Maxios: Large-scale Nonnegative Matrix Factorization for Collaborative Filtering

Simon Shaolei Du, Yilin Liu, Boyi Chen, Lei Li
{simonshaoleidu,daniel.liu,bchen91}@berkeley.edu, lilei22@baidu.com

Predicting ratings for recommendation

Movie ratings

Music ratings

Problem Description

Predicting missing values in User-Item matrix

<table>
<thead>
<tr>
<th>User</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>User A</td>
<td>1</td>
<td>?</td>
<td>5</td>
<td>4</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>User B</td>
<td>?</td>
<td>2</td>
<td>3</td>
<td>?</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>User C</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>?</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>User D</td>
<td>?</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>User E</td>
<td>1</td>
<td>2</td>
<td>?</td>
<td>?</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Problem characteristics:
• **Large scale**: millions of users, sub-millions of items
• **Highly Sparse**
• **Need to interpret** ratings
  - non-negativity constraints

Limitation of Existing Methods

• **EM based methods [Liu 2010]**: time consuming to compute a full user-item matrix (HUGE!) each iteration
• **ALS based method [Zhang et al 2006, Kim & Choi 2009]**: costly update in each iteration
• **Multiplicative updates [Lee & Seung 1999]**: slow convergence

Proposed Maxios

Weighted NMF formulation:
\[
\min_{U \geq 0, V \geq 0} \|A - W \odot (UV)\|_F^2
\]
\[
W_{ij} = 1 \quad \text{if } A_{ij} \text{ is not missing} \\
= 0 \quad \text{otherwise}
\]
\(\odot\): elementwise product

Reformulation using ADMM
\[
\min \|A - W \odot (UV)\|_F^2 + I_+(X) + I_+(Y)
\]
\(\text{s.t. } U = X, V = Y\)

Parallel update steps:

Update each row of U independently
\[
U_{i}^{t+1} = (A_i \odot W_i) (V^t \odot W_i^t)^T + \alpha X_i^t - \Lambda_i^t \\
\cdot (V^t \odot W_i^t) (V^t \odot W_i^t)^T + \alpha \Lambda_i^t)^{-1}
\]
Update each row of X independently
\[
X_{i}^{t+1} = P_+ (U_{i}^{t+1} + \frac{\Lambda_i^t}{\alpha})
\]
Updates for V and Y are in similar fashion.

Implementation

Maxios is built on top of Spark, a distributed in-memory computing platform. Sparse data representation.

Workload Allocation

Preprocess to balance the workload of worker nodes

Experiments

<table>
<thead>
<tr>
<th>Data</th>
<th>users</th>
<th>Items</th>
<th>nnz</th>
<th>sparsity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Netflix</td>
<td>0.5M</td>
<td>17770</td>
<td>0.7B</td>
<td>1.18%</td>
</tr>
<tr>
<td>Yahoo</td>
<td>2M</td>
<td>98213</td>
<td>0.1B</td>
<td>0.06%</td>
</tr>
</tbody>
</table>

• Baseline Algorithms
  • Multiplicative Updates
  • Alternating Least Squares

Netflix Results

Rank k = 10, training and testing RMSE

Yahoo! Music Results

Proposed a scalable NMF solution. Benefits of Maxios:
• Reducing computation overhead by utilizing sparsity. Weighted formulation avoids computing a User-Item matrix in each iteration.
  • Highly scalable
    - independent update of each row of U, X and each column of V, Y
  • Fast Updating
    - Maxios enables closed-form updates for U, V, X, Y via ADMM
    - benefits from distributed in-memory computing in Spark

Contribution

Yahoo music ranking results

Rank k = 10, training and testing RMSE

Rank k = 50, training and testing RMSE