Recommender Systems

- Scenario:
  - Users have a potential interest in certain items
- Goal:
  - Provide recommendations for individual users
- Examples:
  - recommendations to customers in an on-line store
  - movie recommendations

	Book 1	Book 2	Book 3	Book 4	Book 5	Book 6
Customer A	X			X		
Customer B		X	X		X	
Customer C		X	X			
Customer D		X				X
Customer E	X				X	

Recommender Systems

- User provide recommendations
  - implicit
    (buying decisions, click streams, reading time of articles,...)
  - explicit
    (feedback forms, texts, mining public sources, ...)
- The system
  - computes recommendations
  - directs them to the right users
    - filter out items with negative recommendations
    - sort items
    - present evaluations
    - place ads tailored to the user

Example: amazon.com

- "If I have 2 million customers on the Web, I should have 2 million stores on the Web" (Jeff Bezos, CEO)

- Types of recommendations:
  - display of customer comments
  - personalized recommendations based on buying decisions
  - customers who bought also bought.... (books/authors/artists)
  - email notifications of new items matching pre-specified criteria
  - explicit feedback (rate this item) to get recommendations
  - customers provide recommendation lists for topics

Recommendation Techniques

- non-personalized recommendations
  - most frequently bought items (Harry Potter)
- attribute-based recommendations
  - books of the same authors
  - books with similar titles
  - books in same category

- item-to-item correlations
  - users who bought this book, also bought...
  - items are similar if they are bought by the same users
- user-to-user correlations
  - people like you also bought...
  - users are similar if they buy the same items

Attribute-Based Recommendations

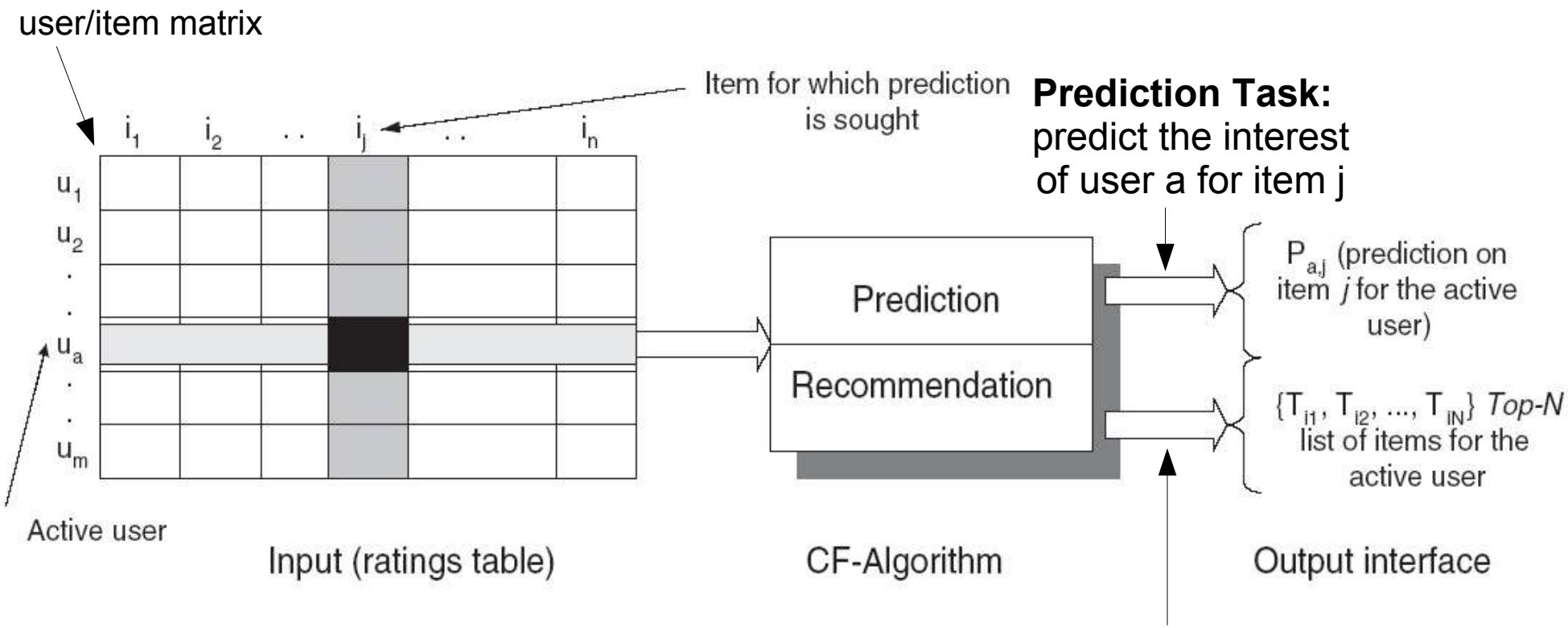
- Recommendations depend on properties of the items
- Each item is described by a set of attributes
  - Movies: e.g director, genre, year, actors
  - Documents: bag-of-word
- Similarity metric defines relationship between items
  - e.g. cosine similarity

Collaborative Filtering

- Recommends products to a target customer based on opinions of other customers

- REPRESENTATION:
  - user/item matrix (customer/product matrix)
  - similar to document/term matrix
- NEIGHBORHOOD FORMATION:
  - identify similar customers based on similar buying decisions / recommendations (e.g., cosine similarity), may be optional (i.e., all users are neighborhood)
- RECOMMENDATION GENERATION:
  - derive a recommendation based on the information obtained from similar customers (e.g., most frequent items in neighborhood, weighted sum,...)

