UFMG UNIVERSIDADE FEDERAL DE MINAS GERAIS

Recommender Systems

Item-based Collaborative Filtering

Rodrygo Santos
rodrygo@dcc.ufmg.br

CF formulation

UFMG COMPUTER SCIENCE

- *What if user u has historical preferences?*

			i					
			1	3		2		
	1	2	5		4		1	
	4			3	5		4	
u		2	?		5	4		4
		3	4		5		3	

Personalized CF

UFMG COMPUTER SCIENCE

- Memory-based
  - Exploit similarities between users or items
    - Nearest neighbor approaches
    - Graph-based approaches
- Model-based
  - Build a model of user-item interactions
    - Bayesian models
    - Clustering models
    - Latent semantic models
    - ....

3

Personalized CF

UFMG COMPUTER SCIENCE

- Memory-based
  - Exploit similarities between users or items
    - Nearest neighbor approaches
    - Graph-based approaches

OUR FOCUS

4

Memory-based CF

UFMG COMPUTER SCIENCE

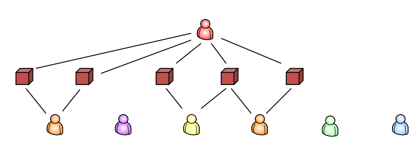
- How will I like item *i*?
  - User-based CF
    - *How similar users like item i?*

5

User-based CF

UFMG COMPUTER SCIENCE

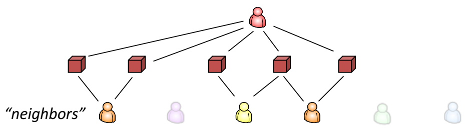
- Look for similar users



6

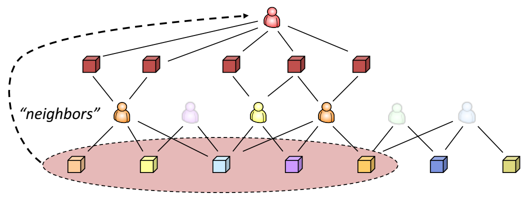
User-based CF

- Look for similar users



User-based CF

- Recommend what they like



User-based CF limitations

- User-based CF is great…
  - *… except for* ***sparsity***
- Too many items (think *"tens of millions"*)
  - Too few ratings per user (think *"hundreds"*)
- Hard to find neighbors
  - Complete failure to recommend
- Hard to trust neighbors
  - Noisy recommendations

User-based CF limitations

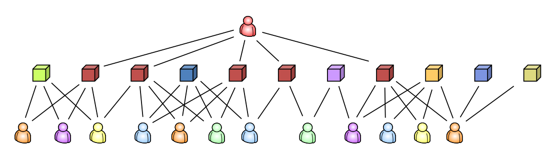
- User-based CF is great…
  - *… except for* ***efficiency***
- Computing all pairwise correlations is O(m^2n)
  - Infeasible to compute online
- Offline precomputation is problematic
  - User profiles are unstable

Memory-based CF

- How will I like item *i*?
  - User-based CF
    - *How similar users like item i?*
  - Item-based CF
    - *How do I like items similar to i?*

Item-based CF

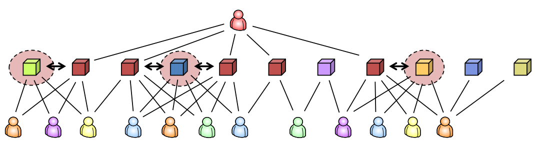
- Look for similar items



Item-based CF

UFMG COMPUTER SCIENCE

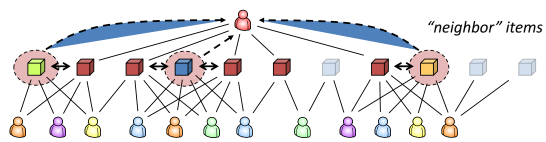
- Look for similar items



Item-based CF

UFMG COMPUTER SCIENCE

- Recommend these items



Item-based CF

UFMG COMPUTER SCIENCE

- Resilience to sparsity
  - Average user has a few ratings
  - Average item has lots of ratings
  - ***Better coverage and confidence in similarities***
- Resilience to changes after a new rating
  - Just another of many ratings for the item
  - Could completely redefine the rater's profile
  - ***Better stability to allow precomputation***

Memory-based CF

UFMG COMPUTER SCIENCE

- How will I like item *i*?
  - User-based CF
    - *How similar users like item i?*
  - Item-based CF
    - *How do I like items similar to i?*
- Key difference: neighborhoods
  - User-based: unstable, hard to precompute
  - Item-based: stable, easy to precompute

