Algorithms in Recommendation Systems

Presented by:
Dumitrel Loghin, Anuja Meetoo Appavoo, Suhendry Effendy, Paramasiven Appavoo, Lu Bingxin, Li Jing, Suman Sourav
Familiar interfaces ...

**Man of Steel (2013)**

Action | Adventure | Fantasy

A young itinerant worker is forced to confront his secret extraterrestrial origin when Earth is invaded by members of his own race.

**People who liked this also liked...**

**Customers Who Bought This Item Also Bought**

- Introduction to Algorithms
  - Thomas H. Cormen
  - Hardcover
  - $70.13

  - Robert Sedgewick
  - Hardcover
  - $59.12

- Expert C Programming: Deep C Secrets
  - Peter van der Linden
  - Hardcover
  - $33.76

- Probability and Statistics for...
  - Ronald E. Walpole
  - Hardcover
  - $140.98
What are Recommendation Systems?

- systems (algorithms) trying to predict user preferences for new items

- all modern web apps have a recommender system
  - books and items (Amazon)
  - music (Spotify)
  - movies (IMDB)
  - friends (Facebook)
  - ...
Why using Recommendation Systems?
Approaches

● Collaborative filtering
  ○ user-based
  ○ item-based
  ○ major challenges

● Content-based

● Knowledge-based
Collaborative filtering

Ratings matrix

<table>
<thead>
<tr>
<th></th>
<th>I1</th>
<th>I2</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>?</td>
<td>✓</td>
<td>?</td>
</tr>
<tr>
<td>U2</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>...</td>
<td>?</td>
<td>✓</td>
<td>?</td>
</tr>
</tbody>
</table>

Recommendation System

Rating $r_{a,p}$

List of top recommended items
Collaborative filtering

● Assumptions
  ○ users rate items explicitly or implicitly
  ○ user’s taste preserves over time
  ○ use other users ratings (wisdom of the crowd)

● Approach
  ○ user-item matrix
  ○ predict the rating for a particular item or compute a list of recommended items
    ■ based on other users ratings
    ■ based on similar items ratings
User-based Collaborative Filtering
User-based collaborative filtering

Ratings matrix

<table>
<thead>
<tr>
<th></th>
<th>I1</th>
<th>I2</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>u1</td>
<td>?</td>
<td>✓</td>
<td>?</td>
</tr>
<tr>
<td>u2</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>...</td>
<td>?</td>
<td>✓</td>
<td>?</td>
</tr>
</tbody>
</table>

Recommendation System

Rating $r_{a,p}$

List of top recommended items

User $a$

Item $p$
Similarity and prediction equations

- Active user \(a\) and item \(p\), \(r_{a,p}\) is unknown
- First, compute similarities with other users

\[
sim(a, b) = \frac{\sum_{p \in P} (r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P} (r_{a,p} - \bar{r}_a)^2} \sqrt{\sum_{p \in P} (r_{b,p} - \bar{r}_b)^2}}
\]

- Then predict \(r_{a,p}\)

\[
pred(a, p) = \bar{r}_a + \frac{\sum_{b \in N} \sim(a, b) \cdot (r_{b,p} - \bar{r}_b)}{\sum_{b \in N} |\sim(a, b)|}
\]
## Example

<table>
<thead>
<tr>
<th></th>
<th>Iron Man</th>
<th>Fast and Furious</th>
<th>Avatar</th>
<th>Transformers</th>
<th>Man of Steel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>?</td>
</tr>
<tr>
<td>Jane</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Tom</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Bob</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Suzy</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

similarity:
- 0.85
- 0.70
- 0
- -0.79
User similarities

Graph showing user similarities for movies:
- Alice
- Jane
- Tom
- Bob
- Suzy

Movies:
- Iron Man
- Fast and Furious
- Avatar
- Transformers
- Man of Steel
Possible improvements

- Better similarity and prediction functions
  - give more weight to items having diverse ratings
  - give more weight to similar users - case amplification

- Better neighborhood selection
  - select only the most similar users
  - select users with whom the user has more common rated items
Using only similar users

diagram showing case amplification with lines for Alice, Jane, and Tom across movies Iron Man, Fast and Furious, Avatar, Transformers, and Man of Steel.
Algorithm

\textbf{UserBasedCF}(User a, Item p)

compute average rating for a

for each \( b \) in all other users

\hspace{1em} if \( b \) purchased \( p \)

\hspace{2em} compute average rating for \( b \)

\hspace{2em} compute \( sim(a,b) \)

\hspace{1em} [select the neighborhood]

compute rating \( r_{a,p} \)

Steps

\begin{align*}
\text{m} & \quad \text{m} \\
\text{n} & \quad \text{m} \\
\text{O(n)} & \quad \text{m} \\
\text{O(nm)} & \quad \text{} 
\end{align*}
Collaborative filtering challenges

- Scalability - huge rating matrix
  - $10^8$ users and growing
  - $10^7$ items and growing fast
- Sparsity - many undefined ratings
- Cold start - new users, new items
- Conspiracy - users agreement or shilling attacks using bots
- Privacy - user profiles
Item-based Collaborative Filtering
Shilling Attack

- User-based collaborative filtering is vulnerable to attack
  - Rely on user specified judgements (anyone)
  - Fake user profile to manipulate ratings
    - Push attack: Increase rating of one’s items
    - Nuke attack: Lower rating of competitors’ items

- Biased recommendation
  - Decrease user satisfaction

- Real case: Sony Pictures admitted it used fake quotes from non-existent movie critics to promote a number of newly released films (June 2001)
Scalability

- E-commerce recommendation systems often operate in a challenging environment
  - Millions of users and catalog items
  - High quality recommendations needed in real-time
- User-based collaborative filtering
  - Need to scan vast no. of neighbours
  - **Real-time prediction infeasible!**
  - Does not scale for most real-world scenarios :(

How does Amazon handle all its users and catalog items???
Item-based collaborative filtering

- Use similarity between items to predict user ratings
  - Item similarities are considered to be more stable than user similarities (Sarwar et al. 2001)

User rating / purchase history

Item similarity matrix (model learning phase)

Recommendation Component (prediction based on learned model)

List of top N recommended items for user

Underlying principle: We tend to buy products similar to what we like.
Example

<table>
<thead>
<tr>
<th></th>
<th>Iron Man</th>
<th>Fast and Furious</th>
<th>Avatar</th>
<th>Transformers</th>
<th>Man of Steel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>?</td>
</tr>
<tr>
<td>Jane</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Tom</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Bob</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Suzy</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

- Look for movies (items) similar to Man of Steel
- Use Alice’s ratings for these movies to predict her rating for Man of Steel
**Example**

<table>
<thead>
<tr>
<th></th>
<th><strong>Iron Man</strong></th>
<th>Fast and Furious</th>
<th>Avatar</th>
<th>Transformers</th>
<th>Man of Steel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>?</td>
</tr>
<tr>
<td>Jane</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Tom</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Bob</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Suzy</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

- Look for movies (items) similar to Man of Steel
- Use Alice’s ratings for these movies to predict her rating for Man of Steel
Example

<table>
<thead>
<tr>
<th></th>
<th>Iron Man</th>
<th>Fast and Furious</th>
<th>Avatar</th>
<th>Transformers</th>
<th>Man of Steel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>?</td>
</tr>
<tr>
<td>Jane</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Tom</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Bob</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Suzy</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

- Look for movies (items) similar to Man of Steel
- Use Alice’s ratings for these movies to predict her rating for Man of Steel
Example

- Look for movies (items) similar to Man of Steel
- Use Alice’s ratings for these movies to predict her rating for Man of Steel

<table>
<thead>
<tr>
<th></th>
<th>Iron Man</th>
<th><strong>Fast and Furious</strong></th>
<th>Avatar</th>
<th>Transformers</th>
<th>Man of Steel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>?</td>
</tr>
<tr>
<td>Jane</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Tom</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Bob</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Suzy</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
**Example**

