Improve Collaborative Filtering through Bordered Block Diagonal Form Matrices

Yongfeng Zhang, Min Zhang, Yiqun Liu, Shaoping Ma
Dept. CS, Tsinghua University, China
Outline

- Backgrounds
- Our Approach
- Experiments
- Discussions
- Conclusions
Backgrounds

- **Recommender Systems**
  - Playing an important role on the web
  - E-Commerce and Review Services, e.g. Amazon and Yelp

- **Collaborative Filtering**
  - The ability to recommend without clear content information
  - Have achieved significant success

- **Rating Prediction**
  - Make rating predictions on user-item rating matrix based on observed ratings
  - One of the core tasks of CF
  - Widely investigated
Backgrounds

- The use of user-item communities
  - Benefits the efficiency and effect in many cases

- Matrix Clustering
  - Extract user-item sub-matrices (clusters)
  - Conduct Collaborative Filtering on each sub-matrices

- Some existing popular approaches
  - User / Item Clustering [Corner & Herlocker, SIGIR’99]
  - Co-Clustering [George & Merugu, ICDM’05]
  - User-Item Subgroups Mining [Xu & Bu et al, WWW’12]

- Our Concerns
  - Clusters may not be a ‘natural’ representation of communities
  - Usually forces a user/item to be in a single cluster
Observations

- **Common Interests and Special Interests**
  - **Common Interests**: Items favored by users from different communities
  - **Special Interests**: items favored by some specific groups of users
  - **Common Interests** can be shared by different user groups
    - e.g. The hot movies
Bordered Block Diagonal Form (BBDF) structure

The Intuition
- Row Borders: Super Users
- Column Borders: Super Items, e.g. hot movies
- Diagonal Blocks: User-Item Communities
BBDF and GPVS

Graph Partitioning by Vertex Separator (GPVS [Karypis, 2011])
The ABBDF structure

- An underlying assumption in BBDF structure.
  - There is no edge between communities.

- May not be a reasonable assumption
  - User might indeed focus on some domains
  - They do step into other domains sometimes
the ABBDF structure

- Approximate Bordered Block Diagonal Form (ABBDF)

- A special form of ABBDF structure
  * The ABBDF structure without border
  * Can be achieved with Graph Partitioning by Edge Separator (GPES) algorithms
  * Remove some edges (non-zeros in off-diagonal areas) and split the graph
More general conclusions

Any Community Detection result on a bipartite graph can be represented as an ABBDF structure
  - Not only GPVS or GPES algorithms

Corollary: Can be represented as an BBDF structure if there is no inter-community edge.
Algorithms

- How to permute matrices into (A)BBDF structures?

- BBDF Permutation Algorithm
  - Algorithm1, Basic-BBDF-Permutation procedure
  - Algorithm2, BBDF-Permutation procedure

- ABBDF Permutation Algorithm
  - Algorithm3, ABBDF-Permutation procedure
  - Algorithm4, Improve-Density procedure
BBDF permutation algorithm

The basic procedure for BBDF permutation

**Algorithm 1 Basic-BBDF-Permutation**

**Require:**
- User-Item rating matrix $A$.
- Bipartite graph $G = (V, E) = (R \cup C, E)$ of $A$.
- $R/C$ are row/column vertex sets of $V$ correspondingly.

**Ensure:**
- Average density of resulting diagonal blocks $\bar{\rho}$.

1: $\Gamma_v \leftarrow \{V_1, V_2, \ldots, V_k; V_S\} \leftarrow \text{GPVS}(G)$
2: Permute rows of $A$ in order of $R_1R_2\cdots R_kR_S$
3: Permute columns of $A$ in order of $C_1C_2\cdots C_kC_S$
4: return $\bar{\rho}(D_1D_2\cdots D_k)$

Remove a set of vertices $V_S$ and split the graph into $k$ connected components.

Remove the vertex set $V_S$ to borders and permute the remaining to diagonals.

