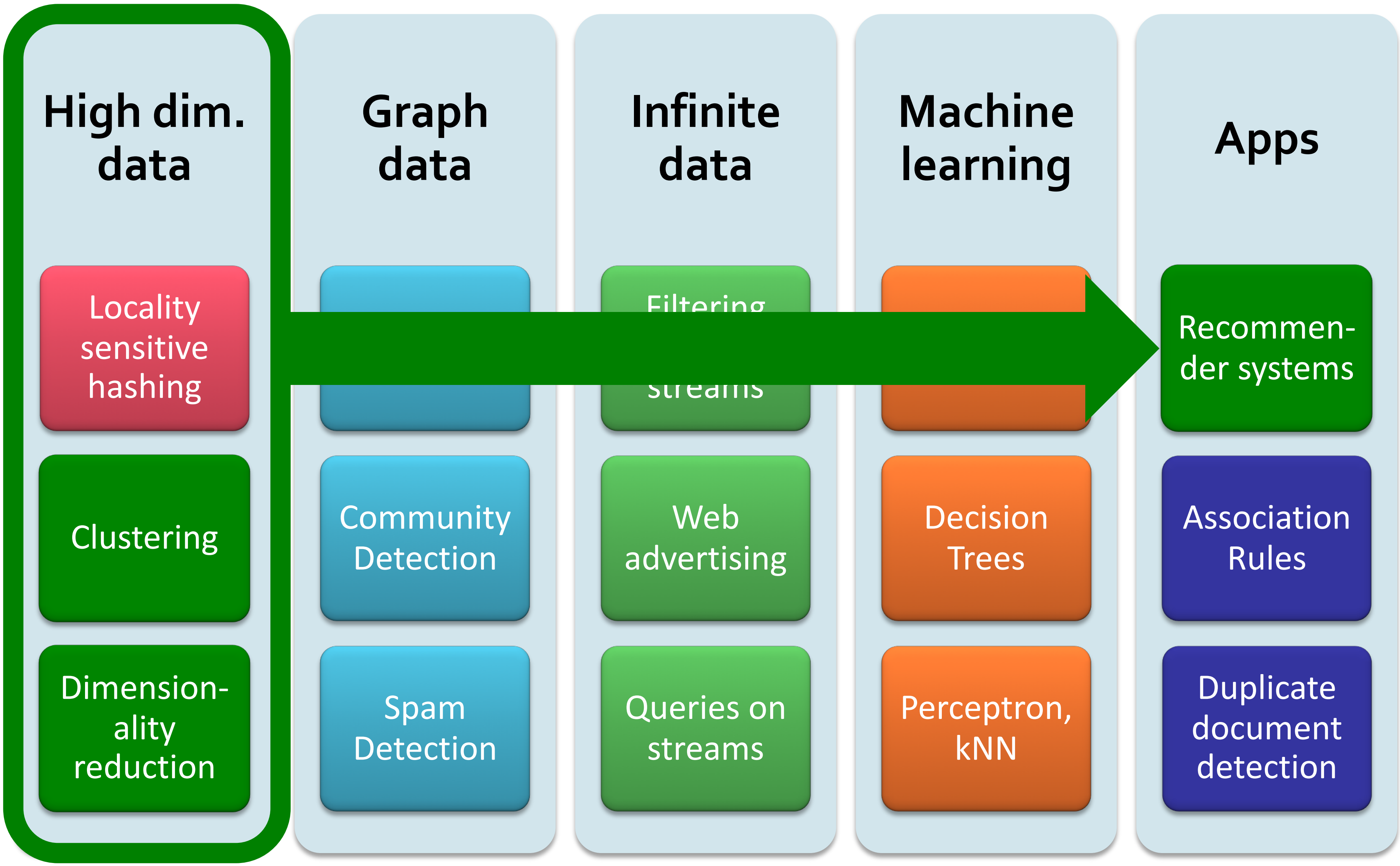
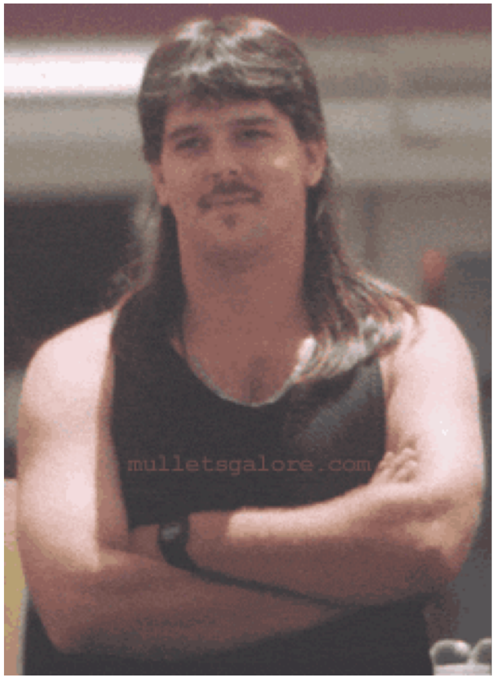
Recommender Systems

