Factored Proximity Models for Top-N Recommendations

Athanasios N. Nikolakopoulos\textsuperscript{1}, Vassilis Kalantzis\textsuperscript{2}, Efstratios Gallopoulos\textsuperscript{3} and John D. Garofalakis\textsuperscript{3}

\textsuperscript{1}Digital Technology Center, University of Minnesota, USA.
\textsuperscript{2}Dept. of Computer Science & Engineering, University of Minnesota, USA.
\textsuperscript{3}Dept. of Computer Engineering & Informatics, University of Patras, Greece.

August 08, 2017

IEEE International Conference on Big Knowledge, IEEE ICBK 2017
Recommender Systems

- Widely Applicable Technology
  - Value for Customers
  - Value for Companies

Collaborative Filtering

- **Model:** Ratings!
- Recommendation task
  - Rating Prediction
  - Top-N Lists
Sparsity Problem

- Limits the quality recommendations; especially for \textit{Long-Tail Items}.
- Intrinsic RS Characteristic
  - Cold-Start Problem

Promising Approaches

- Graph-based Models
  (+) Good Performance
  (-) Scalability Issues
- Latent-Factor Models.

Our Focus: \textit{Efficient and High-Quality Top-N Recommendations Even under Extreme Sparsity}
EigenRec Framework
EigenRec Framework

**EigenRec**

- Build a symmetric $m \times m$ **Inter-Item Proximity Matrix** $A$, each element of which is defined to be a product of a *Scaling* and a *Similarity* component.

$$[A]_{ij} \triangleq \xi(i, j) \cdot \kappa(i, j)$$

- Build a **Lower Dimensional Model** using the principal eigenvectors of $A$
- Project the Users’s feedback vectors onto the Latent Subspace:

$$\Pi \triangleq RV_f V_f^T$$

**Simple Baseline Choices**

**Scaling function**

$$\xi(i, j) \triangleq f(i, j; d) = (\|r_i\| \|r_j\|)^d$$

**Similarity functions**

$$\kappa(i, j) \triangleq \begin{cases} 
\cos(v_i, v_j) \\
\text{pc}(v_i, v_j) \\
\text{jaccard}(v_i, v_j)
\end{cases}$$
**PureSVD within EigenRec**

**PureSVD**

\[ \Pi_{\text{PureSV}} \triangleq U_f \Sigma_f Q_f^T \equiv \cdots \equiv R Q_f Q_f^T \]

Where \( Q_f \), the matrix containing the \( f \) principal eigenvectors of:

\[ R^T R = \text{items} \begin{bmatrix} - & \mathbf{r}_{v_1}^T & - \end{bmatrix} \times \text{users} \begin{bmatrix} \mathbf{r}_{v_j} \end{bmatrix} \]

\[ = \text{items} \begin{bmatrix} \vdots \end{bmatrix} \underbrace{\| \mathbf{r}_{v_i} \| \| \mathbf{r}_{v_j} \| \cdot \cos \theta_{ij}}_{\text{scaling}} \cdot \underbrace{\} \text{similarity} \]

- **PureSVD \equiv EigenRec** with Cosine similarity and \( f(i, j; 1) \)
Computing EigenRec

**EigenRec:**

**Input:** Inter-Item proximity matrix \( A \in \mathbb{R}^{m \times m} \). Rating Matrix \( R \in \mathbb{R}^{n \times m} \). Latent Factors \( f \).

**Output:** Matrix \( \Pi \in \mathbb{R}^{n \times m} \) whose rows are the recommendation vectors for every user.

1. \( q_j = 0 \), set \( r \leftarrow q \) as a random vector
2. \( \beta_0 \leftarrow \|r\|_2 \)
3. \( \text{for } j \leftarrow 1, 2, \ldots, \text{ do} \)
4. \( q_j \leftarrow r / \beta_{j-1} \)
5. \( r \leftarrow Aq_j \)
6. \( r \leftarrow r - q_{j-1} \beta_{j-1} \)
7. \( \alpha_j \leftarrow q_j^T r \)
8. \( r \leftarrow r - q_j \alpha_j \)
9. \( r \leftarrow (I - Q_j Q_j^T)r \),
10. \( \beta_j \leftarrow \|r\|_2 \)
11. Solve the tridiag problem \((Q_j^T A Q_j) \Xi_j = \Theta_j \Xi_j\)
12. Form the \( j \) approximate eigenvectors \( Q_j \Xi_j \) of \( A \)
13. If the \( f \) top eigenvectors have converged, stop.
14. \( \text{end for} \)
15. Compute latent factors \( V = Q_f \Xi \)
16. return \( \Pi \leftarrow RVV^T \)

