Recommendation Systems

Stony Brook University
CSE545, Spring 2019
Recommendation Systems

- What other item will this user like? (based on previously liked items)
- How much will user like item X?
Recommendation Systems

- What other item will this **user** like? (based on previously liked items)

How much will user like item X?

?
Recommendation Systems

- What other item will this user like? (based on previously liked items)

- How much will user like item X?
Recommendation Systems
Recommendation Systems

Past User Ratings
Recommendation Systems

Why Big Data?

- Data with many potential features (and sometimes observations)
- An application of techniques for finding similar items
  - locality sensitive hashing
  - dimensionality reduction
Recommendation System: Example

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

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

Recommendations

Items

Products, web sites, blogs, news items, ...

Examples:

- amazon.com
- Pandora
- StumbleUpon
- del.icio.us
- Netflix
- Movielens
  helping you find the right movies
- last.fm
  the social music revolution
- Google News
- YouTube
- Xbox Live
Enabled by Web Shopping

- Does Wal-Mart have everything you need?
Enabled by Web Shopping

- Does Wal-Mart have everything you need?
Enabled by Web Shopping

- Does Wal-Mart have everything you need?
- A lot of products are only of interest to a small population (i.e. “long-tail products”).
- However, most people buy many products that are from the long-tail.

- Web shopping enables more choices
  - Harder to search
  - Recommendation engines to the rescue
Enabled by Web Shopping

- Does Wal-Mart have everything you need?

- A lot of products are only of interest to a small population (i.e. “long-tail products”).

- However, most people buy many products that are from the long-tail.

- Web shopping enables more choices
  - Harder to search
  - Recommendation engines to the rescue

"If you like Britney, you’ll love …"

Just as lower prices can entice consumers down the Long Tail, recommendation engines drive them to obscure content they might not find otherwise.
A Model for Recommendation Systems

Given:  *users*,  *items*,  *utility matrix*
A Model for Recommendation Systems

Given: *users*, *items*, *utility matrix*

<table>
<thead>
<tr>
<th>user</th>
<th>Game of Thrones</th>
<th>Fargo</th>
<th>Brooklyn Nine-Nine</th>
<th>Silicon Valley</th>
<th>Walking Dead</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td></td>
<td></td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>
A Model for Recommendation Systems

Given: users, items, utility matrix

<table>
<thead>
<tr>
<th>user</th>
<th>Game of Thrones</th>
<th>Fargo</th>
<th>Brooklyn Nine-Nine</th>
<th>Silicon Valley</th>
<th>Walking Dead</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td></td>
<td></td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>?</td>
<td>?</td>
<td>5</td>
<td>2</td>
<td>?</td>
</tr>
</tbody>
</table>
Recommendation Systems

Problems to tackle:

1. Gathering ratings

2. Extrapolate unknown ratings
   a. Explicit: based on user ratings and reviews
      (problem: only a few users engage in such tasks)
   b. Implicit: Learn from actions (e.g. purchases, clicks)
      (problem: hard to learn low ratings)

3. Evaluation
Recommendation Systems

Problems to tackle:

1. Gathering ratings
2. Extrapolate unknown ratings
   a. Explicit: based on user ratings and reviews
      (problem: only a few users engage in such tasks)
   b. Implicit: Learn from actions (e.g. purchases, clicks)
      (problem: hard to learn low ratings)
3. Evaluation

Common Approaches

1. Content-based
2. Collaborative
3. Latent Factor
Recommendation Systems

Problems to tackle:

1. Gathering ratings

2. Extrapolate unknown ratings
   a. Explicit: based on user ratings and reviews
      (problem: only a few users engage in such tasks)
   b. Implicit: Learn from actions (e.g. purchases, clicks)
      (problem: hard to learn low ratings)

3. Evaluation

Common Approaches

1. Content-based
2. Collaborative
3. Latent Factor
Utility Matrix:

<table>
<thead>
<tr>
<th></th>
<th>o1</th>
<th>o2</th>
<th>o3</th>
<th>...</th>
<th>oN</th>
</tr>
</thead>
<tbody>
<tr>
<td>movies</td>
<td>f1</td>
<td>f2</td>
<td>f3</td>
<td>f4</td>
<td>...</td>
</tr>
</tbody>
</table>