High Dimensional Data



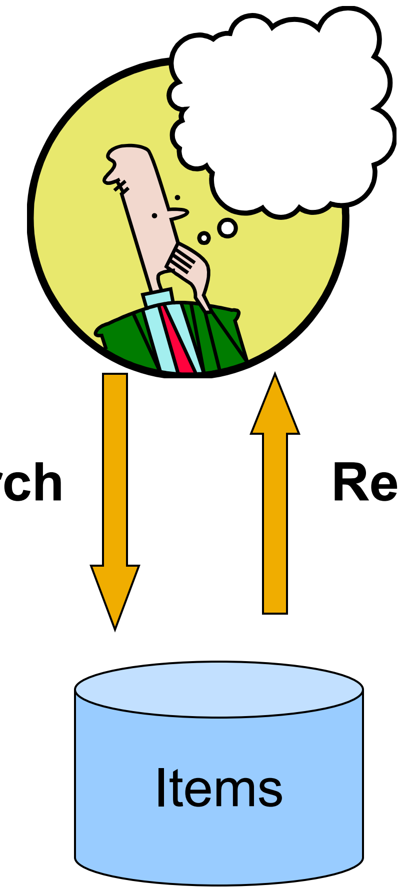
Follow me on LinkedIn for more:
Steve Nouri
https://www.linkedin.com/in/stevenouri/

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



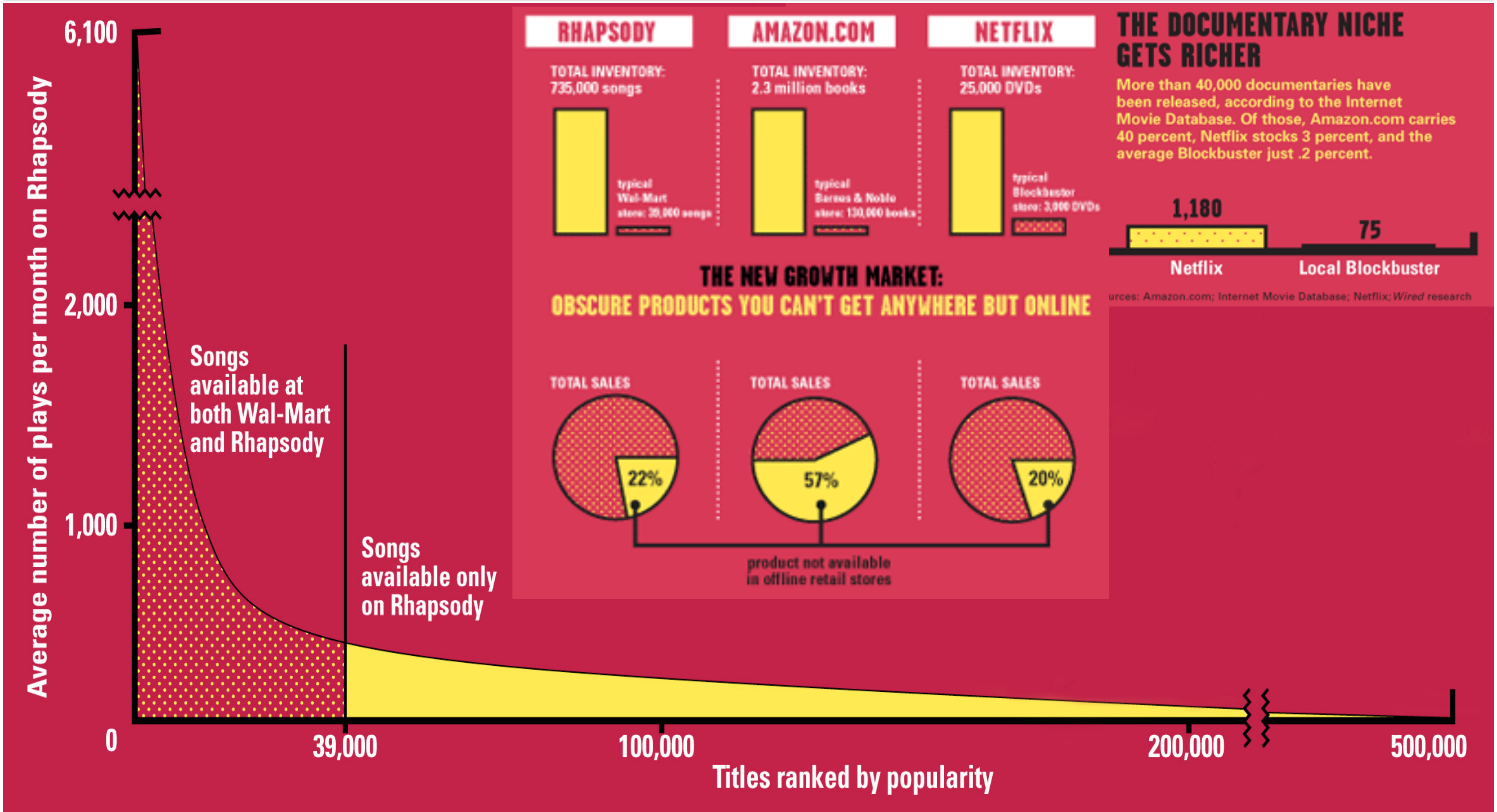
Products, web sites, blogs, news items, …