Return the average density of resulting diagonal blocks in this stage.

---

Remove a set of vertices $V_S$ and split the graph into $k$ connected components.

Remove the vertex set $V_S$ to borders and permute the remaining to diagonals.

Return the average density of resulting diagonal blocks in this stage.

---

**R**

1. 2. 3.

**C**

1. 2. 3. 4. 5. 6. 7. 8. 9.
BBDF permutation algorithm (cont.)

- BBDF Permutation algorithm
  - Permute sub-matrices into BBDF structure recursively

**Algorithm 2 BBDF-Permutation**

**Require:**
- User-Item rating matrix $A$.
- Bipartite graph $G = (V, E)$ of $A$.
- Density requirement $\rho$.

**Ensure:**
- Matrix $A$ permuted into BBDF structure.

1. $\rho_A \leftarrow \rho(A)$
2. **if** $\rho_A < \rho$ **then** $\triangleright$ else do nothing
3. $\bar{\rho} \leftarrow$ Basic-BBDF-Permutation($A, G$)
4. **if** $\bar{\rho} > \rho_A$ **then** $\triangleright$ else do nothing
5. **for each** diagonal block $D_i$ in $A$ **do**
6. BBDF-Permutation($D_i, G_{V_i}, \rho$) $\triangleright V_i$ denotes the vertex set of $D_i$, $G_{V_i}$ is the subgraph induced by $V_i$
7. **end for**
8. **end if**
9. **end if**

The expected minimum average density of diagonal blocks.

If the density of a sub-matrix has not reached the requirement, split it using the basic procedure.

If the average density improves after split, take the split and recurse. Else, stop recursion.
Algorithm 3 ABBDF-Permutation($A, \mathcal{G}, \rho$)

Require:
User-Item rating matrix $A$.
Bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}) = (\mathcal{R} \cup \mathcal{C}, \mathcal{E})$ of $A$.
Density requirement $\rho$.

Ensure:
Matrix $A$ permuted into ABBDF structure.

1: if $\rho(A) \geq \rho$ then
2: return
3: else
4: $\Gamma_e \leftarrow \{\mathcal{V}_1 \mathcal{V}_2 \cdots \mathcal{V}_k\} \leftarrow \text{GPES}(\mathcal{G})$
5: Permute rows of $A$ in order of $\mathcal{R}_1 \mathcal{R}_2 \cdots \mathcal{R}_k$
6: Permute columns of $A$ in order of $\mathcal{C}_1 \mathcal{C}_2 \cdots \mathcal{C}_k$
7: $\{\mathcal{V}_1' \mathcal{V}_2' \cdots \mathcal{V}_k'; \mathcal{V}_S'\} \leftarrow \text{Improve-Density}(A, \mathcal{G}, \Gamma_e)$
8: for each diagonal block $D_i$ in $A$ do
9: ABBDF-Permutation($D_i, \mathcal{G}_{\mathcal{V}_i'}, \rho$)
10: end for
11: end if

Split the corresponding graph using GPES, resulting in a ABBDF matrix without borders.

If the average density of diagonal blocks didn’t improve, try to improve it by moving some rows/columns to borders.
For each row and column from each diagonal block, check whether its removal improves average density.

Permute the row/column to borders whose removal improves average density most.

Until average density is higher than the original matrix.
Make Rating Predictions

- Extract sub-matrices representing communities from the (A)BBDF structure

(a) BBDF matrix

(b) Submatrices extracted

\[
X = \begin{bmatrix}
J_{11} & J_{12} & J_{B_1} & J_2 & J_B \\
D_{11} & C_{11} & C_1^1 & C_1^2 & C_1^3 \\
R_{11} & R_{12} & B_1 & D_2 & C_2 \\
\bar{R}_1^1 & \bar{R}_1^2 & \bar{R}_1^3 & R_2 & B \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
D_{11} & C_{11} & C_1^1 & C_1^2 & C_1^3 \\
R_{11} & B_1 & C_1^3 & R_1^2 & R_1^3 & B \\
D_2 & C_2 \\
R_2 & B \\
\end{bmatrix}
\]
Make rating predictions in 2 steps:

- Step 1: Conduct CF in each of the sub-matrices
- Step 2: Average predictions in duplicated blocks
  - E.g. $S_{2x}$ is predicted twice in sub-matrices A and B
Experiment Setup

- Dataset Description
  - 4 real-world datasets: MovieLens-100k, MovieLens-1m, Dianping, and Yahoo! Music.

