COMP 465: Data Mining
Recommender Systems

Slides Adapted From: www.mmds.org (Mining Massive Datasets)

Example: Recommender Systems

- **Customer X**
  - Buys Metallica CD
  - Buys Megadeth CD

- **Customer Y**
  - Does search on Metallica
  - Recommender system suggests Megadeth from data collected about customer X

Recommendations

<table>
<thead>
<tr>
<th>Search</th>
<th>Recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items</td>
<td>Products, web sites, blogs, news items, …</td>
</tr>
</tbody>
</table>

Examples:

- amazon.com
- StumbleUpon
- del.icio.us
- movielens
- last.fm
- Google
- YouTube
- XBOX
- LIVE

From Scarcity to Abundance

- **Shelf space is a scarce commodity for traditional retailers**
  - Also: TV networks, movie theaters,…

- **Web enables near-zero-cost dissemination of information about products**
  - From scarcity to abundance

- **More choice necessitates better filters**
  - Recommendation engines
  - How *Into Thin Air* made *Touching the Void* a bestseller: [http://www.wired.com/wired/archive/12.10/tail.html](http://www.wired.com/wired/archive/12.10/tail.html)
Sidenote: The Long Tail

- Editorial and hand curated
  - List of favorites
  - Lists of “essential” items
- Simple aggregates
  - Top 10, Most Popular, Recent Uploads
- Tailored to individual users
  - Amazon, Netflix, ...

Formal Model

- $X =$ set of Customers
- $S =$ set of Items
- Utility function $u: X \times S \rightarrow R$
  - $R =$ set of ratings
  - $R$ is a totally ordered set
  - e.g., 0-5 stars, real number in $[0,1]$

Utility Matrix

<table>
<thead>
<tr>
<th></th>
<th>Avatar</th>
<th>LOTR</th>
<th>Matrix</th>
<th>Pirates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>0.2</td>
<td>0.3</td>
<td>0.4</td>
</tr>
<tr>
<td>Bob</td>
<td>0.5</td>
<td>0.3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Carol</td>
<td>0.2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>David</td>
<td></td>
<td></td>
<td>0.4</td>
<td></td>
</tr>
</tbody>
</table>
Key Problems

- **(1) Gathering “known” ratings for matrix**
  - How to collect the data in the utility matrix

- **(2) Extrapolate unknown ratings from the known ones**
  - Mainly interested in high unknown ratings
  - We are not interested in knowing what you don’t like but what you like

- **(3) Evaluating extrapolation methods**
  - How to measure success/performance of recommendation methods

(1) Gathering Ratings

- **Explicit**
  - Ask people to rate items
  - Doesn’t work well in practice – people can’t be bothered

- **Implicit**
  - Learn ratings from user actions
  - E.g., purchase implies high rating
  - What about low ratings?

(2) Extrapolating Utilities

- **Key problem:** Utility matrix $U$ is sparse
  - Most people have not rated most items
  - **Cold start:**
    - New items have no ratings
    - New users have no history

- **Three approaches to recommender systems:**
  - 1) Content-based
  - 2) Collaborative
  - 3) Latent factor based

Content-based Recommender Systems
**Main idea:** Recommend items to customer $x$ similar to previous items rated highly by $x$

**Example:**
- **Movie recommendations**
  - Recommend movies with same actor(s), director, genre, ...
- **Websites, blogs, news**
  - Recommend other sites with "similar" content

**Item Profiles**

- For each item, create an **item profile**
- **Profile is a set (vector) of features**
  - **Movies:** author, title, actor, director,...
  - **Text:** Set of "important" words in document
- **How to pick important features?**
  - Usual heuristic from text mining is TF-IDF
  - **Term ... Feature**
  - **Document ... Item**

**Sidenote: TF-IDF**

$f_{ij} = \text{frequency of term (feature) } i \text{ in doc (item) } j$

$TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}}$

$n_i = \text{number of docs that mention term } i$

$N = \text{total number of docs}$

$IDF_i = \log \frac{N}{n_i}$

**TF-IDF score:** $w_{ij} = TF_{ij} \times IDF_i$

**Doc profile** = set of words with highest TF-IDF scores, together with their scores
User Profiles and Prediction

- User profile possibilities:
  - Weighted average of rated item profiles
  - **Variation:** weight by difference from average rating for item
- ... **Prediction heuristic:**
  - Given user profile \( \mathbf{x} \) and item profile \( \mathbf{i} \), estimate
    \[
    u(\mathbf{x}, \mathbf{i}) = \cos(\mathbf{x}, \mathbf{i}) = \frac{\mathbf{x} \cdot \mathbf{i}}{||\mathbf{x}|| ||\mathbf{i}||}
    \]

Pros: Content-based Approach

- **+**: No need for data on other users
  - No cold-start or sparsity problems
- **+**: Able to recommend to users with unique tastes
- **+**: Able to recommend new & unpopular items
  - No first-rater problem
- **+**: Able to provide explanations
  - Can provide explanations of recommended items by listing content-features that caused an item to be recommended