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
  - **Association rules:** How **Into Thin Air** made **Touching the Void** a bestseller: http://www.wired.com/wired/archive/12.10/tail.html

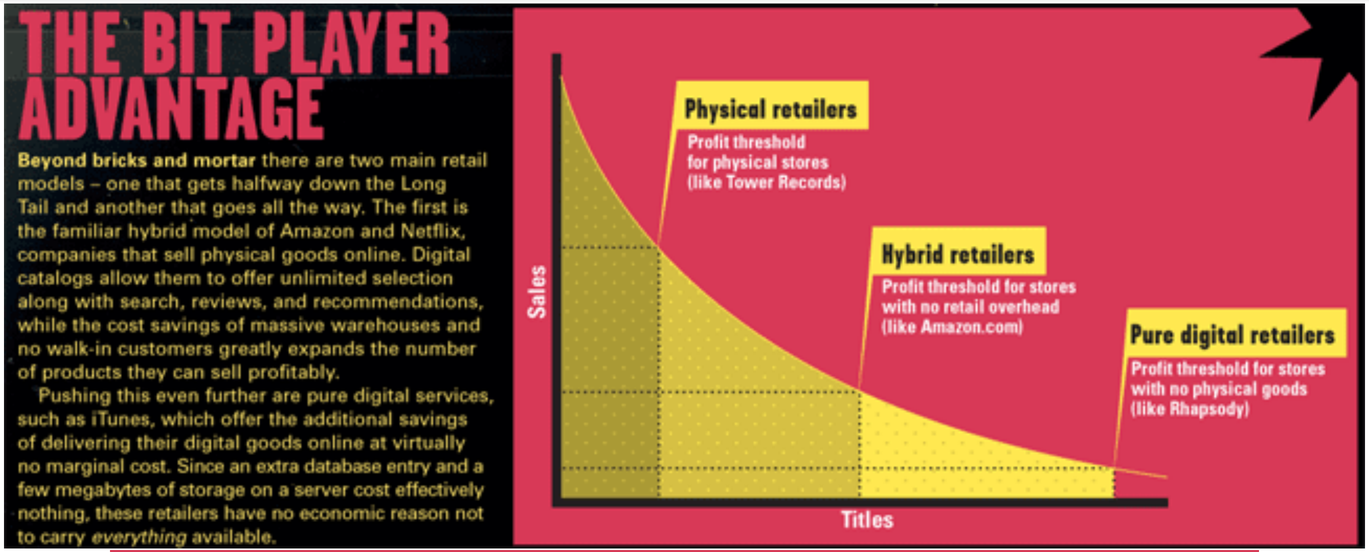
Sidenote: The Long Tail



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

Source: Chris Anderson (2004)

Physical vs. Online



THE BIT PLAYER ADVANTAGE

Beyond bricks and mortar there are two main retail models – one that gets halfway down the Long Tail and another that goes all the way. The first is the familiar hybrid model of Amazon and Netflix, companies that sell physical goods online. Digital catalogs allow them to offer unlimited selection along with search, reviews, and recommendations, while the cost savings of massive warehouses and no walk-in customers greatly expands the number of products they can sell profitably.

Pushing this even further are pure digital services, such as iTunes, which offer the additional savings of delivering their digital goods online at virtually no marginal cost. Since an extra database entry and a few megabytes of storage on a server cost effectively nothing, these retailers have no economic reason not to carry *everything* available.

