Data Mining Recommendation Systems

Joseph Burdis
Fall 2024
CUNY Graduate Center

Topic List

- Formulation of the recommendation system
- Content-based approach
- Collaborative filter approach

Reference: Mining Massive Datasets, Chapter 9, http://mmds.org

Problem Formulation

Recommendation System

- Amazon
- Spotify
- Netflix
- Youtube
- Google News
- ...

Recommendation Matters

In 1988, a British mountain climber named Joe Simpson wrote a book called *Touching the Void*, a harrowing account of near death in the Peruvian Andes. It only achieved a modest success and was soon forgotten.

A decade later, an American writer and mountaineer Jon Krakauer wrote a best-selling book *Into Thin Air*, another book about a mountain-climbing tragedy. It caused Touching the Void to sell again. The book was on the New York Time bestseller list for 14 weeks.

It was Amazon recommendation who suggested Touching the Void to readers of Into Thin Air. The latter book was actually out-of-print then. A new edition has to be published to meet readers' request.

Types of Recommendations

Editorial and hand curated

- List of favorites
- List of “essential” items

Simple aggregates

- Top 10
- Most popular

Tailored to individual users

- Amazon, Netflix, …

Formal Model: Utility Matrix

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

Key Problems

- How to collect data in the utility matrix?
    - Explicit: ask people to rate items
    - Implicit: learn ratings from user actions
- How to extrapolate unknown ratings from the known ones?
    - Most people have not rated most items
    - Cold start: new item, new user
- How to measure the performance of extrapolation methods?

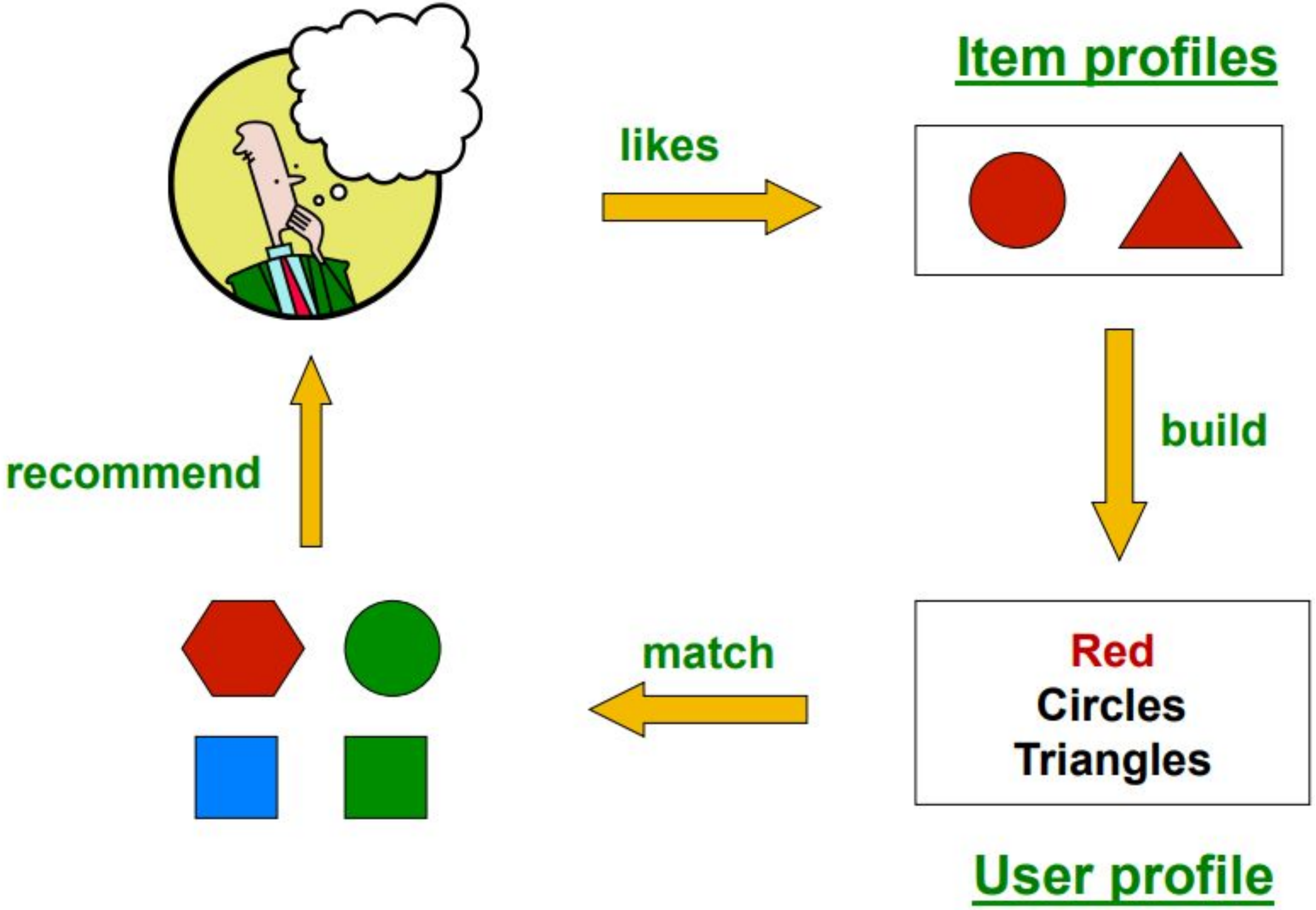
Content-Based Recommendations

Content-Based Recommendations

Main Idea: Recommend items to user x similar to previous items rated highly by x.

