Selection of Negative Samples for One-class Matrix Factorization

Hsiang-Fu Yu* Mikhail Bilenko† Chih-Jen Lin‡

Abstract

Many recommender systems have only implicit user feedback. The two possible ratings are positive and negative, but only part of positive entries are observed. One-class matrix factorization (MF) is a popular approach for such scenarios by treating some missing entries as negative. Two major ways to select negative entries are by sub-sampling a set with similar size to that of observed positive entries or by including all missing entries as negative. They are referred to as "subsampled" and "full" approaches in this work, respectively. Currently detailed comparisons between these two selection schemes on large-scale data are still lacking. One important reason is that the "full" approach leads to a hard optimization problem after treating all missing entries as negative. In this paper, we successfully develop efficient optimization techniques to solve this challenging problem so that the "full" approach becomes practically viable. We then compare in detail the two approaches "subsampled" and "full" for selecting negative entries. Results show that the "full" approach of including much more missing entries as negative yields better results.

1 Introduction

Matrix factorization (MF) is a popular technique for collaborative filtering. With observed ratings given by m users to n items, MF aims to find a model such that we can use it to predict the unobserved rating of a user on an item. To this end, MF learns a model from existing observations by solving the following optimization problem.

$$\min_{W,H} \sum_{(i,j)\in\Omega} (A_{ij} - \boldsymbol{w}_i^\top \boldsymbol{h}_j)^2 + \sum_i \lambda_i \|\boldsymbol{w}_i\|^2 + \sum_j \bar{\lambda}_j \|\boldsymbol{h}_j\|^2, \tag{1.1}$$

where each entry $A_{ij} \in \mathbb{R}$ (e.g., score of 1 to 5) is the rating given by user i to item j and

$$\Omega = \{(i,j) : A_{ij} \text{ is observed}\}$$

is the set of observed ratings. With the regularization parameters $\lambda_i, \bar{\lambda}_j$, the goal is to find two low-rank latent matrices

$$W = \begin{bmatrix} \boldsymbol{w}_1^\top \\ \vdots \\ \boldsymbol{w}_m^\top \end{bmatrix} \in \mathbb{R}^{m\times k} \text{ and } H = \begin{bmatrix} \boldsymbol{h}_1^\top \\ \vdots \\ \boldsymbol{h}_n^\top \end{bmatrix} \in \mathbb{R}^{n\times k}$$

so that $\boldsymbol{w}_i^\top \boldsymbol{h}_j$ is a good approximation of the observed rating A_{ij}. Note that k is a pre-specified latent factor satisfying

$$k \ll m \text{ and } k \ll n.$$

The rating-based MF with $A_{ij} \in \mathbb{R}$ has been well studied in the literature [e.g., 13]. However, in some applications, A_{ij} possesses only two possible values with $A_{ij} \in \{0, 1\}$. We refer to 1 as a positive rating while 0 as a negative rating. Further, only part of positive entries are observed. For example, in [22] for Xbox movies recommendation, we only know movies that have been watched by users. By assuming that users do not watch movies they do not like, we have some partial positive data but lack any negative information. To handle such a one-class scenario, one popular approach [11, 15, 20–22] is to treat some missing entries as negative and the sum of losses in (1.1) becomes

$$\sum_{(i,j)\in\Omega^+} (1 - \boldsymbol{w}_i^\top \boldsymbol{h}_j)^2 + \sum_{(i,j)\in\Omega^-} (0 - \boldsymbol{w}_i^\top \boldsymbol{h}_j)^2, \tag{1.2}$$

where Ω^+ is the set of observed positive entries, and Ω^- includes negative entries sampled from missing entries in A. The rationale is that among the large number of items, a user likes only a small subset of them. Therefore, most of the missing entries in A are negative. Currently two major approaches to select the set Ω^- are:

1. Subsampled: the size of Ω^- is roughly similar to that of Ω^+.

   $$|\Omega^-| = O(|\Omega^+|) \ll mn. \tag{1.3}$$

   Some reasons support this setting. First, given the so few observed positive entries, a large Ω^- may cause serious imbalance between positive and negative entries. Second, using a too large Ω^- causes difficulties to solve the optimization problem. Studies that have considered this setting include [21, 22].

*Amazon Inc. This work was done when H.-F. Yu was in University of Texas at Austin.
†Microsoft Inc.
‡National Taiwan University.

2. Full: all missing entries are considered as negative:

$$\Omega^- = \{(i,j) \mid (i,j) \notin \Omega^+\}. \tag{1.4}$$

One apparent reason of using this setting is that all missing entries are considered. From this viewpoint, the Subsampled approach is just an approximation of the Full approach.

Handling the huge number of $|\Omega^-| = O(mn)$ elements causes the Full approach to be practically infeasible. However, under some circumstances the alternating least squares (ALS) optimization method given in [11, 20] can have similar complexity to that for Subsampled.

At the first glance it is unclear if the two approaches give different performances. Surprisingly, few works have compared them. The main existing study is [21], which reports the following observations on two rather small data sets (thousands of users and items).

- The Subsampled approach is worse than the Full.
- By a bagging approach to select 20 different Ω^- sets and average the resulting predictions, the performance is as good as the Full.

We feel there is a need to detailedly study the two approaches because of the following concerns.

- Experiments in [21] are for small data. Observations may be very different for large-scale data sets.
- The lack of studies caused that in some papers incapable baselines may be used to compare with newly proposed techniques. For example, in [12] to propose a one-class MF approach incorporating meta-features, they compare with a Subsampled setting. It is possible that Full is a better baseline.
- Because of the huge set Ω^- used by the Full approach, traditional optimization methods for MF fail to handle large data. Thus, the Full approach may not be useful even if it can give better models.

In this paper, we make major contributions as follows:

- We propose a novel coordinate descent algorithm for the Full approach which is significantly faster than existing alternating minimization based methods [11, 20, 21].
- We conduct thorough comparisons between Full and Subsampled with their best settings on large-scale data sets..

Our conclusion is that Full yields much better results than Subsampled. With our proposed optimization techniques for training large problems, the Full approach by treating all missing entries as negative becomes practically viable.

One-class MF is a case of PU (positive-unlabeled) learning [10], which includes other important applications such as link prediction [17]. Our proposed optimization methods can be easily applied to them.

Some studies extend the positive versus negative setting to other optimization problems. For example, instead of a squared loss, various ranking loss functions are considered in [16, 25]. We will include one ranking-loss approach in our empirical comparison. In [27], a variable is simultaneously optimized to decide if a missing entry is negative or not. To be focused, in the current work we do not discuss this approach.

Table 1: Notation

m, n, k	numbers of users, items, and latent variables
A	$m \times n$ rating matrix
W, H	$m \times k$ and $n \times k$ latent matrices
$\boldsymbol{w}_i, \boldsymbol{h}_j$	$k \times 1$ vector; ith row of W and jth row of H
$\lambda_i, \bar{\lambda}_j$	regularization parameters
Ω	set of observed entries for standard MF
Ω^+	set of observed positive entries for one-class MF
Ω^-	set of selected negative entries for one-class MF
Ω_i^+	set of user i's observed entries (one-class MF)
$\bar{\Omega}_j^+$	set of item j's observed entries (one-class MF)
C	$m \times n$ matrix for weights of entries

This paper is organized as follows. Section 2 reviews past studies on both Subsampled and Full approaches. In Section 3, we propose methods to solve large optimization problems for the Full approach. Section 4 gives detailed comparisons, while discussions and conclusions are in Section 5. Table 1 gives main notation in this paper. Supplementary materials and code for experiments are available at http://www.csie.ntu.edu.tw/~cjlin/papers/one-class-mf.

2 Existing Studies on Subsampled and Full Approaches

We briefly discuss existing studies of using the Full and the Subsampled settings. This discussion is important for the investigation in Section 3 to propose efficient optimization algorithms. To begin, we extend (1.1) to have the following general weighted MF formulation.

$$\min_{W,H} \sum_{i,j\in\Omega} C_{ij}(A_{ij} - \boldsymbol{w}_i^\top \boldsymbol{h}_j)^2 + \sum_i \lambda_i \|\boldsymbol{w}_i\|^2 + \sum_j \bar{\lambda}_j \|\boldsymbol{h}_j\|^2, \tag{2.5}$$

where C_{ij} is a cost associated with the loss. For one-class MF, the set Ω includes both positive and negative entries:

$$\Omega = \Omega^+ \cup \Omega^-.$$

For each user i we define

$\Omega_i \equiv \Omega_i^+ \cup \Omega_i^-$, where

$\Omega_i^+ \equiv \{j \mid (i,j)$ is an observed entry$\}$, and

$\Omega_i^- \equiv \{j \mid (i,j)$ is a missing entry selected as negative$\}$.

Similarly, for each item j, we let

$$\bar{\Omega}_j^+ \equiv \text{ set of observed entries of item } j.$$

Usually in (2.5) we set

$$\lambda_i = \lambda|\Omega_i^+|,\ \forall i \text{ and } \bar{\lambda}_j = \lambda|\bar{\Omega}_j^+|,\ \forall j. \tag{2.6}$$

2.1 The Full Approach Because Ω^- is a fixed set, weights C_{ij} are the main concern. In [20, 21], they consider $C_{ij} = 1, \forall (i,j) \in \Omega^+$ and the following settings for $C_{ij}, \forall (i,j) \in \Omega^-$.

