CS 5683: Big Data Analytics

Recommender Systems: Content-based Systems & Collaborative Filtering

Arunkumar Bagavathi

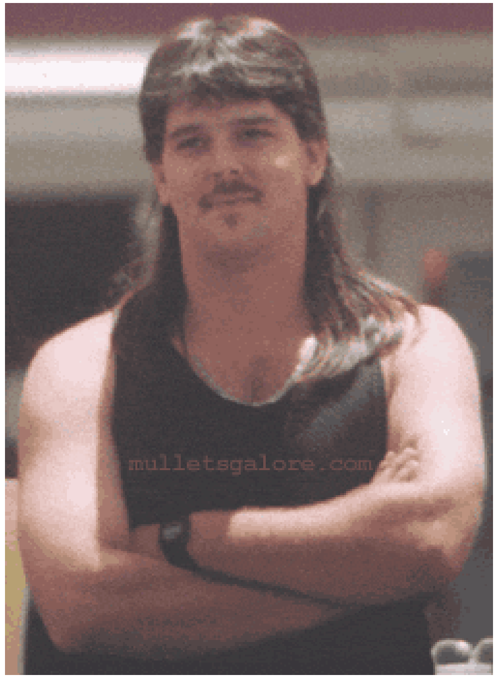
Department of Computer Science

Oklahoma State university

Topics Overview

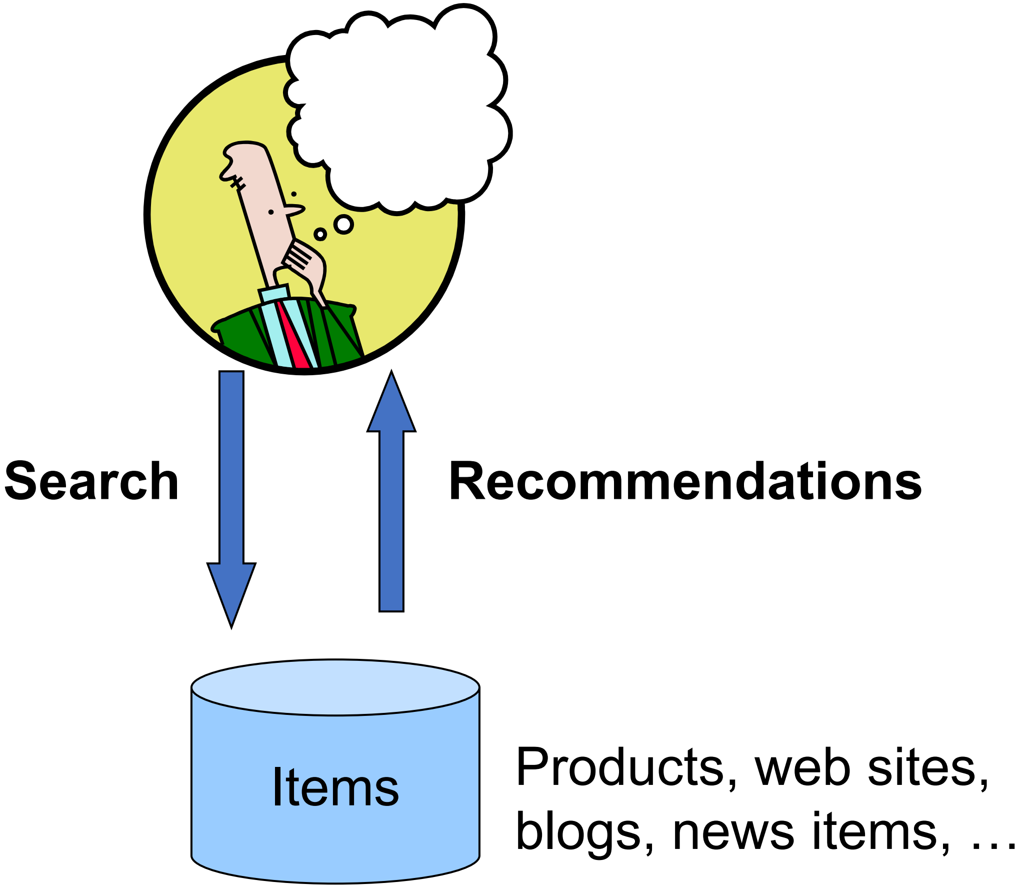
High. Dim. Data	Text Data	Graph Data	Others
Data Features	Clustering	PageRank	Data Streams Mining
Dimensionality Reduction	Non-linear Dim. Reduction	ML for Graphs	Intro. to Apache Spark
Application Rec. Systems	*Application* IR	Community Detection	

