Information Filtering

SI650 / EECS549 Information Retrieval

October 8, 2025

Slides adapted from David Juergen

SCHOOL OF INFORMATION
UNIVERSITY OF MICHIGAN

Food for thought: Which Platform has better recommendations?

- Amazon also-purchased
- Spotify songs
- Netflix movies (or Hulu, Apple, Amazon, …)
- TikTok’s For You Page (FYP)
- Twitter trending tweets/topics
- *others?*

Lecture Plan

- Filtering vs. Retrieval
- Content-based filtering (adaptive filtering)
- Collaborative filtering (recommender systems)
  - How they work
  - Matrix vs network models
  - Latent factor models
  - Even a little deep learning!

Short vs. Long Term Info Need

- Short-term information need (Ad hoc retrieval)
  - "Temporary need", e.g., info about used cars
  - Information source is relatively static
  - User "pulls" information
  - Application example: library search, Web search
- Long-term information need (Filtering)
  - "Stable need", e.g., new data mining algorithms
  - Information source is dynamic
  - System "pushes" information to user
  - Applications: news filter, recommender systems

Examples of Information Filtering

- News filtering
- Email filtering
- Movie/book recommenders
- Literature recommenders
- And many others …

Content-based Filtering vs. Collaborative Filtering

- Basic filtering question: Will user *U* like item *X*?
- Two different ways of answering it
  - – Look at what U likes ➔ characterize X ➔ content-based filtering
  - – Look at who likes X ➔ characterize U ➔ collaborative filtering
- Can be combined

Collaborative filtering is also called "Recommender Systems"

- Content-based filtering is also called

  1. "Adaptive Information Filtering" in TREC
  2. "Selective Dissemination of Information (SDI) in Library & Information Science

Why Bother?

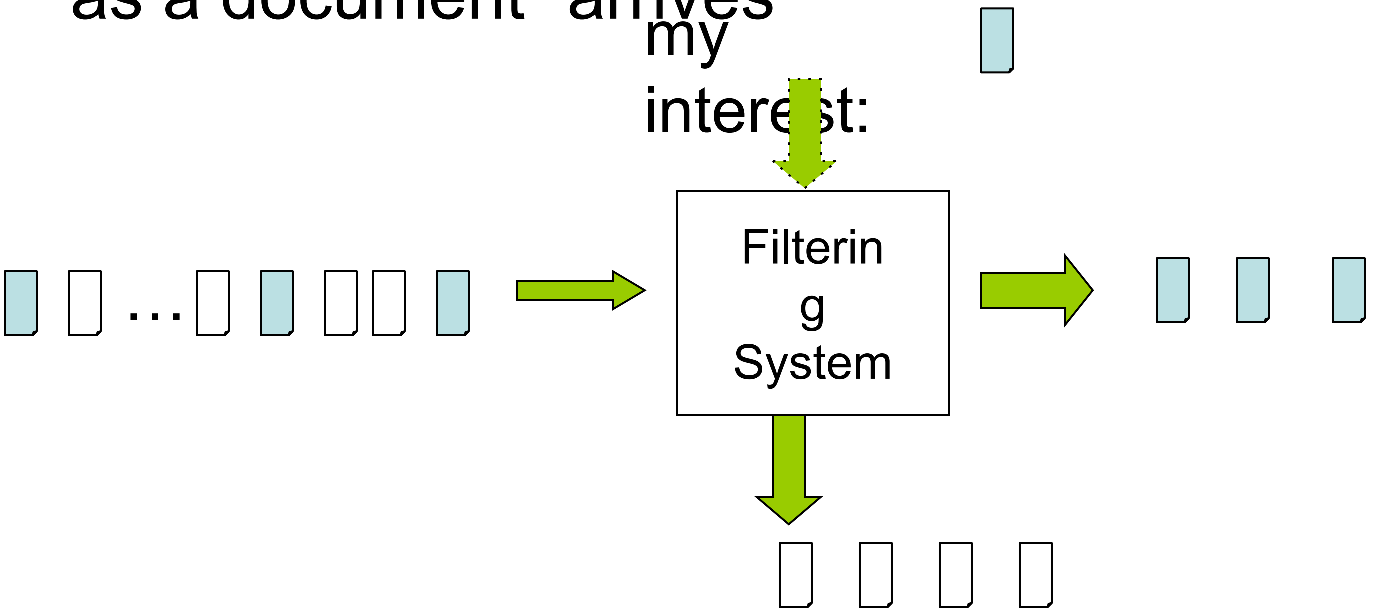
- Why do we need recommendation?
- Why should amazon provide recommendation?
- Why does Netflix spend millions to solicit improvement on their recommendation algorithm?

- Why can’t we rely on a search engine for all the recommendation?

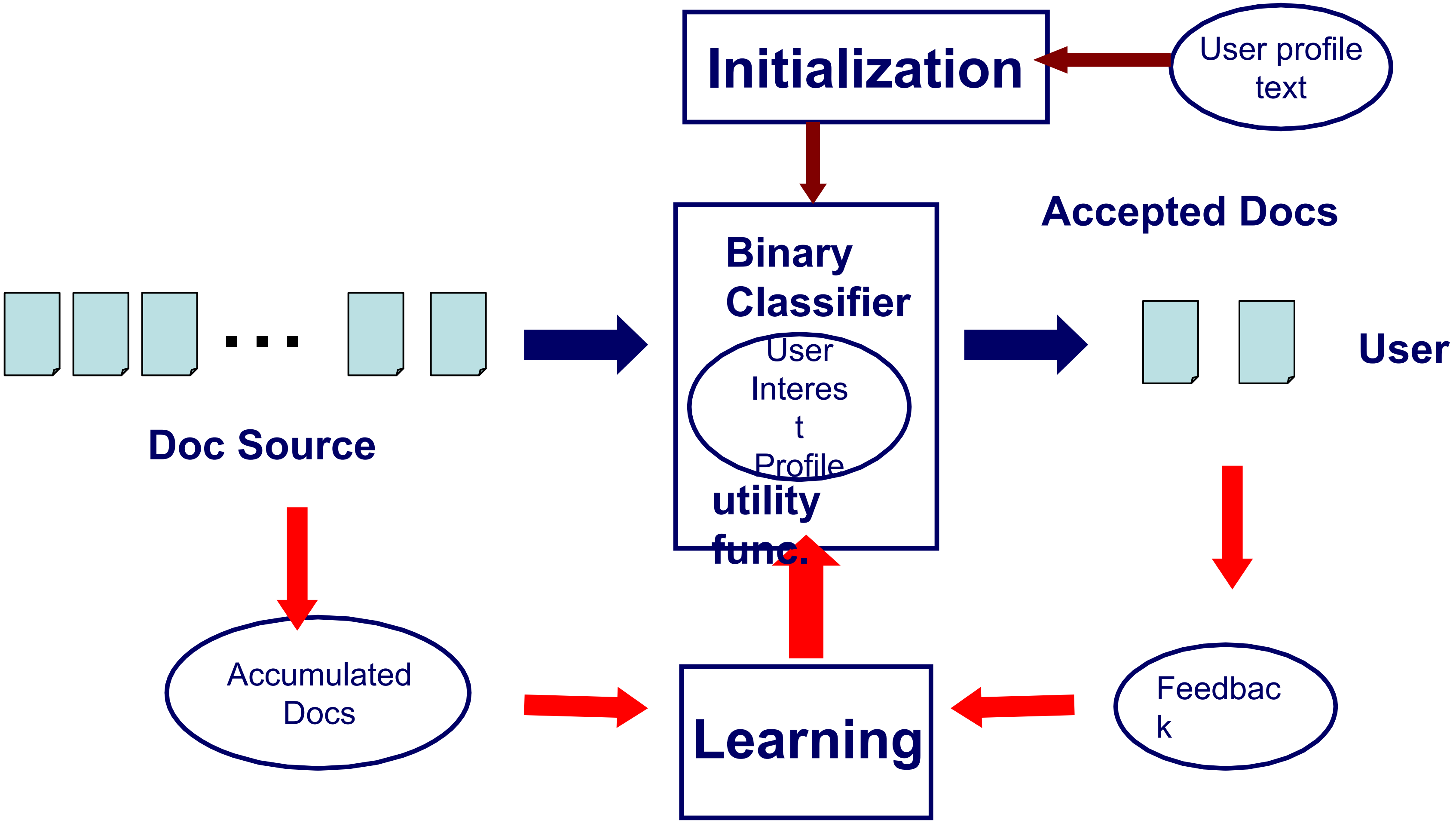
Part I: Adaptive Filtering

Adaptive Information Filtering (AIF)

- Stable & long term interest, dynamic information source
- System must make a delivery decision immediately as a document "arrives"



A Typical AIF System



Initialization

User profile text

Accepted Docs

Binary Classifier

User Interest Profile

utility func.

. . .

