\lambda_V}{2} \|V\|_F^2, \quad (1)$$

where $g$ is the logistic function used to map the value into the range of $[0, 1]$ as is in [11]:

$$g(x) = 1/(1 + \exp(-x)). \tag{2}$$

$\lambda_U$ and $\lambda_V$ are $L_2$-regularization strength parameter to avoid over-fitting and $I_{ij}$ is an indicator function which equals 1 if user $i$ have rated item $j$ and 0 otherwise.

Gradient descent algorithm can be adopted to reach a local minimum of the objective given in Eq. (1). Thus, the feature matrices on users and items can be updated iteratively by

$$U_i \leftarrow U_i - \eta \frac{\partial \mathcal{L}}{\partial U_i}, \qquad V_j \leftarrow V_j - \eta \frac{\partial \mathcal{L}}{\partial V_j}, \tag{3}$$

where $\eta$ is the learning rate. In practice, it takes dozens of iterations for PMF to converge. Once trained, the predicted rating that user $u$ would assign to item $i$ can be computed as the expected value of the Gaussian distribution $\hat{r}_{ui} = g_{ui}$. Note that this value may need to be converted back to the original rating range.

### B. Ranking Matrix Factorization

Top-one probability based ranking matrix factorization (RMF) [17] also factorizes the user-item matrix $R$ into $U$ and $V$. Different from PMF, it minimizes the cross entropy of two top-one probability distributions defined on actual rating $r_{ij}$ and predicted rating $g_{ij}$. Top-one probability is the probability of an item being ranked top in an active user's ranking list. Using actual ratings $R$, the top-one probability associated with an item $i$ in a ranking $\pi$ for user $u$ is defined as:

$$p_R(r_{ui}) = \frac{\exp(r_{ui})}{\sum_{k=1}^{M} I_{uk} \exp(r_{uk})}. \tag{4}$$

Using predicted rating, the top-one probability of the learned model is defined as:

$$p_{UV}(g_{ui}) = \frac{\exp(g_{ui})}{\sum_{k=1}^{M} I_{uk} \exp(g_{uk})}. \tag{5}$$

RMF minimizes the cross entropy between $p_R$ and $p_{UV}$ [17]. Cross entropy of two distributions $p$ and $q$ is defined as:

$$H(p, q) = E_p[-\log\ q] = -\sum_x p(x) \log q(x). \tag{6}$$

This quantity measures the divergence between two distributions and is minimized when $p = q$. Hence, to find the optimal $U$ and $V$, RMF is to minimize the following objective function:

$$\mathcal{L} = \sum_{i=1}^{N} \left\{ -\sum_{j=1}^{M} I_{ij} \frac{\exp(r_{ij})}{\sum_{k=1}^{M} I_{ik} \exp(r_{ik})} \log \left\{ \frac{\exp(g_{ij})}{\sum_{k=1}^{M} I_{ik} \exp(g_{ik})} \right\} \right\} + \frac{\lambda_U}{2} \|U\|_F^2 + \frac{\lambda_V}{2} \|V\|_F^2. \tag{7}$$

Similarly, RMF is not a convex optimization problem and the local minima can be sought using the gradient descent method. The gradients of $\mathcal{L}$ with respect to $U$ and $V$ can be calculated in closed form. We can adopt Eq. (3) to update $U$ and $V$ until they converge. Once trained, the model recommends the items in decreasing order of their top-one probability to an active user.

## IV. Online Matrix Factorization

Both batch-trained PMF and RMF assume that all the ratings are available before the training, which make them unsuitable for many practical application scenarios. To capture the information embedded in a newly received rating, the model has to be retrained using *all* available data. To adapt PMF or RMF to such scenarios, it is better to train the model in an *online* manner, which incrementally adapts the model to the newly observed ratings. In the following, we present our online algorithms for both PMF and RMF to demonstrate such merits.

### A. Online PMF

We present two algorithms of online PMF, the stochastic gradient descent PMF (SGD-PMF) and dual averaging PMF (DA-PMF).

*1) Stochastic Gradient Descent for PMF:* Please note that in Eq. (3), at each iteration, the low-rank matrices move toward the *average* gradient descent, $\partial\mathcal{L}/\partial U_u$ and $\partial\mathcal{L}/\partial V_i$, by a small step controlled by $\eta$. If the collected ratings are coming sequentially, we could adjust the model stochastically by taking into account that rating only. This corresponds to the scheme of stochastic gradient descent.

Suppose the new coming rating is $(u, i, r) \in \mathcal{Q}$, in Eq. (1), the terms related to this perticular rating are:

$$\mathcal{L}_{(u,i,r)} = (r_{ui} - g_{ui})^2 + \frac{\lambda_U}{2} \|U_u\|_2^2 + \frac{\lambda_V}{2} \|V_i\|_2^2. \tag{8}$$

The first term in Eq. (8) is the squared error between the observation and predicted value, and the following two terms are the corresponding regularizations. Notice that here the trade-off constants, $\lambda_U$ and $\lambda_V$, are on different scale from those in Eq. (1).

