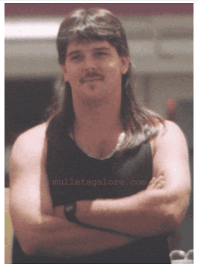
# Recommender Systems: Content-based Systems & Collaborative Filtering

Mining of Massive Datasets
http://www.mmds.org

---

## Example: Recommender Systems

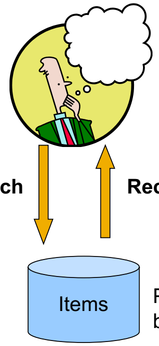


- **Customer X**
  - Buys Metallica CD
  - Buys Megadeth CD

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

---

## Recommendations



Search

Recommendations

Items

Products, web sites, blogs, news items, …

**Examples:**

amazon.com

PANDORA

StumbleUpon

del.icio.us

NETFLIX

movielens
helping you find the *right* movies

last.fm
the social music revolution

Google News

YouTube

XBOX LIVE

---

## From Scarcity to Abundance

- **Shelf space is a scarce commodity for traditional retailers**
  - Also: TV networks, movie theaters,…
- **Web enables near-zero-cost dissemination of information about products**
  - From scarcity to abundance
- **More choice necessitates better filters**
  - Recommendation engines

## Types of Recommendations

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

## Formal Model

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

## Utility Matrix

| | Avatar | LOTR | Matrix | Pirates |
|---|---|---|---|---|
| Alice | 1 | | 0.2 | |
| Bob | | 0.5 | | 0.3 |
| Carol | 0.2 | | 1 | |
| David | | | | 0.4 |

## Key Problems

- **(1) Gathering "known" ratings for matrix**
  - How to collect the data in the utility matrix
- **(2) Extrapolate unknown ratings from the known ones**
  - Mainly interested in high unknown ratings
    - We are not interested in knowing what you don't like but what you like
- **(3) Evaluating extrapolation methods**
  - How to measure success/performance of recommendation methods

## (1) Gathering Ratings

- **Explicit**
  - Ask people to rate items
  - Doesn’t work well in practice – people can’t be bothered
- **Implicit**
  - Learn ratings from user actions
    - E.g., purchase implies high rating
  - What about low ratings?

## (2) Extrapolating Utilities

- **Key problem:** Utility matrix ***U*** is **sparse**
  - Most people have not rated most items
  - **Cold start:**
    - New items have no ratings
    - New users have no history
- **Three approaches to recommender systems:**
  - **1)** Content-based
  - **2)** Collaborative
  - **3)** Latent factor based

## Content-based Recommender Systems

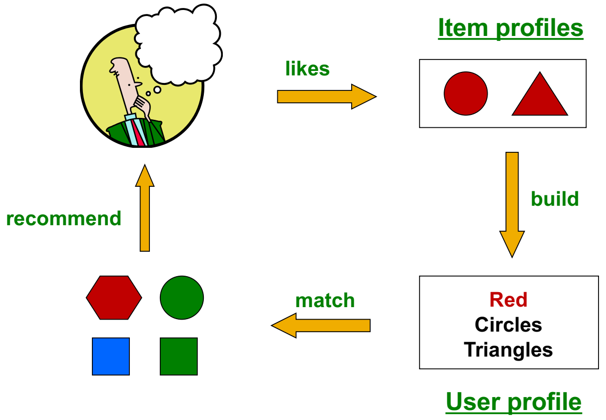
## Content-based Recommendations

- **Main idea:** Recommend items to customer ***x*** similar to previous items rated highly by ***x***

***Example:***

- **Movie recommendations**
  - Recommend movies with same actor(s), director, genre, …
- **Websites, blogs, news**
  - Recommend other sites with “similar” content

## Plan of Action



likes

Item profiles

build

Red
Circles
Triangles

User profile

match

recommend

## Item Profiles

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

## Sidenote: TF-IDF

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

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

**Note:** we normalize TF to discount for "longer" documents

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

$N$ = total number of docs

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

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

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

## User Profiles and Prediction

- **User profile possibilities:**
  - Weighted average of rated item profiles
  - **Variation:** weight by difference from average rating for item
  - …
- **Prediction heuristic:**
  - Given user profile $x$ and item profile $i$, estimate

$$u(\boldsymbol{x}, \boldsymbol{i}) = \cos(\boldsymbol{x}, \boldsymbol{i}) = \frac{\boldsymbol{x} \cdot \boldsymbol{i}}{||\boldsymbol{x}|| \cdot ||\boldsymbol{i}||}$$

