Item Based Recommenders

Fall 2010
Item Based Recommendations

Produced by grouping similar items together, rather than users

Item Based Algorithm
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

User Based Algorithm
for every item i that u has no preference for yet
  for every other user v that has a preference for i
    compute a similarity s between u and v
    add v's preference for i, weighted by s, to a running average
return the top items, ranked by weighted average
Users and items

Users

U1

U2

U3

U4

Items

I1

I2

I3

I4

I5

I6
Users relate with items

- When an item is bought, clicked on, or recommended by a user that item can be associated with the user.

**Users**

- U1
- U2
- U3
- U4

**Items**

- I1: U1, U2
- I2: U2, U3, U4
- I3: U1, U4
- I4: U3
- I5: U3
- I6: U1, U2, U4
Items grouped

- Items I1 and I6 share two users, as do items I2 and I6 and items I3 and I6
- Items I1, I2, I3, and I6 are similar items

Users

- U1
- U2
- U3
- U4

Items

- I1: U1, U2
- I2: U2, U3, U4
- I3: U1, U4
- I4: U3
- I5: U3
- I6: U1, U2, U4
Items grouped

- Groups can be used to recommend items
- Items I1 and I6 are similar so we can recommend I1 to U4
- Items I2 and I6 are similar so we can recommend I2 to U1 and I6 to U3
- Items could be similar for other reasons than sharing users
Item Based vs. User Based

- Recommenders scale with the number of items or users they must deal with, so there are scenarios in which each type can perform better than the other.
- Similarity estimates between items are more likely to converge over time than similarities between users.
- We can compute and cache similarities that converge, which can give item based recommenders a performance advantage.
- Item based recommenders begin with a list of a user's preferred items and therefore do not need a nearest item neighborhood as user based recommenders do.
// open the data set
DataModel model = new FileDataModel(new File("data"));

// create an ItemSimilarity object for testing similarity
ItemSimilarity sim = new LogLikelihoodSimilaritySimilarity(model);

// Create an Item Based recommender using the model and log likelihood similarity measure
Recommender recommender = new GenericItemBasedRecommender(model, sim);
// no neighborhood is necessary as with user-based similarity

// produce numRecommendations for userId
for (RecommendedItem recommendation :
    recommender.recommend(userId, numRecommendations))
{
    System.out.println(recommendation);
}

Comparing Similarity Measures

- Training/testing split is 70/30
- Data set is a subset of the grouplens 10m set

<table>
<thead>
<tr>
<th></th>
<th>Euclidean Distance</th>
<th>Pearson Correlation</th>
<th>Tanimoto Coefficient</th>
<th>Log Likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run time</td>
<td>1s</td>
<td>1s</td>
<td>4s</td>
<td>3s</td>
</tr>
<tr>
<td>Item-Based</td>
<td>0.85</td>
<td>0.92</td>
<td>0.80</td>
<td>0.80</td>
</tr>
<tr>
<td>Average Absolute Distance</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User-Based</td>
<td>0.86</td>
<td>0.82</td>
<td>0.85</td>
<td>0.80</td>
</tr>
<tr>
<td>Average Absolute Distance</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Slope One Recommender

- Slope one [1] uses the difference in user ratings between items to predict a user's rating
- Does not use a similarity measure
- Preprocessing step calculates the average difference in rating between every pair of items
- Recommendation step uses these differences to predict a user’s rating for a given item
Slope One Recommender

**Preprocessing Algorithm**
for every item i
  for every other item j
    for every user u expressing a preference for both i and j
      add the difference in u’s preference for i and j to an average

**Recommendation Algorithm**
for every item i the user u expresses no preference for
  for every item j that user u expresses a preference for
    find the average preference difference between j and i
    add this diff to u’s preference value for j
    add this to a running average
return the top items, ranked by these averages
Slope One Recommender - Example

- Begin by computing the average preference value difference between all item pairs
- Items 102 and 101: \( \frac{(3.5 - 5) + (5 - 2) + (3.5 - 4.5)}{3} = \frac{0.5}{3} \)
- Items 103 and 101: \( \frac{(4 - 2) + (1 - 4.5)}{2} = \frac{-1.5}{2} \)
- Items 104 and 101: \( \frac{(2 - 2) + (4 - 4.5)}{2} = \frac{-0.5}{2} \)
- Items 103 and 102: \( \frac{(4 - 5) + (1 - 3.5)}{2} = \frac{-3.5}{2} \)
- And so on...