Collaborative Filtering (CF)



- the matrix contains recommendations of each user for each product, e.g.
  - 1 if the user bought the item
  - 0 if the user did not buy the item

Source: Sarwar, Karypis, Konstan, Riedl, WWW-10, 2001

Memory-Based Collaborative Filtering

- For the Prediction Task, most systems use a formula like this:

$$v_p(u_a,i)=m(u_a)+\kappa\sum_{u\in U} w(u_a,u)(v(u,i)-m(u))$$

m(u) ... expected value (mean) over all votes of user
v(u,i) ... vote of user u for item i
v_p ... predicted vote
$w(u_1,u_2)$ ... weight between user u_1 and user u_2
u_a ... active user
κ... normalization factor for weights in the sum

- Weight matrix $w(u_1,u_2)$:
  - cosine similarity:

$$w(u_1,u_2)=\frac{\sum_{i\in I} v(u_1,i)v(u_2,i)}{\sqrt{\sum_{i\in I_{u_1}} v(u_1,i)^2 \sum_{i\in I_{u_2}} v(u_2,i)^2}}$$

  - correlation:
    = cosine similarity of adjusted votes $v_m(u,i)=v(u,i)-m(u)$ restricted to all items where both users vote

$$w(u_1,u_2)=\frac{\sum_{i\in I_{u_1}\cap I_{u_2}} v_m(u_1,i)v_m(u_2,i)}{\sqrt{\sum_{i\in I_{u_1}\cap I_{u_2}} v_m(u_1,i)^2 \sum_{i\in I_{u_1}\cap I_{u_2}} v_m(u_2,i)^2}}$$

Extensions

- Default Voting
  - default votes for items without explicit votes
  - allows to compute correlation from union instead of intersection (more items → more reliable)
- Inverse user frequency
  - reduce weights for objects popular with many users
  - *assumption:* universally liked items are less useful
    - cf. IDF
- Combine collaborative filtering with content-based similarities
  - user similarities:
    based on user profiles
  - item similarities:
    e.g., product categories, textual similarities, etc.

Extensions (Ctd.)

- Addition of pseudo users
  - use background knowledge (e.g., musical genres)
  - generate pseudo users that comment positively on all items of the genre
  - might be extracted automatically by wrappers (Cohen & Fan 2000)

Model-Based Collaborative Filtering

- learn an explicit model that predicts ratings and/or items
- examples
  - clustering of users
    - each user is characterized by her recommendations
    - apply any clustering algorithm that works for clustering documents
  - clustering of items
    - each item is characterized by the users that recommend it
    - apply any clustering algorithm that works for clustering documents
  - clustering of both users and items
    - advantage: items and users are mutually dependent, a good clustering needs to consider both dimensions.
  - association rules
    - model associations between items
    - advantage: explicit, understandable representation

Collaborative Filtering with Clustering

- Cluster users and items simultaneously
  - Mutual reinforcement of similarity
  - separate clusterings might be suboptimal
- Need advanced clustering techniques
  - e.g., Gibbs sampling (Ungar & Foster, 1998)

	Batman	Rambo	Andre	Hiver	Whispers	StarWars
Lyle			1			1
Ellen			1	1		1
Jason				1	1	
Fred	1					1
Dean	1	1				1
Karen	?	?	1	?	?	?

From *Clustering methods in collaborative filtering,* by Ungar and Foster

Association Rule Discovery

- Association Rules describe frequent co-occurences in sets

- Example Problems:
  - Which products are frequently bought together by customers? (*Basket Analysis*)
    - DataTable = Receipts x Products
    - Results could be used to change the placements of products in the market
  - Which courses tend to be attended together?
    - DataTable = Students x Courses
    - Results could be used to avoid scheduling conflicts....
  - Which words co-occur in a text?
    - cf. efficient generation of n-grams

Association Rules

- General Form:

$$A_1, A_2, ..., A_n \Rightarrow B_1, B_2, ..., B_m$$

- Interpretation:
  - When items A_i appear, items B_i also appear with a certain probability
- Examples:
  - `Bread, Cheese => RedWine.`
    Customers that buy bread and cheese, also tend to buy red wine.
  - `MachineLearning => WebMining, MLPraktikum.`
    Students that take 'Machine Learning' also take 'Web Mining' and the 'Machine Learning Praktikum'

Basic Quality Measures

- **Support** $s(A \to B) = \frac{n(A \cup B)}{n}$
  - relative frequency of examples for which both the head and the body of the rule are true
- **Confidence** $c(A \to B) = \frac{n(A \cup B)}{n(A)}$
  - relative frequency of examples for which the head is true among those for which the body is true
- Example:
  - `Bread, Cheese => RedWine` (S = 0.01, C = 0.8)

    80% of all customers that bought bread and cheese also bought red wine.
    1% of all customers bought all three items.

Using Association Rules for Recommendations

- APRIORI:
  - efficient algorithm for finding all rules that have a given *mimimum support* and a given *minimum confidence*
  - phase 1: find frequent item sets (-> n-grams)
  - phase 2: construct all rules with min confidence from item set

- Simple Use of APRIORI for recommendations:
  1. Input: database of all customers x all items they have bought
  2. Find association rules
  3. Find all rules whose conditions match the items previously bought by the active user
  4. Sort these rules by their confidence
  5. Predict the first N items on the top of the list