Similarly, by adopting the gradient descent method, we obtain the following update equations:

$$U_u \leftarrow U_u - \eta((g_{ui} - r) g'_{ui} V_i + \lambda_U U_u), \tag{9}$$

$$V_i \leftarrow V_i - \eta((g_{ui} - r) g'_{ui} U_u + \lambda_V V_i), \tag{10}$$

where $\eta$ is the step size controlling how much change to make at each step. This naturally gives an online algorithm, where at each iteration, we make a small change for user $u$'s feature vector $U_u$ and item $i$'s feature vector $V_i$ when a rating $(u, i, r)$ is revealed. We call this method stochastic gradient descent PMF (SGD-PMF).

SGD-PMF is *stochastic* in the sense that every time we adjust the parameter, we accommodate it to that particular rating seen at that instance. We do not provide the convergence of stochastic gradient descent for PMF here, as its detailed proof can be referred to [24].

*2) Dual-Averaging Method for PMF:* Recently, dual-averaging method [25] ignited the development of online optimization algorithms. By imposing different regularizations, variants of online learning algorithms have been developed and they achieve good result in the corresponding applications [26], [19]. Due to the success and efficiency of dual-averaging method, we decide to adopt it to solve the online PMF problem.

Dual-average method for PMF absorbs previous rating information in an approximate average gradient of the loss. Then it updates the parameters by solving an analytically tractable optimization problem. Hence, in DA-PMF, we keep track of the average gradient, $Y$, of the square loss with respect to $U$ and $V$. Given a newly observed triplet $(u, i, r)$, we can update the average gradient with respect to $U_u$ by the following rule:

$$Y_{U_u} \leftarrow \frac{t_u - 1}{t_u} Y_{U_u} + \frac{1}{t_u}(g_{ui} - r) g'_{ui} V_i, \tag{11}$$

where $t_u$ denotes the number of items $u$ has rated. Note that $Y_{U_u}$ in Eq. (11) is an approximation of $\{\sum_{i \in \mathcal{I}_u}(g_{ui} - r) g'_{ui} V_i\}/t_u$ which is the average gradient of the squared loss with respect to $U_u$. $\mathcal{I}_u$ denotes all the items that $u$ has rated. Similarly, we can obtain $Y_{V_i}$, the average gradient of $\mathcal{L}$ with respect to $V_i$, as follows:

$$Y_{V_i} \leftarrow \frac{t_v - 1}{t_v} Y_{V_i} + \frac{1}{t_v}(g_{ui} - r) g'_{ui} U_u, \tag{12}$$

where $t_v$ denotes the number of users who have rated item $v$.

Once we get the average gradient, we update $U_u$ and $V_i$ by solving the following minimization problems:

$$U_u = \arg\min_w \{Y_{U_u}^T w + \lambda_U \|w\|_2^2\}, \tag{13}$$

$$V_i = \arg\min_w \{Y_{V_i}^T w + \lambda_V \|w\|_2^2\}. \tag{14}$$

To get an intuition of why choosing such optimization objectives defined in Eq. (13) and Eq. (14), we should note that $Y_{U_u}^T w$, the dot product of $Y_{U_u}$ and $w$, is minimized when $w$ is on the opposite direction as $Y_{U_u}$. We have shown that $Y_{U_u}$ is the average gradient, whose direction will lead to a larger $\mathcal{L}$. By taking an opposite direction of $Y_{U_u}$, we expect $\mathcal{L}$ to be decreased. Regularization term $\lambda_V \|w\|_2^2$ is used to limit the value that $w$ can assume. Thus using such an optimization objective, we can get $U_u$ and $V_i$ that lead to a decreased $\mathcal{L}$. Convergence of such formulation is referred to [26]. The solution to Eq. (13) and Eq. (14) can be found analytically by taking the derivative to 0, which is summarized in Algorithm 1.

*3) Time Complexity and Memory Cost:* Both SGD-PMF and DA-PMF are efficient in terms of time complexity and memory cost. For SGD-PMF, only user feature matrix and item feature matrix have to be stored in memory, so the memory cost is $O((N+M)K)$. $K$ is generally on the scale of tens even for a very big dataset. For each observation $(u, i, r) \in \mathcal{Q}$, only $O(K)$ steps are needed to update the model. Since $K$ is quite small, it can be taken as constant time. So SGD-PMF scales linearly with the number of observed ratings. For DA-PMF, beside user feature matrix and item feature matrix, average gradient matrix $Y_U, Y_V$ and index vector $T_U, T_V$ are also stored. Total memory cost is still $O((N + M)K)$. Similar to SGD-PMF, DA-PMF needs $O(K)$ steps for each observation and thus it scales linearly with the number of observed ratings. So both SGD-PMF and DA-PMF can be effective to large-scale datasets.