<table>
<thead>
<tr>
<th></th>
<th>101</th>
<th>102</th>
<th>103</th>
<th>104</th>
</tr>
</thead>
<tbody>
<tr>
<td>User X</td>
<td>5</td>
<td>3.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Y</td>
<td>2.0</td>
<td>5.0</td>
<td>4.0</td>
<td>2.0</td>
</tr>
<tr>
<td>User Z</td>
<td>4.5</td>
<td>3.5</td>
<td>1</td>
<td>4.0</td>
</tr>
</tbody>
</table>
Slope One Recommender - Example

- When done we have a table we can use to look up the average difference between any two items
- This completes the preprocessing step
- Empty cells contain inverses that are omitted here

<table>
<thead>
<tr>
<th>Item</th>
<th>101</th>
<th>102</th>
<th>103</th>
<th>104</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>0.17</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>-0.75</td>
<td>-1.75</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>104</td>
<td>-0.25</td>
<td>-1.25</td>
<td>0.5</td>
<td>-</td>
</tr>
</tbody>
</table>
Slope One Recommender - Example

- Let’s recommend an item for user X
- There are two potential candidates: item 103 and item 104
- We want to predict X’s preferences for both items and recommend the one user X would prefer
- We need to do this using all of X’s existing items: 101,102

<table>
<thead>
<tr>
<th></th>
<th>101</th>
<th>102</th>
<th>103</th>
<th>104</th>
</tr>
</thead>
<tbody>
<tr>
<td>User X</td>
<td>5</td>
<td>3.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Y</td>
<td>2.0</td>
<td>5.0</td>
<td>4.0</td>
<td>2.0</td>
</tr>
<tr>
<td>User Z</td>
<td>4.5</td>
<td>3.5</td>
<td>1</td>
<td>4.0</td>
</tr>
</tbody>
</table>
Slope One Recommender - Example

- Predict X's preference for item 103 using item 101
  - Look up the pre-computed average preference difference between items 103 and 101: -0.75
  - Use this to predict X’s rating for item 103 based on item 101
    - X’s rating for item 101: 5
    - X’s predicted preference for item 103 using 101: -0.75 + 5 = 4.25

<table>
<thead>
<tr>
<th>Item</th>
<th>101</th>
<th>102</th>
<th>103</th>
<th>104</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>0.17</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>-0.75</td>
<td>-1.75</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>104</td>
<td>-0.25</td>
<td>-1.25</td>
<td>0.5</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>101</th>
<th>102</th>
<th>103</th>
<th>104</th>
</tr>
</thead>
<tbody>
<tr>
<td>User X</td>
<td>5</td>
<td>3.5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Slope One Recommender - Example

- Predict X's preference for item 103 using item 102
  - Look up the pre-computed average preference difference between items 103 and 102: -1.75
  - Use this to predict X’s rating for item 103 based on item 102
    - X’s rating for item 102: 3.5
    - X’s predicted preference for item 103 using 102: -1.75 + 3.5 = 1.75

<table>
<thead>
<tr>
<th>Item</th>
<th>101</th>
<th>102</th>
<th>103</th>
<th>104</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>102</td>
<td>0.17</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>103</td>
<td>-0.75</td>
<td>-1.75</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>104</td>
<td>-0.25</td>
<td>-1.25</td>
<td>0.5</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User X</th>
<th>101</th>
<th>102</th>
<th>103</th>
<th>104</th>
</tr>
</thead>
<tbody>
<tr>
<td>User X</td>
<td>5</td>
<td>3.5</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Slope One Recommender - Example