Example Recommender Systems



- **Customer X**
  - Buys the book “Origin”
  - Buys the audio book “Inferno”
- **Customer Y**
  - Does search on “Origin”
  - Recommender system suggests Inferno from data collected about customer **X**

Recommendations



Examples:

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

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, …
  - Focus in this course

Formal Recommender Model

- ***X*** = set of **Customers**
- ***S*** = set of **Items**

- **Utility function** ***u***: ***X*** × ***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

Key Problems

(1) Gathering "known" ratings for matrix

- How to collect the data in the utility matrix?

(2) Extrapolate unknown ratings from the known ones

- Mainly interested in unknown ratings
  - We are not interested in knowing what you don't like but what you like

(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

- **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 factors

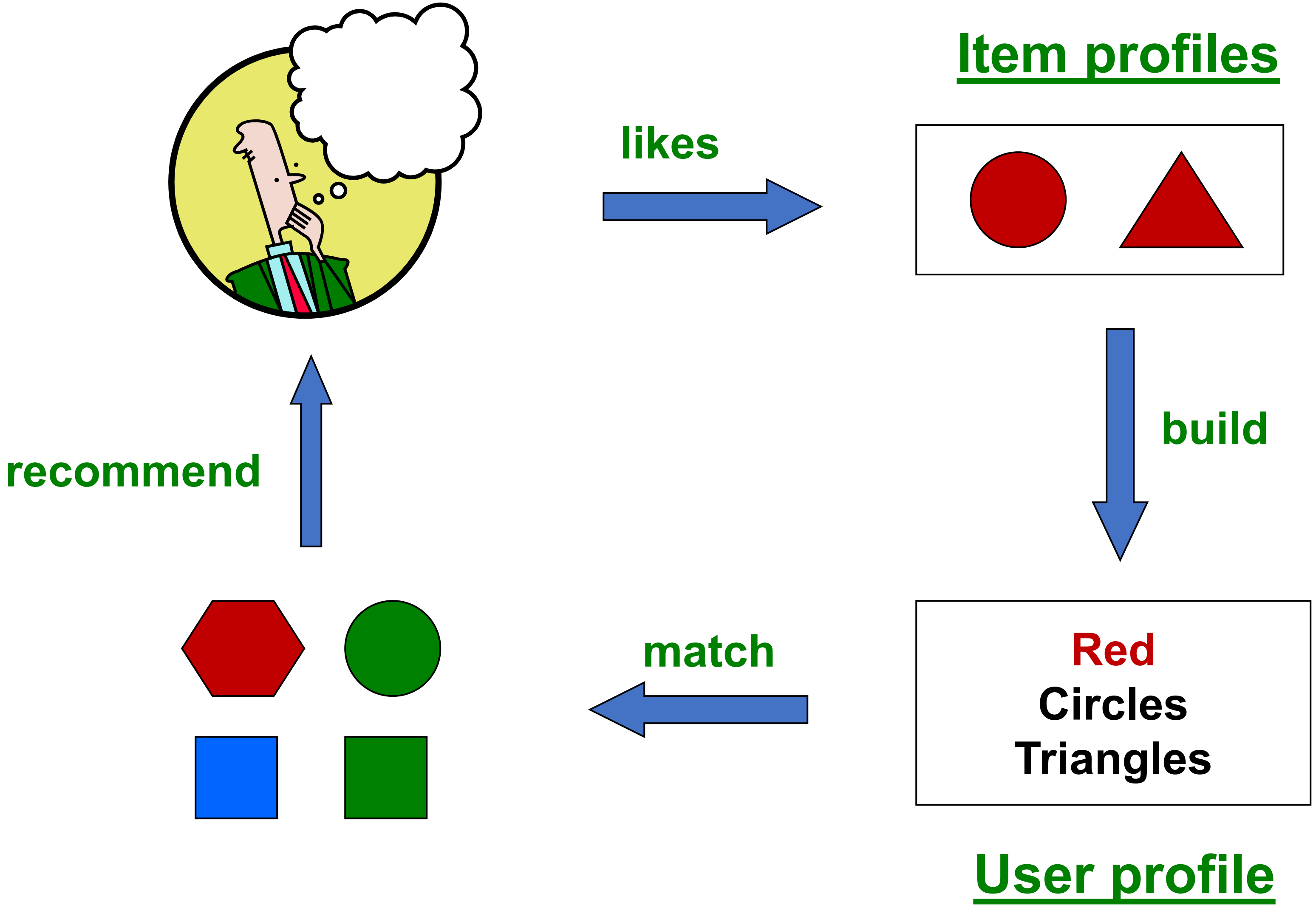
(1) Content-based Recommender Systems

- **Main idea:** Recommend items to customer ***x*** which are 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

