Preference Completion: Large-scale Collaborative Ranking from Pairwise Comparisons

Dohyung Park    Joe Neeman    Jin Zhang
Sujay Sanghavi   Inderjit S. Dhillon

The University of Texas at Austin
The Problem

Given: for each user, a small number of pairwise comparisons:

“User $i$ prefers item $j_1$ over $j_2$”

To find: personalized preference order for each user.

Alice ($x_1^*$) > >

Bob ($x_2^*$) > >

Charlie ($x_3^*$) > >

::
The Problem

Given: for each user, a small number of pairwise comparisons:

“User \(i\) prefers item \(j_1\) over \(j_2\)”

To find: personalized preference order for each user.

- Many kinds of user input can be turned into pairwise comparisons:
  - click/no-click: Each “click” is preferred to a (randomly chosen) “no-click”
  - one-of-many: chosen item preferred to others presented
  - numerical ratings: for each user, higher rated item preferred to lower rated one.

- Pairwise comparisons less subjective than numerical ratings
A Classic Model in Ranking

**Bradley-Terry-Luce (BTL) model:** for a single user setting
- Assume a ground-truth score vector $x^* \in \mathbb{R}^d$.
- Governs pairwise preferences:
  $$Pr(j_1 > j_2) = \frac{1}{1 + \exp(-(x_{j_1}^* - x_{j_2}^*))}$$
- Popular for rank aggregation (fitting a single rank order to inconsistent preference data)
Our Approach

- Each user has its own, personal score vector $X_i$: Items with higher score more likely to be preferred by that user.

- Taken together, the vectors form a low-rank matrix:
  for $d_1$ users and $d_2$ items,

  $$X \in \mathbb{R}^{d_1 \times d_2} \text{ and } \text{rank}(X) \leq r \quad (r \ll d_1, d_2)$$

- Low-rank allows for generalization from a very small number of per-user comparisons (similar to the case of matrix completion)
Our contribution

- **Convex ERM**
  - We prove it has nearly optimal sample complexity: each user needs to make only $O(r \log^2 (d_1 + d_2))$ pairwise comparisons

- **Alternating SVM (AltSVM)**
  - A scalable non-convex algorithm for the hinge loss, which we found works best in the practical large-scale settings.
  - Parallel implementation: near-linear speedup with number of cores in shared-memory machine
  - Outperforms existing (rating-based) algorithms both statistically and computationally.
Problem Setting

- $d_1$ users, $d_2$ items

Input
- $\Omega \subseteq [d_1] \times [d_2] \times [d_2]$ : Set of (user, item 1, item 2) triples
- $Y \triangleq \{ Y_{ijk} \in \{+1, -1\} : (i, j, k) \in \Omega \}$ : Pairwise comparisons

Output
- Predicted “score matrix” $X \in \mathbb{R}^{d_1 \times d_2}$

\[ Y_{ijk} = \begin{cases} 
+1 & \text{“user } i \text{ prefers item } j \text{ to item } k” \\
-1 & \text{“user } i \text{ prefers item } k \text{ to item } j” 
\end{cases} \]
Convex ERM

\[
\min_{X \in \mathbb{R}^{d_1 \times d_2}} \sum_{(i,j,k) \in \Omega} \mathcal{L}(Y_{ijk}(X_{ij} - X_{ik}))
\]

subject to \( \|X\|_* \leq \sqrt{\lambda d_1 d_2} \)

- Convex optimization over \( d_1 \times d_2 \) dimensional space
- Parameters: set \( \lambda = O(r) \).
  Reason: If rank(\(X\)) = \(r\) and \(\|X\|_\infty \leq C\), then \(\|X\|_* \leq C \sqrt{rd_1 d_2}\)
- \(\mathcal{L}\): appropriately chosen loss function.
  E.g. logistic for the BTL model. Our results for more general losses.
Convex ERM

Statistical performance: setup

- Each user-item-item triple \((i, j, k)\) is sampled with probability \(p_{ijk}\).
- No user-item pair is sampled too frequently.