- Predict X's preference for item 103
  - Preference for item 103 based on item 101: 4.25
  - Preference for item 103 based on item 102: 1.75
  - Averaging these predictions together gives us a final prediction of X's preference value for item 103

\[
\frac{4.25 + 1.75}{2} = 3
\]
Slope One Recommender - Example

Next predict X's preference for item 104 using 101

- Look up the pre-computed average preference difference between items 104 and 101: -0.25
- Use this to predict X’s rating for item 104 based on item 101
  - X’s rating for item 101: 5
  - X’s predicted preference for item 104: -0.25 + 5 = 4.75

<table>
<thead>
<tr>
<th>Item</th>
<th>101</th>
<th>102</th>
<th>103</th>
<th>104</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>0.17</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>-0.75</td>
<td>-1.75</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>104</td>
<td>-0.25</td>
<td>-1.25</td>
<td>0.5</td>
<td>-</td>
</tr>
<tr>
<td>User X</td>
<td>5</td>
<td>3.5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Slope One Recommender - Example

• Predict X's preference for item 104 using 102
  • Look up the pre-computed average preference difference between items 104 and 102: \(-1.25\)
  • Use this to predict X’s rating for item 104 based on item 102
  • X’s rating for item 102: 3.5
  • X’s predicted preference for item 104: \(-1.25 + 3.5 = 2.25\)

<table>
<thead>
<tr>
<th>Item</th>
<th>101</th>
<th>102</th>
<th>103</th>
<th>104</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>0.17</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>-0.75</td>
<td>-1.75</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>104</td>
<td>-0.25</td>
<td>-1.25</td>
<td>0.5</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>101</th>
<th>102</th>
<th>103</th>
<th>104</th>
</tr>
</thead>
<tbody>
<tr>
<td>User X</td>
<td>5</td>
<td>3.5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Slope One Recommender - Example

- Predict X's preference for item 104
  - Preference for item 104 based on item 101: 4.75
  - Preference for item 104 based on item 102: 2.25
  - Averaging these predictions together gives us a final prediction of X's preference value for item 104
    \[
    \frac{4.75 + 2.25}{2} = 3.5
    \]
- Now make a recommendation based on the predicted values
  - Item 103: 3
  - Item 104: 3.5
  - Item 104 has the highest predicted preference value
  - Recommend item 104 to user X
private float doEstimatePreference(long userID, long itemID) {
    double count = 0.0, totalPreference = 0.0;
    PreferenceArray prefs = getDataModel().getPreferencesFromUser(userID);
    RunningAverage[] averages = diffStorage.getDiffs(userID, itemID, prefs);
    int size = prefs.length();

    // loop through all existing preferences and compute running avg.
    for (int i = 0; i < size; i++) {
        RunningAverage averageDiff = averages[i];
        if (averageDiff != null) {
            double averageDiffValue = averageDiff.getAverage();
            totalPreference += prefs.getValue(i) + averageDiffValue;
            count += 1.0;
        }
    }

    return (float) (totalPreference / count);
}
Slope One - Weighting

- We haven’t considered how much a preference value varies
- Items with less variance in their preference values should be weighted higher than items with high variance
- Use the standard deviation of an item’s average preference
- \( w_i = \frac{c_i}{1+\sigma_i} \), where
  - \( c_i \) = the number of users expressing a preference for item \( i \)
  - \( \sigma_i \) = the standard deviation of the average preference for \( i \)
- Users’ estimated preference values are multiplied by this weight when a recommendation is calculated
SVD Recommender

Singular value decomposition [2] factors a m x n matrix M of rank r into three matrices of sizes m x m, m x n, and n x n

\[ U = \begin{pmatrix} u_{1,1} & u_{1,2} \\ u_{2,1} & u_{2,2}' \end{pmatrix}, \quad \Sigma = \begin{pmatrix} \Sigma_1 & 0 & 0 \\ 0 & \Sigma_2 & 0' \end{pmatrix}, \quad V^T = \begin{pmatrix} v_{1,1} & v_{1,2} & v_{1,3} \\ v_{2,1} & v_{2,2} & v_{2,3} \\ v_{3,1} & v_{3,2} & v_{3,3} \end{pmatrix} \]

