Distributed Itembased Collaborative Filtering with Apache Mahout

Sebastian Schelter
ssc@apache.org
twitter.com/sscdotopen

7. October 2010
Overview

1. What is Apache Mahout?
2. Introduction to Collaborative Filtering
3. Itembased Collaborative Filtering
4. Computing similar items with Map/Reduce
5. Implementations in Mahout
6. Further information
What is Apache Mahout?

A scalable Machine Learning library

- scalable to reasonably large datasets (core algorithms implemented in Map/Reduce, runnable on Hadoop)
- scalable to support your business case (Apache License)
- scalable community

Usecases

- **Clustering** (group items that are topically related)
- **Classification** (learn to assign categories to documents)
- **Frequent Itemset Mining** (find items that appear together)
- **Recommendation Mining** (find items a user might like)
Recommendation Mining

= Help users find items they might like
Users, Items, Preferences

Terminology

- **users** interact with **items** (books, videos, news, other users,...)
- **preferences** of each user towards a small subset of the items known (numeric or boolean)

Algorithmic problems

- **Prediction**: Estimate the preference of a user towards an item he/she does not know
- Use Prediction for **Top-N-recommendation**: Find the N items a user might like best
Explicit and Implicit Ratings

Where do the preferences come from?

Explicit Ratings

- users explicitly express their preferences (e.g. ratings with stars)
- willingness of the users required

Implicit Ratings

- interactions with items are interpreted as expressions of preference (e.g. purchasing a book, reading a news article)
- interactions must be detectable
Collaborative Filtering

How does it work?

- **the past predicts the future:** all predictions are derived from historical data (the preferences you already know)
- completely **content agnostic**
- very popular (e.g. used by Amazon, Google News)

Mathematically

- **user-item-matrix** is created from the preference data
- task is to **predict missing entries** by finding patterns in the known entries
### A sample user-item-matrix

<table>
<thead>
<tr>
<th></th>
<th>The Matrix</th>
<th>Alien</th>
<th>Inception</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Bob</td>
<td>?</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Peter</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
Itembased Collaborative Filtering

Algorithm

- ** neighbourhood-based approach
- works by **finding similarly rated items** in the user-item-matrix
- estimates a user's preference towards an item by looking at his/her preferences towards similar items

Highly scalable

- item similarities tend to be relatively **static**, can be **precomputed offline** periodically
- **less items than users** in most scenarios
- looking at a **small number of similar items** is sufficient
Example

Similarity of „The Matrix“ and „Inception“

- rating vector of „The Matrix“: (5, -, 4)
- rating vector of „Inception“: (4, 5, 2)

- isolate all **cooccurred ratings** (all cases where a user rated both items)
- pick a **similarity measure** to compute a similarity value between -1 and 1
e.g. Pearson-Correlation

\[
\text{corr}(i, j) = \frac{\sum_{u \in U} (R_{u,i} - \bar{R}_i)(R_{u,j} - \bar{R}_j)}{\sqrt{\sum_{u \in U} (R_{u,i} - \bar{R}_i)^2} \sqrt{\sum_{u \in U} (R_{u,j} - \bar{R}_j)^2}} = 0.47
\]
Example

Prediction: Estimate Bob's preference towards „The Matrix“

- look at all items that
  a) are similar to „The Matrix“
  b) have been rated by Bob
  => „Alien“, „Inception“

- estimate the unknown preference with a weighted sum

\[
P_{Bob, \text{Matrix}} = \frac{S_{\text{Matrix, Alien}} \cdot r_{Bob, \text{Alien}} + S_{\text{Matrix, Inception}} \cdot r_{Bob, \text{Inception}}}{|S_{\text{Matrix, Alien}}| + |S_{\text{Matrix, Inception}}|} = 1.5
\]
Algorithm in Map/Reduce

How can we compute the similarities efficiently with Map/Reduce?

Key ideas

- we can ignore pairs of items without a cooccurring rating
- we need to see all cooccurring ratings for each pair of items in the end

Inspired by an algorithm designed to compute the pairwise similarity of text documents

Mahout's implementation is more generalized to be usable with other similarity measures, see DistributedVectorSimilarity and RowSimilarityJob for more details
Algorithm in Map/Reduce - Pass 1

**Map** - make user the key

<table>
<thead>
<tr>
<th>(Alice, Matrix, 5)</th>
<th>(Alice, Alien, 1)</th>
<th>(Alice, Inception, 4)</th>
<th>(Bob, Alien, 2)</th>
<th>(Bob, Inception, 5)</th>
<th>(Peter, Matrix, 4)</th>
<th>(Peter, Alien, 3)</th>
<th>(Peter, Inception, 2)</th>
</tr>
</thead>
</table>

**Reduce** - create inverted index

<table>
<thead>
<tr>
<th>Alice (Matrix, 5)</th>
<th>Alice (Alien, 1)</th>
<th>Alice (Inception, 4)</th>
<th>Bob (Alien, 2)</th>
<th>Bob (Inception, 5)</th>
<th>Peter (Matrix, 4)</th>
<th>Peter (Alien, 3)</th>
<th>Peter (Inception, 2)</th>
</tr>
</thead>
</table>
Algorithm in Map/Reduce - Pass 2

**Map** - emit all cooccurred ratings

<table>
<thead>
<tr>
<th>User</th>
<th>Item 1</th>
<th>Item 2</th>
<th>Ratings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Matrix</td>
<td>Alien</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Inception</td>
<td>4</td>
</tr>
<tr>
<td>Bob</td>
<td>Alien</td>
<td>Inception</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>Peter</td>
<td>Matrix</td>
<td>Alien</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Inception</td>
<td>2</td>
</tr>
</tbody>
</table>

**Reduce** - compute similarities

<table>
<thead>
<tr>
<th>Item 1</th>
<th>Item 2</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix</td>
<td>Alien</td>
<td>5,1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4,3</td>
</tr>
<tr>
<td>Matrix</td>
<td>Inception</td>
<td>5,4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4,2</td>
</tr>
<tr>
<td>Alien</td>
<td>Inception</td>
<td>1,4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2,5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3,2</td>
</tr>
<tr>
<td>Alien</td>
<td>Inception</td>
<td>-0.47</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-0.63</td>
</tr>
<tr>
<td>Matrix</td>
<td>Alien</td>
<td>-0.47</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.47</td>
</tr>
<tr>
<td>Alien</td>
<td>Inception</td>
<td>-0.63</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.47</td>
</tr>
</tbody>
</table>
Implementations in Mahout

**ItemSimilarityJob**

- computes **all item similarities**
- various configuration options:
  - similarity measure to use (e.g. cosine, Pearson-Correlation, Tanimoto-Coefficient, your own implementation)
  - maximum number of similar items per item
  - maximum number of cooccurrences considered
  - ...

- Input: preference data as CSV file, each line represents a single preference in the form `userID,itemID,value`
- Output: pairs of itemIDs with their associated similarity value
Implementations in Mahout

RecommenderJob

- **Distributed Itembased Recommender**
- various configuration options:
  - similarity measure to use
  - number of recommendations per user
  - filter out some users or items
  - ...

- Input: the preference data as CSV file, each line contains a preference in the form `userID,itemID,value`
- Output: userIDs with associated recommended itemIDs and their scores
Further information

Mahout's website, wiki and mailinglist
- http://mahout.apache.org
- user@mahout.apache.org

Mahout in Action, available through Manning's Early Access Program
- http://manning.com/owen

B. Sarwar et al: „Itembased collaborative filtering recommendation algorithms“, 2001

T. Elsayed et al: „Pairwise document similarity in large collections with MapReduce“, 2008