Supplementary Materials:

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

Supp-1 Detailed Derivations of Algorithms

Supp-1.1 Detailed Derivation of $L^-(\mathrm{vec}(W))$

From (4), (6), and (7), we have

$$\begin{aligned}
L^-(\mathrm{vec}(W)) &= \left\| \sqrt{\mathrm{diag}(\boldsymbol{p})} \big(\bar{a}\mathbf{1}_m\mathbf{1}_n^\top - XWH^\top\big) \sqrt{\mathrm{diag}(\boldsymbol{q})} \right\|_F^2 \\
&= \mathrm{tr}\left(\sqrt{\mathrm{diag}(\boldsymbol{q})}\big(\bar{a}\mathbf{1}_n\mathbf{1}_m^\top - HW^\top X^\top\big)\,\mathrm{diag}(\boldsymbol{p}) \times \right. \\
&\qquad \left. \big(\bar{a}\mathbf{1}_m\mathbf{1}_n^\top - XWH^\top\big)\sqrt{\mathrm{diag}(\boldsymbol{q})}\right) \\
&= \mathrm{tr}\left(\big(\bar{a}\mathbf{1}_n\mathbf{1}_m^\top - HW^\top X^\top\big)\,\mathrm{diag}(\boldsymbol{p}) \times \right. \\
&\qquad \left. \big(\bar{a}\mathbf{1}_m\mathbf{1}_n^\top - XWH^\top\big)\mathrm{diag}(\boldsymbol{q})\right) \\
&= \mathrm{tr}\big(\bar{a}^2\mathbf{1}_n\mathbf{1}_m^\top\,\mathrm{diag}(\boldsymbol{p})\mathbf{1}_m\mathbf{1}_n^\top\,\mathrm{diag}(\boldsymbol{q})\big) \\
&\quad + \mathrm{tr}\big(HW^\top X^\top\,\mathrm{diag}(\boldsymbol{p})XWH^\top\,\mathrm{diag}(\boldsymbol{q})\big) \\
&\quad - 2\,\mathrm{tr}\big(\bar{a}\mathbf{1}_n\mathbf{1}_m^\top\,\mathrm{diag}(\boldsymbol{p})XWH^\top\,\mathrm{diag}(\boldsymbol{q})\big),
\end{aligned}$$

where $\mathbf{1}_m$ and $\mathbf{1}_n$ are vectors of ones with order m and n, respectively, and $\mathrm{tr}(\cdot)$ denotes the trace of a matrix (i.e., the sum of diagonal entries).

Supp-1.2 Gradient Calculation $\nabla g(\tilde{\boldsymbol{w}})$

The following property is useful in subsequent derivations.

Property 1. *Let $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m]^\top$, $H = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n]^\top$, and $D \in \mathbb{R}^{m\times n}$. We have*

$$X^\top DH = \sum_{ij} \boldsymbol{x}_i D_{ij} \boldsymbol{h}_j^\top \tag{1.1}$$

From (7), the gradient of $g(\tilde{\boldsymbol{w}})$ is

$$\nabla g(\tilde{\boldsymbol{w}}) = \nabla L^+(\tilde{\boldsymbol{w}}) + \nabla L^-(\tilde{\boldsymbol{w}}) + \lambda_w \nabla \mathcal{R}_{\tilde{w}}(\tilde{\boldsymbol{w}}),$$

where

$$\begin{aligned}
\nabla L^+(\tilde{\boldsymbol{w}}) &= \sum_{(i,j)\in\Omega^+} \ell_{ij}^{+\prime} \tilde{\boldsymbol{x}}_{ij} = \sum_{(i,j)\in\Omega^+} \mathrm{vec}\Big(\ell_{ij}^{+\prime} \boldsymbol{x}_i \boldsymbol{h}_j^\top\Big) \\
&= \mathrm{vec}\big(X^\top D^+ H\big). \qquad (1.2)
\end{aligned}$$

The last equality follows from (1.1), $\ell_{ij}^{+\prime}$ is the abbreviation of $\ell_{ij}^{+\prime}(Y_{ij}, \tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{x}}_{ij})$ and,

$$D_{ij}^+ = \begin{cases} \ell_{ij}^{+\prime}, & \forall (i,j) \in \Omega^+, \\ 0, & \text{otherwise.} \end{cases}$$

Similar to the situation of computing $L^+(\tilde{\boldsymbol{w}})$ in Section 3.1, if $B = XW$ has been calculated, then constructing a sparse matrix D^+ requires only $O(|\Omega^+|k)$ operations and $O(|\Omega^+|)$ space. With a similar derivation in (1.2), we have

$$\nabla L^-(\tilde{\boldsymbol{w}}) = \mathrm{vec}\big(X^\top D^- H\big),$$

where D^- is a dense $m \times n$ matrix with $D_{ij}^- = \ell_{ij}^{-\prime}$, $\forall (i,j) \in [m] \times [n]$. From the definition of $\ell^-(\cdot,\cdot)$ in (6),

$$\begin{aligned}
&\frac{\partial}{\partial b}\ell^-(\bar{a}, b) = 2p_i q_j (b - \bar{a}) \\
\Rightarrow\quad & \ell_{ij}^{-\prime} = 2p_i q_j \big(\tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{x}}_{ij} - \bar{a}\big) = 2p_i q_j \Big(\big(XWH^\top\big)_{ij} - \bar{a}\Big).
\end{aligned}$$

Thus, we have

$$D^- = 2\,\mathrm{diag}(\boldsymbol{p})\big(XWH^\top - \bar{a}\mathbf{1}_m\mathbf{1}_n^\top\big)\,\mathrm{diag}(\boldsymbol{q}).$$

Straightforward computation of entire D^- requires $O(mnk + \mathrm{nnz}(X)k)$ time and $O(mn)$ space; both requirements are infeasible for large values of m and n. Fortunately, $\nabla L^-(\tilde{\boldsymbol{w}}) = \mathrm{vec}\big(X^\top D^- H\big)$ can be computed without explicitly forming D^- as follows:

$$\begin{aligned}
&X^\top D^- H \\
=&2X^\top\,\mathrm{diag}(\boldsymbol{p})XWH^\top\,\mathrm{diag}(\boldsymbol{q})H \\
&\quad - 2\bar{a}X^\top\,\mathrm{diag}(\boldsymbol{p})\mathbf{1}_m\mathbf{1}_n^\top\,\mathrm{diag}(\boldsymbol{q})H \qquad (1.3) \\
=&2X^\top\,\mathrm{diag}(\boldsymbol{p})\underbrace{(XW)}_{B}\underbrace{\big(H^\top\,\mathrm{diag}(\boldsymbol{q})H\big)}_{M} - 2\bar{a}\,\underbrace{\big(X^\top\boldsymbol{p}\big)}_{\boldsymbol{d}}\underbrace{\big(\boldsymbol{q}^\top H\big)}_{\boldsymbol{k}^\top},
\end{aligned}$$

where B, M, $\boldsymbol{d}$, and $\boldsymbol{k}$ are the same matrices/vectors used in the objective value evaluation introduced in Algorithm 1. Combining (1.2), (1.3), and $\tilde{\boldsymbol{w}} = \mathrm{vec}(W)$, $\nabla g(\tilde{\boldsymbol{w}})$ can be computed using the following sequence of matrix-matrix products:

$$\begin{aligned}
\nabla g(\tilde{\boldsymbol{w}}) &= \mathrm{vec}\big(X^\top D^+ H + X^\top D^- H + \lambda_w \nabla \mathcal{R}_{\tilde{w}}(\tilde{\boldsymbol{w}})\big) \\
&= \mathrm{vec}\left(X^\top\big[D^+ H + 2\,\mathrm{diag}(\boldsymbol{p})(BM) + 2\lambda_w\lambda_g LB\big]\right. \\
&\qquad \left. + 2\lambda_w W - 2\bar{a}\boldsymbol{d}\boldsymbol{k}^\top\right),
\end{aligned}$$

where the last equality follows from (1.3) and

$$\begin{aligned}
\nabla \mathcal{R}_{\tilde{w}}(\tilde{\boldsymbol{w}}) &= 2\,\mathrm{vec}\big((I + \lambda_g X^\top L X)W\big) \\
&= 2\,\mathrm{vec}\big(W + \lambda_g X^\top LB\big).
\end{aligned}$$

Thus, the cost is

$$O(\mathrm{nnz}(X)k + \big|\Omega^+\big|k + \mathrm{nnz}(L)k + mk^2 + dk) \tag{1.4}$$

time. A detailed procedure is shown in Algorithm 2. Note that if the computation of $\nabla g(\tilde{\boldsymbol{w}})$ follows immediately after the objective value evaluation of the same $\tilde{\boldsymbol{w}}$, then M, $\boldsymbol{d}$, $\boldsymbol{q}$, $\boldsymbol{k}$, and B can be re-used to save additional computation.[2]

Note that we may use a different sequence of matrix-matrix products for (1.3) if the number of features d is very small such that

$$d^2 + dk < mk. \tag{1.5}$$

By pre-computing $\bar{M} = X^\top\,\mathrm{diag}(\boldsymbol{p})X$ in $O(\mathrm{nnz}(X)d)$ time and storing it in $O(d^2)$ space, the following sequence to compute $\nabla g(\tilde{\boldsymbol{w}})$,

$$\begin{aligned}
\nabla g(\tilde{\boldsymbol{w}}) &= \mathrm{vec}\big(X^\top D^+ H + X^\top D^- H + \lambda_w \nabla \mathcal{R}_{\tilde{w}}(\tilde{\boldsymbol{w}})\big) \\
&= \mathrm{vec}\left(X^\top\big(D^+ H + 2\lambda_w\lambda_g LB\big) + 2\bar{M}WM\right. \\
&\qquad \left. + 2\lambda_w W - 2\bar{a}\boldsymbol{d}\boldsymbol{k}^\top\right),
\end{aligned}$$

costs

$$O(\mathrm{nnz}(X)k + \big|\Omega^+\big|k + \mathrm{nnz}(L)k + d^2k + dk^2)$$

time, which is smaller than that in (1.4) because of (1.5).

[2] $\mathrm{diag}(\boldsymbol{p})(BM)$ can also be re-used.

