BBS654
Data Mining

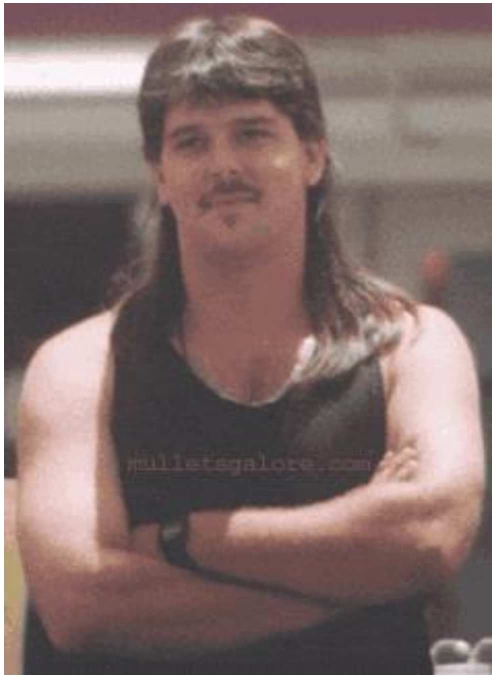
Pinar Duygulu

Slides are adapted from

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

Mustafa Ozdal

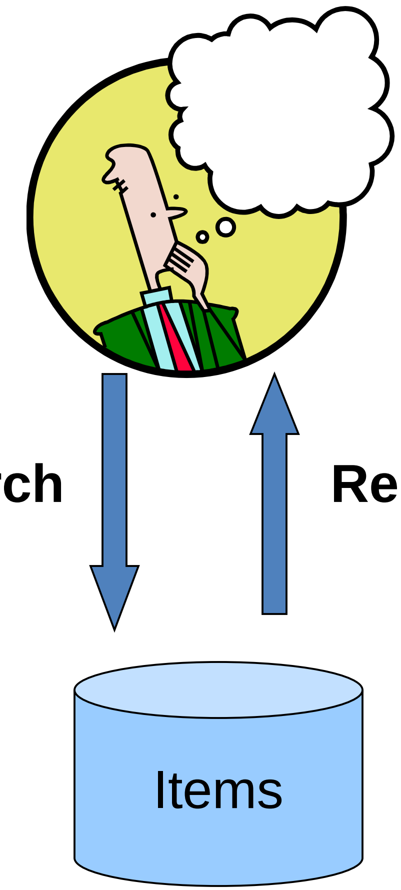
Example: Recommender Systems



- **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



Search

Recommendations

Products, web sites, blogs, news items, …

Examples:

amazon.com

PANDORA

StumbleUpon

del.icio.us

NETFLIX

movielens
helping you find the *right* movies

last.fm
the social music revolution

Google News

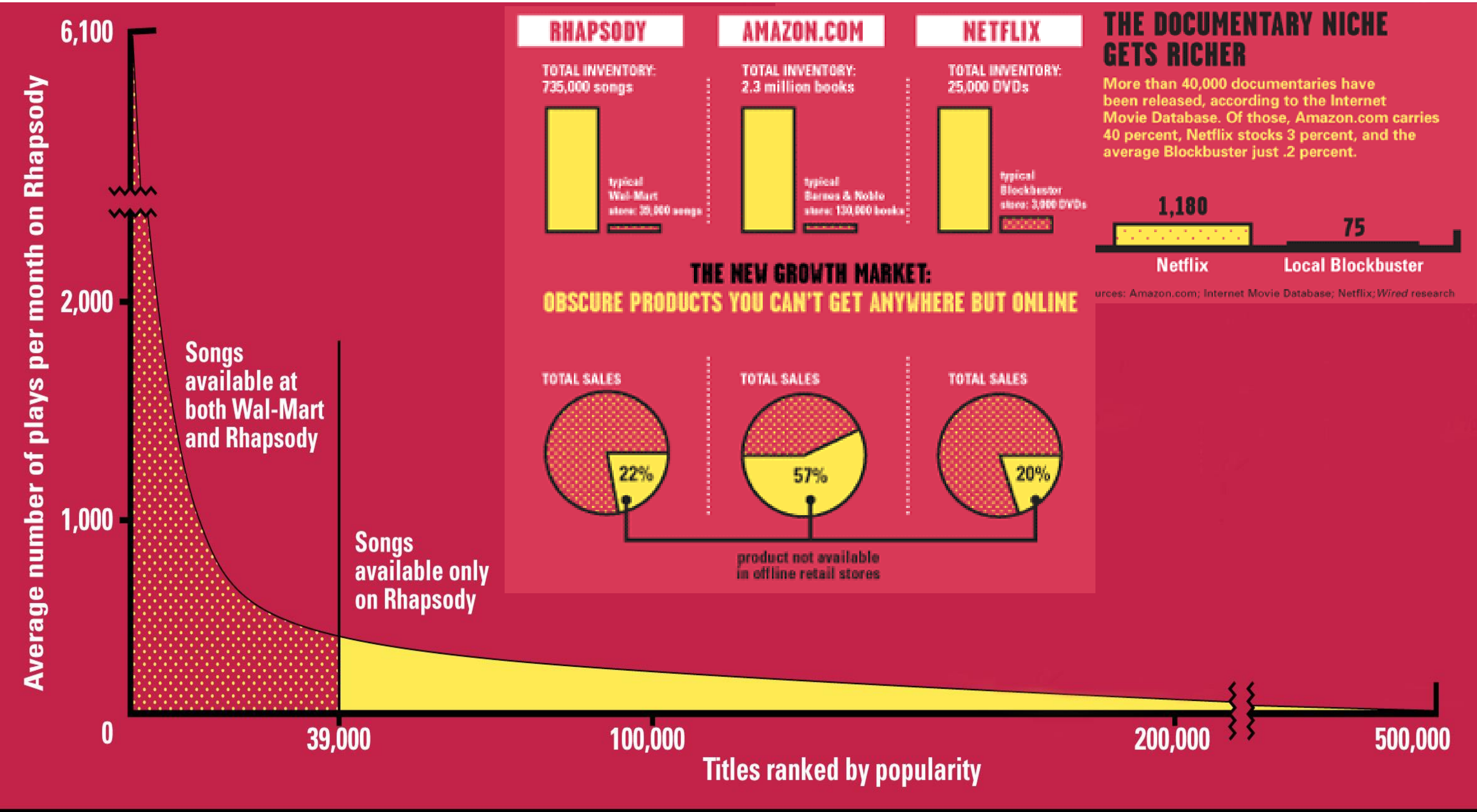
YouTube

XBOX LIVE

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

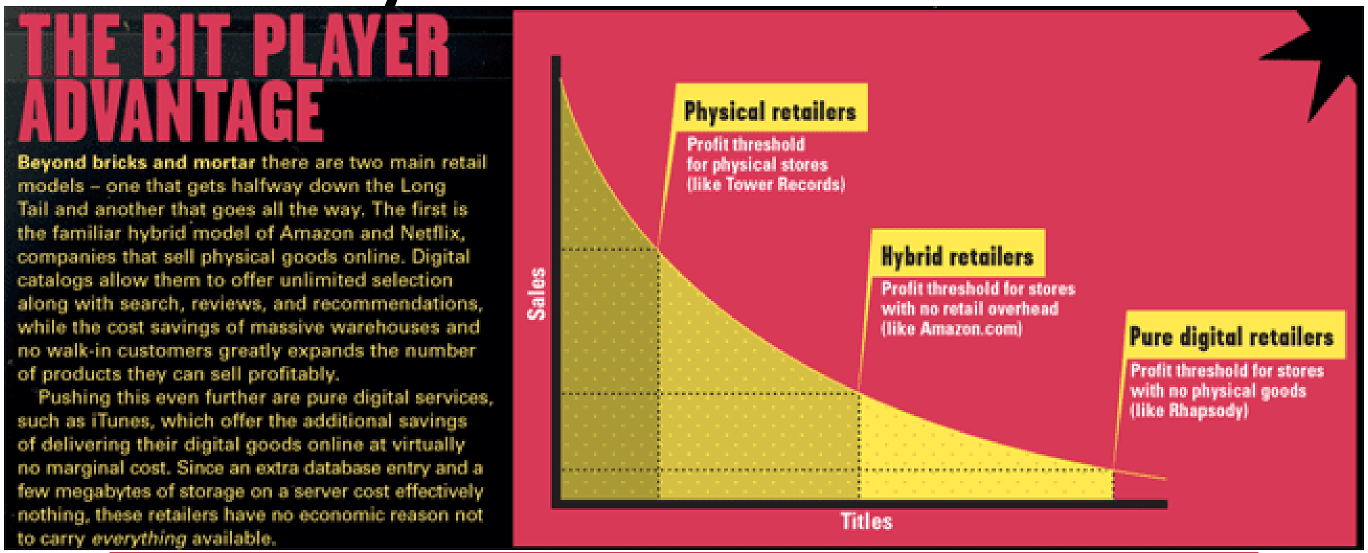
Sidenote: The Long Tail



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

Source: Chris Anderson (2004)

Physical vs. Online



Read http://www.wired.com/wired/archive/12.10/tail.html to learn more!

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, …

Formal Model

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

- **Utility function** $\boldsymbol{u}: \boldsymbol{X} \times \boldsymbol{S} \rightarrow \boldsymbol{R}$
  - ***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 high 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 factor based