- C_{ij} is a constant.
- $C_{ij} \propto |\Omega_i^+|$. That is, under the same j, $C_{ij} = |\Omega_i^+|\Delta$, $\forall i$, where Δ is a constant. The reason is that users associated with few items provide less reliable information.
- $C_{ij} \propto n - |\bar{\Omega}_j^+|$. Note that $|\bar{\Omega}_j^+|$ indicates the popularity of item j. Because popular items are less likely to be negative, weights for them should be smaller.

In addition, C_{ij} can be decided by some user and item features. For example, [15] considers

$$C_{ij} = \begin{cases} 1 - \text{sim}(i,j) & \text{if } (i,j) \in \Omega^- \\ 1 & \text{otherwise} \end{cases},$$

where $\text{sim}(i,j)$ = similarity between user i and item j.

2.2 The Subsampled Approach Because Ω^- is no longer a fixed set, we have more options as follows:

- Size of $|\Omega^-|$.
- Sampling strategies for selecting Ω^-.
- Weights C_{ij}.

The number of combined choices is huge. We may further consider a user- or an item-oriented setting. For example, instead of $|\Omega^-| \propto |\Omega^+|$, we can consider $|\Omega_i^-| \propto |\Omega_i^+|$, $\forall i$. Here we show details of two studies that have used the Subsampled approach. [21] considers

$$|\Omega^-| \propto |\Omega^+| \quad \text{and} \quad P_{ij} \propto 1, |\Omega_i^+|, \text{ or } \frac{1}{|\bar{\Omega}_j^+|},$$

where P_{ij} is the probability to select $(i,j) \notin \Omega^+$ as negative. Their settings of P_{ij} follow from the same explanation in Section 2.1 for choosing C_{ij}. In another study [22], for a baseline setting they have

$$|\Omega_i^-| = |\Omega_i^+| \quad \text{and} \quad P_{ij} \propto |\bar{\Omega}_j^+|.$$

Neither papers specifically says their C_{ij} values. However, given that weight information has been used for selecting Ω^-, some simple settings such as $C_{ij} = 1$ may suffice. The discussion shows that deciding a suitable sampling scheme is not easy. We see that these two studies use opposite strategies: one has $P_{ij} \propto 1/|\bar{\Omega}_j^+|$, while the other has $P_{ij} \propto |\bar{\Omega}_j^+|$.

Table 2: A summary of results in Section 3 by showing the complexity per iteration of optimization methods. For ALS and CD, an iteration means to update W and H once, while for SG it means to conduct $|\Omega|$ SG updates. *: It remains a challenge to apply SG to one-class MF; see Section 3.2.

	ALS	CD	SG
General MF			
	$O(\|\Omega\|k^2 + (m+n)k^3)$	$O(\|\Omega\|k)$	$O(\|\Omega\|k)$
One-class MF (Full approach)			
Direct	$O(mnk^2 + (m+n)k^3)$	$O(mnk)$	$O(mnk)$
New	$O(\|\Omega^+\|k^2 + (m+n)k^3)$	$O(\|\Omega^+\|k + (m+n)k^2)$	NA*

3 Efficient Algorithms for the Full Approach

In this section, we propose methods to solve the large optimization problem for the Full approach. To handle the difficulty caused by $|\Omega| = mn$, [20] assumes that weights C_{ij} are under some conditions.[1] We consider a similar setting by assuming that

$$C_{ij} = \begin{cases} 1 & \text{if } (i,j) \in \Omega^+ \\ p_i q_j & \text{otherwise} \end{cases} \text{ and } A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in \Omega^+ \\ \bar{a} & \text{otherwise,} \end{cases} \tag{3.7}$$

where $\boldsymbol{p} \in \mathbb{R}^m$ and $\boldsymbol{q} \in \mathbb{R}^n$ are vectors. All existing settings discussed in Section 2.1 except [15] satisfy this assumption. For example, if $C_{ij} \propto |\Omega_i^+|, \forall (i,j) \notin \Omega^+$, then we have $C_{ij} = p_i q_j$ with $p_i = |\Omega_i^+|$. For negative entries we consider a slightly more general setting of $A_{ij} = \bar{a}$, although in practice $\bar{a} = 0$ is most used.

We discuss three optimization approaches in this section.

- ALS: Alternating Least Squares
- CD: Coordinate Descent
- SG: Stochastic Gradient

Table 2 lists the complexity per iteration, where details are in subsections. Clearly an mn term causes prohibitive running time if these methods are directly applied to the optimization problem of the Full approach. For ALS, [11, 20] have successfully reduced Full's complexity. We derive their method in Section A.1 of supplementary materials using our notation. Subsequently we discuss details of the other two optimization approaches.

3.1 Coordinate Descent (CD) The early use of CD for MF was in [7], but here we consider the efficient implementation in [29]. The idea is to update one column of W and H at a time. Specifically, if the tth

[1] The condition in [11] is a special case of [20], where they assume $C_{ij} \geq 1$, $\forall (i,j) \in \Omega^+$ and $C_{ij} = 1$ otherwise.

Algorithm 1 Coordinate descent for one-class MF.

while not optimal **do**
  for $t = 1, \ldots, k$ **do**
    Let W, H's tth columns be initial $\boldsymbol{u}$, $\boldsymbol{v}$
    Approximately solve (3.8) by ALS:
    for $s = 1, \ldots, S$ **do**
      Solve (3.8) by fixing $\boldsymbol{v}$
      Solve (3.8) by fixing $\boldsymbol{u}$
    Let $\boldsymbol{u}$, $\boldsymbol{v}$ be W, H's tth columns

column is chosen, we let two vector variables

$$\boldsymbol{u} \in \mathbb{R}^m \text{ and } \boldsymbol{v} \in \mathbb{R}^n$$

denote the corresponding columns in W and H, respectively and form the following optimization problem.

$$\min_{\boldsymbol{u},\boldsymbol{v}} \sum_{(i,j)\in\Omega} C_{ij}(\hat{R}_{ij} - u_i v_j)^2 + \sum_i \lambda_i u_i^2 + \sum_j \bar{\lambda}_j v_j^2, \tag{3.8}$$

where

$$\hat{R}_{ij} \equiv A_{ij} - \boldsymbol{w}_i^\top \boldsymbol{h}_j + W_{it}H_{jt}. \tag{3.9}$$

Note that we add the constant $W_{it}H_{jt}$ to $\hat{R}_{ij}$ so that $\hat{R}_{ij} - u_i v_j$ is the error for the (i, j) entry. Problem (3.8) is like an MF problem of using only one latent variable (i.e., $k = 1$), so we can solve it by ALS. Our procedure is described in Algorithm 1. Essentially we have a *two-level* CD procedure. At the outer level, sequentially a pair of columns from W and H are selected, while at the inner level, these two columns are alternatively updated several times; see the S inner iterations in Algorithm 1. For rating-based MF, [30] has shown that taking some inner iterations leads to shorter running time than conducting only one update. In general S is a small constant; here we use 5.

Next we discuss details for solving (3.8) when $\boldsymbol{v}$ is fixed. By a similar derivation for (A.2), the optimal solution is

$$u_i = \Big(\sum_{j\in\Omega_i} C_{ij}v_j^2 + \lambda_i\Big)^{-1}\Big(\sum_{j\in\Omega_i} C_{ij}\hat{R}_{ij}v_j\Big). \tag{3.10}$$

If $\hat{R}_{ij}, \forall j \in \Omega_i$ are available (see implementation details in Section 3.1.1), the number of operations for (3.10) is $O(|\Omega_i|)$. Therefore, the cost of going through all columns in W and H is

$$O(|\Omega|) \times k. \tag{3.11}$$

A comparison with (A.3) shows that CD is more efficient than ALS.

For the two approaches (Subsampled and Full) for one-class MF, $|\Omega|$ in (3.11) becomes $|\Omega^+|$ and mn, respectively. Because mn is too large, we investigate if it can be reduced to $O(|\Omega^+|)$ under the assumption in (3.7). The first term of (3.10) can be rewritten as

$$\sum_{j\in\Omega_i} C_{ij}v_j^2 = \sum_{j\in\Omega_i^+} (1 - p_i q_j)v_j^2 + p_i \sum_{j=1}^n q_j v_j^2, \tag{3.12}$$

where $\sum_{j=1}^n q_j v_j^2$ is independent of i and can be precomputed in $O(n)$. Then the cost of (3.12) is $O(|\Omega_i^+|)$. For the second term in (3.10), using (3.7) and (3.9) we have

$$\sum_{j\in\Omega_i} C_{ij}\hat{R}_{ij}v_j = \sum_{j\in\Omega_i^+} ((1 - p_i q_j)\hat{R}_{ij} + p_i q_j(A_{ij} - \bar{a}))v_j$$
$$+ p_i \sum_{j=1}^n q_j(\bar{a} - \boldsymbol{w}_i^\top \boldsymbol{h}_j + W_{it}H_{jt})v_j. \tag{3.13}$$

If $\hat{R}_{ij}, \forall j \in \Omega_i^+$ are available, the first summation in (3.13) can be calculated in $O(|\Omega_i^+|)$ time. The second summation can be written as

$$\bar{a}\sum_{j=1}^n q_j v_j - \boldsymbol{w}_i^\top \sum_{j=1}^n q_j \boldsymbol{h}_j v_j + W_{it}\sum_{j=1}^n q_j H_{jt} v_j, \tag{3.14}$$

in which each summation is independent of i. The main computational task is $\sum_{j=1}^n q_j \boldsymbol{h}_j v_j$ that can be precomputed in $O(nk)$. Therefore, the cost for updating $\boldsymbol{u}$ is

$$O(|\Omega^+| + nk).$$

Then the cost to go through all W and H's k columns is

$$O(|\Omega^+|k + (m + n)k^2). \tag{3.15}$$

If k is not large, in general $(m + n)k^2$ is no more than $|\Omega^+|k$. Further, if a small number S of inner iterations are taken, the $O((m+n)k^2)$ operations needed before them becomes a smaller portion of the total cost. Therefore, our procedure for the Full approach has comparable complexity to that for the Subsampled.

