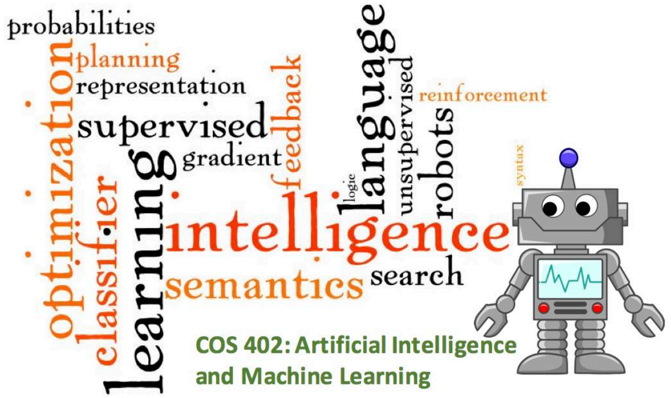


COS 402 – Machine Learning and Artificial Intelligence
Fall 2016

Lecture 11: Recommender Systems

Sanjeev Arora   Elad Hazan

PRINCETON UNIVERSITY

(Borrows from slides of D. Jurafsky Stanford U.)

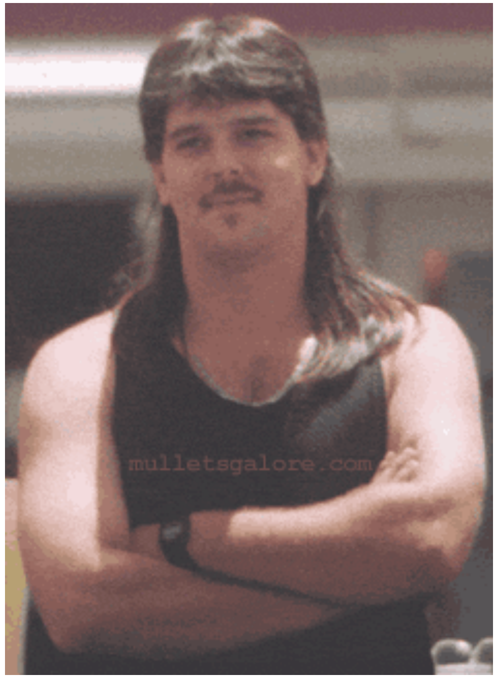
Admin

- Exercise 5 (written), next Tue, in class
- Midterm next Thu
- Survey results, analysis and follow-up

Recap

- Learning from examples
- Movie / philosophy of AI. Dr. Singer on Google ML.
- Language
  - Probabilistic model of language
  - Semantics via word embedding
  - Today: recommender systems
- Knowledge representation
- Reinforcement learning

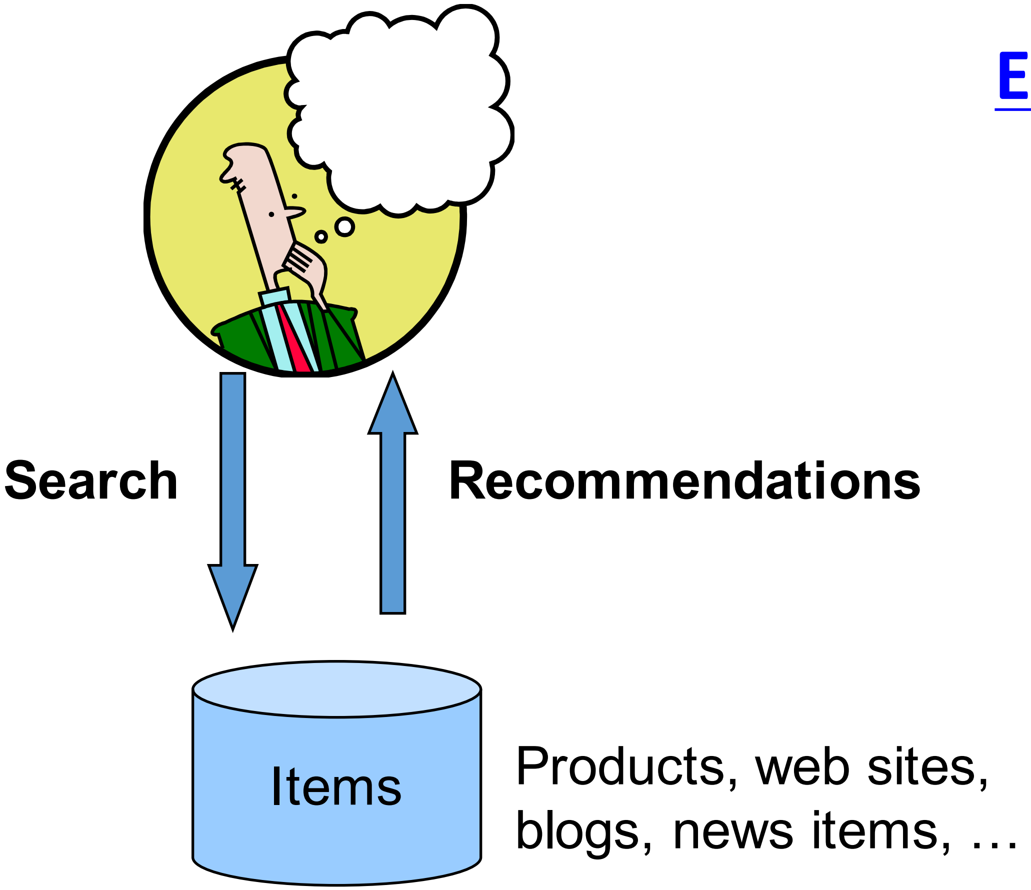
Recommender Systems



- **Customer X**
  - Buys Metallica CD
  - Buys Megadeth CD

- **Customer Y**
  - Does search on Metallica
  - Recommender system suggests Megadeth from data collected about customer **X**