<table>
<thead>
<tr>
<th></th>
<th>Iron Man</th>
<th><strong>Fast and Furious</strong></th>
<th>Avatar</th>
<th>Transformers</th>
<th>Man of Steel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>?</td>
</tr>
<tr>
<td>Jane</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Tom</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Bob</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Suzy</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

- **Look for movies (items) similar to Man of Steel**
- **Use Alice’s ratings for these movies to predict her rating for Man of Steel**
Example

<table>
<thead>
<tr>
<th></th>
<th>Iron Man</th>
<th>Fast and Furious</th>
<th>Avatar</th>
<th>Transformers</th>
<th>Man of Steel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>?</td>
</tr>
<tr>
<td>Jane</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Tom</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Bob</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Suzy</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

- Look for movies (items) similar to Man of Steel
- Use Alice’s ratings for these movies to predict her rating for Man of Steel
## Example

<table>
<thead>
<tr>
<th></th>
<th>Iron Man</th>
<th>Fast and Furious</th>
<th>Avatar</th>
<th>Transformers</th>
<th>Man of Steel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Jane</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Tom</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Bob</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Suzy</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

- Look for movies (items) similar to Man of Steel
- Use Alice’s ratings for these movies to predict her rating for Man of Steel
### Example

<table>
<thead>
<tr>
<th></th>
<th>Iron Man</th>
<th>Fast and Furious</th>
<th>Avatar</th>
<th>Transformers</th>
<th>Man of Steel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Jane</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Tom</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Bob</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Suzy</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

- **Look for movies (items) similar to Man of Steel**
- **Use Alice’s ratings for these movies to predict her rating for Man of Steel**
## Example

<table>
<thead>
<tr>
<th></th>
<th>Iron Man</th>
<th>Fast and Furious</th>
<th>Avatar</th>
<th>Transformers</th>
<th>Man of Steel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>?</td>
</tr>
<tr>
<td>Jane</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Tom</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Bob</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Suzy</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

- **Look for movies (items) similar to Man of Steel**
- **Use Alice’s ratings for these movies to predict her rating for Man of Steel**
**Example**

- Look for movies (items) similar to Man of Steel
- **Use Alice’s ratings for these movies to predict her rating for Man of Steel**

<table>
<thead>
<tr>
<th></th>
<th>Iron Man</th>
<th>Fast and Furious</th>
<th>Avatar</th>
<th>Transformers</th>
<th>Man of Steel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>?</td>
</tr>
<tr>
<td>Jane</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Tom</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Bob</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Suzy</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
You must be wondering...

How can similar items be identified?

How can user rating be predicted based on predictions of similar items?
Identifying similar items

- Cosine similarity measure
- Ratings are seen as vector in n-dimensional space
- Similarity calculated based on angle between 2 vector

- Similarity between 2 items a and b

\[
\text{sim}(a, b) = \cos(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}
\]

**Note:**
- ‘·’ is dot product
- Euclidean length, defined as \(\|\vec{x}\| = \sqrt{x_1^2 + \cdots + x_n^2}\)

**Similarity values are between 0 and 1, where values near to 1 indicate strong similarity**

Differences in average rating behavior of users not considered!!!
(Some users may generally give high ratings while others may give lower ratings as a preference)
Adjusted cosine measure

- Takes average user ratings into account
- Subtracts user average from ratings

\[ \text{sim}(a, b) = \frac{\sum_{u \in U} (r_{u,a} - \bar{r}_u)(r_{u,b} - \bar{r}_u)}{\sqrt{\sum_{u \in U} (r_{u,a} - \bar{r}_u)^2} \sqrt{\sum_{u \in U} (r_{u,b} - \bar{r}_u)^2}} \]

**Note**: \( U \) refers to set of users having rated items \( a \) and \( b \)

- Values range from \(-1\) to \(+1\), as in Pearson measure

Similarity for items with only one common user is 1

**Only items with one common user end up being most similar :(**

Solution: Need to have a minimum number of users in common for 2 items to be considered for similarity
### Adjusted cosine measure - Example

<table>
<thead>
<tr>
<th></th>
<th>Iron Man</th>
<th>Fast and Furious</th>
<th>Avatar</th>
<th>Transformers</th>
<th>Man of Steel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1.00</td>
<td>-1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>?</td>
</tr>
<tr>
<td>Jane</td>
<td>0.60</td>
<td>-1.40</td>
<td>-0.40</td>
<td>0.60</td>
<td>0.60</td>
</tr>
<tr>
<td>Tom</td>
<td>0.20</td>
<td>-0.80</td>
<td>0.20</td>
<td>-0.80</td>
<td>1.20</td>
</tr>
<tr>
<td>Bob</td>
<td>-0.20</td>
<td>-0.20</td>
<td>-2.20</td>
<td>2.80</td>
<td>0.80</td>
</tr>
<tr>
<td>Suzy</td>
<td>-1.80</td>
<td>2.20</td>
<td>2.20</td>
<td>-0.80</td>
<td>-1.80</td>
</tr>
</tbody>
</table>

Adjusted cosine similarity value for Man of Steel and Iron Man:

\[
\frac{0.6 \times 0.6 + 0.2 \times 1.2 + (-0.2) \times 0.8 + (-1.8) \times (-1.8)}{\sqrt{0.6^2 + 0.2^2 + (-0.2)^2 + (-1.8)^2} \times \sqrt{0.6^2 + 1.2^2 + 0.8^2 + (-1.8)^2}} = 0.80
\]
## Adjusted cosine measure - Example

### Offline computation

<table>
<thead>
<tr>
<th></th>
<th>Iron Man</th>
<th>Fast and Furious</th>
<th>Avatar</th>
<th>Transformers</th>
<th>Man of Steel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>?</td>
</tr>
<tr>
<td>Jane</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Tom</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Bob</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Suzy</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

| Similarity | 0.80 | -0.90 | -0.76 | 0.42 |
Predicting user rating

- Calculate weighted sum of Alice’s ratings for movies (items) similar to Man of Steel

\[
pred(u, p) = \frac{\sum_{i \in L(u)} \text{sim}(i, p) \cdot r_{u,i}}{\sum_{i \in L(u)} \text{sim}(i, p)}
\]

$L(u)$ = list of similar items to item $p$ rated by user $u$

- Number of similar items considered for prediction limited to a specific size
  - similar idea to user-based collaborative filtering
### Predicting user rating - Example

<table>
<thead>
<tr>
<th></th>
<th>Iron Man</th>
<th>Transformers</th>
<th>Avatar</th>
<th>Fast and Furious</th>
<th>Man of Steel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Jane</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Tom</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Bob</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Suzy</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

| Similarity | 0.80 | -0.90 | -0.76 | 0.42 |

- \( \text{pred}(\text{Alice}, \text{Man of Steel}) \)
  
  \[
  = \frac{(0.80 \times 5) + (0.42 \times 4)}{0.80 + 0.42} = 4.7
  \]
In practice... sparse matrix

<table>
<thead>
<tr>
<th></th>
<th>I_1</th>
<th>I_2</th>
<th>I_3</th>
<th>I_4</th>
<th>I_5</th>
<th>I_6</th>
<th>I_7</th>
<th>I_8</th>
<th>I_9</th>
<th>I_10</th>
<th>I_11</th>
<th>I_12</th>
<th>I_13</th>
<th>I_14</th>
<th>I_m</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Customers have very few purchases / rate very few items
## Item-based collaborative filtering algo

**Compute item similarity matrix**

For each item in product catalog, \( I_p \)
- For each customer \( C \) who purchased \( I_p \)
  - For each item \( I_q \) purchased by customer \( C \)
    - Record that a customer purchased \( I_p \) and \( I_q \)
  - For each item \( I_q \)
    - Compute the similarity between \( I_p \) and \( I_q \)