Supp-1.3 Hessian-vector Multiplication $\nabla^2 g(\tilde{\boldsymbol{w}})\tilde{\boldsymbol{s}}$

Given a current point $\tilde{\boldsymbol{w}}$ and a vector $\tilde{\boldsymbol{s}} \in \mathbb{R}^{dk}$, computing $\nabla^2 g(\tilde{\boldsymbol{w}})\boldsymbol{s}$ is a common operation required in many iterative optimization algorithms such as conjugate gradient. Let $S \in \mathbb{R}^{d\times k}$ be the matrix such that $\tilde{\boldsymbol{s}} = \mathrm{vec}(S)$. From (7), we have

$$\nabla^2 g(\tilde{\boldsymbol{w}})\tilde{\boldsymbol{s}} = \nabla^2 L^+(\tilde{\boldsymbol{w}})\tilde{\boldsymbol{s}} + \nabla^2 L^-(\tilde{\boldsymbol{w}})\tilde{\boldsymbol{s}} + \lambda_w \nabla^2 \mathcal{R}_{\tilde{w}}(\tilde{\boldsymbol{w}})\tilde{\boldsymbol{s}},$$

where

$$\begin{aligned}
\nabla^2 L^+(\tilde{\boldsymbol{w}})\tilde{\boldsymbol{s}} &= \sum_{(i,j)\in\Omega^+} \ell_{ij}^{+\prime\prime} \tilde{\boldsymbol{x}}_{ij}\tilde{\boldsymbol{x}}_{ij}^\top \tilde{\boldsymbol{s}} \\
&= \sum_{(i,j)\in\Omega^+} \ell_{ij}^{+\prime\prime} \big((\boldsymbol{h}_j \boldsymbol{h}_j^\top) \otimes (\boldsymbol{x}_i \boldsymbol{x}_i^\top) \big) \,\mathrm{vec}(S) \\
&= \sum_{(i,j)\in\Omega^+} \mathrm{vec}\Big(\ell_{ij}^{+\prime\prime} \boldsymbol{x}_i \boldsymbol{x}_i^\top S \boldsymbol{h}_j \boldsymbol{h}_j^\top \Big) \qquad (1.6) \\
&= \sum_{(i,j)\in\Omega^+} \mathrm{vec}\big(\boldsymbol{x}_i \underbrace{\ell_{ij}^{+\prime\prime} \boldsymbol{x}_i^\top S \boldsymbol{h}_j}_{U_{ij}^+} \boldsymbol{h}_j^\top \big) \\
&= \mathrm{vec}\big(X^\top U^+ H \big). \qquad (1.7)
\end{aligned}$$

Note that (1.6) is from (4), (1.7) is from (1.1), U^+ is a sparse matrix with

$$U_{ij}^+ = \begin{cases} \ell_{ij}^{+\prime\prime} \boldsymbol{x}_i^\top S \boldsymbol{h}_j, & \forall (i,j) \in \Omega^+, \\ 0 & \text{otherwise}, \end{cases}$$

and $\ell_{ij}^{+\prime\prime} = \ell_{ij}^{+\prime\prime}(Y_{ij}, \tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{x}}_{ij})$, $\forall (i,j) \in \Omega^+$. Similar to the gradient computation, $\ell_{ij}^{+\prime\prime}$ can be computed in $O(k)$ time if $B = XW$ is pre-computed. To efficiently compute U_{ij}^+, $\forall (i,j) \in \Omega^+$, we first compute a matrix-matrix product XS in $O(\mathrm{nnz}(X)k)$ time and store it in an $m \times k$ matrix $N = [\ldots, \boldsymbol{n}_i, \ldots]^\top$ such that $\boldsymbol{n}_i = S^\top \boldsymbol{x}_i$. Then the entire sparse matrix U^+ can be constructed in $O(|\Omega^+|k + \mathrm{nnz}(X)k)$ time.

With a similar derivation as above, we have

$$\nabla^2 L^-(\tilde{\boldsymbol{w}})\boldsymbol{s} = \mathrm{vec}\big(X^\top U^- H \big),$$

where U^- is a dense $m \times n$ matrix with

$$U_{ij}^- = \ell_{ij}^{-\prime\prime} \boldsymbol{x}_i^\top S \boldsymbol{h}_j, \ \forall (i,j) \in [m] \times [n].$$

From the definition of $\ell^-(\cdot,\cdot)$ in (6),

$$\frac{\partial^2}{\partial b^2} \ell_{ij}^-(\bar{a}, b) = 2 p_i q_j, \ \forall b \quad \Rightarrow \quad \ell_{ij}^{-\prime\prime} = 2 p_i q_j.$$

Thus, we have

$$U^- = 2\,\mathrm{diag}(\boldsymbol{p}) X S H^\top \,\mathrm{diag}(\boldsymbol{q}).$$

Similar to the situation in the gradient computation, computing $\nabla^2 L^-(\tilde{\boldsymbol{w}})\tilde{\boldsymbol{s}} = \mathrm{vec}\big(X^\top U^- H\big)$ can be done without forming the entire U^- explicitly as follows:

$$X^\top U^- H = 2 X^\top \,\mathrm{diag}(\boldsymbol{p}) \underbrace{(XS)}_{N} \underbrace{H^\top \,\mathrm{diag}(\boldsymbol{q}) H}_{M}, \qquad (1.8)$$

where N is available when we construct the sparse matrix U^+, and M can be pre-computed and stored (same M in the objective value evaluation and gradient computation). Combining (1.7), (1.8), and $\tilde{\boldsymbol{w}} = \mathrm{vec}(W)$, $\nabla^2 g(\tilde{\boldsymbol{w}})\tilde{\boldsymbol{s}}$ can be computed using the following sequence of matrix-matrix products:

$$\begin{aligned}
&\nabla^2 g(\tilde{\boldsymbol{w}})\tilde{\boldsymbol{s}} = \mathrm{vec}\big(X^\top U^+ H + X^\top U^- H + \lambda_w \nabla^2 \mathcal{R}_{\tilde{w}}(\tilde{\boldsymbol{w}})\tilde{\boldsymbol{s}} \big) \\
&= \mathrm{vec}\big(X^\top \big[U^+ H + 2\,\mathrm{diag}(\boldsymbol{p}) N M \big] + \lambda_w \nabla^2 \mathcal{R}_{\tilde{w}}(\tilde{\boldsymbol{w}})\tilde{\boldsymbol{s}} \big) \\
&= \mathrm{vec}\big(X^\top \big[U^+ H + 2\,\mathrm{diag}(\boldsymbol{p}) N M + 2\lambda_w \lambda_g L N \big] + 2\lambda_w S \big),
\end{aligned}$$

where the last equality follows from

$$\begin{aligned}
\nabla^2 \mathcal{R}_{\tilde{w}}(\tilde{\boldsymbol{w}})\tilde{\boldsymbol{s}} &= 2\,\mathrm{vec}\big(S + \lambda_g X^\top L X S \big) \\
&= 2\,\mathrm{vec}\big(S + \lambda_g X^\top L N \big).
\end{aligned}$$

Thus, the cost is

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

time. Note that we can move the computation of $\ell_{ij}^{+\prime\prime}$, $\forall (i,j) \in \Omega^+$ into the pre-processing phase to save some repeated operations when the Hessian-vector products are performed many times with the same $\tilde{\boldsymbol{w}}$, a common situation in iterative optimization algorithms such as conjugate gradient. Further, when the squared loss $\ell(a,b) = (a-b)^2$ is used, for entries in Ω^+, $\ell_{ij}^{+\prime\prime} = 2(1 - p_i q_j)$ is independent of the choice of $\tilde{\boldsymbol{w}}$ and can be pre-computed in the beginning of the entire optimization procedure. A detailed procedure is presented in Algorithm 3.

Similar to the gradient calculation, when $d^2 + dk < mk$, we can reduce the cost by using another sequence of matrix-matrix products to compute $\nabla^2 g(\tilde{\boldsymbol{w}})\tilde{\boldsymbol{s}}$ in

$$O(|\Omega^+|k + \mathrm{nnz}(X)k + \mathrm{nnz}(L)k + d^2 k + dk^2)$$

time with $\bar{M} = X^\top \,\mathrm{diag}(\boldsymbol{p}) X$ as follows:

$$X^\top \big(U^+ H + 2\lambda_w \lambda_g L N \big) + 2\bar{M} S M + 2\lambda_w S.$$

Supp-1.4 Line Search Procedure and Trust Region Method

Line search procedures and trust region methods are two common techniques to determine a step size and guarantee the asymptotic convergence for an optimization algorithm. In this section, we briefly introduce how to apply these two techniques to (7).

Line Search Procedure A line search procedure is used to ensure the sufficient decrease of the objective function value after a search direction ΔW is given. Common choices for the search direction include the negative gradient direction and the Newton direction. However, even if ΔW is a descent direction, $g(W + \Delta W)$ is not necessary smaller than $g(W)$. Thus, in a backward line search procedure, we try a sequence of step size $\alpha = 1, \beta, \beta^2, \ldots$, with $\beta < 1$ such that

$$g(W + \alpha \Delta W) < g(W) + \sigma \alpha \langle \nabla g(W), \Delta W \rangle, \qquad (1.9)$$

where $\sigma < 1/2$. Note that even with our proposed efficient function value evaluation procedure (Algorithm 1), recalculating function value at $W + \alpha \Delta W$ for each α can still be expensive.