Recommendations



Examples:

From Scarcity to Abundance

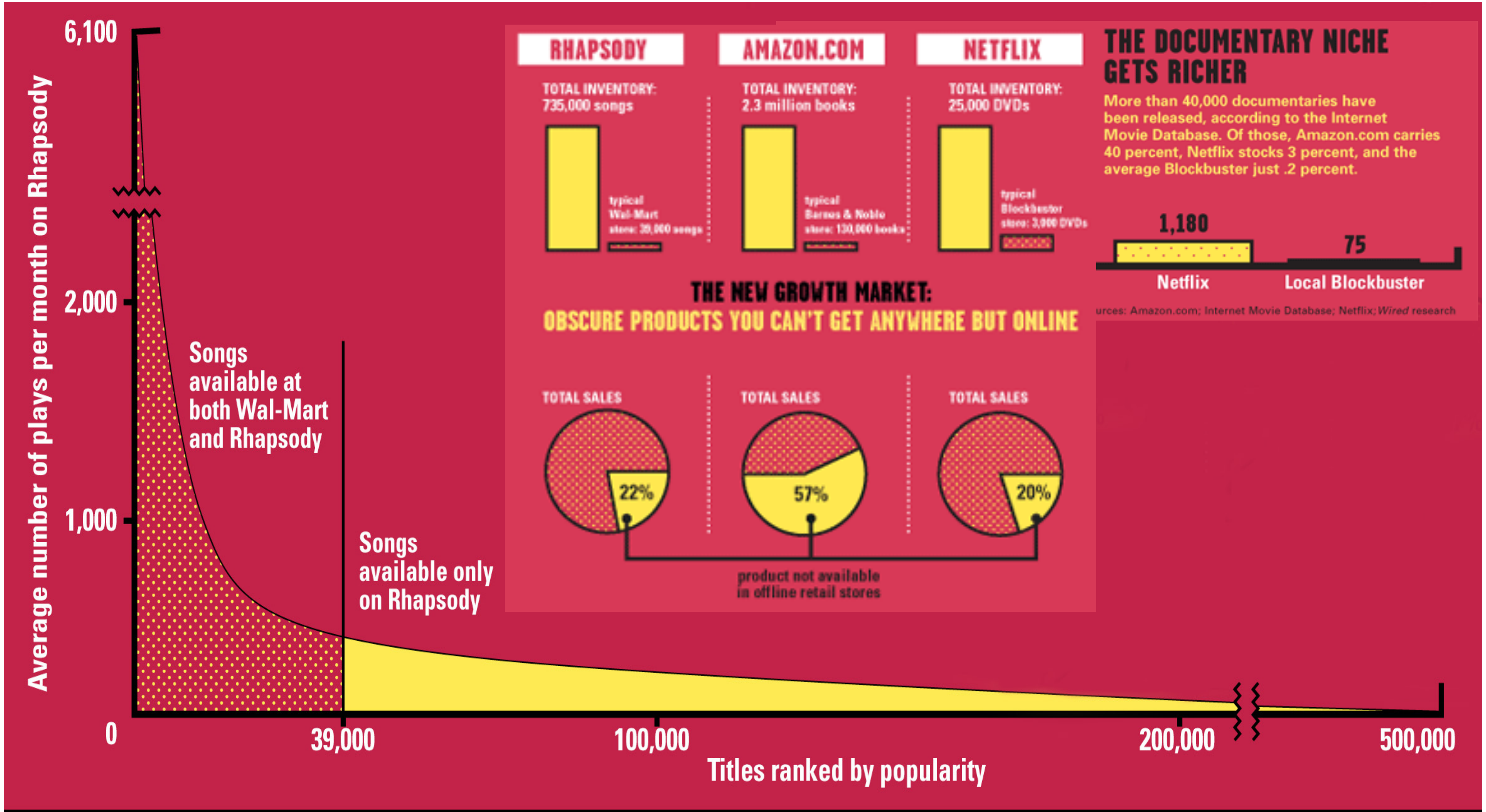
- **Shelf space is a scarce commodity for traditional retailers**
  - Also: TV networks, movie theaters,…

- **Web enables near-zero-cost dissemination of information about products**
  - From scarcity to abundance

- **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


MAJESTIC
NOW PLAYING:
EVERY MOVIE EVER MADE

Sidenote: The Long Tail



Sources: Erik Brynjolfsson and Jeffrey Hu, MIT, and Michael Smith, Carnegie Mellon; Barnes & Noble; Netflix; RealNetworks

Source: Chris Anderson (2004)

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, …

Today class

Formal Model

- $\boldsymbol{X}$ = set of **Customers**
- $\boldsymbol{S}$ = set of **Items**

- **Utility function** $\boldsymbol{u}: \boldsymbol{X} \times \boldsymbol{S} \rightarrow \boldsymbol{R}$
  - $\boldsymbol{R}$ = set of ratings
  - $\boldsymbol{R}$ 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

Key Problems

- **(1) Gathering “known” ratings for matrix**
  - How to collect the data in the utility matrix
- **(2) Extrapolate unknown ratings from known ones – MAIN LEARNING PROBLEM**
  - Mainly interested in high unknown ratings
- **(3) Evaluating extrapolation methods**
  - How to measure success/performance of recommendation methods

(1) Gathering Ratings

- **Explicit**
  - Ask people to rate items
  - Doesn’t work well in practice – people can’t be bothered
  - Crowdsourcing: Pay people to label items
- **Implicit**
  - Learn ratings from user actions
    - E.g., purchase implies high rating
  - What about low ratings?

(2) Extrapolating Utilities

- **Key problem:** Utility 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

Today!