**Algorithm 1** Dual-Averaging Method for PMF (DA-PMF)

Parameters: $N, M, K, \lambda_U, \lambda_V$
Input: Observation triplet $(u, i, r) \in \mathcal{Q}$
Initialize $U \in \mathbb{R}^{K \times N}$ and $V \in \mathbb{R}^{K \times M}$ randomly
Initialize average gradient matrix to $\mathbf{0}$

$$Y_U \in \mathbb{R}^{K \times N} \leftarrow \mathbf{0}; \quad Y_V \in \mathbb{R}^{K \times M} \leftarrow \mathbf{0}$$

Initialize index vector $T_U \in \mathbb{Z}^N \leftarrow \mathbf{0}$ and $T_V \in \mathbb{Z}^M \leftarrow \mathbf{0}$
**for all** $(u, i, r) \in \mathcal{Q}$ **do**
  Increase index $T_{U_u} \leftarrow T_{U_u} + 1$ and $T_{V_i} \leftarrow T_{V_i} + 1$
  $t_u \leftarrow T_{U_u}, \quad t_v \leftarrow T_{V_i}$
  Update average gradient $Y_U$ and $Y_V$

$$Y_{U_u} \leftarrow \frac{t_u - 1}{t_u} Y_{U_u} + \frac{1}{t_u}(g_{ui} - r) g'_{ui} V_i$$

$$Y_{V_i} \leftarrow \frac{t_v - 1}{t_v} Y_{V_i} + \frac{1}{t_v}(g_{ui} - r) g'_{ui} U_u$$

  Update latent user and item feature $U_u$ and $V_i$

$$U_u \leftarrow -\frac{1}{2\lambda_U} Y_{U_u}, \quad V_i \leftarrow -\frac{1}{2\lambda_U} Y_{V_i} \tag{15}$$

**end for**

### B. Online RMF

We consider both the stochastic gradient descent method (SGD-RMF) and the dual-averaging method (DA-RMF) for RMF in this section.

*1) Stochastic Gradient Descent for RMF:* The loss $\mathcal{L}$ of RMF is defined in Eq. (7). Let $\mathcal{H} = \mathcal{L} - \frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_V}{2}\|V\|_F^2$ be the loss without the regularization, i.e., it only captures the cross entropy between $p_R$ and $p_{U,V}$.

Unlike the squared loss used in PMF, which can be easily dissected when a new rating is observed, cross entropy is adopted as the loss in RMF. It measures the divergence between the top-one probability distribution defined using actual rating and that defined using predicted rating. The top-one probability indicates the probability of an item being ranked in the top position. It is essentially a categorical distribution and normalization is engaged to ensure it is a proper probability measure. When a newly observed rating $(u, i, r)$ is revealed, this probability mass function will include one more term indicating the probability of the new item being ranked in the top position. Due to normalization, the top-one probability corresponds to other items are further decayed. $\mathcal{H}$, the cross entropy of these two distributions, also changes accordingly. Note that these changes are coupled together and are thus very difficult to dissect.

We resort to algorithms that approximate the gradient descent of $\mathcal{H}$ with respect to $U_u$ and $V_i$ given in Eq. (16) and

Eq. (17):

$$\frac{\partial \mathcal{H}}{\partial U_i} = \sum_{j=1}^{M} I_{ij} \left\{ \frac{\exp(g_{ij})}{\sum_{k=1}^{M} I_{ik} \exp(g_{ik})} - \frac{\exp(r_{ij})}{\sum_{k=1}^{M} I_{ik} \exp(r_{ik})} \right\} g'_{ij} V_j, \tag{16}$$

$$\frac{\partial \mathcal{H}}{\partial V_j} = \sum_{i=1}^{N} I_{ij} \left\{ \frac{\exp(g_{ij})}{\sum_{k=1}^{M} I_{ik} \exp(g_{ik})} - \frac{\exp(r_{ij})}{\sum_{k=1}^{M} I_{ik} \exp(r_{ik})} \right\} g'_{ij} U_i. \tag{17}$$

Now, we consider the update of the gradients as the observed rating appears one by one. Denote $Y_{U_u}^{t_u}$ as the gradient of $\mathcal{H}$ with respect to $U_u$ when the $t_u$-th rating is assigned by user $u$ to item $i$. Denote $Y_{V_i}^{t_v}$ as the gradient of $\mathcal{H}$ with respect to $V_i$ when item $i$ receives its $t_v$-th rating. To simplify the expression, we let $\mathcal{I}_u^{t_u}$ denote the set of items rated by user $u$ when the $t_u$-th rating is observed, i.e., $|\mathcal{I}_u^{t_u}| = t_u$. We update the corresponding gradients by the following rules:

$$Y_{U_u}^{t_u+1} \leftarrow \frac{\sum_{k \in \mathcal{I}_u^{t_u}} \exp(r_{uk})}{\sum_{k \in \mathcal{I}_u^{t_u+1}} \exp(r_{uk})} Y_{U_u}^{t_u} + \left\{ \frac{\exp(g_{ui})}{\sum_{k \in \mathcal{I}_u^{t_u+1}} \exp(g_{uk})} - \frac{\exp(r_{ui})}{\sum_{k \in \mathcal{I}_u^{t_u+1}} \exp(r_{uk})} \right\} g'_{ui} V_i, \tag{18}$$

$$Y_{V_i}^{t_v+1} \leftarrow (1 - \alpha^{c \times t_v}) Y_{V_i}^{t_v} + \left\{ \frac{\exp(g_{ui})}{\sum_{k \in \mathcal{I}_u^{t_u+1}} \exp(g_{uk})} - \frac{\exp(r_{ui})}{\sum_{k \in \mathcal{I}_u^{t_u+1}} \exp(r_{uk})} \right\} g'_{ui} U_u. \tag{19}$$

