# Collaborative Filtering

# Agenda

- **Collaborative Filtering (CF)**
  - Pure CF approaches
  - User-based nearest-neighbor
  - The Pearson Correlation similarity measure
  - Memory-based and model-based approaches
  - Item-based nearest-neighbor
  - The cosine similarity measure
  - Data sparsity problems
  - Recent methods (SVD, Association Rule Mining, Slope One, RF-Rec, …)
  - Discussion and summary

# Collaborative Filtering (CF)

- **The most prominent approach to generate recommendations**
  - used by large, commercial e-commerce sites
  - well-understood, various algorithms and variations exist
  - applicable in many domains (book, movies, DVDs, ..)
- **Approach**
  - use the "wisdom of the crowd" to recommend items

- **Basic assumption and idea**
  - Users give ratings to catalog items (implicitly or explicitly)
  - Customers who had similar tastes in the past, will have similar tastes in the future

# Pure CF Approaches

- **Input**
  - Only a matrix of given user–item ratings
- **Output types**
  - A (numerical) prediction indicating to what degree the current user will like or dislike a certain item
  - A top-N list of recommended items

# User-based nearest-neighbor collaborative filtering (1)

- **The basic technique**
  - Given an "active user" (Alice) and an item $i$ not yet seen by Alice
    - find a set of users (peers/nearest neighbors) who liked the same items as Alice in the past **and** who have rated item $i$
    - use, e.g. the average of their ratings to predict, if Alice will like item $i$
    - do this for all items Alice has not seen and recommend the best-rated
- **Basic assumption and idea**
  - If users had similar tastes in the past they will have similar tastes in the future
  - User preferences remain stable and consistent over time

# User-based nearest-neighbor collaborative filtering (2)

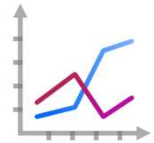
- **Example**
  - A database of ratings of the current user, Alice, and some other users is given:

| | Item1 | Item2 | Item3 | Item4 | Item5 |
|---|---|---|---|---|---|
| Alice | 5 | 3 | 4 | 4 | ? |
| User1 | 3 | 1 | 2 | 3 | 3 |
| User2 | 4 | 3 | 4 | 3 | 5 |
| User3 | 3 | 3 | 1 | 5 | 4 |
| User4 | 1 | 5 | 5 | 2 | 1 |

  - Determine whether Alice will like or dislike *Item5*, which Alice has not yet rated or seen

# User-based nearest-neighbor collaborative filtering (3)

- **Some first questions**
  - How do we measure similarity?
  - How many neighbors should we consider?
  - How do we generate a prediction from the neighbors' ratings?



| | Item1 | Item2 | Item3 | Item4 | Item5 |
|---|---|---|---|---|---|
| Alice | 5 | 3 | 4 | 4 | ? |
| User1 | 3 | 1 | 2 | 3 | 3 |
| User2 | 4 | 3 | 4 | 3 | 5 |
| User3 | 3 | 3 | 1 | 5 | 4 |
| User4 | 1 | 5 | 5 | 2 | 1 |