**Predict the user rating of product**

**Generate the list of top recommended items for user**

\((n: \text{no of customers, } m: \text{no of catalog items})\)

**Space requirements:** \( O(mn) \)
Item-based collaborative filtering

- Scales independently of no of users or items
  - Depend only on no of items rated by active user

- Less affected by shilling attack (Lam and Riedl 2004)
  - Predicted rating for item determined by comparing its item vector with those of other items
  - Attacker has no control over ratings given by other users to any item
### Correlation with Alice

<table>
<thead>
<tr>
<th></th>
<th>$l_1$</th>
<th>$l_2$</th>
<th>$l_3$</th>
<th>$l_4$</th>
<th>$l_5$</th>
<th>$l_6$</th>
<th>Correlation with Alice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice (active user)</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td></td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>-1.00</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>0.76</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0.72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>0.21</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td>-1.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>0.94</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>-1.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>1.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>0.89</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0.93</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.85</td>
<td>-0.55</td>
<td>0.00</td>
<td>0.48</td>
<td>-0.59</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Correlation with $l_6$**

- **most similar user without attack**
- **most similar user with attack**

**Popular item**
Collaborative Filtering
Major Challenges

Suhendry Effendy & Paramasiven
Scalability Problem

User-based CF : $O(n^2m)$ at worst, or $O(n^2)$ in practice.
Item-based CF : $O(nm^2)$ at worst, or $O(nm)$ in practice.

How to deal with millions of users and items?
Scalability Problem

User-based CF : $O(n^2m)$ at worst, or $O(n^2)$ in practice.
Item-based CF : $O(nm^2)$ at worst, or $O(nm)$ in practice.

How to deal with millions of users and items?

Several approaches:

- Clustering CF
- Bayesian CF
- Regression-Based CF
- MDP-Based CF, etc.
Clustering CF

Users or items are grouped by their similarity.
Clustering CF

How to make use the clustering?

- only consider users/items in the same cluster.
- smaller size = faster running time

<table>
<thead>
<tr>
<th></th>
<th>I₁</th>
<th>I₂</th>
<th>I₃</th>
<th>I₄</th>
<th>I₅</th>
<th>I₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>U₁</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U₂</td>
<td>×</td>
<td></td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>U₃</td>
<td>×</td>
<td>×</td>
<td></td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>U₄</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U₅</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U₆</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>I₁</th>
<th>I₂</th>
<th>I₃</th>
<th>I₄</th>
<th>I₅</th>
<th>I₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>U₁</td>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>U₂</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U₃</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U₄</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U₅</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U₆</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Clustering CF

Item Clustering
   ➔ based on item type (e.g., books, gadgets, etc.)
   ➔ based on item similarity.

User Clustering
   ➔ based on user’s similarity.

similarity
   ≠ similar rates or preferences
   = rate similar set of items
Clustering CF

Item Clustering
  ➔ based on item type (e.g., books, gadgets, etc.)
  ➔ based on item similarity.

User Clustering
  ➔ based on user’s similarity.

similarity
≠ similar rates or preferences
= rate similar set of items

Jaccard Similarity
\[
\frac{|A \cap B|}{|A \cup B|}
\]

Cosine Similarity
\[
\frac{A \cdot B}{\|A\| \|B\|}
\]
Clustering Algorithm for CF

Modularity Maximization (Newman)
  ➔ NP-Hard Problem (Brandes et al.)
  ➔ usually used in community detection problem.
  ➔ constant approximation algorithm (Dinh and Thai)
  ➔ proposed by Pham et al. for clustering CF

RecTree (Chee et al.)
  ➔ recursive CF clustering
  ➔ K-Means
Modularity maximization

Modularity is the fraction of the edges that fall within the given groups minus the expected such fraction if edges were distributed at random.

\[
Q = \sum_{i=1}^{q} (e_{ii} - r_i)
\]

- \(e_{ii}\) = % of edges in cluster i
- \(r_i\) = probability of random edge belong to cluster i
Modularity Maximization

Modularity is the fraction of the edges that fall within the given groups minus the expected such fraction if edges were distributed at random.

$$Q = \sum_{i=1}^{q} (e_{ii} - a_i^2)$$

- $e_{ii} = \%$ of edges in cluster $i$
- $a_i = \%$ of degree of nodes in cluster $i$
Modularity Maximization

\[ Q = \sum_{i=1}^{q} (e_{ii} - a_i^2) \]

\( e_{ii} = \) % of edges in cluster i

\( a_i = \) % of degree of nodes in cluster i
Modularity Maximization

\[ Q = \sum_{i=1}^{q} (e_{ii} - a_i^2) \]

\[ e_{ii} = \text{% of edges in cluster } i \]

\[ a_i = \text{% of degree of nodes in cluster } i \]

\[ \frac{4}{16} - \left( \frac{6}{16} \right)^2 = -0.109 \]

\[ \frac{8}{16} - \left( \frac{10}{16} \right)^2 = -0.109 \]
Modularity is the fraction of the edges that fall within the given groups minus the expected such fraction if edges were distributed at random.

\[
Q = \sum_{i=1}^{q} (e_{ii} - a_i^2)
\]

High modularity

= more edges within the cluster than you expect by chance
Modularity Maximization

\[ Q = \sum_{i=1}^{q} \left( e_{ii} - a_i^2 \right) \]

\[ e_{ii} = \% \text{ of edges in cluster } i \]

\[ a_i = \% \text{ of degree of nodes in cluster } i \]

\[ Q = 0.367 \]

better modularity!
Modularity Maximization

But our graph is weighted (jaccard or cosine)!

$$Q = \sum_{i=1}^{q} (e_{ii} - a_{i}^2)$$

generalize this to weighted network!

e_{ii} = \% \text{ of weight of edges in cluster } i

a_{i} = \% \text{ of weight of edges of nodes in cluster } i

High modularity

= more weight within the cluster than you expect by chance
Modularity Maximization

Simple Greedy Algorithm

Start:
   each node is in its own cluster.

Iterate:
   for each node, move it to other cluster which improve its modularity the most.

Stop when the desired total modularity is achieved or cannot be improved.

Time complexity: $O(nq)$ per iteration - modularity gain can be computed in $O(1)$
In practice, it’s converge very quickly.
Modularity Maximization

Variation #2

Start:
   each node is in its own cluster.

Iterate:
   for each node, move it to other cluster with the highest modularity gain (could be negative).

Return the clustering with highest observed modularity.
Modularity Maximization

Variation #3 (Blondel et al.)

Start:

each node is in its own cluster.

Iterate:

1-pass:

for each node, move it to other cluster which improve its modularity the most.

create graph $G'$ with each cluster (found in 1-pass) as one node, use $G'$ for the next iteration.

This will return a hierarchical clustering -- select the best clustering (highest modularity).
Modularity Maximization
Modularity Maximization
Modularity Maximization
Modularity Maximization
Modularity Maximization
Modularity Maximization
Modularity Maximization

... and so on
RecTree

Recursively partition the data into 2 clusters.

K-Means with K = 2.

Stop when:

➔ the partition size is small enough.
➔ the recursion is too deep.

O(n \lg n/b), if:

➔ partition size = b
➔ recursion depth = \lg n

RecTree is an acronym for Recommendation Tree
Back to CF

Time complexity to build user-based clustering CF

- assume each cluster size = b
- compute similarities for one cluster = $O(b^2)$
- number of cluster = $n / b$
- total complexity = $O(n.b)$
Clustering CF

Advantage of Clustering CF

- Faster computation.
  - Small cluster size vs. entire data.