- The matrices are then used to produce a lower dimensional representation of the underlying data, which can be thought of as features of the original data
- Neighborhoods and recommendations are then computed using the lower dimensional data
- Considering features rather than individual items can help with sparse data
- SVDRecommender takes the number of features to target and the number of steps to run as arguments
SVD Recommender

Singular value decomposition [2] factors a $m \times n$ matrix $M$ of rank $r$ into three matrices of sizes $m \times m$, $m \times n$, and $n \times n$

- $M = U \Sigma V^T$
- A lower dimensional representation of the data is created by removing all but the $r$ largest singular values from $\Sigma$
- This representation can be seen as an approximation of $M$
- With Mahout the rank $r$ is passed to the SVD Recommender as the number of features to target
- The lower the rank the fewer nonzero values in $\Sigma$
Num Features is number of singular values to keep. If we think there are 20 categories of users, we might set this to 20. A value of 10 for num steps is often reasonable.

DataModel model = new FileDataModel(new File("data"));
Recommender recommender =
  new SVDRecommender(model, numFeatures, numSteps);

for (RecommendedItem recommendation :
  recommender.recommend(userId,numRecommend))
{
  System.out.println(recommendation);
}
One pass Clustering

- Choose a user and declare it to be in a cluster of size one.
- Now compute distance from this cluster to all remaining users.
- Add “closest” node to the cluster. If no node is really close (within some threshold), start a new cluster between the two closest nodes.
Example (One pass Clustering)

- Choose user A as the first cluster
- Now compute similarity coefficient (SC) as SC(A,B), SC(A,C) and SC(A,D). B is the closest so we now cluster B.
Example

- Now we treat AB as a single cluster but to measure its similarity compute the distance \( \min(\text{SC}(A, E), \text{SC}(B, E)) \) to compute \( \text{SC}(AB, E) \).

- Let's assume it's too far from AB to E, D, and C. So now we choose one of these non-clustered element and place it in a cluster. Let's choose E.
Example

- Now we compute the distance from E to D and E to C. E to D is closer so we form a cluster of E and D.
Example (Cont’d)

• Now we compute the centroid of D and E, which is DE.
• Now we compute the distance from DE to C, $SC(\text{DE}, \text{C})$. It is within the threshold so we now include C in this cluster.
Cluster Recommender

Cluster based recommenders [3] operate by grouping users

- Recommendations are made for the entire cluster, not for individual users as with previous methods
- Can work well for new users with few preferences
- Similarity between clusters is defined as either
  - the similarity between the most similar users in a cluster
  - the similarity between the least similar users in a cluster
- Mahout requires as an argument either the number of clusters to create or a cluster similarity threshold
Cluster Recommender - Example

- Euclidean Distance similarity, and a target of 2 clusters
- Begin by creating one cluster for each user

<table>
<thead>
<tr>
<th></th>
<th>101</th>
<th>102</th>
<th>103</th>
<th>104</th>
<th>105</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>5.0</td>
<td>3.5</td>
<td>2.5</td>
<td></td>
<td>4.0</td>
</tr>
<tr>
<td>U2</td>
<td>2.0</td>
<td>5.0</td>
<td>5.0</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td>U3</td>
<td>4.5</td>
<td>3.5</td>
<td></td>
<td>4.0</td>
<td>5.0</td>
</tr>
<tr>
<td>U4</td>
<td>1.0</td>
<td>4.0</td>
<td></td>
<td>4.5</td>
<td>1.5</td>
</tr>
<tr>
<td>U5</td>
<td>2.5</td>
<td>4.0</td>
<td></td>
<td></td>
<td>2.0</td>
</tr>
<tr>
<td>U6</td>
<td>3.5</td>
<td></td>
<td></td>
<td>4.0</td>
<td>4.0</td>
</tr>
<tr>
<td>U7</td>
<td></td>
<td></td>
<td>5.0</td>
<td>4.0</td>
<td></td>
</tr>
</tbody>
</table>
Cluster Recommender - Example