User

Doc Source

Accumulated Docs

Learning

Feedback

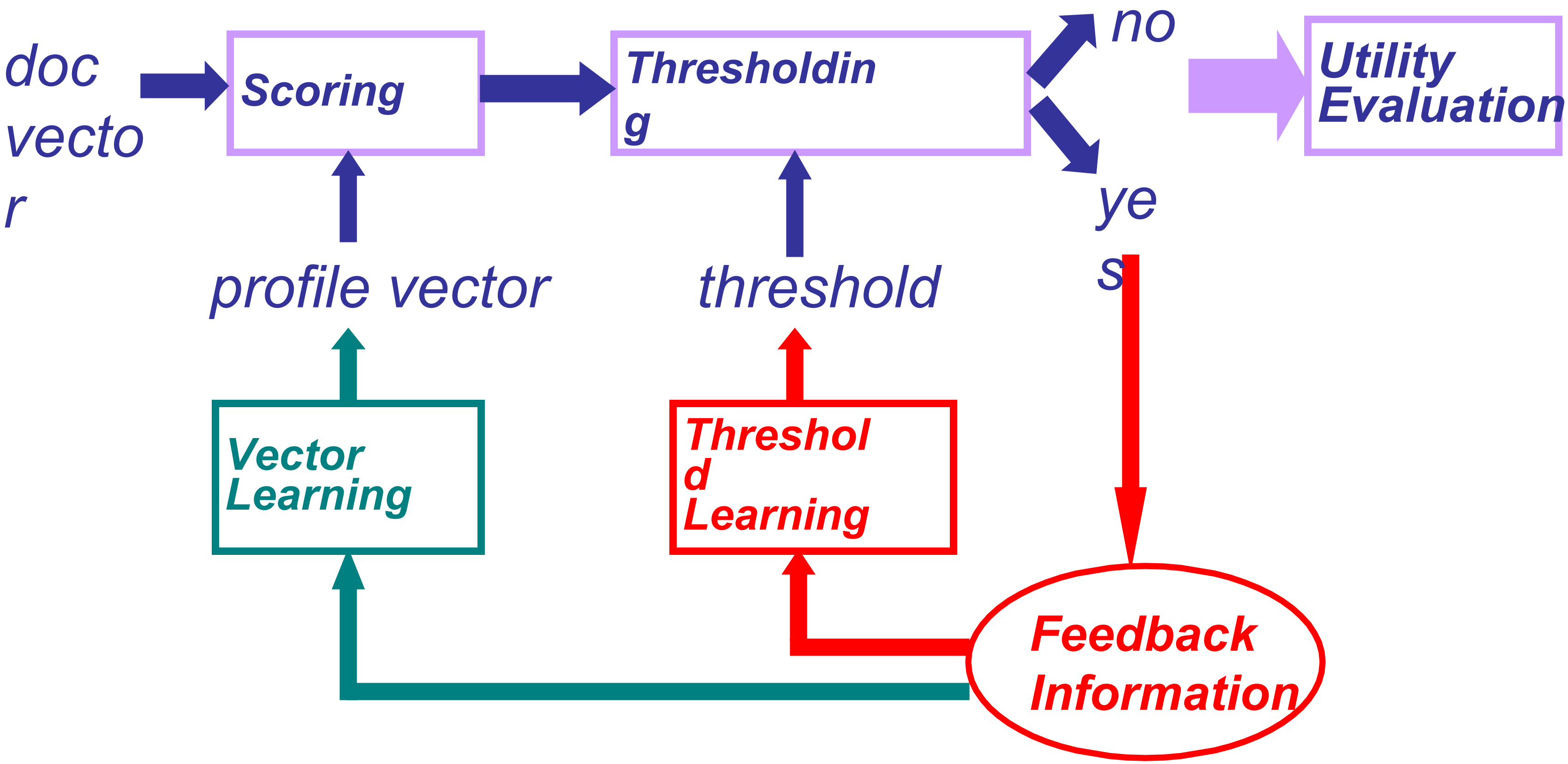
Three Basic Problems in AIF

- Making filtering decision (Binary classifier)
  - – Doc text, profile text → yes/no
- Initialization
  - – Initialize the filter based on only the profile text or very few examples
- Learning from
  - – Limited relevance judgments (only on “yes” docs)
  - – Accumulated documents
- All trying to maximize the utility

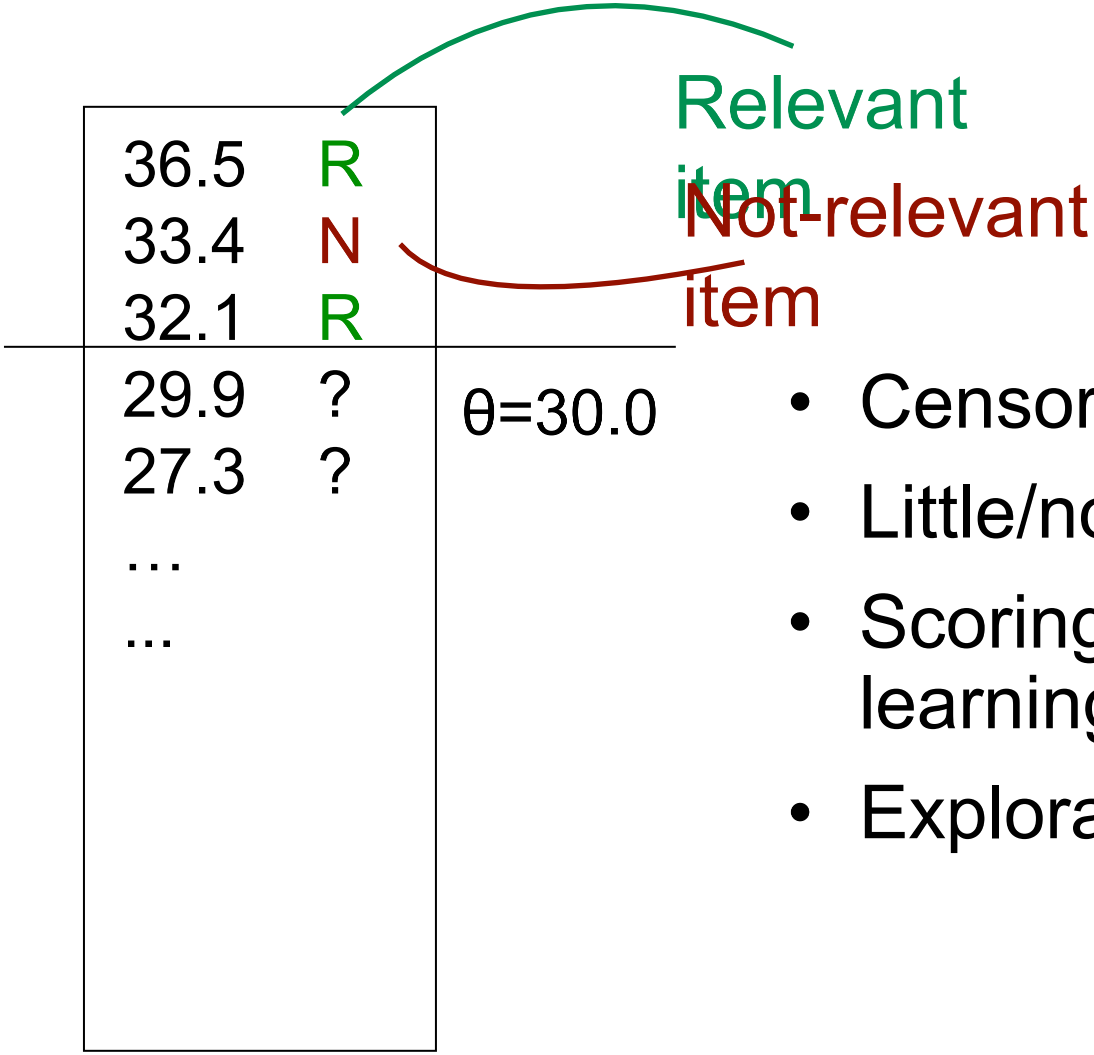
Major Approaches to AIF

- “Extended” retrieval systems
  - – “Reuse” retrieval techniques to score documents
  - – Use a score threshold for filtering decision
  - – Learn to improve scoring with traditional feedback
  - – New approaches to threshold setting and learning
- “Modified” categorization systems
  - – Adapt to binary, unbalanced categorization
  - – New approaches to initialization
  - – Train with “censored” training examples

A General Vector-Space Approach

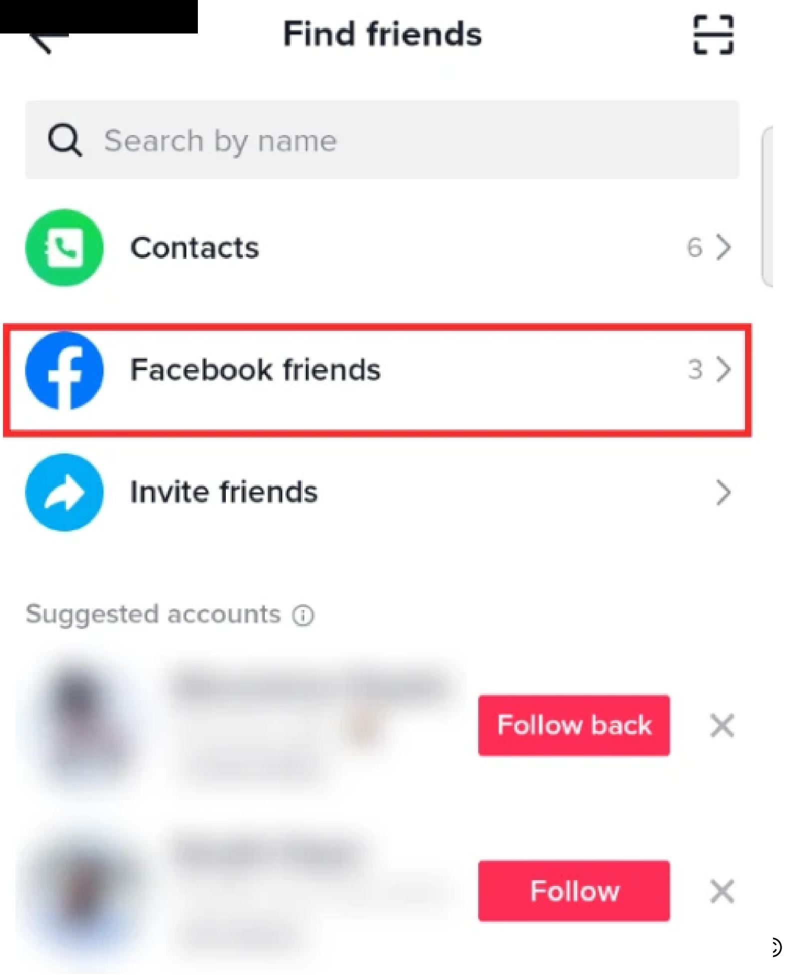


Difficulties in Threshold Learning



- Censored data
- Little/none labeled data
- Scoring bias due to vector learning
- Exploration vs. Exploitation

How might platforms identify user preferences?



- Many platforms request information from a user’s peers to infer the new user’s preferences
- Platforms also *test* on users—explore preferences through specific videos to compare against predictions
- Lots of strategy on identify user feedback

Part II: Collaborative Filtering

What is Collaborative Filtering (CF)?