Drawback

- Researchers report that the prediction quality is lower (especially on user-based clustering CF).
Data sparsity

- Algorithms for sparse data
  - Graph-based method
  - Matrix factorization method
  - Also resolves:
    - First rater - new items
    - Population bias - unique taste
    - Scalability
      - Ratings can be precomputed offline
      - Parallelization is permissible
      - Rating is estimated for any unrated item in $O(1)$ for a given user
Algorithms for sparse datasets (1)
Graph-based method (Huang et al. 2004)

- Exploit the supposed “transitivity” in user tastes
  - Example: Which item $i_x$ could be recommended to a user $u_1$?

\[ u_1 = \{i_2, i_4\} \]
\[ u_2 = \{i_2, i_3, i_4\} \]
\[ u_3 = \{i_1, i_3\} \]

\[
\begin{array}{cccc}
  & i_1 & i_2 & i_3 & i_4 \\
 u_1 & 0 & 1 & 0 & 1 \\
 u_2 & 0 & 1 & 1 & 1 \\
 u_3 & 1 & 0 & 1 & 0 \\
\end{array}
\]

- $i_3$ is recommended to $u_1$ because:
  - ∃ a three-step path between $u_1$ and $i_3$
  - $u_1 \rightarrow i_2 \rightarrow u_2 \rightarrow i_3$
Algorithms for sparse datasets (1)
Graph-based method (Huang et al. 2004)

- Exploit the supposed “transitivity” in user tastes
  - Example: Which item $i_x$ could be recommended to a user $u_1$?

$u_1 = \{i_2, i_4\}$
$u_2 = \{i_2, i_3, i_4\}$
$u_3 = \{i_1, i_3\}$

<table>
<thead>
<tr>
<th></th>
<th>$i_1$</th>
<th>$i_2$</th>
<th>$i_3$</th>
<th>$i_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_1$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$u_2$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$u_3$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- $i_3$ is recommended to $u_1$ because:
  - $\exists$ a three-step path between $u_1$ and $i_3$
  - $u_1 \rightarrow i_2 \rightarrow u_2 \rightarrow i_3$
Algorithms for sparse datasets (1)

Graph-based method (Huang et al. 2004)

- Exploit the supposed "transitivity" in user tastes
  - Example: Which item $i_x$ could be recommended to a user $u_1$?

\[
\begin{align*}
\mathbf{u}_1 &= \{i_2, i_4\} \\
\mathbf{u}_2 &= \{i_2, i_3, i_4\} \\
\mathbf{u}_3 &= \{i_1, i_3\}
\end{align*}
\]

<table>
<thead>
<tr>
<th></th>
<th>$i_1$</th>
<th>$i_2$</th>
<th>$i_3$</th>
<th>$i_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbf{u}_1$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$\mathbf{u}_2$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$\mathbf{u}_3$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- $i_3$ is recommended to $u_1$ because:
  - \( \exists \) a three-step path between $u_1$ and $i_3$
  - $u_1 \rightarrow i_2 \rightarrow u_2 \rightarrow i_3$
Algorithms for sparse datasets (1)
Graph-based method (Huang et al. 2004)

- Exploit the supposed “transitivity” in user tastes
  - Example: Which item $i_x$ could be recommended to a user $u_1$?

$u_1 = \{i_2, i_4\}$
$u_2 = \{i_2, i_3, i_4\}$
$u_3 = \{i_1, i_3\}$

<table>
<thead>
<tr>
<th></th>
<th>$i_1$</th>
<th>$i_2$</th>
<th>$i_3$</th>
<th>$i_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_1$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$u_2$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$u_3$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- $i_3$ is recommended to $u_1$ because:
  - $\exists$ a three-step path between $u_1$ and $i_3$
    - $u_1 \rightarrow i_2 \rightarrow u_2 \rightarrow i_3$
  - Another 3-step path: $u_1 \rightarrow i_4 \rightarrow u_2 \rightarrow i_3$
Algorithms for sparse datasets (1)

Graph-based method (Huang et al. 2004)

- Consider longer paths (indirect associations) to compute recommendations in sparse matrices
  - Using path length 5, for instance
- Using path length of 3:
  - Recommend $i_3$ to $u_1$
- Using path length of 5:
  - 2 paths exist between $i_1$ and $u_1$
  - $i_1$ is also recommendable to $u_1$
Algorithms for sparse datasets (1)

Graph-based method (P. Symeonidis et al. 2011)

- Improve the relevance of recommendations
- Combining graphs
  - Unipartite graph
    - user-user
    - friendship network/ explicit social network
  - Bipartite graph
    - user-item (shown earlier)
  - Multi-modal graphs
    - friendship among users
    - user ratings on items
- Can be used by sites like Flixter
  - A community where users share film reviews and ratings
The intuition

- Given a list of movies that your friend have not viewed
- How do you recommend?
  - Watch it because I watched it and liked it, **OR**
  - Match attributes (comedy, horror, ...) of movies with those attributes of other movies appreciated by friend

The Netflix 2009 $1,000,000 prize winner for the recommender’s system based their solution on matrix factorization! - [http://www.netflixprize.com/](http://www.netflixprize.com/)

Simon Funk - (Real-name: **Brandyn Webb**) independent software developer who works on Netflix prize in his spare time. He freely publishes his code...
Algorithms for sparse datasets (2)
Matrix Factorization - Simon Funk method

- Factorize rating matrix
  - Define set $K = \{a_1, a_2, ..., a_k\}$, attributes of an item $v(i,j) \in [0,1] \Rightarrow \sum_{j=1}^{k} v(i,j) = 1$ for $i = C$, a constant
  - Recommended rating of item $i$ for user $j$ is:
    - $r(i,j) = U_{i (row)} \cdot V_{j (col)}^T$, for known $r(i,j)$

$R = U_{N \times |K|} \cdot V_{M \times |K|}^T$
### Matrix Factorization - Simon Funk method

- Estimated rating of item $i_{m+1}$ for $u_x$, 
  \[ r_{x, m+1} \approx U_x \text{ (row)} \cdot V^T_{m+1} \text{ (col)} \]

#### matrices

<table>
<thead>
<tr>
<th>$R$</th>
<th>$U$</th>
<th>$V^T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i_1$</td>
<td>$a_1$</td>
<td>$i_1$</td>
</tr>
<tr>
<td>$i_2$</td>
<td>$a_2$</td>
<td>$i_2$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$i_m$</td>
<td>...</td>
<td>$i_m$</td>
</tr>
<tr>
<td>$i_{m+1}$</td>
<td>$a_k$</td>
<td>$i_{m+1}$</td>
</tr>
</tbody>
</table>

| $u_1$ | $u_1$ |
| $u_2$ | $u_2$ |
| ... | ... |
| $u_n$ | $u_n$ |
As such the missing ratings in $R_{N*Z}$ can be estimated from $R_{N*M}$, where $M < Z$:

$$R_{N*Z} \approx U_{N*K} \cdot V_{Z*K}^T$$
Analysis of matrix factorization
Dimensionality reduction

- Vectors of the rating matrix, \( R \), are of extremely high dimension
  - an item vector is an \( n \)-dimensional vector with missing user values
  - a user vector is an \( m \)-dimensional vector with missing item values
  - users and items can possibly be grouped (e.g. similar profile)
  - So can we represent users and items in smaller dimensions
  - Ideally by a constant, \( k \)
  - users and items, each represented in \( k \) dimensions
Analysis of matrix factorization

Complexity

- Given a Matrix (N*M),
  - # of users = n, # of items = m
- Derive k aspect’s values for m items
  - mk operations (or input: producer-defined)
- Derive k aspect’s preferences for n users
  - k systems of linear eq to solve for each user
  - nCk operations, C is a constant
- Compute approximate ratings for $R^{NM}$
  - 2k for each rating (matrix row * col operation)
  - m * n * 2k, at most
- Time complexity $O(mn)$
  - Dimensionality reduction
    - Complexity reduction from $O(m^2n)$ to $O(mn)$
      - One m is “reduced” to the constant 2k :)
Google News
Collaborative Filtering in use!

