A Unified Algorithm for One-class Structured Matrix Factorization with Side Information

Hsiang-Fu Yu
University of Texas at Austin
rofuyu@cs.utexas.edu

Hsin-Yuan Huang
National Taiwan University
momohuang@gmail.com

Inderjit S. Dhillon
University of Texas at Austin
inderjit@cs.utexas.edu

Chih-Jen Lin
National Taiwan University
cjlin@csie.ntu.edu.tw

Abstract

In many applications such as recommender systems and multi-label learning the task is to complete a partially observed binary matrix. Such PU learning (positive-unlabeled) problems can be solved by one-class matrix factorization (MF). In practice side information such as user or item features in recommender systems are often available besides the observed positive user-item connections. In this work we consider a generalization of one-class MF so that two types of side information are incorporated and a general convex loss function can be used. The resulting optimization problem is very challenging, but we derive an efficient and effective alternating minimization procedure. Experiments on large-scale multi-label learning and one-class recommender systems demonstrate the effectiveness of our proposed approach.

1 Introduction

Many practical applications can be modeled in a positive and unlabeled data learning (PU learning) framework. Between two types of objects, some connections between object i of the first type and j of the second type are observed, but for most remaining situations, either i and j are not connected or the connection is not observed. A typical example is the collaborative filtering with one-class information (Pan et al. 2008; Hu, Koren, and Volinsky 2008; Pan and Scholz 2009; Li et al. 2010; Paquet and Koenigstein 2013). Given m users and n items, we observe part of a $0/1$ matrix $Y \in \mathbb{R}^{m\times n}$ with $Y_{ij} = 1$, where $(i, j) \in \Omega^+$. An observed entry indicates that user i likes item j. The goal is to know for any unobserved pair $(i, j) \notin \Omega^+$, whether i likes j or not. Thus, only part of positive-labeled entries are observed, while there are many unknown negative entries. This setting is very different from traditional collaborative filtering, where a real-valued rating (observed or not) is associated with any Y_{ij}. Note that this may be a more common scenario; for example, most people watch the video that interest them, without leaving any rating information.

Another important application that falls into the PU leaning framework is multi-label classification (Kong et al. 2014; Hsieh, Natarajan, and Dhillon 2015). The two types of objects are instances and labels. An entry $Y_{ij} = 1$ indicates that instance i is associated with label j. In practice, only part of instance i's labels have been observed.

PU learning is essentially a matrix completion problem. Given $Y_{ij} = 1,\ \forall (i, j) \in \Omega^+$, we would like to predict if other entries are zero or one. One common approach for matrix completion is through matrix factorization (MF) by finding two low-rank latent matrices

$$W = [\ldots, \boldsymbol{w}_i, \ldots]^\top \in \mathbb{R}^{m\times k} \text{ and } H = [\ldots, \boldsymbol{h}_j, \ldots]^\top \in \mathbb{R}^{n\times k}$$

such that $Y_{ij} = 1 \approx \boldsymbol{w}_i^\top \boldsymbol{h}_j,\ \forall (i, j) \in \Omega^+$. Note that k is a pre-specified number satisfying $k \ll m$ and $k \ll n$. Unlike traditional matrix completion problems where $Y_{ij},\ \forall (i, j) \in \Omega^+$ have different values, here approximating all the observed positive entries will result in the all-one prediction for all $(i, j) \notin \Omega^+$. For such a one-class scenario, in the approximation process one must treat some $(i, j) \notin \Omega^+$ as negative entries so that $Y_{ij} = 0 \approx \boldsymbol{w}_i^\top \boldsymbol{h}_j$. Therefore, existing one-class MF works solve the following optimization problem.

$$\min_{W,H} \sum_{i,j\in\Omega} C_{ij}(Y_{ij} - \boldsymbol{w}_i^T \boldsymbol{h}_j)^2 + \sum_i \lambda_i \|\boldsymbol{w}_i\|^2 + \sum_j \bar{\lambda}_j \|\boldsymbol{h}_j\|^2, \quad (1)$$

where C_{ij} is a cost associated with the loss, and $\lambda_i, \bar{\lambda}_j$ are regularization parameters. The set $\Omega = \Omega^+ \cup \Omega^-$ includes both observed positive and selected negative entries. The rationale of selecting some $(i, j) \notin \Omega^+$ and treating their $Y_{ij} = 0$ is because in general each user likes only a small set of items (or each instance is associated with a small set of labels).

The selection of negative entries in Ω^- is an important issue. Roughly there are two approaches:

- Subsampled: we subsample some unobserved entries to have Ω^- with $|\Omega^-| = O(|\Omega^+|)$.
- Full: $\Omega^- = [m]\times[n]\backslash\Omega^+$. That is, every unobserved entry is considered as negative. Unfortunately, the huge size of Ω^- makes this approach *computationally expensive.*

Recently, Yu, Bilenko, and Lin (2016) successfully developed efficient optimization methods to solve (1) for the Full approach. They show that the Full approach gives *significantly* better results than the Subsampled approach. A shortcoming is that their methods work only with the squared loss $(Y_{ij} - \boldsymbol{w}_i^\top \boldsymbol{h}_j)^2$, which is a real-value regression loss. However Y is a 0/1 matrix in the one-class setting, so a 0/1 classification loss might be more suitable. Yet,

Table 1: Various one-class MF formulations supported by our proposed algorithms. SQ-SQ: we apply the square loss on entries in both Ω^+ and Ω^-. SQ-wSQ: we apply the square loss on entries in Ω^+ and the weighted square loss on Ω^-. General-wSQ: we apply a general loss on entries in Ω^+ and the weighted square loss on Ω^-. For the Full approach of most formulations in this family, our proposed algorithms are the first efficient approach with time complexity linear in $O(|\Omega^+|)$. [1]: (Pan and Scholz 2009), [2]: (Yu, Bilenko, and Lin 2016), [3]: (Yu et al. 2014), and [4]: (Rao et al. 2015).