In the above analysis we assume that $\hat{R}_{ij}, j \in \Omega_i^+$ are available. In Section 3.1.1, we show that they can be obtained in $O(|\Omega_i^+|)$ cost, the same as other operations for the first term in (3.13). We also discuss other implementation details such as column or row access of W and H.

3.1.1 Implementation Details The discussion so far assumes that $\hat{R}_{ij}$ defined in (3.9) are available. Here we investigate how they can be cheaply maintained. We

begin with discussing the situation for rating-based MF and then extend the result to one-class MF. We note that for $(i,j) \in \Omega$,

$$\hat{R}_{ij} = A_{ij} - \boldsymbol{w}_i^\top \boldsymbol{h}_j + W_{it}H_{jt}.$$

Directly computing $\hat{R}_{ij}$ requires $O(k)$ operations for the dot product $\boldsymbol{w}_i^\top \boldsymbol{h}_j$. Thus, the construction of (3.8) costs $O(|\Omega|k)$ operations. In [29], the time complexity can be reduced to $O(|\Omega|)$ by maintaining the following residual R_{ij}

$$R_{ij} = A_{ij} - \boldsymbol{w}_i^\top \boldsymbol{h}_j, \ \forall (i,j) \in \Omega.$$

If R_{ij} is available, then each $\hat{R}_{ij}$ can be computed in $O(1)$ without the dot product:

$$\hat{R}_{ij} \leftarrow R_{ij} + W_{it}H_{jt}, \forall (i,j) \in \Omega. \tag{3.16}$$

The total cost for (3.16) is $O(|\Omega|)$. Maintaining R_{ij} after $\boldsymbol{u}, \boldsymbol{v}$ are obtained can also be done in $O(|\Omega|)$.

$$R_{ij} \leftarrow \hat{R}_{ij} - u_i v_j, \forall (i,j) \in \Omega.$$

Therefore, in one iteration of going through W and H's all columns, the maintenance cost of R and $\hat{R}$ is the same as that in (3.11) for updating W and H.

Now consider the one-class scenario. For the Full approach, $O(|\Omega|) = O(mn)$ is too high to maintain all R_{ij}. However, based on the observation that only $\hat{R}_{ij}\ \forall (i,j) \in \Omega^+$ are involved in (3.13), we can store and maintain only $R_{ij}\ \forall (i,j) \in \Omega^+$. As a result, the total time/space cost to maintain the residual is $O(|\Omega^+|)$.

Next we discuss an issue of memory usage. In (3.13), for

$$-\boldsymbol{w}_i^\top \sum_{j=1}^{n} q_j \boldsymbol{h}_j v_j + W_{it} \sum_{j=1}^{n} q_j H_{jt} v_j, i = 1, \ldots, m, \tag{3.17}$$

both W and H's rows ($\boldsymbol{w}_i$, $\boldsymbol{h}_j$) and columns ($H_{jt}, j = 1, \ldots, n$) are needed, but in practice one does not want to double the storage by storing W (or H) in both row-oriented and column-oriented formats. Here we demonstrate that all operations can be conducted by using the column-oriented format. A careful check shows that the m values in (3.17) can be calculated by

$$-\sum_{s:s\neq t} \bar{\boldsymbol{w}}_s \bar{\boldsymbol{h}}_s^\top \begin{bmatrix} q_1 v_1 \\ \vdots \\ q_n v_n \end{bmatrix}, \tag{3.18}$$

where $\bar{\boldsymbol{w}}_s, \bar{\boldsymbol{h}}_s, s = 1, \ldots, k$ are column vectors in W and H. That is,

$$W = \begin{bmatrix} \bar{\boldsymbol{w}}_1 & \cdots & \bar{\boldsymbol{w}}_k \end{bmatrix} \text{ and } H = \begin{bmatrix} \bar{\boldsymbol{h}}_1 & \cdots & \bar{\boldsymbol{h}}_k \end{bmatrix}.$$

Note that (3.18) can be pre-calculated in $O(nk)$ before $u_i, i = 1, \ldots, m$ are updated. Further, in (3.18) all we need is to access W and H column-wisely. Regarding the calculation of $\hat{R}_{ij}$ in (3.16), we can extract the t-th column of W and H, and then go through all $(i,j) \in \Omega^+$.

3.1.2 Related Works An earlier study that has reduced the complexity of ALS to (3.15) is [23]. It is indeed a combination of ALS and CD. Under fixed H, instead of calculating the closed-form solution (A.2), they solve (A.1) by a fixed number of CD iterations. It has been shown in [30, Section 3.4] that for rating-based MF, the CD procedure considered here is much faster than the approach by [23]. Besides, their procedure is more complicated for needing the eigen-decomposition of a k by k matrix.

The recent work on PU (positive-unlabeled) learning [10] has mentioned that the CD framework in [29] can be modified to have the complexity (3.15), but detailed derivations are not given.

3.2 Stochastic Gradient (SG) SG has been extensively used for MF [e.g., 13]. It reformulates (2.5) to the following objective function.

$$\min_{W,H} \sum\nolimits_{(i,j)\in\Omega} \ell_{ij}(A_{ij}, \boldsymbol{w}_i^\top \boldsymbol{h}_j), \tag{3.19}$$

where $\ell_{ij} = C_{ij}(A_{ij} - \boldsymbol{w}_i^\top \boldsymbol{h}_j)^2 + \frac{\lambda_i}{|\Omega_i|}\|\boldsymbol{w}_i\|^2 + \frac{\bar{\lambda}_j}{|\bar{\Omega}_j|}\|\boldsymbol{h}_j\|^2$. Note that the regularization term is averaged in each l_{ij} because SG would like the expectation on one single instance to be the same as the whole. Taking $\|\boldsymbol{w}_i\|^2$ as an example, we can see that

$$\sum_{j\in\Omega_i} \frac{\lambda_i}{|\Omega_i|}\|\boldsymbol{w}_i\|^2 = \lambda_i \|\boldsymbol{w}_i\|^2.$$

Recall that in (2.6) $\lambda_i = \lambda|\Omega_i^+|$ and $\lambda_j = \lambda|\bar{\Omega}_j^+|$; therefore, in the rating-based MF, where $\Omega = \Omega^+$, we have $\lambda_i/|\Omega_i| = \bar{\lambda}_j/|\bar{\Omega}_j| = \lambda$, which is exactly the choice in the common SG update rule used in [13]. At each step SG randomly selects an entry $(i,j) \in \Omega$ uniformly and updates $\boldsymbol{w}_i$ and $\boldsymbol{h}_j$ using the partial gradient.

$$\boldsymbol{w}_i \leftarrow \boldsymbol{w}_i - \eta \nabla_{\boldsymbol{w}_i} \ell_{ij}(A_{ij}, \boldsymbol{w}_i^\top \boldsymbol{h}_j),$$
$$\boldsymbol{h}_j \leftarrow \boldsymbol{h}_j - \eta \nabla_{\boldsymbol{h}_j} \ell_{ij}(A_{ij}, \boldsymbol{w}_i^\top \boldsymbol{h}_j),$$

where η is the learning rate. Because learning rates may significantly affect the convergence speed of SG, we adjust them by the advanced setting in [5]. Each SG update costs $O(k)$ operations. For SG usually an outer iteration refers to $|\Omega|$ updates, so the cost per iteration is $O(|\Omega| \times k)$. A huge difference between Subsampled and Full occurs because $|\Omega| = |\Omega^+|$ and mn, respectively. Implementing the Full approach faces two challenges:

1 Because we store only $A_{ij}, (i,j) \in \Omega^+$ rather than $\bar{a}$ of Ω^-, for any picked (i,j), an efficient mechanism is needed to check if it is in Ω^+. However, the implementation is not easy. For example, it takes $O(\log|\Omega^+|)$ using binary search and $O(1)$ using hash. Neither is very efficient.
2 The $O(mnk)$ computational cost to go through the entire Ω is prohibitive.

To address the first issue, we reformulate (3.19) to

$$\sum_{(i,j)\in\Omega^+} \ell^+_{ij}(A_{ij}, \boldsymbol{w}_i^\top \boldsymbol{h}_j) + \sum_{i=1}^{m}\sum_{j=1}^{n} \ell^-_{ij}(\bar{a}, \boldsymbol{w}_i^\top \boldsymbol{h}_j), \tag{3.20}$$

where

$$\ell^+_{ij} = C_{ij}(A_{ij} - \boldsymbol{w}_i^\top \boldsymbol{h}_j)^2 - p_i q_j (\bar{a} - \boldsymbol{w}_i^\top \boldsymbol{h}_j)^2 + \lambda \frac{|\Omega_i^+|\|\boldsymbol{w}_i\|^2}{n + |\Omega_i^+|} + \frac{\lambda|\bar{\Omega}_j^+|\|\boldsymbol{h}_j\|^2}{m + |\bar{\Omega}_j^+|},$$

$$\ell^-_{ij} = p_i q_j (\bar{a} - \boldsymbol{w}_i^\top \boldsymbol{h}_j)^2 + \frac{\lambda|\Omega_i^+|\|\boldsymbol{w}_i\|^2}{n + |\Omega_i^+|} + \frac{\lambda|\bar{\Omega}_j^+|\|\boldsymbol{h}_j\|^2}{m + |\bar{\Omega}_j^+|}.$$

Note that the regularization term is re-distributed because $mn + |\Omega^+|$ terms are now involved. We design the following procedure to perform updates on ℓ^+_{ij} or ℓ^-_{ij}.

