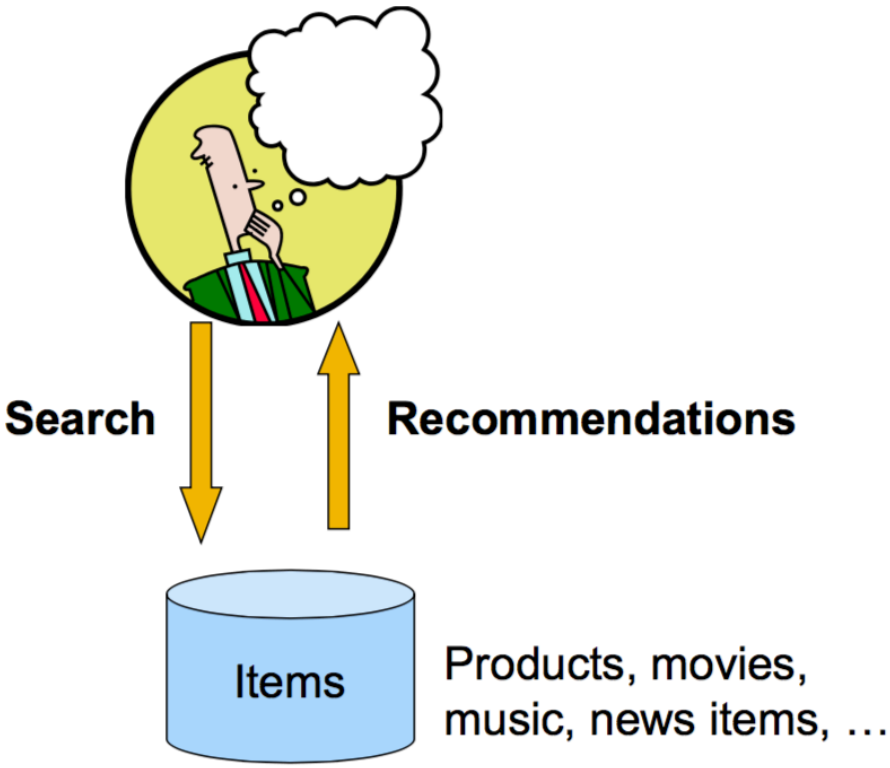
Making Recommendations: Content Filtering Collaborative Filtering Latent Factors

Jay Urbain, Ph.D.
Electrical Engineering and Computer Science Department
Milwaukee School of Engineering