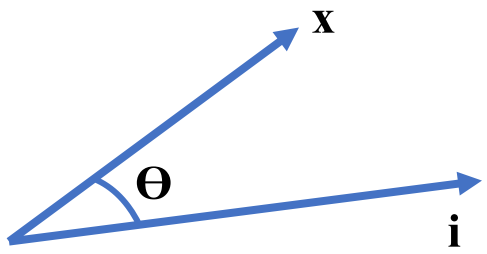
- For each item, create an **item profile**
- **Profile is a set (vector) of features** – one entry per feature and each entry could be Boolean or real valued
  - **Movies:** author, title, actor, director,…
  - **Text:** Set of “important” words in document
  - **People:** Set of friends
- **How to pick important features?**
  - Usual heuristic from text mining is **TF-IDF** (Term frequency * Inverse Doc Frequency)
    - **Term … Feature**
    - **Document … Item**

User Profiles and Prediction

- **User profile possibilities:**
  - (Weighted) average of rated item profiles $\boldsymbol{i_1,\ldots,i_n}$
  - **Variation:** weight by difference from average rating for item
  - …

- **Prediction heuristic:**
  - Given user profile ***x*** and item profile ***i***, estimate

$$u(\boldsymbol{x},\boldsymbol{i}) = \cos(\theta) = \frac{\boldsymbol{x}\cdot\boldsymbol{i}}{||\boldsymbol{x}||\cdot||\boldsymbol{i}||}$$



Pros: Content-based Approach

- **+: No need for data on other users**
  - No 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: Content-based Approach

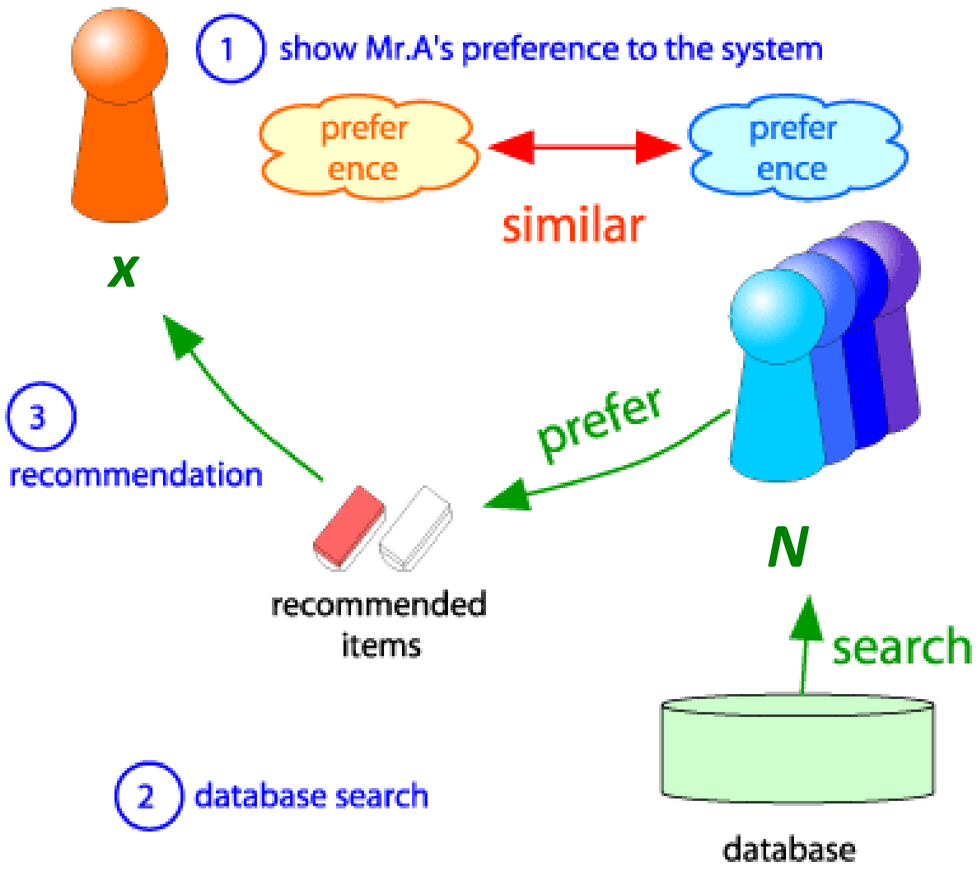
- **–: Finding the appropriate features is hard**
  - E.g., images, movies, music
- **–: Recommendations for new users**
  - **How to build a user profile?**
- **–: Overspecialization**
  - Never recommends items outside user's content profile
  - People might have multiple interests
  - **Unable to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