1 Randomly choose $u \in (0,1)$.
2 If

$$u < \frac{|\Omega^+|}{mn + |\Omega^+|},$$

randomly select $(i,j) \in \Omega^+$, and use $\nabla_{\boldsymbol{w}_i}\ell^+_{ij}$ and $\nabla_{\boldsymbol{h}_j}\ell^+_{ij}$ to update $\boldsymbol{w}_i$ and $\boldsymbol{h}_j$.
Otherwise, randomly select $(i,j) \in \{1,\dots,m\} \times \{1,\dots,n\}$, and use $\nabla_{\boldsymbol{w}_i}\ell^-_{ij}$ and $\nabla_{\boldsymbol{h}_j}\ell^-_{ij}$ to update $\boldsymbol{w}_i$ and $\boldsymbol{h}_j$.

The procedure effectively alleviates the issue of checking if $(i,j) \in \Omega^+$ or not. It also generates an un-biased gradient estimate to the original optimization problem (2.5). See details in Section A.2.

For the second challenge, unfortunately we have not devised a good strategy to reduce the mn term to $O(|\Omega^+|)$. From the investigation of ALS and CD, the key to remove the $O(mn)$ calculation is that for computing $\sum_{j=1}^{n}(\cdots)$, $\forall i$, we can reformulate it to

$$(\text{terms related to } i) \times \sum\nolimits_j (\text{terms related to } j), \forall i;$$

see, for example, the calculation in (3.14). Then the summation over j can be pre-computed. Therefore, we must be able to aggregate things related to i (or j) in the algorithm. This is very different from the design of SG, in which an individual (i,j) is chosen at a time. We may modify SG in various ways. For example, in (3.20), the gradient of the second summation can be calculated without the $O(mn)$ cost, so we can conduct SG updates for terms in the first summation over Ω^+, while apply regular gradient descent for the second. Another possibility is to run SG in the ALS framework. That is, when H is fixed, we apply SG to update W. Unfortunately, these modifications move SG toward other methods such as ALS. Such modifications may be unnecessary as we can directly apply ALS or CD.

One past study that has observed the difficulty of sampling from the huge number of entries is [24]. In a ranking setting they show that SG converges slowly by uniform sampling. They then propose a context-dependent setting to oversample informative entries. However, such settings do not guarantee the complexity reduction like what we achieved for ALS and CD. Further, existing methods to parallelize SG for MF such as [4, 8] may become not applicable because A is split into blocks. In contrast, ALS and CD under our new settings for one-class MF can be easily parallelized.

Based on the discussion, SG may be less suitable for the Full approach. We experimentally confirm this result in Section 4.

Table 3: Data statistics for training and test sets. Zero columns/rows in A are removed, so m and n may be different from those of the original data. Data sets ml1m and ml10m are movielens with 1m and 10m (user, item) pairs.

	delicious	ml1m	ml10m	netflix	yahoo-music
m	2,000	6,040	69,878	480,189	1,000,990
n	3,000	3,952	10,677	17,770	624,961
$\|\Omega^+\|$	197,130	517,770	4,505,820	51,228,351	82,627,856
$\|\Omega^+_{\text{test}}\|$	49,306	57,511	499,864	5,690,839	9,178,962

4 Experimental Results

Our experiments include two parts. The first part is the comparison of the proposed optimization methods for the Full approach (Section 4.1) in terms of the computational efficiency. In the second part we investigate in Section 4.2 the performances of the following approaches for one-class MF:

- Full: in the experiments for Full approaches, we consider a simplified setting of (3.7) as follows.

$$C_{ij} = \begin{cases} 1 & \forall (i,j) \in \Omega^+, \\ \alpha, & \forall (i,j) \in \Omega^-. \end{cases} \tag{4.21}$$

  The selection of the parameter α will be discussed in Section C.1.
- Subsampled: see Section B.1.1 for details about the variants of the Subsampled approaches.

- Ensemble: the ensemble of models from the Subsampled approach. See details in Section B.1.1
- BPR: The approach in [25] by considering an AUC (i.e., rank-based) loss; see details in Section B.1.2. We use the implementation in [6].

Datasets. The only publicly available one-class MF data that we are aware of is delicious from [21].[2] We use the same 4-to-1 training/test split by [21] in our experiment.[3] Besides this data, following past works [12], we modify some publicly available rating-based MF data in Table 3 for experiments. We consider observed entries with ratings ≥ 4 as positive.[4] For some problems, training and test sets are available, but the test sets are too small. Therefore, other than delicious, we merge training and test sets of every problem first, and then do a 9-to-1 split to obtain training/test sets for our experiments. See the implementation details and evaluation criteria in Sections B.1 and B.2, respectively.

4.1 Comparison: Optimization for Full

We compare the proposed optimization methods in Section 3 for the Full approach. For ALS and CD, we use the methods in [20] (see also Section A) and in Section 3.1, respectively. They are superior to direct implementations on all mn values. For SG, we consider some variants for checking the effectiveness of the reformulation (3.20).

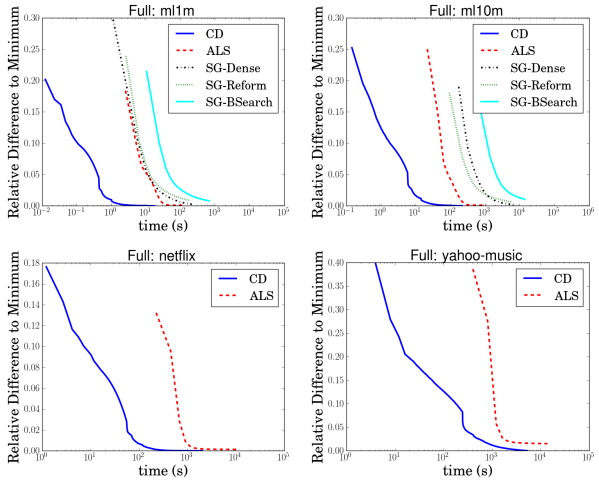
1 SG-Dense: the original formulation (3.19) is used and the whole Ω set (mn elements) is stored for easily checking if $(i,j) \in \Omega^+$ or not.
2 SG-BSearch: we still use (3.19), but do not store Ω. A binary search is conducted to check if $(i,j) \in \Omega^+$.
3 SG-Reform: the reformulation (3.20) is used.

Because the goal is to compare methods for the same optimization problem, we simply set $\alpha = 0.1$ in (4.21). That is, $C_{ij} = 0.1, \forall (i,j) \notin \Omega^+$ and 1 otherwise. Other settings include $k = 64$ and $\lambda = 0.1$.

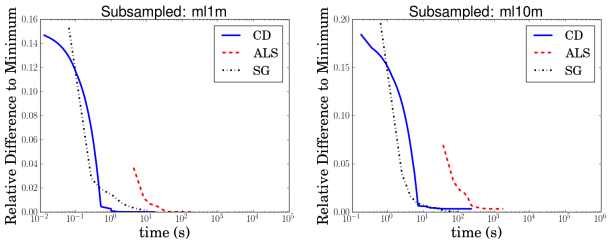
In Figure 1(a), we present the relation between running time (log-scaled) and the relative difference to the minimum objective value obtained among all the algorithms. Results show that CD is much faster than ALS. This finding is expected because past studies for general MF has shown the superiority of CD. Now the mn terms in the complexity analysis of both methods are reduced to $O(|\Omega^+|)$, so the relationship still holds.

For smaller data sets ml1m and ml10m, SG-Dense is applicable by storing all mn elements of Ω. We observe that SG-Reform is close to SG-Dense, so our reformulation in (3.20) is effective. SG-BSearch is the slowest because of the binary search on checking if $(i,j) \in \Omega^+$. Unfortunately, all the SG variants are significant slower than CD and ALS for larger problems like netflix or yahoo-music, for which curves cannot even be generated in Figure 1(a).

Although we have explained in Section 3 that SG may be less suitable for Full, the very poor results are still somewhat surprising. Therefore, we investigate the running speed when these methods are applied to the Subsampled approach. We consider $|\Omega^-| = |\Omega^+|$, uniform sampling strategy, and the optimization problem (1.2). The comparison results are in Figure 1(b). Clearly SG becomes as fast as CD and much better than ALS. This result is consistent with past studies on rating-based MF. We thus conclude that SG fails for the Full approach because of the large mn elements.



(a) Full (SG may be too slow to be shown)



(b) Subsampled

Figure 1: Comparison of optimization methods. The y-axis is the relative difference to the minimal value obtained among all the methods, while the x-axis is the running time in log-scale. See Section 4.1 for details of SG variants.

[2] A tag recommendation data set from ECML'09 challenge is used in [24], but it is in fact a multi-label data set, where the tagging behavior does not reveal user preference. Thus, this data set is not suitable in our experiments. Furthermore, the data set is relatively small.

[3] We use the first 4-to-1 split from http://www.rongpan.net/data/delicious.tar.bz2.