Credits: See last page with references

Recommendations

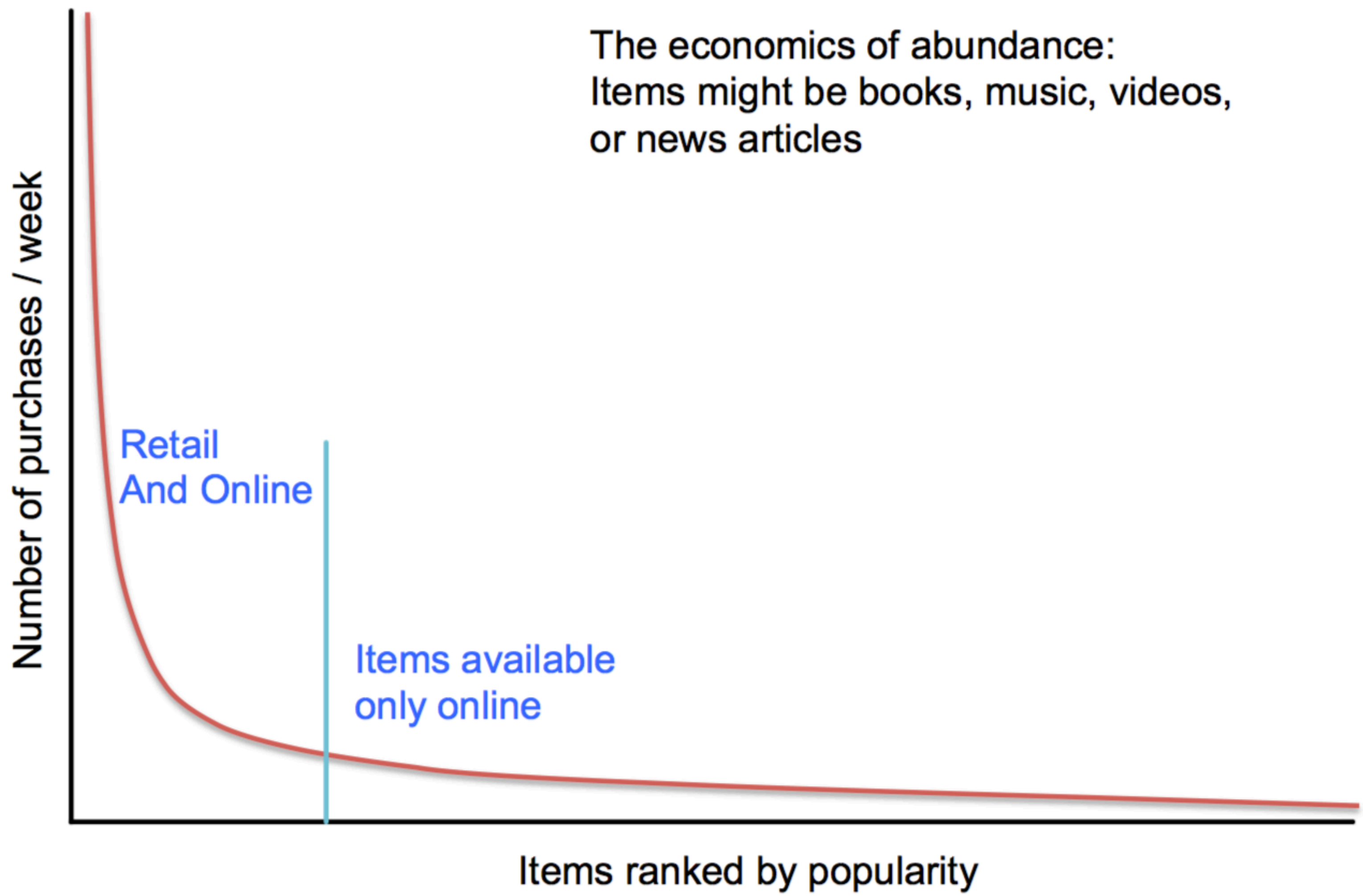


amazon.com
PANDORA
StumbleUpon
del.icio.us
NETFLIX
movielens
helping you find the *right* movies
last.fm
the social music revolution
Google News
YouTube
XBOX LIVE

From scarcity to abundance

- Shelf space is a scarce commodity for traditional retailers
  - Also: TV networks, movie theaters,...
- The web enables near zero-cost dissemination of information about products
  - From scarcity to abundance
  - Gives rise to the “Long Tail” phenomenon (Wired)

Long Tail


The economics of abundance:
Items might be books, music, videos,
or news articles
Number of purchases / week
Retail
And Online
Items available
only online
Items ranked by popularity

Long Tail

- More choice necessitates better filters
- Recommendation engines
- How **Into Thin Air** made **Touching the Void** a bestseller (http://www.wired.com/wired/archive/12.10/tail.html)
- Examples
  - Books, movies, music, new car sales, medical treatments
  - People (friend recommendations on Facebook, LinkedIn, and Twitter)

Types of recommendations

- Editorial and hand curated
  - List of favorites
  - Lists of “essential” items
- Simple aggregates
  - Top 10, Most Popular, Recent Uploads
- **Tailored to individual users**
  - Amazon, Netflix, Pandora ...
  - Our focus here

Recommendation Model

- ***C*** = set of **Customers**
- ***S*** = set of **Items**
- **Utility function *u***: ***C*** × ***S*** →***R***
  - ***R*** = set of ratings
  - ***R*** is a totally ordered set, e.g., **0-5** stars, real number in **[0,1]**

Utility Matrix

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

Problems

- Gathering “known” ratings for matrix
  - How to collect the data in the utility matrix
- Extrapolate unknown ratings from the known ones
  - Mainly interested in high unknown ratings
  - Not typically interested in knowing what you don’t like but what you like
- Evaluating extrapolation methods
  - How to measure success/performance of recommendation methods

Gathering Ratings

- Explicit
  - Ask people to rate items
  - Doesn't scale: only a small fraction of users leave ratings and reviews
- Implicit
  - Learn ratings from user actions
  - E.g., purchase implies high rating
  - What about low ratings?