We propose an effective trick to reduce the cost for line search by taking that $\tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{x}}_{ij}$ is a linear function of $\tilde{\boldsymbol{w}}$, $\ell^-_{ij}(\bar{a}, \tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{x}}_{ij})$ is a quadratic function of $\tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{x}}_{ij}$, and $\mathcal{R}(\tilde{\boldsymbol{w}})$ is a quadratic function of $\tilde{\boldsymbol{w}}$. The idea is to have

$$\begin{aligned}
&g(W+\alpha\Delta W)\\
=&L^+(W+\alpha\Delta W)+L^-(W+\alpha\Delta W)+\lambda_w\mathcal{R}(W+\alpha\Delta W)\\
=&g(W)-L^+(W)+L^+(W+\alpha\Delta W)+\alpha\text{Val_1}+\alpha^2\text{Val_2},
\end{aligned}$$

where Val_1 and Val_2 can be calculated with the cost of one function evaluation. If these two values are available, then the sufficient decrease condition in (1.9) can be easily checked as follows.

$$\begin{aligned}
&g(W+\alpha\Delta W)-g(W)\\
=&-L^+(W)+L^+(W+\alpha\Delta W)+\alpha\text{Val_1}+\alpha^2\text{Val_2}\\
<&\sigma\alpha\langle\nabla g(W),\Delta W\rangle.
\end{aligned}$$

Moreover, when the line-search procedure terminates, we have the next iterate $W+\alpha\Delta W$ and the new function value. From (8)-(13) in the main text,

$$\begin{aligned}
&L^-(W+\alpha\Delta W)\\
=&\bar{a}^2(\boldsymbol{p}^\top\mathbf{1}_m)(\boldsymbol{q}^\top\mathbf{1}_n)+\langle B+\alpha\Delta B,\operatorname{diag}(\boldsymbol{p})(B+\alpha\Delta B)M\rangle\\
&\qquad-2\bar{a}\boldsymbol{d}^\top(W+\alpha\Delta W)\boldsymbol{k}\\
=&L^-(W)+\alpha^2\langle\Delta B,\operatorname{diag}(\boldsymbol{p})\Delta BM\rangle+\\
&\alpha\big(\langle\Delta B,\operatorname{diag}(\boldsymbol{p})BM\rangle+\langle B,\operatorname{diag}(\boldsymbol{p})\Delta BM\rangle-2\bar{a}\boldsymbol{d}^\top\Delta W\boldsymbol{k}\big)
\end{aligned}$$

and

$$\begin{aligned}
&\mathcal{R}(W+\alpha\Delta W)\\
=&\operatorname{tr}\Big((W+\alpha\Delta W)^\top(W+\alpha\Delta W)\Big)\\
&+\lambda_g\operatorname{tr}\Big((B+\alpha\Delta B)^\top L(B+\alpha\Delta B)\Big)\\
=&\mathcal{R}(W)+\alpha\big(2\operatorname{tr}\big(W^\top\Delta W+\lambda_g B^\top L\Delta B\big)\big)\\
&+\alpha^2\operatorname{tr}\big(\Delta W^\top\Delta W+\lambda_g\Delta B^\top L\Delta B\big),\\
=&\mathcal{R}(W)+\alpha(2\langle W,\Delta W\rangle+2\lambda_g\langle B,L\Delta B\rangle)\\
&+\alpha^2\Big(\|\Delta W\|_F^2+\lambda_g\langle\Delta B,L\Delta B\rangle\Big),
\end{aligned}$$

where

$$\Delta B = X\Delta W.$$

The above calculation shows that in addition to the matrix B, we can maintain

$$\hat{M}=\operatorname{diag}(\boldsymbol{p})BM,\text{ and }\hat{L}=LB.$$

Before the line-search procedure,

$$\Delta B = X\Delta W, \tag{1.10}$$

$$\Delta\hat{M}=\operatorname{diag}(\boldsymbol{p})\Delta BM, \tag{1.11}$$

$$\Delta\delta=\boldsymbol{d}^\top\Delta W\boldsymbol{k},\text{ and} \tag{1.12}$$

$$\Delta\hat{L}=L\Delta B \tag{1.13}$$

are calculated. Then

$$\begin{aligned}
\text{Val_1}=&\Big\langle\Delta B,\hat{M}\Big\rangle+\Big\langle B,\Delta\hat{M}\Big\rangle-2\bar{a}\Delta\delta\\
&+2\lambda_w\Big(\langle W,\Delta W\rangle+\lambda_g\Big\langle B,\Delta\hat{L}\Big\rangle\Big),\\
\text{Val_2}=&\Big\langle\Delta B,\Delta\hat{M}\Big\rangle\\
&+\lambda_w\Big(\|\Delta W\|_F^2+\lambda_g\Big\langle\Delta B,\Delta\hat{L}\Big\rangle\Big).
\end{aligned}$$

Clearly, the computation for (1.10)-(1.13) costs

$$O(\operatorname{nnz}(X)k+mk^2+dk+\operatorname{nnz}(L)k),$$

which is the same as the cost of one function evaluation. For Val_1 and Val_2, the cost $O(dk+mk)$ is much smaller. To check the sufficient decrease condition (1.9), we precalculate $\nabla g(W)^\top\Delta W$ before the line-search procedure, so the calculation under an α is $O(|\Omega^+|k)$ for calculating $L^+(W+\alpha\Delta W)$.

If the squared loss is considered on entries in Ω^+, the same technique for calculating $L^-(W+\alpha\Delta W)$ can be applied for calculating $L^+(W+\alpha\Delta W)$. Then the $O(|\Omega^+|k)$ cost is needed only before the line-search procedure rather than at each step of checking an α value.

Trust Region Methods Trust region method is an alternative technique of line search to guarantee asymptotic convergence, which is also widely used in many machine learning applications (Lin, Weng, and Keerthi 2008). At the t-th iteration of a trust region method, we have the current iterate $\tilde{\boldsymbol{w}}^t$, a size Δ^t of trust region, and a quadratic model $q_t(\tilde{\boldsymbol{s}})$ as the approximation of the value $g(\tilde{\boldsymbol{w}}^t+\tilde{\boldsymbol{s}})-g(\tilde{\boldsymbol{w}}^t)$. Two common choices for $q_t(\tilde{\boldsymbol{s}})$ are the first-order approximation

$$q_t(\tilde{\boldsymbol{s}})=\nabla g(\tilde{\boldsymbol{w}}^t)^\top\tilde{\boldsymbol{s}}+\frac{1}{2}\tilde{\boldsymbol{s}}^\top\tilde{\boldsymbol{s}} \tag{1.14}$$

and the second-order approximation

$$q_t(\tilde{\boldsymbol{s}})=\nabla g(\tilde{\boldsymbol{w}}^t)^\top\tilde{\boldsymbol{s}}+\frac{1}{2}\tilde{\boldsymbol{s}}^\top\nabla^2 g(\tilde{\boldsymbol{w}}^t)\tilde{\boldsymbol{s}}. \tag{1.15}$$

Next we find $\tilde{\boldsymbol{s}}^t$ to approximately solve

$$\min_{\|\tilde{\boldsymbol{s}}\|\le\Delta^t} q_t(\tilde{\boldsymbol{s}}). \tag{1.16}$$

If (1.16) is the choice, we can apply conjugate gradient to obtain $\tilde{\boldsymbol{s}}^t$ with our efficient Hessian-vector product procedure in Algorithm 3. In Algorithm 4, we present a detailed conjugate gradient procedure to solve (1.16). See Lin, Weng, and Keerthi (2008) for more details about the convergence property and the implementation issues of Algorithm 4.

We then update $\tilde{\boldsymbol{w}}^t$ and Δ^t as follows:

$$\tilde{\boldsymbol{w}}^{t+1}=\begin{cases}\tilde{\boldsymbol{w}}^t+\tilde{\boldsymbol{s}} & \text{if } \rho^t>\eta_0,\\ \tilde{\boldsymbol{w}}^t & \text{otherwise,}\end{cases} \tag{1.17}$$

$$\Delta^{t+1}\in\begin{cases}[\sigma_1\min(\|\tilde{\boldsymbol{s}}^t\|,\Delta^t),\sigma_2\Delta^t] & \text{if } \rho^t\le\eta_1,\\ [\sigma_1\Delta^t,\sigma_3\Delta^t] & \text{if } \rho^t\in(\eta_1,\eta_2),\\ [\Delta^t,\sigma_3\Delta^t] & \text{if } \rho^t\ge\eta_2,\end{cases} \tag{1.18}$$

where ρ^t is the ratio of the actual reduction in the function to the predicted reduction in the quadratic model:

$$\rho^t = \frac{g(\tilde{\boldsymbol{w}}^t + \tilde{\boldsymbol{s}}^t) - g(\tilde{\boldsymbol{w}}^t)}{q_t(\tilde{\boldsymbol{s}}^t)},$$

and η_0, η_1, η_2, and $\sigma_1, \sigma_2, \sigma_3$ are pre-specified positive constants such that $\eta_1 < \eta_2 < 1$ and $\sigma_1 < \sigma_2 < 1 < \sigma_3$. The idea behind a trust region method is that if the ratio ρ^t is large enough, we can accept the current step $\tilde{\boldsymbol{s}}^t$ and enlarge the size of trust region Δ^t; otherwise, we reject the current step and shrink the size of trust region. More details can be found in Lin, Weng, and Keerthi (2008).

We can see that the computation of the ratio ρ^t requires a function value evaluation at $g(\tilde{\boldsymbol{w}}^t + \tilde{\boldsymbol{s}}^t)$ and $q_t(\tilde{\boldsymbol{s}}^t)$. To compute $g(\tilde{\boldsymbol{w}}^t + \tilde{\boldsymbol{s}}^t)$, we can apply Algorithm 1. To compute $q_t(\tilde{\boldsymbol{s}}^t)$, we can utilize our proposed algorithms as subroutines to perform a straightforward computation. For example, assume the second-order approximation in (1.15) is considered and the conjugate gradient described in Algorithm 4 is applied to solve (1.16). In the procedure in Algorithm 4, $\nabla g(\tilde{\boldsymbol{w}}^t)$ is required as an input. Further, we have the final residual vector

$$\boldsymbol{r}^{i^*} = -\nabla q_t(\bar{\boldsymbol{s}}^{i^*}) = -\nabla g(\tilde{\boldsymbol{w}}^t) - \nabla^2 g(\tilde{\boldsymbol{w}}^t)\bar{\boldsymbol{s}}^{i^*}$$

available when Algorithm 4 terminates, where i^* is the number of CG iterations performed, and $\tilde{\boldsymbol{s}}^t = \bar{\boldsymbol{s}}^{i^*}$. Then, we can compute

$$q_t(\tilde{\boldsymbol{s}}^t) = -\frac{1}{2}\Big(\big(-\nabla g(\tilde{\boldsymbol{w}}^t)\big)^\top \tilde{\boldsymbol{s}}^t - \big(\tilde{\boldsymbol{s}}^t\big)^\top \boldsymbol{r}^{i^*}\Big)$$

in $O(dk)$ time.

Supp-1.5 Complete Optimization Procedures

With the efficient procedures proposed in Section 3, we are able to implement many optimization algorithms to solve (7) such as gradient descent with line search, nonlinear conjugate gradient, truncated Newton methods with line search, and trust region Newton methods. To illustrate how Algorithms 1-3 are used in practice, in Algorithm 5, we use a trust region Newton method (TRON) (Lin, Weng, and Keerthi 2008) as an example to show the complete optimization procedure.