**Computational Aspects:**

- The MV product in \( j \) Lanczos steps is \( O(j \cdot \text{nnz}) \)
- Making the \( j \)-th vector orthogonal to the previous ones costs \( O(jm) \)

**Parallel Implementation:**

<table>
<thead>
<tr>
<th></th>
<th>f=50</th>
<th>100</th>
<th>150</th>
<th>200</th>
<th>300</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MovieLens20M</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8 cores</td>
<td>11.0</td>
<td>18.0</td>
<td>24.2</td>
<td>28.7</td>
<td>36.8</td>
</tr>
<tr>
<td>16 cores</td>
<td>6.9</td>
<td>11.6</td>
<td>15.6</td>
<td>18.9</td>
<td>24.3</td>
</tr>
<tr>
<td>32 cores</td>
<td>4.8</td>
<td>8.1</td>
<td>11.1</td>
<td>13.5</td>
<td>17.6</td>
</tr>
<tr>
<td>64 cores</td>
<td>3.5</td>
<td>6.0</td>
<td>8.2</td>
<td>9.9</td>
<td>12.5</td>
</tr>
</tbody>
</table>

The Code is available here:
https://github.com/nikolakopoulos/EigenRec
Qualitative Evaluation
Effects of Prior Popularity

Methodology

- Randomly sample 1.4% of the ratings of the dataset \( \Rightarrow \) probe set \( P \)
- Use each item \( v_j \), rated with 5 stars by user \( u_i \) in \( P \) \( \Rightarrow \) test set \( T \)
- Randomly select another 1000 unrated items of the same user for each item in \( T \)
- Form ranked lists by ordering all the 1001 items

Metrics

- Recall
- Precision
- R-Score
- NDCG@k
- MRR

![Graphs showing the comparison of Cosine, Pearson, and Jaccard metrics for Yahoo and ML-1M datasets. The graphs illustrate the distribution of the metrics with PureSVD highlighted.](image-url)
Standard Top-N Recommendations

Methodology

- Randomly sample 1.4% of the ratings of the dataset ⇒ probe set $\mathcal{P}$
- Use each item $v_j$, rated with 5 stars by user $u_i$ in $\mathcal{P}$ ⇒ test set $\mathcal{T}$
- Randomly select another 1000 unrated items of the same user for each item in $\mathcal{T}$
- Form ranked lists by ordering all the 1001 items

Metrics

- Recall
- Precision
- R-Score
- NDCG@k
- MRR
Long-Tail Recommendations

Methodology

- We order the items according to their popularity (measured in terms of number of ratings)
- We further partition the test set $\mathcal{T}$ into two subsets, $\mathcal{T}_{\text{head}}$ and $\mathcal{T}_{\text{tail}}$
- We discard the popular items and we evaluate EigenRec and the other algorithms on the $\mathcal{T}_{\text{tail}}$ test set, using the procedure explained previously.

Metrics

- Recall
- Precision
- R-Score
- NDCG@$k$
- MRR
Cold-Start Recommendations I

Cold-Start Problem

- Difficulty of making reliable recommendations due to an initial lack of ratings
- In beginning stages, when there is not sufficient number of ratings for the collaborative filtering algorithms to uncover similarities ⇒ New Community Problem
- Introduction of new users to an existing system where they have not rated many items ⇒ New Users Problem
**New Community:**

- **Methodology:** Randomly select to include 33%, 66%, and 100% of the overall ratings on two new artificially sparsified versions of the dataset.

**New Users:**

- **Methodology:** Randomly select 50 users having rated at least 100 items and randomly delete 95% of each users’ ratings.
Conclusions and Future Work
Conclusions and Future Work

EigenRec

- Computationally Efficient framework for Top-N Recommendations
- Allows for flexible modeling and control of the effects of prior popularity
- Natural generalization of PureSVD
  - (+) Optimize its Top-N recommendation performance
  - (+) Alleviate its inherent popularity bias
  - (+) Compute it more efficiently
- Good Top-N Recommendation Performance

Future Directions

- Explore more elaborate Similarity and Scaling functions
- Explore the Hierarchical structure of the Itemspace
A. N. Nikolakopoulos, V. Kalantzis, E. Gallopoulos, and J. Garofalakis. 
**EigenRec: Generalizing PureSVD for Effective and Efficient Top-N Recommendations.**
*Knowledge and Information Systems*, 2018.

A. N. Nikolakopoulos, V. Kalantzis, E. Gallopoulos, and J. Garofalakis. 
**Factored Proximity Models for Top-N Recommendations.**

P. Cremonesi, Y. Koren, and R. Turrin. 
**Performance of recommender algorithms on top-n recommendation tasks.**

**An experimental investigation of kernels on graphs for collaborative recommendation and semisupervised classification.**
*Neural Netw.*, 31:53–72, July 2012.
Thank you for your Attention!
Questions?