Extrapolating Utilities

- Key problem: matrix ***U*** is **sparse**
- Most people have not rated most items
- Cold start:
  - New items have no ratings
  - New users have no history
- Three approaches to recommender systems
  1. Content-based
  2. Collaborative
  3. Latent factor based

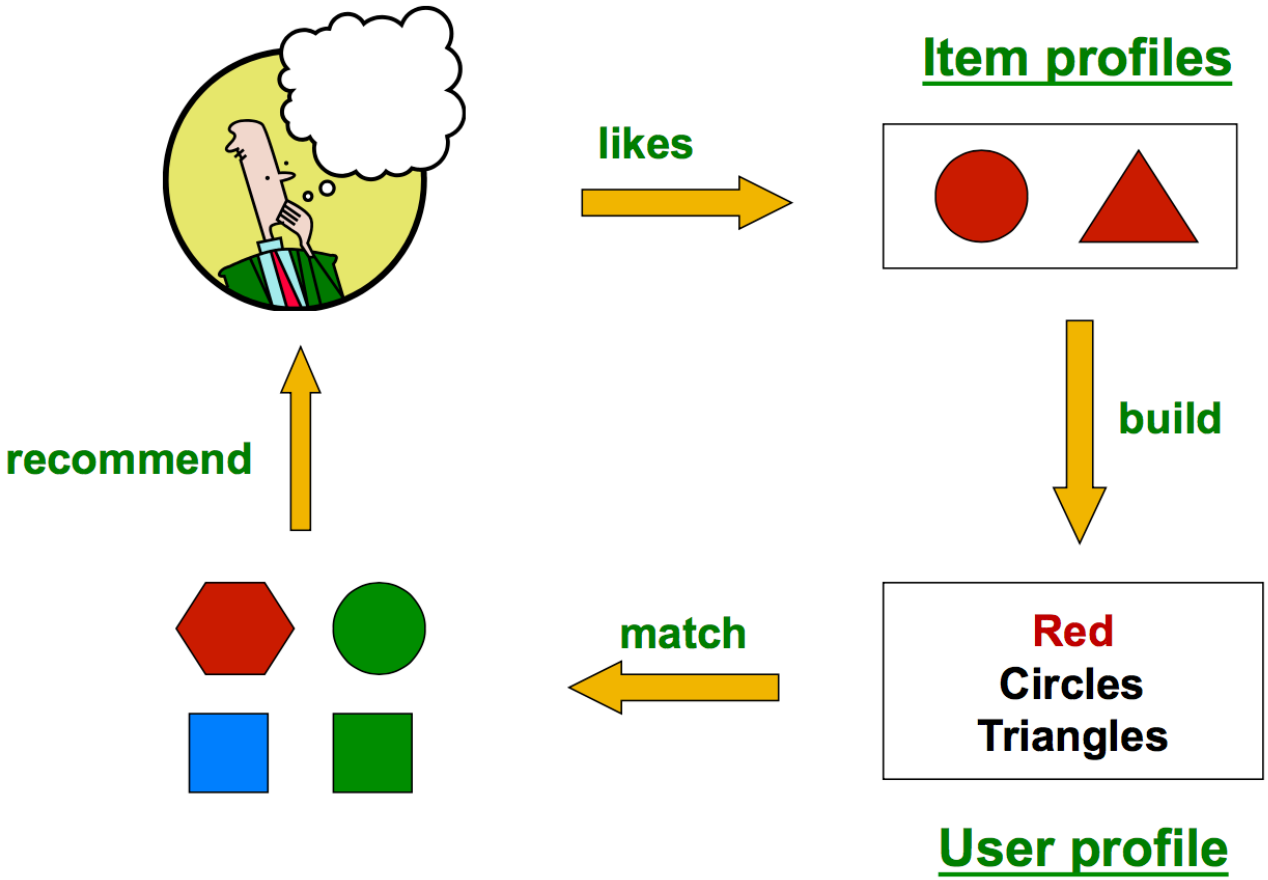
Content-based

Main idea: Recommend items to customer ***x*** similar to previous items rated highly by ***x***

