Lec 6: Recommender Systems 2
Collaborative Filtering
Content-based filtering

• Idea: a user is likely to have similar level of interest for similar items

• System learns importance of item features, and builds a model of what user likes

QUESTION: Where is the User Model Stored?
Problems with Content-based Filtering

• Need to know about item content
  – requires manual or automatic indexing
  – Item features do not capture everything

• “User cold-start” problem
  – Needs to learn what content features are important for the user, so takes time

• What if user’s interests change?
Problems with Content-based Filtering

• Lack of serendipity
  [Wikipedia: “the effect by which one accidentally discovers something fortunate, especially while looking for something entirely unrelated”]
QUESTION: How do you decide which movie to watch
What is Collaborative Filtering?

- Community of users
- To predict a user’s opinion, use the opinions of others
- Advantages:
  - No need to analyse (index) content
  - Can capture more subtle things
  - Serendipity
Collaborative Filtering Effectiveness

- Recommending And Evaluating Choices In A Virtual Community Of Use. Will Hill, Larry Stead, Mark Rosenstein and George Furnas, Bellcore; CHI 1995
  - Evaluates Collaborative predictions of movie ratings with predictions based on movie reviews
Figure 3  Two Scatterplots of Actual Ratings by Predicted Ratings. Plot on left shows movie critics as predictor ($r=0.22$). Plot on right shows virtual community as predictor ($r=0.62$) (all values are jittered for the purpose of visual presentation, 3269 predictions each for 291 users)
Types of Collaborative Filtering

- User-based collaborative filtering
- Item-based collaborative filtering
User-based Collaborative Filtering

- People who agreed in the past are likely to agree again
- To predict a user’s opinion for an item, use the opinion of similar users
- Similarity between users is decided by looking at their overlap in opinions for other items
Ex: User-based Collaborative Filtering

<table>
<thead>
<tr>
<th></th>
<th>Item 1</th>
<th>Item 2</th>
<th>Item 3</th>
<th>Item 4</th>
<th>Item 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>8</td>
<td>1</td>
<td>?</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>User 2</td>
<td>2</td>
<td>?</td>
<td>5</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>User 3</td>
<td>5</td>
<td>4</td>
<td>7</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>User 4</td>
<td>7</td>
<td>1</td>
<td>7</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>User 5</td>
<td>1</td>
<td>7</td>
<td>4</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>User 6</td>
<td>8</td>
<td>3</td>
<td>8</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>
### Similarity between users

<table>
<thead>
<tr>
<th></th>
<th>Item 1</th>
<th>Item 2</th>
<th>Item 3</th>
<th>Item 4</th>
<th>Item 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>8</td>
<td>1</td>
<td>?</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>User 2</td>
<td>2</td>
<td>?</td>
<td>5</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>User 4</td>
<td>7</td>
<td>1</td>
<td>7</td>
<td>3</td>
<td>8</td>
</tr>
</tbody>
</table>

- How similar are users 1 and 2?
- How similar are users 1 and 5?
- How do you calculate similarity?
Similarity between users: simple way

<table>
<thead>
<tr>
<th>Item 1</th>
<th>Item 2</th>
<th>Item 3</th>
<th>Item 4</th>
<th>Item 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>8</td>
<td>1</td>
<td>?</td>
<td>2</td>
</tr>
<tr>
<td>User 2</td>
<td>2</td>
<td>?</td>
<td>5</td>
<td>7</td>
</tr>
</tbody>
</table>

- Only consider items both users have rated
- For each item:
  - Calculate difference in the users’ ratings
  - Take the average of this difference over the items

\[
\text{Sim}(\text{User1}, \text{User2}) = \frac{\sum_{j} | \text{rating (User1, Item j)} - \text{rating (User2, Item j)} |}{\text{Num. of items}}
\]"
### Problems: Similarity between users

<table>
<thead>
<tr>
<th></th>
<th>Item 1</th>
<th>Item 2</th>
<th>Item 3</th>
<th>Item 4</th>
<th>Item 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>User 2</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

$$\text{Sim(}\text{User1, User2}) = \frac{12}{5} = 2.4$$

<table>
<thead>
<tr>
<th></th>
<th>Item 1</th>
<th>Item 2</th>
<th>Item 3</th>
<th>Item 4</th>
<th>Item 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>User 4</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

$$\text{Sim(}\text{User3, User4}) = \frac{15}{5} = 3$$
Better Solution

• Use Statistical Correlation Metrics (e.g., Pearson's)
  – These measure how well two data sets fit on a straight line
  – Corrects for grade inflation