- Making filtering decisions for an individual user based on the judgments of other users
- Inferring individual's interest/preferences from that of other similar users
- General idea
  - Given a user u, find similar users $\{u_1, \ldots, u_m\}$
  - Predict u's preferences based on the preferences of $u_1, \ldots, u_m$

CF: Intuitions

- User similarity (*Paul Resnick* vs. *Rahul Sami*)
  - If Paul liked the paper, Rahul will like the paper
  - If Paul liked the movie, Rahul will like the movie. Or will he?
  - Suppose Paul and Rahul viewed similar movies in the past six months …
- Item similarity
  - Since 90% of those who liked Star Wars also liked Star Trek, and, you liked Star Wars
  - You may also like Star Trek

The content of items “didn’t

Rating-based vs. Preference-based

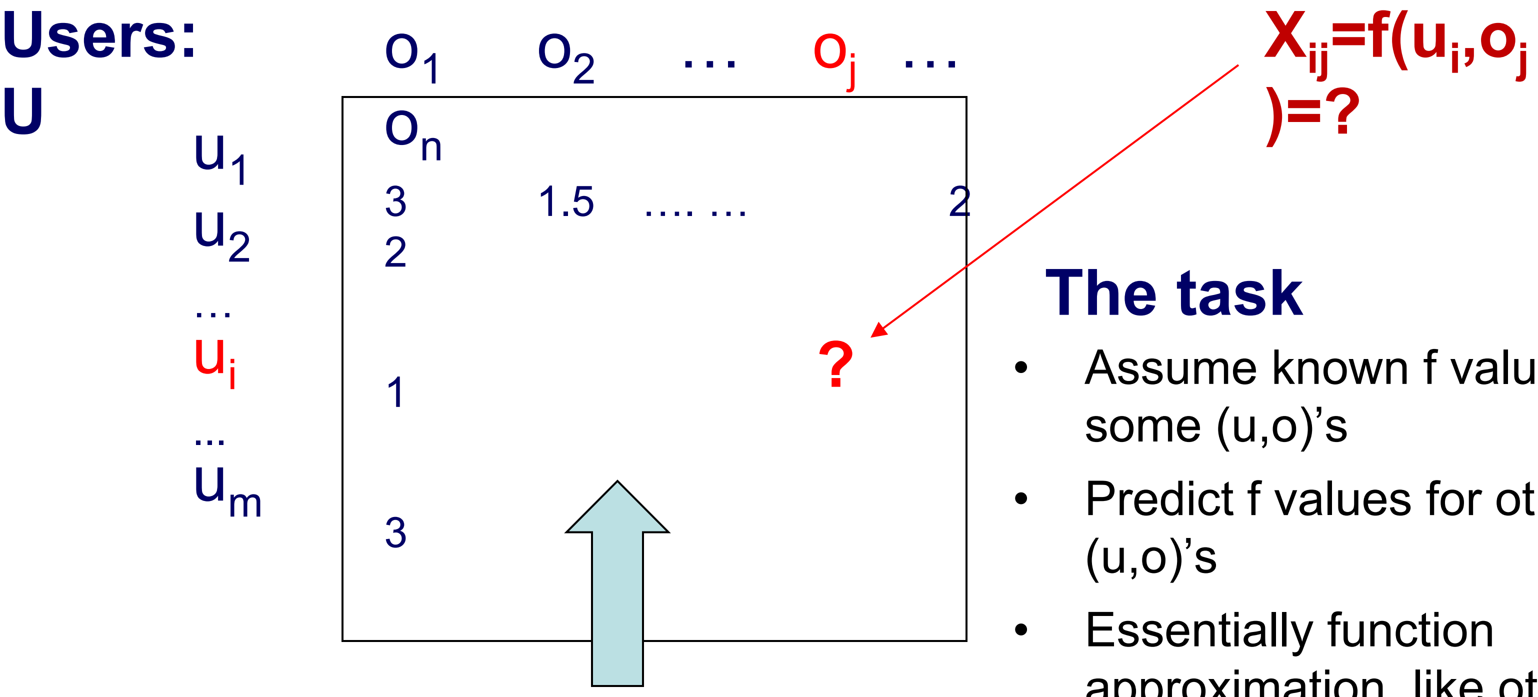
- Rating-based: User's preferences are encoded using numerical ratings on items
  - Complete ordering
  - Absolute values can be meaningful
  - But, values must be normalized to combine
- Preference-based: User's preferences are represented by partial ordering of items (Learning to Rank!)
  - Partial ordering
  - Easier to exploit implicit preferences

Putting Together

Data/Method	**User-User CF**	**Item-Item CF**
Using Rating-Based Data	Finds users with similar rating patterns.	Finds items that receive similar ratings from the same users.
Using Preference-Based Data	Finds users who have interacted with a similar set of items.	Finds items that are frequently interacted with by the same users.

A Formal Framework for Rating

Objects: O



Unknown function
$f: U \times O \rightarrow R$

The task

- Assume known f values for some (u,o)'s
- Predict f values for other (u,o)'s
- Essentially function approximation, like other learning problems

Where are the intuitions?

- Similar users have similar preferences
  - If u ≈ u’, then for all o’s, f(u,o) ≈ f(u’,o)
- Similar objects have similar user preferences
  - If o ≈ o’, then for all u’s, f(u,o) ≈ f(u,o’)
- More broadly,
  - If u ≈ u’ and o ≈ o’, then f(u,o) ≈ f(u’,o’)
  - “Local smoothness” makes it possible to predict unknown values by interpolation or extrapolation
- What does “local” mean?

Two Groups of Approaches

- Memory-based approaches
  - f(u, o) = g(u)(o) ≈ g(u')(o) if u ≈ u'
    (g =preference function)
  - Find "neighbors" of u and combine g(u')(o)'s
- Model-based approaches (not covered)
  - Assume structures/model: object clusters, user clusters, f' defined on clusters
  - $f(u, o) = f'(c_u, c_o)$
  - Estimation & Probabilistic inference

Memory-based Approaches

(Resnick et al. 94)

- General ideas:
  - x_{ij}: rating of object j by user i
  - n_i: average rating of all objects by user i
  - Normalized ratings: $v_{ij} = x_{ij} - n_i$
  - Memory-based prediction

$$v_{dj} = k \sum_{i=1}^{m} w(a,i) v_{ij} \quad k = 1 / \sum_{i=1}^{m} w(a,i) \quad \Rightarrow \quad x_{aj} = v_{aj} + n_a$$

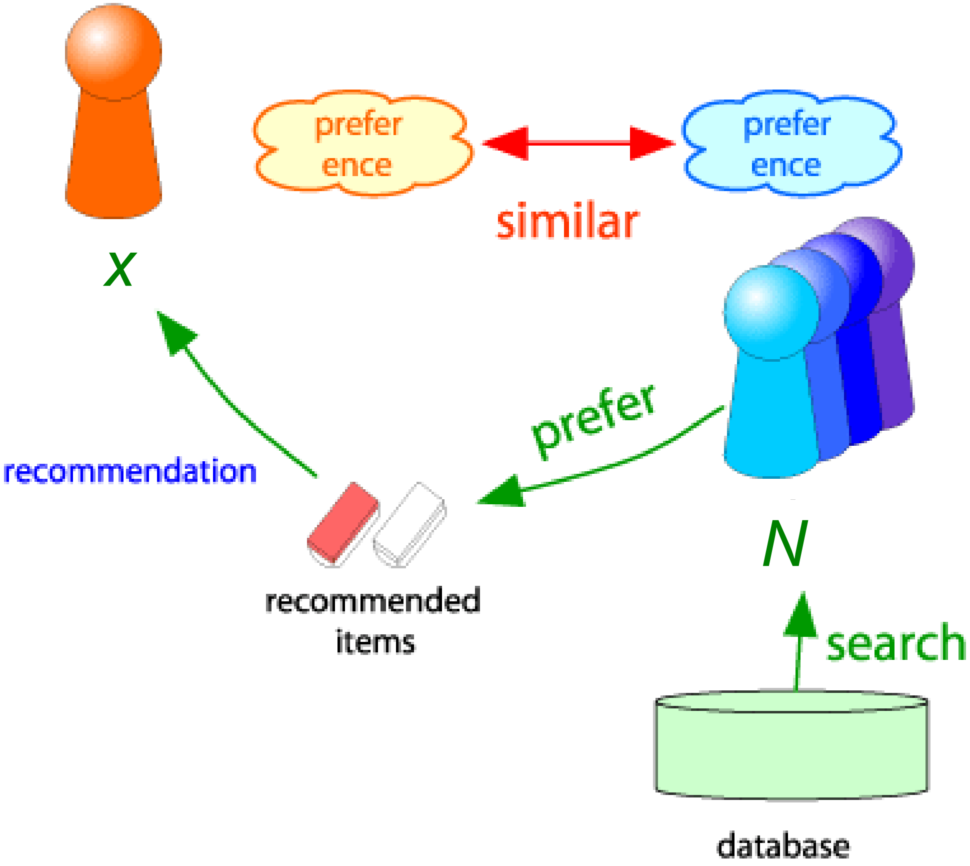
- Specific approaches differ in *w(a, i)* -- the distance/similarity between user *a* and *i*

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***



J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

$$sim(x,y) = \frac{\sum_i r_{xi} \cdot r_{yi}}{\sqrt{\sum_i r_{xi}^2} \cdot \sqrt{\sum_i r_{yi}^2}}$$

Similarity Metric

	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

Finding similar users

- Let r_x be the vector of user x's ratings
- Jaccard similarity measure
  - Problem: Ignores the values of the rating
- Cosine similarity measure
  - $sim(x,y) = cos(r_x, r_y) = \frac{r_x \cdot r_y}{\|r_x\| \cdot \|r_y\|}$
  - Problem: Treats missing values as negative
- Pearson correlation coefficient
  - S_{xy} = items rated by both users x and y
  - $sim(x,y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})(r_{ys} - \overline{r_y})}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \overline{r_y})^2}}$

r_x = [*, _, _, *, ***]

r_y = [*, _, **, **, _]

r_x, r_y as sets:

r_x = {1, 4, 5}

r_y = {1, 3, 4}

r_x, r_y as points:

r_x = {1, 0, 0, 1, 3}

r_y = {1, 0, 2, 2, 0}

$\mathbf{r_x}$, $\mathbf{r_y}$ … avg. rating of **x**, **y**

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Rating Predictions

- From similarity metrics to recommendations:
  - 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 rated item ***i***
  - Prediction for item ***s*** by user ***x***
    - $r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$
    - $r_{xi} = \frac{\sum_{y in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$
    - Other options?
  - *Many* other tricks possible

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

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

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

Item-Item CF (|N|=2)

users

movies

12	11	10	9	8	7	6	5	4	3	2	1	
	4		5			5			3		1	1
3	1	2			4			4	5			2
	5	3	4		3		2	1		4	2	3
	2			4			5		4	2		4
5	2					2	4	3	4			5
	4			2			3		3		1	6