[4] For yahoo-music, scores ≥ 80 are considered as positive.

Further, by comparing Figures 1(a) and 1(b), CD takes similar time for Full and Subsampled, and so does our ALS implementation. Therefore, our study enables the Full approach to be computationally feasible for one-class MF.

4.2 Comparison: Full versus Subsampled We compare one-class MF approaches listed in the beginning of this section. After finding that Full is better in initial experiments, we consider the following settings to check if the difference between Full and others is significant.

- For Full, we conduct parameter selection for each evaluation criterion by splitting the training set to two parts for training (90%) and evaluation (10%). The parameters achieving the best validation results are used to train the final model. We then report the performance on the test set.
- For Subsampled, Ensemble and BPR, we omit the validation procedure. Instead, for each evaluation criterion, we consider all parameter combinations (including several *sampling strategies* for Subsampled) and report the best test result. Therefore, we overfit the test set to get an optimistic estimate of the performance. Moreover, we combine 20 Subsampled models [21] as our Ensemble model.[5]

To see if Full is significantly better, we deliberately overestimate the performance of others. Parameters used for each method are in Table 5. In the testing phase Ω^+ is excluded because they may have been well fitted in training.

Table 4 shows that for nDCG, nHLU, and MAP, Full is significantly better than all other approaches. For AUC, all approaches give similar values close to one. For a recommender system, AUC is less suitable because it considers the overall number of violating pairs without emphasizing the position of top-ranked (i.e., recommended) items.

Although Ensemble improves the performance of the pure Subsampled approach on ml1m, ml10m, and netflix, the performance gap between Ensemble and Full is still large in Table 4. This observation is very different from the finding in [21], where Ensemble is able to yield competitive performance as the Full approach. Based on the observation in Table 4 that the larger the size of the data set, the larger the gap between Ensemble and Full, we think that the finding regarding the Ensemble performance in [21] only holds for small data sets.[6]

Table 4: Comparison of one-class MF approaches. The best performed method for each criterion is bold-faced.

		nDCG @1	nDCG @10	nHLU	MAP	AUC
delicious	Subsampled	41.40	27.80	26.63	14.44	0.86635
	Ensemble	44.15	32.21	31.25	18.98	0.89196
	BPR	21.95	27.68	27.98	16.51	0.88889
	Full	**56.45**	**38.53**	**36.86**	**21.51**	**0.89244**
ml1m	Subsampled	14.45	14.79	15.62	11.14	0.93694
	Ensemble	21.80	20.67	21.39	15.06	**0.94969**
	BPR	8.10	15.80	17.07	11.28	0.94474
	Full	**28.91**	**23.66**	**24.11**	**16.35**	0.94536
ml10m	Subsampled	10.71	10.30	10.79	8.22	0.97255
	Ensemble	15.99	18.51	19.77	13.97	**0.97958**
	BPR	16.28	16.73	17.67	12.24	0.97700
	Full	**25.64**	**23.81**	**24.94**	**17.70**	0.97372
netflix	Subsampled	10.25	9.96	10.48	7.54	0.97196
	Ensemble	14.91	15.00	15.60	10.50	**0.97618**
	BPR	18.28	15.84	16.07	10.33	0.97583
	Full	**27.05**	**22.59**	**22.69**	**13.92**	0.96962
yahoo-music	BPR	12.36	12.46	13.12	8.74	**0.99546**
	Full	**38.67**	**34.74**	**35.28**	**26.78**	0.99290

Note that for the data set delicious used in [21], here we see a bigger performance gap between Full and Ensemble. The reason might be that first our k is 64 rather than their 16, and second we have selected the parameter α for Full by a validation procedure.

5 Discussions and Conclusions

Before giving conclusions, we discuss some interesting issues and analysis in this section and Section C of supplementary materials.

Superiority of Full over Subsampled. For Subsampled, by selecting only a small subset of missing entries as negative data, we fit them by using W and H but ignore others. Then the generalization ability may not be good. The importance of considering all missing data can also be seen in the best α selected by the validation procedure; see (4.21) for the definition of α. Because of the large number of negative data, we expect that α should be small. Interestingly, we find that α can be neither too small nor too large. A too small α causes W and H to underfit negative missing data, while a too large α causes W and H to wrongly fit positive missing data. In the theoretical study of PU (positive-unlabeled) learning [10], the authors were able to prove an error bound under the setting of $\alpha = \bar{\rho}/(2-\bar{\rho})$, where $\bar{\rho}$ is the percentage of positive entries that are

[5] In the published version of the paper, the parameters of the Ensemble models were set as the same best parameter of Subsampled models for each individual model. We applied a grid search on Ensemble models and all the results of Table 4 have been refreshed by Sheng-Wei Chen (D09944003@ntu.edu.tw) in November 2020.

[6] The statistics of two data sets used in [21] are $m = 3,158; n = 1,536; |\Omega^+| = 84,117$ and $m = 3,000; n = 2,000; |\Omega^+| = 246,436$, respectively.

observed. When $\bar{\rho} \to 1$, most missing entries are negative, so $\alpha \approx 1$ causes $\boldsymbol{w}_i^\top \boldsymbol{h}_j$ to fit 0 for missing data. In contrast, when $\bar{\rho} \to 0$, some missing entries are indeed positive, so we should not let $\boldsymbol{w}_i^\top \boldsymbol{h}_j$ be very close to zero. Therefore, α should be smaller.

Connection to word2vec. Recently, word2vec [18] for NLP applications identifies a latent vector representing each word from a set of observed word-context pairs. It can be considered as a one-class MF problem [14]. Currently negative entries are selected by the Subsampled approach. In particular, the skip gram negative sampling (SGNS) objective [14] tries to maximize the probability for the observed word-context pairs while simultaneously maximizing that for unobserved (i.e., negative) pairs. Under the assumption that a randomly selected context for a given word is likely not an observed pair, SGNS randomly subsamples a few "negative" contexts from the entire set of contexts for each observed word-context pair. One can see that SGNS is essentially a Subsampled approach for word embedding learning. Thus, our investigation on Subsampled versus Full may be useful for this NLP technique. It is also interesting to ask whether the efficient techniques developed in this paper can be extended to handle non squared-L_2 loss functions such as the one used in word2vec.

Conclusions. In this paper, we have developed efficient techniques to solve the hard optimization problem of the Full approach, which treats every missing entry as negative. We then conduct thorough experiments to show that the Full approach gives much better performances than the Subsampled approach. Therefore, our work has made the Full approach very useful for large-scale one-class matrix factorization.

References

[1] Y. Bachrach, Y. Finkelstein, R. Gilad-Bachrach, L. Katzir, N. Koenigstein, N. Nice, and U. Paquet. Speeding up the Xbox recommender system using a Euclidean transformation for inner-product spaces. In *RecSys*, pages 257–264, 2014.

[2] G. Ballard, A. Pinar, T. G. Kolda, and C. Seshadri. Diamond sampling for approximate maximum all-pairs dot-product (MAD) search. In *ICDM*, 2015.

[3] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, 1998.

[4] W.-S. Chin, Y. Zhuang, Y.-C. Juan, and C.-J. Lin. A fast parallel stochastic gradient method for matrix factorization in shared memory systems. *ACM TIST*, 6:2:1–2:24, 2015.

[5] W.-S. Chin, Y. Zhuang, Y.-C. Juan, and C.-J. Lin. A learning-rate schedule for stochastic gradient methods to matrix factorization. In *PAKDD*, 2015.

[6] W.-S. Chin, B.-W. Yuan, M.-Y. Yang, Y. Zhuang, Y.-C. Juan, and C.-J. Lin. LIBMF: A library for parallel matrix factorization in shared-memory systems. *JMLR*, 17(86):1–5, 2016.

[7] A. Cichocki and A.-H. Phan. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE Transaction on Fundamentals*, E92-A:708–721, 2009.

[8] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *KDD*, pages 69–77, 2011.

[9] P. Gopalan, J. M. Hofman, and D. M. Blei. Scalable recommendation with hierarchical poisson factorization. In *UAI*, pages 326–335, 2015.

[10] C.-J. Hsieh, N. Natarajan, and I. Dhillon. PU learning for matrix completion. In *ICML*, 2015.

[11] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, 2008.

[12] N. Koenigstein and U. Paquet. Xbox movies recommendations: Variational bayes matrix factorization with embedded feature selection. In *RecSys*, 2013.

[13] Y. Koren, R. M. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42, 2009.

[14] O. Levy and Y. Goldberg. Neural word embedding as implicit matrix factorization. In *NIPS*, pages 2177–2185, 2014.

[15] Y. Li, J. Hu, C. Zhai, and Y. Chen. Improving one-class collaborative filtering by incorporating rich user information. In *CIKM*, 2010.

[16] D. Lim, J. McAuley, and G. Lanckriet. Top-N recoomendation with missing implicit feedback. In *RecSys*, pages 309–312, 2015.

[17] A. K. Menon and C. Elkan. Link prediction via matrix factorization. In *ECML/PKDD*, 2011.

[18] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[19] B. Neyshabur and N. Srebro. On symmetric and asymmetric LSHs for inner product search. In *Proceedings of the Thirty-Second International Conference on Machine Learning (ICML)*, pages 1926–1934, 2015.

[20] R. Pan and M. Scholz. Mind the gaps: Weighting the unknown in large-scale one-class collaborative filtering. In *KDD*, 2009.

[21] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *ICDM*, 2008.