This lecture

Content-based
Recommender Systems

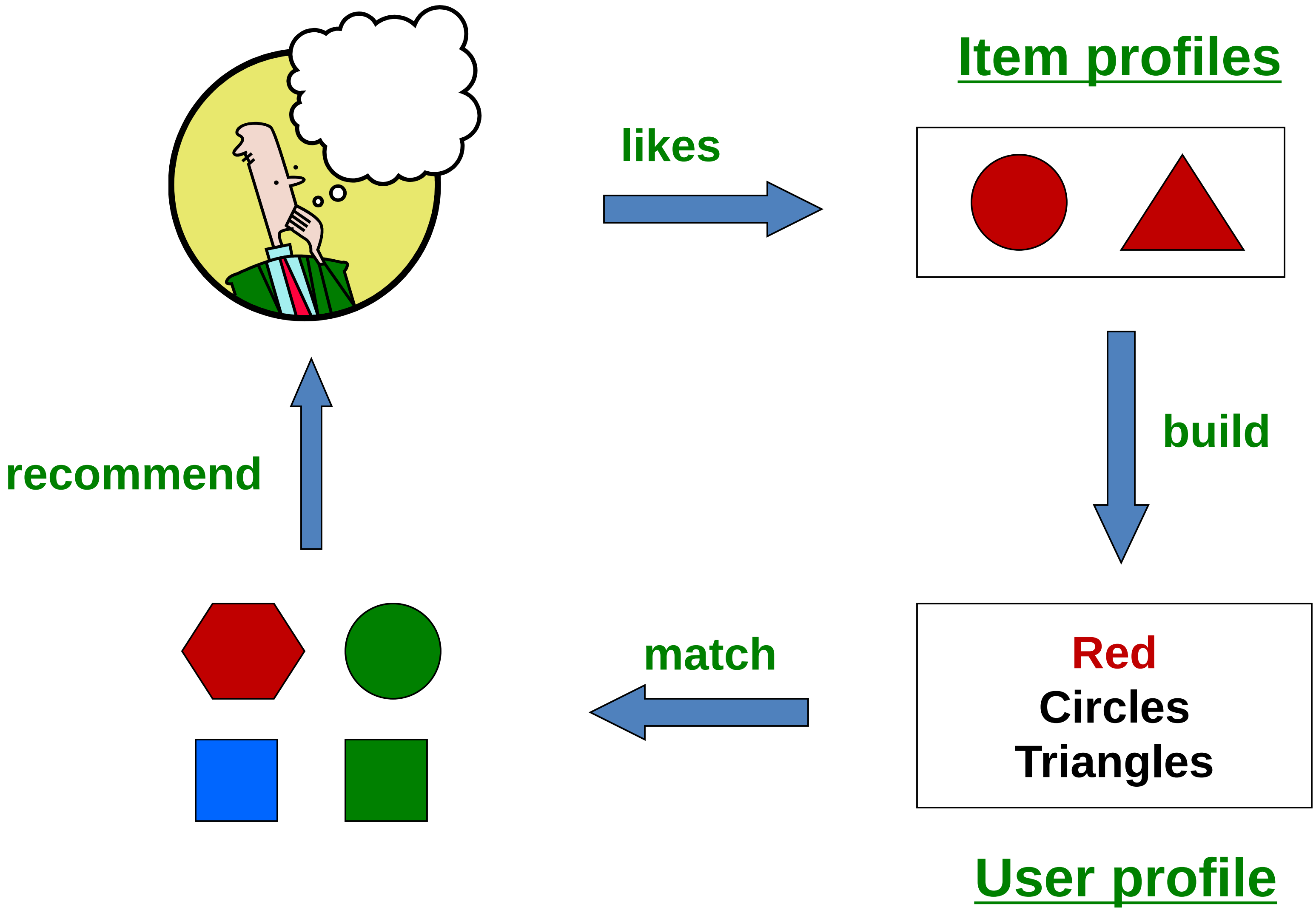
Content-based Recommendations

- **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, 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)
    - **Term … Feature**
    - **Document … Item**

Sidenote: 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

Two Types of Document Similarity

- In the LSH lecture: Lexical similarity
  - Large identical sequences of characters

- For recommendation systems: Content similarity
  - Occurrences of common important words
  - TF-IDF score: If an uncommon word appears more frequently in two documents, it contributes to similarity.

- Similar techniques (e.g. MinHashing and LSH) are still applicable.

Representing Item Profiles

- A vector entry for each feature
  - Boolean features

    e.g. One bool feature for every actor, director, genre, etc.
  - Numeric features

    e.g. Budget of a movie, TF-IDF for a document, etc.

	Spielberg	Scorsese	Tarantino	Lynch	Budget
Jurassic Park	1	0	0	0	63M
Departed	0	1	0	0	90M
Eraserhead	0	0	0	1	20K
Twin Peaks	0	0	0	1	10M

- We may need weighting terms for normalization of features

User Profiles – Option 1

- *<u>Option 1</u>*: Weighted average of rated item profiles

Utility matrix (ratings 1-5)

	Jurassic Park	Minority Report	Schindler's List	Departed	Aviator	Eraser head	Twin Peaks
User 1	4		5			1	1
User 2	2	3			1	5	4
User 3		5	4	5	5		3

User profile(ratings 1-5)

	Spielberg	Scorcese	Lynch
User 1	4.5	0	1
User 2	2.5	1	4.5
User 3	4.5	5	3

Missing scores similar to bad scores

User Profiles – Option 2 (Better)

- ***Option 2***: Subtract average values from ratings first

Utility matrix (ratings 1-5)

	Jurassic Park	Minority Report	Schindler's List	Departed	Aviator	Eraser head	Twin Peaks	Avg
User 1	4		5	0		1	1	**2.75**
User 2	2	3			1	5	4	**3**
User 3		5	4	5	5		3	**4.4**

User Profiles – Option 2 (Better)

- *Option 2*: Subtract average values from ratings first

Utility matrix (ratings 1-5)

	Jurassic Park	Minority Report	Schindler's List	Departed	Aviator	Eraser head	Twin Peaks	Avg
User 1	1.25		2.25			-1.75	-1.75	**2.75**
User 2	-1	0			-2	3	1	**3**
User 3		0.6	-0.4	0.6	0.6		-1.4	**4.4**

User profile

	Spielberg	Scorcese	Lynch
User 1	1.75	0	-1.75
User 2	-0.5	-2	2
User 3	-0.1	0.6	-1.4

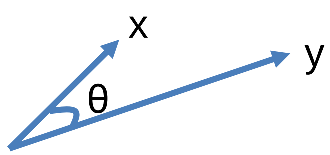
Prediction Heuristic

- Given:
  - A feature vector for user U
  - A feature vector for movie M
- Predict user U's rating for movie M
- Which distance metric to use?