\[
\sum_k p_{ijk} \leq \kappa \frac{m}{d_1 d_2} \quad \text{(for fixed } m = \mathbb{E}|\Omega|\text{)}
\]

- Expected risk: with \(\Omega\) and \(Y\)'s chosen as above, for any matrix \(X\),

\[
\mathbb{E}_{\Omega,Y} R(X) := \text{expected value of } \sum_{(i,j,k) \in \Omega} \mathcal{L}(Y_{ijk}(X_{ij} - X_{ik}))
\]
Convex ERM

Theorem

Suppose

- $\mathcal{L}(\cdot) : 1$-Lipschitz
- $\hat{X} : \text{The optimum of the convex program}

Then, in the above setting,

$$
\mathbb{E}_{\Omega,Y} R(\hat{X}) \leq \inf_{\{X: \|X\|_* \leq \sqrt{\lambda d_1 d_2}\}} R(X) + C \kappa \sqrt{\frac{\lambda (d_1 + d_2)}{m}} \log(d_1 + d_2).
$$

The best expected risk

Excess risk bound

$O(r \log^2 d)$ comparisons/user are sufficient if $d_1, d_2 \approx d$. 
Consistency in the Multi-user BTL model

- Assume there is a ground-truth $X^* \in \mathbb{R}^{d_1 \times d_2}$.

$$
\Pr\{ Y_{ijk} = +1 \} = \frac{e^{X^*_{ij} - X^*_{ik}}}{1 + e^{X^*_{ij} - X^*_{ik}}}
$$

- ML estimation: Solving the ERM with $\mathcal{L}(z) = \log(1 + \exp(z)) - z$.

Corollary

Suppose that $Y \sim BTL(X^*)$ where $\|X^*\|_* \leq \sqrt{\lambda d_1 d_2}$. Under the sampling assumption,

$$
\frac{1}{d_1 d_2^2} D(\mathbb{P}_{X^*} \| \mathbb{P}_{\hat{X}}) \leq C_\kappa \sqrt{\frac{\lambda (d_1 + d_2)}{m}} \log(d_1 + d_2).
$$

Can recover the true $X^*$ with $O(r \log^2 d)$ comparisons/user.
Is the $O(r \log^2 d)$ sample complexity good?

Theorem

For any estimator $\hat{X}$ as a function of $\Omega$ and $\mathcal{Y}$, there exists $X^*$ such that\(^a\)

\[
\mathbb{E}_{\Omega,\mathcal{Y}} R(\hat{X}) \geq R(X^*) + c \min \left\{ 1, \sqrt{\frac{\lambda(d_1 + d_2)}{m}} \right\},
\]

with probability at least $\frac{1}{2}$.

\(^a\)Under the assumption $\mathcal{L}'(0) < 0$, $\lambda \geq 1$, and $m \geq d_1 + d_2$

Need at least $O(r)$ comparisons/user.
However, In Practice..

The size of user-item matrices?

- Netflix prize: 480,000 users × 17,000 movies
- Personalization datasets often even larger

Convex optimization needs to train and store $10^{10} \sim 10^{15}$ parameters.
Non-Convex Algorithm

\[
\begin{align*}
\text{minimize} & \quad \mathcal{L}(Y_{ijk} \cdot u_i^\top (v_j - v_k)) \\
\text{subject to} & \quad U \in \mathbb{R}^{d_1 \times r}, V \in \mathbb{R}^{d_2 \times r}
\end{align*}
\]

- Train a factored form \( X = UV^\top \) (\( X_{ij} = u_i^\top v_j \))

Now only \((d_1 + d_2)r\) parameters
Non-Convex Algorithm

\[
\begin{align*}
\text{minimize} & \quad \sum_{(i,j,k) \in \Omega} \mathcal{L}(Y_{ijk} \cdot u_i^T (v_j - v_k)) + \frac{\lambda}{2} (\|U\|_F^2 + \|V\|_F^2) \\
\end{align*}
\]