Perfect Correlation for User3, User4

Inverse Correlation for User1, User2
Similarity to Recommendations

• Similarity provides a ranking for other users, or a weight to associate with each user
  – Identify Similar Users, and recommend what they have rated highly
  – To calculate rating of an item to recommend, give weight to each user's recommendations based on how similar they are to you.
Algorithm 1: using entire matrix

Aggregation function: often weighted sum

Weight depends on similarity
Handling Large Matrices

• MovieLens database
  – 100k dataset = 1682 movies & 943 users
  – 1mn dataset = 3900 movies & 6040 users

• Netflix dataset
  – 17700 movies, 250k users, 100 million ratings

Realistically, cannot make use of all users in real time
Algorithm 2: K-Nearest-Neighbour

Neighbours are people who have historically had the same taste as our user.

Aggregation function: often weighted sum

Weight depends on similarity.
Problems with User-based Collaborative Filtering (1)

- User Cold-Start problem
  not enough known about new user to decide who is similar (and perhaps no other users yet..)

- Need way to motivate early rater
Problems with User-based Collaborative Filtering (2)

• Sparsity
  when recommending from a large item set, users will have rated only some of the items
  (makes it hard to find similar users)
Problems with User-based Collaborative Filtering (3)

• Scalability
  – with millions of ratings, computations become slow

• Item Cold-Start problem
  – Cannot predict ratings for new item till some similar users have rated it [No problem for content-based]
Demographic Recommenders

• To predict a user’s opinion for an item, use the opinion of similar users [as in user-based Collaborative Filtering]
• But, similarity between users is decided by looking at demographics (stereotypes)
• Otherwise, default to all users
  – Most popular lists, etc.
Item-based Collaborative Filtering

• User is likely to have the same opinion for similar items [same idea as in Content-Based Filtering]
• Similarity between items is decided by looking at how other users have rated them [different from Content-based, where item features are used]
  Star Wars = [Action, Sci-fi...]
  Star Wars = [User1:8, User2:3, User3:7...]
• Advantage (compared to user-based CF):
  – Prevents User Cold-Start problem
  – Improves scalability (similarity between items is more stable than between users)
### Example:
**Item-based Collaborative Filtering**

<table>
<thead>
<tr>
<th></th>
<th>Item 1</th>
<th>Item 2</th>
<th>Item 3</th>
<th>Item 4</th>
<th>Item 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>8</td>
<td>1</td>
<td>?</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>User 2</td>
<td>2</td>
<td>?</td>
<td>5</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>User 3</td>
<td>5</td>
<td>4</td>
<td>7</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>User 4</td>
<td>7</td>
<td>1</td>
<td>7</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>User 5</td>
<td>1</td>
<td>7</td>
<td>4</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>User 6</td>
<td>8</td>
<td>3</td>
<td>8</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>
## Similarity between items

<table>
<thead>
<tr>
<th>Item 3</th>
<th>Item 4</th>
<th>Item 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>

- How similar are items 3 and 4?
- How similar are items 3 and 5?
- How do you calculate similarity?
Similarity between items: simple way

- Only consider users who have rated both items
- For each user:
  - Calculate difference in ratings for the two items
  - Take the average of this difference over the users

\[
\text{Sim(Item 3, Item 4) = } \frac{\sum_j | \text{rating (User j, Item 3)} - \text{rating (User j, Item 4)} |}{\text{Number of Users}}
\]

<table>
<thead>
<tr>
<th>Item 3</th>
<th>Item 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
</tr>
</tbody>
</table>
Algorithms

• As User-Based: can use nearest-neighbours or all

Aggregation function: often weighted sum

Weight depends on similarity
Problems with Item-based Collaborative Filtering (1)

• Item Cold-Start problem
  Cannot predict which items are similar till we have ratings for this item
  [No problem for content-based.
   Also a problem for User-based collaborative filtering, but it is a bigger problem here.]
Hybrid Recommender Systems

• Use a combination of Content-based and Collaborative Filtering
• Or a combination of User-based and Item-based Collaborative Filtering
• Why would you want to do this?
Some ways to make a Hybrid

- **Weighted.** Ratings of several recommendation techniques are combined together to produce a single recommendation.
- **Switching.** The system switches between recommendation techniques depending on the current situation.
- **Mixed.** Recommendations from several different recommenders are presented simultaneously (e.g. Amazon).
- **Cascade.** One recommender refines the recommendations given by another.
Disclaimer

• There is a LOT of work on recommender systems
• I have simplified things and left things out....
Linear Algebra

General Class of Problems: Matrix with many missing values:

- How to fill in the missing values?
- How to efficiently handle large and sparse matrices?