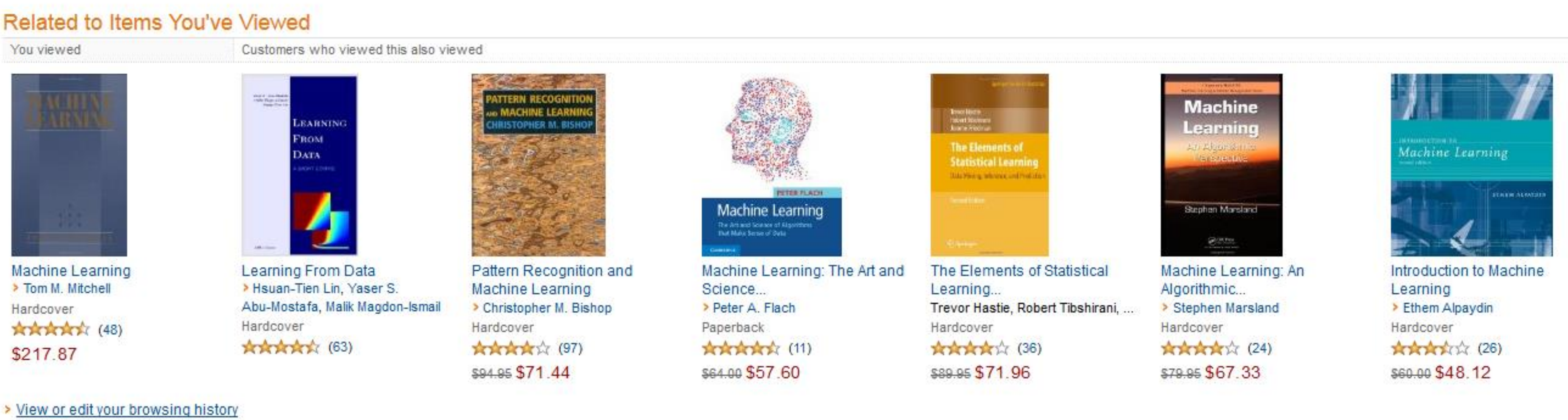
Recommender Systems: Algorithms, Applications, and Evaluation*