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, …

Today’s class

Formal Model

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

- **Utility function** u: $X \times S \rightarrow R$
  - R = set of ratings
  - R is a totally ordered set
  - e.g., **1-5** 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) Extrapolating 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 don't like being 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

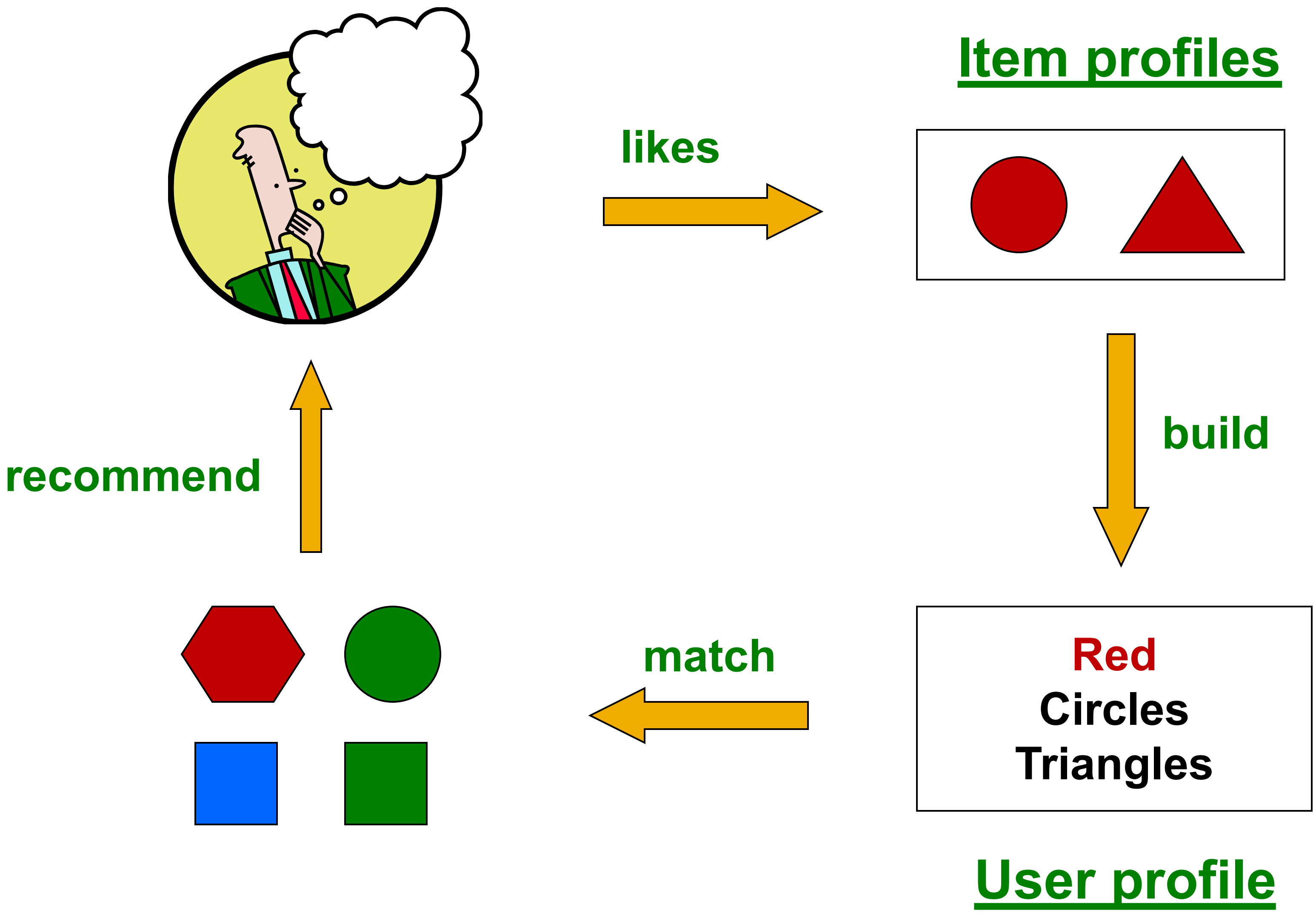
Content-based Recommendations

- **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, 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

User Profiles and Prediction

- **User profile possibilities:**
  - Weighted average of rated item profiles
  - **Variation:** weight by difference from average rating for item
- **Prediction heuristic: Cosine similarity of user and item profiles)**
  - Given user profile ***x*** and item profile ***i***, estimate $u(\boldsymbol{x}, \boldsymbol{i}) = \cos(\boldsymbol{x}, \boldsymbol{i}) = \frac{\boldsymbol{x} \cdot \boldsymbol{i}}{||\boldsymbol{x}|| \cdot ||\boldsymbol{i}||}$
- **How do you quickly find items closest to x?**
  - Job for LSH!