- We have more than two clusters, so merge two clusters
- Find the most similar pair of clusters by finding the Euclidean Distance between each pair of users in the clusters
- Recall that the distance is computed as $D(X, Y) = \frac{n}{1 + \sqrt{\sum_i (x_i - y_i)^2}}$

<table>
<thead>
<tr>
<th>Distance</th>
<th>U1</th>
<th>U2</th>
<th>U3</th>
<th>U4</th>
<th>U5</th>
<th>U6</th>
<th>U7</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>-</td>
<td>0.58</td>
<td>1.0</td>
<td>0.55</td>
<td>0.71</td>
<td>0.8</td>
<td>1.0</td>
</tr>
<tr>
<td>U2</td>
<td>-</td>
<td>0.66</td>
<td>1.0</td>
<td>0.94</td>
<td>0.57</td>
<td>0.25</td>
<td></td>
</tr>
<tr>
<td>U3</td>
<td>-</td>
<td></td>
<td>0.56</td>
<td>0.65</td>
<td>1.0</td>
<td>0.83</td>
<td></td>
</tr>
<tr>
<td>U4</td>
<td>-</td>
<td></td>
<td></td>
<td>0.8</td>
<td>0.44</td>
<td>0.22</td>
<td></td>
</tr>
<tr>
<td>U5</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td>0.62</td>
<td>0.33</td>
<td></td>
</tr>
<tr>
<td>U6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>U7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Cluster Recommender - Example

- Similarity between two clusters is defined as the furthest distance between a pair of users in the two clusters.
- Three one-member clusters have a similarity of 1.0.
- Start by merging the clusters containing U1 and U3.

![Diagram of clusters]

U1, U3  U2  U4  U5  U6  U7
Cluster Recommender - Example

- We know that similarity can’t be greater than 1.0, so we can also now merge the other two clusters with a similarity of 1.0

<table>
<thead>
<tr>
<th>Distance</th>
<th>U1</th>
<th>U2</th>
<th>U3</th>
<th>U4</th>
<th>U5</th>
<th>U6</th>
<th>U7</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>-</td>
<td>0.58</td>
<td>1.0</td>
<td>0.55</td>
<td>0.71</td>
<td>0.8</td>
<td>1.0</td>
</tr>
<tr>
<td>U2</td>
<td>-</td>
<td>0.66</td>
<td>1.0</td>
<td>0.94</td>
<td>0.57</td>
<td>0.25</td>
<td></td>
</tr>
<tr>
<td>U3</td>
<td>-</td>
<td>0.56</td>
<td>0.65</td>
<td>1.0</td>
<td>0.83</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U4</td>
<td>-</td>
<td>0.8</td>
<td>0.44</td>
<td>0.22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U5</td>
<td>-</td>
<td>0.62</td>
<td>0.33</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U6</td>
<td>-</td>
<td>1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U7</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Cluster Recommender - Example

• Compute similarity again using the new clusters and merge the two most similar clusters: \([U2,U4]\) and \(U5\)

<table>
<thead>
<tr>
<th></th>
<th>(U1, U3)</th>
<th>(U2, U4)</th>
<th>(U5)</th>
<th>(U6, U7)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(U1, U3)</td>
<td>-</td>
<td>0.55</td>
<td>0.65</td>
<td>0.8</td>
</tr>
<tr>
<td>(U2, U4)</td>
<td>-</td>
<td></td>
<td>0.8</td>
<td>0.22</td>
</tr>
<tr>
<td>(U5)</td>
<td></td>
<td>-</td>
<td></td>
<td>0.33</td>
</tr>
<tr>
<td>(U6, U7)</td>
<td></td>
<td></td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
• Compute similarity again using the new clusters and merge the two most similar clusters: [U1,U3] and [U6,U7]
• We have reached the target of 2 clusters and can stop merging