We give another Algorithm 6 to demonstrate the use of line search in a gradient descent method for solving (7). Note that we use a Newton direction in Algorithm 5 and a negative gradient direction in Algorithm 6, respectively, though in either algorithm any descent direction can be considered.

Algorithm 4 Conjugate gradient procedure to approximately solve the trust-region subproblem (1.16).

Input: $\nabla g(\tilde{\boldsymbol{w}}^t)$, a relative stopping parameter $\xi < 1$, and a size of trust region $\Delta^t > 0$

Output: $\tilde{\boldsymbol{s}}^t = \bar{\boldsymbol{s}}^{i^*}$ and $\boldsymbol{r}^{i^*} = -\nabla g(\tilde{\boldsymbol{w}}^t) - \nabla^2 g(\tilde{\boldsymbol{w}}^t)\tilde{\boldsymbol{s}}^t$

// Initialization
$\bar{\boldsymbol{s}}^0 = \boldsymbol{0}$ *// initial point*
$\boldsymbol{r}^0 = -\nabla g(\tilde{\boldsymbol{w}}^t)$ *// initial residual*
$\boldsymbol{d}^0 = \boldsymbol{r}^0$ *// initial direction*
for $i = 0, 1, 2, 3, \ldots$ (CG iterations) **do**
  if $\|\boldsymbol{r}^i\| \leq \xi\|\boldsymbol{r}^0\|$ **then**
    Stop the **for** loop and return $\tilde{\boldsymbol{s}}^t = \bar{\boldsymbol{s}}^i$ and $\boldsymbol{r}^i$
  Compute and store $\nabla^2 g(\tilde{\boldsymbol{w}}^t)\boldsymbol{d}^i$ by Algorithm 3
  $\alpha^i = \|\boldsymbol{r}^i\|^2 / \langle \boldsymbol{d}^i, \nabla^2 g(\tilde{\boldsymbol{w}}^t)\boldsymbol{d}^i \rangle$
  $\bar{\boldsymbol{s}}^{i+1} = \bar{\boldsymbol{s}}^i + \alpha^i \boldsymbol{d}^i$
  if $\|\bar{\boldsymbol{s}}^{i+1}\| \geq \Delta^t$ **then**
    Compute τ such that $\|\bar{\boldsymbol{s}}^i + \tau \boldsymbol{d}^i\| = \Delta^t$
    $\bar{\boldsymbol{s}}^{i+1} = \bar{\boldsymbol{s}}^i + \tau \boldsymbol{d}^i$
    $\boldsymbol{r}^{i+1} = \boldsymbol{r}^i - \tau \nabla^2 g(\tilde{\boldsymbol{w}}^t)\boldsymbol{d}^i$
    Stop the **for** loop and return $\tilde{\boldsymbol{s}}^t = \bar{\boldsymbol{s}}^{i+1}$ and $\boldsymbol{r}^{i+1}$
  $\boldsymbol{r}^{i+1} = \boldsymbol{r}^i - \alpha^i \nabla^2 g(\tilde{\boldsymbol{w}}^t)\boldsymbol{d}^i$
  $\beta^i = \|\boldsymbol{r}^{i+1}\|^2 / \|\boldsymbol{r}^i\|^2$
  $\boldsymbol{d}^{i+1} = \boldsymbol{r}^{i+1} + \beta^i \boldsymbol{d}^i$

Algorithm 5 A trust-region Newton method for (7).

Input: feature matrix $X = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_m]^\top \in \mathbb{R}^{m\times d}$, label matrix $Y = [\boldsymbol{y}_1, \dots, \boldsymbol{y}_m]^\top \in \mathbb{R}^{m\times n}$, $H \in \mathbb{R}^{n\times k}$, and parameters k, λ_w, λ_g, $\boldsymbol{p}$, $\boldsymbol{q}$, $\bar{a}$, and an initial size $\Delta > 0$ of trust region. Choose positive constants $\eta_0, \eta_1 < \eta_2 < 1$ and $\sigma_1 < \sigma_2 < 1 < \sigma_3$.

$M = H^\top \operatorname{diag}(\boldsymbol{q}) H$ $\cdots O(nk^2)$
$\boldsymbol{d} = X^\top \boldsymbol{p}$ $\cdots O(\operatorname{nnz}(X))$
$\boldsymbol{k} = H^\top \boldsymbol{q}$ $\cdots O(nk)$
Compute $\boldsymbol{p}^\top \mathbf{1}_m$ and $\boldsymbol{q}^\top \mathbf{1}_n$ $\cdots O(m+n)$
$B = XW$, where $B = [\cdots, \boldsymbol{b}_i, \cdots]^\top$ $\cdots O(\operatorname{nnz}(X)k)$
$\hat{M} = \operatorname{diag}(\boldsymbol{p}) B M$ $\cdots O(mk^2)$
$\delta = \boldsymbol{d}^\top W \boldsymbol{k}$ $\cdots O(dk)$
$\hat{L} = LB$ $\cdots O(\operatorname{nnz}(L)k)$
$L^+ = \sum_{(i,j)\in\Omega^+} \ell_{ij}(Y_{ij}, \boldsymbol{b}_i^\top \boldsymbol{h}_j)$ $O(|\Omega^+|k)$
$L^- = \bar{a}^2(\boldsymbol{p}^\top \mathbf{1}_m)(\boldsymbol{q}^\top \mathbf{1}_n) + \left\langle B, \hat{M}\right\rangle - 2\bar{a}\delta$ $\cdots O(mk)$
obj $= L^+ + L^- + \lambda_w\left(\|W\|_F^2 + \lambda_g \left\langle B, \hat{L}\right\rangle\right)$
$\cdots O(dk + mk)$
// Compute gradient
$D_{ij}^+ = \ell_{ij}^{+\prime}(Y_{ij}, \boldsymbol{b}_i^\top \boldsymbol{h}_j), \forall (i,j) \in \Omega^+$ $\cdots O(|\Omega^+|k)$
$G = X^\top(D^+ H + 2\hat{M})$ $\cdots O(|\Omega^+|k + \operatorname{nnz}(X)k + mk^2)$
$G = G - 2\bar{a}\boldsymbol{d}\boldsymbol{k}^\top + 2\lambda_w\lambda_g X^\top \hat{L}$
$\cdots O(dk + \operatorname{nnz}(X)k)$
// Pre-processing for Hessian-vector products
$\ell_{ij}^{+\prime\prime} = \ell_{ij}^{+\prime\prime}(Y_{ij}, \boldsymbol{b}_i^\top \boldsymbol{h}_j), \forall (i,j) \in \Omega^+$ $\cdots O(|\Omega^+|k)$
for $t = 1, 2, 3, \dots$ **do**
  // Solve (1.16) *approximately*
  Get $\tilde{\boldsymbol{s}} = \operatorname{vec}(S)$ and $\boldsymbol{r}^{i^*} = \operatorname{vec}(R)$ by Algorithm 4 with Δ
  // Compute $q_t(\tilde{\boldsymbol{s}})$
  $q_t(\tilde{\boldsymbol{s}}) = -\frac{1}{2}(-\langle G, S\rangle - \langle R, S\rangle)$ $\cdots O(dk)$
  // Compute $g(\tilde{\boldsymbol{w}}^t + \tilde{\boldsymbol{s}})$
  $W_{\text{new}} = W + S$ $\cdots O(dk)$
  $B_{\text{new}} = XW_{\text{new}}$ $\cdots O(\operatorname{nnz}(X)k)$
  $\hat{M} = \operatorname{diag}(\boldsymbol{p}) B_{\text{new}} M$ $O(mk^2)$
  $\delta = \boldsymbol{d}^\top W_{\text{new}} \boldsymbol{k}$ $\cdots O(dk)$
  $\hat{L} = LB_{\text{new}}$ $\cdots O(\operatorname{nnz}(L)k)$
  $L^+ = \sum_{(i,j)\in\Omega^+} \ell_{ij}(Y_{ij}, (\boldsymbol{w}_{\text{new}})_i^\top \boldsymbol{h}_j)$ $\cdots O(|\Omega^+|k)$
  $L^- = \bar{a}^2(\boldsymbol{p}^\top \mathbf{1}_m)(\boldsymbol{q}^\top \mathbf{1}_n) + \left\langle B_{\text{new}}, \hat{M}\right\rangle - 2\bar{a}\delta$
  $\cdots O(mk)$
  $\text{obj}_{\text{new}} = L^+ + L^- + \lambda_w\left(\|W_{\text{new}}\|_F^2 + \lambda_g \left\langle B_{\text{new}}, \hat{L}\right\rangle\right)$
  $\cdots O(mk + dk)$
  $\rho = (\text{obj}_{\text{new}} - \text{obj})/q_t(\tilde{\boldsymbol{s}})$ $\cdots O(1)$
  Update Δ based on ρ and (1.18) $\cdots O(1)$
  if $\rho > \eta_0$ **then**
    $\text{obj} = \text{obj}_{\text{new}}$ $\cdots O(1)$
    $W = W_{\text{new}}$ $\cdots O(dk)$
    $B = B_{\text{new}}$ $\cdots O(mk)$
    // Compute gradient
    $D_{ij}^+ = \ell_{ij}^{+\prime}(Y_{ij}, \boldsymbol{b}_i^\top \boldsymbol{h}_j), \forall (i,j) \in \Omega^+$ $\cdots O(|\Omega^+|k)$
    $G = X^\top(D^+ H + 2\hat{M})$
    $\cdots O(|\Omega^+|k + \operatorname{nnz}(X)k + mk^2)$
    $G = G - 2\bar{a}\boldsymbol{d}\boldsymbol{k}^\top + 2\lambda_w\lambda_g X^\top \hat{L}$
    $\cdots O(dk + \operatorname{nnz}(X)k)$
    // Pre-processing for Hessian-vector products
    $\ell_{ij}^{+\prime\prime} = \ell_{ij}^{+\prime\prime}(Y_{ij}, \boldsymbol{b}_i^\top \boldsymbol{h}_j), \forall (i,j) \in \Omega^+$ $\cdots O(|\Omega^+|k)$

Algorithm 6 A gradient-descent method with backtracking line search to minimize (7).