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

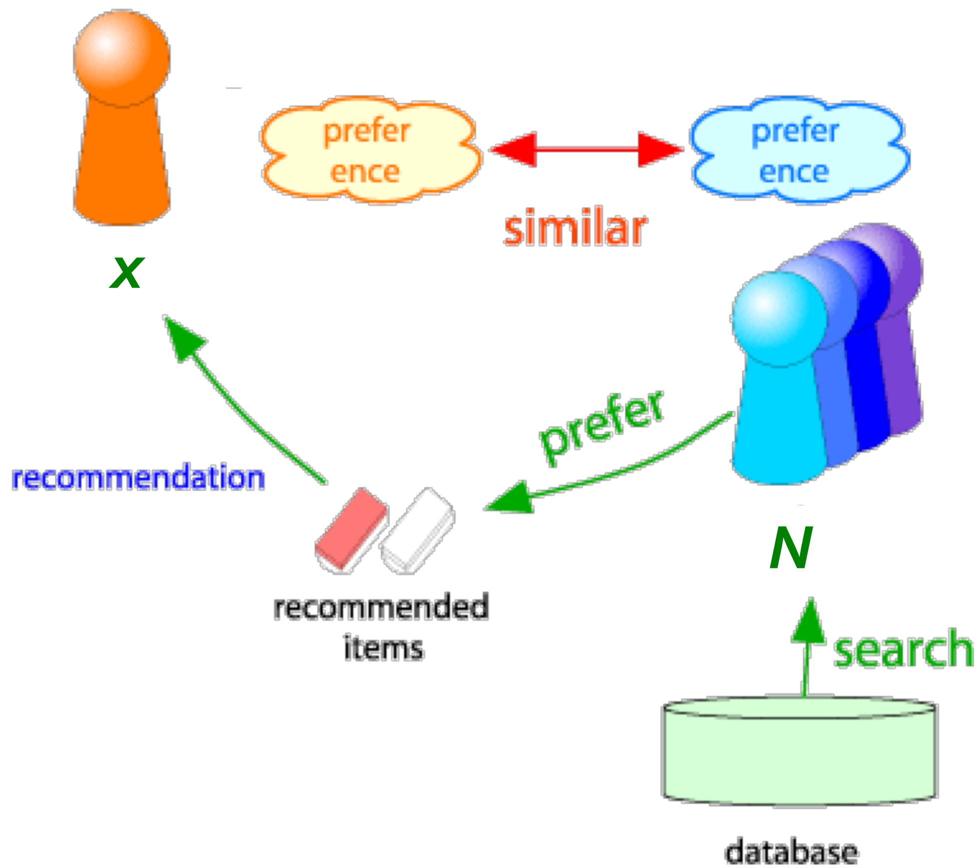
- **–: Finding the appropriate features is hard**
  - E.g., images, movies, music
- **–: Recommendations for new users**
  - **How to build a user profile?**
- **–: Overspecialization**
  - Never recommends items outside user's content profile
  - People might have multiple interests
  - **Unable to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

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 some 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

Cosine sim:

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

	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:** 0.380 **>** 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*:**
  - $r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$
  - Or even better: $r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$

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

- **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 ***x*** on item ***j***
N(i;x)… set items which were rated by ***x*** and 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	$s_{1,m}$
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 dot products between rows

Item-Item CF (|N|=2)

users

movies	1	2	3	4	5	6	7	8	9	10	11	12	$s_{1,m}$
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**

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

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) $-\ \mu$
- **b**[redacted] = rating deviation of movie 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

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

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

movies

users

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

Test Data Set

Evaluating Predictions

- **Compare predictions with known ratings**
  - **Root-mean-square error** (RMSE)
    - $\sqrt{\frac{1}{N}\sum_{xi}\left(r_{xi} - r_{xi}^{*}\right)^{2}}$ where $\boldsymbol{r_{xi}}$ is predicted, $\boldsymbol{r^{*}_{xi}}$ is the true rating of ***x*** on ***i***
      - ***N is the number of points we are making comparisons on***
  - **Precision at top 10**:
    - % 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 the system can make predictions
  - **Precision:**
    - Accuracy of predictions
  - **Receiver operating characteristic** (ROC)
    - Tradeoff curve between false positives and false negatives