<table>
<thead>
<tr>
<th></th>
<th>ML-100K</th>
<th>ML-1M</th>
<th>DianPing</th>
<th>Yahoo!Music</th>
</tr>
</thead>
<tbody>
<tr>
<td>#users</td>
<td>943</td>
<td>6,040</td>
<td>11,857</td>
<td>1,000,990</td>
</tr>
<tr>
<td>#items</td>
<td>1,682</td>
<td>3,952</td>
<td>22,365</td>
<td>624,961</td>
</tr>
<tr>
<td>#ratings</td>
<td>100,000</td>
<td>1,000,209</td>
<td>510,551</td>
<td>256,804,235</td>
</tr>
<tr>
<td>#ratings/user</td>
<td>106.045</td>
<td>165.598</td>
<td>43.059</td>
<td>256.550</td>
</tr>
<tr>
<td>#ratings/item</td>
<td>59.453</td>
<td>253.089</td>
<td>22.828</td>
<td>410.912</td>
</tr>
<tr>
<td>average density</td>
<td>0.0630</td>
<td>0.0419</td>
<td>0.00193</td>
<td>0.000411</td>
</tr>
</tbody>
</table>
Experiment Setup (cont.)

- Experimented the framework on 4 CF algorithms
  - User-based
  - Item-based
  - SVD (Singular Value Decomposition)
  - NMF (Nonnegative Matrix Factorization)

- Evaluation Metric
  - Root Mean Square Error (RMSE)

\[
RMSE = \sqrt{\frac{\sum_{i=1}^{N} (r_i - \hat{r_i})^2}{N}}
\]
Community Analysis

- Density requirement v.s. # diagonal blocks
  - Low density -> A small number of big communities
  - High density -> A large number of small communities

- Example of BBDF permutation results on DianPing
An appropriate density requirement gives reasonable community detection results.
Density requirement v.s. # diagonal blocks

- **BBDF**: # Diagonal Blocks grows at first and tends to be stable at last.
- **ABBDF**: # Diagonal Blocks grows consistently with the growth of density requirement.
Similar results are observed on the other datasets.

Graphs showing the number of diagonal blocks for different datasets (MovieLens-100K, MovieLens-1M, Dianping, Yahoo) with varying density requirements.
Prediction accuracy tends to be stable (the BBDF algorithm stops to split matrices when density requirement is too high.)

Gains better prediction accuracy given appropriate density requirement.
ABBDF: RMSE v.s. Density Requirements

- Tends to gain better prediction accuracy at first.
- But the performance tends to drop rapidly given high density requirements.
- The algorithm leads to many small scattered communities.
Similar results were observed on the other datasets.
Discussions

- Potential advantage: Selective re-training in practical systems
  - Ratings are made by users continuously in real-world systems
  - Have to retrain a CF model every period of time
  - Only need to retrain those really in need of re-training
    - E.g. The RMSE has reached a criterion
Wrap up

In this work:
- Investigated the relationship between (A)BBDF structure and community detection
- Designed density-based algorithms to transform a matrix into (A)BBDF structure
- Proposed a framework to make rating predictions on this structure

Future directions
- (A)BBDF structure is independent of specific community detection algorithm
  - Investigate other kinds of (A)BBDF permutation algorithms except for GPVS and GPES
  - Conduct selective re-training using our framework
Thanks!