## Pros: Content-based Approach

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

## Cons: Content-based Approach

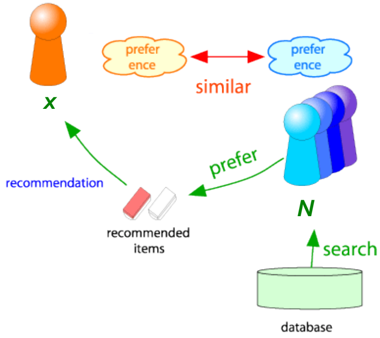
- **–: Finding the appropriate features is hard**
  - E.g., images, movies, music
- **–: Recommendations for new users**
  - **How to build a user profile?**
- **–: Overspecialization**
  - Never recommends items outside user’s content profile
  - People might have multiple interests
  - **Unable to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

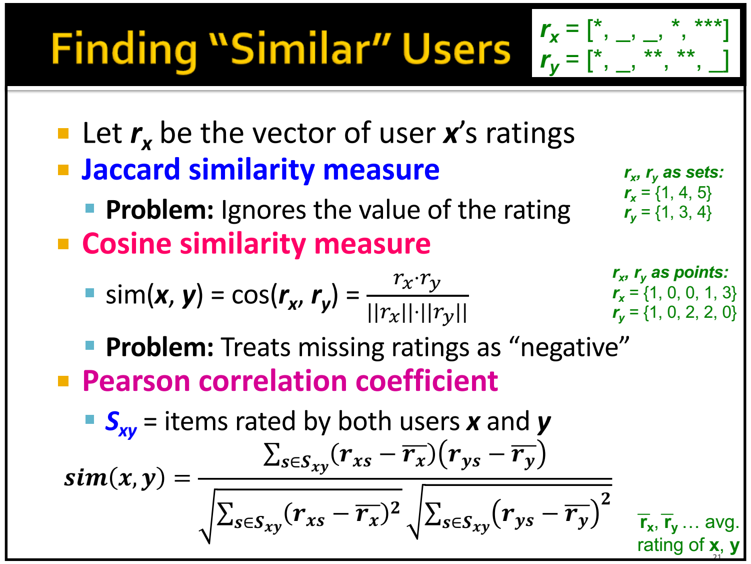
## Collaborative Filtering

Harnessing quality judgments of other users

## Collaborative Filtering

- Consider user ***x***
- Find set ***N*** of other users whose ratings are “**similar**” to ***x***’s ratings
- Estimate ***x***’s ratings based on ratings of users in ***N***





## Finding "Similar" Users

$r_x$ = [*, _, _, *, ***]
$r_y$ = [*, _, **, **, _]

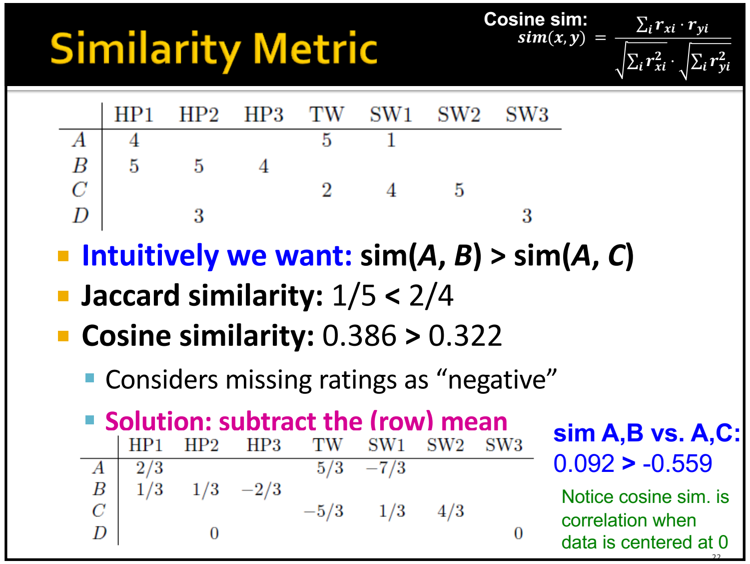
- Let $r_x$ be the vector of user *x*'s ratings
- **Jaccard similarity measure**
  - **Problem:** Ignores the value of the rating
- **Cosine similarity measure**
  - sim(*x*, *y*) = cos($r_x$, $r_y$) = $\frac{r_x \cdot r_y}{||r_x|| \cdot ||r_y||}$
  - **Problem:** Treats missing ratings as "negative"
- **Pearson correlation coefficient**
  - $S_{xy}$ = items rated by both users *x* and *y*

$$sim(x,y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})(r_{ys} - \overline{r_y})}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \overline{r_y})^2}}$$