Problems 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

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

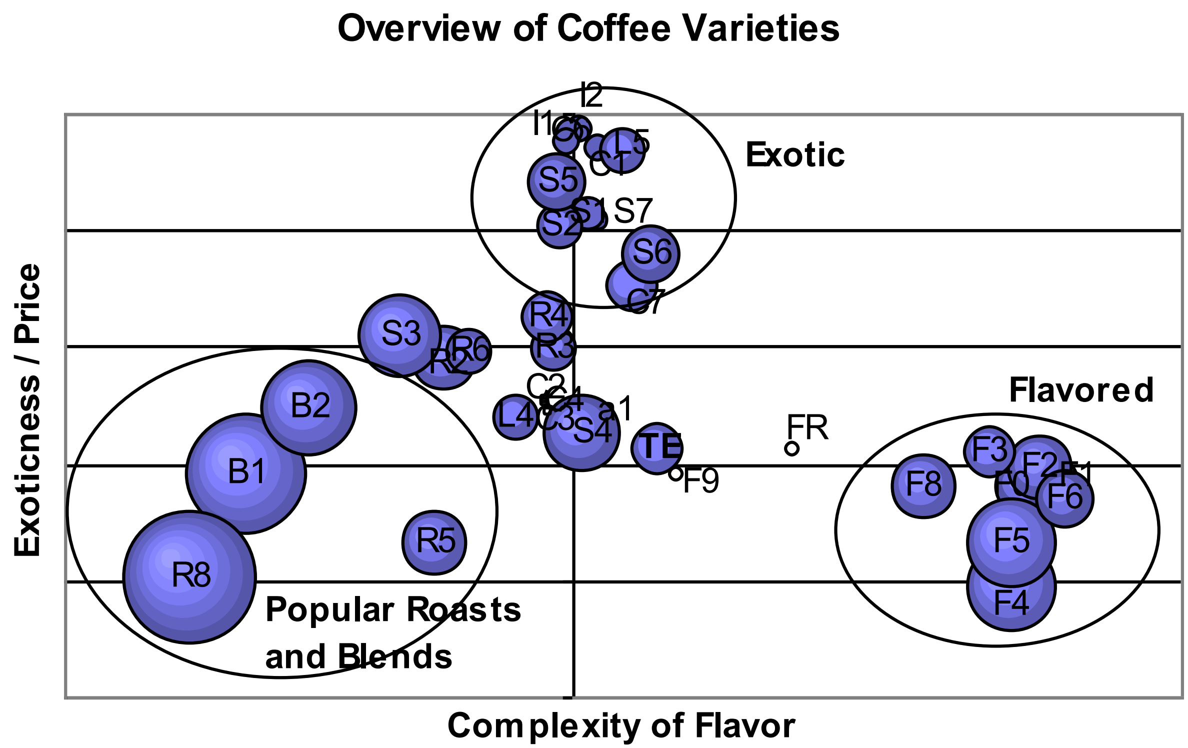
On Thursday:
The Netflix prize and the Latent Factor Models