It should be noted that we initialize $Y_U^0 = \mathbf{0}, Y_V^0 = \mathbf{0}$ for all users and items. Index vector $T_U$, whose $u$-th entry is $t_u$, and $T_V$, whose $i$-th entry is $t_v$, are on a per-user and per-item basis respectively.

Here, we argue that Eq. (18) and Eq. (19) approximate the ideal gradients in Eq. (16) and Eq. (17), respectively. It is obvious that Eq. (18) recovers Eq. (16) provided that $g_{ui}$ approximates $r_{ui}$ well at each iteration. To see that Eq. (19) indeed approximates Eq. (17), consider what might happen between two ratings item $i$ receives. Note that the summation in Eq. (17) is over all the users who have rated item $i$. Since top-one probability is on a per-user basis, we cannot find a single value that properly describes the decay of previously rated item as is the case in Eq. (18). Consider the event that might happen between the observations of $(u_1, i, r_1)$ and $(u_2, i, r_2)$. Let the set of users who have rated $i$ be $\mathcal{U}_i$. The change of the top-one probability happens when a $u \in \mathcal{U}_i$ rate another movie $i'$ and thus causing the top-one probability with respect to $i$ to decay due to the normalization. However, as more and more ratings are revealed, the change will be smaller and smaller. So we need to decay previous gradient to reflect this change. Therefore, $\alpha \in (0, 1)$ in Eq. (19) controls the initial decay rate and $c$ controls how this rate drops.

The update rule for SGDPMF is

$$U_u \leftarrow U_u - \eta(Y_{U_u} + \lambda_U U_u); \quad V_i \leftarrow V_i - \eta(Y_{V_i} + \lambda_V V_i). \tag{20}$$

**Algorithm 2** SGD-RMF/DA-RMF

Parameter: $N$, $M$, $K$, $\eta$, $\lambda_U$, $\lambda_V$, $\alpha$, $c$
Input: Observation triplet $(u, i, r) \in \mathcal{Q}$
Initialize $U \in \mathbb{R}^{K \times N}$ and $V \in \mathbb{R}^{K \times M}$ randomly
Initialize average gradient matrix to $\mathbf{0}$

$$Y_U \in \mathbb{R}^{K \times N} \leftarrow \mathbf{0}; \; Y_V \in \mathbb{R}^{K \times M} \leftarrow \mathbf{0}$$

Initialize index vector $T_V \in \mathbb{Z}^M \leftarrow \mathbf{0}$
Initialize sum vector $S_R \in \mathbb{R}^N \leftarrow \mathbf{0}$ and $S_{UV} \in \mathbb{R}^N \leftarrow \mathbf{0}$
**for all** $(u, i, r) \in \mathcal{Q}$ **do**
  $t_v \leftarrow T_V(v)$
  $s_r \leftarrow S_R(u)$
  $s_{uv} \leftarrow S_{UV}(u)$
  $s'_r \leftarrow s_r + \exp(r)$
  $s'_{uv} \leftarrow s_{uv} + \exp(g_{ui})$
  $Y_{U_u} \leftarrow \frac{s_r}{s'_r} Y_{U_u} + \{\frac{\exp(g_{ui})}{s'_{uv}} - \frac{\exp(r)}{s'_r}\} g'_{ui} V_i$
  $Y_{V_i} \leftarrow (1 - \alpha^{ct_v}) Y_{V_i} + \{\frac{\exp(g_{ui})}{s'_{uv}} - \frac{\exp(r)}{s'_r}\} g'_{ui} U_u$
  For SGD-RMF, update using Eq. (20)
  For DA-RMF, update using Eq. (21)
  $S_R(u) \leftarrow s'_r$
  $S_{UV}(u) \leftarrow s'_{uv}$
  $T_V(v) \leftarrow t_v + 1$
**end for**

*2) Dual-Averaging Method for RMF:* The gradient we calculated as in Eq. (18) and Eq. (19) is average gradient. By solving the same optimization objective as defined in Eq. (13) and Eq. (14) at each iteration, we obtain the Dual-Averaging RMF (DA-RMF) algorithm. The update rule for DAPMF is

$$U_u \leftarrow -\frac{1}{2\lambda_U} Y_{U_u}; \quad V_i \leftarrow -\frac{1}{2\lambda_V} Y_{V_i} \tag{21}$$

Both SGDPMF and DAPMF are summarized in Algorithm 2.

