# CS 277, Data Mining

# Recommender Systems

Padhraic Smyth
Department of Computer Science
Bren School of Information and Computer Sciences
University of California, Irvine

Thanks to Yehuda Koren and Jure Leskovec for contributing material for many of these slides.

# Reading on Recommender Systems (on Web page)

- Good overviews
  - Recommender systems, Melville and Sindwhani, Encyclopaedia of Machine Learning, 2010 (a good starting point)
  - Chapter on recommendation algorithms from the online text Mining of Massive Data Sets, Rajaraman, Leskovec, and Ullman.
  - Amazon.com recommendations: item-to-item collaborative filtering, Linden, Smith, and York, 2003 (overview of the basic components of Amazon's recommender system)

- Matrix Factorization
  - Matrix factorization techniques for recommender systems, Koren, Bell, Volinsky, IEEE Computer, 2009
  - Advances in collaborative filtering, Koren and Bell, chapter from the Recommender Systems Handbook, 2011

- Other Aspects
  - Recommender systems: from algorithms to user experience, Konstain and Riedl, 2012 (emphasizes that the user experience is important, not just predictive accuracy)
  - Factorization machines with libFM, S. Rendle, 2012, with associated publicly-available software for libFM

# "Ratings" Data

- Data with users u and items i
  - E.g., items are products purchases, movies viewed, songs listened to, etc

- Can represent as an N x M sparse binary matrix
  - N = number of users, M = number of items

- Entries $r_{ui}$

  **Explicit Ratings**: $r_{ui}$ = user u's rating of item i (e.g. on a scale of 1 to 5)

  **Implicit Ratings**: $r_{ui}$ = 1 if user u purchased/read/listened to item i

  $r_{ui}$ = 0 if no purchase or rating

  (note that 0 means a user's preference is unknown, not that they don't like the item)

- Automated recommender systems
  - Given a user and their ratings (if any) recommend to this user other items that the user may be interested in

# Examples of Recommender Systems

- Shopping
  - Amazon, eBay
- Movie and music recommendations:
  - Netflix, YouTube, Last.fm, Pandora
- News
  - New York Times, Yahoo! front page
- Reading
  - Goodreads
- Digital libraries
- Web page/blog recommendations

# So why do we need Automated Recommendations?

- Many “sellers” moving to online stores
  - For music and movies for example, virtually all sales are now online

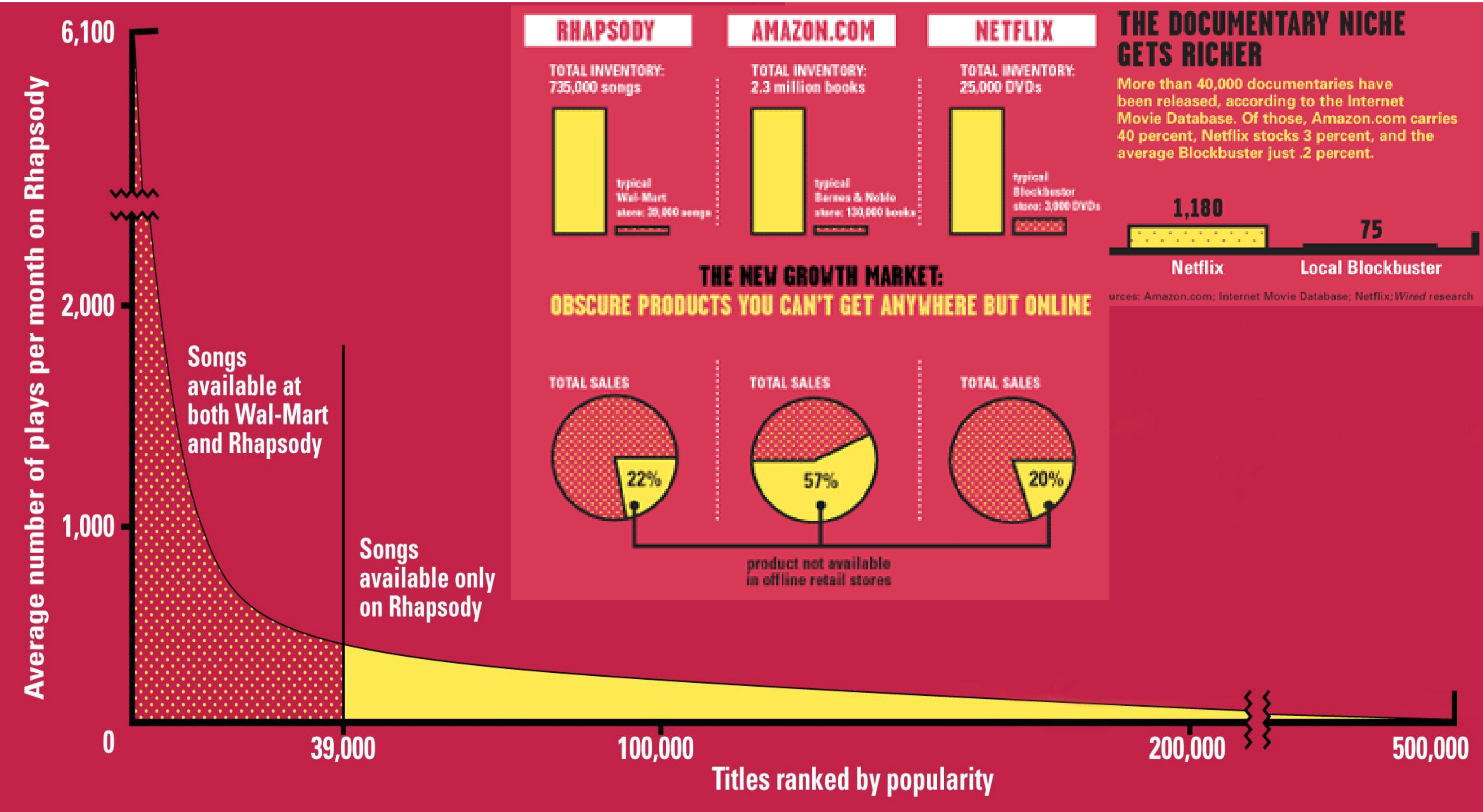
- Online store can stock many more items than a “brick and mortar” store