Breaking it down

UFMG COMPUTER SCIENCE

- Similar to user-based CF
  - Data normalization
  - Similarity computation
  - Neighborhood selection
  - Rating aggregation

Breaking it down

UFMG COMPUTER SCIENCE

- ***Data normalization***
  - Mean centering (for graded feedback)
    - Subtract user's mean
    - Subtract item's mean
  - Unit centering (for binary feedback)
    - Divide by user's Euclidean norm $\|\vec{u}\|$

Breaking it down

- ***Similarity computation***
  - Pearson's correlation
  - Cosine similarity
    - Adjusted after normalization

Breaking it down

- ***Neighborhood selection***
  - *k* items most similar to…
    - … items rated by ***u***
    - … items just viewed by ***u***
    - … items added to ***u***'s basket
  - What *k*?
    - Small *k* → inaccurate scores (few neighbors)
    - Large *k* → too much noise (low-similarity neighbors)
    - *k* = 20 is a good starting point

How to find neighbors?

- *Who are **i**'s nearest neighbors (rated by **u**)?*

		x	*i*		*y*	*z*		*w*
			1	3		2		
	1	2	5		4		1	
	4			3	5		4	
u		2	?		5	4		4
		3	4		5		3	

$sim(\vec{i},\vec{x}) = 0.82$

$sim(\vec{i},\vec{y}) = 0.65$

$sim(\vec{i},\vec{z}) = 0.07$

$sim(\vec{i},\vec{w}) = 0.00$

How to find neighbors?

- *Who are **i**'s nearest neighbors (rated by **u**)?*

		x	*i*		*y*			
			1	3		2		
	1	2	5		4		1	
	4			3	5		4	
u		2	?		5	4		4
		3	4		5		3	

$sim(\vec{i},\vec{x}) = 0.82$

$sim(\vec{i},\vec{y}) = 0.65$

2-NN

Model building

- Stability allows making it model-based
  - Precompute similarities for all pairs
  - Model contains list of neighbors for each item
- Still a costly operation
  - Naively: O(n^2m)
  - For symmetric similarity functions
    - Only need to compute one direction
  - Can often skip pairs
    - e.g., items without a common rater

Model storage

- No need to keep all neighbors
  - More neighbors → better coverage
  - Less neighbors → better efficiency
- Balance memory usage and accuracy
  - Keep enough neighbors to recommend (typically, *k* << *x* << *n*)

Breaking it down

UFMG COMPUTER SCIENCE

- ***Rating aggregation***
  - Min / max / average / median rating
  - Weighted average (by similarity)
  - Supervised aggregation
  - Common practice
    - Weighted average: simple and effective

How to find neighbors?

UFMG COMPUTER SCIENCE

- *Who are **i**'s nearest neighbors (rated by **u**)?*

		x	i		y			
			1	3		2		
	1	2	5		4		1	
	4			3	5		4	
u		2	?		5	4		4
		3	4		5		3	

$sim(\vec{i},\vec{x}) = 0.82$

$sim(\vec{i},\vec{y}) = 0.65$

2-NN

How to predict?

UFMG COMPUTER SCIENCE

- *How will user **u** like item **i**?*

		x	i		y			
			1	3		2		
	1	2	5		4		1	
	4			3	5		4	
u		2	?		5	4		4
		3	4		5		3	

$sim(\vec{i},\vec{x}) = 0.82 \quad r_{ux} = 2$

$sim(\vec{i},\vec{y}) = 0.65 \quad r_{uy} = 5$

How to predict?

UFMG COMPUTER SCIENCE

- *How will user **u** like item **i**?*

		x	i		y			
			1	3		2		
	1	2	5		4		1	
	4			3	5		4	
u		2	3.3		5	4		4
		3	4		5		3	

$$\hat{r}_{ui} = \frac{\sum_{j \in N} sim(\vec{i},\vec{j}) \times r_{uj}}{\sum_{j \in N} \left| sim(\vec{i},\vec{j}) \right|}$$

$$\hat{r}_{ui} = \frac{0.82 \times 2 + 0.65 \times 5}{0.82 + 0.65}$$

$$\hat{r}_{ui} = 3.33$$

Summary

UFMG COMPUTER SCIENCE

- Item-based CF is ***effective***
  - More resilient to data sparsity
- Item-based CF is ***efficient***
  - Stability allows neighborhood precomputation
- Item-based CF is ***flexible***
  - Profile-based neighborhood
  - Session-based neighborhood
  - Basket-based neighborhood