Input: feature matrix $X = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_m]^\top \in \mathbb{R}^{m\times d}$, label matrix $Y = [\boldsymbol{y}_1, \dots, \boldsymbol{y}_m]^\top \in \mathbb{R}^{m\times n}$, $H \in \mathbb{R}^{n\times k}$, and parameters k, λ_w, λ_g, $\boldsymbol{p}$, $\boldsymbol{q}$, and $\bar{a}$. Choose $\beta < 1$ and $\sigma < 1/2$.

$M = H^\top \operatorname{diag}(\boldsymbol{q}) H$ $\cdots O(nk^2)$
$\boldsymbol{d} = X^\top \boldsymbol{p}$ $\cdots O(\operatorname{nnz}(X))$
$\boldsymbol{k} = H^\top \boldsymbol{q}$ $\cdots O(nk)$
Compute $\boldsymbol{p}^\top \mathbf{1}_m$ and $\boldsymbol{q}^\top \mathbf{1}_n$ $\cdots O(m+n)$
$B = XW$, where $B = [\cdots, \boldsymbol{b}_i, \cdots]^\top$ $\cdots O(mk^2)$
$\hat{M} = \operatorname{diag}(\boldsymbol{p}) B M$ $\cdots O(mk^2)$
$\hat{L} = LB$ $\cdots O(\operatorname{nnz}(L)k)$
$L^+ = \sum_{(i,j)\in\Omega^+} \ell_{ij}(Y_{ij}, \boldsymbol{b}_i^\top \boldsymbol{h}_j)$ $O(|\Omega^+|k)$
for $t = 1, 2, 3, \dots$ **do**
  // Compute gradient
  $D_{ij}^+ = \ell_{ij}^{+\prime}(Y_{ij}, \boldsymbol{b}_i^\top \boldsymbol{h}_j), \forall (i,j) \in \Omega^+$ $\cdots O(|\Omega^+|k)$
  $G = X^\top(D^+ H + 2\hat{M})$ $\cdots O(|\Omega^+|k + \operatorname{nnz}(X)k + mk^2)$
  $G = G - 2\bar{a}\boldsymbol{d}\boldsymbol{k}^\top + 2\lambda_w\lambda_g X^\top \hat{L}$ $\cdots O(dk + \operatorname{nnz}(X)k)$

  // Obtain a descent direction ΔW
  $\Delta W = -G$

  // Perform backtracking line search
  $\Delta B = X\Delta W$ $\cdots O(\operatorname{nnz}(X)k)$
  $\Delta\hat{M} = \operatorname{diag}(\boldsymbol{p})\Delta B M$ $\cdots O(mk^2)$
  $\Delta\delta = \boldsymbol{d}^\top \Delta W \boldsymbol{k}$ $\cdots O(dk)$
  $\Delta\hat{L} = L\Delta B$ $\cdots O(\operatorname{nnz}(L)k)$
  $\text{Val\_1} = \left\langle \Delta B, \hat{M}\right\rangle + \left\langle B, \Delta\hat{M}\right\rangle - 2\bar{a}\Delta\delta$
  $+2\lambda_w\left(\langle W, \Delta W\rangle + \lambda_g\left\langle B, \Delta\hat{L}\right\rangle\right) \cdots O(dk + mk)$
  $\text{Val\_2} = \left\langle \Delta B, \Delta\hat{M}\right\rangle$
  $+\lambda_w\left(\|\Delta W\|_F^2 + \lambda_g\left\langle \Delta B, \Delta\hat{L}\right\rangle\right) \cdots O(dk + mk)$
  $c = \langle G, \Delta W\rangle$ $\cdots O(dk)$
  for $\alpha = 1, \beta, \beta^2, \dots$ **do**
    $B_{\text{new}} = B + \alpha\Delta B$ $\cdots O(mk)$
    $L_{\text{new}}^+ = \sum_{(i,j)\in\Omega^+} \ell_{ij}(Y_{ij}, \boldsymbol{b}_i^{\text{new}\top} \boldsymbol{h}_j)$ $O(|\Omega^+|k)$
    if $-L^+ + L_{\text{new}}^+ + \alpha\text{Val\_1} + \alpha^2\text{Val\_2} < \alpha\sigma c$ **then**
      $W = W + \alpha\Delta W$ $\cdots O(dk)$
      $B = B_{\text{new}}$ $\cdots O(mk)$
      $\hat{M} = \hat{M} + \alpha\Delta\hat{M}$ $\cdots O(mk)$
      $\hat{L} = \hat{L} + \alpha\Delta\hat{L}$ $\cdots O(mk)$
      $L^+ = L_{\text{new}}^+$ $\cdots O(1)$
      Break the line-search **for**-loop

Supp-2 Experimental Details

Supp-2.1 Evaluation Criteria and Data Sources

We used Precision@k and nDCG@k as evaluation criteria. For a predicted score vector $\hat{\boldsymbol{y}} \in \mathbb{R}^L$ and the true label vector $\boldsymbol{y} \in \{0,1\}^L$, Precision@$k$ is defined as

$$p@k \equiv \frac{1}{k} \sum_{l \in \text{toprank}_k(\hat{\boldsymbol{y}})} y_l,$$

and nDCG@k is defined as

$$\text{DCG@}k \equiv \sum_{l \in \text{toprank}_k(\hat{\boldsymbol{y}})} \frac{y_l}{\log(l+1)},$$

$$\text{nDCG@}k \equiv \frac{\text{DCG@}k}{\sum_{l=1}^{\min(k, \|\boldsymbol{y}\|_0)} \frac{1}{\log(l+1)}}.$$

For multi-label learning, the data sets are downloaded from *Mulan: A Java Library for Multi-Label Learning*[3] and *The Extreme Classification Repository*.[4] To avoid the $0/0$ situation in the calculation of nDCG, data instances in the test set without any active labels (i.e., $\|\boldsymbol{y}\|_0 = 0$) are removed. For recommender systems with graph information, the data sources have been mentioned in the main text.

Supp-2.2 Parameter Selection

To select a suitable set of parameters, we hold out 1/5 of the training instances for multi-label problems. Similarly, we hold out 1/5 of the observed entries for graph structured one-class MF. The chosen parameters are those that give the highest validation performance on the hold out set. Since the best parameter for different measures ($p@1$ to $p@5$) may slightly vary, we used the one corresponding to the highest $p@5$. However, if the best parameter for $p@5$ has a very bad performance for other measures on the hold out set, we will choose the second best parameter for $p@5$.

For our methods, we fix $\bar{a}$ in (3) to -1 because of the following reasons. First, for LR-wLR, we have

$$\ell^+_{ij}(Y_{ij}, \tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{x}}_{ij}) = \ell^-_{ij}(Y_{ij}, \tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{x}}_{ij})$$
$$= \log\Big(1 + e^{-Y_{ij} \tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{x}}_{ij}}\Big),$$

where $Y_{ij} = \pm 1$. Therefore, for $(i,j) \in \Omega^-$, its $Y_{ij} = \bar{a}$ should be -1. For other formulations such as LR-wSQ, we have

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

The logistic loss $\ell^+(Y_{ij}, \tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{x}}_{ij})$ intends to have

$$(i,j) \in \Omega^+ \quad \text{if } \tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{x}}_{ij} \geq 0,$$

but for the squared loss $\ell^-(Y_{ij}, \tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{x}}_{ij})$, it intends to have

$$(i,j) \in \Omega^- \quad \text{if } \tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{x}}_{ij} \approx 0.$$

Clearly, a conflict occurs when $\tilde{\boldsymbol{w}}^\top \tilde{\boldsymbol{x}}_{ij} > 0$ but ≈ 0. On the other hand, for LEML (Yu et al. 2014), which is SQ-SQ, we use $\bar{a} = 0$ in order be consistent with their experiment setting. For our methods that include a weighted loss (SQ-wSQ, LR-wSQ and LR-wLR), $p_i q_j$ in assumption (3) is set to a constant ρ. We perform a grid search for $\log_2(\rho) \in \{-9, -7, -5, -3, -1, 0\}$. For Subsampled approaches, we fix $\rho = 1$ and perform the grid search on the ratio $|\Omega^-|/|\Omega^+| \in \{0.25, 0.5, 1, 2\}$.

For regularization parameters λ_w and λ_h, we set $\lambda_w = \lambda_h$ and perform a grid search on $\log_2(\lambda_w) \in \{-6, -4, -2, 0, 2, 4, 6\}$. For problems without graph information, we set $\lambda_g = 0$. For problems with graph information, λ_g is non-zero and is obtained by a grid search on $\log_2(\lambda_g \lambda_w) \in \{-6, -3, 0, 3, 6, 8, 10\}$. We consider the same setting for LEML.

For the kernel Nyström method in the last experiment, we use the same regularization and weight-loss parameters selected earlier for the setting without the Nyström method. For the Gaussian kernel $K(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\gamma \|\boldsymbol{x} - \boldsymbol{y}\|^2)$, we check $\gamma \in \{0.5/\sigma^2, 1/\sigma^2, 2/\sigma^2\}$, where σ^2 is the variance of all feature values in the entire training data set. We also conduct a grid search on the number of columns using $D \in \{500, 1000\}$.

For all one-class MF formulations, we run 15 alternating iterations in our validation process and obtain the validation performance at each iteration. We then use the iteration index yielding the best validation performance as the number of iterations to be run in training a model for predicting the test set. Note that the held-out validation set is included so we use the whole training set for generating the final model.

For the non-linear method SLEEC, since it includes 8 hyper-parameters, a grid search on this high dimensional space is not easy. Therefore we first fix the number of clusters to 1. Then the number of learners also becomes 1 because the use of several learners is due to the randomness in clustering. Next we conduct a grid search on the two most sensitive parameters: the k in kNN and the number of neighbors considered during the singular value projection method as suggested by the paper. For all other parameters we use the suggested values given by the authors.

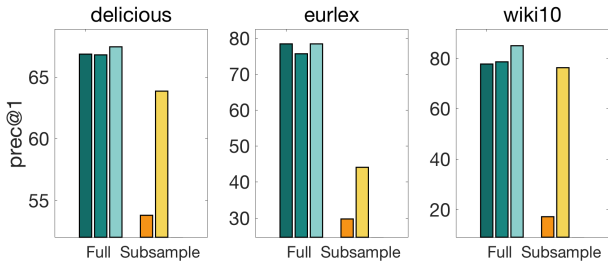
Supp-2.3 More Experimental Results

The results are shown in both tables and figures.