* Slides partly based on Jure Leskovec, Anand Raghuraman, and Jeff Ullman

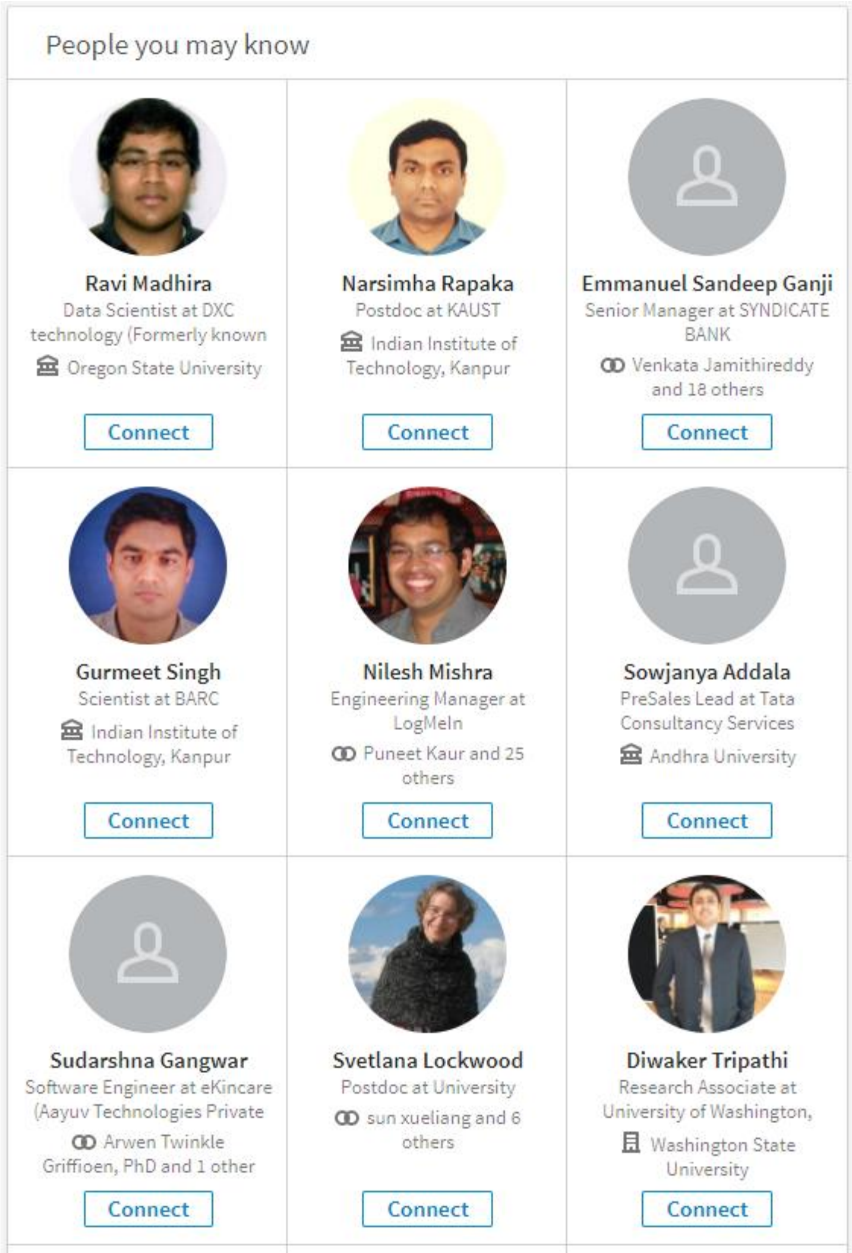
Recommendation Systems: Example

- Amazon recommendation engine

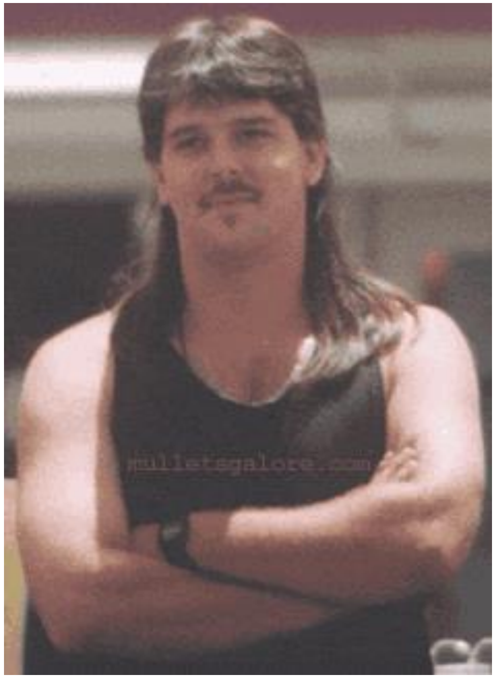


Recommendation Systems: Example

- LinkedIn recommendation engine



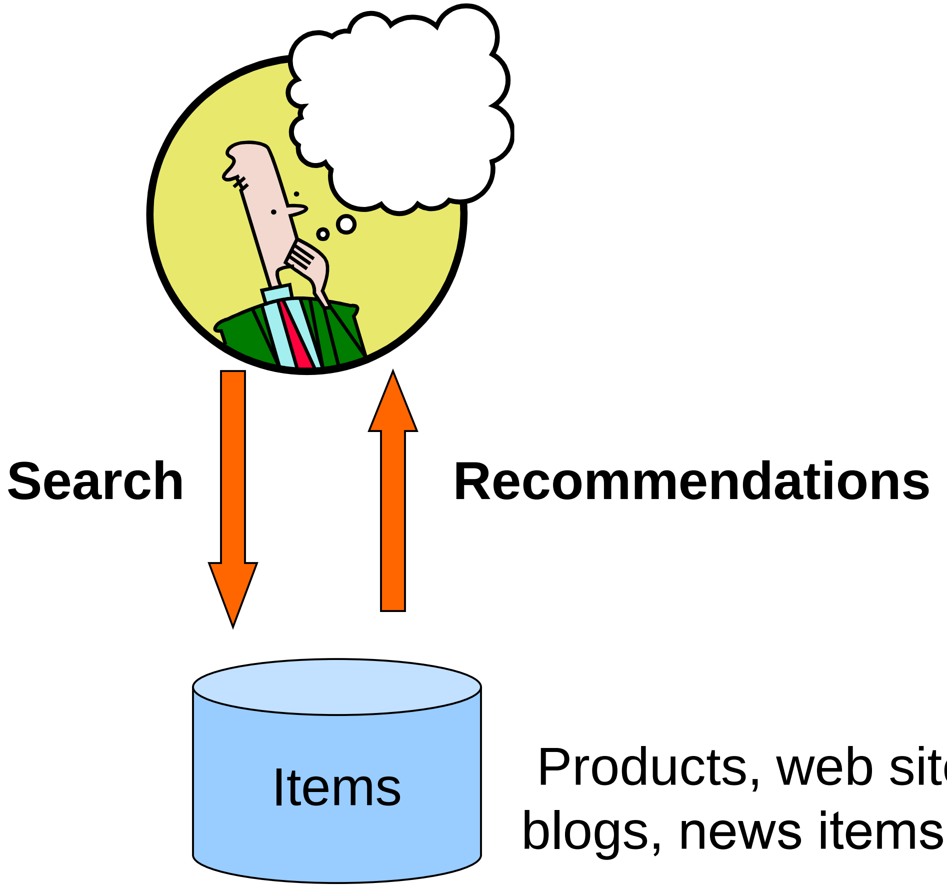
Recommendation Systems: Example



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

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

Recommendation Systems: Overview



Products, web sites, blogs, news items, …

Examples:

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** ⇨ **Our Focus!**
  - Amazon, Netflix, …

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: Example

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

Recommendations: Key Challenges

- **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 (we are not interested in knowing what you don’t like)
- **Evaluating prediction methods**
  - How to measure success/performance of recommendation methods

Recommendations: Key Challenge #1

- **Gathering "known" ratings for matrix**
  - How to collect the data in the utility matrix?

- **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)

Recommendations: Key Challenge #2

- **Extrapolate unknown ratings from the known ones**
  - Mainly interested in ``high'' unknown ratings

- **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

Recommendation Algorithms

- **Three main approaches for recommendations**
  - Content-based Filtering
  - Collaborative Filtering
  - Latent Factor Models

Recommendation Algorithms

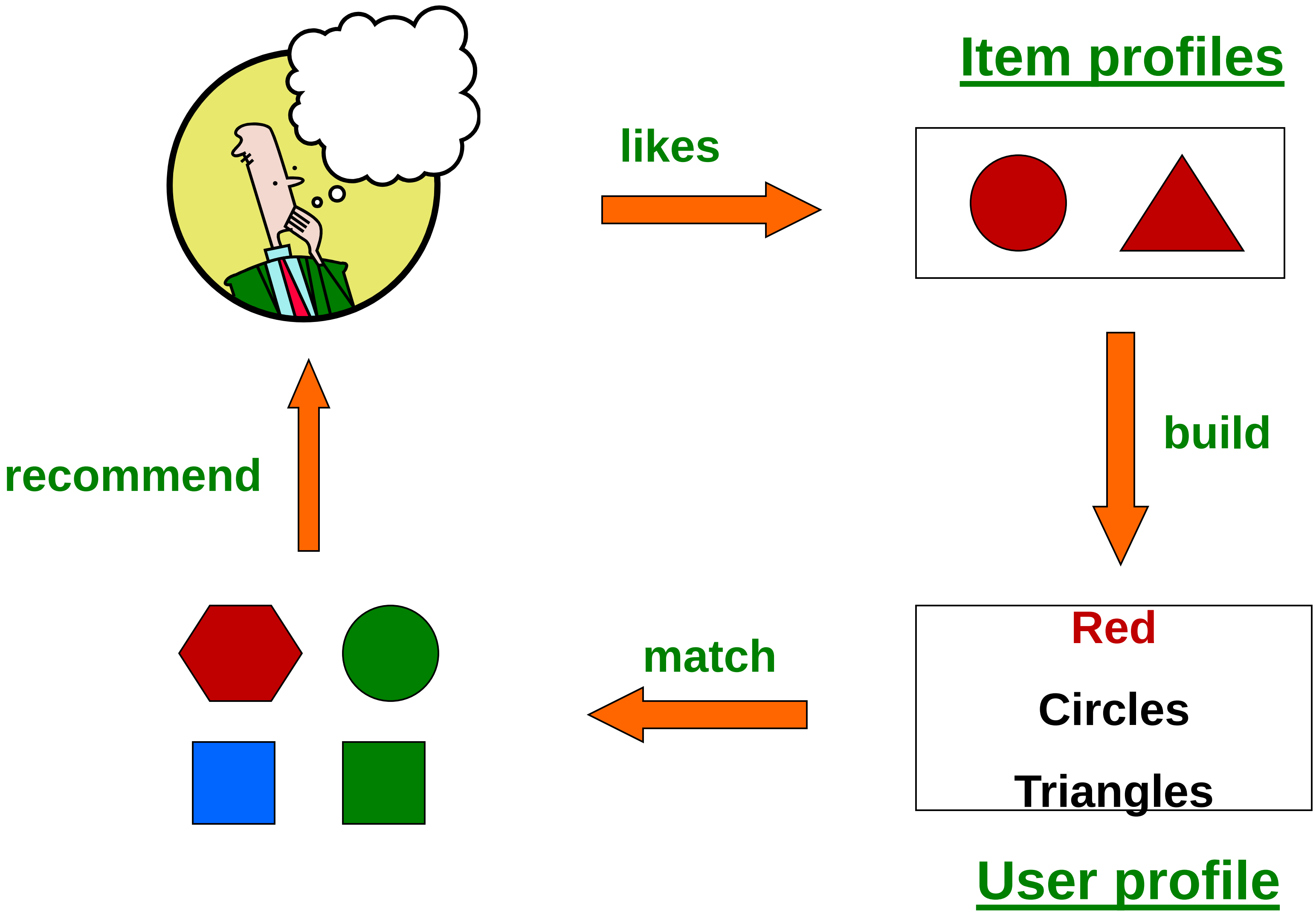
- **Three main approaches for recommendations**
  - Content-based Filtering
  - Collaborative Filtering
  - Latent Factor Models