Content-based Recommender Systems

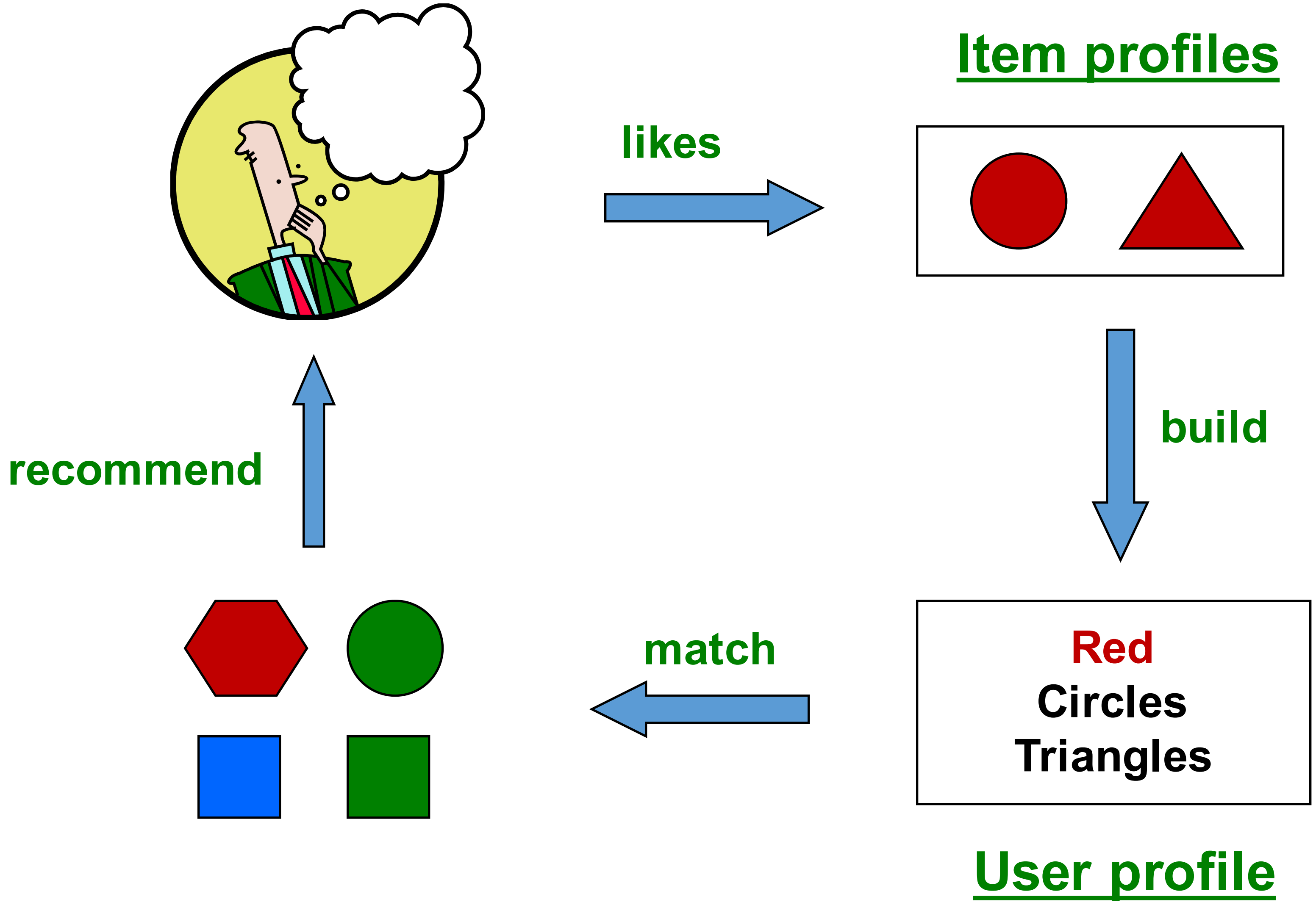
Content-based Recommendations

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

Example:

- **Movie recommendations**
  - Recommend movies with same actor(s), director, genre, …
- **Websites, blogs, news**
  - Recommend other sites with “similar” content

Plan of Action



Item Profiles

- For each item, create an **item profile**
- **Profile is a set (vector) of features**
  - **Movies:** author, genre, director, actors, year…
  - **Text:** Set of "important" words in document
- **How to pick important features?**
  - **TF-IDF** (Term frequency * Inverse Doc Frequency)

	Actor A	Actor B	…		Johnny Depp	Comic Genre	Spy Genre	Pirate Genre	Avg Rating
Movie X	0	1	1	0	1	1	0	1	3
Movie Y	1	1	0	1	0	1	1	0	4

User Profiles

- **Want a vector with the same components/dimensions as items**
  - Could be 1s representing user purchases
  - Or arbitrary numbers from a rating
- **User profile is aggregate of items:**
  - Average(weighted?)of rated item profiles

	Natalie Portman	Actor A	Actor B	…	
User U	0.2	.005	0	0	…

Prediction

- User and item vectors have the same components/dimensions, recommend the items whose vectors are most similar to the user vector!