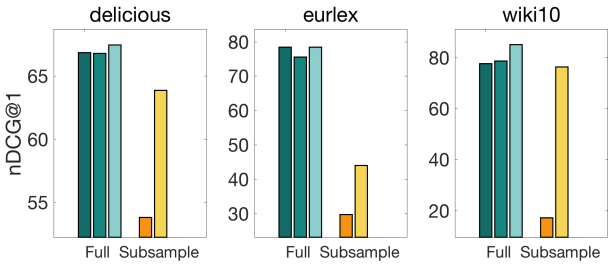
- In Figure Supp-1, we show the results of Full versus Subsampled for multi-label learning in bar charts.
- In Figure Supp-2 and Figure Supp-3, we show the results of various loss functions for three types of one-class MF problems in bar charts.
- In Figure Supp-4, we show the results of the comparison of non-linear multi-label classifiers in bar charts.
- In Table Supp-1, we show the detailed results of recommender systems with and without graph information in terms of precision@1 to precisions@5 and nDCG@1 to nDCG@5.
- In Table Supp-2, we show the detailed results of the comparison among non-linear multi-label classifiers in terms of precision@1 to precisions@5 and nDCG@1 to nDCG@5.
- In Table Supp-3, we show the detailed results of the multi-label learning in terms of precision@1 to precisions@5 and nDCG@1 to nDCG@5.

[3] http://mulan.sourceforge.net/datasets-mlc.html

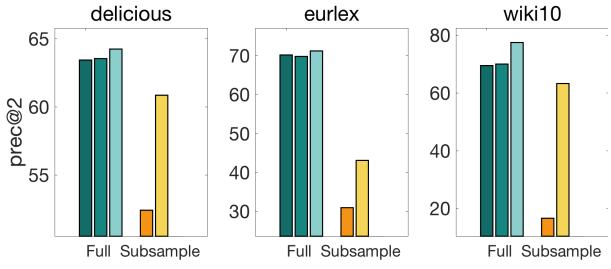
[4] https://manikvarma.github.io/downloads/XC/XMLRepository.html



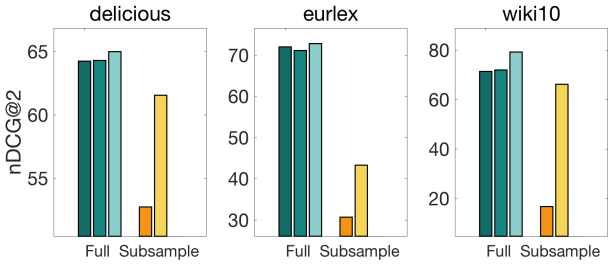
(a) Prediction Performance (precision).



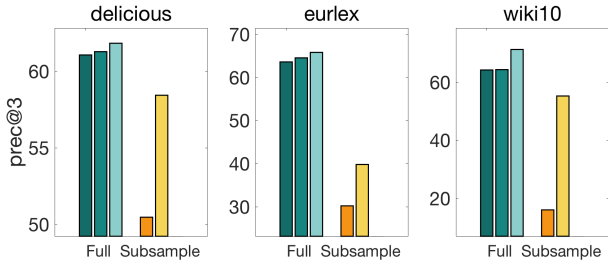
(b) Prediction Performance (nDCG).



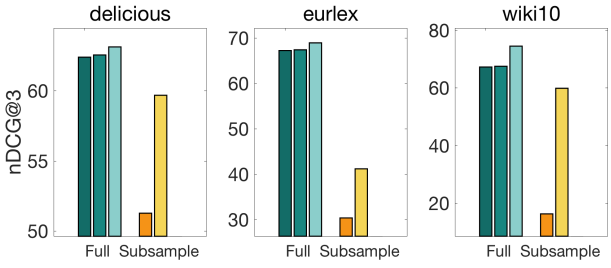
(c) Prediction Performance (precision).



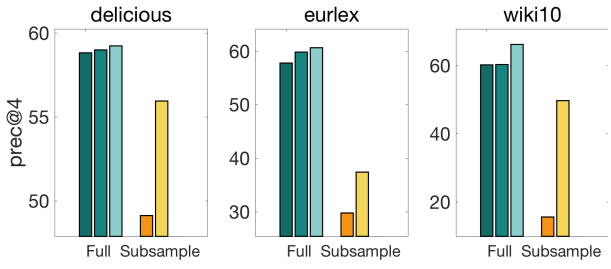
(d) Prediction Performance (nDCG).



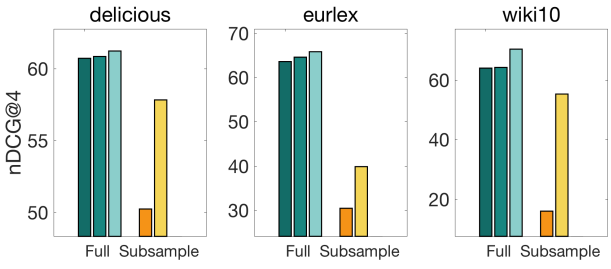
(e) Prediction Performance (precision).



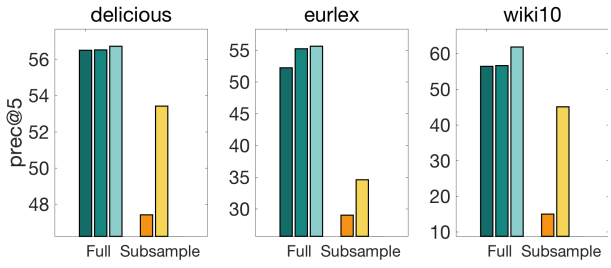
(f) Prediction Performance (nDCG).



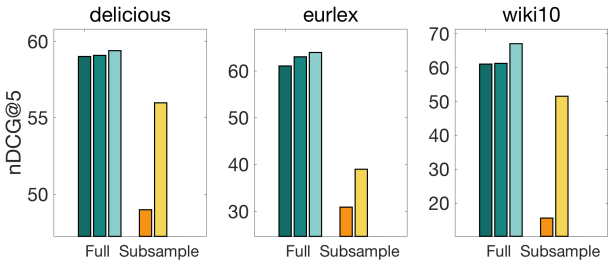
(g) Prediction Performance (precision).



(h) Prediction Performance (nDCG).

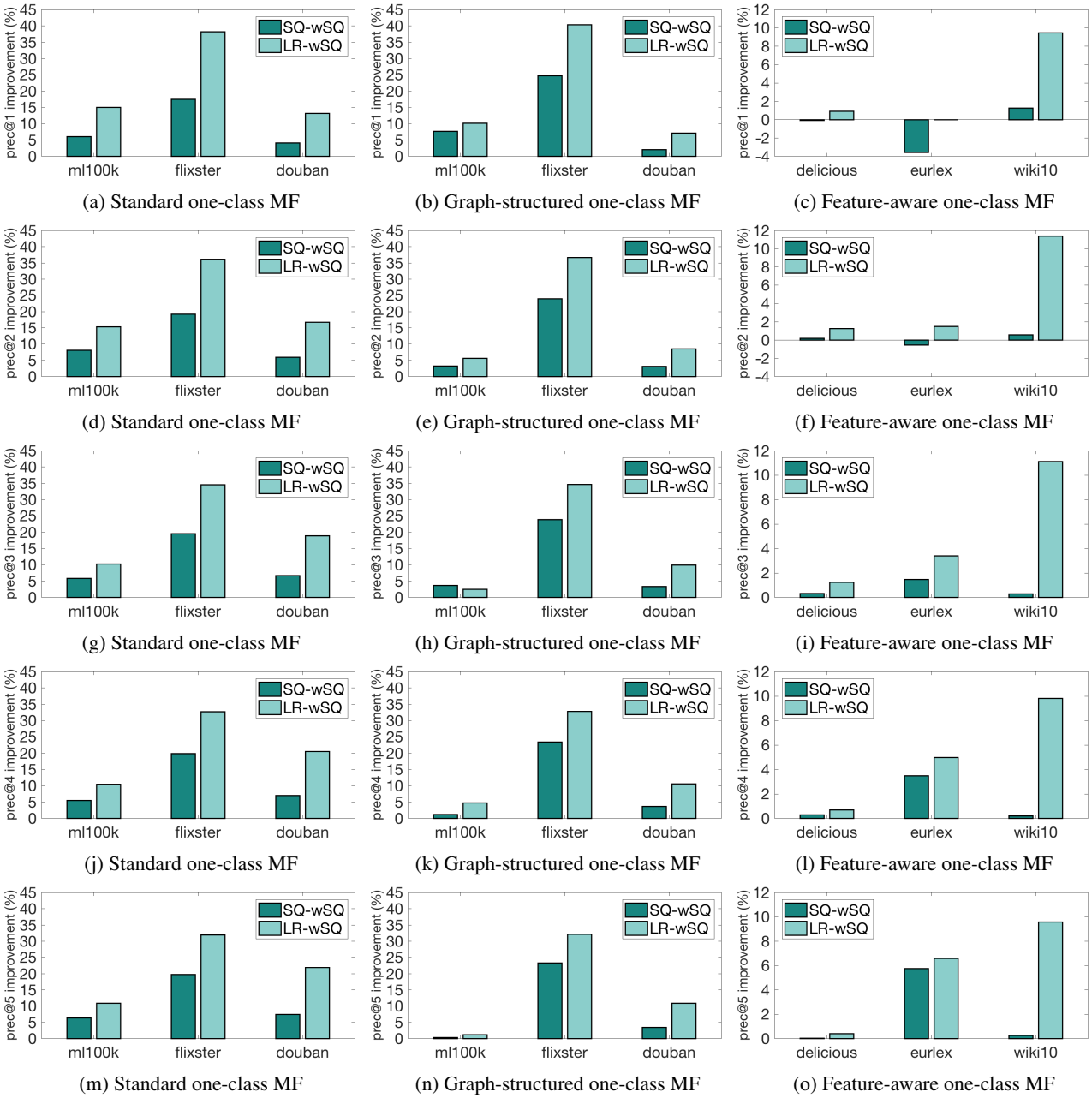


(i) Prediction Performance (precision).



(j) Prediction Performance (nDCG).

Figure Supp-1: Full v.s. Subsampled on multi-label learning with various loss functions. See Figure 1 for the legend information.