Content based Recommendations: Overview

- **Key Idea:** Recommend items to customer ***x*** similar to previous items rated highly by ***x***

- **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, title, actor, director,…
  - **Text:** Set of "important" words in document

- **How to pick important features?**
  - Usual heuristic from text mining is **TF-IDF** (Term frequency * Inverse Doc Frequency)

Aside: TF-IDF

f_{ij} = frequency of term (feature) ***i*** in doc (item) ***j***

$$TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}}$$

Note: we normalize TF to discount for "longer" documents

n_i = number of docs that mention term ***i***
N = total number of docs

$$IDF_i = \log \frac{N}{n_i}$$

TF-IDF score: $w_{ij} = TF_{ij} \times IDF_i$

Doc profile = set of words with highest **TF-IDF** scores, together with their scores

User Profiles and Prediction

- **User profile possibilities:**
  - Weighted average of rated item profiles
  - **Variation:** weight by difference from average rating for item
  - …
- **Prediction heuristic:**
  - Given user profile $\boldsymbol{x}$ and item profile $\boldsymbol{i}$, estimate

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

Content based Recommendations: 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

Content based Recommendations: Cons

- **Finding the appropriate features is hard**
  - E.g., images, movies, music (deep learning for automatically extract features)
- **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

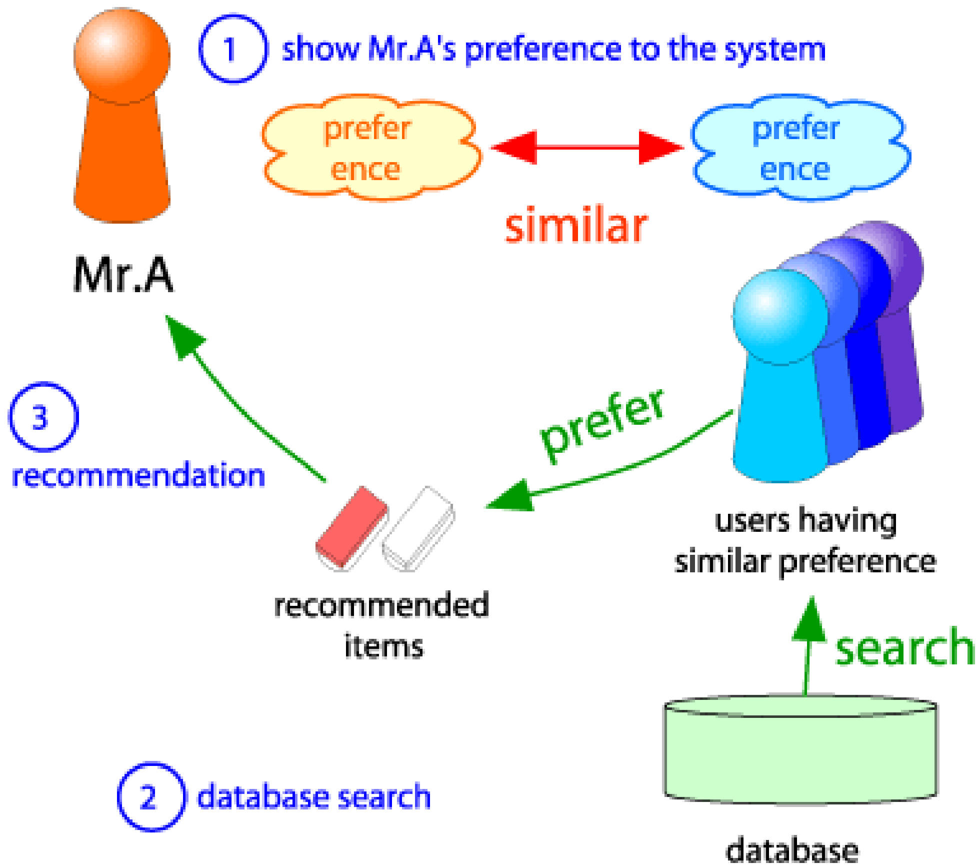
Recommendation Algorithms

- **Three main approaches for recommendations**
  - Content-based Filtering
  - Collaborative Filtering
  - Latent Factor Models

Collaborative Filtering: Overview

- **Key Idea:** Harnessing quality judgments of other users

- **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***



Finding ``Similar'' Users

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

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

- Let $\boldsymbol{r_x}$ be the vector of user $\boldsymbol{x}$'s ratings
- **Jaccard similarity measure**
  - **Problem:** Ignores the value of the rating
- **Cosine similarity measure**
  - sim($\boldsymbol{x}$, $\boldsymbol{y}$) = cos($\boldsymbol{r_x}$, $\boldsymbol{r_y}$) = $\frac{r_x \cdot r_y}{||r_x|| \cdot ||r_y||}$
  - **Problem:** Treats missing ratings as "negative"
- **Pearson correlation coefficient**
  - $\boldsymbol{S_{xy}}$ = items rated by both users $\boldsymbol{x}$ and $\boldsymbol{y}$

Finding ``Similar'' Users

- Let $\boldsymbol{r_x}$ be the vector of user $\boldsymbol{x}$'s ratings

$$r_x = [*, \_, \_, *, ***]$$

$$r_y = [*, \_, **, **, \_]$$

- **Jaccard similarity measure**
  - **Problem:** Ignores the value of the rating

r_x, r_y as sets:

r_x = {1, 4, 5}

r_y = {1, 3, 4}

- **Cosine similarity measure**
  - sim($\boldsymbol{x}$, $\boldsymbol{y}$) = $\boxed{\cos(r_x, r_y) = \frac{r_x \cdot r_y}{||r_x|| \cdot ||r_y||}}$
  - **Problem:** Treats missing ratings as "negative"

r_x, r_y as points:

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

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

Finding ``Similar'' Users

- Let r_x be the vector of user x's ratings

$$r_x = [*, \_, \_, *, ***]$$

$$r_y = [*, \_, **, **, \_]$$

- **Pearson correlation coefficient**
  - S_{xy} = items rated by both users x and y

$$sim(x,y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})(r_{ys} - \overline{r_y})}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \overline{r_y})^2}}$$

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

Similarity Metric: Examples

	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) > sim(A, C)

0.092 **>** -0.559

From Similarity Measure to Predictions: User-User Collaborative Filtering

- 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 *i* of *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?