***$r_x$, $r_y$ as sets:***
$r_x$ = {1, 4, 5}
$r_y$ = {1, 3, 4}

***$r_x$, $r_y$ as points:***
$r_x$ = {1, 0, 0, 1, 3}
$r_y$ = {1, 0, 2, 2, 0}

$\overline{r_x}$, $\overline{r_y}$ … avg. rating of **x**, **y**



## Similarity Metric

**Cosine sim:**
$$sim(x,y) = \frac{\sum_i r_{xi} \cdot r_{yi}}{\sqrt{\sum_i r_{xi}^2} \cdot \sqrt{\sum_i r_{yi}^2}}$$

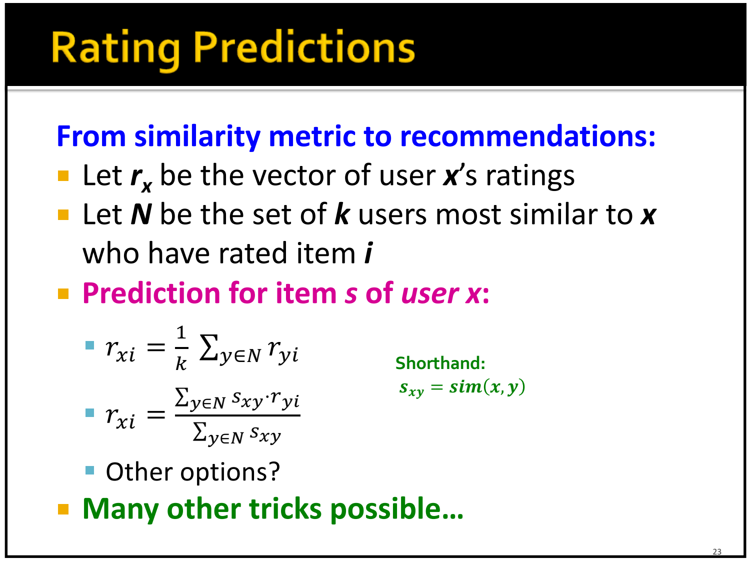
| | HP1 | HP2 | HP3 | TW | SW1 | SW2 | SW3 |
|---|---|---|---|---|---|---|---|
| *A* | 4 | | | 5 | 1 | | |
| *B* | 5 | 5 | 4 | | | | |
| *C* | | | | 2 | 4 | 5 | |
| *D* | | 3 | | | | | 3 |

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

| | HP1 | HP2 | HP3 | TW | SW1 | SW2 | SW3 |
|---|---|---|---|---|---|---|---|
| *A* | 2/3 | | | 5/3 | −7/3 | | |
| *B* | 1/3 | 1/3 | −2/3 | | | | |
| *C* | | | | −5/3 | 1/3 | 4/3 | |
| *D* | | 0 | | | | | 0 |