☐ - unknown rating

🟨 - rating between 1 to 5

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Item-Item CF (|N|=2)

users

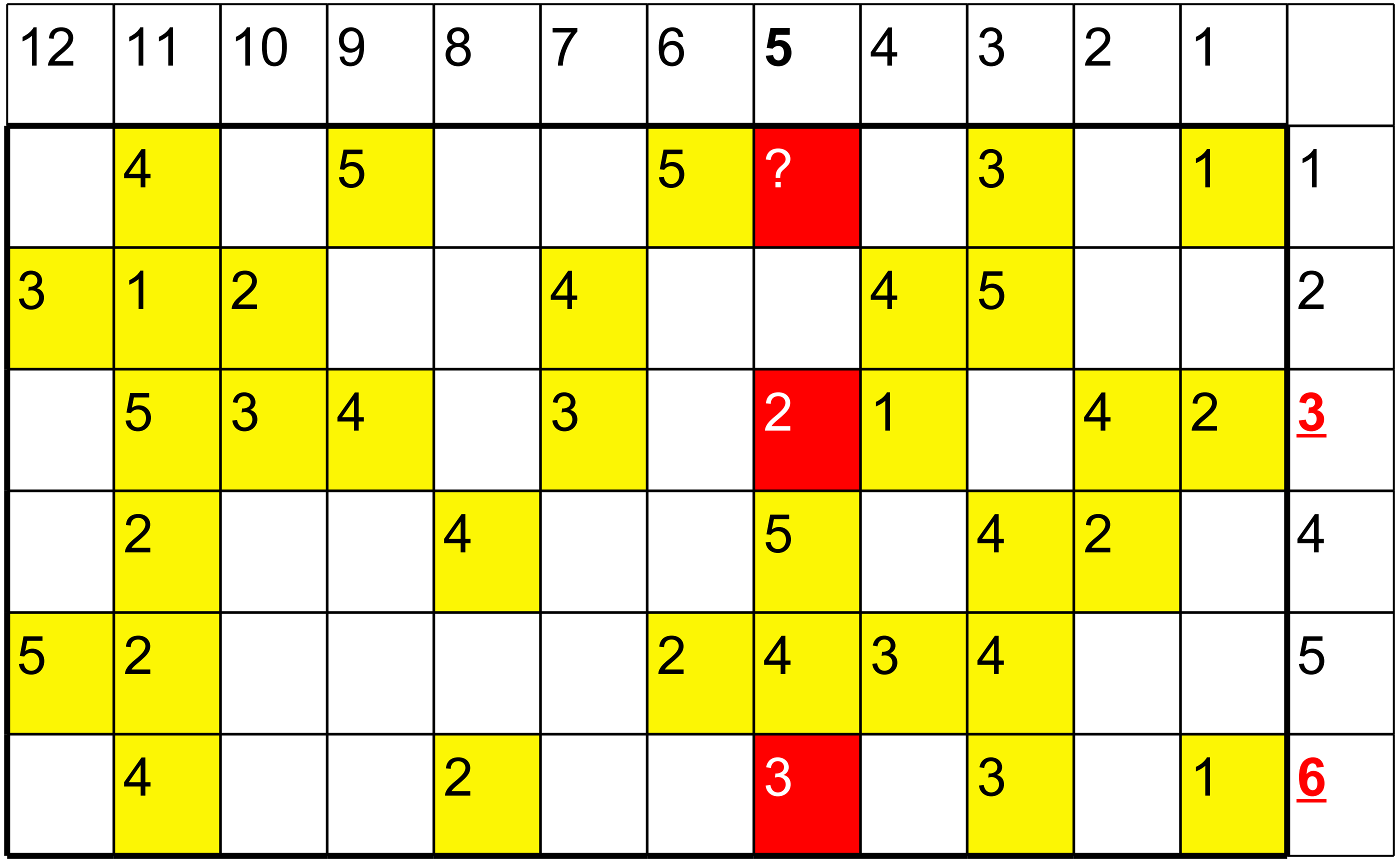
movies

12	11	10	9	8	7	6	**5**	4	3	2	1	
	4		5			5	?		3		1	1
3	1	2			4			4	5			2
	5	3	4		3		2	1		4	2	3
	2			4			5		4	2		4
5	2					2	4	3	4			5
	4			2			3		3		1	6

🟥 - estimate rating of movie **1** by user **5**

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Item-Item CF (|N|=2)



users

movies

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

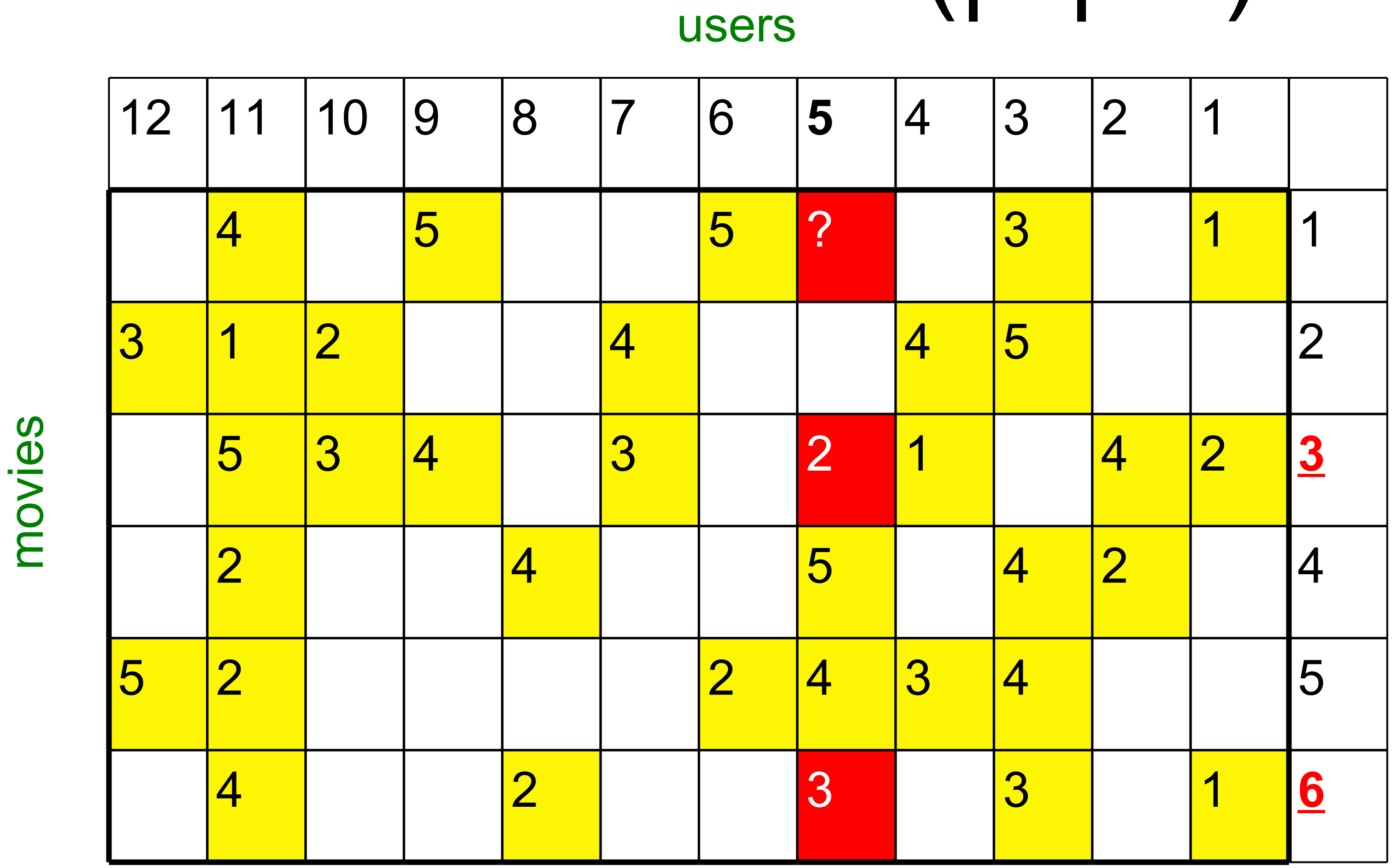
$m_1 = (1+3+5+5+4)/5 = \mathbf{3.6}$

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

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Item-Item CF (|N|=2)



12	11	10	9	8	7	6	**5**	4	3	2	1	
	4		5			5	?		3		1	1
3	1	2			4			4	5			2
	5	3	4		3		2	1		4	2	3
	2			4			5		4	2		4
5	2					2	4	3	4			5
	4			2			3		3		1	6

sim(1,m)

1.00

-0.18

0.41

-0.10

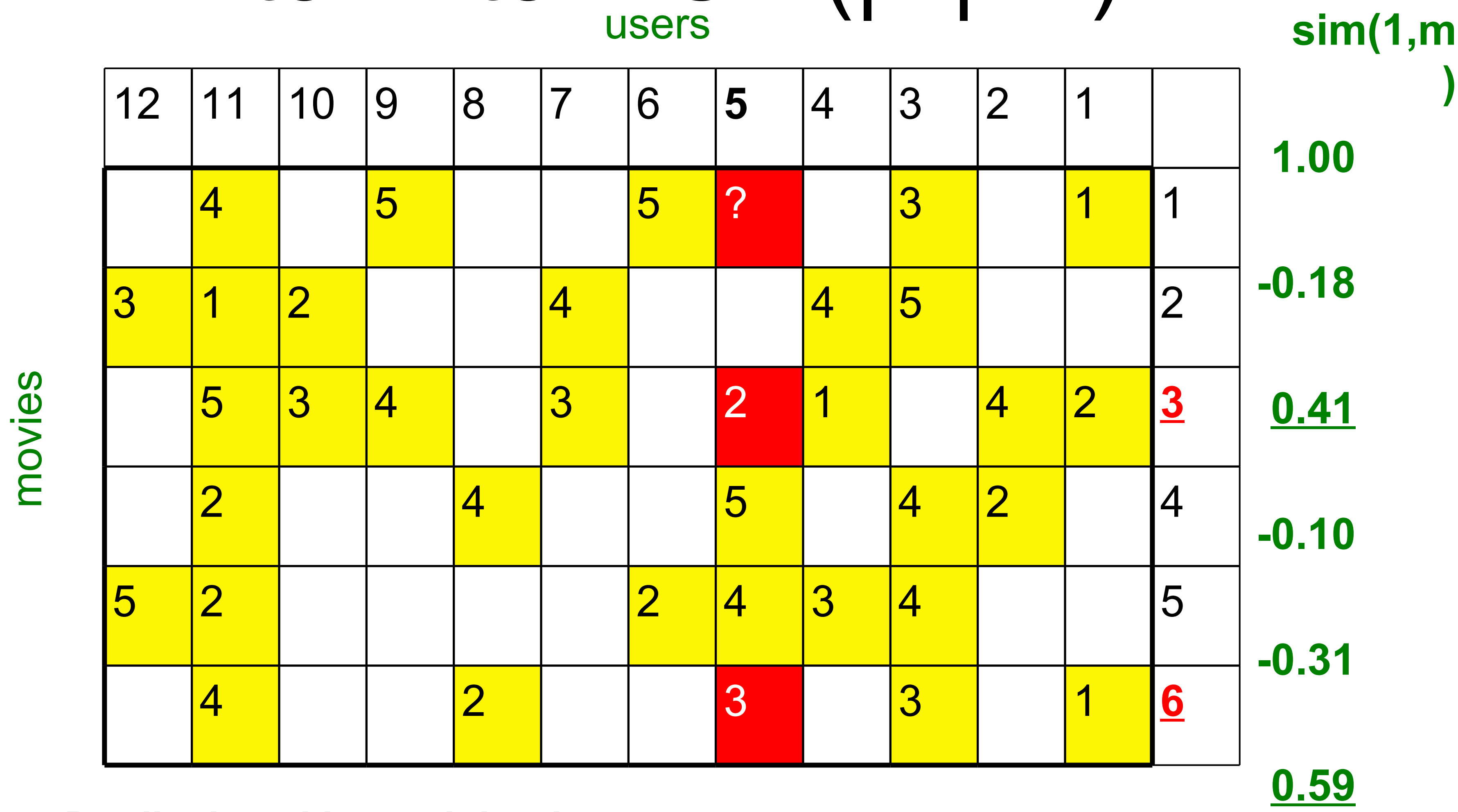
-0.31

0.59

Compute similarity weights:

$s_{1,3}$=0.41, $s_{1,6}$=0.59

Item-Item CF (|N|=2)



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

Predict by taking weighted average:

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

$$r_{ix} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{jx}}{\sum s_{ij}}$$

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

CF: Common Practice

Before:

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

- Define similarity s_{ij} of items i and j
- Select k nearest neighbors $N(i; x)$
  - Items most similar to i, that were rated by x
- Estimate rating r_{xi} as the weighted average:

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

baseline estimate for r_{xi}

$$b_{xi} = \mu + b_x + b_i$$

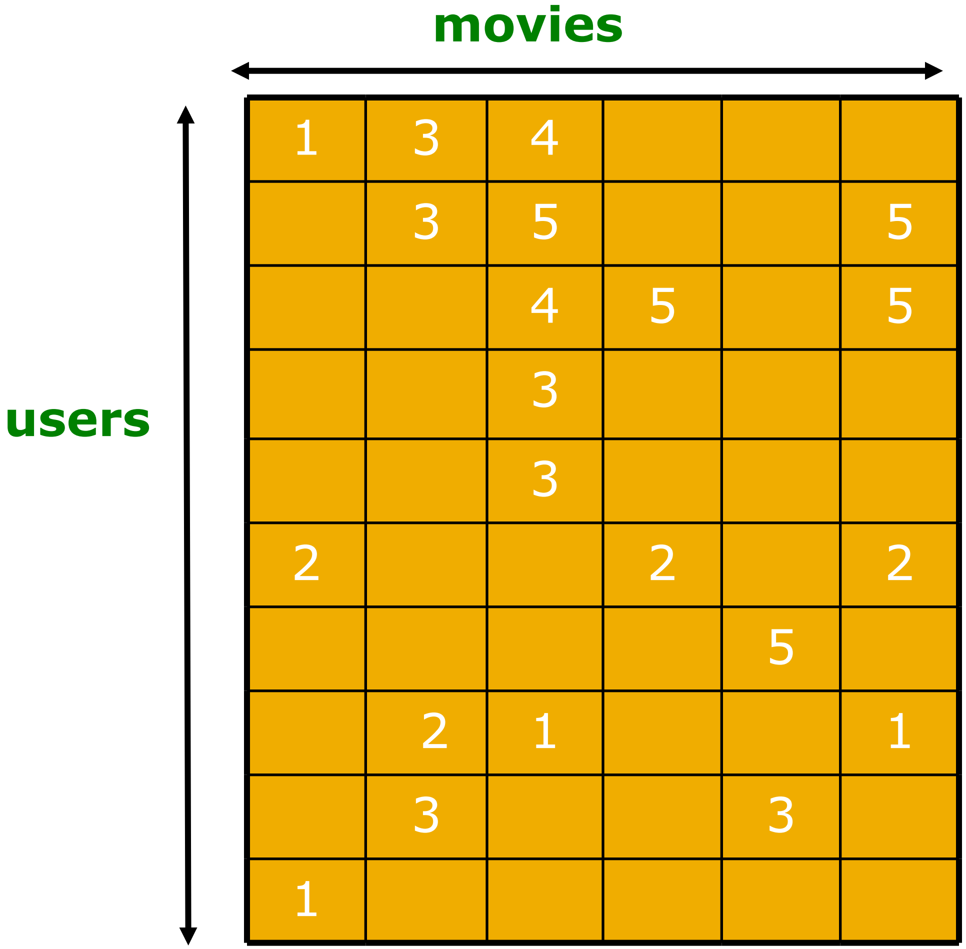
- μ = overall mean movie rating
- b_x = rating deviation of user ***x***
  = (*avg. rating of user **x***) – μ
- b_i = rating deviation of movie ***i***

Item-Item vs. User-User

	Marvel	Harry Pot.	Matrix	Pirates
Alice	1		0.8	
Bob		0.5		0.3
Carol	0.9		1	0.8
Rob			1	0.4

- **In practice, it has been observed that item-item often works better than user-user. Why?**
- Items are simpler, users have multiple tastes

Evaluation



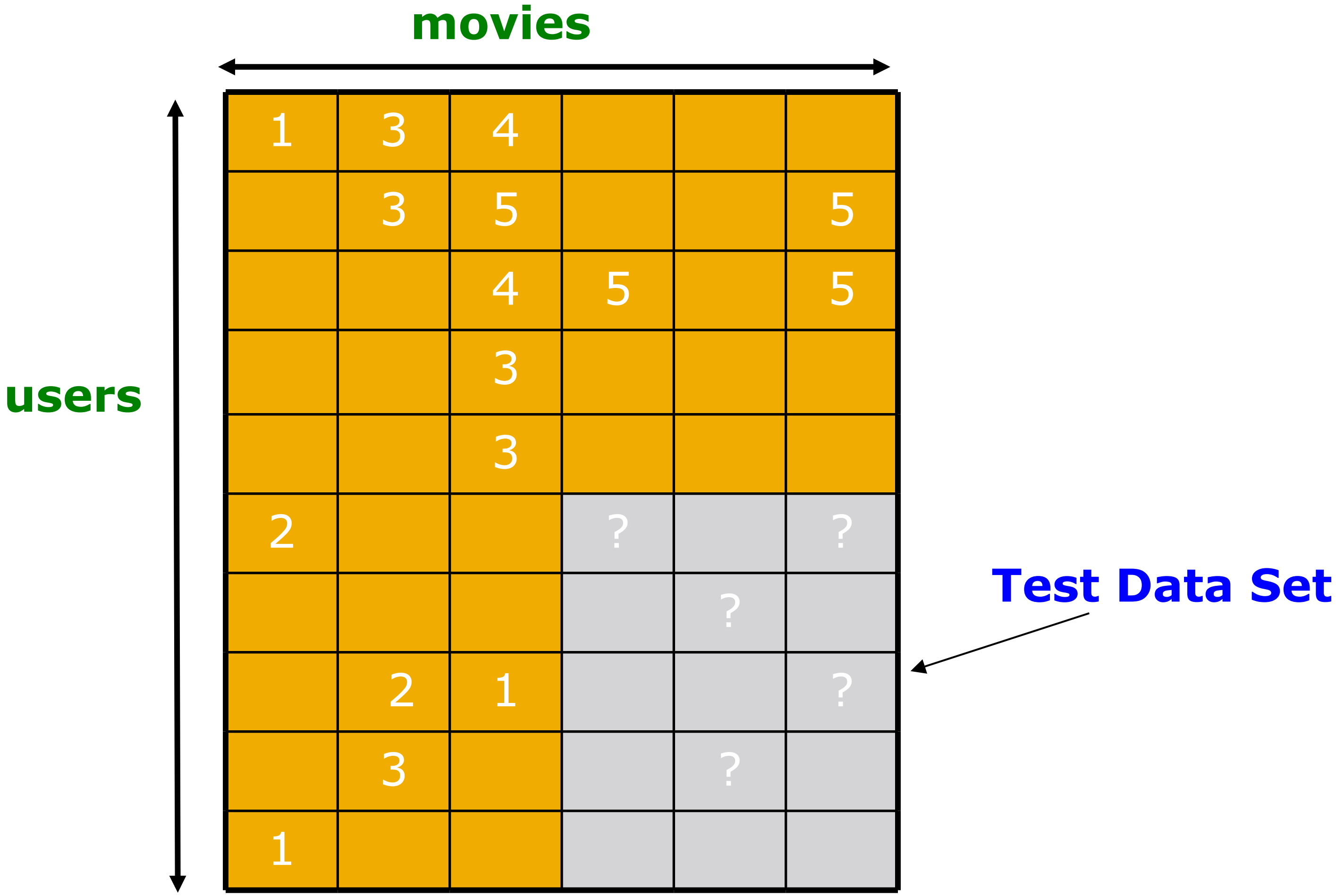
movies

users

1	3	4			
	3	5			5
		4	5		5
		3			
		3			
2			2		2
				5	
	2	1			1
	3			3	
1					

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Evaluation



1	3	4			
	3	5			5
		4	5		5
		3			
		3			
2			?		?
				?	
	2	1			?
	3			?	
1					

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Evaluating Predictions

- **Compare predictions with known ratings**
  - Root-mean-square error (RMSE)
    - where r_{xi} is predicted and r^*_{xi} is true rating
  - Prediction@10
    - % of predictions in top 10
  - Rank Correlation
    - Spearman's correlation between system's and user's ranking of items
- **Another approach**: 0/1 model
  - Coverage: number of items/users for which the system can make a prediction
  - Prediction: Accuracy of ratings
  - Receiver Operating Character (ROC): tradeoff curve between true positives and false negatives

Pros/Cons of Collaborative Filtering

- **+ Works for any kind of item**
  - No feature selection needed
- **- Cold Start:**
  - Need enough users in the system to find a match
- **- Sparsity:**
  - The user/ratings matrix is sparse
  - Hard to find users that have rated the same items
- **- First rater:**
  - Cannot recommend an item that has not been previously rated
  - New items, Esoteric items
- **- Popularity bias:**
  - Cannot recommend items to someone with unique taste
  - Tends to recommend popular items

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Hybrid Methods

- **Implement two or more different recommenders and combine predictions**
  - Perhaps using a linear model

- **Add content-based methods to collaborative filtering**
  - Item profiles for new item problem
  - Demographics to deal with new user problem