- But this brings a problem: how do customers find products they like when there are potentially millions of products?
  - Approach: Automated Recommendations

- Simple approaches:
  - Hand-crafted/editorial lists (e.g., for online newspapers)
  - Most-popular lists

- Personalized approaches
  - Recommendations personalized to each user, e.g., Amazon, Netflix, Facebook, etc

# Business Aspects of “The Long Tail”




Sources: Erik Brynjolfsson and Jeffrey Hu, MIT, and Michael Smith, Carnegie Mellon; Barnes & Noble; Netflix; RealNetworks

Source: Chris Anderson (2004)

# Example: Netflix Movie Ratings

100,000 movies

40 million users

| | | | | | |
|---|---|---|---|---|---|
| 1 | 3 | 4 | | | |
| | 3 | 5 | | | 5 |
| | | 4 | 5 | | 5 |
| | | 3 | | | |
| | | 3 | | | |
| 2 | | | 2 | | 2 |
| | | | | 5 | |
| | 2 | 1 | | | 1 |
| | 3 | | | 3 | |
| 1 | | | | | |

# The $1 Million Question

# Million Dollars Awarded Sept 21$^{st}$ 2009

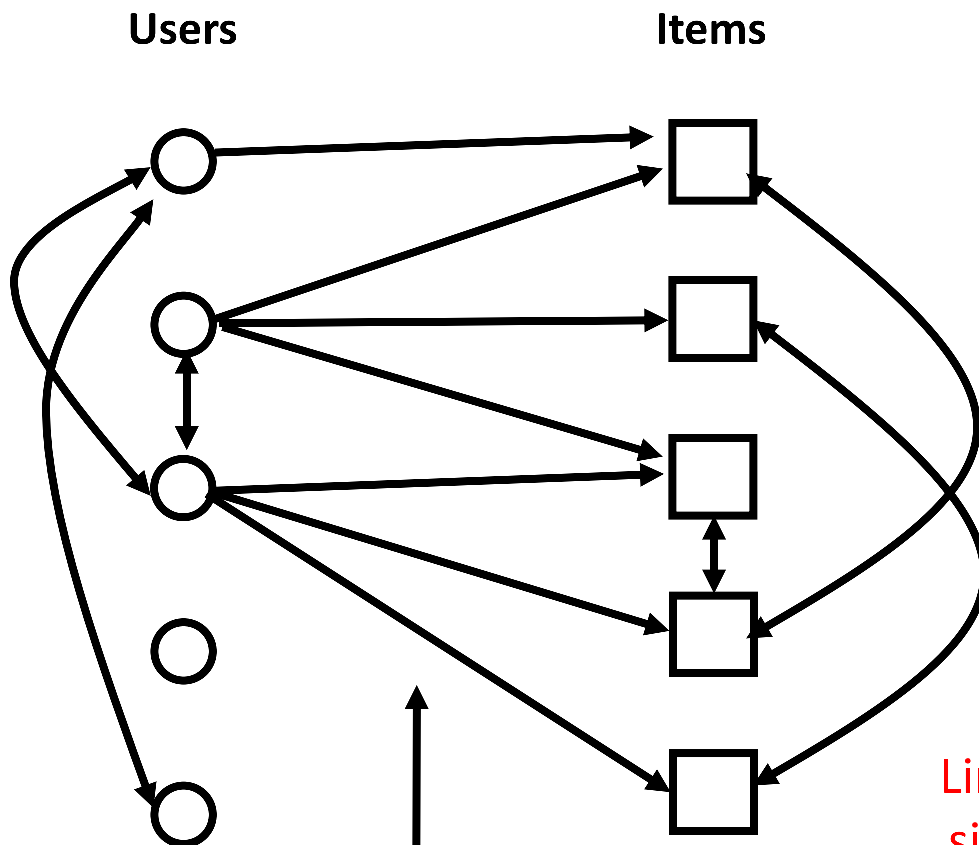
# Additional Content/Side Information

- Often have additional information about users and/or items

- Examples of additional user information
  - Search queries
  - Pages browsed
  - Demographics
  - Social network (connections to other users)

- Examples of additional item information
  - Item attributes, e.g., size, sales numbers
  - Item descriptions (e.g., in text format)
  - Item relationships (e.g., as a hierarchy)

# The Recommender Space as a Bipartite Graph




Links derived from similar attributes, explicit connections

User-User Links

Item-Item Links

Observed preferences

(Ratings, purchases, page views, play lists, bookmarks, etc)

Links derived from similar attributes, similar content, explicit cross references

# Challenges for Recommender Systems

- Data Sparsity
  - Users with very little historical data and few user attributes
  - Items with little or no content information

- "Cold Start" problem
  - How can we make  recommendations for new users? And new items?

# Gathering Ratings Data

- Explicit Ratings
  - E.g., Netflix movie ratings from 1 to 5