	SQ-SQ	SQ-wSQ	General-wSQ
loss ℓ^+_{ij} on Ω^+	square	square	general loss
loss ℓ^-_{ij} on Ω^-	square	weighted square	weighted square
Standard	[1],[2]	[1],[2]	*this paper*
Feature-aware	LEML [3]	*this paper*	*this paper*
Graph-structured	GRALS [4]	*this paper*	*this paper*
Feature+Graph	*this paper*	*this paper*	*this paper*

developing efficient methods for the Full approach with a classification loss remains a challenging problem.

In problem (1), $Y_{ij}, (i,j) \in \Omega^+$ are the only given information. However, for most applications, some "side information" is also available. For example, besides the preference of user i on item j, user or item features may also be known. For multi-label learning, a data instance always comes with a feature vector. Further, relationships among users (items) may be available, which can be represented as a graph. How to effectively incorporate side information into the PU learning framework is thus a crucial research issue.

In this paper, we consider a formulation which unifies and generalizes many existing structured matrix-factorization formulations for PU learning with side information. In Section 2, we introduce the formulation and review related works. Our main contribution in Section 3 is to develop an efficient alternating minimization framework for the Full approach with *any convex loss* function. Experiments in Section 4 consider multi-label classification and recommender systems to illustrate the effectiveness of our approach. Results show a clear performance improvement using a classification loss. A summary showing how we generalize existing works is in Table 1, indicating that our proposed algorithms enable the computational feasibility of the Full approach for many one-class MF formulations with or without side information. The experimental codes and supplementary materials can be found at `http://www.csie.ntu.edu.tw/~cjlin/papers/ocmf-side/`.

2 One-class Structured Matrix Factorization

Assume that two types of side information are available for users (or instances). First, each user is associated with a feature vector, and second, a similarity matrix indicating the relationships between users is available. Our discussion can be easily generalized to the situation that items or both users and items come with side information. The proposed extension of (1) is the following optimization problem.

$$\min_{W,H} \quad f(W,H), \text{ where } f(W,H) = \tag{2}$$

$$\sum\nolimits_{(i,j)\in\Omega} C_{ij}\ell(Y_{ij}, \boldsymbol{x}_i^\top W \boldsymbol{h}_j) + \lambda_w \mathcal{R}(W) + \lambda_h \mathcal{R}(H).$$

In (2), λ_w, λ_h and λ_g are regularization parameters;

$$\mathcal{R}(W) = \operatorname{tr}\big(W^\top W + \lambda_g W^\top X^\top L X W\big), \mathcal{R}(H) = \operatorname{tr}\big(H^\top H\big)$$

are regularizers[1]; L is a positive definite matrix;

$$X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m]^\top \in \mathbb{R}^{m\times d}$$

includes feature vectors corresponding to users; $\ell(a,b)$ is a loss function convex in b. We also use

$$\ell_{ij}(a,b) = C_{ij}\ell(a,b)$$

to denote the loss term for the (i,j) entry. Typically L is in a form of a graph Laplacian matrix with $L = D - S$, where D is a diagonal matrix with $D_{ii} = \sum_{t=1}^m S_{it}$ and $S \in \mathbb{R}^{m\times m}$ is a similarity matrix among users. Then in $\mathcal{R}(W)$ we have

$$\operatorname{tr}\big(W^\top X^\top L X W\big) = \frac{1}{2}\sum\nolimits_{i_1,i_2} S_{i_1,i_2}\big\|W^\top \boldsymbol{x}_{i_1} - W^\top \boldsymbol{x}_{i_2}\big\|^2.$$

If the relationship between users i_1 and i_2 is strong, the larger S_{i_1,i_2} will make $W^\top \boldsymbol{x}_{i_1}$ closer to $W^\top \boldsymbol{x}_{i_2}$. This use of the graph information in the regularization term has been considered in various learning methods (Smola and Kondor 2003; Li and Yeung 2009; Zhou et al. 2012; Zhao et al. 2015; Rao et al. 2015; Natarajan, Rao, and Dhillon 2015), where Natarajan, Rao, and Dhillon (2015) focus on PU learning. We further note that in the optimization problem (2), $W^\top \boldsymbol{x}_i$ becomes the latent representation of the i-th user (or instance). Thus in contrast to $W \in \mathbb{R}^{m\times k}$ in the standard MF formulation, now we have $W \in \mathbb{R}^{d\times k}$. Then the prediction on unseen instances, such as in multi-label classification, can be done naturally using $W^\top \boldsymbol{x}$ as feature vector $\boldsymbol{x}$'s latent representation. There are other approaches to incorporate side information into MF such as Singh and Gordon (2008), but we focus on the formulation (2) in this paper.

Some past works have considered optimization problems related to (2). In (Rao et al. 2015), for rating-based MF, they consider the same regularization term $\mathcal{R}(W)$ by assuming that pairwise relationships are available via a graph. For squared losses they derive efficient alternating least squares methods to solve the optimization problem. Ours differs from them on: (i) they do not incorporate the side information of feature vectors in X; (ii) we consider one-class rather than rating-based MF; (iii) we consider general losses. Natarajan, Rao, and Dhillon (2015) consider the graph structured one-class MF without feature vectors X and proposes a Frank-Wolf algorithm, which requires $O(mn)$ space and the full eigendecomposition of the graph Laplacian matrix. For multi-label learning, Yu et al. (2014) solve (2) without the graph information for two scenarios:

- the general loss with $C_{ij} = 1$ and $|\Omega| \ll mn$; and
- the squared loss with $C_{ij} = 1$ and $\Omega = [m] \times [n]$.

[1] $\operatorname{tr}(\cdot)$ denotes the trace of a matrix

Theirs does not cover the situation of using a general loss with weighted C_{ij} and $\Omega = [m] \times [n]$.