[22] U. Paquet and N. Koenigstein. One-class collaborative filtering with random graphs. In *WWW*, 2013.

[23] I. Pilászy, D. Zibriczky, and D. Tikk. Fast ALS-based matrix factorization for explicit and implicit feedback datasets. In *RecSys*, 2010.

[24] S. Rendle and C. Freudenthaler. Improving pairwise learning for item recommendation from implicit feedback. In *WSDM*, 2014.

[25] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*, 2009.

[26] A. Shrivastava and P. Li. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *NIPS*, pages 2321–2329, 2014.

[27] V. Sindhwani, S. S. Bucak, J. Hu, and A. Mojsilovic. One-class matrix completion with low-density factorizations. In *ICDM*, 2010.

[28] R. C. Whaley, A. Petitet, and J. J. Dongarra. Automated empirical optimizations of software and the atlas project. *Parallel Comput.*, 27:3–35, 2001.

[29] H.-F. Yu, C.-J. Hsieh, S. Si, and I. S. Dhillon. Scalable coordinate descent approaches to parallel matrix factorization for recommender systems. In *ICDM*, 2012.

[30] H.-F. Yu, C.-J. Hsieh, S. Si, and I. S. Dhillon. Parallel matrix factorization for recommender systems. *KAIS*, 41: 793–819, 2014.

Supplementary Materials:

A Additional Details for Algorithms

A.1 Alternating Least Squares (ALS) Alternating Least Squares (ALS) has been a popular optimization method since the beginning of matrix factorization. It iteratively updates W and H by the following loop

1: **while** not optimal **do**
2: Solve (2.5) by fixing H
3: Solve (2.5) by fixing W

Consider the situation when H is fixed. We would like to solve for $i = 1, \ldots, m$,

(A.1)
$$\min_{\boldsymbol{w}_i} \quad f(\boldsymbol{w}_i) \equiv \sum_{j \in \Omega_i} C_{ij}\big(A_{ij} - \boldsymbol{w}_i^\top \boldsymbol{h}_j\big)^2 + \lambda_i \|\boldsymbol{w}_i\|^2.$$

For a quadratic function like $f(\boldsymbol{w}_i)$, by

$$f(\boldsymbol{w}_i) = f(\boldsymbol{0}) + \nabla f(\boldsymbol{0})^\top \boldsymbol{w}_i + \frac{1}{2}\boldsymbol{w}_i^\top \nabla^2 f(\boldsymbol{0}) \boldsymbol{w}_i,$$

the optimal $\boldsymbol{w}_i = -\big[\nabla^2 f(\boldsymbol{0})\big]^{-1} \nabla f(\boldsymbol{0})$, where

$$\nabla f(\boldsymbol{w}_i) = -\sum_{j \in \Omega_i} C_{ij}(A_{ij} - \boldsymbol{w}_i^\top \boldsymbol{h}_j)\boldsymbol{h}_j + 2\lambda_i \boldsymbol{w}_i,$$
$$\nabla^2 f(\boldsymbol{w}_i) = \sum_{j \in \Omega_i} C_{ij} \boldsymbol{h}_j \boldsymbol{h}_j^\top + 2\lambda_i I_k.$$

Thus, the optimal solution is

(A.2)
$$\boldsymbol{w}_i = \left(\sum_{j \in \Omega_i} C_{ij} \boldsymbol{h}_j \boldsymbol{h}_j^\top + \lambda_i I_k\right)^{-1} \sum_{j \in \Omega_i} C_{ij} A_{ij} \boldsymbol{h}_j,$$

where $I_k \in \mathbb{R}^{k \times k}$ is an identity matrix. Thus the cost of one ALS iteration to update W and H is

(A.3)
$$O(|\Omega| \times k^2 + (m+n)k^3),$$

where $|\Omega| \times k^2$ is for $\sum_{j \in \Omega_i} C_{ij} \boldsymbol{h}_j \boldsymbol{h}_j^\top, \forall i$ and mk^3 is for m matrix inversions. From (A.3), (1.3), and (1.4), for Subsampled and Full approaches the cost is respectively

$$O(|\Omega^+|k^2 + (m+n)k^3) \text{ and } O(mnk^2 + (m+n)k^3).$$

The mn term is huge, but [20] has successfully reduced Full's complexity under the assumption in (3.7).[7] Here we derive there results using our notation. Specifically we rewrite a summation in (A.2) as follows.

$$\sum_{j \in \Omega_i} C_{ij} \boldsymbol{h}_j \boldsymbol{h}_j^\top = \sum_{j \in \Omega_i^+} (1 - p_i q_j) \boldsymbol{h}_j \boldsymbol{h}_j^\top + p_i \sum_{j=1}^n q_j \boldsymbol{h}_j \boldsymbol{h}_j^\top.$$

The crucial point of the above reformulation is that $\sum_{j=1}^n q_j \boldsymbol{h}_j \boldsymbol{h}_j^\top$ can be pre-computed with $O(nk^2)$ operations, so the cost is reduced to be the same as that for the Subsampled approach. The second term in (A.2) involves $O(mnk)$ operations, which can be reduced to $O(|\Omega^+|k)$ because from (3.7),

$$\sum_{j \in \Omega_i} C_{ij} A_{ij} \boldsymbol{h}_j = \sum_{j \in \Omega_i^+} A_{ij} \boldsymbol{h}_j - p_i \bar{a} \sum_{j \in \Omega_i^+} q_j \boldsymbol{h}_j + p_i \bar{a} \sum_{j=1}^n q_j \boldsymbol{h}_j,$$

where $\sum_{j=1}^n q_j \boldsymbol{h}_j$ can be pre-computed. Then the mn term does not appear at all in the calculation of (A.2).

A.2 Stochastic Gradient (SG) We show that the reformulation in (3.20) generates an un-biased gradient estimate to the original optimization problem (2.5). Because of the property that one of the $mn + |\Omega^+|$ terms in (3.20) is uniformly sampled at each time, the expected gradient with respect to $\boldsymbol{w}_i$ is

$$\frac{\left(\sum_{(i,j) \in \Omega^+} \nabla_{\boldsymbol{w}_i} \ell_{ij}^+(A_{ij}, \boldsymbol{w}_i^\top \boldsymbol{h}_j) + \sum_{i=1}^m \sum_{j=1}^n \nabla_{\boldsymbol{w}_i} \ell^-(\bar{a}, \boldsymbol{w}_i^\top \boldsymbol{h}_j)\right)}{mn + |\Omega|},$$

which is proportional to the gradient of the original objective function in (3.19).

B Experiments Details

B.1 Implementation Details We tried the best to have efficient implementations for each optimization method. Here we show some places of applying highly-tuned linear algebra packages. When $q_j = \alpha$, $\forall j$, $\sum_{j=1}^n q_j \boldsymbol{h}_j \boldsymbol{h}_j^\top$ in ALS is essentially the product $\alpha H H^\top$ so we employ fast matrix-matrix operations in optimized BLAS (Basic Linear Algebra Subprograms). We use efficient `posv()` subroutine in the optimized LAPACK (Linear Algebra PACKage) to solve the system of linear equations in (A.2).[8] All three methods in Section 3 can be parallelized, but to focus on algorithmic differences, we use single-core implementations in our experiments.

[7] [20] considers a more general setting so that C can be a low-rank matrix but in most cases the rank-one situation in (3.7) is considered.

[8] In all timing experiments, ATLAS [28] is used as the optimized BLAS/LAPACK.

B.1.1 Subsampled and Ensemble: Sampling Schemes Let $\texttt{multinomial}(\{p_i\})$ be the distribution such that the i-th user index is selected with probability p_i and $\texttt{multinomial}(\{q_j\})$ be the distribution such that the j-th item index is selected with probability q_j. Most sampling schemes considered for the Subsampled approach in the past [21, 22] can be described by the following procedure:

- Sample $i' \sim \texttt{multinomial}(\{p_i\})$
- Sample $j' \sim \texttt{multinomial}(\{q_j\})$
- Add (i', j') into Ω^-

Following the discussion in Section 2.2, we give the detailed specification for the five Subsampled variants and the Ensemble approach compared in the experiments:

Sampling scheme	$\{p_i\}$	$\{q_j\}$
user	$p_i \propto \lvert\Omega_i^+\rvert$	$q_j = 1/n$
item-f	$p_i = 1/m$	$q_j \propto \lvert\bar{\Omega}_j^+\rvert$
item-w	$p_i = 1/m$	$q_j \propto m - \lvert\bar{\Omega}_j^+\rvert$
item-s	$p_i = 1/m$	$q_j \propto 1/\lvert\bar{\Omega}_j^+\rvert$
uniform	$p_i = 1/m$	$q_j = 1/n$

For the Ensemble approach [21], 20 Subsampled models with the uniform sampling scheme are aggregated to generate the ensemble model as follows:

$$A^{\text{Ensemble}} = \frac{1}{20}\sum_{s=1}^{20} W_s H_s^\top,$$

where (W_s, H_s) is the s-th Subsampled model. The result for Ensemble is derived from the ranking induced by A^{Ensemble}.