- Add regularizer to control overfitting
Non-Convex Implementation

- Updating $U$ (while $V$ is fixed): Ranking SVM [Joachims, 2002]
  “Find the personalized weight vector $u_i$ for each user.”

$$\forall i, \quad u_i \leftarrow \arg \min_{u \in \mathbb{R}^r} \frac{\lambda}{2} \|u\|_2^2 + \sum_{j,k:(i,j,k)\in\Omega} \mathcal{L}(Y_{ijk} \cdot u^\top (v_j - v_k))$$

Can be decomposed into $d_1$ independent $r$-dimensional SVMs

- Updating $V$ (while $U$ is fixed) “Embed $d_2$ item vectors into $\mathbb{R}^r$."

$$V \leftarrow \arg \min_{V \in \mathbb{R}^{d_2 \times r}} \left\{ \frac{\lambda}{2} \|V\|_F^2 + \sum_{(i,j,k)\in\Omega} \mathcal{L}(Y_{ijk} \cdot \langle A^{(ijk)}, V \rangle) \right\}$$

Also a SVM! but too large ($d_1 \times r$ dimensional)

- Solution: dual coordinate ascent still $O(r)$
Non-Convex Implementation

Dual Coordinate Descent [Hsieh et al., 2007]

- Dual problem

$$
\min_{\beta \in \mathbb{R}^{|\Omega|}, \beta \geq 0} \frac{1}{2} \left\| \sum_{(i,j,k) \in \Omega} \beta_{ijk} A^{(ijk)} \right\|_F^2 + \frac{1}{\lambda} \sum_{(i,j,k) \in \Omega} L^* \left( -\lambda \beta_{ijk} \right)
$$

- Coordinate descent: Fix all but one variables, and optimize.

$$
\delta^* \leftarrow \arg \min_{\delta \geq -\beta_{ijk}} \frac{1}{2} \left( \| v_j + \delta Y_{ijk} u_i \|_2^2 + \| v_k - \delta Y_{ijk} u_i \|_2^2 \right) + L^* \left( -\lambda (\beta_{ijk} + \delta) \right),
$$

$$
\beta \leftarrow \beta + \delta^*,
$$

$$
\nu_j \leftarrow \nu_j + \delta^* Y_{ijk} u_i,
$$

$$
\nu_k \leftarrow \nu_k - \delta^* Y_{ijk} u_i.
$$

O(r) computation
Alternating SVM (AltSVM)

While not converged do

1. Stochastic dual coordinate descent for $V$.
   - For $t = 1, \ldots, T$,
   - Randomly pick $(i, j, k) \in \Omega$.
   - Do coordinate descent for the dual variable $\beta_{ijk}$.
   - Update $v_j$ and $v_k$.

2. Stochastic dual coordinate descent for $U$.
   - For $t = 1, \ldots, T$,
   - Randomly pick $(i, j, k) \in \Omega$.
   - Do coordinate descent for the dual variable $\alpha_{ijk}$.
   - Update $u_i$.

Decomposability does not matter.

$O(r)$ computation
Alternating SVM (AltSVM)

While not converged do

1. Stochastic dual coordinate descent for $V$.
   - For $t = 1, \ldots, T$,
   - Randomly pick $(i, j, k) \in \Omega$.
   - Do coordinate descent for the dual variable $\beta_{ijk}$.
   - Update $v_j$ and $v_k$.  \hfill $O(r)$ computation

2. Stochastic dual coordinate descent for $U$.
   - For $t = 1, \ldots, T$,
   - Randomly pick $(i, j, k) \in \Omega$.
   - Do coordinate descent for the dual variable $\alpha_{ijk}$.
   - Update $u_i$.  \hfill $O(r)$ computation

Decomposability does not matter.
Parallellization

Each coordinate descent updates at most $2r$ out of $(d_1 + d_2)r$ variables.

- Can apply parallel asynchronous stochastic DCD without locking.\(^1\)

<table>
<thead>
<tr>
<th># cores</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time(seconds)</td>
<td>963.1</td>
<td>691.8</td>
<td>365.1</td>
<td>188.3</td>
<td>111.0</td>
</tr>
<tr>
<td>Speedup</td>
<td>1x</td>
<td>1.4x</td>
<td>2.6x</td>
<td>5.1x</td>
<td>8.7x</td>
</tr>
</tbody>
</table>

Table: Scalability of AltSVM on the binarized MovieLens1m dataset.