  - Difficult in practice: users often don't want to spend time assigning ratings
  - Bias and Variance in ratings
    - Research in cognitive science tells us that humans are not very good at consistently making judgements on a numerical scale
      (Easier for users to make A versus B judgements)
    - Bias: some users may consistently provide ratings lower than others
    - Variance: users may assign different ratings to the same item at different times

- Implicit Ratings
  - Ratings = user actions
  - E.g., purchased an item, listened to a song, viewed a movie for longer than k minutes
  - Issue here is that action need not imply preference, e.g., purchasing an item does not necessarily indicate a high or low preference

# Train/Test Setup (for Explicit Ratings)

Items

Training Users

| | | | | | |
|---|---|---|---|---|---|
| 1 | 3 | 4 | | | |
| | 3 | 5 | | | 5 |
| | | 4 | 5 | | 5 |
| | | 3 | | | |
| | | 3 | | | |
| 2 | | | 2 | | 2 |
| | | | | 5 | |
| | 2 | 1 | | | 1 |
| | 3 | | | 3 | |
| 1 | | | | | |

Test User

| | | | | | |
|---|---|---|---|---|---|
| 1 | | 3 | | 2 | 1 |

Predict each rating for a test user given all their other known ratings and the test data

# Evaluation Metrics

- Mean squared/absolute error in predicting all ratings for all test users
  - Assumes that all predictions are equally useful
  - E.g.,

$$\text{MSE} = 1/|R| \sum_{(u,i)\, \varepsilon\, R} ( r_{ui} - p_{ui} )^2$$

where $r_{ui}$ is the actual rating by user u of item i,
$p_{ui}$ is the algorithms prediction of user u's rating of item i,
and R is the set of ratings being used in the test set

- Precision-based metrics
  - E.g., rank the predictions and measure the precision of the top K predictions
  - Puts more emphasis on predicting positively rated items

# Evaluation with (Implicit) Binary Purchase Data

- Cautionary note:
  - It is not clear that prediction on historical data is a meaningful way to evaluate recommender algorithms, especially for purchasing
  - Consider:
    - User purchases products A, B, C
    - Algorithm ranks C highly given A and B
    - However, what if the user would have purchased C anyway?