- Cosine distance is a good candidate
  - Works on weighted vectors
  - Only directions are important, not the magnitude
    - The magnitudes of vectors may be very different in movies and users

Reminder: Cosine Distance

- Consider x and y represented as vectors in an n-dimensional space

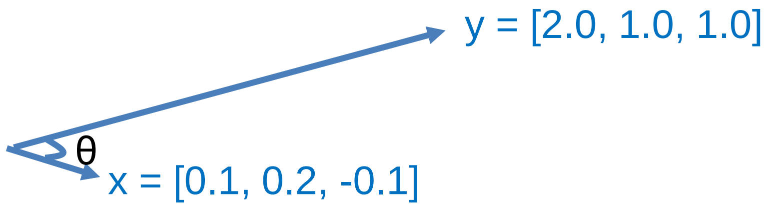


$$\cos(\theta) = \frac{x.y}{||x||.||y||}$$

- The cosine distance is defined as the θ value
  - Or, cosine similarity is defined as cos(θ)

- Only direction of vectors considered, not the magnitudes
- Useful when we are dealing with vector spaces

Reminder: Cosine Distance - Example



$$\cos(\theta) = \frac{x.y}{||x||.||y||} = \frac{0.2 + 0.2 - 0.1}{\sqrt{0.01 + 0.04 + 0.01}\,.\sqrt{4 + 1 + 1}}$$

$$= \frac{0.3}{\sqrt{0.36}} = 0.5 \Rightarrow \theta = 60^0$$

Note: The distance is independent of vector magnitudes

Prediction Example

Predict the rating of user U for movies 1, 2, and 3

	Actor 1	Actor 2	Actor 3	Actor 4
User U	-0.6	0.6	-1.5	2.0
Movie 1	1	1	0	0
Movie 2	1	0	1	0
Movie 3	0	1	0	1

User and movie feature vectors

Prediction Example

Predict the rating of user U for movies 1, 2, and 3

	Actor 1	Actor 2	Actor 3	Actor 4	Vector Magn.
User U	-0.6	0.6	-1.5	2.0	2.6
Movie 1	1	1	0	0	1.4
Movie 2	1	0	1	0	1.4
Movie 3	0	1	0	1	1.4

Prediction Example

Predict the rating of user U for movies 1, 2, and 3

	Actor 1	Actor 2	Actor 3	Actor 4	Vector Magn.	Cosine Sim
User U	-0.6	0.6	-1.5	2.0	2.6	
Movie 1	1	1	0	0	1.4	0
Movie 2	1	0	1	0	1.4	-0.6
Movie 3	0	1	0	1	1.4	0.7

Prediction Example

Predict the rating of user U for movies 1, 2, and 3

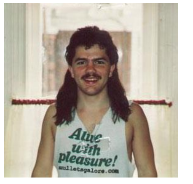
	Actor 1	Actor 2	Actor 3	Actor 4	Vector Magn.	Cosine Sim	Cosine Dist
User U	-0.6	0.6	-1.5	2.0	2.6		
Movie 1	1	1	0	0	1.4	0	90^0
Movie 2	1	0	1	0	1.4	-0.6	124^0
Movie 3	0	1	0	1	1.4	0.7	46^0

Prediction Example

Predict the rating of user U for movies 1, 2, and 3

	Actor 1	Actor 2	Actor 3	Actor 4	Vector Magn.	Cosine Sim	Cosine Dist	Interpretation
User U	-0.6	0.6	-1.5	2.0	2.6			
Movie 1	1	1	0	0	1.4	0	90^0	Neither likes nor dislikes
Movie 2	1	0	1	0	1.4	-0.6	124^0	Dislikes
Movie 3	0	1	0	1	1.4	0.7	46^0	Likes

Content-Based Approach: True or False?



Likes Metallica, Sinatra and Bieber

- Need data on other users

  False
- Can handle users with unique tastes

  True – no need to have similarity with other users
- Can handle new items easily

  True – well-defined features for items
- Can handle new users easily

  False – how to construct user-profiles?
- Can provide explanations for the predicted recommendations

  True – know which features contributed to the ratings

Pros: Content-based Approach

- **+: 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: Content-based Approach