If side information is not considered, Yu, Bilenko, and Lin (2016) give a detailed study on one-class MF under the Full setting. They consider the squared loss with

$$C_{ij} = \begin{cases} 1 & \text{if } (i,j) \in \Omega^+, \\ p_i q_j & \text{otherwise,} \end{cases} \text{ and } Y_{ij} = \bar{a},\ \forall (i,j) \notin \Omega^+, \quad (3)$$

where $\boldsymbol{p} \in \mathbb{R}^m$ and $\boldsymbol{q} \in \mathbb{R}^n$ are vectors chosen so that $p_i q_j < 1$ because weights for observed entries should be higher and $\bar{a}$ is a real number. A common choice is $\bar{a} = 0$ or -1. This work proposes efficient algorithms, and demonstrates that using $\Omega^- = [m] \times [n] \setminus \Omega^+$ gives much better results than selecting a small set Ω^- with $|\Omega^-| = O(|\Omega^+|)$.

3 Algorithms for One-class Structured MF

We develop efficient optimization algorithms to minimize (2) by considering C_{ij} in (3), *any convex loss*, and $\Omega = [m] \times [n]$ (i.e., the Full approach). We consider an alternating minimization framework to iteratively update

$$W \leftarrow \arg\min_W f(W,H) \text{ and } H \leftarrow \arg\min_H f(W,H),$$

and show that each sub-problem can be efficiently solved. Because $\min_H f(W,H)$ can be treated as a special form of

$$\min_W f(W,H) \text{ with } \lambda_g = 0 \text{ and } X = I,$$

where I is the identity matrix, we focus on the sub-problem when H is fixed. Let

$$\tilde{\boldsymbol{x}}_{ij} = \boldsymbol{h}_j \otimes \boldsymbol{x}_i = \operatorname{vec}\big(\boldsymbol{x}_i \boldsymbol{h}_j^\top\big) \in \mathbb{R}^{dk},\ \tilde{\boldsymbol{w}} = \operatorname{vec}(W) \in \mathbb{R}^{dk},$$

where $\operatorname{vec}(\cdot)$ is the vector by stacking the columns of a matrix. From

$$\boldsymbol{x}_i^\top W \boldsymbol{h}_j = \big(\boldsymbol{h}_j^\top \otimes \boldsymbol{x}_i^\top\big) \operatorname{vec}(W) = \tilde{\boldsymbol{x}}_{ij}^\top \tilde{\boldsymbol{w}}, \quad (4)$$

$\min_W\ f(W,H)$ can be reformulated as follows:

$$\min_{\tilde{\boldsymbol{w}}}\ g(\tilde{\boldsymbol{w}}), \text{ where} \quad (5)$$

$$g(\tilde{\boldsymbol{w}}) = \sum\nolimits_{(i,j)\in\Omega} \ell_{ij}\big(Y_{ij}, \tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{x}}_{ij}\big) + \lambda_w R_{\tilde{w}}(\tilde{\boldsymbol{w}}), \text{and}$$

$$R_{\tilde{w}}(\tilde{\boldsymbol{w}}) = \tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{w}} + \lambda_g \big((X^\top L X) \otimes I_k\big) \tilde{\boldsymbol{w}}.$$

Because $|\Omega| = mn$ and the number of variables dk can be large, and closed-form solutions may not be available under some loss functions, we may apply iterative optimization methods to solve (5), where the calculations of the function, the gradient, and Hessian-vector product are often needed.

Handling all mn loss values is difficult, so we make the sub-problem easier by restricting the loss function on $(i,j) \in \Omega^-$ to be quadratic. Therefore, with C_{ij} in (3), we have the following two modified loss functions.

$$\ell^+_{ij}(Y_{ij}, \tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{x}}_{ij}) = \ell_{ij}(Y_{ij}, \tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{x}}_{ij}) - p_i q_j (\bar{a} - \tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{x}}_{ij})^2$$

$$\ell^-_{ij}(\bar{a}, \tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{x}}_{ij}) = p_i q_j (\bar{a} - \tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{x}}_{ij})^2. \quad (6)$$

Then, (5) is equivalent to (7).

$$g(\tilde{\boldsymbol{w}}) = \lambda_w \mathcal{R}_{\tilde{w}}(\tilde{\boldsymbol{w}}) + \quad (7)$$

$$\underbrace{\sum_{(i,j)\in\Omega^+} \ell^+_{ij}(Y_{ij}, \tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{x}}_{ij})}_{L^+(\tilde{\boldsymbol{w}})} + \underbrace{\sum_{(i,j)\in[m]\times[n]} \ell^-_{ij}(\bar{a}, \tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{x}}_{ij})}_{L^-(\tilde{\boldsymbol{w}})},$$

where $L^+(\tilde{\boldsymbol{w}})$ is the term corresponding to all the entries in Ω^+, and $L^-(\tilde{\boldsymbol{w}})$ is the term considering all mn entries.

For the loss function, we do not require them to be differentiable because some optimization methods need just function evaluations. However, if they are, we let

$$\ell'_{ij}(a,b) = \frac{\partial}{\partial b} \ell_{ij}(a,b), \text{ and } \ell''_{ij}(a,b) = \frac{\partial^2}{\partial b^2} \ell_{ij}(a,b),$$

and assume that once a and b are given, these values can be calculated in a constant time.

With the reformulation (7), we can derive efficient procedures to perform function, gradient evaluation and Hessian-vector products. In particular, the $O(mn)$ terms in $L^-(\tilde{\boldsymbol{w}})$ can be efficiently computed with the cost comparable to that for $O(|\Omega^+|)$ terms in $L^+(\tilde{\boldsymbol{w}})$. While Yu et al. (2014) and Yu, Bilenko, and Lin (2016) also achieve such results, their optimization problems are our special cases. The generalization is not trivial. For example, Yu et al. (2014) state that for general losses, the $O(mn)$ term in the overall complexity is inevitable, but here we successfully remove it.

We show details for the function evaluation, but leave most details of gradient evaluation and Hessian-vector products in the supplementary materials. To cover both dense and sparse feature/graph matrices in our complexity analysis we use $\operatorname{nnz}(\cdot)$ to denote the number of nonzeros in a matrix.