Cons: Content-based Approach

- **−**: Finding the appropriate features is hard
  - E.g., images, movies, music
- **−**: Recommendations for new users
  - How to build a user profile?
- **−**: Overspecialization
  - Never recommends items outside user’s content profile
  - People might have multiple interests
  - Unable to exploit quality judgments of other users

Collaborative Filtering

Harnessing quality judgments of other users
Collaborative Filtering

- Consider user $x$
- Find set $N$ of other users whose ratings are \textit{similar} to $x$’s ratings
- Estimate $x$’s ratings based on ratings of users in $N$

Similarity Metric

- Intuitively we want: $\text{sim}(A, B) > \text{sim}(A, C)$
- Jaccard similarity: $1/5 < 2/4$
- Cosine similarity: $0.386 > 0.322$
  - Considers missing ratings as \textit{negative}
  - Solution: subtract the (row) mean

Rating Predictions

From similarity metric to recommendations:
- Let $r_x$ be the vector of user $x$’s ratings
- Let $N$ be the set of $k$ users most similar to $x$ who have rated item $i$
- Prediction for item $s$ of user $x$:
  - $r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$  
  - $r_{xi} = \frac{\sum_{y \in N} s_{xy} r_{yi}}{\sum_{y \in N} s_{xy}}$
  - Other options?
  - Many other tricks possible...

Item-Item Collaborative Filtering

- So far: User-user collaborative filtering
- Another view: Item-item
  - For item $i$, find other similar items
  - Estimate rating for item $i$ based on ratings for similar items
  - Can use same similarity metrics and prediction functions as in user-user model

\[ r_{ui} = \frac{\sum_{j \in N(i, x)} s_{ij} \cdot r_{uj}}{\sum_{j \in N(i, x)} s_{ij}} \]

- $s_{ij}$: similarity of items $i$ and $j$
- $r_{ui}$: rating of user $u$ on item $j$
- $N(i, x)$: set items rated by $x$ similar to $i$
**Item-Item CF (|N|=2)**

- **Users:**
  - 1: 3, 5, 4
  - 2: 5, 4
  - 3: 2, 4, 1, 2, 3, 4, 5
  - 4: 2, 4, 5, 2
  - 5: 4, 3, 2, 2, 5
  - 6: 1, 3, 3, 2, 4

- **Movies:**
  - 1: 3
  - 2: 4
  - 3: 2, 4, 1, 2, 3, 4, 5
  - 4: 2, 4, 5, 2
  - 5: 4, 3, 2, 2, 5
  - 6: 1, 3, 3, 2, 4

- Unknown rating: □
- Rating between 1 to 5: ▶

**Neighbor selection:**
- Identify movies similar to movie 1, rated by user 5

**Compute similarity weights:**
- \( s_{1,3} = 0.41 \), \( s_{1,6} = 0.59 \)

**Item-Item CF (|N|=2)**

- **Users:**
  - 1: 3, 5, 4
  - 2: 5, 4
  - 3: 2, 4, 1, 2, 3, 4, 5
  - 4: 2, 4, 5, 2
  - 5: 4, 3, 2, 2, 5
  - 6: 1, 3, 3, 2, 4

- **Movies:**
  - 1: 3
  - 2: 4
  - 3: 2, 4, 1, 2, 3, 4, 5
  - 4: 2, 4, 5, 2
  - 5: 4, 3, 2, 2, 5
  - 6: 1, 3, 3, 2, 4

- **Sim(1,m):**
  - 1.00
  - -0.18
  - 0.41
  - -0.10
  - -0.31
  - 0.59

**Here we use Pearson correlation as similarity:**
1) Subtract mean rating, \( m_i \), from each movie \( i \):
   \[ m_{1,2,3} = \left( \frac{1+3+5+5+4}{5} \right) = 3.6 \]
2) Compute cosine similarities between rows:
3) Compute similarity weights:
   \[ s_{1,3} = 0.41 \], \[ s_{1,6} = 0.59 \]
Item-Item CF ($|N|=2$)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>3</td>
<td>2.6</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>5</td>
<td>4</td>
<td>4</td>
<td></td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Predict by taking weighted average:

$r_{1,2} = \frac{(0.41*2 + 0.59*3)}{(0.41+0.59)} = 2.6$

Item-Item vs. User-User

- In theory, user-user and item-item are dual approaches.
- In practice, item-item outperforms user-user in many use cases.
- Items are “simpler” than users
  - Items belong to a small set of “genres”, users have varied tastes.
  - Item Similarity is more meaningful than User Similarity

Pros/Cons of Collaborative Filtering

- **+ Works for any kind of item**
  - No feature selection needed
- **- Cold Start:**
  - Need enough users in the system to find a match
- **- Sparsity:**
  - The user/ratings matrix is sparse
  - Hard to find users that have rated the same items
- **- First rater:**
  - Cannot recommend an item that has not been previously rated
  - New items, Esoteric items
- **- Popularity bias:**
  - Cannot recommend items to someone with unique taste
  - Tends to recommend popular items