- Aggregates news article from several thousand sources
- Displays them to signed-in users in a personalized way
- Collaborative filtering approach based on
  - the click history of the active user
  - the history of the larger community
- Main challenges
  - Vast amount of articles and users
  - Generate recommendation list in real-time
  - Constant stream of new items
  - Immediate reaction to the user interaction
Google News
from yr 2007 to yr 2010

• Methods
  ○ Two clustering techniques are used
  ○ Analyze history co-visits for dealing with new users

• Scalability of CF
  ○ Google's MapReduce technique is used for parallelization in order to make computation scalable
  [Abhinandan D. et al. 2007]

• Hybrid method
  ○ Combination of collaborative filtering mechanism with content-based (the next topic…)
  ○ Improved the quality of news recommendation and increased traffic to the site
  [Liu et al. 2010]
Content-Based Recommendation
Why Content-based Recommendation?

- Collaborative filtering does not require any information or content about the items themselves, only using the ratings of items given by users.
- It might be reasonable to exploit such information.
What is Content-based Recommendation

Content-based Recommender

- user preferences (such as ratings for items)
- relevant item(s) matching the user’s preference
- Item descriptions

recommend items similar to what the user has liked in the past, instead of what similar users like

not using user community information

a different form of cold-start: require an initial description of preferences from user
Real-world example

Content-based method is often combined with collaborative filtering method, contributing to personalize the system based on a user’s interest.
Real-world example
-Pandora Radio

**Step 1:** Enter artist or song title

User preferences

www.beavc.org/08presentations/pandora.ppt
Real-world example
-Pandora Radio

**Item descriptions**

**STEP 2:** The entry is analyzed on 400 distinct musical characteristics

www.beavc.org/08presentations/pandora.ppt
Real-world example
- Pandora Radio

STEP 3: Similar songs are played on newly formed station
High level architecture of a content-based recommender

- **Represented Items**: Items that are rated by the user.
- **Profile Learner**: A ranked list of potentially interesting items or a binary relevance judgement for an item.
- **Content Analyzer**: Structured representation of user interests.
- **Profiles**: New items.
- **Filtering Component**: Active user $u_a$.
- **Feedback**: Positive +: items are relevant or liked by the user. Negative -: items are nonrelevant or disliked by the user.
- **Information Source**: List of recommendations.
Content-based recommendation as classification problem

Each item is to be classified as whether interesting to user or relevant with user preferences or not. Two classes: positive (+) like/relevant; negative (-) dislike/nonrelevant

<table>
<thead>
<tr>
<th>id</th>
<th>word1</th>
<th>word2</th>
<th>word3</th>
<th>...</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc1</td>
<td>0.2</td>
<td>0.4</td>
<td>3</td>
<td>.....</td>
<td>+</td>
</tr>
<tr>
<td>doc2</td>
<td>5.3</td>
<td>2.5</td>
<td>2.7</td>
<td>.....</td>
<td>+</td>
</tr>
<tr>
<td>doc3</td>
<td>3.2</td>
<td>4.6</td>
<td>0</td>
<td>.....</td>
<td>+</td>
</tr>
<tr>
<td>doc4</td>
<td>2.9</td>
<td>3.5</td>
<td></td>
<td>.....</td>
<td>-</td>
</tr>
</tbody>
</table>

Training set

<table>
<thead>
<tr>
<th>id</th>
<th>word1</th>
<th>word2</th>
<th>word3</th>
<th>...</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc11</td>
<td>0.2</td>
<td>0.4</td>
<td>3</td>
<td>.....</td>
<td>?</td>
</tr>
<tr>
<td>doc12</td>
<td>5.3</td>
<td>2.5</td>
<td>2.7</td>
<td>.....</td>
<td>?</td>
</tr>
<tr>
<td>doc13</td>
<td>3.2</td>
<td>4.6</td>
<td>0</td>
<td>.....</td>
<td>?</td>
</tr>
<tr>
<td>doc14</td>
<td>1.7</td>
<td>2.9</td>
<td>3.5</td>
<td>.....</td>
<td>?</td>
</tr>
</tbody>
</table>

Test set

items rated by the user

item representation

not-yet-seen items

machine learning

Learning algorithm

Learn model

Apply model

Model

User Profile

Learning algorithm

Learn model

Apply model

Model

User Profile
Item descriptions

- Some items are **structured** and can easily be represented by a set of attributes
  - movie
    - actor, director, genre, subject
  - book
    - title, genre, author, type, price, keyword
- Some items are **unstructured text documents** which have no attributes with well-defined values
  - the information source of most content-based methods
    - web pages
    - news articles
    - emails
Item Representation

- Item content
  - a set of descriptors or terms
    - typically the words that occur in a document for unstructured text

- User profile
  - often represented with the same terms as the item
    so that both the user profile and the items can be compared in a meaningful way
## Item Representation
-for structured data

<table>
<thead>
<tr>
<th>Title</th>
<th>Genre</th>
<th>Author</th>
<th>Type</th>
<th>Price</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>The Lace Reader</em></td>
<td>Fiction, Mystery</td>
<td>Brunonia</td>
<td>Hardcover</td>
<td>49.90</td>
<td>American contemporary fiction, detective, historical</td>
</tr>
<tr>
<td><em>Into the Fire</em></td>
<td>Romance, Suspense</td>
<td>Suzanne</td>
<td>Hardcover</td>
<td>45.90</td>
<td>American fiction, murder, neo-Nazism</td>
</tr>
</tbody>
</table>

...
Item Representation
-for structured data

<table>
<thead>
<tr>
<th>Title</th>
<th>Genre</th>
<th>Author</th>
<th>Type</th>
<th>Price</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>The Lace Reader</em></td>
<td>Fiction, Mystery</td>
<td>Brunonia Barry</td>
<td>Hardcover</td>
<td>49.90</td>
<td>American contemporary fiction, detective, historical</td>
</tr>
<tr>
<td><em>Into the Fire</em></td>
<td>Romance, Suspense</td>
<td>Suzanne Brockmann</td>
<td>Hardcover</td>
<td>45.90</td>
<td>American fiction, murder, neo-Nazism</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

same list of terms (features)

Item of books

<table>
<thead>
<tr>
<th>Title</th>
<th>Genre</th>
<th>Author</th>
<th>Type</th>
<th>Price</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Fiction, Suspense</td>
<td>Brunonia Barry, Ken Follett</td>
<td>Paperback</td>
<td>25.65</td>
<td>detective, murder, New York</td>
</tr>
</tbody>
</table>

Alice’s User profile
Item Representation

-for structured data

<table>
<thead>
<tr>
<th>Title</th>
<th>Genre</th>
<th>Author</th>
<th>Type</th>
<th>Price</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>The Lace Reader</em></td>
<td>Fiction, Mystery</td>
<td>Brunonia Barry</td>
<td>Hardcover</td>
<td>49.90</td>
<td>American contemporary fiction, detective, historical</td>
</tr>
<tr>
<td><em>Into the Fire</em></td>
<td>Romance, Suspense</td>
<td>Suzanne Brockmann</td>
<td>Hardcover</td>
<td>45.90</td>
<td>American fiction, murder, neo-Nazism</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Dice coefficient**

\[ sim(i, u) = \frac{2|\text{keyword}(i) \cap \text{keyword}(u)|}{|\text{keyword}(i)| + |\text{keyword}(u)|} \]

Item of books (not yet seen by Alice)

Alice’s User profile

Measure similarity between items and user profile to make recommendations
Item Representation

-for unstructured text

- A standard approach to represent unstructured document content -- Vector space model
  - selects keywords (terms) from documents
  - represent document as vector in a multi dimensional space (terms as dimensions): \( d_j = \{w_{1j}, w_{2j}, ..., w_{nj}\} \)

  - *user profile* can be represented just like documents by one or more profile vectors

  - Boolean term vector
  - Weighted term vector
Item Representation
- Vector Space Model