3.1 Evaluation of the Objective Value $g(\tilde{\boldsymbol{w}})$

To compute $L^+(\tilde{\boldsymbol{w}})$ in (7), we first compute $B = XW$, where $B = [\ldots, \boldsymbol{b}_i, \ldots]^\top$ in $O(\operatorname{nnz}(X)k)$ time and store it in an $O(mk)$ space. Then for each $(i,j) \in \Omega^+$, from (4), $\tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{x}}_{ij}$ is a simple inner product $\boldsymbol{b}_i^\top \boldsymbol{h}_j$ that costs only $O(k)$ time. As a result, we can compute the entire $L^+(\tilde{\boldsymbol{w}})$ in $O(\operatorname{nnz}(X)k + |\Omega^+|k)$ time and $O(mk)$ space.

To compute $L^-(\tilde{\boldsymbol{w}})$, we first notice its independence of any Y_{ij} and consider the following equivalent formulation (see a detailed derivation in the supplementary materials).

$$L^-(\operatorname{vec}(W)) = \operatorname{tr}\big(\bar{a}^2 \mathbf{1}_n \mathbf{1}_m^\top \operatorname{diag}(\boldsymbol{p}) \mathbf{1}_m \mathbf{1}_n^\top \operatorname{diag}(\boldsymbol{q})\big) \quad (8)$$

$$+ \operatorname{tr}\big(H W^\top X^\top \operatorname{diag}(\boldsymbol{p}) X W H^\top \operatorname{diag}(\boldsymbol{q})\big) \quad (9)$$

$$- 2 \operatorname{tr}\big(\bar{a} \mathbf{1}_n \mathbf{1}_m^\top \operatorname{diag}(\boldsymbol{p}) X W H^\top \operatorname{diag}(\boldsymbol{q})\big), \quad (10)$$

where $\mathbf{1}_m \in \mathbb{R}^m$ and $\mathbf{1}_n \in \mathbb{R}^n$ are vectors of ones, respectively. Note that with the availability of

- $B = XW$, obtained from the calculation of $L^+(\tilde{\boldsymbol{w}})$,
- $M = H^\top \operatorname{diag}(\boldsymbol{q}) H$, which can be pre-computed in $O(nk^2)$ time and stored in $O(k^2)$ space,
- $\boldsymbol{d} = X^\top \boldsymbol{p}$, which can be pre-computed in $O(\operatorname{nnz}(X))$ time and stored in $O(d)$ space,
- $\boldsymbol{k} = H^\top \boldsymbol{q}$, which can be pre-computed in $O(nk)$ time and stored in $O(k)$ space, and
- $\boldsymbol{p}^\top \mathbf{1}_m$ and $\boldsymbol{q}^\top \mathbf{1}_n$, which can be pre-computed in $O(m + n)$ time and stored in $O(1)$ spaces,

values in (8)-(10) can be computed efficiently:

$$(8) = \bar{a}^2 \operatorname{tr}\big(\mathbf{1}_m^\top \operatorname{diag}(\boldsymbol{p}) \mathbf{1}_m \mathbf{1}_n^\top \operatorname{diag}(\boldsymbol{q}) \mathbf{1}_n\big)$$

$$= \bar{a}^2 \big(\boldsymbol{p}^\top \mathbf{1}_m\big)\big(\boldsymbol{q}^\top \mathbf{1}_n\big), \quad (11)$$

$$(9) = \operatorname{tr}\Big(\underbrace{(XW)^\top}_{B^\top} \operatorname{diag}(\boldsymbol{p}) \underbrace{(XW)}_{B} \underbrace{\big(H^\top \operatorname{diag}(\boldsymbol{q}) H\big)}_{M}\Big)$$

$$= \langle B, \operatorname{diag}(\boldsymbol{p}) B M \rangle, \quad (12)$$

Algorithm 1 Objective value evaluation: $g(\tilde{\boldsymbol{w}})$

```
• Pre-processing step: before any function evaluation
    $M = H^\top \operatorname{diag}(\boldsymbol{q})H$    $\cdots O(nk^2)$
    $\boldsymbol{d} = X^\top \boldsymbol{p}$    $\cdots O(\operatorname{nnz}(X))$
    $\boldsymbol{k} = H^\top \boldsymbol{q}$    $\cdots O(nk)$
    Compute $\boldsymbol{p}^\top \mathbf{1}_m$ and $\boldsymbol{q}^\top \mathbf{1}_n$    $\cdots O(m+n)$
• Each time the function value at $\tilde{\boldsymbol{w}}$ is evaluated:
    $B = XW$, where $B = [\cdots \boldsymbol{b}_i \cdots]^\top$    $\cdots O(\operatorname{nnz}(X)k)$
    $L^+ = \sum_{(i,j)\in\Omega^+} \ell_{ij}^+(Y_{ij}, \boldsymbol{b}_i^\top \boldsymbol{h}_j)$    $\cdots O(|\Omega^+|k)$
    $L^- = \bar{a}^2 (\boldsymbol{p}^\top \mathbf{1}_m)(\boldsymbol{q}^\top \mathbf{1}_n) + \langle B, \operatorname{diag}(\boldsymbol{p})BM\rangle$
        $-2\bar{a}\boldsymbol{d}^\top W \boldsymbol{k}$    $\cdots O(mk^2 + dk)$
    Return $L^+ + L^- + \lambda_w(\|W\|_F^2 + \lambda_g \operatorname{tr}(B^\top L B))$
        $\cdots O(dk + \operatorname{nnz}(L)k)$
```

$$(10) = -2\bar{a}\operatorname{tr}\big(\underbrace{\mathbf{1}_m^\top \operatorname{diag}(\boldsymbol{p})(X)}_{\boldsymbol{d}^\top} W \underbrace{H^\top \operatorname{diag}(\boldsymbol{q})\mathbf{1}_n}_{\boldsymbol{k}}\big)$$

$$= -2\bar{a}\boldsymbol{d}^\top W \boldsymbol{k}, \tag{13}$$

where $\langle \cdot, \cdot \rangle$ is the element-wise inner product between two same-sized matrices. As a result, computing $L^-(\tilde{\boldsymbol{w}})$ costs $O(mk^2 + dk)$ time when $M, \boldsymbol{d}, \boldsymbol{k}, \boldsymbol{p}^\top \mathbf{1}_m$, and $\boldsymbol{q}^\top \mathbf{1}_n$ are pre-computed and B has been obtained in calculating $L^+(\tilde{\boldsymbol{w}})$. In general neither mk nor d is larger than $|\Omega^+|$, so by our setting the cost is comparable to that for $L^+(\tilde{\boldsymbol{w}})$.