(a) Standard one-class MF (b) Graph-structured one-class MF (c) Feature-aware one-class MF

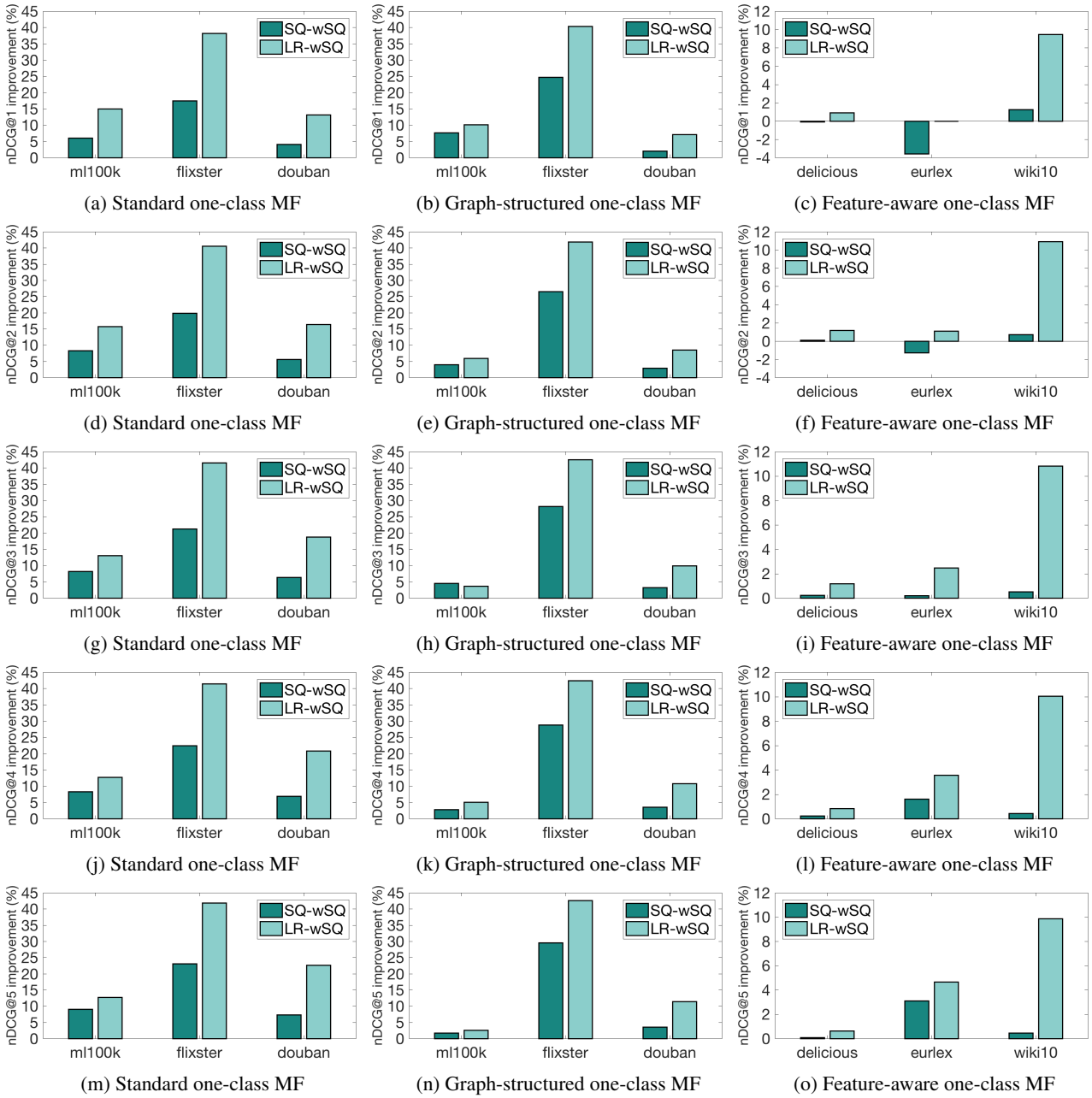
(d) Standard one-class MF (e) Graph-structured one-class MF (f) Feature-aware one-class MF

(g) Standard one-class MF (h) Graph-structured one-class MF (i) Feature-aware one-class MF

(j) Standard one-class MF (k) Graph-structured one-class MF (l) Feature-aware one-class MF

(m) Standard one-class MF (n) Graph-structured one-class MF (o) Feature-aware one-class MF

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



(a) Standard one-class MF (b) Graph-structured one-class MF (c) Feature-aware one-class MF

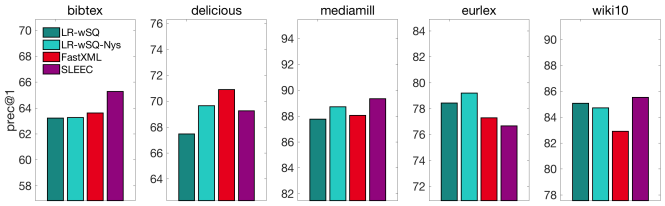
(d) Standard one-class MF (e) Graph-structured one-class MF (f) Feature-aware one-class MF

(g) Standard one-class MF (h) Graph-structured one-class MF (i) Feature-aware one-class MF

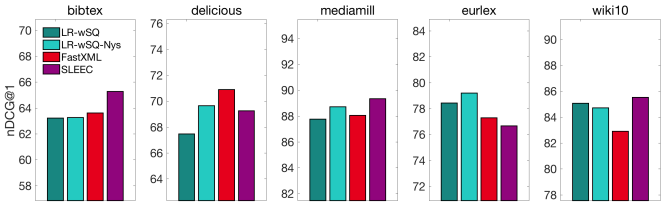
(j) Standard one-class MF (k) Graph-structured one-class MF (l) Feature-aware one-class MF

(m) Standard one-class MF (n) Graph-structured one-class MF (o) Feature-aware one-class MF

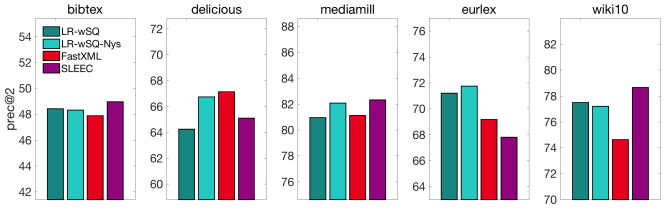
Figure Supp-3: Comparison on various loss functions. Y-axis is the improvement of nDCG@k over the SQ-SQ formulation in percentage.



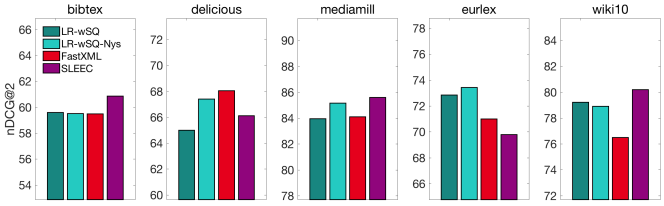
(a) Prediction Performance (precision).



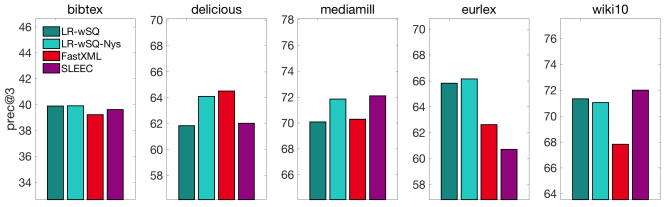
(b) Prediction Performance (nDCG).



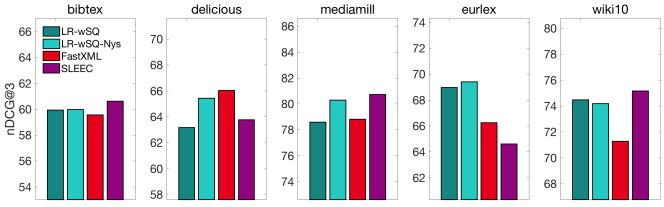
(c) Prediction Performance (precision).



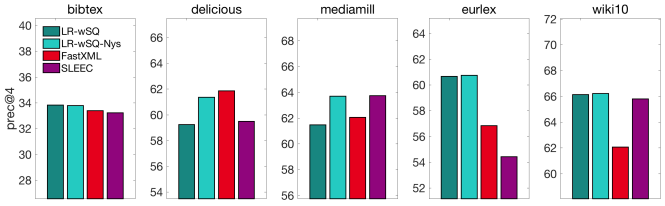
(d) Prediction Performance (nDCG).



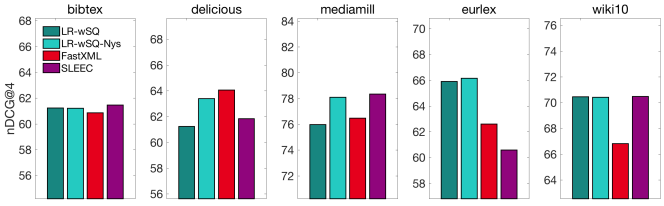
(e) Prediction Performance (precision).



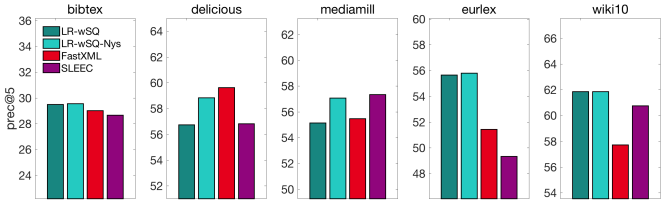
(f) Prediction Performance (nDCG).



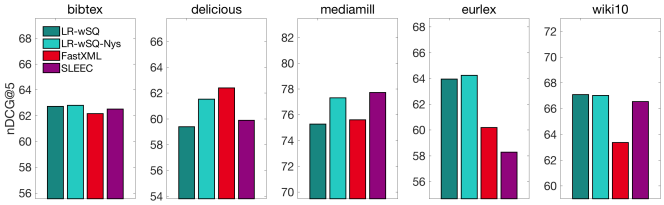
(g) Prediction Performance (precision).



(h) Prediction Performance (nDCG).



(i) Prediction Performance (precision).



(j) Prediction Performance (nDCG).

Figure Supp-4: Comparison on non-linear multi-label classifiers.