Challenge: How to measure similarity?

- Movie recommendations: same genre, same actors, same director, ...
- Websites: similar topic, similar content, similar opinion, ...


Item profiles
likes
build
Red
Circles
Triangles
User profile
match
recommend

Item Profiles

Content-based approach require an **item profile** for each item.

Item profile is a vector of features

- Movies: author, title, actors, director, ...
- Text: topic, keywords, ...

Challenge:

- How to effectively represent an item?
- How to identify important features?

Example: TF-IDF

For text mining, it is common to use TF-IDF to measure the importance of words.

TF-IDF: Term Frequency * Inverse Document Frequency

- If a word appears frequently in a document, it is likely to be important.
- If a word rarely appears in a random document but it appears here, it is likely to be important.

More precisely:

- TF: frequency of word / maximum word frequency in document
- IDF: log (total number of documents / number of docs that has this word)
- TF-IDF score: TF * IDF
- Usage: pick words with highest TF-IDF scores

User Profile

User profile can be created by averaging the related item profiles

- Weighted average of rated item profiles
- Or: weight by difference from average rating for each item

Measure similarity:

- Cosine distance: Given user profile x and item profile i:
- $u(x, i) = \cos(x, i) = x * i / (||x|| * || i ||)$

Pros and Cons

+ No need for data on other users
+ Able to recommend to users with unique tastes
+ Able to recommend new & unpopular items
+ Able to provide explanations

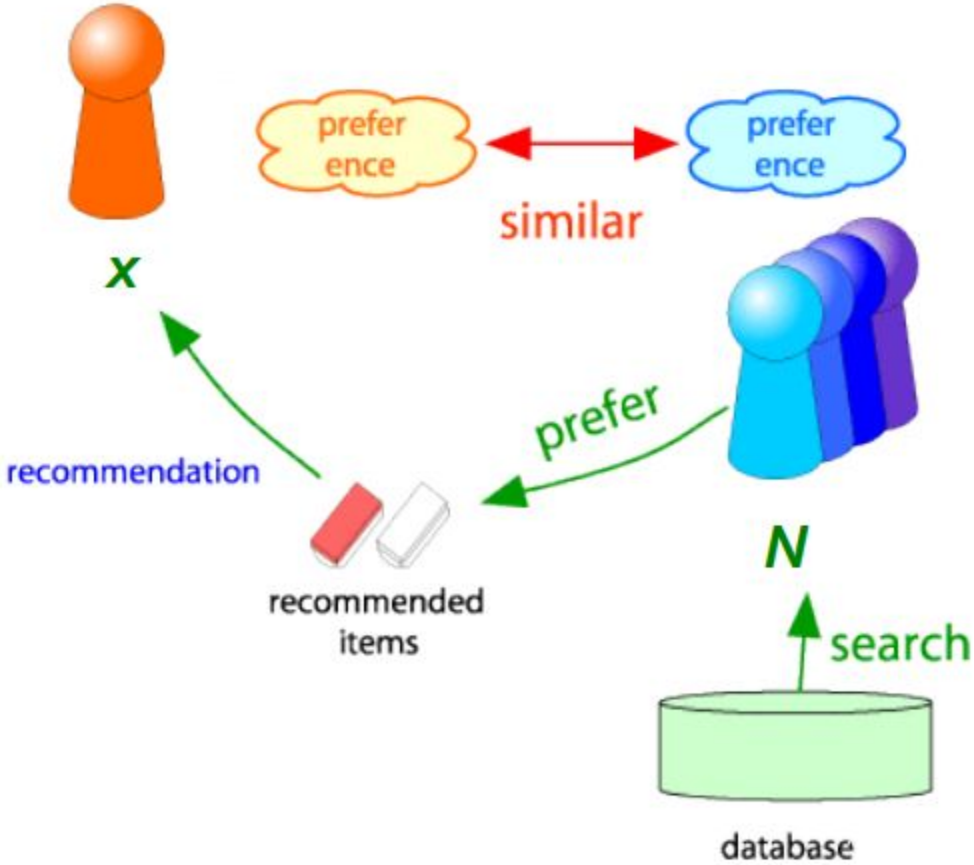
- Finding the appropriate features is challenging
- Hard to build profile for new users
- Never recommend items outside user's content profile
- Averaging of multiple user interests may result error
- Unable to exploit quality judgments of other users

Collaborative Filtering

Collaborative Filtering

Main idea: Find recommendations by looking at similar users

- 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 these N users
- Recommend items with potential high rating


prefer
ence
prefer
ence
similar
x
prefer
recommendation
N
recommended
items
search
database

Challenge: Finding Similar Users

- Represent each user’s ratings as a vector
- Jaccard similarity = |intersection| / |union|
    - Ignores value of the ratings
    - Only considers overlap of rated items
- Cosine similarity
    - Missing ratings have similar impacts as negative ratings
- Pearson correlation coefficient
- Matrix decomposition

	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

Let r_x be the vector of user x's ratings. Let N be the set of k users most similar to x.