- Given user profile ***x*** and item profile ***i***,
- estimate $u(\boldsymbol{x}, \boldsymbol{i}) = \cos(\boldsymbol{x}, \boldsymbol{i}) = \frac{\boldsymbol{x} \cdot \boldsymbol{i}}{||\boldsymbol{x}|| \cdot ||\boldsymbol{i}||}$

Pros

- **+: No need for data on other users**
  - No cold-start or sparsity problems
- **+: Able to recommend to users with unique tastes**
- **+: Able to recommend new & unpopular items**
  - No first-rater problem
- **+: Able to provide explanations**
  - Can provide explanations of recommended items by listing content-features that caused an item to be recommended

Cons

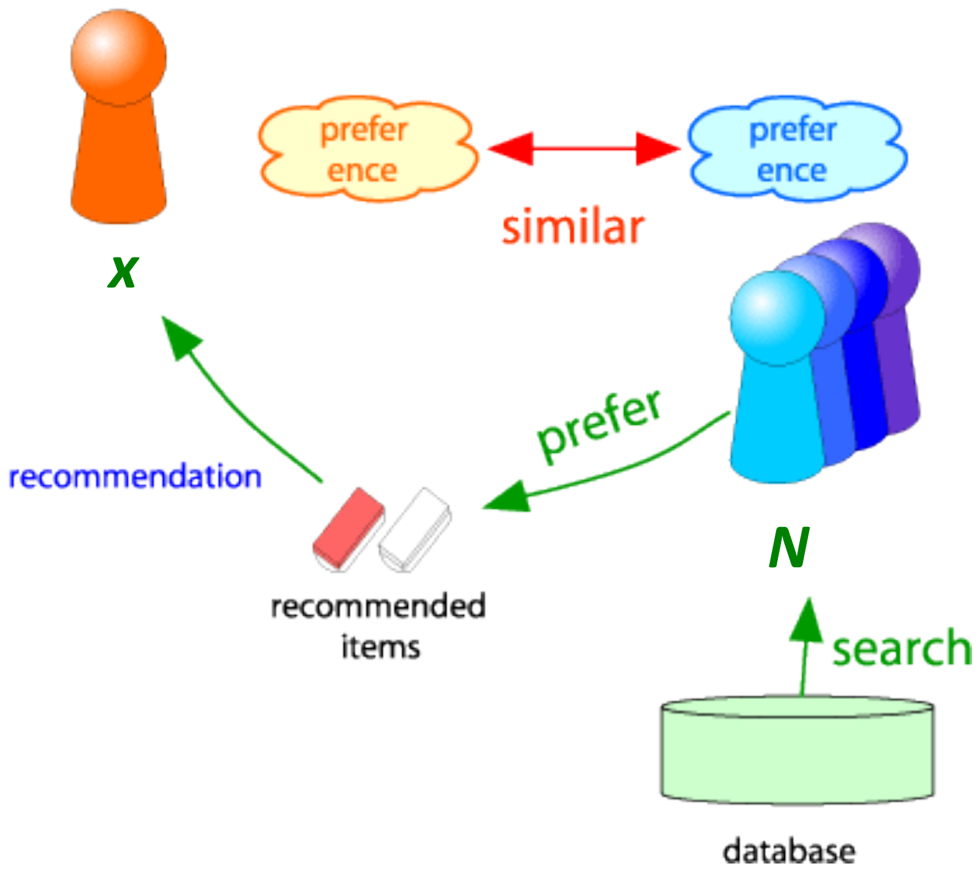
- **–: Finding the appropriate features is hard**
  - E.g., images, movies, music
- **–: Recommendations for new users**
  - **How to build a user profile?**
- **–: Overspecialization**
  - Never recommends items outside user's
    content profile
  - People might have multiple interests
  - **Unable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

Collaborative Filtering

Harnessing quality judgments of other users

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***



MAIN: a methodological learning-based approach

A methodological learning-based approach

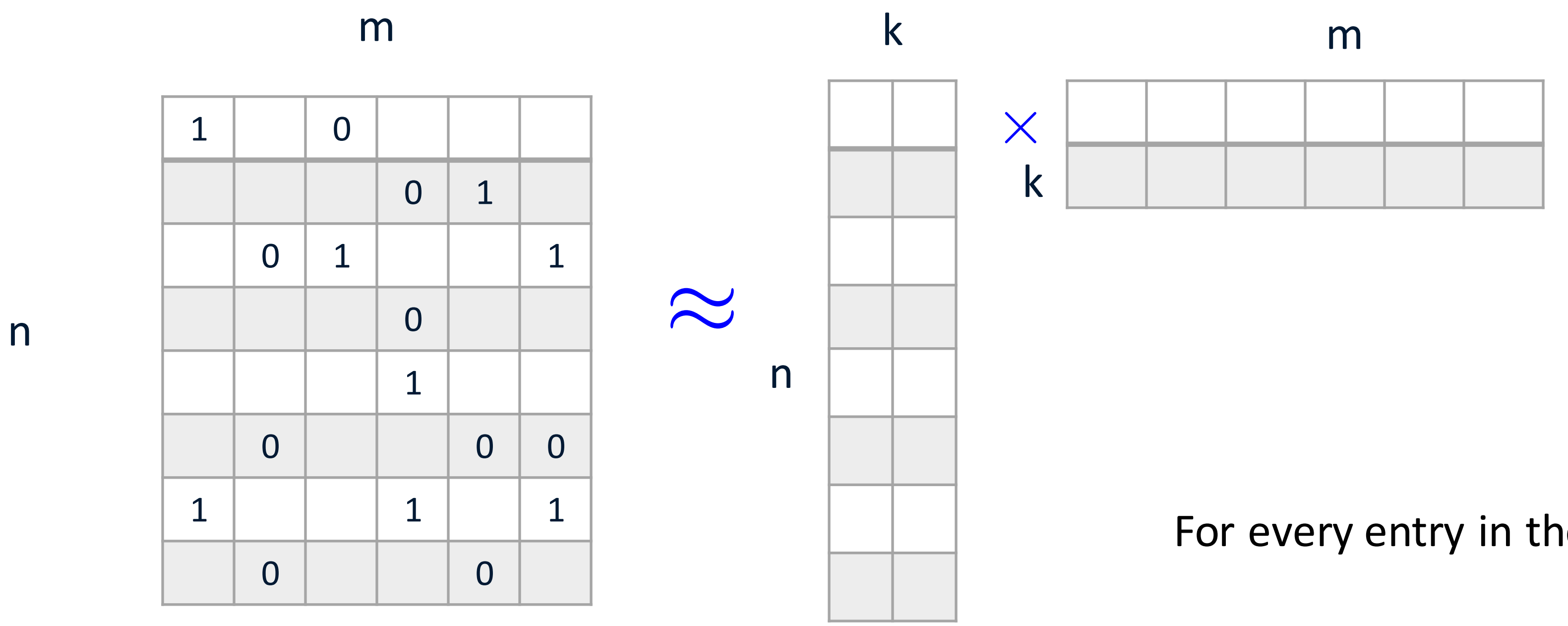
movies

users

1		0			
			0	1	
	0	1			1
			0		
			1		
	0			0	0
1			1		1
	0			0	

How many factors determine preference?