- rows: N observations
- columns: p features
- p features
- N observations
Goal: Complete Matrix

f1, f2, f3, f4, ...                     fp
o1  o2  o3                         fp
...  ...  ...                         fp
oN  oN  oN                         fp

movies
Problem: Given Incomplete Matrix

Given a matrix with rows representing users (o1, o2, o3, ..., oN) and columns representing movies (f1, f2, f3, f4, ..., fp), where many entries are missing.
Complete Matrix using Latent Factors

Dimensionality reduction
Try to best represent but with on p' columns.
Complete Matrix using Latent Factors

Find latent factors

Reconstruct matrix
Dimensionality Reduction - PCA

Linear approximates of data in \( r \) dimensions.

Found via *Singular Value Decomposition*:

\[
X_{\text{[nxp]}} = U_{\text{[nxr]}} D_{\text{[rxr]}} V_{\text{[pxr]}}^T
\]

\( X \): original matrix, \( U \): “left singular vectors”,
\( D \): “singular values” (diagonal), \( V \): “right singular vectors”

Projection (dimensionality reduced space) in 3 dimensions:

\[
(U_{\text{[nx3]}} D_{\text{[3x3]}} V_{\text{[px3]}}^T)
\]

To reduce features in new dataset:

\[
X_{\text{new}} V = X_{\text{new\_small}}
\]
Dimensionality Reduction - PCA

Linear approximates of data in $r$ dimensions.

Found via *Singular Value Decomposition*:

$$X_{[n \times p]} = U_{[n \times r]} D_{[r \times r]} V_{[p \times r]}^T$$

$X$: original matrix, $U$: “left singular vectors”, $D$: “singular values” (diagonal), $V$: “right singular vectors”
Dimensionality Reduction - PCA - Example

\[ X_{[n \times p]} = U_{[n \times r]} D_{[r \times r]} V_{[p \times r]^T} \]

Users to movies matrix

<table>
<thead>
<tr>
<th>SciFi</th>
<th>Alien</th>
<th>Serenity</th>
<th>Casablanca</th>
<th>Amelie</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User 1</th>
<th>User 2</th>
<th>User 3</th>
<th>User 4</th>
<th>User 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>9.5</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1.3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \end{bmatrix} \times \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \]
Dimensionality Reduction - PCA

Linear approximates of data in \( r \) dimensions.

Found via *Singular Value Decomposition*:

\[
X_{[nxp]} = U_{[nxr]} D_{[rxr]} V^T_{[pxr]}
\]

\( X \): original matrix,
\( U \): “left singular vectors”,
\( D \): “singular values” (diagonal),
\( V \): “right singular vectors”
Dimensionality Reduction - PCA

- **Goal:** Minimize the sum of reconstruction errors:

\[
\sum_{i=1}^{N} \sum_{j=1}^{D} \| x_{ij} - z_{ij} \|^2
\]

- where \( x_{ij} \) are the “old” and \( z_{ij} \) are the “new” coordinates

\( X \): original matrix
\( U \): “left singular vectors”
\( D \): “singular values” (diagonal)
\( V^T \): “right singular vectors”

To check how well the original matrix can be reproduced:

\( Z_{[nxp]} = U D V^T \), How does Z compare to original X?
Dimensionality Reduction - PCA - Example

$$X_{[n \times p]} = U_{[n \times r]} D_{[r \times r]} V_{[p \times r]}^T$$
Recommendation Systems

Problems to tackle:

1. Gathering ratings
2. Extrapolate unknown ratings
   a. Explicit: based on user ratings and reviews
      (problem: only a few users engage in such tasks)
   b. Implicit: Learn from actions (e.g. purchases, clicks)
      (problem: hard to learn low ratings)
3. Evaluation

Common Approaches

1. Content-based
2. Collaborative
3. Latent Factor
Recommendation Systems

Problems to tackle:

1. Gathering ratings

2. Extrapolate unknown ratings
   a. Explicit: based on user ratings and reviews
      (problem: only a few users engage in such tasks)
   b. Implicit: Learn from actions (e.g. purchases, clicks)
      (problem: hard to learn low ratings)

3. Evaluation

Common Approaches

1. Content-based
2. Collaborative
3. Latent Factor
Content-based Rec Systems

Based on similarity of items to past items that they have rated.
Content-based Rec Systems

Based on similarity of items to past items that they have rated.
Content-based Rec Systems

Based on similarity of items to past items that they have rated.