Table Supp-1: Comparison on graph structured one-class MF. Loss1-Loss2 denotes the formulation with the Loss1 on entries in Ω^+ and Loss2 on entries in Ω^-. wSQ/wLR denote the weighted square/logistic loss functions respectively. Note that the SQ-SQ formulation for the *graph structured one-class MF* part is equivalent to the formulation considered in Rao et al. (2015).

(a) Precision@k (b) nDCG@k

		time	p@1	p@2	p@3	p@4	p@5	n@1	n@2	n@3	n@4	n@5
					Standard one-class MF						*Standard one-class MF*	
ml100k	SQ-SQ	1.2	26.93	22.20	20.00	18.47	16.95	26.93	23.99	23.00	22.84	22.54
	SQ-wSQ	0.9	28.56	23.99	21.16	19.48	18.04	28.56	25.98	24.88	24.72	24.59
	LR-wSQ	1.8	30.98	25.61	22.04	20.41	18.80	30.98	27.76	26.00	25.75	25.42
					Graph structured one-class MF						*Graph structured one-class MF*	
	SQ-SQ	1.3	28.55	24.62	22.04	20.26	19.05	28.55	26.54	25.57	25.28	25.38
	SQ-wSQ	0.9	30.75	25.43	22.85	20.49	19.12	30.75	27.59	26.73	25.99	25.82
	LR-wSQ	7.0	31.45	26.01	22.58	21.21	19.28	31.45	28.12	26.50	26.55	26.04
					Standard one-class MF						*Standard one-class MF*	
flixster	SQ-SQ	23.7	14.87	12.41	10.96	10.06	9.35	14.87	14.46	14.44	14.64	14.84
	SQ-wSQ	51.2	17.46	14.79	13.09	12.05	11.19	17.46	17.33	17.51	17.94	18.26
	LR-wSQ	161.3	20.54	16.89	14.74	13.34	12.34	20.54	20.33	20.44	20.72	21.04
					Graph structured one-class MF						*Graph structured one-class MF*	
	SQ-SQ	27.4	14.66	12.37	10.97	10.06	9.35	14.66	14.35	14.38	14.59	14.78
	SQ-wSQ	54.0	18.28	15.34	13.60	12.42	11.53	18.28	18.16	18.43	18.80	19.15
	LR-wSQ	180.6	20.58	16.91	14.77	13.36	12.36	20.58	20.36	20.50	20.77	21.08
					Standard one-class MF						*Standard one-class MF*	
douban	SQ-SQ	64.8	17.42	15.07	13.56	12.47	11.62	17.42	15.84	14.97	14.44	14.09
	SQ-wSQ	100.3	18.13	15.97	14.47	13.35	12.48	18.13	16.74	15.93	15.43	15.13
	LR-wSQ	514.3	19.71	17.59	16.12	15.03	14.16	19.71	18.44	17.78	17.45	17.29
					Graph structured one-class MF						*Graph structured one-class MF*	
	SQ-SQ	58.3	18.89	16.61	15.07	13.97	13.11	18.89	17.41	16.60	16.14	15.89
	SQ-wSQ	75.0	19.27	17.12	15.58	14.47	13.56	19.27	17.93	17.14	16.71	16.45
	LR-wSQ	571.4	20.23	18.02	16.57	15.44	14.53	20.23	18.89	18.24	17.88	17.70

Table Supp-2: Comparison of state-of-the-art approaches on multi-label learning.

(a) Precision@k (b) nDCG@k

		time	p@1	p@2	p@3	p@4	p@5	n@1	n@2	n@3	n@4	n@5
bibtex	LR-wSQ	22	63.22	48.43	39.89	33.83	29.50	63.22	59.59	59.93	61.24	62.73
	LR-wSQ-Nys	54	63.26	48.33	39.91	33.80	29.55	63.26	59.53	59.97	61.21	62.81
	FastXML	17	63.62	47.89	39.22	33.39	29.01	63.62	59.49	59.55	60.87	62.16
	SLEEC	249	65.29	48.97	39.63	33.24	28.76	65.29	60.87	60.62	61.46	62.52
delicious	LR-wSQ	23	67.47	64.24	61.85	59.25	56.73	67.47	64.99	63.16	61.23	59.40
	LR-wSQ-Nys	94	69.64	66.74	64.11	61.35	58.84	69.64	67.41	65.43	63.39	61.54
	FastXML	34	70.90	67.14	64.52	61.87	59.63	70.90	68.05	66.04	64.06	62.40
	SLEEC	1,124	69.27	65.10	62.03	59.49	56.82	69.27	66.11	63.78	61.84	59.89
mediamill	LR-wSQ	61	87.77	80.96	70.08	61.48	55.15	87.77	83.96	78.60	75.97	75.28
	LR-wSQ-Nys	144	88.72	82.08	71.85	63.70	57.08	88.72	85.17	80.29	78.10	77.32
	FastXML	256	88.04	81.12	70.29	62.06	55.48	88.04	84.10	78.81	76.48	75.61
	SLEEC	1,764	89.34	82.33	72.09	63.73	57.34	89.34	85.60	80.73	78.36	77.73
eurlex	LR-wSQ	404	78.43	71.21	65.82	60.67	55.64	78.43	72.84	68.97	65.91	63.96
	LR-wSQ-Nys	1,134	79.20	71.75	66.15	60.75	55.80	79.20	73.44	69.41	66.15	64.25
	FastXML	164	77.29	69.17	62.63	56.84	51.44	77.29	71.00	66.25	62.60	60.19
	SLEEC	1,497	76.67	67.80	60.71	54.45	49.34	76.67	69.80	64.61	60.58	58.28
wiki10	LR-wSQ	449	85.07	77.50	71.33	66.14	61.85	85.07	79.21	74.47	70.46	67.10
	LR-wSQ-Nys	666	84.72	77.20	71.05	66.22	61.86	84.72	78.90	74.18	70.41	67.02
	FastXML	744	82.91	74.63	67.82	62.06	57.73	82.91	76.50	71.27	66.83	63.37
	SLEEC	183	85.52	78.65	72.00	65.81	60.75	85.52	80.20	75.16	70.48	66.55

Table Supp-3: Comparison of various Full and Subsampled formulations on multi-label learning. Loss1-Loss2 denotes the formulation with the Loss1 on entries in Ω^+ and Loss2 on entries in Ω^-. wSQ/wLR denote the weighted square/logistic loss functions respectively. Note that the Full approach with SQ-SQ is equivalent to the LEML (Yu et al. 2014) formulation.

		(a) Precision@k						(b) nDCG@k				
		time	p@1	p@2	p@3	p@4	p@5	n@1	n@2	n@3	n@4	n@5
bibtex	*the* Subsampled *approach*											
	SQ-SQ	12	46.04	23.16	31.31	27.59	24.63	46.04	45.47	46.71	48.87	50.66
	LR-LR	9	52.92	39.42	32.63	28.27	25.08	52.92	49.17	49.59	51.32	52.94
	the Full *approach*											
	SQ-SQ	13	63.14	47.58	38.77	33.09	28.81	63.14	58.99	58.95	60.34	61.71
	SQ-wSQ	12	63.30	48.35	39.78	33.74	29.37	63.30	59.61	59.96	61.26	62.65
	LR-wSQ	22	63.22	48.43	39.89	33.83	29.50	63.22	59.59	59.93	61.24	62.73
	LR-wLR	85	61.91	46.66	38.21	32.66	28.43	61.91	57.75	57.91	59.40	60.76
delicious	*the* Subsampled *approach*											
	SQ-SQ	8	53.80	52.42	50.46	49.13	47.41	53.80	52.73	51.28	50.26	49.02
	LR-LR	19	63.89	60.86	58.44	55.95	53.43	63.89	61.56	59.71	57.84	55.99
	the Full *approach*											
	SQ-SQ	3	66.88	63.44	61.09	58.84	56.51	66.88	64.24	62.42	60.72	59.03
	SQ-wSQ	8	66.81	63.55	61.29	59.01	56.52	66.81	64.31	62.57	60.86	59.07
	LR-wSQ	23	67.47	64.24	61.85	59.25	56.73	67.47	64.99	63.16	61.23	59.40
	LR-wLR	446	67.91	65.04	62.27	59.81	57.27	67.91	65.74	63.68	61.85	59.99
mediamill	*the* Subsampled *approach*											
	SQ-SQ	83	83.20	79.21	69.62	61.66	55.50	83.20	81.59	77.17	75.10	74.60
	LR-LR	93	87.62	80.72	69.87	61.37	55.08	87.62	83.74	78.39	75.82	75.15
	the Full *approach*											
	SQ-SQ	8	87.70	81.05	69.89	61.23	54.72	87.70	83.99	78.43	75.75	74.88
	SQ-wSQ	41	84.85	79.82	69.92	61.79	55.47	84.85	82.41	77.75	75.57	74.92
	LR-wSQ	61	87.77	80.96	70.08	61.48	55.15	87.77	83.96	78.60	75.97	75.28
	LR-wLR	437	87.78	81.17	70.38	61.85	55.33	87.78	84.17	78.87	76.33	75.50
eurlex	*the* Subsampled *approach*											
	SQ-SQ	93	29.75	30.94	30.23	29.75	29.04	29.75	30.67	30.38	30.47	30.92
	LR-LR	166	44.08	43.09	39.83	37.39	34.58	44.08	43.32	41.17	39.90	38.99
	the Full *approach*											
	SQ-SQ	335	78.43	70.18	63.67	57.80	52.21	78.43	72.04	67.30	63.63	61.12
	SQ-wSQ	672	75.63	69.79	64.60	59.82	55.21	75.63	71.11	67.45	64.66	63.02
	LR-wSQ	404	78.43	71.21	65.82	60.67	55.64	78.43	72.84	68.97	65.91	63.96
wiki10	*the* Subsampled *approach*											
	SQ-SQ	121	17.23	16.58	16.14	15.59	15.04	17.23	16.73	16.38	15.98	15.56
	LR-LR	764	76.30	63.29	55.28	49.72	45.09	76.30	66.23	59.91	55.40	51.62
	the Full *approach*											
	SQ-SQ	247	77.71	69.58	64.20	60.23	56.45	77.71	71.42	67.20	64.04	61.07
	SQ-wSQ	872	78.69	69.97	64.39	60.35	56.59	78.69	71.94	67.55	64.31	61.34
	LR-wSQ	449	85.07	77.50	71.33	66.14	61.85	85.07	79.21	74.47	70.46	67.10