The low-rank assumption



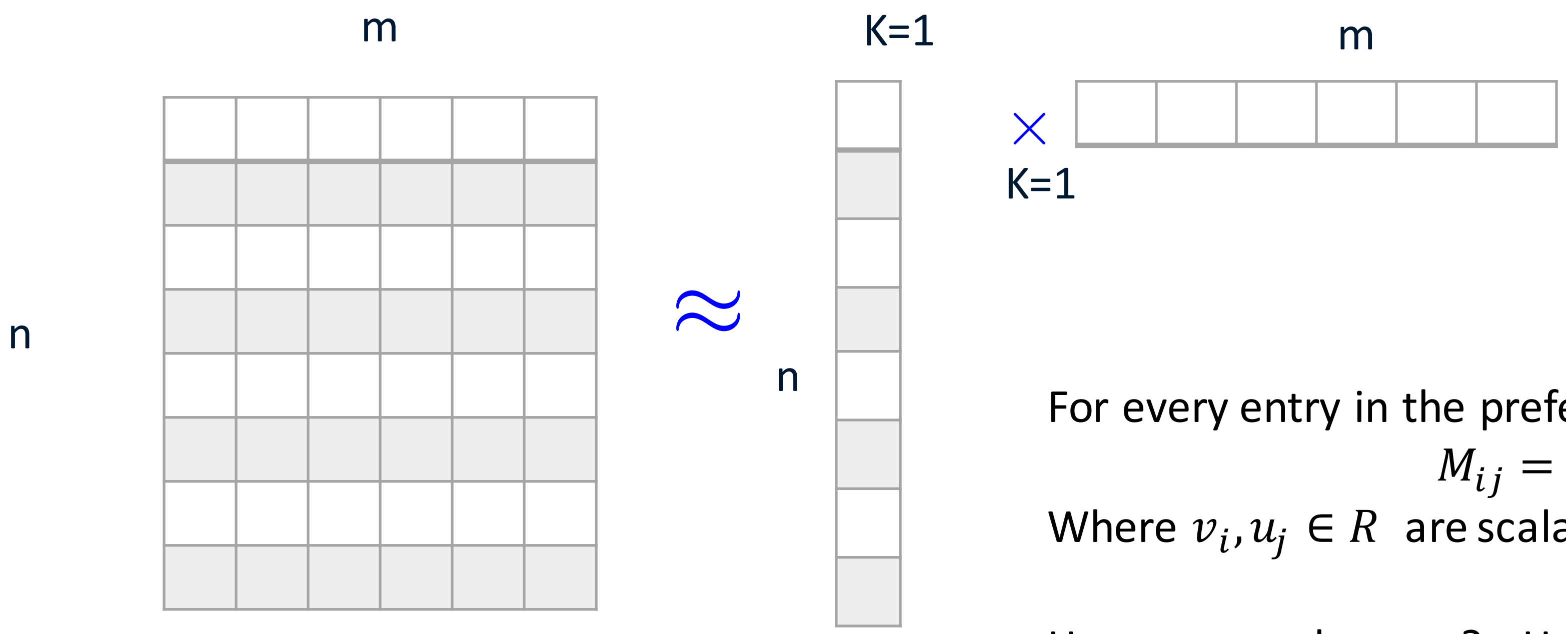
For every entry in the preference matrix