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Tip: Add Data

- **Leverage all the data**
  - Don't try to reduce data size in an effort to make fancy algorithms work
  - Simple methods on large data do best
- **Add more data**
  - e.g., add IMDB data on genres
- **More data beats better algorithms**

http://anand.typepad.com/datawocky/2008/03/more-data-usual.html

Latent Factor Modeling

and the Netflix Prize

The Netflix Prize

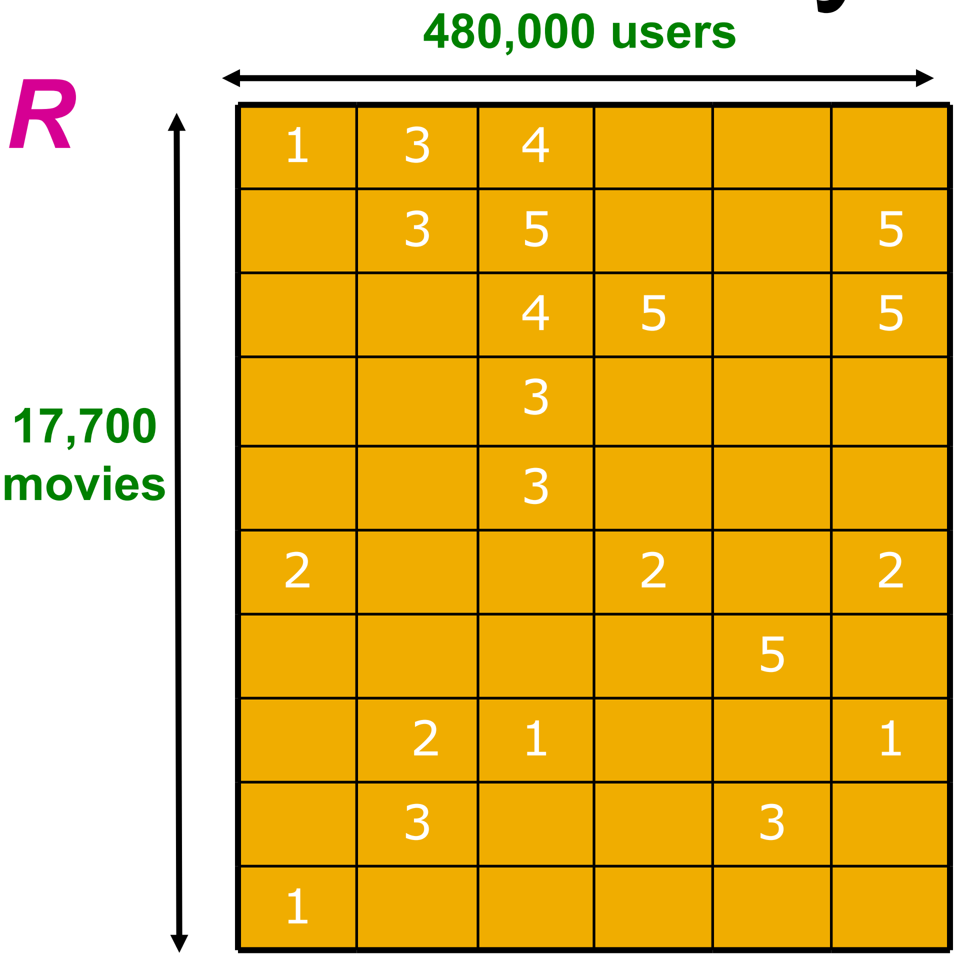
- Training data
  - 100M ratings, 480K users, 17,770 movies
  - 6 years of data: 2000-2005
- Test data
  - Last few ratings of each user (2.8M)
  - Evaluation criterion: Root Mean Squared Error (RMSE)

$$= \frac{1}{|R|}\sqrt{\sum_{(i,x) inR} (\hat{r}_{ix} - r_{ix})^2}$$

- Competition: 2,700+ teams
- $1 million price for 10% improvement over Netflix’s method

The Netflix Utility Matrix *R*

Matrix R



480,000 users

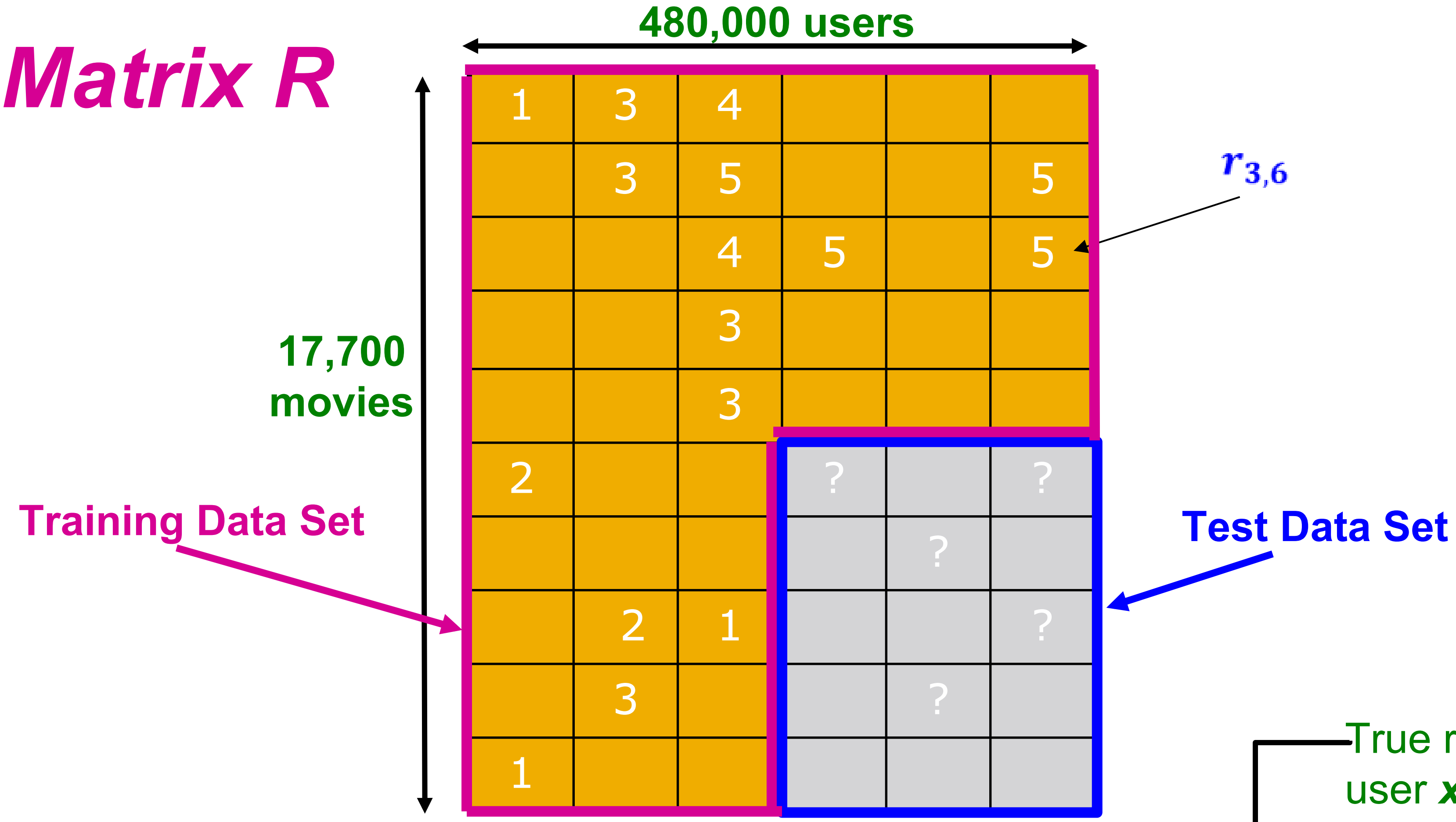
17,700 movies

1	3	4			
	3	5			5
		4	5		5
		3			
		3			
2			2		2
				5	
	2	1			1
	3			3	
1					

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Utility Matrix *R*: Evaluation

Matrix R



1	3	4			
	3	5			5
		4	5		5
		3			
		3			
2			?		?
				?	
	2	1			?
	3			?	
1					

True rating of user ***x*** on item ***i***

$$\textbf{RMSE} = \frac{1}{|R|}\sqrt{\sum_{(i,x)\in R}(\hat{r}_{xi} - r_{xi})^2}$$

Predicted rating

Global average: 1.1296

User average: 1.0651

Movie average: 1.0533

Netflix: 0.9514

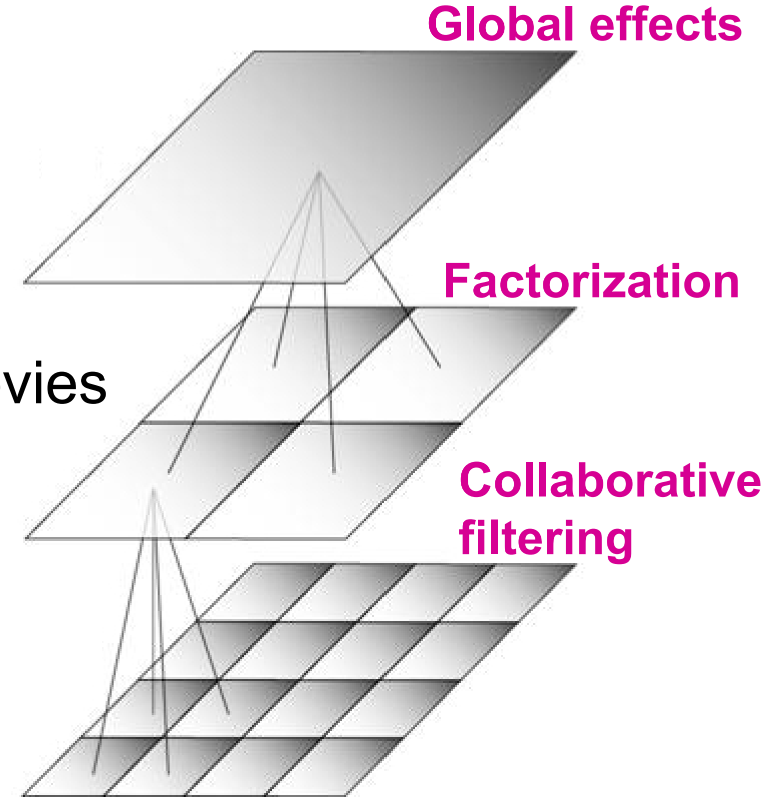
Basic Collaborative filtering: 0.94

CF+Biases+learned weights: 0.91

Grand Prize: 0.8563

BellKor Recommender System

- **The winner of the Netflix Challenge!**
- **Multi-scale modeling of the data:**
  Combine top level, “regional” modeling of the data, with a refined, local view:
  - **Global:**
    - Overall deviations of users/movies
  - **Factorization:**
    - Addressing “regional” effects
  - **Collaborative filtering:**
    - Extract local patterns



Modeling Local & Global Effects

- In practice we get better estimates if we model *deviations*:

$$\hat{r}_{xi} = b_{xi} + \frac{\sum_{j \in N(i;x)} s_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i;x)} s_{ij}}$$

baseline estimate for r_{xi}

$$b_{xi} = \mu + b_x + b_i$$

μ = overall mean rating
b_x = rating deviation of user ***x***
= (*avg. rating of* user ***x***) – ***μ***
b_i = (*avg. rating of* movie ***i***) – ***μ***

Problems/Issues:
1) Similarity measures are “arbitrary”
2) Pairwise similarities neglect interdependencies among users
3) Taking a weighted average can be restricting

Solution: Instead of s_{ij} use w_{ij} that we estimate directly from data

Idea: Interpolation Weights w_{ij}

- Use a weighted sum instead of a weighted average

$$\hat{r}_{xi} = b_{xi} + \sum_{j \in N(i;x)} w_{ij}(r_{xj} - b_{xj})$$

- A few notes:
  - N(i;x) is the set of movies rated by user x that are similar to move i
  - w_{ij} is the interpolation weight (some real number)
    - we allow $\sum_{j \in N(i;x)} w_{i,j} \neq 1$
  - w_{ij} models interactions between pairs of movies (it does not depend on user x)