*3) Time Complexity and Memory Cost:* The memory cost for both SGD-RMF and DA-RMF includes the user and item feature matrix $U, V$, approximate average gradient $Y_U, Y_V$, index vector $T_V$, and sum vector $S_R, S_{UV}$. The total memory cost is still $O((N + M)K)$, which is independent of the number of observed ratings. In terms of time complexity, the steps needed for each observed triplet $(u, i, r)$ is still $O(K)$. Namely, both SGD-RMF and DA-RMF scale linearly with the number of observed ratings.

## V. Experiments

In this section, we conduct experiments to compare the performance of our online algorithms with batch-trained algorithms. The questions we want to address include:

1) How is the stochastic gradient descend and dual-averaging algorithms compared with the batch mode algorithms?
2) How do the online algorithms perform under different settings?
3) How well do stochastic gradient descend and dual-averaging methods scale to large datasets?
4) In which way do model parameters, i.e., $\lambda, \eta$, affect the algorithms' performance?

TABLE I
STATISTICS OF DATASETS

| | MovieLens | Yahoo!Music |
|---|---|---|
| **No. ratings** | 1,000,209 | 252,800,275 |
| **No. users** | 6,040 | 1,000,990 |
| **No. items** | 3,952 | 624,961 |
| **Rating range** | $[1, 5]$ | $[0, 100]$ |

### A. Data Sets

We choose MovieLens[1] and Yahoo!Music[2] to study empirical performance of our algorithms. Table I shows the basic statistics of each dataset. Experiments studying the effect of parameters are performed on MovieLens. Comparisons of online algorithms versus their batch-trained algorithms are also conducted on MovieLens. We select Yahoo!Music to evaluate how the online algorithms scale to large datasets. Yahoo!Music is a very large dataset containing more than 250 million ratings. Yet this data set is extremely sparse, where only 0.4% entries are known. However, RMF experiments do not utilize Yahoo!Music because we found most users apply only a few values to indicate their fondness. The problem with many items sharing the same rating value is that the user's actual preferences over these items are implicit and thus rendering the NDCG metric insensitive to different ranking.

### B. Evaluation Metrics

We adopt Root Mean Square Error(RMSE)[3] to evaluate rating-oriented algorithms. RMSE evaluates the root of average square error between true rating and predicted rating. Denote the test set by $\mathcal{T}$, the definition of RMSE is given by $\text{RMSE} = \sqrt{\sum_{(u,i,r)\in\mathcal{T}} (\hat{r}_{u,i} - r)^2 / |\mathcal{T}|}$.

To evaluate the ranking accuracy, we adopt Normalized Discounted Cumulative Gain (NDCG)[4] [27] as the metric. Let $\pi(i)$ be the item ranked on the $i$th position in ranking $\pi$ and the actual rating assigned to $i$ is $r_{\pi(i)}$. Then NDCG at $n$ is defined as:

$$\text{NDCG@n} = \sum_{i=1}^{n} \frac{2^{r_{\hat{\pi}(i)}} - 1}{\log(1+i)} \Big/ \sum_{i=1}^{n} \frac{2^{r_{\pi^*(i)}} - 1}{\log(1+i)}, \qquad (22)$$

where $\pi^*$ is the optimal ranking. An appealing property of NDCG is that it gives more weight on the items ranked higher than the item ranked lower. This is consistent with our experience that user seldom looks past the first few recommended items.

### C. Evaluation Protocol

To better understand the behavior of our online algorithms under different settings, we conduct experiments with the following three settings:

1) T1: Randomly choose 10% of all $(u, i, r)$ triplets for training, and use remaining 90% for evaluation.
2) T5: Randomly choose 50% of all $(u, i, r)$ triplets for training, and use remaining 50% for evaluation.
3) T9: Randomly choose 90% of all $(u, i, r)$ triplets for training, and use remaining 10% for evaluation.

[1]http://www.cs.umn.edu/Research/GroupLens
[2]http://kddcup.yahoo.com
[3]The lower the RMSE, the better the performance.
[4]The higher the NDCG, the better the performance.

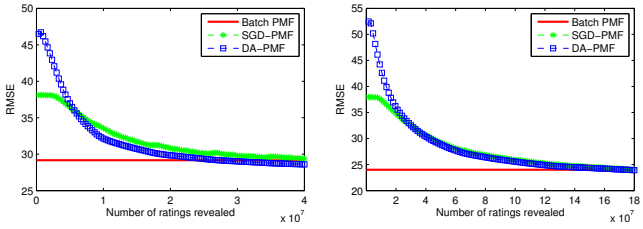
### D. Comparisons

In this section, we compare our online algorithms with full-fledged batch-trained PMF and RMF and investigate how they scale to large datasets. The batch-trained algorithms have been tuned to achieve the best performance on the test set, which is consistent with the results in [1], [17]. The online algorithms are also tuned on the test set, which can be referred to Sec. V-E.

*1) Online versus Batch:* The top row of Figure 1 shows that DA-PMF and SGD-PMF perform comparable as batch-trained PMF in MovieLens. Under T1, online algorithms even outperforms batched-trained algorithms a little bit. This may be due to the fact that under the scenario of few training samples, online learning algorithms are less likely to be trapped in a local optima. Overall, our online PMF algorithms perform as well as batch-trained algorithm.