On Thursday: 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

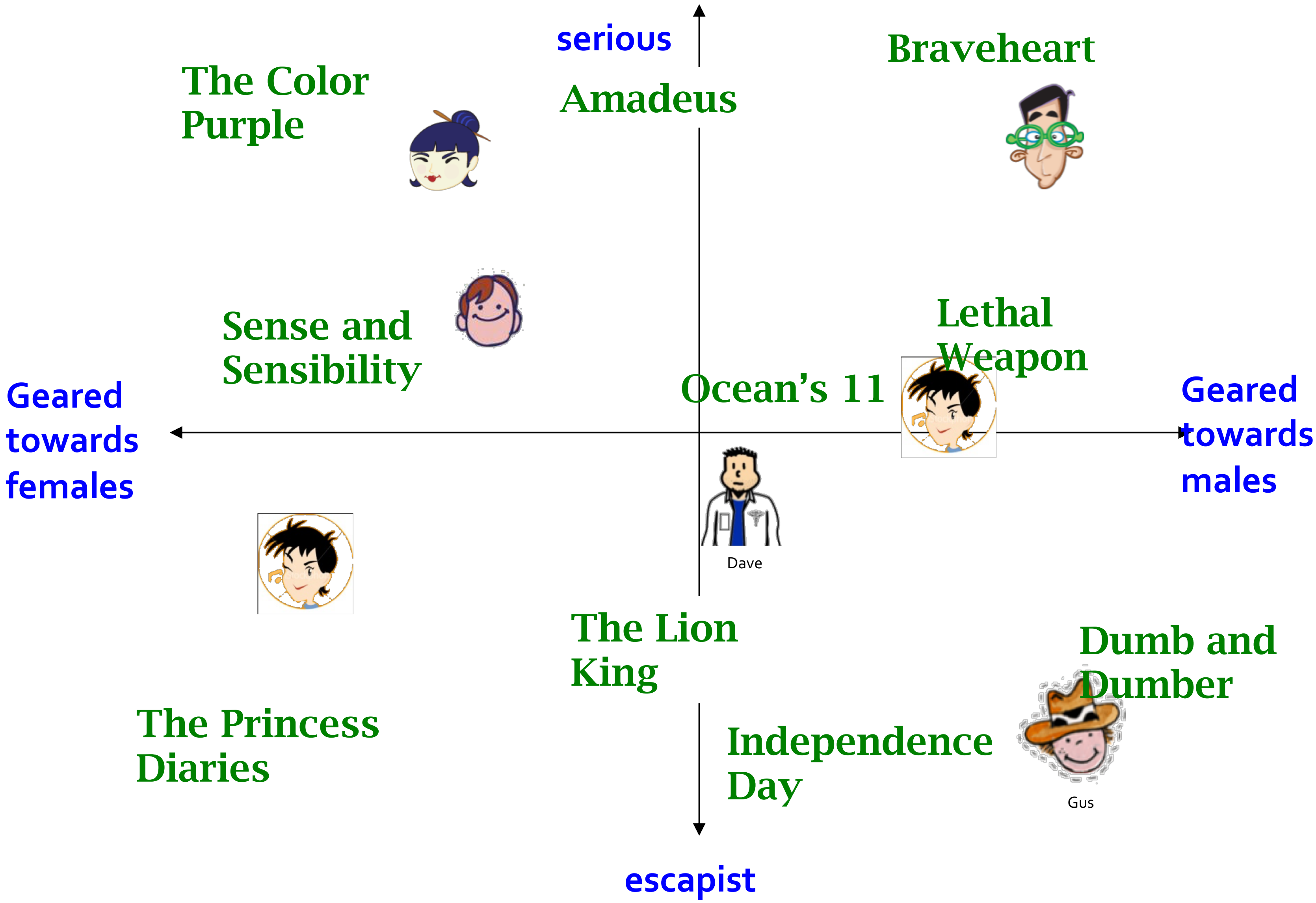
On Thursday: Latent Factor Models

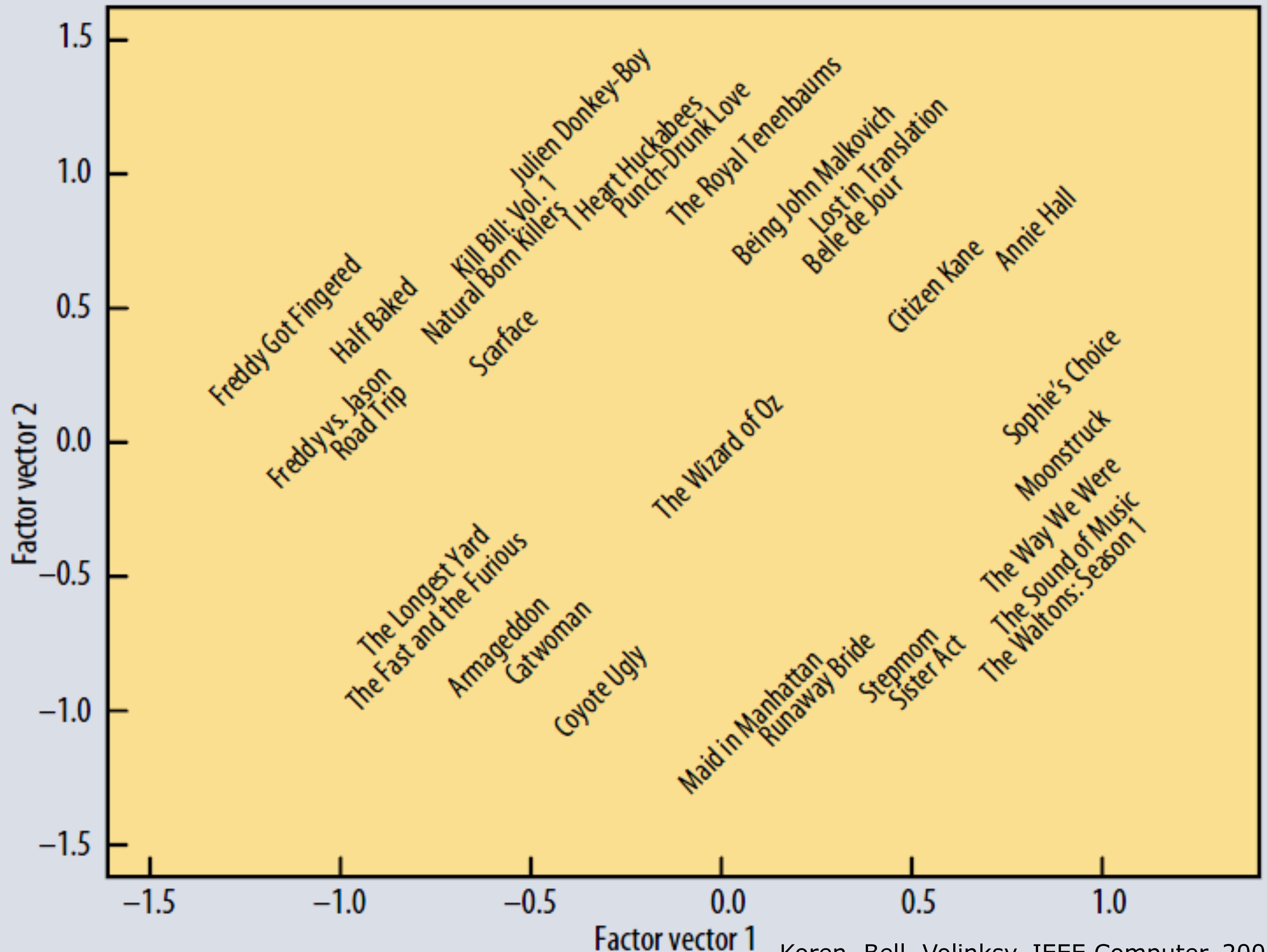
- **Next topic:** **Recommendations via Latent Factor models**



The bubbles above represent products sized by sales volume. Products close to each other are recommended to each other.