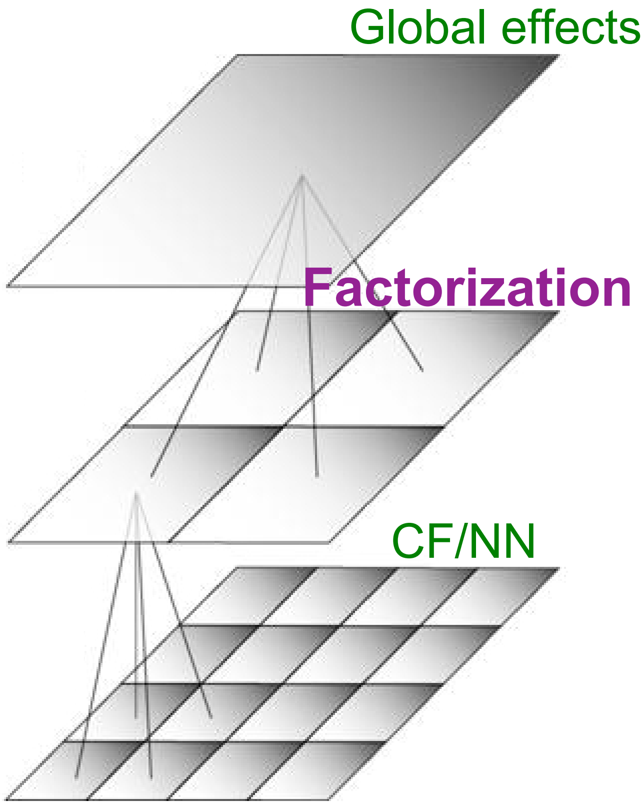
Idea: Interpolation Weights w_{ij}

- $\widehat{r_{xi}} = b_{xi} + \sum_{j \in N(i,x)} w_{ij}\left(r_{xj} - b_{xj}\right)$
- **How to set w_{ij}?**
  - Remember, error metric is: $\frac{1}{|R|}\sqrt{\sum_{(i,x)\in R}(\hat{r}_{xi} - r_{xi})^2}$

    or equivalently **SSE:** $\sum_{(i,x)\in R}(\hat{r}_{xi} - r_{xi})^2$

Interpolation weights

- So far: $\hat{r}_{xi} = b_{xi} + \sum_{j \in N(i;x)} w_{ij}(r_{xj} - b_{xj})$
  - Weights w_{ij} are derived based on their role —no use of an arbitrary similarity measure ($w_{ij} \neq s_{ij}$)
  - We are explicitly accounting for interrelationships among the neighboring movies
- Next: **Latent Factor Models**
  - Extract *regional* correlations



Latent Factor Models

SVD: $A = U \Sigma V^T$

- “SVD” on Netflix data: $\mathbf{R} \approx Q \cdot P^T$

users

items

	4		5			5			3		1
3	1	2			4			4	5		
	5	3	4		3		2	1		4	2
	2			4			5		4	2	
5	2					2	4	3	4		
	4			2			3		3		1

R

≈

factors

items

.2	-.4	.1
.5	.6	-.5
.5	.3	-.2
.3	2.1	1.1
-2	2.1	-.7
.3	.7	-1

Q

users

factors

-.9	2.4	1.4	.3	-.4	.8	-.5	-2	.5	.3	-.2	1.1
1.3	-.1	1.2	-.7	2.9	1.4	-1	.3	1.4	.5	.7	-.8
.1	-.6	.7	.8	.4	-.3	.9	2.4	1.7	.6	-.4	2.1

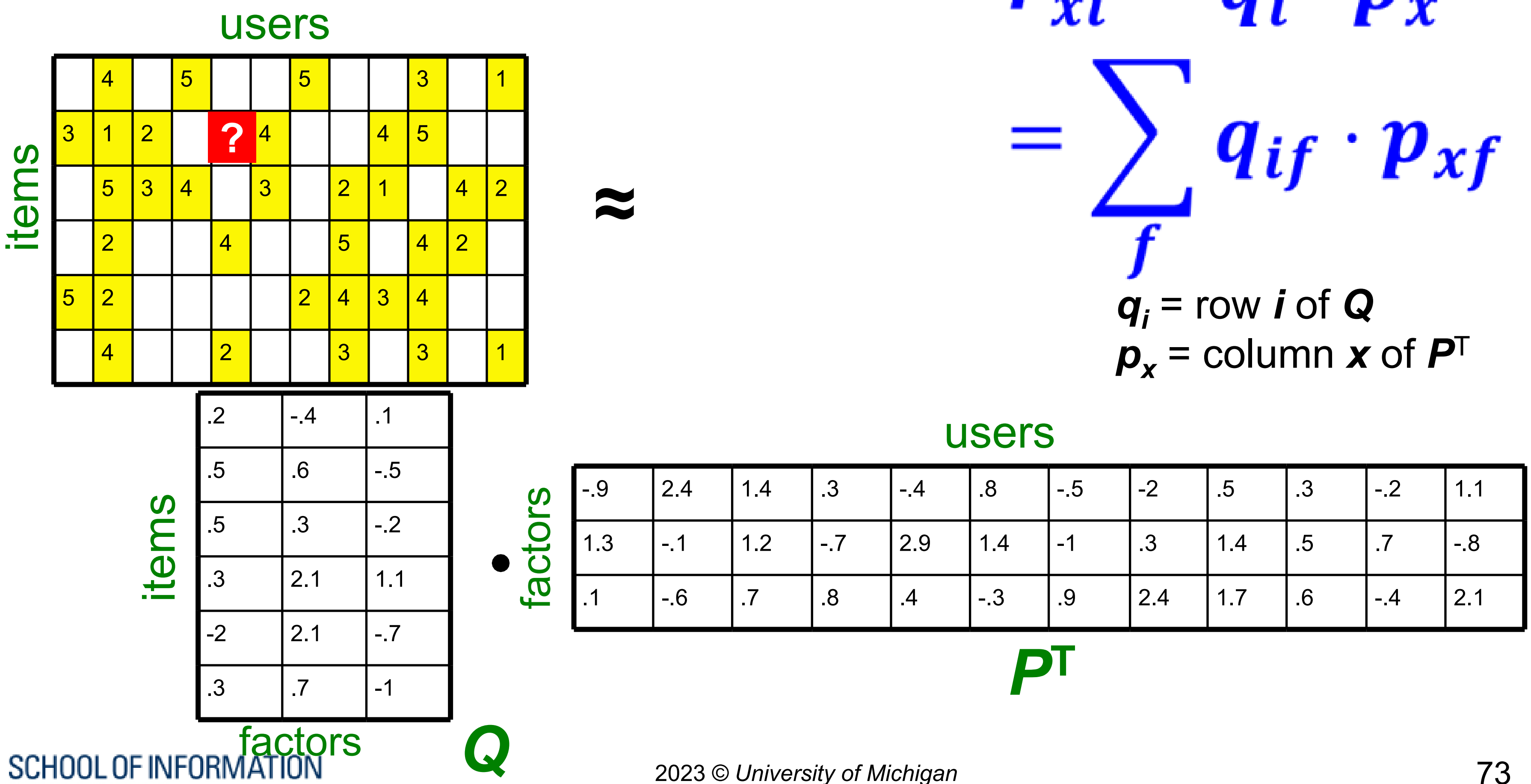
P^T

Ratings as Products of Factors

- **How to estimate the missing rating of user x for item i?**

$$\hat{r}_{xi} = q_i \cdot p_x$$
$$= \sum_f q_{if} \cdot p_{xf}$$

q_i = row i of Q
p_x = column x of P^T



	4		5			5			3		1
3	1	2		?	4			4	5		
	5	3	4		3		2	1		4	2
	2			4			5		4	2	
5	2					2	4	3	4		
	4			2			3		3		1

.2	-.4	.1
.5	.6	-.5
.5	.3	-.2
.3	2.1	1.1
-2	2.1	-.7
.3	.7	-1

-.9	2.4	1.4	.3	-.4	.8	-.5	-2	.5	.3	-.2	1.1
1.3	-.1	1.2	-.7	2.9	1.4	-1	.3	1.4	.5	.7	-.8
.1	-.6	.7	.8	.4	-.3	.9	2.4	1.7	.6	-.4	2.1

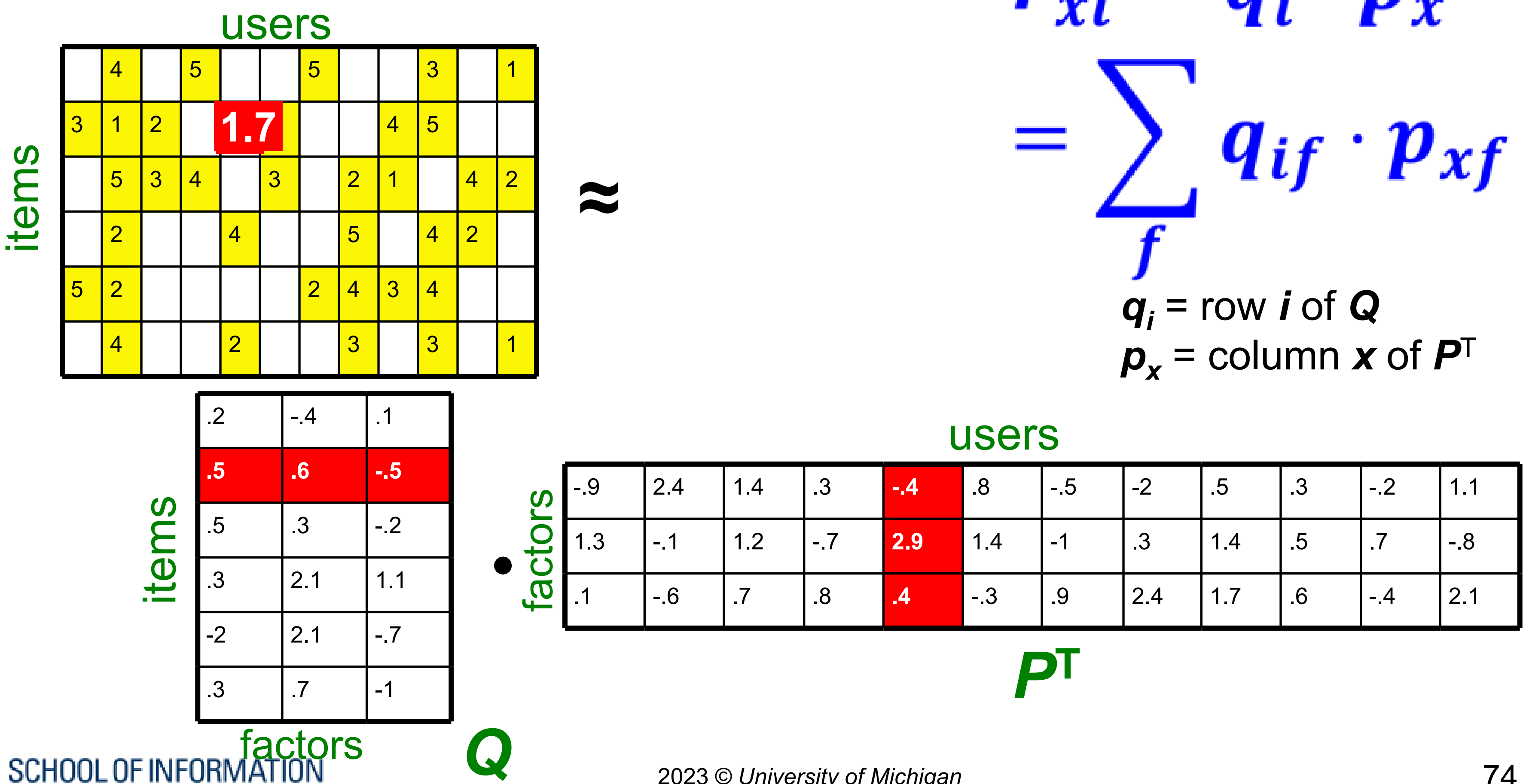
J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Ratings as Products of Factors