# Measuring user similarity (1)

- **A popular similarity measure in user-based CF: Pearson correlation**

  $a, b$ : users

  $r_{a,p}$ : rating of user $a$ for item $p$

  $P$ : set of items, rated both by $a$ and $b$

  – Possible similarity values between $-1$ and 1

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

# Measuring user similarity (2)

- **A popular similarity measure in user-based CF: Pearson correlation**

  $a, b$ : users

  $r_{a,p}$ : rating of user $a$ for item $p$

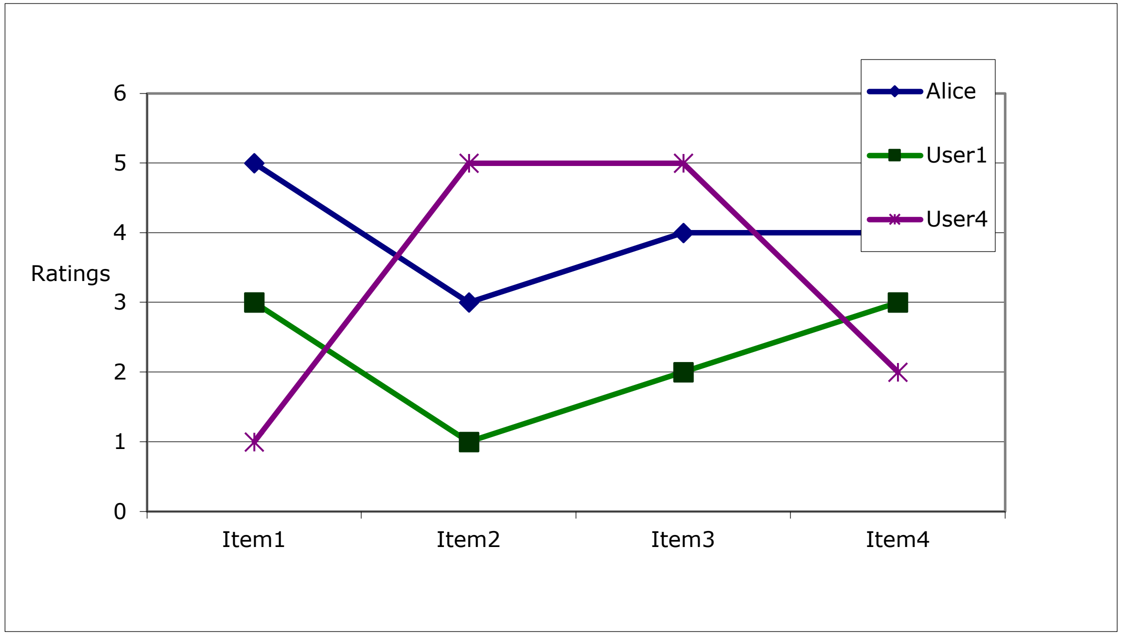
  $P$ : set of items, rated both by $a$ and $b$

  – Possible similarity values between $-1$ and 1

| | Item1 | Item2 | Item3 | Item4 | Item5 | |
|---|---|---|---|---|---|---|
| Alice | 5 | 3 | 4 | 4 | ? | |
| User1 | 3 | 1 | 2 | 3 | 3 | sim = 0,85 |
| User2 | 4 | 3 | 4 | 3 | 5 | sim = 0,00 |
| User3 | 3 | 3 | 1 | 5 | 4 | sim = 0,70 |
| User4 | 1 | 5 | 5 | 2 | 1 | sim = -0,79 |

# Pearson correlation

- **Takes differences in rating behavior into account**




- **Works well in usual domains, compared with alternative measures**
  - such as cosine similarity

## Making predictions

- A common prediction function:

$$\boldsymbol{pred(a,p) = \overline{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{b \in N} sim(a,b)}}$$

- Calculate, whether the neighbors' ratings for the unseen item $i$ are higher or lower than their average
- Combine the rating differences – use the similarity with $a$ as a weight
- Add/subtract the  neighbors' bias from the active user's average and use this as a prediction

# Improving the metrics / prediction function

- **Not all neighbor ratings might be equally "valuable"**
  – Agreement on commonly liked items is not so informative as agreement on controversial items
  – **Possible solution**: Give more weight to items that have a higher variance
- **Value of number of co-rated items**
  – Use "significance weighting", by e.g., linearly reducing the weight when the number of co-rated items is low
- **Case amplification**
  – Intuition: Give more weight to "very similar" neighbors, i.e., where the similarity value is close to 1.
- **Neighborhood selection**
  – Use similarity threshold or fixed number of neighbors

# Memory-based and model-based approaches

- **User-based CF is said to be "memory-based"**
  - the rating matrix is directly used to find neighbors / make predictions
  - does not scale for most real-world scenarios
  - large e-commerce sites have tens of millions of customers and millions of items
- **Model-based approaches**
  - based on an offline pre-processing or "model-learning" phase
  - at run-time, only the learned model is used to make predictions
  - models are updated / re-trained periodically
  - large variety of techniques used
  - model-building and updating can be computationally expensive
  - *item*-based CF is an example for model-based approaches