To compute $\mathcal{R}_{\tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{w}})$, via $B = XW$ obtained earlier, we calculate $\mathcal{R}_{\tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{w}}) = \operatorname{tr}(\tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{w}} + \lambda_g B^\top L B)$ in $O(dk + \operatorname{nnz}(L)k)$ time. Details of the description of the function evaluation are given in Algorithm 1.

3.2 Gradient and Hessian-vector Product

Given a current point $\tilde{\boldsymbol{w}}$, calculating gradient $\nabla g(\tilde{\boldsymbol{w}})$ and Hessian-vector product $\nabla^2 g(\tilde{\boldsymbol{w}})\tilde{\boldsymbol{s}}$ for a vector $\tilde{\boldsymbol{s}} \in \mathbb{R}^{dk}$ are two common operations in iterative optimization algorithms such as Newton methods. Let $S \in \mathbb{R}^{d\times k}$ be the matrix such that $\tilde{\boldsymbol{s}} = \operatorname{vec}(S)$. With careful derivations (see Section Supp-1.2 and Section Supp-1.3 in the supplementary materials), we show that both operations can be computed by a sequence of matrix-matrix products with a time complexity only linear in $|\Omega^+|$ instead of $|\Omega| = mn$:

$$\nabla g(\tilde{\boldsymbol{w}}) = \operatorname{vec}\big(X^\top[D^+H + 2\operatorname{diag}(\boldsymbol{p})(BM) + 2\lambda_w\lambda_g LB] + 2\lambda_w W - 2\bar{a}\boldsymbol{d}\boldsymbol{k}^\top\big), \tag{14}$$

$$\nabla^2 g(\tilde{\boldsymbol{w}})\tilde{\boldsymbol{s}} = \operatorname{vec}\big(X^\top[U^+H + 2\operatorname{diag}(\boldsymbol{p})NM + 2\lambda_w\lambda_g LN] + 2\lambda_w S\big), \tag{15}$$

where $N = XS$, D^+ and U^+ are sparse matrices with $\forall(i,j) \in \Omega^+$:

$$D_{ij}^+ = \ell_{ij}^{+\prime}, \quad U_{ij}^+ = \ell_{ij}^{+\prime\prime}\boldsymbol{x}_i^\top S \boldsymbol{h}_j,$$

$$\ell_{ij}^{+\prime} = \ell_{ij}^{+\prime}(Y_{ij}, \tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{x}}_{ij}), \text{ and } \ell_{ij}^{+\prime\prime} = \ell_{ij}^{+\prime\prime}(Y_{ij}, \tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{x}}_{ij}).$$

With (14), gradient calculation can be done in

$$O(\operatorname{nnz}(X)k + |\Omega^+|k + \operatorname{nnz}(L)k + mk^2 + dk) \tag{16}$$

time. See Algorithm 2 for details. Similarly, with (15), each Hessian-vector products can be done in

$$O(\operatorname{nnz}(X)k + |\Omega^+|k + \operatorname{nnz}(L)k + mk^2 + dk) \tag{17}$$

time. See details in Algorithm 3. From (16) and (17), clearly the cost is related to $O(|\Omega^+|)$ rather than $O(|\Omega|) = O(mn)$.

Algorithm 2 Gradient evaluation: $\nabla g(\tilde{\boldsymbol{w}})$

```
• Pre-processing step: before any gradient evaluation
    $M = H^\top \operatorname{diag}(\boldsymbol{q})H$    $\cdots O(nk^2)$
    $\boldsymbol{d} = X^\top \boldsymbol{p}$    $\cdots O(\operatorname{nnz}(X))$
    $\boldsymbol{k} = H^\top \boldsymbol{q}$    $\cdots O(nk)$
• Each time the gradient at $\tilde{\boldsymbol{w}}$ is evaluated:
    $B = XW$, where $B = [\cdots \boldsymbol{b}_i \cdots]^\top$    $\cdots O(\operatorname{nnz}(X)k)$
    $D_{ij}^+ = \ell_{ij}^{+\prime}(Y_{ij}, \boldsymbol{b}_i^\top \boldsymbol{h}_j),\ \forall(i,j) \in \Omega^+$    $\cdots O(|\Omega^+|k)$
    $G = X^\top(D^+H + 2\operatorname{diag}(\boldsymbol{p})(BM) + 2\lambda_w\lambda_g LB)$
        $\cdots O(|\Omega^+|k + \operatorname{nnz}(X)k + \operatorname{nnz}(L)k + mk^2)$
    Return $\operatorname{vec}(G + 2\lambda_w W - 2\bar{a}\boldsymbol{d}\boldsymbol{k}^\top)$    $\cdots O(dk)$
```