**sim A,B vs. A,C:**
0.092 **>** -0.559

Notice cosine sim. is correlation when data is centered at 0

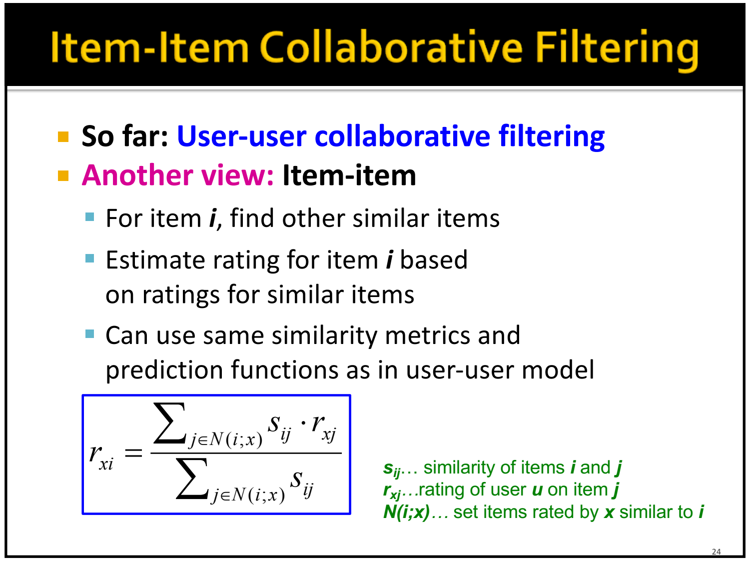


## Rating Predictions

**From similarity metric to recommendations:**

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

**Shorthand:**
$s_{xy} = sim(x,y)$

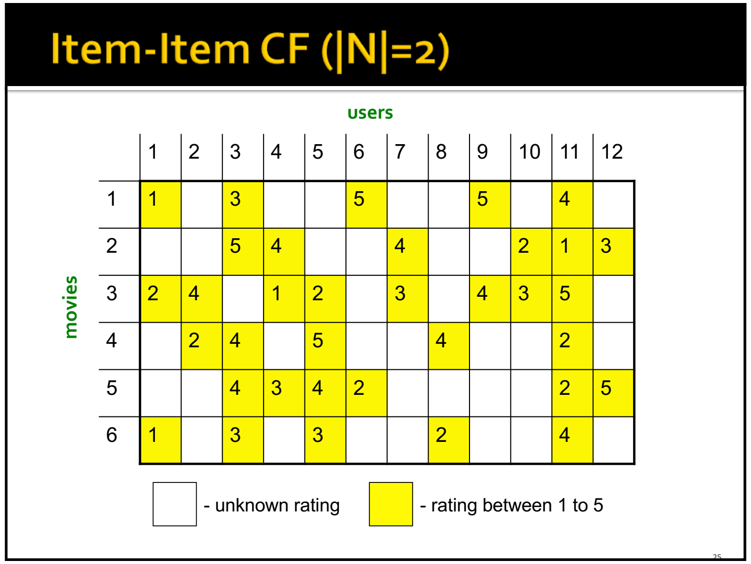


## Item-Item Collaborative Filtering

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

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

$s_{ij}$… similarity of items ***i*** and ***j***
$r_{xj}$…rating of user ***u*** on item ***j***
***N(i;x)***… set items rated by ***x*** similar to ***i***



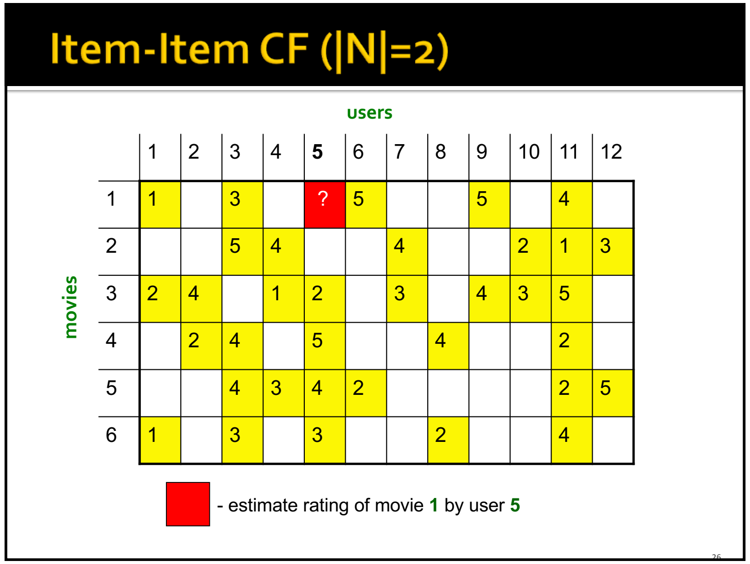
## Item-Item CF (|N|=2)