Examples:

- Movies
  - Same actor(s), director, genre, ...
- Websites, blogs, news
  - Articles with "similar" content
- People
  - Recommend people with many common friends

Build item profiles, make recommendations from profile



Item profiles

- For each item, create an **item profile**
- Profile is a set of features
  - **Movies:** author, title, actor, director,...
  - **Images, videos:** metadata and tags
  - **People:** Set of friends
- Think of the item profile as a feature vector
  - One attribute per feature per item (e.g., each actor, director,...)
  - Vector might be boolean or real-valued or mix
- Predict how similar any item is to any other item

User profiles

- User has rated items with profiles $\boldsymbol{i_1,\ldots,i_n}$
- Simple: (weighted) average of rated **item profiles**
- Variant: Normalize weights using average rating of user
- More sophisticated aggregations possible – learn feature weights.

Example 1: Boolean Utility Matrix

- Items are movies, only feature is "Actor"
  - Item profile: vector with 0 or 1 for each Actor
- Suppose user ***x*** has watched 5 movies
  - 2 movies featuring actor A
  - 3 movies featuring actor B

- User profile = mean of item profiles
  - Feature A's weight = 2/5 = 0.4
  - Feature B's weight = 3/5 = 0.6

Example 2: Star Ratings

- Same example, 1-5 star ratings
  - Actor A's movies rated 3 and 5
  - Actor B's movies rated 1, 2 and 4
- Useful step: Normalize ratings by subtracting user's mean rating (mean=3)
  - Actor A's normalized ratings = 0, +2
  - Profile weight = (0 + 2)/2 = 1
  - Actor B's normalized ratings = -2, -1, +1
  - Profile weight = -2/3

Making Predictions

- User profile **x,** Item profile **I**
- Estimate U(**x**,**i**) = cos(θ) = (**x** . **i**)/(|**x**||**i**|)
- Or other similarity measurement

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Pros: Content-based approach

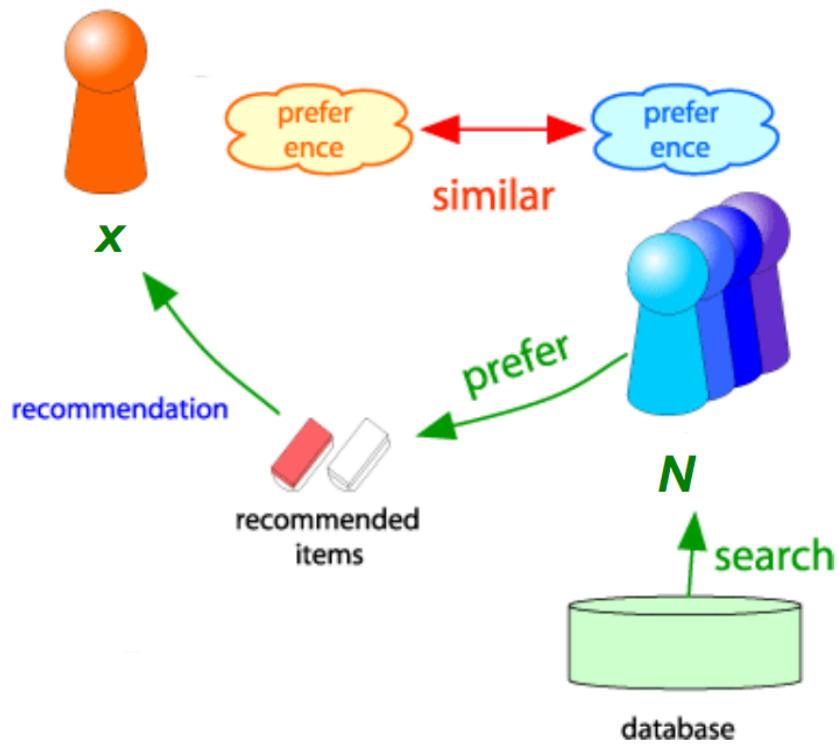
- No need for data on other users
- Able to recommend to users with unique tastes
- Able to recommend new & unpopular items
- No first-rater problem
- Explanations for recommended items
- Content features that caused an item to be recommended

Cons: Content-based approach

- Finding the appropriate features is hard
  - E.g., images, movies, music
- Overspecialization
  - Never recommends items outside user's content profile
  - People might have multiple interests
  - Unable to exploit quality judgments of other users
- Cold-start problem for new users
  - How to build a user profile?

