Recommendation Systems
- *Predict* items a user may be interested in based on information about the user and the items
- An effective way to help people cope with information overload
- Examples: Amazon, Netflix, Tivo, ...

So How Can We Do It?
- The content based approach
  - E.g. full text search results
- The user feedback based approach
  - E.g. rating
- *Which one is better?? Any room for improvement??*

Collaborative Filtering
- Rate items based on the ratings of other users *who have similar taste as you*

Problem Definitions
- **Prediction**
  - Given: a user and $k$ items
  - Return: predicted rating for each item
- **Recommendation**
  - Given: a user
  - Return: $k$ items from the database with the highest predicted rating

Basic Assumptions
- Items are evaluated by users explicitly or implicitly
  - Ratings, reviews
  - Purchases, browsing behaviors
  - ...
- We may map explicit and implicit evaluations to a rating scale, e.g. 1-5.
Heuristic

- People who agreed in the past are likely to agree in the future

Problem Formulation

- User-Item Matrix

<table>
<thead>
<tr>
<th>Item</th>
<th>Ken</th>
<th>Lee</th>
<th>Meg</th>
<th>Nan</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>5</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>??</td>
<td>2</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

So what would be Ken's rating for Item 6?

Pearson Correlation Coefficient

- Let \( x \) and \( y \) be two users, and \( r_{x,i} \) be the rating of item \( i \) by user \( x \)

\[
W_{x,y} = \frac{\text{cov}(x, y)}{\sigma_x \sigma_y} = \frac{\sum (r_{x,i} - \bar{r}_i)(r_{y,i} - \bar{r}_i)}{\sqrt{\sum (r_{x,i} - \bar{r}_i)^2} \sqrt{\sum (r_{y,i} - \bar{r}_i)^2}}
\]

So what is \( W_{\text{ken,lee}} \)?? what's the range of \( W_{i,j} \)?

Predicted Rating

- \( p_{x,i} \) is the predicted rating of item \( i \) by user \( x \)

\[
p_{x,i} = \bar{r}_i + \frac{\sum (r_{x,i} - \bar{r}_i) \times W_{x,i}}{\sum W_{x,i}}
\]

So what is \( p_{\text{ken,6}} \)??

Variations and Optimizations

- Similarity measure
- Significance weighting
- Item rating variance
- Neighborhood selection
- Combine neighborhood ratings

Similarity Measures ...

- Pearson Correlation
- Spearman Correlation
  - Uses ranks instead of raw rating scores
- Cosine similarity
- Mean squared difference
- Entropy-based
- ...
... Similarity Measures

Cosine similarity:
\[ \cos(X, Y) = \frac{X \cdot Y}{\|X\| \|Y\|} = \frac{\sum x_i y_i}{\sqrt{\sum x_i^2} \sqrt{\sum y_i^2}} \]

Mean squared difference:
\[ msd(X, Y) = \sum (x_i - y_i)^2 \]

Entropy-based association:
\[ h(X, Y) = -\sum p_{i,j} \ln p_{i,j} \]

Significance Weighting

Weight users in addition to the similarity measure
\[ W = \begin{cases} 
1 & n \geq 50 \\
\frac{n}{50} & n < 50 
\end{cases} \]
where \( n \) is the number of items rated by both users.

Item Rating Variance

- Some items are more telling about tastes than others
  - E.g. “Sleepless in Seattle” is more telling about taste than “Titanic”
  - Give more weight to items with high variance in ratings

Neighborhood Selection

- Select a subset of users for better performance and accuracy
  - Correlation threshold
  - Best \( n \) neighbors

Combine Neighborhood Ratings

- Deviation from mean
- Weighted average
- Weighted average of z-scores

Mean absolute deviation:
\[ s = \frac{1}{n} \sum |x_i - \bar{x}| \]

Standardized measurement (z-score):
\[ z_i = \frac{x_i - \bar{x}}{s} \]

Algorithm Quality Metrics

- Coverage – percentage of items for which the system can produce a prediction
- Accuracy
  - Statistical metrics
    - Mean Absolute Error (MAE)
- Decision-support metrics
- Efficiency
  - Throughput – number of recommendations per second
And The Winners Are

- Similarity measure
  - Pearson Correlation
  - Spearman Correlation*
- Significance weighting
- Neighborhood selection
  - Best $n$ neighbors with $n=20$
- Combine neighborhood ratings
  - Deviation from mean

Other Recommendation Algorithms

- Combine collaborative and content-based filtering
- Item-item collaborative filtering
- Bayesian networks
- ...