(2) Collaborative Filtering

- Harness quality judgement of other users

- Consider user ***X***
- Find a set ***N*** of other users whose ratings are "**similar**" to ***X***'s ratings

- Estimate X's ratings based on ratings of users in ***N***



Finding "Similar" Users

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

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

- Let r_x be the vector of user x's ratings
- **Jaccard similarity measure**
  - $sim(x, y) = \frac{|r_x \cap r_y|}{|r_x \cup r_y|}$
  - **Problem:** Ignores the value 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 ratings as "negative"
- **Pearson correlation coefficient**
  - Scale the rating r_{xs} of user x by the user's average rating
  - Low ratings to negative values and empty ratings to zeros

$$r`_{xs} = r_{xs} - \overline{r_x}$$

r_x, r_y as sets:
r_x = {1, 4, 5}
r_y = {1, 3, 4, 2}

r_x, r_y as points:
r_x = {1, 0, 0, 1, 3}
r_y = {1, 0, 2, 2, 0}

$\overline{r_x}$ = avg. rating of **x**

Similarity Metric

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

	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" – **Problem!**
  - **Solution: normalize by subtracting the (row) mean –** centered cosine similarity / pearson correlation

	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

User-User Collaborative Filtering

From similarity metric to recommendations:

- Let $\boldsymbol{r_x}$ 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 *i* of *user x*:**

  1. $r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$
  2. $r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$

Shorthand:
$\boldsymbol{s_{xy} = sim(x, y)}$

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 Collaborative Filtering (|N|=2)

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 Collaborative Filtering (|N|=2)

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 Collaborative Filtering (|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

$m_1 = (1+3+5+5+4)/5 = 3.6$

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 Collaborative Filtering (|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:

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

Item-Item Collaborative Filtering (|N|=2)

users

movies	1	2	3	4	**5**	6	7	8	9	10	11	12	**sim(1,m)**
1	1		3		2.6	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**

Predict by taking weighted average:

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

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

Item-Item Vs. user-User

	Avatar	LOTR	Matrix	Pirates
Alice	1		0.8	
Bob		0.5		0.3
Carol	0.9		1	0.8
David			1	0.4

- **Theory:** User-user and item-item should perform equally
- **In practice, it has been observed that item-item often works better than user-user**
- **Why?** Items are simpler, users have multiple tastes

Collaborative Filtering: Complexity

- Expensive step is finding ***k*** most similar customers (or items): **O(|U|)**
  - **|U| = size of the utility matrix**
- **Too expensive to do at runtime**
  - Could pre-compute
- Naïve pre-computation takes time **O(n ·|U|)**
    - n … set of customers (items)
- **We already know how to do this!**
  - Clustering
  - Dimensionality reduction

Collaborative Filtering: Complexity

- **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

Pros/Cons of Collaborative Filtering

- **+: Works for any kind of items**
  - No feature selection is required
- **–: Cold start**
  - Cannot recommend an item that has not been previously rated
  - New items
- **–: Sparsity**
  - User/ratings matrix is sparse
  - Hard to find users that have rated same items
- **–: Popularity bias**
  - Cannot recommend an item to someone with unique taste
  - Tends to recommend popular items

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

Modelling Local and Global Effects

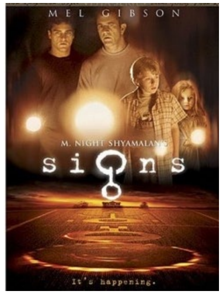
- **Global:**
  - Mean movie rating: **3.7 stars**
  - *The Sixth Sense* is **0.5** stars above avg.
  - Joe rates **0.2** stars below avg.