B.1.2 Some Details of BPR BPR is an approach for one-class collaborative filtering by solving the following optimization problem:

$$\min_{W,H} \sum_{i=1}^{m} \sum_{(j_+,j_-)\in\Omega_i^+\times\Omega_i^-} \Big\{ \log\big(1+\exp\big(-\boldsymbol{w}_i^\top(\boldsymbol{h}_{j_+} - \boldsymbol{h}_{j_-})\big)\big) + \frac{\lambda}{2}\big(\|\boldsymbol{w}_i\|^2 + \|\boldsymbol{h}_{j_+}\|^2 + \|\boldsymbol{h}_{j_-}\|^2\big)\Big\}. \tag{B.4}$$

In (B.4), all the pairs between observed items and unknown items are included into the formulation by a logistic loss. It can be seen as an approach to minimize the empirical AUC [25]. As the number of pairs is very large, [25] proposes to apply SG to solve (B.4). At each SG step for BPR, a pair is selected randomly for the update. In our experiments, we use the BPR implementation available in LIBMF [6].

B.2 Evaluation Criteria For the i-th user, let $\Omega_{\text{test}}(i)$ be the set of candidate items whose preference is predicted and evaluated in the testing phase. We exclude Ω_i^+ because the model has been trained to fit data in Ω^+. Thus $\Omega_{\text{test}}(i) = [n]\backslash\Omega_i^+$. We further denote $\Omega_{\text{test}}^+(i) \subset \Omega_{\text{test}}(i)$ as the subset of items which receive a positive response from the i-th user. Let

$$\pi_i : \Omega_{\text{test}}(i) \to \{1, \ldots, |\Omega_{\text{test}}(i)|\}$$

be the ranking predicted by the recommender system (i.e., $\pi_i(j)$ = rank of the j-th item) and $\pi_i^{-1}(r)$ be the item with rank r; that is, $\pi_i(\pi_i^{-1}) = r$. We consider the following evaluation criteria.

- nDCG@p: normalized discounted cumulated gain. The following criterion is widely used to evaluate the performance of the top p-ranked items.

$$\texttt{DCG}_i@p(\pi_i) = \sum_{r=1}^{p} \frac{[\pi_i^{-1}(r) \in \Omega_{\text{test}}^+(i)]}{\log_2(1+r)}.$$

  This value might be in a different range depending on p and $|\Omega_{\text{test}}^+(i)|$, so people usually consider the normalized DCG as follows:

$$\texttt{nDCG}_i@p = 100 \times \frac{\texttt{DCG}_i@p(\pi_i)}{\texttt{DCG}_i@p^{\max}},$$

  where $\texttt{DCG}_i@p^{\max} = \max_\pi \texttt{DCG}_i@p(\pi)$. In this paper, we report the average nDCG@p, i.e., $\sum_i \texttt{nDCG}_i@p/m$, among all users.

- nHLU: normalized half life utility. HLU was first proposed in [3, Eq. 5] to evaluate the performance of recommender systems. For one-class recommender systems (i.e., the ground truth rating is either 1 or 0), the HLU for a user, $\texttt{HLU}_i(\pi_i)$, can be defined as follows

$$\texttt{HLU}_i(\pi_i) = \sum_{j\in\Omega_{\text{test}}^+(i)} \frac{1}{2^{(\pi_i(j)-1)/(\beta-1)}},$$

  where β is the "half life" parameter denoting the rank of the item on the list such that there is a 50-50 chance the i-th user will review that item.[9] Following the usage in [3, 20, 21], $\beta = 5$ is used in our experiments. Similar to nDCG, the normalized half life utility can be defined as follows:

$$\texttt{nHLU}_i = 100 \times \frac{\texttt{HLU}_i(\pi_i)}{\texttt{HLU}_i^{\max}},$$

  where $\texttt{HLU}_i^{\max} \equiv \max_\pi \texttt{HLU}_i(\pi)$. In this paper, we report the average nHLU, $\sum_i \texttt{nHLU}_i/m$, among all users.[10]

[9] Note that there is a typo in the description of HLU in [20, 21], where "/" is missing.

[10] This is slightly different from the original formulation in [3], where $100 \times \frac{\sum_i \texttt{HLU}_i(\pi_i)}{\sum_i \texttt{HLU}_i^{\max}}$ is reported.

- MAP: mean average precision. For the i-th user, the average precision $\texttt{AP}_i$ is defined as follows:

$$\texttt{AP}_i = \frac{\sum_{j \in \Omega^+_{\text{test}}(i)} 100 \times \frac{|\{j' \in \Omega^+_{\text{test}}(i) : \pi_i(j') \le \pi_i(j)\}|}{\pi_i(j)}}{|\Omega^+_{\text{test}}(i)|}.$$

  In this paper, we report MAP, which is $\sum_i \texttt{AP}_i/m$.
- AUC: area under the ROC curve. For the i-th user, this is equivalent to the ratio of violating pairs among all the pairs from $\Omega^+_{\text{test}}(i) \times \Omega^-_{\text{test}}(i)$, and can be computed as follows [25]:

$$\texttt{AUC}_i = \frac{|\{(j,j') \in \Omega^+_{\text{test}}(i) \times \Omega^-_{\text{test}}(i) : \pi_i(j) > \pi_i(j')\}|}{|\Omega^+_{\text{test}}(i)| \times |\Omega^-_{\text{test}}(i)|}.$$

  In this paper, we report the average AUC, $\sum_i \texttt{AUC}_i/m$, among all users.

Among these four evaluation criteria, nDCG and nHLU are more appropriate to evaluate the ranking performance of recommendation systems, as they put more weights on the top-ranked items. On the other hand, AUC is less appropriate because of its insensitivity to the position of the violating pairs.

B.3 Analysis of Sampling Schemes for the Sub-sampled Approach The numbers reported for the Sub-sampled approach in Table 4 are the best results from all the sampling schemes. Table 6 shows the detailed comparison among various sub-sampling schemes. We can make the following observations:

- The "uniform" scheme performs the best in general, and the "user" scheme is also competitive among all sampling schemes. This result is slightly different from the conclusion in [21], where the "user" scheme is slightly better than the "uniform" scheme.
- The "item-f" scheme is considered as a baseline in [22]. However, this scheme gives the worst performance among all schemes considered here.

C More Discussions

C.1 Performance versus Various Values of α To better understand the influence of the α parameter in (4.21), we generate a synthetic ground truth matrix A as follows:

- $W \leftarrow \texttt{rand}(m,k)$ and $H \leftarrow \texttt{rand}(n,k)$
- $\bar{A} \leftarrow WH^\top$
- $A_{ij} \leftarrow \begin{cases} 1 & \text{if } \bar{A}_{ij} \ge \bar{a}, \\ 0 & \text{otherwise,} \end{cases}$ where $\bar{a}$ is the value such that

$$|\{\bar{A}_{ij} \ge \bar{a}\}| = 0.2 \times mn.$$

As mentioned earlier, theoretical results in [10] suggest that the best α is a function of $\bar{\rho}$, which is the ratio of the observed positive entries over the entire positive entries. For a given $\bar{\rho}$, we construct a training set $\Omega^+(\bar{\rho})$ by sampling positive entries from A such that

$$\frac{|\Omega^+(\bar{\rho})|}{|\{(i,j) : A_{ij} = 1\}|} = \bar{\rho}.$$

As the ground truth is available in the synthetic data set, we report the ranking performance over the entire item set. That is, $\Omega_{\text{test}}(i) = \{1,\dots,n\}, \forall i$ because we do not exclude Ω^+_i.

Figures 2–3 show the results on synthetic matrices A of various sizes ($m = n = 500$, $m = n = 1,000$, and $m = n = 5,000$) with $k = 10$. The x-axis denotes the value of α used in the optimization problem, and we report results for $\alpha \in \{2^{-20}, 2^{-18}, \dots, 2^8\}$. The y-axis denotes the performance for each α with the best λ selected from $\{10^{-12}, \dots, 10^{-1}\}$. Each curve in Figures 2–3 corresponds to a pair of training and test sets generated under a specific $\bar{\rho}$. Four values of $\bar{\rho}$ are considered: 0.95, 0.1, 0.05, and 0.01. We make the following observations:

- The shape of curves for $\bar{\rho} = 0.1$ and $\bar{\rho} = 0.05$ is concave, which means that the best α cannot be too large or too small. The result reconfirms the discussion in Section 5. Therefore, suitable validation procedures are required to get the best performance.
- The theoretical results by [10] suggest that with an appropriate choice of λ, thresholding parameter $\bar{a}$ and

$$\alpha = \frac{\bar{\rho}}{2 - \bar{\rho}},$$

  the Full approach can recover the ground truth binary matrix A from $WH^\top$ obtained by (1.4) with high probability (i.e., high point-wise accuracy). This suggests that if $\bar{\rho}$ is close to 1, the best α should also close to 1; on the other hand, if $\bar{\rho}$ is close to 0, the best α should also close to 0. Although the evaluation criteria considered here is not the point-wise accuracy considered in [10], we can still observe a similar trend on these ranking-based criteria.

C.2 Regularization for One-Class MF In this section, we explain what we claimed earlier in Section 3.2 that the second term of (3.20), $\sum_i \sum_j \ell^-_{ij}$, can be considered as a regularization function for one-class MF. Based on the definition on ℓ^-_{ij}, it is not hard to see

$$\sum_{i=1}^m \sum_{j=1}^n \ell^-_{ij}(\bar{a}, \boldsymbol{w}_i^\top \boldsymbol{h}_j) = \|C_w(WH^\top - \bar{a}E)C_h\|_F^2 + \lambda\|D_w W\|_F^2 + \lambda\|D_h H\|_F^2, \tag{C.5}$$

where E is the all one matrix, $C_w = \sqrt{\texttt{diag}(\boldsymbol{p})}$ and $C_h = \sqrt{\texttt{diag}(\boldsymbol{q})}$ are diagonal matrices with

$$(C_w)_{ii} = \sqrt{p_i},\ \forall i, \quad \text{and} \quad (C_h)_{jj} = \sqrt{q_j},\ \forall j$$

Table 5: Range of parameters considered for each one-class MF approach.