- **How to estimate the missing rating of user x for item i?**

$$\hat{r}_{xi} = q_i \cdot p_x$$
$$= \sum_f q_{if} \cdot p_{xf}$$

q_i = row i of Q
p_x = column x of P^T



	4		5			5			3		1
3	1	2		1.7				4	5		
	5	3	4		3		2	1		4	2
	2			4			5		4	2	
5	2					2	4	3	4		
	4			2			3		3		1

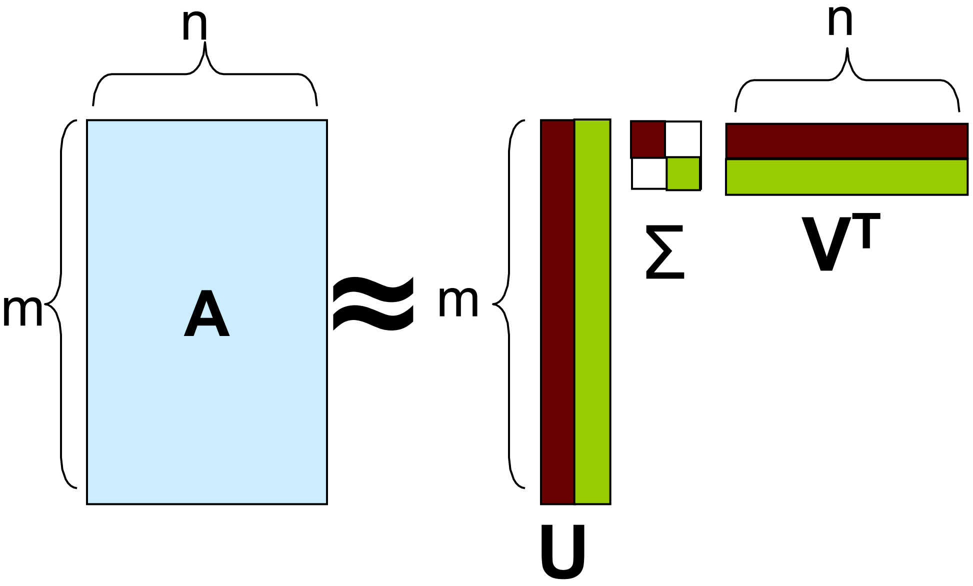
.2	-.4	.1
.5	.6	-.5
.5	.3	-.2
.3	2.1	1.1
-2	2.1	-.7
.3	.7	-1

-.9	2.4	1.4	.3	-.4	.8	-.5	-2	.5	.3	-.2	1.1
1.3	-.1	1.2	-.7	2.9	1.4	-1	.3	1.4	.5	.7	-.8
.1	-.6	.7	.8	.4	-.3	.9	2.4	1.7	.6	-.4	2.1

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Recap: SVD

- **Remember SVD:**
  - **A**: Input data matrix
  - **U**: Left singular vecs
  - **V**: Right singular vecs
  - **Σ**: Singular values



- **So in our case:**
  "SVD" on Netflix data: $R \approx Q \cdot P^T$
  $A = R,\ \ Q = U,\ \ P^{\mathrm{T}} = \Sigma\ V^{\mathrm{T}}$

$$\hat{r}_{xi} = q_i \cdot p_x$$

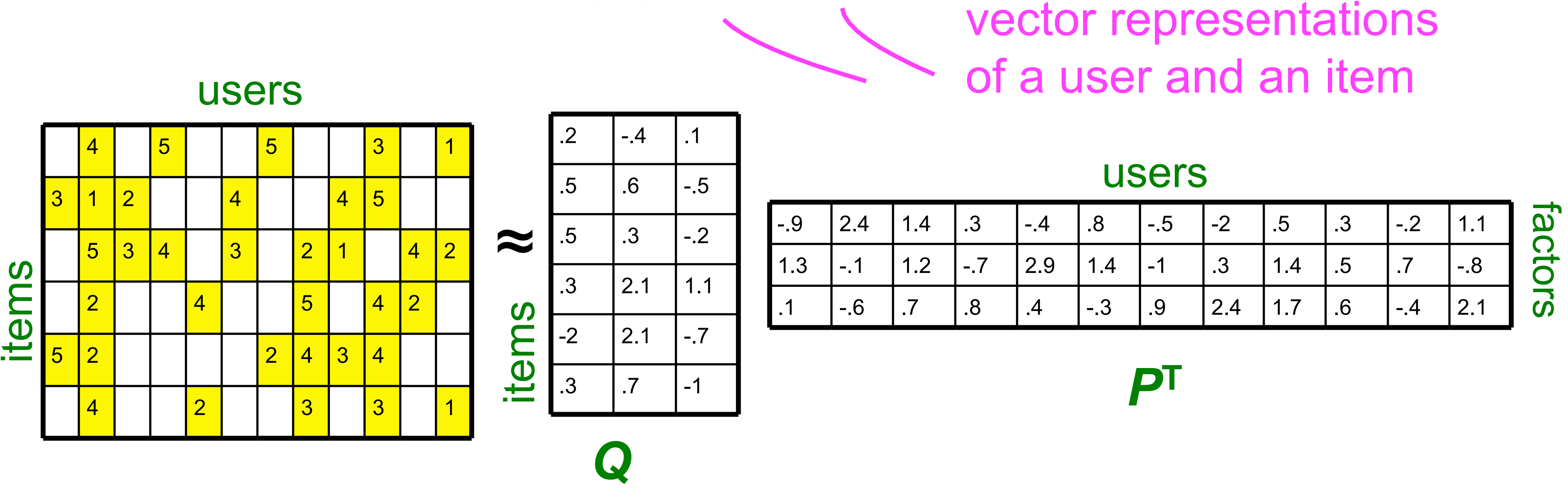
Finding the Latent Factors

Latent Factor Models

- Our goal is to find P and Q such that

$$\min_{P,Q} \sum_{(i,x)\in R} (r_{xi} - q_i \cdot p_x)^2$$

vector representations of a user and an item



users

items

	4		5			5			3		1
3	1	2			4			4	5		
	5	3	4		3		2	1		4	2
	2			4			5		4	2	
5	2					2	4	3	4		
	4			2			3		3		1

≈

items

.2	-.4	.1
.5	.6	-.5
.5	.3	-.2
.3	2.1	1.1
-2	2.1	-.7
.3	.7	-1

Q

users

factors

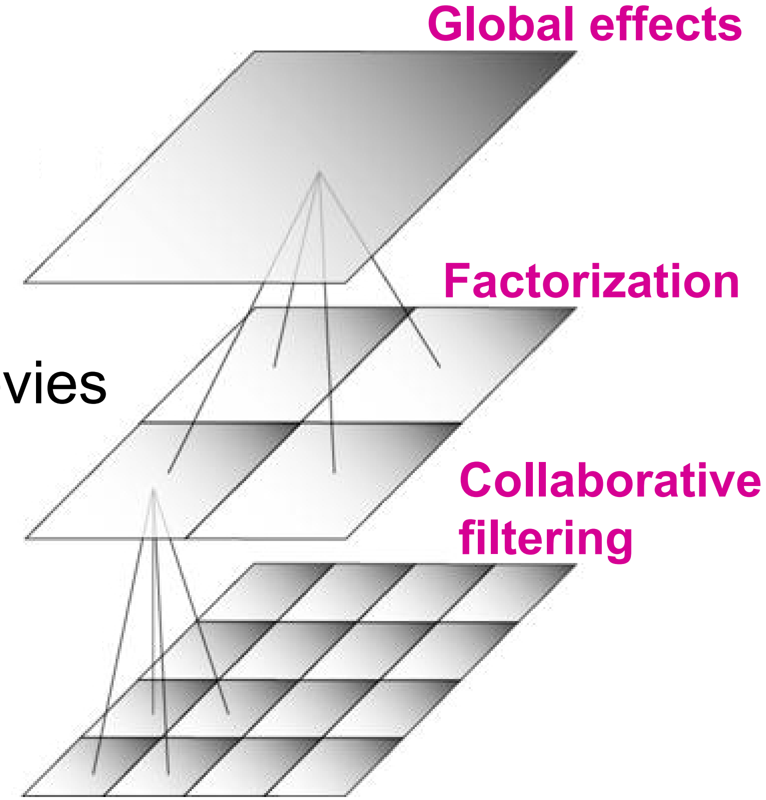
-.9	2.4	1.4	.3	-.4	.8	-.5	-2	.5	.3	-.2	1.1
1.3	-.1	1.2	-.7	2.9	1.4	-1	.3	1.4	.5	.7	-.8
.1	-.6	.7	.8	.4	-.3	.9	2.4	1.7	.6	-.4	2.1

P^T

- High level idea: Initialize P and Q using the initial matrix with the zeros and apply SGD.

BellKor Recommender System

- **The winner of the Netflix Challenge!**
- **Multi-scale modeling of the data:**
  Combine top level, "regional" modeling of the data, with a refined, local view:
  - **Global:**
    - Overall deviations of users/movies
  - **Factorization:**
    - Addressing "regional" effects
  - **Collaborative filtering:**
    - Extract local patterns



J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

What you should know

- What are the different types of filtering
- What are the challenges of adaptive filtering
- What are the challenges of collaborative filtering
- How to compare users and items
- Simple methods for implementing collaborative filtering