1. **Build profiles of items (set of features); examples:**
   - *shows*: producer, actors, theme, review
   - *people*: friends, posts

   *pick words with tf-idf*
Content-based Rec Systems

Based on similarity of items to past items that they have rated.

1. Build profiles of items (set of features); examples:
   - shows: producer, actors, theme, review
   - people: friends, posts

2. Construct user profile from item profiles; approach:
   - average all item profiles
   - variation: weight by difference from their average

pick words with tf-idf
Content-based Rec Systems

Based on similarity of items to past items that they have rated.

1. Build profiles of items (set of features); examples:
   - *shows*: producer, actors, theme, review
   - *people*: friends, posts
2. Construct user profile from item profiles; approach:
   - average all item profiles of items they’ve purchased
   - variation: weight by difference from their average ratings
3. Predict ratings for new items; approach:

   $$utility(user, i) = \cos(x, i) = \frac{x \cdot i}{||x|| \cdot ||i||}$$
Why Content Based?

- Only need users history
- Captures unique tastes
- Can recommend new items
- Can provide explanations
Why Content Based?

- Only need users history
- Captures unique tastes
- Can recommend new items
- Can provide explanations

- Need good features
- New users don’t have history
- Doesn’t venture “outside the box” (Overspecialized)
Why Content Based?

- Only need users history
- Captures unique tastes
- Can recommend new items
- Can provide explanations

- Need good features
- New users don’t have history
- Doesn’t venture “outside the box”
  (Overspecialized)

(not exploiting other users judgments)
Collaborative Filtering Rec Systems

(not exploiting other users judgments)
Recommendation Systems

Problems to tackle:

1. Gathering ratings

2. Extrapolate unknown ratings
   a. Explicit: based on user ratings and reviews
      (problem: only a few users engage in such tasks)
   b. Implicit: Learn from actions (e.g. purchases, clicks)
      (problem: hard to learn low ratings)

3. Evaluation

Common Approaches

1. Content-based
2. Collaborative
3. Latent Factor
Collaborative Filtering Rec Systems
Collaborative Filtering Rec Systems

<table>
<thead>
<tr>
<th>user</th>
<th>Game of Thrones</th>
<th>Fargo</th>
<th>Brooklyn Nine-Nine</th>
<th>Silicon Valley</th>
<th>Walking Dead</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td></td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>5</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Collaborative Filtering Rec Systems

<table>
<thead>
<tr>
<th>user</th>
<th>Game of Thrones</th>
<th>Fargo</th>
<th>Brooklyn Nine-Nine</th>
<th>Silicon Valley</th>
<th>Walking Dead</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td></td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>5</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

General Idea:

1) Find similar users = “neighborhood”

2) Infer rating based on how similar users rated
Collaborative Filtering Rec Systems

<table>
<thead>
<tr>
<th>user</th>
<th>Game of Thrones</th>
<th>Fargo</th>
<th>Brooklyn Nine-Nine</th>
<th>Silicon Valley</th>
<th>Walking Dead</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Given:** user, x; item, i; utility matrix, u

1. Find neighborhood, N # set of k users most similar to x who have also rated i
## Collaborative Filtering Rec Systems

<table>
<thead>
<tr>
<th>user</th>
<th>Game of Thrones</th>
<th>Fargo</th>
<th>Brooklyn Nine-Nine</th>
<th>Silicon Valley</th>
<th>Walking Dead</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td></td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>5</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Given:** \(\text{user}, \, x; \, \text{item}, \, i; \, \text{utility matrix}, \, u\)

1. Find neighborhood, \(N\) \# set of \(k\) users most similar to \(x\) who have also rated \(i\)

**Two Challenges:** (1) user bias, (2) missing values
Collaborative Filtering Rec Sys S

<table>
<thead>
<tr>
<th>user</th>
<th>Game of Thrones</th>
<th>Fargo</th>
<th>Brooklyn Nine-Nine</th>
<th>Silicon Valley</th>
<th>Walking Dead</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4 =&gt; 0.5</td>
<td>5 =&gt; 1.5</td>
<td>2 =&gt; -1.5</td>
<td>=&gt; 0</td>
<td>3 =&gt; -0.5</td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td></td>
<td></td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>5</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Given: user, x; item, i; utility matrix, u