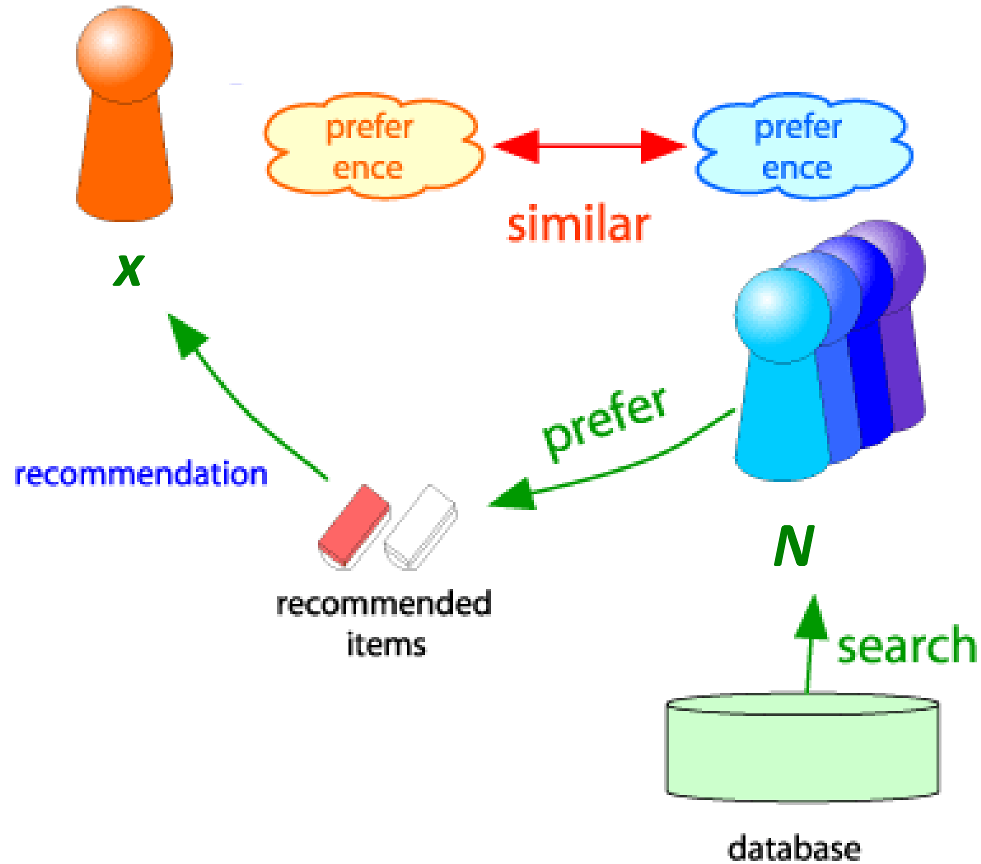
- **–: Finding the appropriate features is hard**
  - E.g., images, movies, music
- **–: Recommendations for new users**
  - **How to build a user profile?**
- **–: Overspecialization**
  - Never recommends items outside user's content profile
  - People might have multiple interests
  - **Unable to exploit quality judgments of other users**
    - e.g. Users who like director X also like director Y
      User U rated X, but doesn't know about Y

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***



Finding "Similar" Users

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

- Let r_x be the vector of user x's ratings
- **Jaccard similarity measure**
  - **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**
  - 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 as sets:
$r_x = \{1, 4, 5\}$
$r_y = \{1, 3, 4\}$

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

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

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

Rating Predictions

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*:**
  - $r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$
  - $r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$
  - Other options?
- **Many other tricks possible…**

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

Rating Predictions

Predict the rating of A for HP2:

	HP1	HP2	HP3	TW	SW1	SW2	SW3	similarity of A
A	4			5	1			
B	5	5	4					0.09
C				2	4	5		-0.56
D		3					3	0

Prediction based on the top 2 neighbors who have also rated HP2

Option 1: $r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$

$r_{A,HP2} = (5+3) / 2 = 4$

Rating Predictions

Predict the rating of A for HP2:

	HP1	HP2	HP3	TW	SW1	SW2	SW3	similarity of A
A	4			5	1			
B	5	$\boxed{5}$	4					$\boxed{0.09}$
C				2	4	5		-0.56
D		$\boxed{3}$					3	$\boxed{0}$

Prediction based on the top 2 neighbors who have also rated HP2

Option 2: $r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$

$r_{A,HP2} = (5 \times 0.09 + 3 \times 0) / \ 0.09 = 5$

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 ***u*** 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**

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:

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

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:

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

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

Before:

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

CF: 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}

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

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

Example

- The global movie rating is $\mu = 2.8$

  i.e. average of all ratings of all users is 2.8
- The average rating of user x is $\mu_x = 3.5$
- Rating deviation of user x is $b_x = \mu_x - \mu = 0.7$

  i.e. this user's avg rating is 0.7 larger than global avg
- The average rating for movie i is $\mu_i = 2.6$
- Rating deviation of movie i is $b_i = \mu_i - \mu = -0.2$

  i.e. this movie's avg rating is 0.2 less than global avg

- Baseline estimate for user x and movie i is

$$b_{xi} = \mu + b_x + b_i = 2.8 + 0.7 - 0.2 = 3.3$$

Example (cont'd)

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

- Items **k a**nd **m**: The most similar items to i that are also rated by x
  Assume both have similarity values of **0.4**
- Assume:
  $r_{xk} = 2$ and $b_{xk} = 3.2$ → deviation of -1.2
  $r_{xm} = 3$ and $b_{xk} = 3.8$ → deviation of -0.8

Example (cont'd)

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

Rating r_{xi} is the baseline rating plus the weighted avg of deviations of the most similar items' ratings:

$$r_{xi} = 3.3 + \frac{0.4 \times (-1.2) + 0.4 \times (-0.8)}{0.4 + 0.4} = 2.3$$

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

Collaborating Filtering: True or False?

- Need data on other users

  True
- Effective for users with unique tastes and esoteric items

  False – relies on similarity between users or items
- Can handle new items easily

  False – cold start problems
- Can handle new users easily

  False – cold start problems
- Can provide explanations for the predicted recommendations

  User-user: False – “because users X, Y, Z also liked it”

  Item-item: True – “because you also liked items i, j, k”