$$M_{ij} = v_i \cdot u_j = \sum_{t=1 \text{ to } k} v_i(t) u_j(t)$$

Where $v_i, u_j \in R^k$

"Preference is determined by k factors"
usually k={5,…,10}

Example – rank 1 and its benefits



For every entry in the preference matrix

$$M_{ij} = v_i \cdot u_j$$

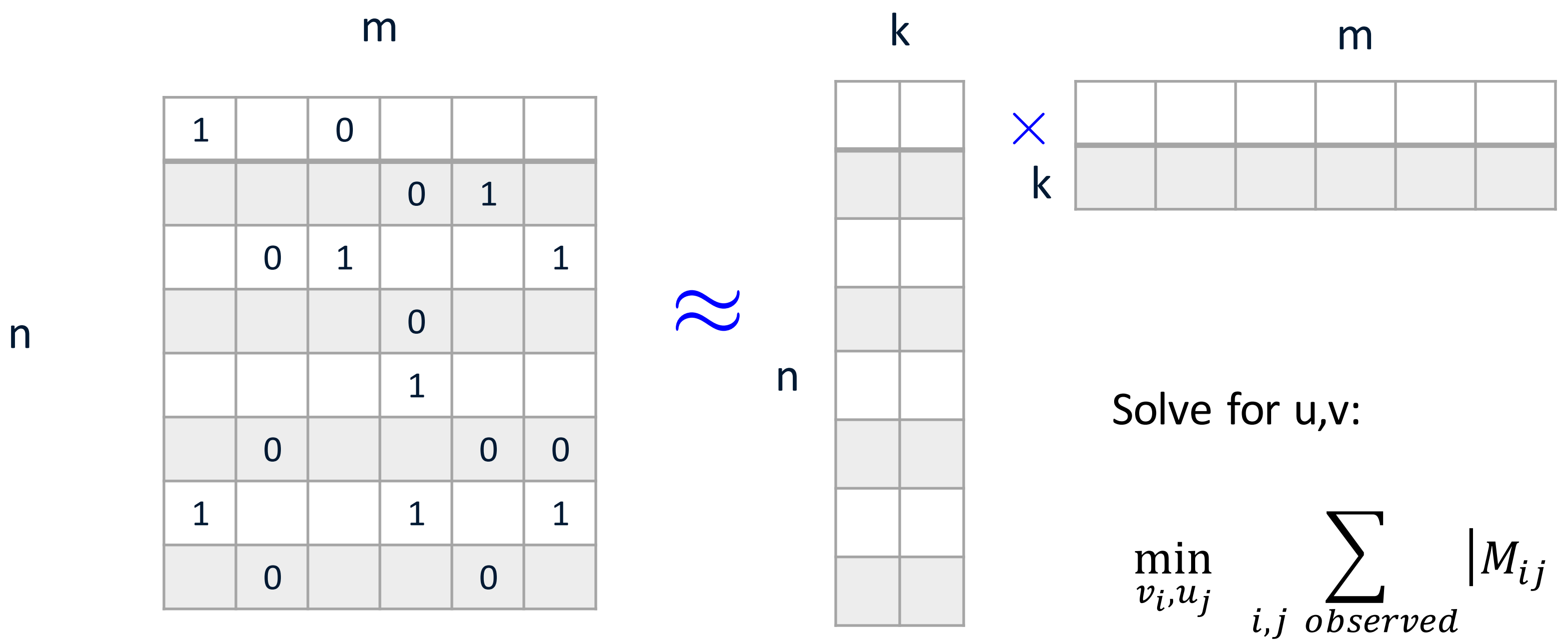
Where $v_i, u_j \in R$ are scalars.

How many unknowns? How many observations are needed to complete the matrix?

(food for thought: relate to statistical learning theory – sample complexity?)

After observing (m+n) entries – can compute the entire matrix!

The matrix completion approach



Solve for u,v:

$$\min_{v_i,u_j} \sum_{i,j\ observed} \left|M_{ij} - v_i \cdot u_j\right|^2$$

Where $v_i, u_j \in R^k$

Total of k(m+n) variables.

An algorithm for predicting recommendations

Input: observations of preferences M_{ij} for $\{(i_1, j_1), (i_2, j_2), \dots, (i_m, j_m)\}$ (m numbers in the range [0,1])

Output: A matrix $\mathrm{M} \in R^{users \times movies}$ that has all predicted preferences

Assumption: there exist low dimensional vectors $\{\mathrm{v_i}, \mathrm{u_j}\}$ such that $\mathrm{M_{ij}} = \mathrm{v_i} \cdot u_j$

Algorithm: Gradient descent! Objective function:

$$f(\{u, v\}) = \sum_{i,j \ observed} \left| M_{ij} - v_i \cdot u_j \right|^2$$

What do we do with the vectors?

Spelling out GD in this case

GD for matrix completion:

$$f(\{u,v\}) = \sum_{i,j \ \ observed} \left|M_{ij} - v_i \cdot u_j\right|^2$$

- Initialize $\mathrm{u_i}, \mathrm{v_j}$ randomly
- For iteration = 1,2,… do:
  - Update $v \leftarrow v - \eta \frac{\partial}{\partial v} f\left(\{v_i, u_j\}\right)$ for all vectors $\{v_i, u_j\}$

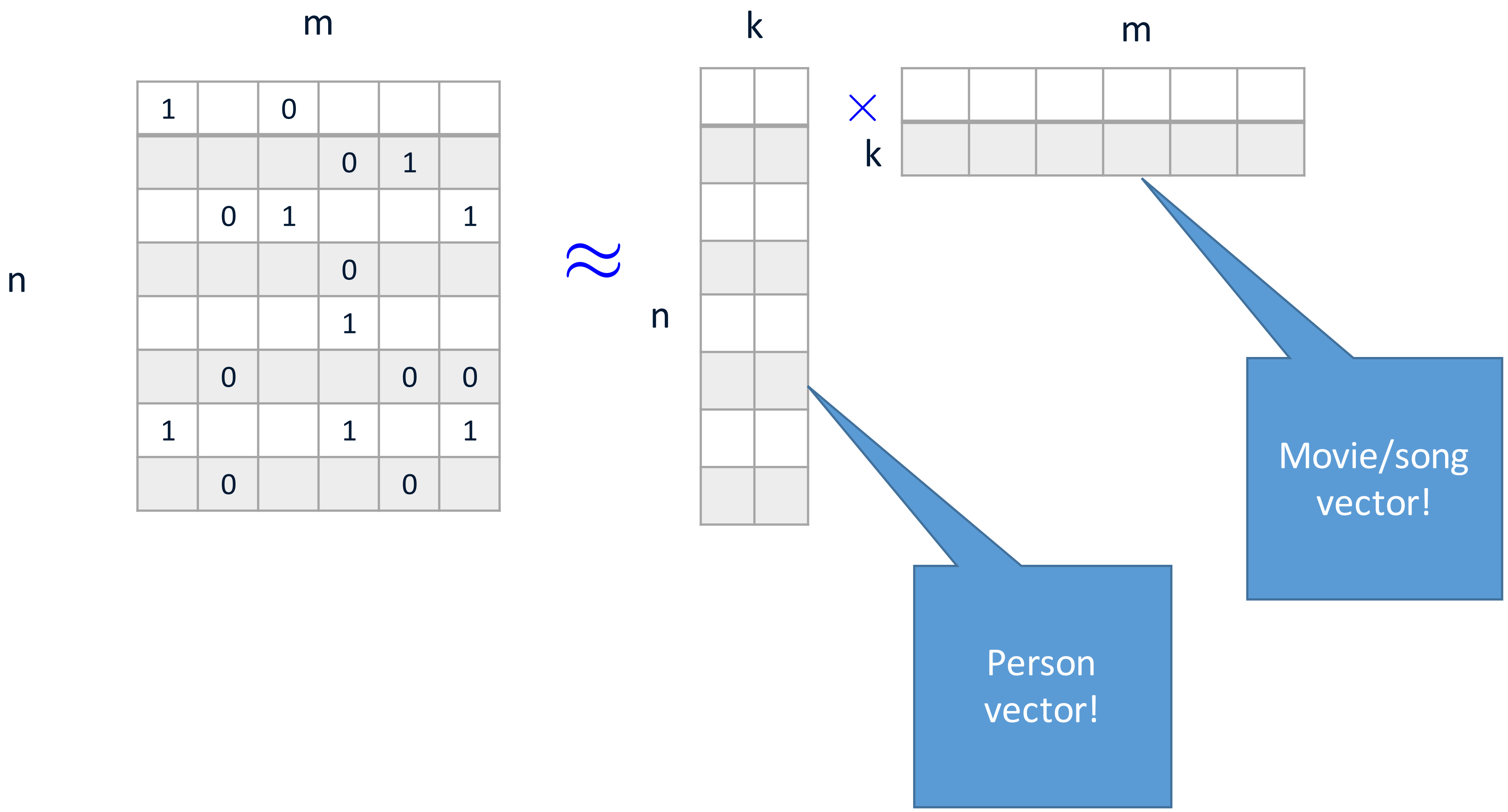
    spelling it out, for each coordinate t of vector v_i, update:

    $$\frac{\partial}{\partial v_a(t)} f = 2 \ \textstyle\sum_j (M_{aj} - v_a \cdot u_j) \cdot u_j(t)$$ Thus,

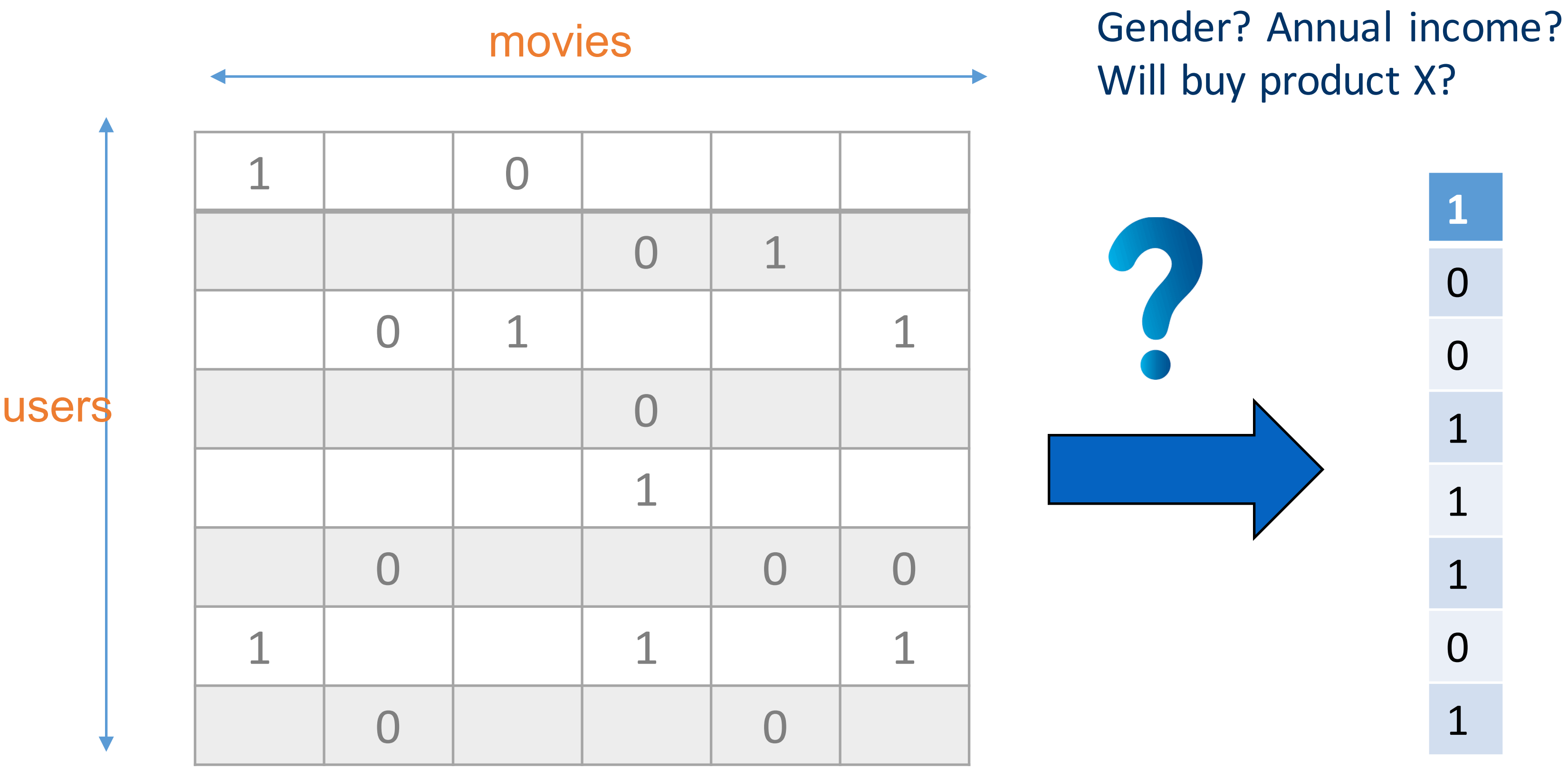
    $$\forall a, t: \quad v_a(t) \leftarrow v_a(t) - \eta \cdot 2 \sum_j (M_{aj} - v_a \cdot u_j) \cdot u_j(t)$$