1. Find neighborhood, N # set of k users most similar to x who have also rated i

Two Challenges: (1) user bias, (2) missing values

Solution: subtract user’s mean, add zeros for missing
Collaborative Filtering Rec Systems

<table>
<thead>
<tr>
<th>user</th>
<th>Game of Thrones</th>
<th>Fargo</th>
<th>Brooklyn Nine-Nine</th>
<th>Silicon Valley</th>
<th>Walking Dead</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4 =&gt; 0.5</td>
<td>5 =&gt; 1.5</td>
<td>2 =&gt; -1.5</td>
<td>=&gt; 0</td>
<td>3 =&gt; -0.5</td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td></td>
<td></td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>5</td>
<td></td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Given: \( user, x; \) \( item, i; \) \( utility\) matrix, \( u \)

0. Update \( u \): mean center, missing to 0

1. Find neighborhood, \( N \) # set of \( k \) users most similar to \( x \) who have also rated \( i \)

   \(-- sim(x, other) = cosine\_sim(u[x], u[other])\)

   \(-- threshold to top k (e.g. k = 30)\)
Collaborative Filtering Rec Systems

<table>
<thead>
<tr>
<th>user</th>
<th>Game of Thrones</th>
<th>Fargo</th>
<th>Brooklyn Nine-Nine</th>
<th>Silicon Valley</th>
<th>Walking Dead</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4 =&gt; 0.5</td>
<td>5 =&gt; 1.5</td>
<td>2 =&gt; -1.5</td>
<td>=&gt; 0</td>
<td>3 =&gt; -0.5</td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td></td>
<td></td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

Given: user, x; item, i; utility matrix, u

0. Update u: mean center, missing to 0
1. Find neighborhood, N # set of k users most similar to x who have also rated i
   -- sim(x, other) = cosine_sim(u[x], u[other])
   -- threshold to top k (e.g. k = 30)
2. Predict utility (rating) of i based on N
Collaborative Filtering Rec Systems

<table>
<thead>
<tr>
<th>user</th>
<th>Game of Thrones</th>
<th>Fargo</th>
<th>Brooklyn Nine-Nine</th>
<th>Silicon Valley</th>
<th>Walking Dead</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4 =&gt; 0.5</td>
<td>5 =&gt; 1.5</td>
<td>2 =&gt; -1.5</td>
<td>=&gt; 0</td>
<td>3 =&gt; -0.5</td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>5</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Given: user, x; item, i; utility matrix, u

0. Update u: mean center, missing to 0
1. Find neighborhood, N # set of k users most similar to x who have also rated i
   -- sim(x, other) = cosine_sim(u[x], u[other])
   -- threshold to top k (e.g. k = 30)
2. Predict utility (rating) of i based on N
   -- average, weighted by sim
   utility(x, i) = \frac{\sum_{y \in N} Sim(x, y) \cdot utility(y, i)}{\sum_{y \in N} Sim(x, y)}
Collaborative Filtering Rec Systems

“User-User collaborative filtering”

Given: user, x; item, i; utility matrix, u
0. Update u: mean center, missing to 0
1. Find neighborhood, N # set of k users most similar to x who have also rated i
   -- sim(x, other) = cosine_sim(u[x], u[other])
   -- threshold to top k (e.g. k = 30)
2. Predict utility (rating) of i based on N
   -- average, weighted by sim
   \[
   \text{utility}(x,i) = \frac{\sum_{y \in N} \text{Sim}(x,y) \cdot \text{utility}(y,i)}{\sum_{y \in N} \text{Sim}(x,y)}
   \]
Collaborative Filtering Rec Systems

“User-User collaborative filtering”

Item-Item:
Flip rows/columns of utility matrix and use same methods.
(i.e. estimate rating of item i, by finding similar items, j)