    ⇒ **Baseline estimation:**

    Joe **will rate** ***The Sixth Sense*** **4 stars**

- **Local neighborhood (CF/NN):**
  - *Joe* didn't like related movie *Signs*
  - ⇒ **Final estimate:**

    Joe **will rate** ***The Sixth Sense*** **3.8 stars**



Collaborative Filtering: 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:

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

Global baseline estimate for r_{xi}

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

- 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

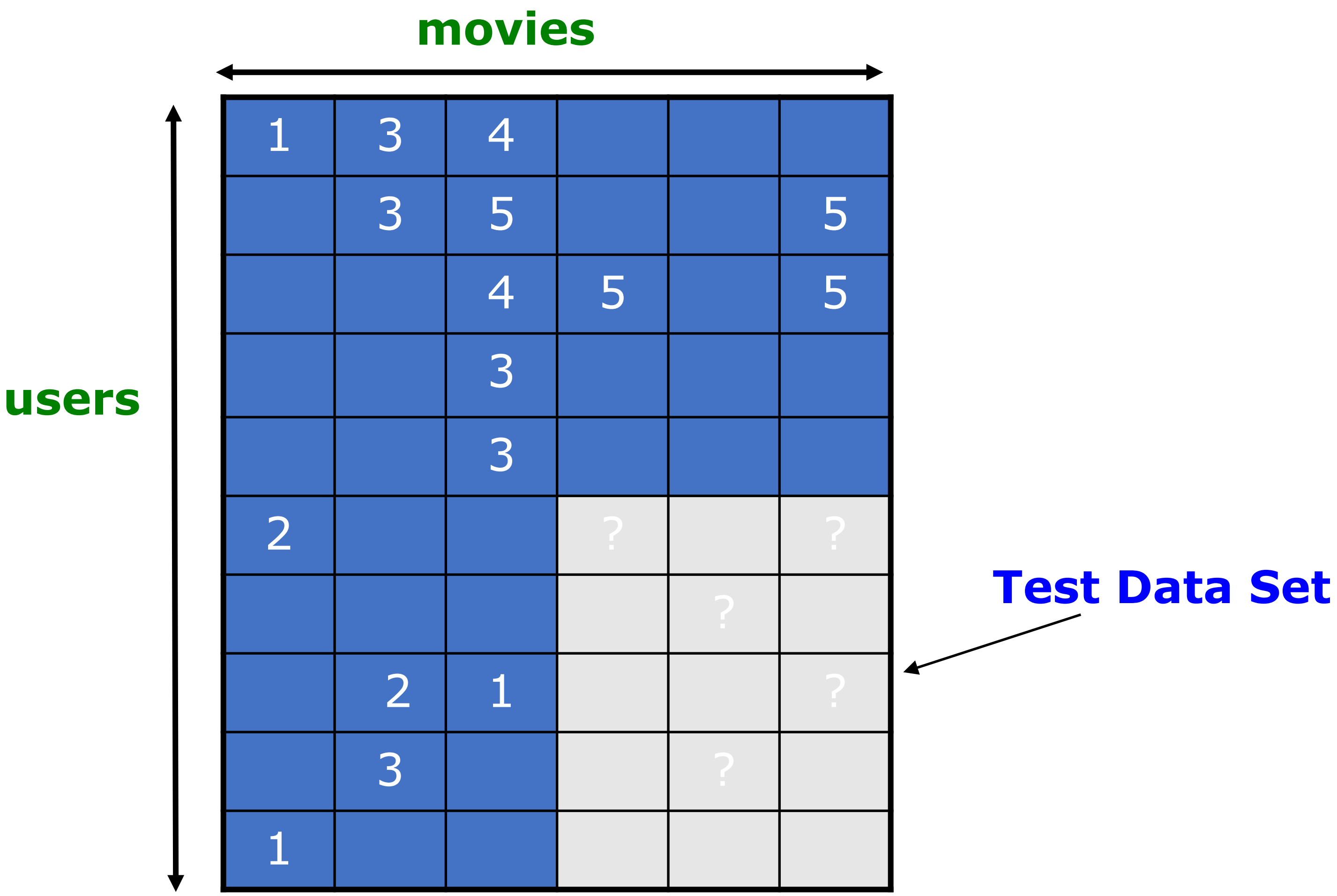
(3) Evaluation – Practical Tips

movies

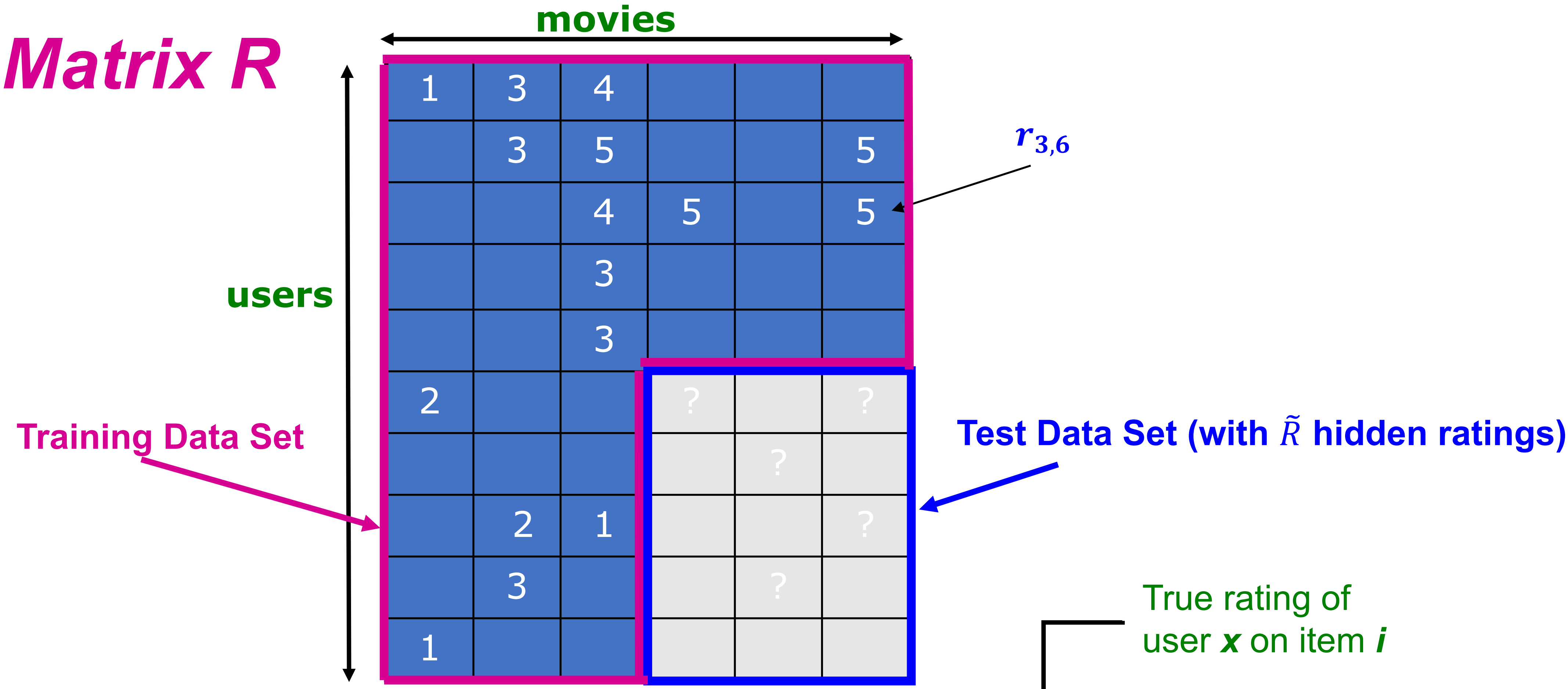
users

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

Evaluation – Practical Tips



Common Practice



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

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

Predicted rating

Evaluating Predictions

- **Compare predictions with known ratings**
  - **Root-mean-square error** (RMSE)
    - $\sqrt{\frac{1}{R^*}\sum_{xi}\left(r_{xi} - r_{xi}^*\right)^2}$ where $\boldsymbol{r_{xi}}$ is predicted, $\boldsymbol{r_{xi}^*}$ is the true rating of $\boldsymbol{x}$ on $\boldsymbol{i}$
  - **Precision/Recall at top *n***:
    - % of those in top 10
  - **Rank Correlation**:
    - Spearman's *correlation* between system's and user's complete rankings

- **Another approach: 0/1 model**
  - **Coverage:**
    - Number of items/users for which system can make predictions
  - **Precision:**
    - Accuracy of predictions
  - **Receiver operating characteristic** (ROC)
    - Tradeoff curve between false positives and false negatives

Problem with Error Measures

- **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

Questions???

Acknowledgements

Most of this lecture slides are obtained from the Mining Massive Datasets course: http://www.mmds.org/