      i.e., making this recommendation would have had no impact
      (or possibly a negative impact!)
  - What we would really like to do is reward recommender algorithms that lead the user to purchase products that they would not have purchased without the recommendation
    - This can't be done based on historical data alone
  - Requires direct "live" experiments (which is often how companies evaluate recommender algorithms)

# General Approaches to Automated Recommender Systems

1. Content-based Recommendations
   – Use attributes/features of items to recommend similar items
   – Ignores ratings data

2. Collaborative filtering
   – Use ratings matrix to recommend items, ignores item and user content data
   – 2 broad types:
     (1) Nearest-neighbor methods
     (2) Matrix factorization methods

3. Hybrid methods
   – Combine both content and ratings data (often provides “state of the art” performance)

# Content-Based Recommender Systems

# Content-Based Recommendation Algorithms

- Approach:
  - recommend items to user U that are similar to previous items that user U liked
  - Does not use ratings of other users when making predictions for user U

- "Similarity" is computed from item attributes, e.g.,
  - Similarity of movies by actors, director, genre
  - Similarity of text by words, topics
  - Similarity of music by genre, year

# Content-Based Recommendations

- Represent all items in some feature space
  - Item feature vectors $\underline{x}_1$, ..... $\underline{x}_M$
- "Map" any user into this same space (based on items they have rated)
  - User U has a feature vector $\underline{x}_U$
- Compute similarity of user's feature vector $\underline{x}_U$ to each item feature vector $\underline{x}_i$
- Definition of Sim($\underline{x}_U$ , $\underline{x}_i$ ) is critical

# Example: Text-Based Content

Items = documents,

Features = words or phrases

- Represent each document as as "bag of words"
- i.e., a vector with the counts of how often each word occurs in the document

Can use TF-IDF from information retrieval to weight features:

$TF_{ij}$ = frequency of term (feature) ***j*** in doc (item) ***i***

$n_i$ = number of docs that mention term ***i***

$N$ = total number of docs

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

"Downweights" terms that are more common

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

# Example: Text-Based Content

feature vector for item i, $\underline{x}_i$ = vector of words represented by their **TF-IDF** scores

feature vector for user u, $\underline{x}_u$ ?

can be computed as the average of the TF-IDF items rated by that user, weighted by the user's ratings

Prediction:

e.g., rank items using cosine similarity Cos ($\underline{x}_u$ , $\underline{x}_i$ ) $= \frac{xu \cdot xi}{||xu|| \cdot ||xi||}$

# Advantages of the Content-Based Approach

+ Only uses data from user U
  - So no need for data from other users
  - Easy to apply with few users

+ Can accommodate new items with no ratings
  - New items can be included as long as we can compute their feature vector

+ Can handle users with unique/unusual preferences

+ Can provide explanations
  - E.g., by listing the content features that caused an item to be recommended

# Weaknesses of the Content-Based Approach

- Items must have useful features on which to base similarity
- Finding which features are relevant to user tastes may be difficult
- Cannot make recommendations for new users
- May be sensitive to
  - similarity metric
  - definition of user features

# Collaborative Filtering for Recommender Systems:

# Nearest-Neighbor Algorithms

# User-User Neighborhood-Based Collaborative Filtering

- Idea: make recommendations for **active user a** based on finding similar users to user **a**
  - Let $\mathbf{K_a}$ be the set of K nearest-neighbors for user **a**
  - Generate predictions for **a** as a weighted combination of $\mathbf{K_a}$'s ratings

- Define similarity weight $\mathbf{w_{a,u}}$ between user **a** and user **u**,
  - e.g., linear correlation coefficient is often used:

$$w_{a,u} = \frac{\sum_{i \in I} (r_{a,i} - \overline{r}_a)(r_{u,i} - \overline{r}_u)}{\sqrt{\sum_{i \in I} (r_{a,i} - \overline{r}_a)^2 \sum_{i \in I} (r_{u,i} - \overline{r}_u)^2}}$$

where I is the set of items rated by **both** users, $\mathbf{r_{u,i}}$ is rating given by u to item I, and $\overline{r}_u$ is the mean rating of user u across items in set I

# User-User Neighborhood-Based Collaborative Filtering

## Prediction Step

$$p_{a,i} = \overline{r}_a + \frac{\sum_{u \in K} (r_{u,i} - \overline{r}_u) \times w_{a,u}}{\sum_{u \in K} w_{a,u}}$$

Where **$p_{a,i}$** is the prediction of rating for item **i** for user **a**,

$\overline{r}_u$ is the average rating of user u

**$w_{a,u}$** is the similarity weight between user **a** and user **u**,

**K** is the set of nearest users to **a**, as determined by **$w_{a,u}$**

Basically we are computing “deltas” for each user in **a**’s neighborhood, weighted by how similar these users are in general to **a**.