  - If needed, normalize each vector, $v \leftarrow \frac{v}{\max\{1, |v|\}}$
- End For
- Return final (or average of last few) vector solutions

Predicting meta-data from rec. data



Predicting meta-data from rec. data [Esther Rolf ’15]



Implications to user privacy, security,…

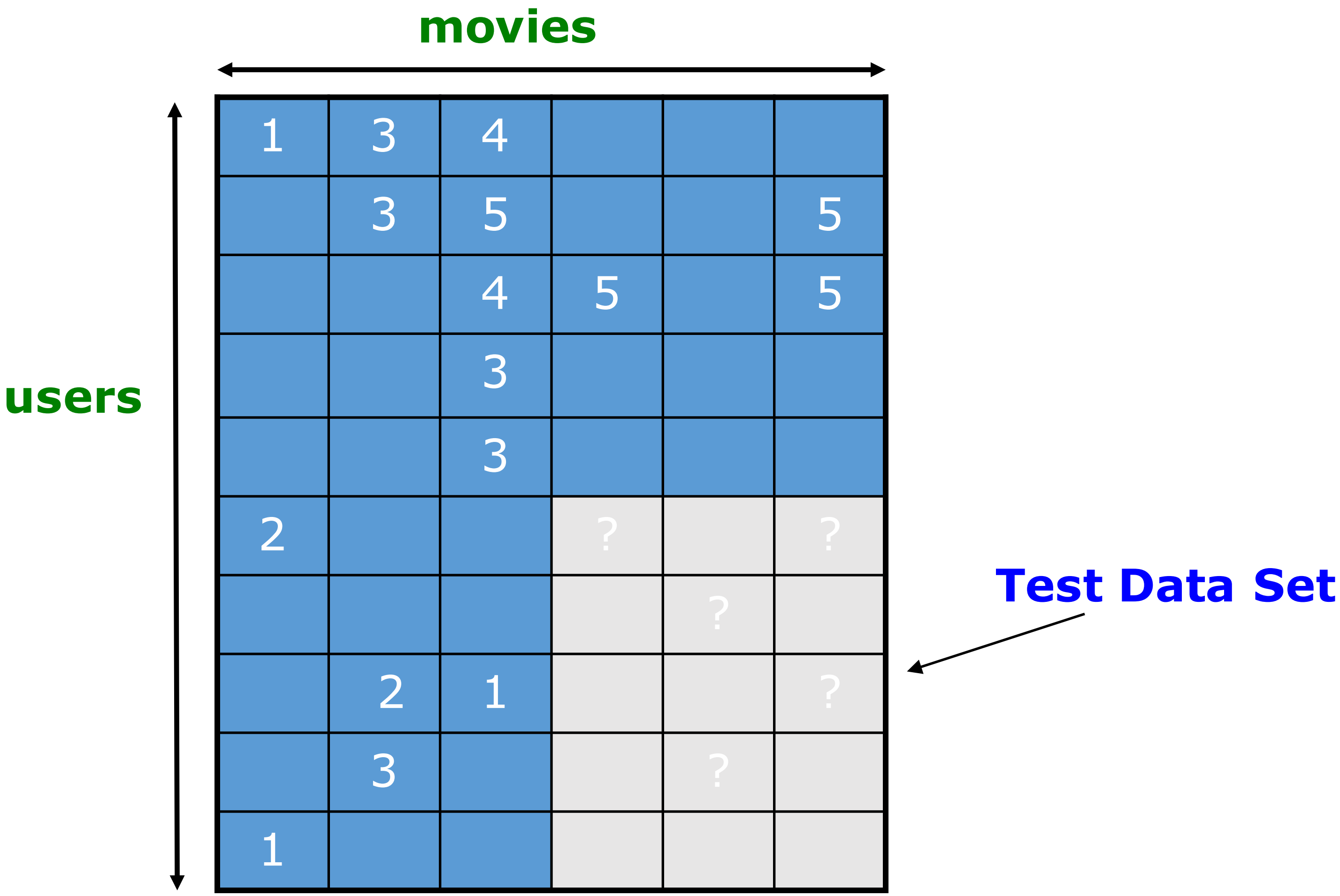
Evaluation

movies

users

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

Evaluation



Evaluating Predictions

- **Compare predictions with known ratings**
  - **Root-mean-square error** (RMSE)
    - $\sqrt{\sum_{xi}\left(r_{xi} - r_{xi}^{*}\right)^{2}}$ where $\boldsymbol{r_{xi}}$ is predicted, $\boldsymbol{r_{xi}^{*}}$ is the true rating of ***x*** on ***i***

- **Narrow focus on accuracy sometimes misses the point**
  - Prediction Diversity
  - Prediction Context
  - Order of predictions
- **In practice, we care only to predict high ratings:**
  - RMSE might penalize a method that does well for high ratings and badly for others

Famous Historical Example: The Netflix Prize

- **Training data**
  - 100 million ratings, 480,000 users, 17,770 movies
  - 6 years of data: 2000-2005
- **Test data**
  - Last few ratings of each user (2.8 million)
  - Evaluation criterion: root mean squared error (RMSE)
  - Netflix Cinematch RMSE: 0.9514
- **Competition**
  - 2700+ teams
  - $1 million prize for 10% improvement on Cinematch
  - BellKor system won in 2009. Combined many factors
    - Overall deviations of users/movies
    - Regional effects
    - Local collaborative filtering patterns
    - Temporal biases

Summary: Recommendation Systems

- The Long Tail
- Content-based Systems
- Collaborative Filtering (touched)
- Latent Factors

- Food for thought: sample complexity?