---

\(^1\)Hsieh, Yu, and Dhillon, “PASSCoDe: Parallel Asynchronous Stochastic Dual Coordinate Descent,” ICML 2015.
Experiments

We compare our algorithm (with hinge loss) to

- CofiRank [Weimer et al., NIPS’07]
- Local Collaborative Ranking [Lee et al., WWW’14]
- Robust Binary Ranking [Yun et al., NIPS’14]
- SGD : Stochastic Gradient Descent on our non-convex formulation.
- Global ranking : Aggregate all comparisons and provide one ranking.

Datasets

- Binarized MovieLens1m : $6,040 \times 3,900$ movies, 1m ratings
- MovieLens10m : $71,567$ users $\times 10,681$ movies, 10m ratings
- Netflix prize : $480,000$ users $\times 17,000$ movies, 100m ratings
Experimental results - Rating data

- Compared in terms of NDCG@10
- AltSVM takes all non-tying comparisons from the sampled ratings

<table>
<thead>
<tr>
<th>Datasets</th>
<th># ratings/user</th>
<th>AltSVM</th>
<th>SGD</th>
<th>Global</th>
<th>CofiRank</th>
<th>LCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>MovieLens10m</td>
<td>20</td>
<td>0.7059</td>
<td>0.6977</td>
<td>0.7264</td>
<td>0.7076</td>
<td>0.6977</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.7508</td>
<td>0.7452</td>
<td>0.7176</td>
<td>0.6977</td>
<td>0.6940</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.7692</td>
<td>0.7659</td>
<td>0.7101</td>
<td>0.6754</td>
<td>0.6899</td>
</tr>
<tr>
<td>Netflix</td>
<td>20</td>
<td>0.7132</td>
<td>-</td>
<td>0.7605</td>
<td>0.6615</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.7642</td>
<td>-</td>
<td>0.7640</td>
<td>0.6527</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.8007</td>
<td>-</td>
<td>0.7656</td>
<td>0.6385</td>
<td>-</td>
</tr>
</tbody>
</table>

![Graph showing NDCG@10 over time for MovieLens10m with 50 ratings/user and rank 10]
Experimental results - Binary data

- Compared in terms of Precision@K
- AltSVM takes $C$ non-tying comparisons for each user.

<table>
<thead>
<tr>
<th>Precision@</th>
<th>$C = 1000$</th>
<th>$C = 2000$</th>
<th>$C = 5000$</th>
<th>SGD</th>
<th>RobiRank</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.2165</td>
<td>0.2973</td>
<td><strong>0.3635</strong></td>
<td>0.1556</td>
<td>0.3009</td>
</tr>
<tr>
<td>2</td>
<td>0.1965</td>
<td>0.2657</td>
<td><strong>0.3297</strong></td>
<td>0.1498</td>
<td>0.2695</td>
</tr>
<tr>
<td>5</td>
<td>0.1572</td>
<td>0.2097</td>
<td><strong>0.2697</strong></td>
<td>0.1236</td>
<td>0.2300</td>
</tr>
<tr>
<td>10</td>
<td>0.1265</td>
<td>0.1709</td>
<td><strong>0.2223</strong></td>
<td>0.1031</td>
<td>0.1922</td>
</tr>
<tr>
<td>100</td>
<td>0.0526</td>
<td>0.0678</td>
<td><strong>0.0819</strong></td>
<td>0.0441</td>
<td>0.0781</td>
</tr>
</tbody>
</table>
Summary

- Two algorithms for collaborative ranking from pairwise comparisons

- Convex relaxation
  - $O(r \log^2 d)$ sample complexity for arbitrarily small excess risk

- Alternating SVM through Stochastic Dual Coordinate Descent
  - Scalable and outperforming existing algorithms in ranking measures