Algorithm 3 Hessian-vector Product: $\nabla^2 g(\tilde{\boldsymbol{w}})\tilde{\boldsymbol{s}}$

```
• Pre-processing step: before any Hessian-vector product
    $M = H^\top \operatorname{diag}(\boldsymbol{q})H$    $\cdots O(nk^2)$
    $B = XW$, where $B = [\cdots \boldsymbol{b}_i \cdots]^\top$    $\cdots O(\operatorname{nnz}(X)k)$
    $\ell_{ij}^{+\prime\prime} = \ell_{ij}^{+\prime\prime}(Y_{ij}, \boldsymbol{b}_i^\top \boldsymbol{h}_j),\ \forall(i,j) \in \Omega^+$    $\cdots O(|\Omega^+|k)$
• Each time the Hessian-vector product at $\tilde{\boldsymbol{w}} = \operatorname{vec}(W)$ is
  required with a given vector $\tilde{\boldsymbol{s}} = \operatorname{vec}(S)$:
    $N = XS$, where $N = [\cdots \boldsymbol{n}_i \cdots]^\top$    $\cdots O(\operatorname{nnz}(X)k)$
    $U_{ij}^+ = \ell_{ij}^{+\prime\prime}\boldsymbol{n}_i^\top \boldsymbol{h}_j,\ \forall(i,j) \in \Omega^+$    $\cdots O(|\Omega^+|k)$
    $G = X^\top(U^+H + 2\operatorname{diag}(\boldsymbol{p})(NM) + 2\lambda_w\lambda_g LN)$
        $\cdots O(|\Omega^+|k + \operatorname{nnz}(X)k + \operatorname{nnz}(L)k + mk^2)$
    Return $\operatorname{vec}(G + 2\lambda_w S)$    $\cdots O(dk)$
```

3.3 Overall Optimization Procedure

With the efficient procedures proposed in Sections 3.1-3.2, we are able to implement many optimization algorithms to solve (7) such as gradient descent with line search, conjugate gradient, truncated Newton methods with line search, and trust region Newton methods. In Section Supp-1.4 of the supplementary materials, we give an efficient line-search procedure that only costs $O(|\Omega^+|k)$ time to check if the function value is sufficiently decreased.

To illustrate how Algorithms 1-3 are used, in the supplementary materials, we give details of a trust-region Newton method in Algorithm 5 and a gradient descent method with line search in Algorithm 6.

4 Experimental Results

We study one-class MF with various types of side information: standard one-class recommender systems with and without graph information, and multi-label learning by one-class MF with feature vectors as the side information.

Compared Formulations We consider three one-class MF formulations described in Table 1: SQ-SQ, SQ-wSQ, and General-wSQ. In SQ-SQ, we apply the square loss on

Table 2: Data statistics.

	m	n	$\|\Omega^+\|$	d	$\mathrm{nnz}(X)$	m_{test}	k
bibtex	4,880	159	11,729	1,836	335,455	2,512	150
mediamill	30,993	101	135,752	120	3,719,160	12,425	100
delicious	12,920	983	245,810	500	234,740	3,182	300
eurlex	17,413	3,993	92,452	5,000	4,121,532	1,933	500
wiki10	14,146	30,938	263,705	101,938	9,526,572	6,616	500

(a) Multi-label learning.

	m	n	$\|\Omega^+\|$	$\mathrm{nnz}(S)$	$\|\Omega^+_{\text{test}}\|$	k
ml100k	943	1,682	49,791	29,235	5,584	64
flixster	147,612	48,794	3,619,304	2,538,746	401,917	100
douban	129,490	58,541	9,803,098	1,711,780	1,087,948	128

(b) One-class recommender systems with graph information, S, among m users

entries in both Ω^+ and Ω^-. The difference of SQ-wSQ to SQ-SQ is that we apply the weighted square loss on Ω^-. For General-wSQ, we consider a classification loss: the logistic loss, and denote the formulation as LR-wSQ. Specifically, we apply the logistic loss on entries in Ω^+ and the weighted square loss on Ω^-. We implement the trust region Newton method (Lin, Weng, and Keerthi 2008) with the efficient function/gradient evaluation and Hessian-vector product proposed in Section 3. We also include LR-wLR into the multi-label learning experiments. LR-wLR is a Full approach where we apply the logistic loss on the entries in Ω^+ and apply the weighted logistic loss on entries in $[m] \times [n] \setminus \Omega^+$. It can be used only for small datasets because, without efficient procedures like those in Section 3, each iteration takes $O(mnk)$ cost.

Datasets and Experimental Settings For one-class MF with graph information as the side information, we consider three rating-based recommender system datasets with graph information, S, among m users (see Table 2): ml100k (Rao et al. 2015), flixster (Jamali and Ester 2010), and douban (Ma et al. 2011) and convert them into one-class problems by the same procedure in (Yu, Bilenko, and Lin 2016). Because the feature information is not available, X is the identity matrix. For multi-label learning, we consider five publicly available datasets (see Table 2) and set $\lambda_g = 0$ because the graph information S is not available. All experiments are run on a 20-core machine with 256GB RAM. BLAS operations are used whenever possible for all the implementations. Sparse matrix operations are parallelized using OpenMP. We use precision and nDCG on the top-rank predictions as our evaluation criteria for both recommender systems and multi-label learning. We only show precision@5 in Figures 1-3 and leave figures with other criteria in the supplementary materials. We adopt a held-out validation approach on the training part of each dataset to perform parameter search. See Section Supp-2 in the supplementary materials for more experimental details.

4.1 Full versus Subsampled

We compare Full and Subsampled approaches on multi-label problems, which are special cases of feature-aware one-class MF. An extensive comparison for the standard one-class MF without features has been performed in (Yu, Bilenko, and Lin 2016). Figure 1 shows both training time and test performance on three large datasets.

Table 3: Comparison between LR-wLR and LR-wSQ.

		time	$p@1$	$p@2$	$p@3$	$p@4$	$p@5$
bibtex	LR-wSQ	22	63.22	48.43	39.89	33.83	29.50
	LR-wLR	85	61.91	46.66	38.21	32.66	28.43
mediamill	LR-wSQ	61	87.77	80.96	70.08	61.48	55.15
	LR-wLR	437	87.78	81.17	70.38	61.85	55.33
delicious	LR-wSQ	23	67.47	64.24	61.85	59.25	56.73
	LR-wLR	446	67.91	65.04	62.27	59.81	57.27

From Figure 1(a), the three Full approaches: SQ-SQ, SQ-wSQ, and LR-wSQ are highly efficient because with our algorithms in Section 3, the cost per iteration of the trust region Newton method is related to $|\Omega^+|$ rather than $|\Omega| = mn$. Thus the costs of these three Full approaches are similar to the costs of the two Subsampled approaches: SQ-wSQ (S) and LR-wLR (S). Regarding the prediction performance, from Figure 1(b), the Full formulations always outperform the Subsampled formulations. Such observations are consistent with those for one-class MF without side information (Yu, Bilenko, and Lin 2016).