• Boolean term vector

<table>
<thead>
<tr>
<th></th>
<th>team</th>
<th>coach</th>
<th>play</th>
<th>ball</th>
<th>score</th>
<th>game</th>
<th>win</th>
<th>lost</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>document1</strong></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><strong>document2</strong></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td><strong>document3</strong></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• feature selection: choose only a subset of the terms in the documents
Item Representation
- Vector Space Model

- **Boolean term vector**

|           | team | coach | play | ball | score | game | win | lost | ...
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>document1</strong></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>document2</strong></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>document3</strong></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>...</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- every word has the same relevance to a document, but it seems intuitive that
  - a word appearing more often is better suited for characterizing the document
  - a term may appear more often in longer documents
Item Representation - Vector Space Model

- Weighted term vector
  - standard measure to weight the words: Term Frequency - Inverse Document Frequency (TF-IDF)
  - a term is assigned a weight based on
    - how often a term appears in a particular document
    - how frequently it occurs in the entire document collection

TF
Assumes that relevant terms appear more often and longer documents are not preferred to short documents

IDF
Assumes that rare terms are more relevant than frequent terms
Aims to reduce the weight of terms that appear in all documents
Weighted Term Vector

TF-IDF

- Given a term $i$ and a document $j$
  - $TF(i, j)$: term frequency of keyword $i$ in document $j$
    \[ TF(i, j) = \frac{freq(i, j)}{\text{maxOthers}(k, j)} \]
    - the number of occurrences of keyword $i$ in document $j$
    - The highest number of occurrences of any other keyword $k$ in document $j$
  - $IDF(i)$: inverse document frequency for keyword $i$
    \[ IDF(i) = \log \frac{N}{n(i)} \]
    - the number of all documents
    - the number of documents where keyword $i$ appears

\[ TF-IDF(i, j) = TF(i, j) \times IDF(i) \]

TF-IDF weight can be normalized to fall in $[0,1]$ interval
Instead of a vector of Boolean values, the vector for each document is represented as the computed TF-IDF weights. The higher the value, a term may appear more often in a particular document or less often in all documents, and thus more relevant to the topic of the document.
Similarity metrics based on vector space model

- Common similarity metrics to compare two vectors $d_i = (w_{1i}, w_{2i}, \ldots, w_{ki})$, $d_j = (w_{1j}, w_{2j}, \ldots, w_{kj})$:

  \[ \text{Cosine similarity} \]
  \[ \text{Dice Coefficient} \]
  \[ \text{Jaccard coefficient} \]

\[
\begin{align*}
\text{sim}(d_i, d_j) &= \frac{d_i \cdot d_j}{|d_i| \cdot |d_j|} = \frac{\sum_k w_{ki} \cdot w_{kj}}{\sqrt{\sum_k w_{ki}^2} \cdot \sqrt{\sum_k w_{kj}^2}} \\
\text{sim}(d_i, d_j) &= 2 \frac{d_i \cdot d_j}{|d_i|^2 + |d_j|^2} = 2 \frac{\sum_k w_{ki} \cdot w_{kj}}{\sum_k w_{ki}^2 + \sum_k w_{kj}^2} \\
\text{sim}(d_i, d_j) &= \frac{d_i \cdot d_j}{|d_i|^2 + |d_j|^2 - d_i \cdot d_j} = \frac{\sum_k w_{ki} \cdot w_{kj}}{\sum_k w_{ki}^2 + \sum_k w_{kj}^2 - \sum_k w_{ki} \cdot w_{kj}}
\end{align*}
\]
Item Representation
-More on vector space model

- **Semantic meaning remains unknown**
  - Polysemy
    - *mouse*
    - The vector space model is *unable to discriminate between different meanings* of the same word
  - Synonymy
    - *car and vehicle*
    - *No associations between different words* are made in the vector space model

Latent semantic indexing [http://recommender-systems.org/latent-semantic-indexing/]
Simple Method: Nearest Neighbors

- Given a set of documents D already rated by the user (like/dislike)
  
  For each not-yet-seen item i
    
    - compute similarity between i and items in D
    - Find the N nearest neighbors of i in D
    - Major voting to predict ratings of i

- $m$: number of items
- $m_1$: number of items in D
- $d$: dimension of vector space
- ratings = \{like, dislike\}

**Time complexity: $O(dm^2)$**

In practice, most users can only rate a much small number of items compared to $m$, $m_1$ approximates to an upper bound, time complexity can approach $O(dm)$
## Probabilistic Methods

Simple approach:
- 2 classes: 1/0
- simple Boolean document representation
- calculate probability that document is labeled 1/0 based on Bayes theorem

\[
P(\text{Label}=1|X) = k \cdot P(X|\text{Label}=1) \cdot P(\text{Label}=1)
\]

### Example Table

<table>
<thead>
<tr>
<th>Doc-ID</th>
<th>recommender</th>
<th>intelligent</th>
<th>learning</th>
<th>school</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>?</td>
</tr>
</tbody>
</table>

\[
P(X|\text{Label} = 1) = P(\text{recommender} = 1|\text{Label} = 1) \\
\times P(\text{intelligent} = 1|\text{Label} = 1) \\
\times P(\text{learning} = 0|\text{Label} = 1) \\
\times P(\text{school} = 0|\text{Label} = 1)
\]

\[
= \frac{3}{3} \times \frac{2}{3} \times \frac{1}{3} \times \frac{2}{3} \approx 0.149
\]
### Probabilistic Methods

For each unlabeled item

for each component

compute the prior probability

Overall time complexity: $O(dm^2)$

In practice: $O(dm)$

---

<table>
<thead>
<tr>
<th>Doc-ID</th>
<th>recommender</th>
<th>intelligent</th>
<th>learning</th>
<th>school</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>?</td>
</tr>
</tbody>
</table>

$P(X|\text{Label} = 1) = P(\text{recommender} = 1|\text{Label} = 1) \times P(\text{intelligent} = 1|\text{Label} = 1) \times P(\text{learning} = 0|\text{Label} = 1) \times P(\text{school} = 0|\text{Label} = 1) 
= \frac{3}{3} \times \frac{2}{3} \times \frac{1}{3} \times \frac{2}{3} \approx 0.149$
Other classification algorithms

- Decision tree
- Rule induction
- Support vector machines
- Neutral network
- etc.
Relevance Feedback

- Take advantage of user relevance judgments in the retrieval process:
  - User issues a (short, simple) query and gets back an initial hit list
  - User marks hits as relevant or non-relevant
  - The system computes a better representation of the information need based on this feedback
  - Single or multiple iterations

- Idea: you may not know what you’re looking for, but you’ll know when you see it
Picture of Relevance Feedback

Initial query

Revised query

x non-relevant documents
o relevant documents
Rocchio Algorithm

- Query and documents are represented by TF-IDF criteria.
- Updation in practice:

$$\tilde{q}_m = \alpha q_0 + \beta \frac{1}{|D_r|} \sum_{\tilde{d}_j \in D_r} \tilde{d}_j - \gamma \frac{1}{|D_{nr}|} \sum_{\tilde{d}_j \in D_{nr}} \tilde{d}_j$$

$q_m =$ modified query vector;
$q_0 =$ original query vector;
$\alpha, \beta, \gamma$: weights (hand-chosen or set empirically);
$D_r =$ set of known relevant doc vectors;
$D_{nr} =$ set of known irrelevant doc vectors

New query
Moves toward relevant documents, but away from irrelevant documents
Rocchio Algorithm: Number Example

query vector = \( \alpha \cdot \) original query vector

\[ + \beta \cdot \) positive feedback vector \]

\[ - \gamma \cdot \) negative feedback vector \]