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



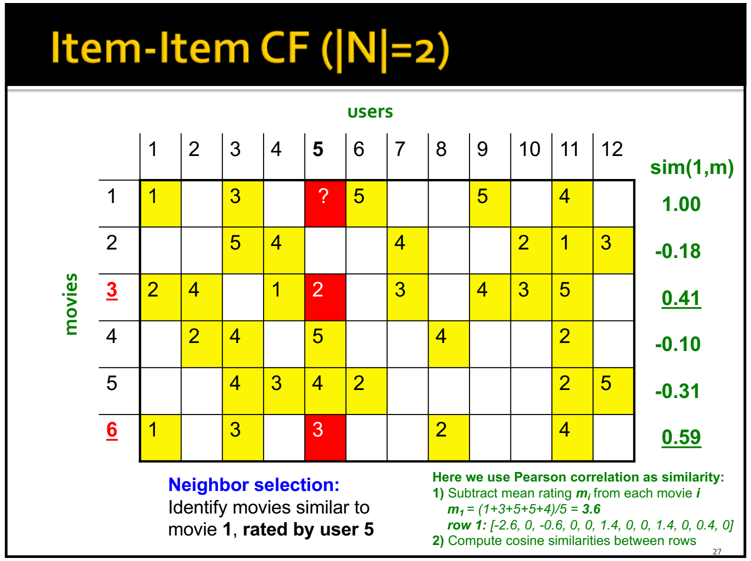
## Item-Item CF (|N|=2)

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**



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

users

movies

| | 1 | 2 | 3 | 4 | **5** | 6 | 7 | 8 | 9 | 10 | 11 | 12 | **sim(1,m)** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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** |

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

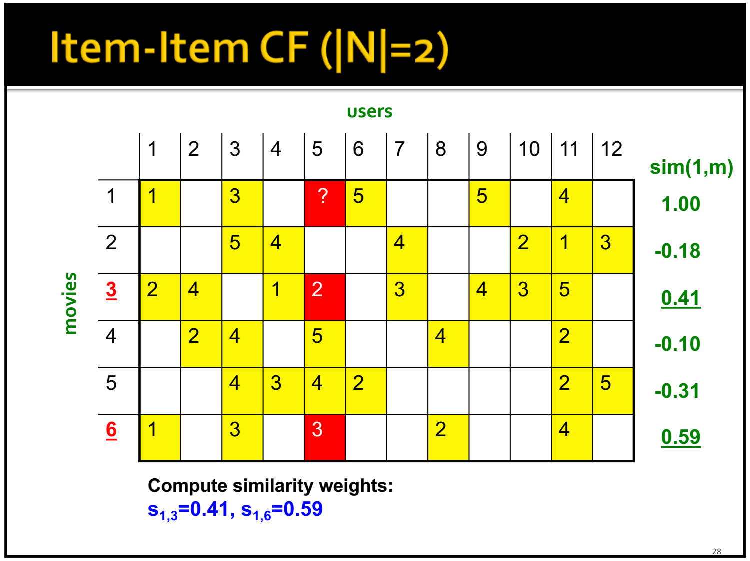
**Here we use Pearson correlation as similarity:**

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

$m_1 = (1+3+5+5+4)/5 = \mathbf{3.6}$

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

**2)** Compute cosine similarities between rows



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

users

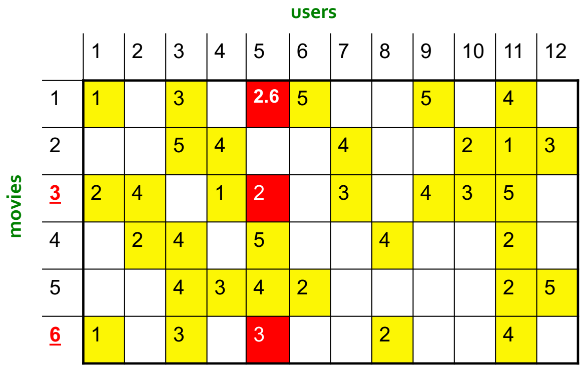
movies

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | **sim(1,m)** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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** |

**Compute similarity weights:**

$s_{1,3}$=0.41, $s_{1,6}$=0.59

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



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:**

$r_{1.5}$ = (0.41*2 + 0.59*3) / (0.41+0.59) = 2.6

$$r_{ix} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{jx}}{\sum s_{ij}}$$