# Extensions/Options

- Significance weighting:
  - The active user can have highly correlated neighbors based on a very few co-rated items
  - This may yield poor predictions
  - Downweighting the similarity weight by a “significance weighting factor” can help here

- Downweight commonly rated items
  - Some items are rated/purchased by everyone, e.g., a Oscar-winning movie
  - These items are often not that useful as less common items
  - Can multiple original ratings by TF-IDF weighting, log $(n / n_i)$ where $n_i$ is the number of users who rated item i
  - Will tend to downweight the more common items (i.e., with high $n_i$ values)

- Various other heuristic modifications….

# User-User Near Neighbor Algorithm

1. For an active user **a**, find the K-nearest neighbor users using $\mathbf{w_{a,u}}$

2. Use the prediction equation to generate predicted rankings on all items

Naïve Time Complexity = O( N M ),
where N = number of users, M = number of items

More realistically:
Time Complexity = O( N r + M) where r = average # items rated by users

This is still too slow to perform in real-time,
e.g., Amazon ~ 100 million customers, ~ 10 million items

# Speed-Up Options for User-User Collaborative Filtering

- Reducing N
  - Randomly sample customers
  - Remove customers with few purchases (large fraction)
- Reducing M
  - Remove rarely purchased items
  - Do dimension reduction
- Cluster customers or items or both…

All of these methods tend to reduce the quality of neighbor-based collaborative filtering methods

Note also that it is usually not practical to do significant user-user computations offline since there is constantly new (and relevant) user data being generated

# Item-Item Collaborative Filtering

Match a user's rated items to similar items

- Tends to scale better than user-user CF methods (see Linden et al, Amazon paper)
- Items are fewer and potentially more stable than users

Similarity between items i and j computed using:

$$w_{i,j} = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_i)^2} \sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_j)^2}}$$

where U = set of users who have rated both items i and j

$\bar{r}_i$ = the average rating of item i across all users in set U

This can be computed offline

# Item-Item Collaborative Filtering

We can now predict the rating for item i by user a as follows:

$$p_{a,i} = \frac{\sum_{j \in K} r_{a,j} w_{i,j}}{\sum_{j \in K} |w_{i,j}|}$$

where K is the neighborhood set of k items, from the set rated by user a, that are most similar to item i

# Toy Example of Item-Item Collaborative Filtering

users

| movies | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | | 3 | | | 5 | | | 5 | | 4 | |
| 2 | | | 5 | 4 | | | 4 | | | 2 | 1 | 3 |
| 3 | 2 | 4 | | 1 | 2 | | 3 | | 4 | 3 | 5 | |
| 4 | | 2 | 4 | | 5 | | | 4 | | | 2 | |
| 5 | | | 4 | 3 | 4 | 2 | | | | | 2 | 5 |
| 6 | 1 | | 3 | | 3 | | | 2 | | | 4 | |

- unknown rating
- rating between 1 to 5

Figures and example courtesy of Jure Leskovec, Stanford

# Toy Example of Item-Item Collaborative Filtering

users

| movies | 1 | 2 | 3 | 4 | **5** | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | | 3 | | ? | 5 | | | 5 | | 4 | |
| 2 | | | 5 | 4 | | | 4 | | | 2 | 1 | 3 |
| 3 | 2 | 4 | | 1 | 2 | | 3 | | 4 | 3 | 5 | |
| 4 | | 2 | 4 | | 5 | | | 4 | | | 2 | |
| 5 | | | 4 | 3 | 4 | 2 | | | | | 2 | 5 |
| 6 | 1 | | 3 | | 3 | | | 2 | | | 4 | |