<table>
<thead>
<tr>
<th>query</th>
<th>0</th>
<th>4</th>
<th>0</th>
<th>8</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>positive feedback</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>negative feedback</td>
<td>8</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>16</td>
</tr>
</tbody>
</table>
Rocchio Algorithm: Number Example

query vector = \( \alpha \cdot \text{original query vector} \) 
+ \( \beta \cdot \text{positive feedback vector} \) 
- \( \gamma \cdot \text{negative feedback vector} \)

query

\[
\begin{array}{cccccc}
0 & 4 & 0 & 8 & 0 & 0
\end{array}
\]
\( \alpha = 1 \)

positive feedback

\[
\begin{array}{cccccc}
2 & 4 & 8 & 0 & 0 & 2
\end{array}
\]
\( \beta = 0.5 \)

negative feedback

\[
\begin{array}{cccccc}
8 & 0 & 4 & 4 & 0 & 16
\end{array}
\]
\( \gamma = 0.25 \)

Typically \( \beta > \gamma \), since positive feedback is more meaningful.
Rocchio Algorithm: Number Example

query vector = $\alpha \cdot$ original query vector
  + $\beta \cdot$ positive feedback vector
  - $\gamma \cdot$ negative feedback vector

query vector: 
\[
\begin{bmatrix}
0 & 4 & 0 & 8 & 0 & 0
\end{bmatrix}
\]
\[
\alpha = 1
\]
\[
\begin{bmatrix}
0 & 4 & 0 & 8 & 0 & 0 & 0
\end{bmatrix}
\]

positive feedback vector: 
\[
\begin{bmatrix}
2 & 4 & 8 & 0 & 0 & 2
\end{bmatrix}
\]
\[
\beta = 0.5
\]
\[
\begin{bmatrix}
1 & 2 & 4 & 0 & 0 & 1
\end{bmatrix}
\]

negative feedback vector: 
\[
\begin{bmatrix}
8 & 0 & 4 & 4 & 0 & 16
\end{bmatrix}
\]
\[
\gamma = 0.25
\]
\[
\begin{bmatrix}
2 & 0 & 1 & 1 & 0 & 4
\end{bmatrix}
\]

Typically $\beta > \gamma$, since positive feedback is more meaningful.
Rocchio Algorithm: Number Example

query vector = $\alpha \cdot$ original query vector
+ $\beta \cdot$ positive feedback vector
- $\gamma \cdot$ negative feedback vector

query
\[
\begin{bmatrix}
0 & 4 & 0 & 8 & 0 & 0
\end{bmatrix}
\]
\[
\alpha = 1
\]

positive feedback
\[
\begin{bmatrix}
2 & 4 & 8 & 0 & 0 & 2
\end{bmatrix}
\]
\[
\beta = 0.5
\]

+ \[
\begin{bmatrix}
1 & 2 & 4 & 0 & 0 & 1
\end{bmatrix}
\]

negative feedback
\[
\begin{bmatrix}
8 & 0 & 4 & 4 & 0 & 16
\end{bmatrix}
\]
\[
\gamma = 0.25
\]

- \[
\begin{bmatrix}
2 & 0 & 1 & 1 & 0 & 4
\end{bmatrix}
\]

new query
\[
\begin{bmatrix}
-1 & 6 & 3 & 7 & 0 & -3
\end{bmatrix}
\]
\[
\text{typically } \beta > \gamma, \text{ since positive feedback is more meaningful.}
\]
\[
\text{negative term weights become 0.}
\]
\[
\begin{bmatrix}
0 & 6 & 3 & 7 & 0 & 0
\end{bmatrix}
\]
Rocchio Algorithm

- Initial query can start with boolean vector
- Negative weights are usually ignored
- Rocchio based relevance feedback improves both recall and precision
- For reaching high recall, many iterations are needed
- Empirically determined values for the balancing weights:
  \[ \alpha = 1 \quad \beta = 0.75 \quad \gamma = 0.15 \]
- Positive feedback is usually more valuable than negative feedback:
  \[ \beta > \gamma \]
Shortcomings of Relevance Feedback

- Relevance Feedback does not work when:
  - The users do not have sufficient initial knowledge
    - (misspelled query, ambiguous vocabulary, …)
  - There exist several prototypes of relevant documents
    - query has disjunctive answer sets ("the pop star that worked at KFC")
    - query concerns an instance of a general concept (felines, cat)
    - documents are gathered into subsets each using a different vocabulary

- Practical problem: refining leads to longer queries that need more time to process
Relevance Feedback and the Web

Few web IR systems use relevance feedback
- hard to explain to users
- users are mainly interested in fast retrieval (i.e. no iterations)
- users usually are not interested in high recall

Nowadays: clickstream-based feedback (which links are clicked on by users)
→ implicit feedback from the writer rather than feedback from the reader
Why do we need knowledge based recommendation?

- Products with low number of available ratings

- Time span plays an important role
  - Five-year-old ratings for computers
  - User lifestyle or family situation changes

- Customers want to define their requirements explicitly
  - “The color of the car should be black
Knowledge based recommendation

Knowledge-based: "Tell me what fits based on my needs"

User profile

Product features

Recommendation component

Recommendation list

Knowledge models

<table>
<thead>
<tr>
<th>item</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>i1</td>
<td>0.9</td>
</tr>
<tr>
<td>i2</td>
<td>1</td>
</tr>
<tr>
<td>i3</td>
<td>0.3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Wizard: My Product Advisor

Now you can:
- Answer more questions that are important to you.
- See recommended cameras based on your preferences so far.
- Review what you have done or start over.

The system decides what the wizard says

Possible user's requests
For a personal discovery of India

Choose a REGION:

and choose an INTEREST:

or enter name of a DESTINATION:
Someplace Similar.

Now pick a personality type that best describes YOU -- this will help us find similar spots based on things you like.

- **CULTURE CREATURE**
  Loves everything cultural - theater, shows, museums... local & historical culture too!

- **BEACH BUM**
  Somebody has to lay around on the beach with little umbrellas pitched in their drinks.

- **TRAIL TREKKER**
  If it’s outdoors - you’re there. Hiking, walking... parks, forests, mountains.

- **SIGHT SEEKER**
  Always looking for that landmark, event, or attraction.

- **CITY SLICKER**
  An urban creature who goes where the action is. Clubs, people... love the pulse of the city.

- **AVID ATHLETE**
  Always on the court or the course... always in the game... whatever game it is.

- **SHOPPING SHARK**
  Stopped looking for a cure for your shopaholism?

- **WINTER WARRIOR**
  Will work for lift ticket. Can become quite abominable if there’s no snow on the ground.

{pick one and click!}
Knowledge-based recommender systems

- **Constraint-based**
  - based on explicitly defined set of recommendation rules
  - fulfill recommendation rules

- **Case-based**
  - based on different types of similarity measures
  - retrieve items that are similar to specified requirements

- **Both approaches are similar in their conversational recommendation process**
Interacting with constraint-based recommenders

▪ Conjunctive Query: \( \sigma_{[\text{criteria}]}(P) \)
  
  \( P \): product assortment

  example: \( \sigma_{[\text{mpix} \geq 10, \text{price} < 300]}(P) = \{p_4, p_7\} \)

▪ The user specifies his or her initial preference
  – all at once or incrementally in a wizard-style

▪ The user is presented with a set of matching items
  – with explanation as to why a certain item was recommended

▪ The user might revise his or her requirements
  – see alternative solutions
  – narrow down the number of matching items
Constraint-based recommendation tasks

- Derive a set of recommendable items
- Find a set of user requirements such that a subset of items fulfills all constraints
  - ask user which requirements should be relaxed/modified such that some items exist that do not violate any constraint
- Find a subset of items that satisfy the maximum set of weighted constraints
- Rank items according to weights of satisfied constraints
- Provide Defaults
  - Static or Derived
Unsatisfied requirements

▪ "no solution could be found"

▪ Constraint relaxation
  – the goal is to identify relaxations to the original set of constraints
  – relax constraints of a recommendation problem until a corresponding solution has been found

▪ Users could also be interested in repair proposals
  – recommender can calculate a solution by adapting the proposed requirements
Constraint-based recommendation problem