# Item-based collaborative filtering

- **Basic idea:**
  - Use the similarity between items (and not users) to make predictions
- **Example:**
  - Look for items that are similar to Item5
  - Take Alice's ratings for these items to predict the rating for Item5

| | Item1 | Item2 | Item3 | Item4 | Item5 |
|---|---|---|---|---|---|
| Alice | 5 | 3 | 4 | 4 | ? |
| User1 | 3 | 1 | 2 | 3 | 3 |
| User2 | 4 | 3 | 4 | 3 | 5 |
| User3 | 3 | 3 | 1 | 5 | 4 |
| User4 | 1 | 5 | 5 | 2 | 1 |

# The cosine similarity measure

- **Produces better results in item-to-item filtering**
- **Ratings are seen as vector in n-dimensional space**
- **Similarity is calculated based on the angle between the vectors**

$$\boldsymbol{sim(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|}}$$

- **Adjusted cosine similarity**
  - take average user ratings into account, transform the original ratings
  - $U$: set of users who have rated both items $a$ and $b$

$$\boldsymbol{sim(\vec{a}, \vec{b}) = \frac{\sum_{u \in U}(r_{u,a} - \overline{r_u})(r_{u,b} - \overline{r_u})}{\sqrt{\sum_{u \in U}(r_{u,a} - \overline{r_u})^2} \sqrt{\sum_{u \in U}(r_{u,b} - \overline{r_u})^2}}}$$

## Making predictions

- A common prediction function:

$$pred(u,p) = \frac{\sum_{i \in ratedItem(u)} sim(i,p) * r_{u,i}}{\sum_{i \in ratedItem(u)} sim(i,p)}$$

- Neighborhood size is typically also limited to a specific size
- Not all neighbors are taken into account for the prediction
- An analysis of the MovieLens dataset indicates that "in most real-world situations, a neighborhood of 20 to 50 neighbors seems reasonable" (Herlocker et al. 2002)

# Pre-processing for item-based filtering

- **Item-based filtering does not solve the scalability problem itself**
- **Pre-processing approach by Amazon.com (in 2003)**
  - Calculate all pair-wise item similarities in advance
  - The neighborhood to be used at run-time is typically rather small, because only items are taken into account which the user has rated
  - Item similarities are supposed to be more stable than user similarities
- **Memory requirements**
  - Up to $N^2$ pair-wise similarities to be memorized (N = number of items) in theory
  - In practice, this is significantly lower (items with no co-ratings)
  - Further reductions possible
    - Minimum threshold for co-ratings
    - Limit the neighborhood size (might affect recommendation accuracy)

# More on ratings – Explicit ratings

- Probably the most precise ratings
- Most commonly used (1 to 5, 1 to 7 Likert response scales)
- Research topics
  - Optimal granularity of scale; indication that 10-point scale is better accepted in movie dom.
  - An even more fine-grained scale was chosen in the joke recommender discussed by Goldberg et al. (2001), where a continuous scale (from −10 to +10) and a graphical input bar were used
    - No precision loss from the discretization
    - User preferences can be captured at a finer granularity
    - Users actually "like" the graphical interaction method
  - Multidimensional ratings (multiple ratings per movie such as ratings for actors and sound)
- Main problems
  - Users not always willing to rate many items
    - number of available ratings could be too small → sparse rating matrices → poor recommendation quality
  - How to stimulate users to rate more items?

# More on ratings – Implicit ratings

- Typically collected by the web shop or application in which the recommender system is embedded
- When a customer buys an item, for instance, many recommender systems interpret this behavior as a positive rating
- Clicks, page views, time spent on some page, demo downloads …
- Implicit ratings can be collected constantly and do not require additional efforts from the side of the user
- Main problem
  - One cannot be sure whether the user behavior is correctly interpreted
  - For example, a user might not like all the books he or she has bought; the user also might have bought a book for someone else
- Implicit ratings can be used in addition to explicit ones; question of correctness of interpretation