LCR: Local Collaborative Ranking

Joonseok Lee
with Samy Bengio, Seungyeon Kim, Guy Lebanon, Yoram Singer
Matrix Completion Problem

- Problem: given a partially-observed noisy matrix $M$, we would like to approximately complete it.

- Application: recommendation systems
  - $M_{u,i}$ is rating of item $i$ by user $u$.
  - Naturally sparse: most are unknown.
  - We want to estimate unrated items.

<table>
<thead>
<tr>
<th>Users</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Items</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Low-rank Assumption

- Common practice: low-rank assumption.
  \[ M \approx UV^T \in \mathbb{R}^{n_1 \times n_2}, \quad U \in \mathbb{R}^{n_1 \times r}, \quad V \in \mathbb{R}^{n_2 \times r} \]
  \[ r \ll \min(n_1, n_2) \]

- Incomplete SVD:
  \[ \min_{U,V} \sum_{(u,i) \in A} ([UV^T]_{u,i} - M_{u,i})^2 \]
Ordering Problem

- Motivation: we usually care about relative order of preference, not exact score.
- Order items according to the (partial) preferences of a given user.
- Example: for the following user who rated 4 ratings,

```
5  >  3  >  3  >  4
  >  >  >  >  >  >  >  >  >  >
```
Talk Agenda & Contribution

- **Paired loss functions**
  - How to solve ordering problem?

- **Local Low-Rank Assumption**
  - Why and how to tackle diminishing returns?

- **Algorithm**
  - Should be scalable for big data.

- **Experimental analysis**
  - Two frameworks.
Ordering Function

- Learn an ordering function $f$, such that $f(u, i) > f(u, j)$ if $M_{u,i} > M_{u,j}$.
  - Not necessarily $f(u, i) \approx M_{u,i}$.

- Pair-wise Loss function $L(\Delta M, \Delta f)$

$$E(f) = \sum_u \sum_{(i,j) \in M_u} L(M_{u,i} - M_{u,j}, f(u, i) - f(u, j))$$

- $\Delta M = M_{u,i} - M_{u,j}$: difference of observed ratings.
- $\Delta f = f_{u,i} - f_{u,j}$: difference of estimated ratings.
Pair-wise Loss $L(\Delta M, \Delta f)$

- **Zero-one loss**
  - Assigns (same) positive loss when $\Delta M \Delta f < 0$.
  - Not differentiable.

<table>
<thead>
<tr>
<th></th>
<th>Multiplicative</th>
<th>Additive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log-loss</td>
<td>$\Delta M \log(1 + e^{-\Delta f})$</td>
<td>$\log(1 + e^{\Delta M - \Delta f})$</td>
</tr>
<tr>
<td>Exp-loss</td>
<td>$\Delta M \exp{-\Delta f}$</td>
<td>$\exp{\Delta M - \Delta f}$</td>
</tr>
<tr>
<td>Hinge-loss</td>
<td>$\Delta M [ -\Delta f]_+$</td>
<td>$[\Delta M - \Delta f]_+$</td>
</tr>
</tbody>
</table>
Global Approximation

- With $f(u, i) = [UV^T]_{u,i}$, solve matrix factorization problem with respect to a paired loss $L$.

$$\min_{U,V} \sum_u \sum_{(i,j) \in M_u} L(M_{u,i} - M_{u,j}, [UV^T]_{u,i} - [UV^T]_{u,j})$$

- We model using form, so as to minimize a pair-wise loss.
Diminishing Returns

- Small improvement as capacity increases.
Why diminishing returns?

- Hypotheses
  - **H1**: \( M \) has low rank; it reflects best possible prediction.
  - **H2**: \( M \) has high rank; diminishing returns due to over-fitting, or convergence to a poor local optimum.

- In recommendation systems,
  - **H2** is a realistic assumption.
  - **H1** is unrealistic globally, but it’s realistic **locally**.

- The rating matrix is only **locally low-rank**.
  - Low-rank only with subset of similar users and items.
Local Low-rank Matrix Approx.

[Lee et al, 2013 ICML]
Learning Algorithm

- Run in Parallel:
  - **Step 1**: Select an anchor point.
  - **Step 2**: Calculate user/item weight using kernel smoothing.
  - **Step 3**: Solve a weighted matrix factorization problem.
Evaluation

- **Goal**: Recommend most preferable items based on precise estimation of order of preference.

- **Criteria**
  - **Zero-One Error**: the ratio of correctly ordered test pairs.
  - **Average Precision**: the ratio of preferred items in the list.
  - **NDCG@k**: optimality of the order of recommendation list.

- **Dataset**: MovieLens, EachMovie, Yelp
Data Split

- **Fixed ratio**
  - For each user, 50% of ratings are used for training, rest of them are for testing.
  - More **realistic**: take cold/cool-start users into account.
    → Used to see effects of parameters.

- **Fixed number**
  - Users with more than 20 ratings are considered. 10 ratings are used for training, and rest of them are for testing.
  - More **stable**: consider users with sufficient ratings only.
  - Widely used in literature with N=10.
    → Used to compare with existing methods.
Effect of Capacity

**Zero-one**: With higher dimension, converges slowly.

**All**: With higher dimension, ultimate performance is better.

**AvgP, NDCG**: With higher dimension, it less overfits.
Effect of Number of Local Models

**Zero-one:** With more local models, converges slowly.

**All:** With more local models, ultimate performance is better.

**AvgP, NDCG:** With more local models, it less overfits.
Effect of Loss Functions


All: Convergence and overfitting depends on loss function.

## Comparison with other methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Average Precision</th>
<th>NDCG@10</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MovieLens</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CofiRank</td>
<td>0.6632</td>
<td>0.6502</td>
</tr>
<tr>
<td>GCR (SVD with ranked loss)</td>
<td>0.7209</td>
<td>0.6990</td>
</tr>
<tr>
<td>LCR</td>
<td><strong>0.7406</strong></td>
<td><strong>0.7152</strong></td>
</tr>
<tr>
<td><strong>EachMovie</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CofiRank</td>
<td><strong>0.7491</strong></td>
<td>0.6635</td>
</tr>
<tr>
<td>GCR (SVD with ranked loss)</td>
<td>0.7088</td>
<td>0.6998</td>
</tr>
<tr>
<td>LCR</td>
<td>0.7307</td>
<td><strong>0.7166</strong></td>
</tr>
<tr>
<td><strong>Yelp</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CofiRank</td>
<td>0.7246</td>
<td>0.6997</td>
</tr>
<tr>
<td>GCR (SVD with ranked loss)</td>
<td>0.7754</td>
<td>0.7465</td>
</tr>
<tr>
<td>LCR</td>
<td><strong>0.7903</strong></td>
<td><strong>0.7575</strong></td>
</tr>
</tbody>
</table>
Take-home Messages

- In recommendation systems, the rating matrix is low-rank only locally.

- Local low-rank assumption is realistic for ordering problem as well as rating prediction.

- LCR (Local Collaborative Ranking) algorithm is highly parallelizable and scalable.
Source code available soon!

- PREA toolkit: http://prea.gatech.edu