Collaborative Filtering

- 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 users in ***N***



Similar Users

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- Consider users $\boldsymbol{x}$ and $\boldsymbol{y}$ with rating vectors $\boldsymbol{r_x}$ and $\boldsymbol{r_y}$
- We need a similarity metric sim($\boldsymbol{x}$, $\boldsymbol{y}$)
- Capture intuition that sim(A,B) > sim(A,C)

Jaccard Similarity

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- sim(A,B) = $| r_A \cap r_B | / | r_A \cup r_B |$
- sim(A,B) = 1/5; sim(A,C) = 2/4
  - sim(A,B) < sim(A,C)
- Problem: Ignores rating values!

Cosine Similarity

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

```
// sim(A,B)
>>> (5*4)/
(math.sqrt(4**2+5**2+1**2)*math.sqrt(5**2+5
**2+4**2))
0.379
```

- sim(A,B) = cos(r_A, r_B)
- sim(A,B) = 0.38, sim(A,C) = 0.32
  - sim(A,B) < sim(A,C), but not by much
- Problem: treats missing ratings as negative

Centered Cosine – Pearson Correlation Coefficient

- Normalize ratings by subtracting row mean

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

	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

Centered Cosine

	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) = cos(r_A, r_B) = 0.09; sim(A,C) = -0.56
  - sim(A,B) > sim(A,C)
- Captures intuition better
  - Missing ratings treated as "average"
  - Handles "tough raters" and "easy raters"
- Also known as Pearson Correlation

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 who have also rated item i
- Prediction for user x and item i

- Option 1: $r_{xi} = 1/k \sum_{y \in N} r_{yi}$
- Option 2: $r_{xi} = \sum_{y \in N} s_{xy} r_{yi} / \sum_{y \in N} s_{xy}$

where $s_{xy} = \text{sim}(x,y)$

Item-Item Collaborative Filtering

- **So far: User-user collaborative filtering**
- **Another view: Item-item**
  - For item ***i***, find other similar items
  - Estimate rating for item ***i*** based on ratings for similar items
  - Can use same similarity metrics and prediction functions as in user-user model

$$\boxed{r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}}$$

s_{ij}… similarity of items i and j
r_{xj}…rating of user x on item j
$N(i;x)$… set items rated by x similar to i

Item-Item CF (n=2)

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	

- unknown rating

- rating between 1 to 5

Item-Item CF (n=2)

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**

Item-Item CF (n=2)

users

movies	1	2	3	4	**5**	6	7	8	9	10	11	12	**sim(1,m)**
1	1		3		?	5			5		4		**1.00**
2			5	4			4			2	1	3	**-0.18**
3	2	4		1	2		3		4	3	5		**0.41**
4		2	4		5			4			2		**-0.10**
5			4	3	4	2					2	5	**-0.31**
6	1		3		3			2			4		**0.59**

Neighbor selection:
Identify movies similar to movie **1**, **rated by user 5**

Here we use Pearson correlation as similarity:

1) Subtract mean rating m_i from each movie i

$m_1 = (1+3+5+5+4)/5 = \mathbf{3.6}$

row 1: *[-2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4, 0, 0.4, 0]*

2) Compute cosine similarities between rows

Item-Item CF (n=2)

users

movies	1	2	3	4	5	6	7	8	9	10	11	12	sim(1,m)
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	-0.18
3	2	4		1	2		3		4	3	5		0.41
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
6	1		3		3			2			4		0.59

Compute similarity weights:
$s_{13}=0.41$, $s_{16}=0.59$

Item-Item CF (n=2)

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:

$r_{15} = (0.41*2 + 0.59*3) / (0.41+0.59) = 2.6$

Item-item vs. User-user

- In theory, user-user and item-item are dual approaches
- In practice, item-item outperforms user-user in many use cases
- Items are "simpler" than users
  - Items belong to a small set of "genres," users have varied tastes
  - Item Similarity is more meaningful than User Similarity