- Select items from this catalog that match the user's requirements

<table>
<thead>
<tr>
<th>id</th>
<th>price($)</th>
<th>mpx</th>
<th>opt-zoom</th>
<th>LCD-size</th>
<th>movies</th>
<th>sound</th>
<th>waterproof</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁</td>
<td>148</td>
<td>8.0</td>
<td>4×</td>
<td>2.5</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>P₂</td>
<td>182</td>
<td>8.0</td>
<td>5×</td>
<td>2.7</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>P₃</td>
<td>189</td>
<td>8.0</td>
<td>10×</td>
<td>2.5</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>P₄</td>
<td>196</td>
<td>10.0</td>
<td>12×</td>
<td>2.7</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>P₅</td>
<td>151</td>
<td>7.1</td>
<td>3×</td>
<td>3.0</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>P₆</td>
<td>199</td>
<td>9.0</td>
<td>3×</td>
<td>3.0</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>P₇</td>
<td>259</td>
<td>10.0</td>
<td>3×</td>
<td>3.0</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>P₈</td>
<td>278</td>
<td>9.1</td>
<td>10×</td>
<td>3.0</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

- User's requirements can, for example, be
  - "the price should be lower than 300 $"
  - "the camera should be suited for sports photography"
Suppose,

\[ \text{REQ} = \{ r_1 : \text{price} \leq 150, \ r_2 : \text{opt-zoom}=5x, \ r_3 : \text{sound}=yes, \ r_4 : \text{waterproof}=yes \} \]

\[ \sigma_{[\text{price} \leq 150, \text{opt-zoom}=5x, \text{sound}=yes, \text{waterproof}=yes]} (P) = \emptyset \]

This requirement is not satisfiable on the given set of products.
Dealing with unsatisfied requirements

**Diagnosis**

A minimal set of user requirements whose repair (adaptation) will allow the retrieval of a recommendation.

- \( P = \{ p_1, p_2, ..., p_n \} \)
- \( \text{REQ} = \{ r_1, r_2, ..., r_m \} \)
- \( \sigma_{[\text{REQ}]}(P) = \emptyset \)

We have to find \( \Delta = \{ d_1, d_2, ..., d_k \} \)

Such that \( \sigma_{[\text{REQ} - \text{di}]}(P) \neq \emptyset \ \forall \ d_i \in \Delta \)
Deal with unsatisfied requirements

**Conflict set CS**

A subset $\{r_1, r_2, ..., r_l\} \subseteq \text{REQ}$, such that $\sigma_{[CS]}(P) = \emptyset$.

A conflict set $CS$ is minimal iff there does not exist a $CS'$ with $CS' \subset CS$.

The corresponding conflict sets are

$CS_1 = \{r_1, r_2\}, CS_2 = \{r_2, r_4\}$ and $CS_3 = \{r_1, r_3\}$
QuickXPlain

**QuickXPlain** \((P, REQ)\)

**Input:** trusted knowledge (items) \(P\); Set of requirements \(REQ\)

**Output:** minimal conflict set \(CS\)

if \(\sigma_{[REQ]}(P) \neq \emptyset\) or \(REQ = \emptyset\) then return \(\emptyset\)
else return \(QX'(P, \emptyset, \emptyset, REQ)\);

**Function** \(QX'(P, B, \Delta, REQ)\)

if \(\Delta \neq \emptyset\) and \(\sigma_{[B]}(P) = \emptyset\) then return \(\emptyset\);
if \(REQ = \{r\}\) then return \(\{r\}\);
let \(\{r_1, \ldots, r_n\} = REQ\);
let \(k = \frac{n}{2}\);
\(REQ_1 \leftarrow r_1, \ldots, r_k\) and \(REQ_2 \leftarrow r_{k+1}, \ldots, r_n\);
\(\Delta_2 \leftarrow QX'(P, B \cup REQ_1, REQ_1, REQ_2)\);
\(\Delta_1 \leftarrow QX'(P, B \cup \Delta_2, \Delta_2, REQ_1)\);
return \(\Delta_1 \cup \Delta_2\);
Example of QuickXPlain

- REQ = {r1: price ≤ 150, r2: opt-zoom = 5x, r3: sound = yes, r4: waterproof = yes}

(1) QX(P, \{r_1, r_2, r_3, r_4\})

(2) QX'(P, {}, {}, \{r_1, r_2, r_3, r_4\})

(3) QX'(P, \{r_1, r_2\}, \{r_1, r_2\}, \{r_3, r_4\})

(4) QX'(P, {}, {}, \{r_1, r_2\})

(5) QX'(P, \{r_1\}, \{r_1\}, \{r_2\})

(6) QX'(P, \{r_2\}, \{r_2\}, \{r_1\})
Deal with unsatisfied requirements

- Calculate diagnoses for unsatisfied requirements

  (1) $CS_1 = \{r_1, r_2\}$

  $\{r_1\}$

  $\{r_2\}$

  (2) $CS_2 = \{r_2, r_4\}$

  $\{r_2\}$

  $\{r_4\}$

  (3) $CS_3 = \{r_1, r_3\}$

  $\{r_1\}$

  $\{r_3\}$

  $d_1 = \{r_1, r_2\}$

  $d_2 = \{r_1, r_4\}$

  $d_3 = \{r_2, r_3\}$

- The diagnoses derived from the conflict sets $\{CS1, CS2, CS3\}$ are $\{d1: \{r1, r2\}, d2: \{r1, r4\}, d3: \{r2, r3\}\}$
Repairs for unsatisfied requirements

- Identify possible adaptations

- Or query the product table $P$ with $n[attributes(d)]\sigma[REQ−d](P)$
  - $n[attributes(d1)]\sigma[REQ−d1](P) = \{price=278, opt-zoom=10\times\}$
  - $n[attributes(d2)]\sigma[REQ−d2](P) = \{price=182, waterproof=no\}$
  - $n[attributes(d3)]\sigma[REQ−d3](P) = \{opt-zoom=4\times, sound=no\}$

<table>
<thead>
<tr>
<th>repair</th>
<th>price(€)</th>
<th>opt-zoom</th>
<th>sound</th>
<th>waterproof</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rep$_1$</td>
<td>278</td>
<td>10×</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Rep$_2$</td>
<td>182</td>
<td>√</td>
<td>√</td>
<td>no</td>
</tr>
<tr>
<td>Rep$_3$</td>
<td>√</td>
<td>4×</td>
<td>no</td>
<td>√</td>
</tr>
</tbody>
</table>
Case-based Approach

Items are retrieved based on similarity.

Critiquing
User specify their change requests that are not satisfied by the recommended item.

e.g.,
→ “lower price”
→ “more pixel”
Case-based Approach

Items are retrieved based on similarity.

Critiquing
User specify their change requests that are not satisfied by the recommended item.

e.g.,
→ “lower price”
→ “more pixel”
Conclusion & Summary
Conclusion

- None of the models discussed are perfect or optimal.

- The choice of model normally depends on the choice of the application.

- In practicality, for better performance combination of models are used rather than the pure form of any model.

- These systems are widely used in today's rapidly growing World Wide Web, and play a pivotal role in almost all major websites.
Collaborative Filtering
- “wisdom of the crowd”
- User-based or item-based CF
- Challenges in CF
  - Scalability -- clustering
  - Data Sparsity -- graph-based, matrix factorization

Content Based Recommendation

Knowledge Based Recommendation
- interactive conversational style
- based on explicit user choice only
References

- http://recommender-systems.org/content-based-filtering/
- Recommender Systems : An Introduction Dietmar Jannach etal
Backup Slides
RecTree Algorithm

**constructRecTree** (parent, data, depth)

create a node and link it to parent

if size(data) ≤ maxSize OR depth ≥ maxDepth:
    computeCorrelationMatrix(data)
else
    call K-Means(data, k = 2)
    for each child cluster from K-Means:
        call constructRecTree(node, cData, depth + 1)

*Time complexity*

O(n lg n/b) -- if maxDepth = lg n, and maxSize = b.