Predicting user x's rating for item i:

- Option 1: $$r_{xi} = 1/k \sum_{y \in N} r_{yi}$$

- Option 2: $$r_{xi} = \sum_{y \in N} s_{xy} r_{yi} / \sum_{y \in N} s_{xy}$$

Item-Item Collaborative Filtering

So far we have been discussing user-user collaborative filtering. In practice, item-item filtering is more successful

- Each item in general receives a large amount of ratings
- Items are simpler than users to measure
- Item similarity is more meaningful than user similarity
- The nature of an item stays constant over time

The computation model is similar to user-user model.

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**

Example: Estimate Movie Rating

- Suppose N = 2
- Use Pearson correlation as similarity
- Use weighted average to predict movie rating

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:

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

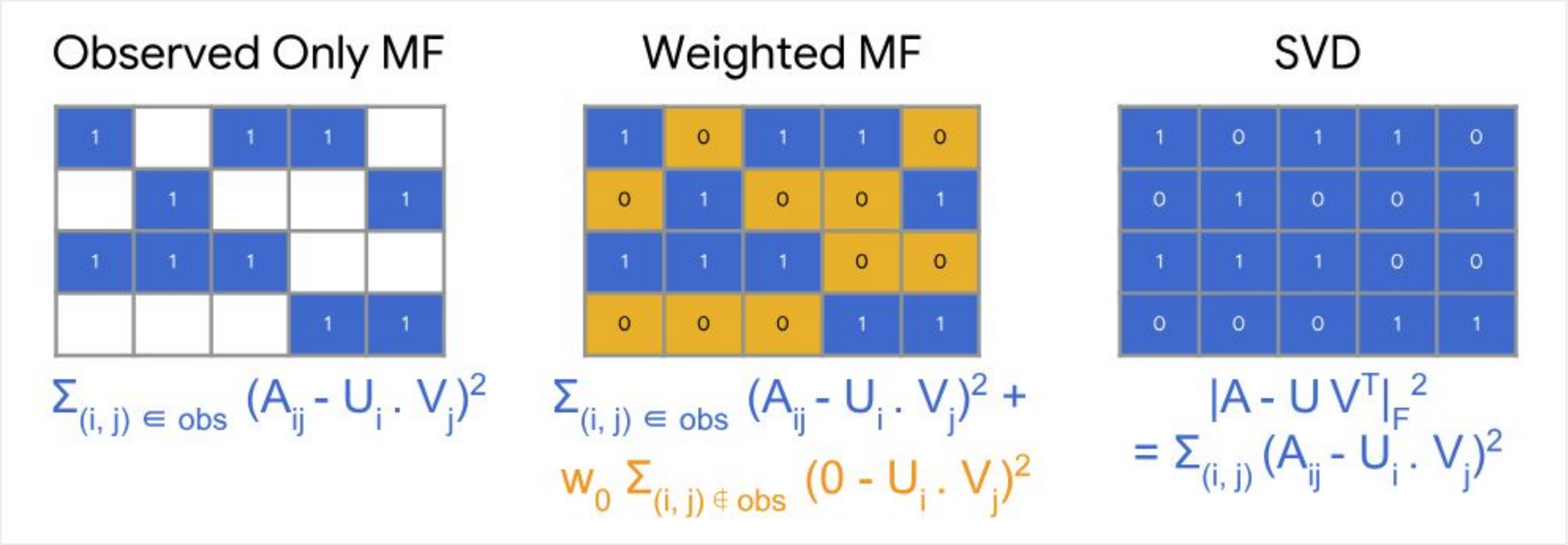
Matrix Factorization

Matrix factorization is a simple embedding model. Given the feedback matrix A $\in R^{m \times n}$, where m is the number of users (or queries) and n is the number of items, the model learns:

- A user embedding matrix $U \in \mathbb{R}^{m \times d}$, where row i is the embedding for user i.
- An item embedding matrix $V \in \mathbb{R}^{n \times d}$, where row j is the embedding for item j.

https://developers.google.com/machine-learning/recommendation/collaborative/matrix

Matrix Factorization



https://developers.google.com/machine-learning/recommendation/collaborative/matrix

Pros and Cons

+ No feature selection needed
- Cold start: need enough users in the system to find a match
- Sparsity: rating matrix is sparse
- First rater: new items has no ratings
- Popularity bias: tend to recommend popular items
- Cannot recommend items to someone with unique taste