# CF: Common Practice

**Before**:

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

- Define **similarity** $s_{ij}$ of items *i* and *j*
- Select *k* nearest neighbors *N(i; x)*
  - Items most similar to *i*, that were rated by *x*
- Estimate rating $r_{xi}$ as the weighted average:

$$r_{xi} = b_{xi} + \frac{\sum_{j \in N(i;x)} s_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i;x)} s_{ij}}$$

baseline estimate for $r_{xi}$

$$b_{xi} = \mu + b_x + b_i$$

- $\mu$ = overall mean movie rating
- $b_x$ = rating deviation of user *x*
  = *(avg. rating of user* ***x****)* – $\mu$
- $b_i$ = rating deviation of movie *i*

# Item-Item vs. User-User

| | Avatar | LOTR | Matrix | Pirates |
|---|---|---|---|---|
| Alice | 1 | | 0.8 | |
| Bob | | 0.5 | | 0.3 |
| Carol | 0.9 | | 1 | 0.8 |
| David | | | 1 | 0.4 |

- **In practice, it has been observed that item-item often works better than user-user**
- **Why?** Items are simpler, users have multiple tastes

# Pros/Cons of Collaborative Filtering

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

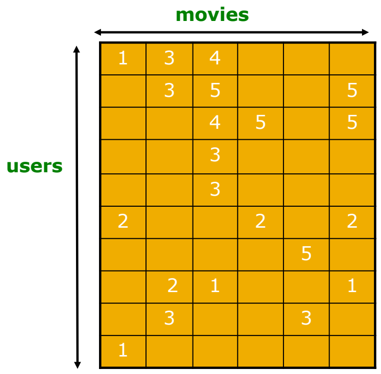
## Hybrid Methods

- **Implement two or more different recommenders and combine predictions**
  - Perhaps using a linear model
- **Add content-based methods to collaborative filtering**
  - Item profiles for new item problem
  - Demographics to deal with new user problem

## Remarks & Practical Tips

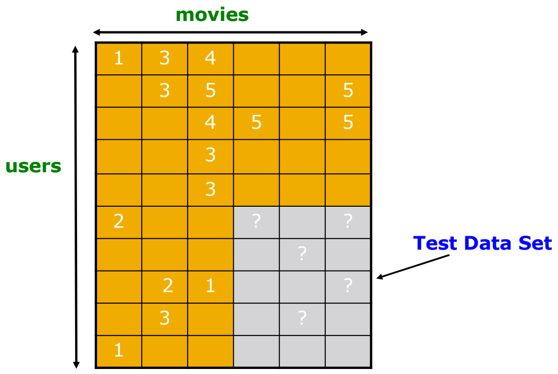
- Evaluation
- Error metrics
- Complexity / Speed

## Evaluation



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

## Evaluation



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

Test Data Set

## Evaluating Predictions

- Compare predictions with known ratings
  - **Root-mean-square error** (RMSE)
    - $\sqrt{\sum_{xi}(r_{xi} - r^{*}_{xi})^2}$ where $r_{xi}$ is predicted, $r^{*}_{xi}$ is the true rating of $x$ on $i$
  - **Precision at top 10**:
    - % of those in top 10
  - **Rank Correlation**:
    - Spearman's *correlation* between system's and user's complete rankings
- Another approach: 0/1 model
  - **Coverage:**
    - Number of items/users for which system can make predictions
  - **Precision:**
    - Accuracy of predictions
  - **Receiver operating characteristic** (ROC)
    - Tradeoff curve between false positives and false negatives

## Problems with Error Measures

- Narrow focus on accuracy sometimes misses the point
  - Prediction Diversity
  - Prediction Context
  - Order of predictions
- In practice, we care only to predict high ratings:
  - RMSE might penalize a method that does well for high ratings and badly for others

## Collaborative Filtering: Complexity

- Expensive step is finding ***k*** most similar customers: O(|X|)
- Too expensive to do at runtime
  - Could pre-compute
- Naïve pre-computation takes time **O(k ·|X|)**
    - X … set of customers
- We already know how to do this!
  - Near-neighbor search in high dimensions (**LSH**)
  - Clustering
  - Dimensionality reduction

## Tip: Add Data

- Leverage all the data
  - Don't try to reduce data size in an effort to make fancy algorithms work
  - Simple methods on large data do best
- Add more data
  - e.g., add IMDB data on genres
- More data beats better algorithms

http://anand.typepad.com/datawocky/2008/03/more-data-usual.html