Given: user, x; item, i; utility matrix, u
0. Update u: mean center, missing to 0
1. Find neighborhood, N # set of k users most similar to x who have also rated i
   -- sim(x, other) = cosine_sim(u[x], u[other])
   -- threshold to top k (e.g. k = 30)
2. Predict utility (rating) of i based on N
   utility(x, i) = \frac{\sum_{y \in N} Sim(x, y) \cdot utility(y, i)}{\sum_{y \in N} Sim(x, y)}
Collaborative Filtering Rec Systems

“User-User collaborative filtering”

*Item-Item:*
- Flip rows/columns of utility matrix and use same methods. (i.e. estimate rating of item i, by finding similar items, j)

Given: user, x; item, i; utility matrix, u
0. Update u: mean center, missing to 0
1. Find neighborhood, N # set of k items most similar to i also rated by x
   -- sim(i, other) = cosine_sim(u[i], u[other])
   -- threshold to top k (e.g. k = 30)
2. Predict utility (rating) by x based on N
   -- average, weighted by sim
   \[
   \text{utility}(x, i) = \frac{\sum_{j \in N} \text{Sim}(i, j) \cdot \text{utility}(x, j)}{\sum_{j \in N} \text{Sim}(i, j)}
   \]
Item-Item v User-User

Item-item often works better than user-user. Why?

Users tend to be more different from each other than items are from other items.

e.g. Mary likes jazz + rock, Bob likes classical + rock, but Mary may still have same rock preferences as Bob
Item-Item v User-User

Item-item often works better than user-user. Why?

Users tend to be more different from each other than items are from other items.

e.g. Mary likes jazz + rock, Bob likes classical + rock,
    but Mary may still have same rock preferences as Bob

In other words, users span genres but items usually do not.
Item-Item: Example

<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>1</td>
<td>3</td>
<td>5</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>5</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></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>2</td>
<td></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>2</td>
<td>5</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>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- unknown rating
- rating between 1 to 5

Item-Item: Example

- estimate rating of movie 1 by user 5

### Item-Item: Example

<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>1</td>
<td>3</td>
<td></td>
<td></td>
<td>?</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>sim(1,m)</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-0.18</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>0.41</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-0.10</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-0.31</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.59</td>
</tr>
</tbody>
</table>

Same as cosine sim when subtracting the mean

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

Here we use Pearson correlation as similarity:
1) Subtract mean rating $m_i$ from each movie $i$
   $$m_i = \frac{(1+3+5+5+4)}{5} = 3.6$$
   **row 1:** [-2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4, 0, 0.4, 0]
2) Compute cosine similarities between rows
Item-Item: Example

Compute similarity weights:
$s_{1,3}=0.41$, $s_{1,6}=0.59$
### Item-Item: Example

**Utility Calculation**

\[
utility(1, 5) = \frac{0.41 \times 2 + 0.59 \times 3}{0.41 + 0.59}
\]

\[
utility(x, i) = \frac{\sum_{j \in N} Sim(i, j) \cdot utility(x, j)}{\sum_{j \in N} Sim(i, j)}
\]
Recommendation Systems

Problems to tackle:

1. Gathering ratings
2. Extrapolate unknown ratings
   a. Explicit: based on user ratings and reviews
      (problem: only a few users engage in such tasks)
   b. Implicit: Learn from actions (e.g. purchases, clicks)
      (problem: hard to learn low ratings)
3. Evaluation

Common Approaches

1. Content-based
2. Collaborative
3. Latent Factor
Options for Parallelizing

1. Approximate solutions to PCA (very large speedups with little drawback!):
   a. **Stochastic Sampling** (also sometimes called "randomized" which is ambiguous): Only using a sample rows (i.e. users for recommendation systems)
   b. **Truncated SVD**: Only optimizing for minimizing reconstruction error based on up to $r$ dimensions (full SVD solves for up to $\min(n, p)$ dimensions and then you just truncate the result for the lower rank version). One you do this, by the way, using a smaller sample becomes much less of a problem.
   c. **Limiting power iterations to a few iterations**: Power iterations from pagerank solves for the first principle component. This can be extended to multiple components.
      (more [here.](#))

2. Distribute the matrix operations. Complex; not as flexible (usually done across processors within node)
3. Data Parallelism: As in other instances stochastic or mini-batch gradient descent.