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

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

Item/User Clustering to Reduce Sparsity

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

	HP	TW	SW
A	4	5	1
B	4.67		
C		2	4.5
D	3		3

REMARKS & PRACTICAL TIPS

- Evaluation
- Error metrics
- Complexity / Speed

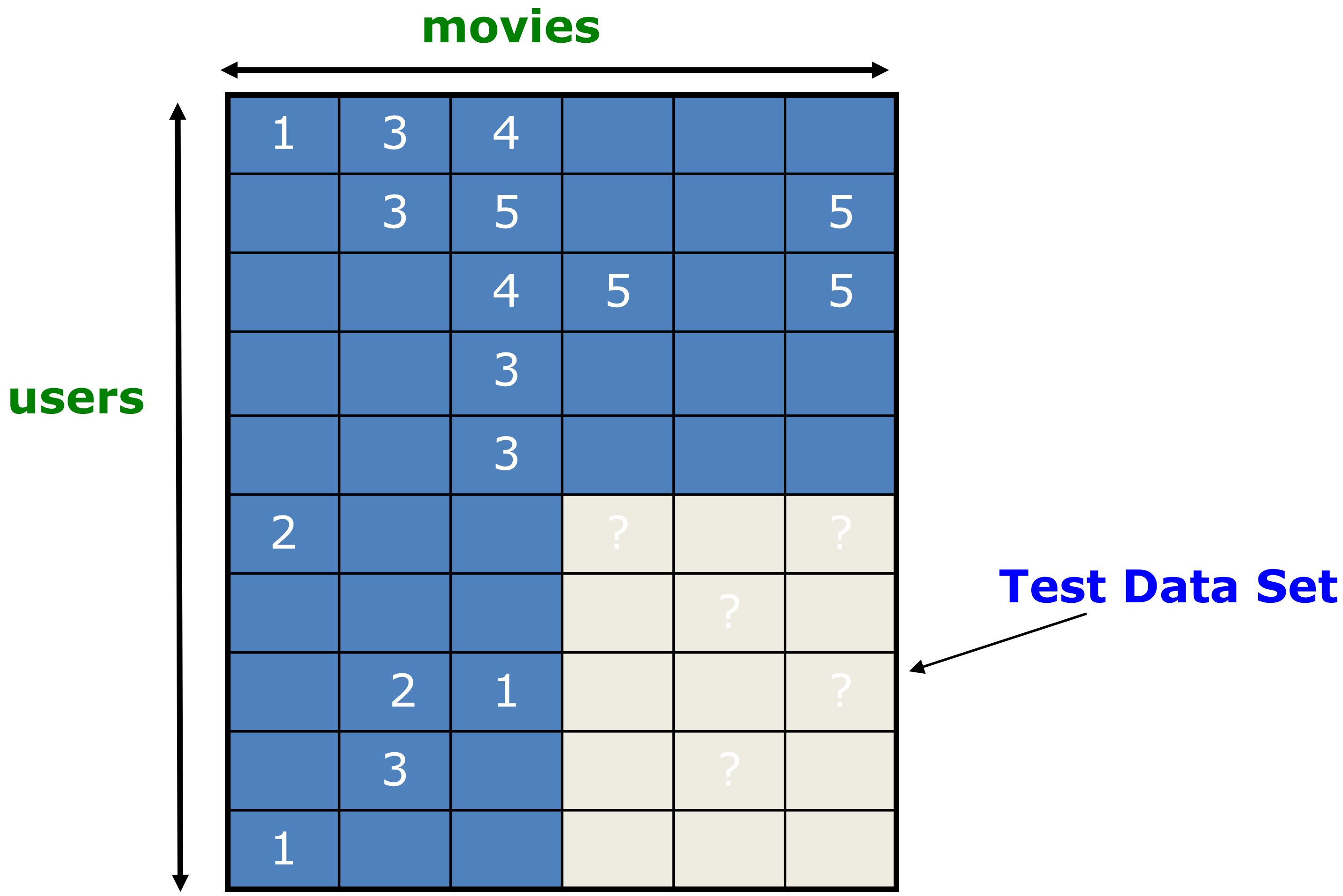
Evaluation

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)

$$\sqrt{\sum_{xi}\left(r_{xi}-r_{xi}^{*}\right)^{2}}$$

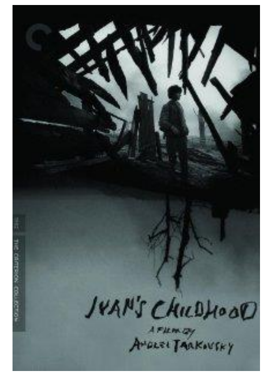
where $\boldsymbol{r_{xi}}$ is predicted, $\boldsymbol{r_{xi}^{*}}$ is the true rating of $\boldsymbol{x}$ on $\boldsymbol{i}$

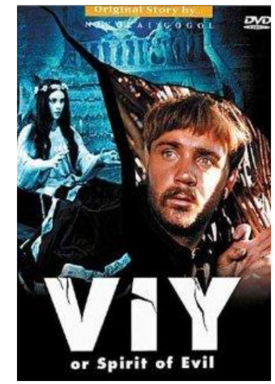
- **Another approach: 0/1 model**
  - **Coverage:**
    - Number of items/users for which system can make predictions
  - **Precision:**
    - Accuracy of predictions
  - **Receiver operating characteristic** (ROC)
    - Tradeoff curve between true positives and false positives

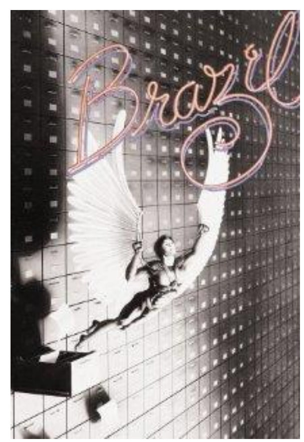
Problems with Error Measures

- **Narrow focus on accuracy sometimes misses the point**
  - Prediction Context
  - Prediction Diversity

Prediction Diversity Problem







Problems with Error Measures

- **In practice, we care only to predict high ratings:**
  - RMSE might penalize a method that does well for high ratings and badly for others
  - Alternative: Precision at top k

Collaborative Filtering: Complexity

- Expensive step is finding ***k*** most similar customers: **O(|X|)**
- **Too expensive to do at runtime**
  - Could pre-compute
- Naïve pre-computation takes time **O(k ·|X|)**
  - X … set of customers
- **We already know how to do this!**
  - Near-neighbor search in high dimensions (**LSH**)
  - Clustering
  - Dimensionality reduction

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