The second row of Figure 1 shows the comparison of various RMF algorithms under different settings. Compared to batch-trained RMF, our online algorithms' performance is off by about 1% in all settings evaluated by NDCG@5. The performance gap is probably due to the approximation when computing the gradient with respect to $V$.

*2) Scaling to Large Dataset:* Figure 2 shows the comparison of online and batch PMF in Yahoo!Music. Note that we evaluate the performance on the evaluation set which contains 4 million ratings. Due to the huge size of Yahoo!Music dataset, we were unable to perform batch-trained PMF using T9 setting. Under T5, batch-trained PMF takes more than 8 hours to finish 120 iterations (to converge) using a C++ implementation in a Linux workstation with Xeon Dual Core 2.4GHz processor and 32GB memory. The online algorithms take only about 10 minutes to finish processing all 180 million ratings to reach a similar performance. The time saving is phenomenal.



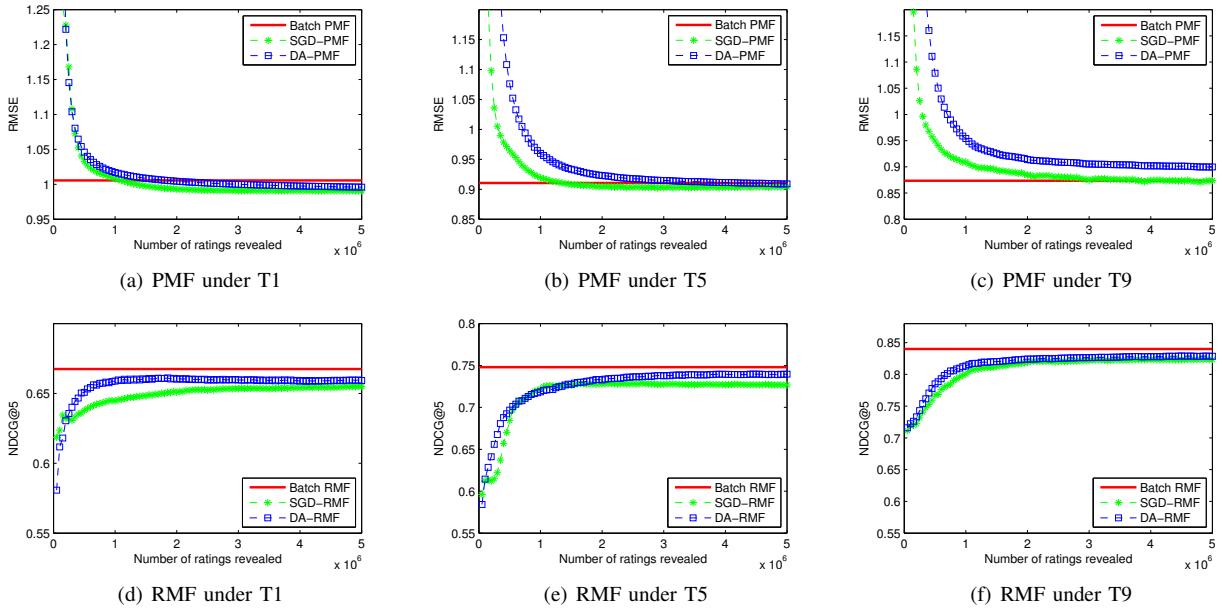

(a) PMF under T1 (b) PMF under T5

Fig. 2. Comparison of PMF in Yahoo!Music

Please note that, in the experiment, we have cycled through the ratings and fed them randomly into the algorithms. The results are reported in Table II and III.

### E. Impact of Parameters

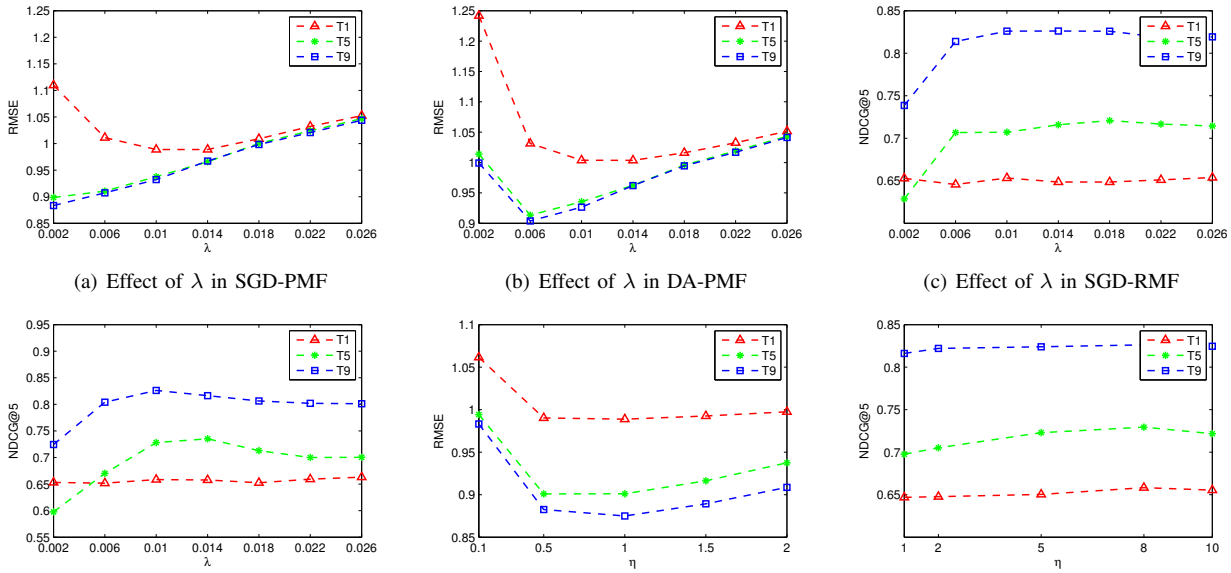
We analyze the impact of parameters in this section. We employ latent feature dimension $K = 10$ consistently for all algorithms, which is a suitable value according to our