4.2 Comparison on Various Loss Functions

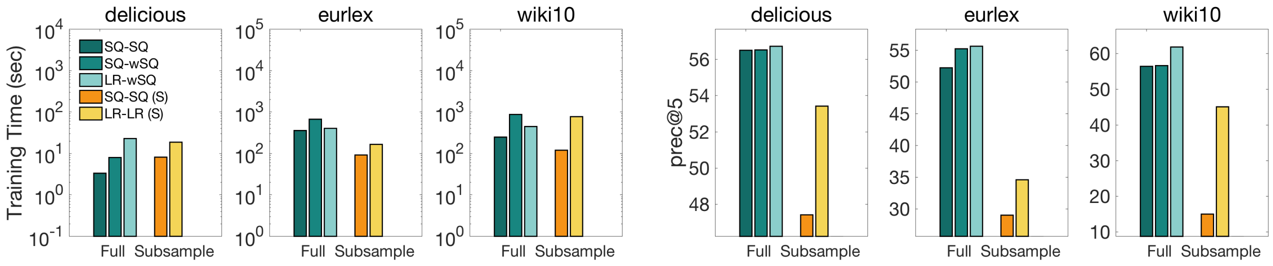
We check various formulations of the Full approach for feature-aware or graph structured one-class MF to study the effectiveness of (i) the use of weighted losses on the entries in Ω^-; and (ii) the use of a classification loss.

LR-wSQ versus LR-wLR To resolve the $O(mnk)$ computational requirement for the Full approach with a general loss function, in Section 3, we propose a technique where the general loss is applied to the observations in Ω^+, while a weighted squared loss is applied to Ω^-. In Table 3, we show the results of LR-wSQ and LR-wLR on three smaller datasets. We observe that LR-wSQ and LR-wLR are similar in terms of prediction performance, while LR-wSQ runs significantly faster. This result indicates that our proposed technique to replace the loss on Ω^- by a weighted loss not only accelerates the training procedure significantly but also yields a similar prediction performance.

SQ-wSQ versus LR-wSQ. By considering three types of one-class problems, in Figure 2, we show the performance improvement of SQ-wSQ and LR-wSQ over SQ-SQ (in percentage) because for recommender systems with graph information, SQ-SQ is equivalent to GRALS (Rao et al. 2015), while for multi-label learning, SQ-SQ is equivalent to LEML (Yu et al. 2014). Clearly LR-wSQ significantly outperforms SQ-wSQ and SQ-SQ. Therefore, the logistic loss seems to be more appropriate for one-class MF problems, and our approach enables its use.

4.3 Non-linear Multi-label Classifiers

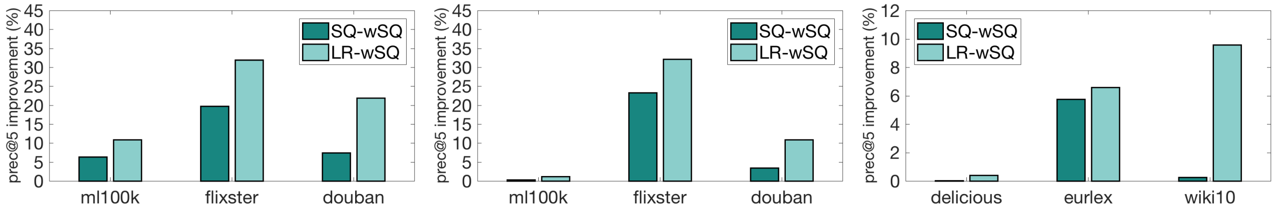
Recently, some non-linear methods such as SLEEC (Bhatia et al. 2015) and FastXML (Prabhu and Varma 2014) have



(a) Training Time.

(b) Prediction Performance.

Figure 1: Full versus Subsampled on multi-label learning with various loss functions. “(S)” indicates Subsampled.

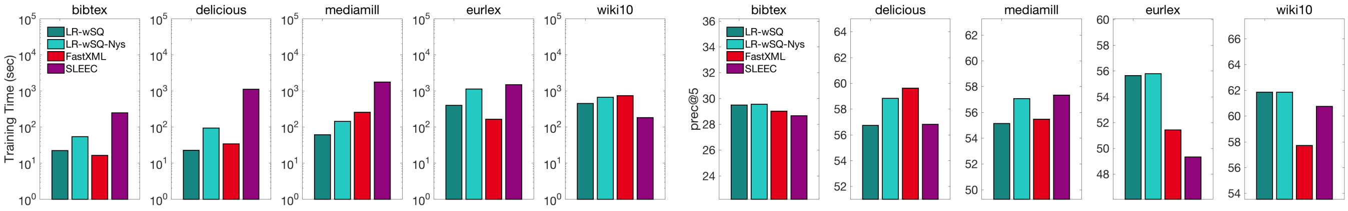


(a) Standard one-class MF

(b) Graph-structured one-class MF

(c) Feature-aware one-class MF

Figure 2: Comparison on various loss functions. Y-axis is the improvement over the SQ-SQ formulation in percentage.



(a) Training Time.

(b) Prediction Performance.

Figure 3: Comparison on non-linear multi-label classifiers.

been considered state-of-the-art approaches for multi-label learning because they outperform traditional binary relevance approaches. Here we demonstrate that the Full setting considered in this paper yields competitive performances.