<table>
<thead>
<tr>
<th></th>
<th>U1,U3</th>
<th>U2,U4,U5</th>
<th>U6,U7</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1,U3</td>
<td>-</td>
<td>0.55</td>
<td>0.8</td>
</tr>
<tr>
<td>U2,U4,U5</td>
<td>0.55</td>
<td>-</td>
<td>0.22</td>
</tr>
<tr>
<td>U6,U7</td>
<td></td>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>
Cluster Recommender - Example

• Let’s recommend an item for U7
• Find the items with the highest average preference within each cluster; for U7’s missing items the average preferences are
  • Item 101: $\frac{5+4.5+3.5}{3} = 4.3$
  • Item 102: $\frac{3.5+3.5}{2} = 3.5$
  • Item 103: $\frac{2.5}{1} = 2.5$
• Recommend item 101 to U7 with an estimated preference value of 4.3

<table>
<thead>
<tr>
<th></th>
<th>101</th>
<th>102</th>
<th>103</th>
<th>104</th>
<th>105</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>5.0</td>
<td>3.5</td>
<td>2.5</td>
<td></td>
<td>4.0</td>
</tr>
<tr>
<td>U3</td>
<td>4.5</td>
<td>3.5</td>
<td></td>
<td>4.0</td>
<td>5.0</td>
</tr>
<tr>
<td>U6</td>
<td>3.5</td>
<td></td>
<td></td>
<td>4.0</td>
<td>4.0</td>
</tr>
<tr>
<td>U7</td>
<td></td>
<td></td>
<td></td>
<td>5.0</td>
<td>4.0</td>
</tr>
</tbody>
</table>
DataModel model = new FileDataModel(new File("data"));

// Use Euclidean Distance for user pair similarity
UserSimilarity usim = new
  EuclideanDistanceSimilaritySimilarity(model);

// Farthest neighbor cluster similarity measure with similarity uSim
ClusterSimilarity clusterSim =
  new FarthestNeighborClusterSimilaritySimilarity(usim);

// Create a recommender
Recommender recommender = new
  TreeClusteringRecommender(model, clusterSim, numCluster);
Comparing Recommenders

- Training/testing split is 70/30
- Data set is 2m records taken from the grouplens 10m set
- SVD: 10 features and 10 steps
- Tree Clustering: 100 clusters

<table>
<thead>
<tr>
<th></th>
<th>Average Absolute Distance</th>
<th>Precision</th>
<th>Recall</th>
<th>Build run time</th>
<th>Evaluation run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slope One</td>
<td>0.75</td>
<td>0.07</td>
<td>0.10</td>
<td>1s</td>
<td>189s</td>
</tr>
<tr>
<td>SVD</td>
<td>0.78</td>
<td>0.14</td>
<td>0.19</td>
<td>1s</td>
<td>149s</td>
</tr>
<tr>
<td>Tree Clustering</td>
<td>0.96</td>
<td>0.07</td>
<td>0.24</td>
<td>34s</td>
<td>102353s (1d 4h+)</td>
</tr>
</tbody>
</table>
# Recommender Summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Parameters</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>User based</td>
<td>user similarity metric, user neighborhood</td>
<td>Fast when there are fewer users than items</td>
</tr>
<tr>
<td>Item based</td>
<td>item similarity metric</td>
<td>Fast when there are fewer items than users, useful with a specialized item similarity definition</td>
</tr>
<tr>
<td>Slope one</td>
<td></td>
<td>Fast at runtime, slow to precompute, good with low item numbers</td>
</tr>
<tr>
<td>SVD</td>
<td>number of target features</td>
<td>Slow to precompute</td>
</tr>
<tr>
<td>Tree clustering</td>
<td>user similarity metric, cluster similarity metric, number of clusters</td>
<td>Fast at runtime, slow to precompute, good with low user numbers</td>
</tr>
</tbody>
</table>
References

D. Lemire and A. Maclachlan, Slope One Predictors for Online Rating-Based Collaborative Filtering. In Proceedings SIAM Data Mining (SDM'05), 2005.