(a) PMF under T1 (b) PMF under T5 (c) PMF under T9

(d) RMF under T1 (e) RMF under T5 (f) RMF under T9

Fig. 1. Comparison of various PMF and RMF algorithms in MovieLens




(a) Effect of $\lambda$ in SGD-PMF (b) Effect of $\lambda$ in DA-PMF (c) Effect of $\lambda$ in SGD-RMF

(d) Effect of $\lambda$ in DA-RMF (e) Effect of $\eta$ in SGD-PMF (f) Effect of $\eta$ in SGD-RMF

Fig. 3. Effect of $\lambda$ and $\eta$ in various online algorithm

TABLE II
ONLINE AND BATCH PMF RESULTS(RMSE)

| | MovieLens | | | Yahoo!Music | |
|---|---|---|---|---|---|
| | **T1** | **T5** | **T9** | **T1** | **T5** |
| **PMF** | 1.005 | 0.910 | 0.873 | 29.16 | 24.00 |
| **DA-PMF** | 0.996 | 0.909 | 0.900 | 28.59 | 23.92 |
| **SGD-PMF** | 0.991 | 0.904 | 0.874 | 29.38 | 24.02 |

TABLE III
ONLINE AND BATCH RMF RESULTS(NDCG@5)

| | MovieLens | | |
|---|---|---|---|
| | **T1** | **T5** | **T9** |
| **RMF** | 0.667 | 0.748 | 0.840 |
| **DA-RMF** | 0.659 | 0.740 | 0.827 |
| **SGD-RMF** | 0.655 | 0.727 | 0.824 |

empirical results. Since all algorithms we consider are matrix factorization based, employing the same latent feature size is fair for comparison. The results reported in Fig. 3 are obtained on MovieLens.

*1) Impact of $\lambda$:* The parameter $\lambda$ controls the trade-off between the regularization and the model loss. We set $\lambda_U = \lambda_V = \lambda$ for simplicity [11], [17]. Figure 3 shows the impact of $\lambda$ in four online algorithms under different settings. The range of $\lambda$ is selected by trail-and-error as shown in Figure 3, where the range of $\lambda$ gives reasonable performance. As we can see, there is a clear trend that the more data we have, the smaller $\lambda$ we need. Since the model complexity is fixed

(i.e., we choose dimension = 10 for all experiments), we need a larger $\lambda$ to avoid over-fitting when there is only limited data. In [1], the author reported that stochastic gradient descent without regularization performs quite well compared to batch-trained algorithm. This is true only when we have access to abundant data. A proper regularization is vital for a model with limited data.

Figure 3(c) and 3(d) show that both ranking-oriented online algorithms are quite stale in the setting T1. This is due to the insufficiency of training data in T1. Only 10% of training data to train the model would not make it well-fitted. This causes the insensitivity of the effect of $\lambda$.

*2) Impact of $\eta$:* The parameter $\eta$ denotes the learning rate, which is only applied in SGD-PMF and SGD-RMF. It will impact the convergence rate and the performance of the algorithms. Figure 3 shows the performance of SGD-PMF and SGD-RMF under T1, T5 and T9 with respect to various $\eta$ values. When performing the experiments, we employ the best $\lambda$ we learnt in previous sections. We see that $\eta = 1.0$ is optimal for SGD-PMF and $\eta = 8.0$ is optimal for SGD-RMF.

There is a subtle difference between online algorithms and batch-trained algorithms. The optimal learning rate $\eta$ depends on the size of the training set in batch-trained algorithms. On the contrary, it is independent for online algorithms. The reason is that we update $U$ and $V$ with respect to only *one* rating in online algorithms each time.

*3) Impact of other parameters:* In the two online RMF algorithms, there are two extra parameters $\alpha$ and $c$ which control the decay and drop-rate of decay respectively. In practice, they do not affect the model performance as significant as $\lambda$. Hence, we do not present their sensitivity analysis here. In the experiment, we set $\alpha = 0.8, c = 0.2$ since they deliver good performance empirically.

