Collaborative Filtering at Scale
Recommender engines with Mahout and Hadoop
Berlin Buzzwords
Sean Owen
8 June 2010
Mahout is …

- Machine learning …
  - Collaborative filtering (recommenders)
  - Clustering
  - Classification
  - Frequent item set mining
  - and more

- … at scale
  - Much implemented on Hadoop
  - Efficient data structures
Collaborative Filtering is …

- Given a user’s preferences for items, guess which other items would be highly preferred
- Only needs preferences; users and items opaque
- Many algorithms!
Collaborative Filtering is ...

Sean likes "Scarface" a lot  
Robin likes "Scarface" somewhat  
Grant likes "The Notebook" not at all  
...

\[(123, 654, 5.0)\]  
\[(789, 654, 3.0)\]  
\[(345, 876, 1.0)\]  
...

Grant may like "Scarface" quite a bit  
...

\[(345, 654, 4.5)\]  
...

Magic
Recommending people food
Item-Based Algorithm

- Recommend items similar to a user’s highly-preferred items
Item-Based Algorithm

- Have user’s preference for items
- Know all items and can compute weighted average to estimate user’s preference
- What is the item – item similarity notion?

for every item i that u has no preference for yet
for every item j that u has a preference for
compute a similarity s between i and j
add u's preference for j, weighted by s,
to a running average
return the top items, ranked by weighted average
Item-Item Similarity

- Could be based on content...
  - Two foods similar if both sweet, both cold

- **BUT** in collaborative filtering, based only on preferences (numbers)
  - Pearson correlation between ratings?
  - Log-likelihood ratio?

- **Simple co-occurrence:**
  Items similar when appearing often in the same user’s set of preferences
Estimating preference

Preference

5

5

2

4.5

Co-occurrence

9

16

5

\[
\frac{5 \cdot 9 + 5 \cdot 16 + 2 \cdot 5}{9 + 16 + 5} = \frac{135}{30}
\]
As matrix math

- User’s **preferences** are a **vector**
  - Each dimension corresponds to one item
  - Dimension value is the preference value

- Item-item **co-occurrences** are a **matrix**
  - Row i / column j is count of item i / j co-occurrence

- Estimating preferences:
  co-occurrence **matrix** × preference (column) **vector**

Collaborative Filtering at Scale
16 animals ate both hot dogs and ice cream

<table>
<thead>
<tr>
<th></th>
<th>16</th>
<th>9</th>
<th>16</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>9</td>
<td>30</td>
<td>19</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>19</td>
<td>23</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>5</td>
<td>10</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>4</td>
<td>20</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

10 animals ate blueberries

0  135
5  251
5  220
2  60
0  70

As matrix math
A different way to multiply

- **Normal**: for each row of matrix
  - Multiply (dot) row with column vector
  - Yields scalar: one final element of recommendation vector

- **Inside-out**: for each element of column vector
  - Multiply (scalar) with corresponding matrix column
  - Yield column vector: parts of final recommendation vector
  - Sum those to get result
  - Can skip for zero vector elements!
As matrix math, again

```
5 9 30
5 19 5 3 2

16 19 5 4 5 15
19 23 5 5 10

5 3 2
5 10 20
2
```

= 135

= 251

= 220

= 60

= 70
What is MapReduce?

1. Input is a series of key-value pairs: (K1,V1)
2. map() function receives these, outputs 0 or more (K2,V2)
3. All values for each K2 are collected together
4. reduce() function receives these, outputs 0 or more (K3,V3)

- Very distributable and parallelizable
- Most large-scale problems can be chopped into a series of such MapReduce jobs
Build user vectors (mapper)

- Input is text file: user, item, preference

- Mapper receives
  - K1 = file position (ignored)
  - V1 = line of text file

- Mapper outputs, for each line
  - K2 = user ID
  - V2 = (item ID, preference)
Build user vectors (reducer)

- Reducer receives
  - K2 = user ID
  - V2,… = (item ID, preference), …

- Reducer outputs
  - K3 = user ID
  - V3 = Mahout Vector implementation

- Mahout provides custom Writable implementations for efficient Vector storage
Count co-occurrence (mapper)

- Mapper receives
  - K1 = user ID
  - V1 = user Vector

- Mapper outputs, for each pair of items
  - K2 = item ID
  - V2 = other item ID
Count co-occurrence (reducer)

- Reducer receives
  - K2 = item ID
  - V2, ... = other item ID, ...

- Reducer tallies each other item; creates a Vector

- Reducer outputs
  - K3 = item ID
  - V3 = column of co-occurrence matrix as Vector
Partial multiply (mapper #1)

- Mapper receives
  - K1 = user ID
  - V1 = user Vector

- Mapper outputs, for each item
  - K2 = item ID
  - V2 = (user ID, preference)
Partial multiply (mapper #2)

- Mapper receives
  - K1 = item ID
  - V1 = co-occurrence matrix column Vector

- Mapper outputs
  - K2 = item ID
  - V2 = co-occurrence matrix column Vector
Partial multiply (reducer)

- Reducer receives
  - $K2 = \text{item ID}$
  - $V2, \ldots = (\text{user ID, preference}), \ldots$
    - and co-occurrence matrix column Vector

- Reducer outputs, for each item ID
  - $K3 = \text{item ID}$
  - $V3 = \text{column vector and (user ID, preference) pairs}$
Aggregate (mapper)

- Mapper receives
  - K1 = item ID
  - V1 = column vector and (user ID, preference) pairs

- Mapper outputs, for each user ID
  - K2 = user ID
  - V2 = column vector times preference
Aggregate (reducer)

- Reducer receives
  - K2 = user ID
  - V2,... = partial recommendation vectors

- Reducer sums to make recommendation Vector and finds top n values

- Reducer outputs, for top value
  - K3 = user ID
  - V3 = (item ID, value)
Reality is a bit more complex
Ready to try

- Obtain and build Mahout from Subversion
  http://mahout.apache.org/versioncontrol.html

- Set up, run Hadoop in local pseudo-distributed mode

- Copy input into local HDFS

  hadoop jar mahout-0.4-SNAPSHOT.jar
  org.apache.mahout.cf.taste.hadoop.item.RecommenderJob
  -Dmapred.input.dir=input
  -Dmapred.output.dir=output
Mahout in Action

- Recommenders
  - Data representation
  - Non-distributed algorithms
  - Distributed algorithms

- Clustering
  - Available in weeks

- Classification
  - In progress

- http://www.manning.com/owen/

Collaborative Filtering at Scale
Questions?

- Gmail: srowen
- user@mahout.apache.org
- http://mahout.apache.org