Collaborative Filtering

Data Mining

Prof. Sujee Lee

Department of Systems Management Engineering

Sungkyunkwan University

Recommender System

- **Recommender System**
  - Systems for recommending items (e.g. books, movies, CD’s, web pages) to users based on examples of their preferences
  - Many online stores provide recommendations (e.g. Amazon.com)
  - Recommenders have been shown to substantially increase sales at online stores
  - There is a very often used approach to recommending.
    - Collaborative Filtering

Collaborative Filtering

- **Collaborative Filtering**
  - A method for building recommendation systems.
  - Based on user interactions and preferences.

- **Types:**
  - **User-Based** Collaborative Filtering: Finds similar users to recommend items.
  - **Item-Based** Collaborative Filtering: Finds similar items to recommend.

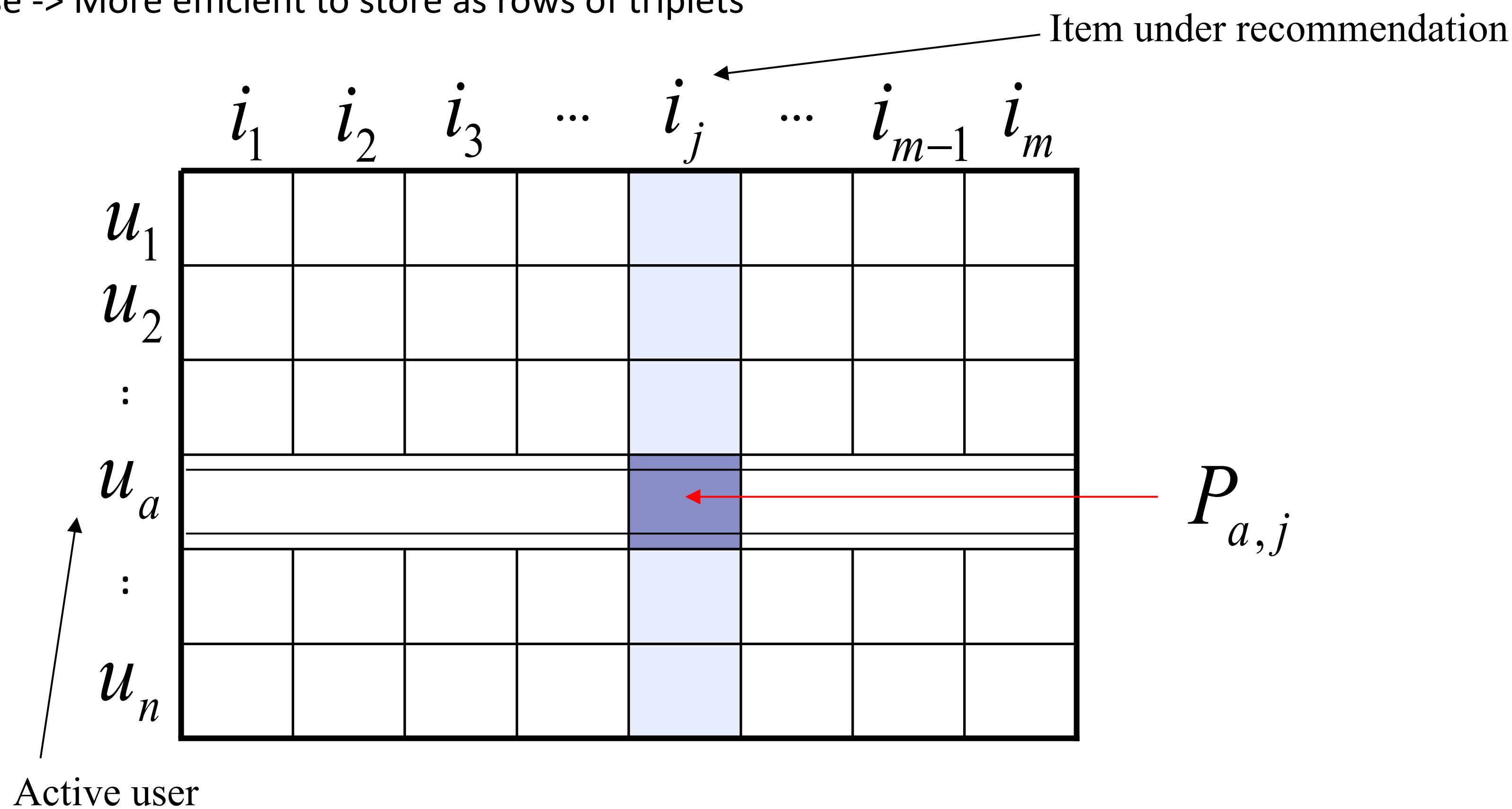
Types of Collaborative Filtering

- **User-Based Collaborative Filtering:**
  - Identifies similar users.
  - **Example**: If User A and User B have similar tastes, recommend items liked by User B to User A.
  - **Pros**: Simple and intuitive.
  - **Cons**: High computational cost with a large user base.