## VI. Conclusion and Future Work

In this paper, we have thoroughly investigated the online learning algorithms for rating-oriented CF model, PMF, and ranking-oriented CF model, RMF. More specifically, we developed Stochastic Gradient Descent and Dual-Averaging methods for both models. Our proposed algorithms scale linearly with the number of observed ratings. Furthermore, they obviate the need to hold all data in memory and thus can be applied to large-scale applications. Experimental results show that our online algorithms achieve comparable performance as their batch-trained algorithms while dramatically boosting efficiency.

There are several directions worthy of considering for future study: 1) to well study the convergence of the online learning algorithms from theoretical perspective; 2) to explore the online optimization framework with different types of regularizations to achieve solutions with different properties; and 3) to propose a systematical way to tune the algorithms' parameters;

## VII. Acknowledgments

The work described in this paper was fully supported by two grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK 413210 and CUHK 415311) and two grants from Google Inc. (one for Focused Grant Project "Mobile 2014" and one for Google Research Awards).

## References

[1] J. Abernethy, K. Canini, J. Langford, and A. Simma, "Online collaborative filtering," University of California at Berkeley, Tech. Rep., 2007.
[2] B. Cao, D. Shen, J.-T. Sun, X. Wang, Q. Yang, and Z. Chen, "Detect and track latent factors with online nonnegative matrix factorization," in *IJCAI*, 2007, pp. 2689–2694.
[3] A. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization: scalable online collaborative filtering," in *WWW*, 2007, pp. 271–280.
[4] N. N. Liu, M. Zhao, E. W. Xiang, and Q. Yang, "Online evolutionary collaborative filtering," in *RecSys*, 2010, pp. 95–102.
[5] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *Journal of Machine Learning Research*, vol. 11, pp. 19–60, 2010.
[6] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," in *SIGIR*, 1999, pp. 230–237.
[7] M. Deshpande and G. Karypis, "Item-based top-n recommendation algorithms," *ACM Trans. Inf. Syst.*, vol. 22, pp. 143–177, January 2004.
[8] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, pp. 76–80, January 2003.
[9] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *WWW*, 2001, pp. 285–295.
[10] N. N. Liu and Q. Yang, "Eigenrank: a ranking-oriented approach to collaborative filtering," in *SIGIR*, 2008, pp. 83–90.
[11] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *NIPS*, 2007.
[12] T. Hofmann, "Latent semantic models for collaborative filtering," *ACM Trans. Inf. Syst.*, vol. 22, pp. 89–115, January 2004. [Online]. Available: http://doi.acm.org/10.1145/963770.963774
[13] H. Ma, I. King, and M. R. Lyu, "Learning to recommend with social trust ensemble," in *SIGIR*, 2009, pp. 203–210.
[14] D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles, "Collaborative filtering by personality diagnosis: A hybrid memory and model-based approach," in *UAI*, 2000, pp. 473–480.
[15] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using markov chain monte carlo," in *ICML*, 2008, pp. 880–887.
[16] N. N. Liu, M. Zhao, and Q. Yang, "Probabilistic latent preference analysis for collaborative filtering," in *CIKM*, 2009, pp. 759–766.
[17] Y. Shi, M. Larson, and A. Hanjalic, "List-wise learning to rank with matrix factorization for collaborative filtering," in *RecSys*, 2010, pp. 269–272.
[18] L. Bottou and Y. LeCun, "Large scale online learning," in *NIPS*, 2003.
[19] H. Yang, I. King, and M. R. Lyu, "Online learning for multi-task feature selection," in *CIKM*, 2010, pp. 1693–1696.
[20] H. Yang, Z. Xu, I. King, and M. R. Lyu, "Online learning for group lasso," in *ICML*, 2010, pp. 1191–1198.
[21] A. Lefevre, F. Bach, and C. Févotte, "Online algorithms for nonnegative matrix factorization with the itakura-saito divergence," in *WASPAA*, 2011, pp. 313–316.
[22] Y. Koren, "Collaborative filtering with temporal dynamics," in *KDD*, 2009, pp. 447–456.
[23] N. D. Lawrence and R. Urtasun, "Non-linear matrix factorization with gaussian processes," in *ICML*, 2009, p. 76.
[24] A. Shapiro and Y. Wardi, "Convergence analysis of gradient descent stochastic algorithms," *Journal of Optimization Theory and Applications*, vol. 91, pp. 439–454, 1996.
[25] Y. Nesterov, "Primal-dual subgradient methods for convex problems," *Mathematical Programming*, vol. 120, no. 1, pp. 221–259, 2009.
[26] L. Xiao, "Dual averaging methods for regularized stochastic learning and online optimization," *J. Mach. Learn. Res.*, vol. 9999, pp. 2543–2596, December 2010.
[27] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Trans. Inf. Syst.*, vol. 20, pp. 422–446, October 2002.