Besides directly applying the proposed approach (i.e., LR-wSQ) on feature vectors $\boldsymbol{x}_i$, $\forall i$, we propose a non-linear extension by converting $\boldsymbol{x}_i \in \mathbb{R}^d$ into a new feature vector $\boldsymbol{z}_i \in \mathbb{R}^D$ via a Nyström method. We consider K-means Nyström (Zhang, Tsang, and Kwok 2008) by using $D \ll m$ centers $\{\boldsymbol{c}_t\}$ from the K-means algorithm on $\{\boldsymbol{x}_i\}$ as landmark points. Consider an RBF kernel matrix $K \in \mathbb{R}^{D \times D}$ parameterized by $\gamma > 0$ with

$$K_{ts} = \exp(-\gamma \|\boldsymbol{c}_t - \boldsymbol{c}_s\|^2), \ \forall t, s \in [D] \times [D].$$

Given an $\boldsymbol{x} \in \mathbb{R}^d$, the transformed vector $\boldsymbol{z} \in \mathbb{R}^D$ is obtained by $\boldsymbol{z} = K^{-1/2}\bar{\boldsymbol{z}}$, where $\bar{\boldsymbol{z}} \in \mathbb{R}^D$ is a vector with

$$\bar{z}_t = \exp(-\gamma \|\boldsymbol{c}_t - \boldsymbol{x}\|^2), \ \forall t \in [D].$$

We append the transformed vector $\boldsymbol{z}_i$ to the original $\boldsymbol{x}_i$. See the supplementary materials for details such as the parameters used. We refer tot this approach as LR-wSQ-Nys.

Figure 3 compares LR-wSQ, LR-wSQ-Nys, FastXML, and SLEEC. We can see that LR-wSQ and LR-wSQ-Nys yield competitive prediction performance.

5 Conclusions

In this paper, we consider a generalization of one-class matrix factorization with two types of side information, which has wide applicability to applications with only positive-unlabeled observations such as recommender systems and multi-label learning. We consider the Full approach for one-class MF where all the unlabeled entries are regarded as negative with a lower confidence, and any convex loss function can be used on the observed entries. The resulting optimization problem is very challenging, but we derive an efficient alternating minimization procedure. Experiments demonstrate the effectiveness of our proposed approach.

Acknowledgement. This work was partially supported by NSF grants CCF-1320746, IIS-1546452 and CCF-1564000, and by MOST of Taiwan grant 104-2221-E-002-047-MY3.

References

Bhatia, K.; Jain, H.; Kar, P.; Varma, M.; and Jain, P. 2015. Sparse local embeddings for extreme multi-label classification. In Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 28*, 730–738.

Hsieh, C.-J.; Natarajan, N.; and Dhillon, I. 2015. PU learning for matrix completion. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2445–2453.

Hu, Y.; Koren, Y.; and Volinsky, C. 2008. Collaborative filtering for implicit feedback datasets. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, 263–272.

Jamali, M., and Ester, M. 2010. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, 135–142.

Kong, X.; Wu, Z.; Li, L.-J.; Zhang, R.; Yu, P. S.; Wu, H.; and Fan, W. 2014. Large-scale multi-label learning with incomplete label assignments. In *Proceedings of SIAM International Conference on Data Mining*, 920–928.

Li, W.-J., and Yeung, D.-Y. 2009. Relation regularized matrix factorization. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*.

Li, Y.; Hu, J.; Zhai, C.; and Chen, Y. 2010. Improving one-class collaborative filtering by incorporating rich user information. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM)*, 959–968.

Lin, C.-J.; Weng, R. C.; and Keerthi, S. S. 2008. Trust region Newton method for large-scale logistic regression. *Journal of Machine Learning Research* 9:627–650.

Ma, H.; Zhou, D.; Liu, C.; Lyu, M. R.; and King, I. 2011. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*, 287–296.

Natarajan, N.; Rao, N.; and Dhillon, I. S. 2015. PU matrix completion with graph information. In *International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, 37–40.

Pan, R., and Scholz, M. 2009. Mind the gaps: Weighting the unknown in large-scale one-class collaborative filtering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 667–676.

Pan, R.; Zhou, Y.; Cao, B.; Liu, N. N.; Lukose, R.; Scholz, M.; and Yang, Q. 2008. One-class collaborative filtering. In *IEEE International Conference on Data Mining (ICDM)*, 502–511.

Paquet, U., and Koenigstein, N. 2013. One-class collaborative filtering with random graphs. In *Proceedings of the 22nd international conference on World Wide Web*, 999–1008.

Prabhu, Y., and Varma, M. 2014. FastXML: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 263–272.

Rao, N.; Yu, H.-F.; Ravikumar, P.; and Dhillon, I. S. 2015. Collaborative filtering with graph information: Consistency and scalable methods. In Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 28*, 2107–2115.

Singh, A. P., and Gordon, G. J. 2008. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, 650–658.

Smola, A. J., and Kondor, R. 2003. Kernels and regularization on graphs. In *Learning theory and kernel machines*. Springer. 144–158.

Yu, H.-F.; Jain, P.; Kar, P.; and Dhillon, I. S. 2014. Large-scale multi-label learning with missing labels. In *Proceedings of the Thirty First International Conference on Machine Learning (ICML)*, 593–601.

Yu, H.-F.; Bilenko, M.; and Lin, C.-J. 2016. Selection of negative samples for one-class matrix factorization. Technical report, National Taiwan University.

Zhang, K.; Tsang, I. W.; and Kwok, J. T. 2008. Improved Nyström low-rank approximation and error analysis. In *Proceedings of the Twenty Fifth International Conference on Machine Learning (ICML)*.

Zhao, Z.; Zhang, L.; He, X.; and Ng, W. 2015. Expert finding for question answering via graph regularized matrix completion. *IEEE Transactions on Knowledge and Data Engineering* 27:993–1004.

Zhou, T.; Shan, H.; Banerjee, A.; and Sapiro, G. 2012. Kernelized probabilistic matrix factorization: Exploiting graphs and side information. In *SIAM International Conference on Data Mining*, 403–414.