Parameter	Subsampled	Ensemble	Full	BPR
λ	$\{10^{-4}, ..., 10^{-1}\}$	$\{10^{-4}, ..., 10^{-1}\}$	$\{10^{-4}, ..., 10^{-1}\}$	$\{10^{-8}, ..., 10^{-1}\}$
k	$\{2^4, 2^5, 2^6\}$	$\{2^4, 2^5, 2^6\}$	$\{2^4, 2^5, 2^6\}$	2^6
$C_{ij} = \alpha, \forall (i,j) \notin \Omega^+$	1	1	$\{2^{-5}, 2^{-3}, 2^{-1}, 2^0\}$	1
$\lvert\Omega^-\rvert / \lvert\Omega^+\rvert$	$\{1, 2\}$	$\{1, 2\}$	-	-
sampling schemes	user/item-f/item-w/item-s uniform/uniform-ens	user/item-f/item-w/item-s uniform/uniform-ens	-	-

Table 6: Comparison among various sampling schemes for the Subsampled approach. See detailed descriptions for each sampling scheme in Section B.1.1.

(a) delicious

Scheme	nDCG @1	nDCG @5	nDCG @10	nHLU	MAP	AUC
user	41.40	32.62	27.74	26.58	14.37	0.85985
item-f	1.55	1.84	1.89	1.98	2.04	0.69648
item-w	39.15	31.21	26.63	25.65	14.15	0.86687
item-s	29.65	24.05	21.92	21.35	12.26	0.84855
uniform	38.40	30.94	26.05	25.29	13.94	0.86578

(b) ml1m

Scheme	nDCG @1	nDCG @5	nDCG @10	nHLU	MAP	AUC
user	13.16	11.54	12.23	12.87	9.14	0.93290
item-f	0.78	0.94	1.17	1.33	1.47	0.78903
item-w	13.56	13.67	14.74	15.67	11.12	0.93723
item-s	7.88	7.08	8.06	8.53	5.80	0.90470
uniform	14.69	13.89	14.94	15.79	11.17	0.93743

(c) ml10m

Scheme	nDCG @1	nDCG @5	nDCG @10	nHLU	MAP	AUC
user	10.55	9.55	10.17	10.87	8.17	0.97250
item-f	0.91	1.36	1.93	2.27	1.95	0.85310
item-w	9.47	9.99	11.96	13.18	9.92	0.97304
item-s	2.84	3.29	4.21	4.81	4.35	0.95693
uniform	9.33	10.07	12.10	13.31	10.00	0.97293

(d) netflix

Scheme	nDCG @1	nDCG @5	nDCG @10	nHLU	MAP	AUC
user	10.26	9.61	9.96	10.48	7.54	0.97192
item-f	1.26	1.74	2.07	2.27	1.66	0.85662
item-w	10.43	10.02	11.01	11.77	8.75	0.97212
item-s	4.45	4.70	5.37	5.80	4.69	0.95741
uniform	10.62	10.22	11.27	12.03	8.91	0.97224

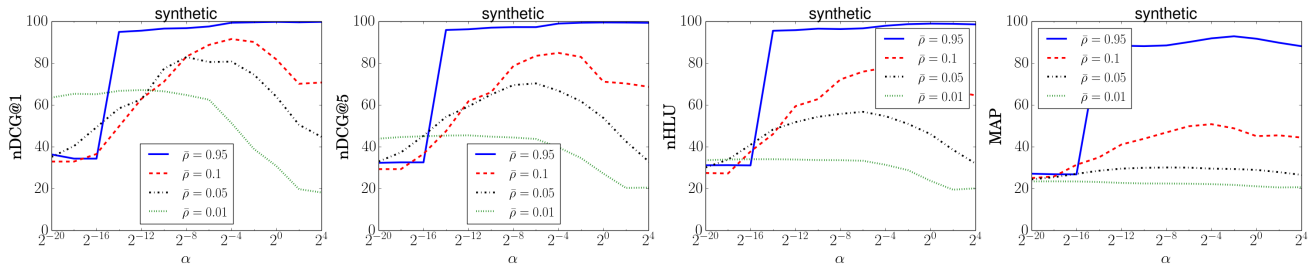


Figure 2: Performance versus various values of α on synthetic data sets with $m = n = 500$.

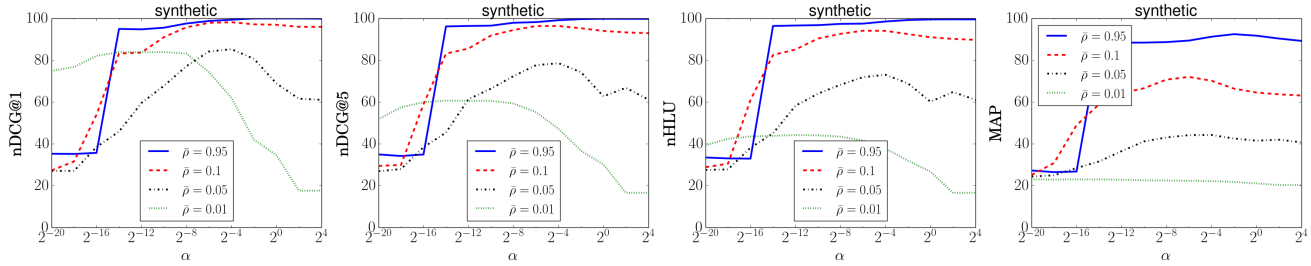


Figure 3: Performance versus various values of α on synthetic data sets with $m = n = 1,000$.

and D_w and D_h are diagonal matrices[11] with

$$(D_w)_{ii} = \sqrt{n|\Omega_i^+|/\big(n + |\Omega_i^+|\big)}, \ \forall i$$
$$(D_h)_{jj} = \sqrt{m|\bar{\Omega}_j^+|/\big(m + |\bar{\Omega}_j^+|\big)}, \ \forall j.$$

As we can see, (C.5), derived from the summation of ℓ^- terms, can be regarded as a special regularization function on the model W and H which encourages the structure that all the missing entries are close to $\bar{a}$. From this point of view, the Full approach can be considered as an empirical risk minimization problem in the following form:

$$\sum_{(i,j)\in\Omega^+} \ell^+(A_{ij}, \boldsymbol{w}_i^\top \boldsymbol{h}_h) + \mathcal{R}(W, H),$$

where the loss function ℓ^+ only applies to the explicitly observed entries, while the implicit responses are captured by the regularization $\mathcal{R}(W, H)$. Interestingly, this interpretation to incorporate implicit responses into regularization also fits in a recent proposed implicit matrix factorization approach [9], where the likelihood of A for a given model (W, H) is

$$\log \mathbb{P}[\![A \mid W, H]\!] = \left(\sum_{(i,j):A_{ij}>0} A_{ij} \log(\boldsymbol{w}_i^\top \boldsymbol{h}_j) - A_{ij}! \right) - \boldsymbol{e}^\top W H^\top \boldsymbol{e}, \tag{C.6}$$

where $A_{ij}!$ is the factorial of the integer rating A_{ij}, and the corresponding MLE problem is

$$\min_{W,H} \sum_{(i,j):A_{ij}>0} \underbrace{-A_{ij} \log(\boldsymbol{w}_i^\top \boldsymbol{h}_j)}_{\ell_{ij}^+} + \underbrace{\boldsymbol{e}^\top W H^\top \boldsymbol{e}}_{\mathcal{R}(W,H)}. \tag{C.7}$$

We can clearly see that the first term of (C.7) corresponds to the Poisson loss for the explicitly observed positive ratings, while the second term can be regarded a regularizer accounting for the implicit responses.

C.3 Computational Issue in Prediction With the proposed efficient algorithms in Section 3, the time complexity of the training procedure for one-class MF is only linear in $|\Omega^+|$ for both Full and Subsampled approaches.

Test time is an important concern for one-class MF because for each user i, we must calculate $\boldsymbol{w}_i^\top \boldsymbol{h}_j, \forall j$ to find the top items. However, item prediction remains a computational challenge. Given a recommender model (W, H) with m users and n items, item prediction for the i-th user requires the ranking or the maximum over $n - \left|\Omega_i^+\right|$ inner products (i.e, $\boldsymbol{w}_i^\top \boldsymbol{h}_j, \ j \in [n] \setminus \Omega_i^+$),

[11] If all the regularization are distributed to ℓ_{ij}^- terms in the reformulation (3.20), $(D_w)_{ii} = \sqrt{\lambda/\lambda_i}$ and $(D_h)_{jj} = \sqrt{\lambda/\bar{\lambda}_j}$.

which is close to $O(nk)$ as $\left|\Omega_i^+\right|$ is usually small. Thus, item prediction for all the m users requires $O(mnk)$ operations, which can be orders of magnitude more than the training phase. As a result, the Ensemble approach particularly suffers from lengthy test time because the use of 20 models causes a 20-fold increase of the prediction cost. For example, the test time for yahoo-music is about 70 times more than the time required for netflix, so we omit the experiments for Subsampled/Ensemble in Table 4. There have been a few recent works about how to reduce the time complexity of searching the maximum of inner products [1, 2, 19, 26].