- **Item-Based Collaborative Filtering:**
  - Identifies items that are frequently used together or share similar features.
  - **Example**: Recommending similar movies to those a user already likes.
  - **Pros**: Scales better with large datasets.
  - **Cons**: Requires sufficient data for item similarity.

Data Structure for Collaborative Filtering

- **Rating Table (Item-user matrix)**
  - Cells are user preferences for items
  - Preferences can be ratings, or binary (buy, click, like)
  - Sparse -> More efficient to store as rows of triplets



Steps to Implement Collaborative Filtering

- **Processing Steps**
  - **Data Collection:**
    - Gather user activity data (e.g., ratings, clicks, purchases).
  - **Data Preprocessing:**
    - Handle missing values.
    - Normalize data (e.g., scale ratings to a consistent range).
  - **Similarity Calculation:**
    - Compute user-user or item-item similarities.
    - Similarity measure: Pearson correlation or cosine similarity
  - **Recommendation Generation:**
    - Predict missing ratings or ranks based on similarity.

User-Based Collaborative Filtering

- r_{ij} : **rating of user *i* on item *j***
- $\overline{r}_i$: **mean rating of user *i***
- I_i : **set of items on which user *i* has rated**
- $w(a,i)$: **similarity between user *i* and the active user *a***
- P_{aj} : **predicted rating of the active user *a* for item *j***
- J : **set of items on which user *i* and *a* has co-rated**
- S : **set of users whose *w(a,i)* can be computed**

$$\boxed{P_{aj} = \overline{r}_a + \kappa_a \sum_{i \in S} w(a,i)(r_{ij} - \overline{r}_i)}$$

$$\overline{r}_i = \frac{1}{|I_i|} \sum_{j \in I_i} r_{ij} \qquad w(a,i) = \frac{\sum_{j \in J} (r_{aj} - \overline{r}_a)(r_{ij} - \overline{r}_i)}{\sqrt{\sum_{j \in J} (r_{aj} - \overline{r}_a)^2 \sum_{j \in J} (r_{ij} - \overline{r}_i)^2}} \qquad \kappa_a = \frac{1}{\sum_{i \in S} |w(a,i)|}$$

- The list of top-N items is recommended to the active user.
- A good way to find a certain user's interesting item is to find other users who have a similar taste.

Example

		SF			Drama			Horror		
		Real Steel	Source Code	Rise of the Apes	Good Will Hunting	The Classic	Love Actually	Rite	Scream 4	Husk
SF Lovers	1	4	5	4		1	1	3	2	
	2	4	4	4				1	1	
	3	5	4		1	2		3	1	
Drama Lovers	4	1	2	1	4	3	5	2	2	2
	5	1	1		3	5	5			
	6		2		3	4	4	1	1	1
Horror Lovers	7	3	3	3	2	1	2	5	4	5
	8	1	2			3	1	4	4	
	9		1			1				5
Active User	10	5	3.87	3.91	1	1.56	1.36	2	1.71	1.73

Similarity Table (Pearson correlation coefficient)

	w(10,1)	w(10,2)	w(10,3)	w(10,4)	w(10,5)	w(10,6)	w(10,7)	w(10,8)	w(10,9)
New user 10	0.66	0.76	0.94	-0.89	-0.81	-0.12	0.05	-0.74	

Item-Based Collaborative Filtering

- r_{ui} : rating of user u on item i (5-star rating scheme is often used.)
- $\overline{r}_i$: mean rating of item i
- U : set of users that have co-rated on item i and j
- $sim(i, j)$: similarity between user i and the active user a
- P_{aj} : predicted rating of the active user a for item j
- U_i : set of users that have rated on item i

$$\boxed{P_{aj} = \frac{\sum_{i \in I_a} sim(i, j) r_{ai}}{\sum_{i \in I_a} |sim(i, j)|}}$$

$$\overline{r}_i = \frac{1}{|U_i|} \sum_{u \in U_i} r_{ui} \quad sim(i, j) = \frac{\sum_{u \in U} (r_{ui} - \overline{r}_i)(r_{uj} - \overline{r}_j)}{\sqrt{\sum_{u \in U} (r_{ui} - \overline{r}_i)^2 \sum_{u \in U} (r_{uj} - \overline{r}_j)^2}}$$

- The list of top-N items is recommended to the active user.
- The intuition behind this approach is that a user would be interested in purchasing items that are similar to the items the user liked earlier, and would tend to avoid items that are similar to the items the user didn't like.

Challenges

- **Cold Start Problem:**
  - Occurs when new users or items lack interaction data.
  - Solution: Combine with Content-Based Filtering.
- **Data Sparsity:**
  - User-item matrix is often mostly empty.
  - Solution: Use Matrix Factorization (e.g., SVD, ALS).
- **Scalability:**
  - Computational cost increases with large datasets.
  - Solution: Approximation techniques, distributed systems.