Item-Item Collaborative Filtering

- 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 *u* on item *j*

N(i;x)… set of items rated by *x* similar to *i*

Item-Item Collaborative Filtering (|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 Collaborative Filtering (|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 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

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

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

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

Collaborative Filtering: Common Practice

- 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}

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

$\boldsymbol{\mu}$ = overall mean movie rating
$\boldsymbol{b_x}$ = rating deviation of user $\boldsymbol{x}$
= *(avg. rating of user* $\boldsymbol{x}$*)* $\boldsymbol{- \mu}$
$\boldsymbol{b_i}$ = rating deviation of movie $\boldsymbol{i}$

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

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

Collaborative Filtering: Pros/Cons

- **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

Collaborative Filtering: Pros/Cons

- **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

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

Recommendations: Key Challenges

- **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 (we are not interested in knowing what you don't like)
- **Evaluating prediction methods**
  - How to measure success/performance of recommendation methods

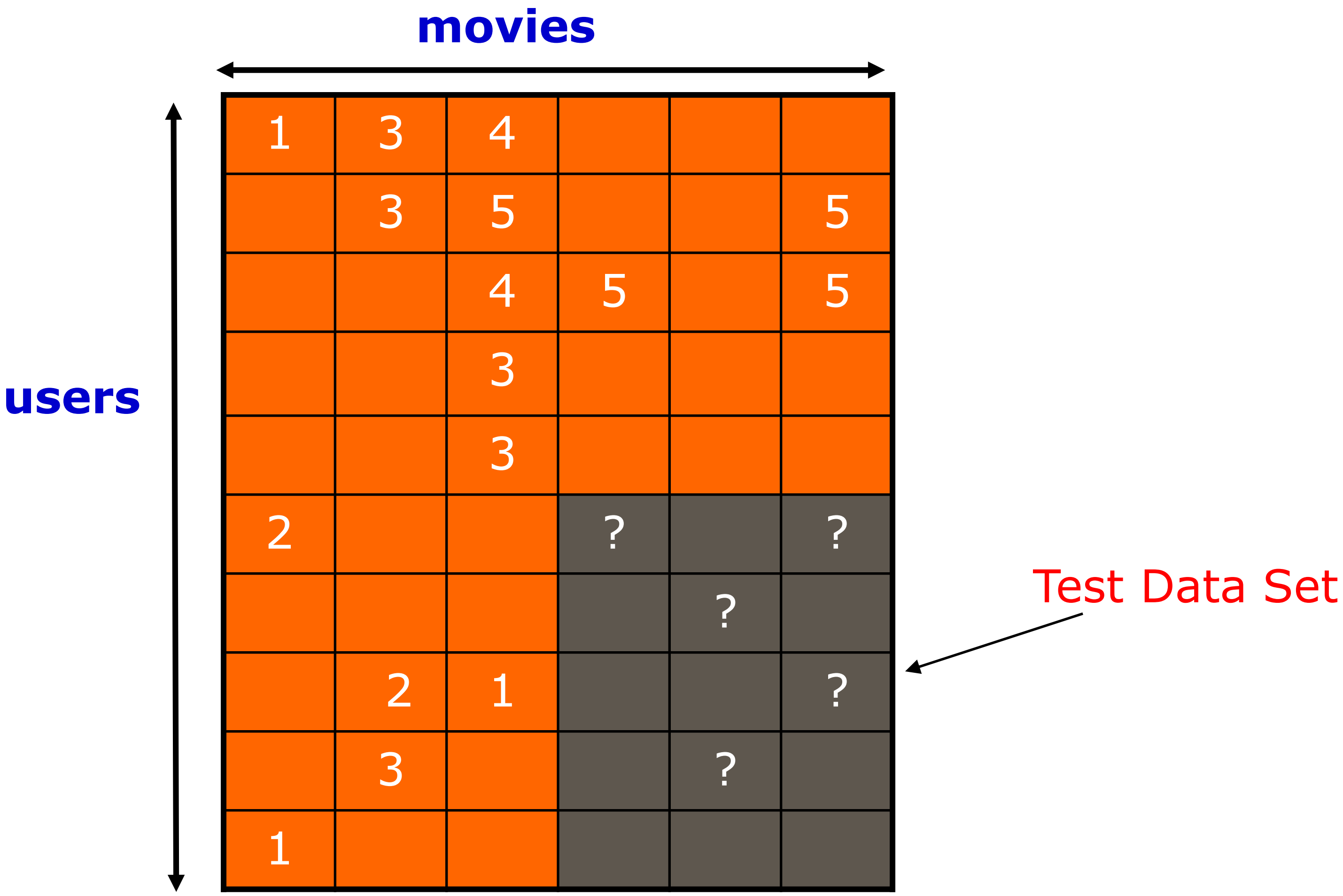
Evaluation

movies

users

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

Evaluation



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

Evaluating Predictions

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

Evaluating Predictions: 0/1 Approach

- **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

Potential Issues 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