- estimate rating of movie **1** by user **5**

Figures and example courtesy of Jure Leskovec, Stanford

# Toy Example of Item-Item Collaborative Filtering

**users**

| movies | 1 | 2 | 3 | 4 | **5** | 6 | 7 | 8 | 9 | 10 | 11 | 12 | $w_{1,j}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | | 3 | | ? | 5 | | | 5 | | 4 | | **1.00** |
| 2 | | | | | | | | | | | | 3 | **-0.18** |
| 3 | | | | | | | | | | | | | **0.41** |
| 4 | | | | | | | | | | | | | **-0.10** |
| 5 | | | 4 | 3 | 4 | 2 | | | | | 2 | 5 | **-0.31** |
| 6 | 1 | | 3 | | 3 | | | 2 | | | 4 | | **0.59** |

**Using Pearson correlation as similarity:**

**1)** Subtract mean rating $r_i$ from each movie ***i***

$r_1 = (1+3+5+5+4)/5 = \mathbf{3.6}$

***row 1:*** *[-2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4, 0, 0.4, 0]*

**2)** Compute correlation between rows

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

Figures and example courtesy of Jure Leskovec, Stanford

# Toy Example of Item-Item Collaborative Filtering

users

| movies | 1 | 2 | 3 | 4 | **5** | 6 | 7 | 8 | 9 | 10 | 11 | 12 | $w_{1,j}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | | 3 | | ? | 5 | | | 5 | | 4 | | **1.00** |
| 2 | | | 5 | 4 | | | 4 | | | 2 | 1 | 3 | **-0.18** |
| **3** | 2 | 4 | | 1 | 2 | | 3 | | 4 | 3 | 5 | | **0.41** |
| 4 | | 2 | 4 | | 5 | | | 4 | | | 2 | | **-0.10** |
| 5 | | | 4 | 3 | 4 | 2 | | | | | 2 | 5 | **-0.31** |
| **6** | 1 | | 3 | | 3 | | | 2 | | | 4 | | **0.59** |

**Find the 2 nearest neighbors, with similarity weights**
**$w_{13}$=0.41, $w_{16}$=0.59**

Figures and example courtesy of Jure Leskovec, Stanford

# Toy Example of Item-Item Collaborative Filtering

users

movies

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | | 3 | | **2.6** | 5 | | | 5 | | 4 | |
| 2 | | | 5 | 4 | | | 4 | | | 2 | 1 | 3 |
| **3** | 2 | 4 | | 1 | 2 | | 3 | | 4 | 3 | 5 | |
| 4 | | 2 | 4 | | 5 | | | 4 | | | 2 | |
| 5 | | | 4 | 3 | 4 | 2 | | | | | 2 | 5 |
| **6** | 1 | | 3 | | 3 | | | 2 | | | 4 | |

**Predict by taking weighted average:**

**$P_{1,5}$ = (0.41*2 + 0.59*3) / (0.41+0.59) = 2.6**

$$p_{a,i} = \frac{\sum_{j \in K} r_{a,j} w_{i,j}}{\sum_{j \in K} |w_{i,j}|}$$

Figures and example courtesy of Jure Leskovec, Stanford

# Comparing User-User and Item-Item Methods

- Research studies indicate that item-item may produce more accurate ratings than user-user: why?
  - More stability across items than users (?)
  - More unusual/idiosyncratic users than items (?)
  - Item dimensionality is smaller (fewer items than users): so more data per item than per user

- Algorithmic advantage of Item-Item
  - Offline
    - build item similarity list
    - Can save time by skipping pairs of items with no common customers
  - Online predictions for an active user
    - Can be done relatively quickly
    - Depends on number of items rated by a user, and number of similar items

  - See paper by Linden, Smith, York on Amazon's item-item recommender system
    (note, there may be typos in their derivation of O(M+N) complexity for on 2nd page)

# Advantages/Weaknesses of Collaborative Filtering

- Advantages
  - Works for any kind of item – no features needed
  - Simple to implement – no feature engineering

- Weaknesses:
  - Not idea for “Cold Start”
    - needs to have users already in system to work
  - Cannot recommend items that have not already been rated
    - e.g., new items, unusual items
  - Popularity bias
    - Hard to make recommendations to someone with unique tastes
    - Can tend to recommend popular items

# Hybrid Methods

- Combine content-based and collaborative filtering
  - E.g., simply implement both methods and combine predictions linearly
  - Weights can be learned by cross-validation

- Integrate content features into collaborative filtering
  - Item features for new items
  - User features for new users

…. When we discuss matrix factorization we will discuss a systematic way to do this

# Next Lecture

- Matrix factorization approaches to recommender systems
- Stochastic gradient and related ideas
- The Netflix Prize competition