Some Libraries

- Taste – http://taste.sourceforge.net/
- COFE – http://eecs.oregonstate.edu/iis/CoFE/
- And more – http://en.wikipedia.org/wiki/Collaborative_filtering#Software_libraries

Non-personalized Recommendation

- What if the user is new to the site?
- What if the site itself is new, i.e. no previous user transactions?

Sales Transactions

| 11: Beef, Chicken, Milk |
| 12: Beef, Cheese |
| 13: Cheese, Boots |
| 14: Beef, Chicken, Cheese |
| 15: Beef, Chicken, Clothes, Cheese, Milk |
| 16: Chicken, Clothes, Milk |
| 17: Chicken, Clothes, Milk |
| 18: Beef, Milk |

Amazon-like recommendation:
Users who purchased milk also purchased the following items:
- Clothes
- Chicken

Support Count

- The support count, or frequency, of an itemset is the number of the transactions that contain the itemset
- Item, Itemset, and Transaction
- Examples:
  - $support\_count\{\{\text{beef}\}\}=5$
  - $support\_count\{\{\text{beef, chicken, milk}\}\}=??$
Frequent Itemset

- An itemset is frequent if its support count is greater than or equals to a minimum support count threshold.
  - \( \text{support_count}(X) \geq \text{min_sup} \)
- Frequent itemset mining

The Apriori Property

- All nonempty subsets of a frequent itemset must also be frequent.
- Or, if an itemset is not frequent, its supersets cannot be frequent either.

Finding Frequent Itemsets – The Apriori Algorithm

- Given \( \text{min_sup} \)
- Find the frequent 1-itemsets \( L_1 \)
- Find the frequent \( k \)-itemsets \( L_k \) by joining the itemsets in \( L_{k-1} \)
- Stop when \( L_k \) is empty

Apriori Algorithm Example

- Support 25%

\[ \begin{array}{c|c|c}
\text{TID} & \text{Transactions} \\
1 & 1, 2, 3 \\
2 & 1, 4 \\
3 & 6, 5 \\
4 & 1, 2, 4 \\
5 & 1, 2, 6, 4, 3 \\
6 & 2, 4, 3 \\
7 & 2, 6, 3 \\
8 & 1, 3 \\
\end{array} \]

\[ \begin{array}{c|c|c|c|c}
\text{Item} & \text{Support} & \text{L}_1 & \text{L}_2 \\
\hline
\text{beef} & 5 & (1) & (1) \\
\text{chicken} & 5 & (2) \\
\text{cheese} & 5 & (3) \\
\text{chicken} & 4 & (4) \\
\text{clothes} & 1 & (5) \\
\text{food} & 3 & (6) \\
\end{array} \]

\[ \begin{array}{c|c|c}
\text{C}_2 \to \text{L}_1 & \text{C}_3 \to \text{L}_2 \\
(1,2) & 3 & (1,2) \\
(1,3) & 3 & (1,3) \\
(1,4) & 3 & (1,4) \\
(1,6) & 1 \\
(2,3) & 4 & (2,3) \\
(2,4) & 2 & (2,4) \\
(2,6) & 3 & (2,6) \\
(3,4) & 1 \\
(3,6) & 3 & (3,6) \\
(4,6) & 1 \\
\end{array} \]
From $L_{k-1}$ to $C_k$

- Let $l_i$ be an itemset in $L_{k-1}$, and $l_i[j]$ be the $j$th item in $l_i$.
- Items in an itemset are sorted, i.e. $l_i[1]<l_i[2]<...<l_i[k-1]$.
- $l_1$ and $l_2$ are joinable if:
  - Their first $k-2$ items are the same, and
  - $l_1[k-1]<l_2[k-2]$.

From $C_k$ to $L_k$

- Reduce the size of $C_k$ using the Apriori property:
  - any $(k-1)$-subset of a candidate must be frequent, i.e. in $L_{k-1}$.
- Scan the dataset to get the support counts.

References

- Data Mining: Concepts and Techniques by Jiawei Han and Micheline Kamber.