Latent Factor Models (i.e., SVD++)



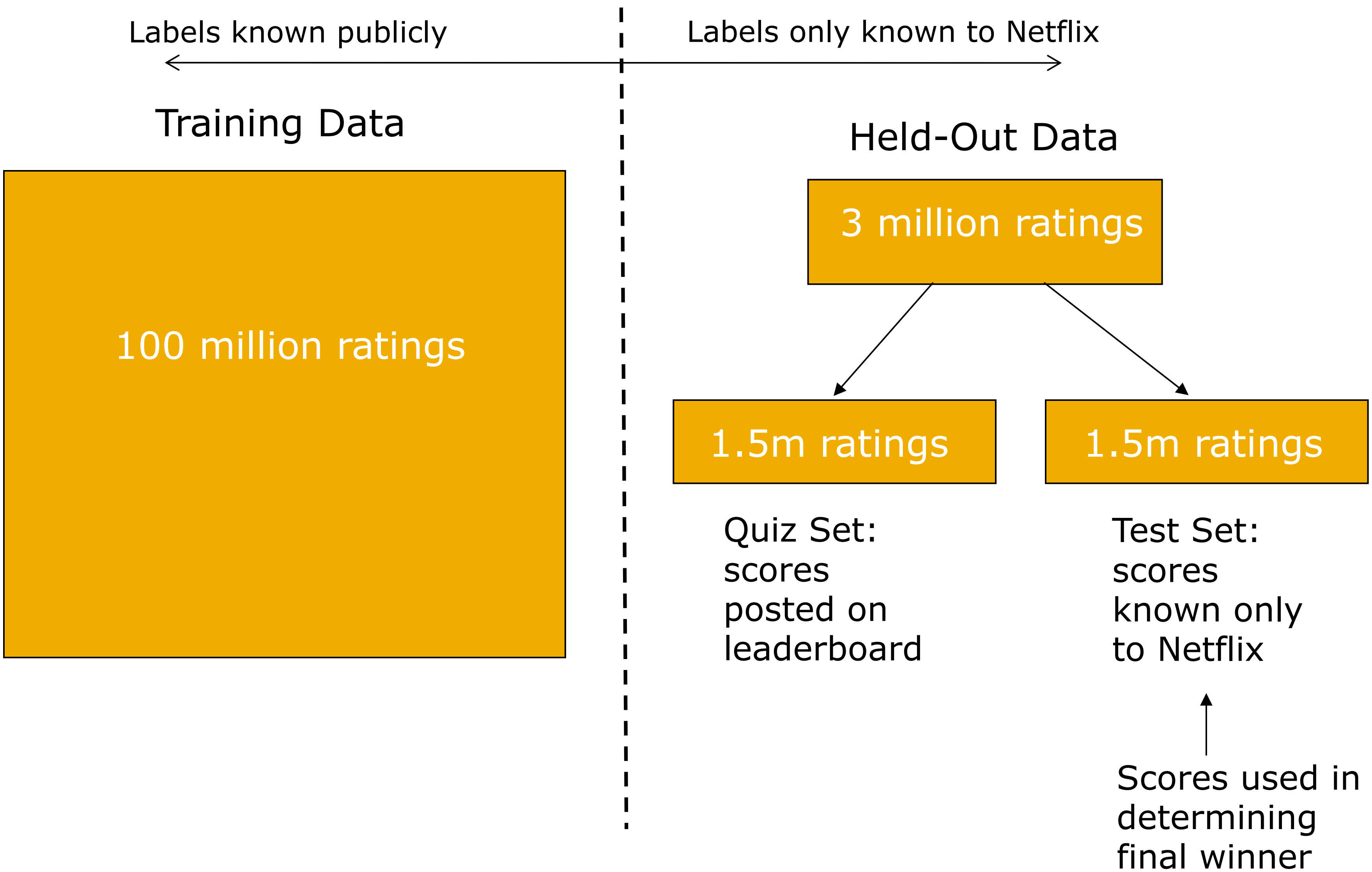


Koren, Bell, Volinksy, IEEE Computer, 2009

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 Square Error (RMSE) $= \sqrt{\frac{1}{|R|} \sum_{(i,x)\in R} (\hat{r}_{xi} - r_{xi})^2}$
  - **Netflix's system RMSE: 0.9514**
- **Competition**
  - 2,700+ teams
  - **\$1 million** prize for 10% improvement on Netflix

Competition Structure



The Netflix Utility Matrix *R*

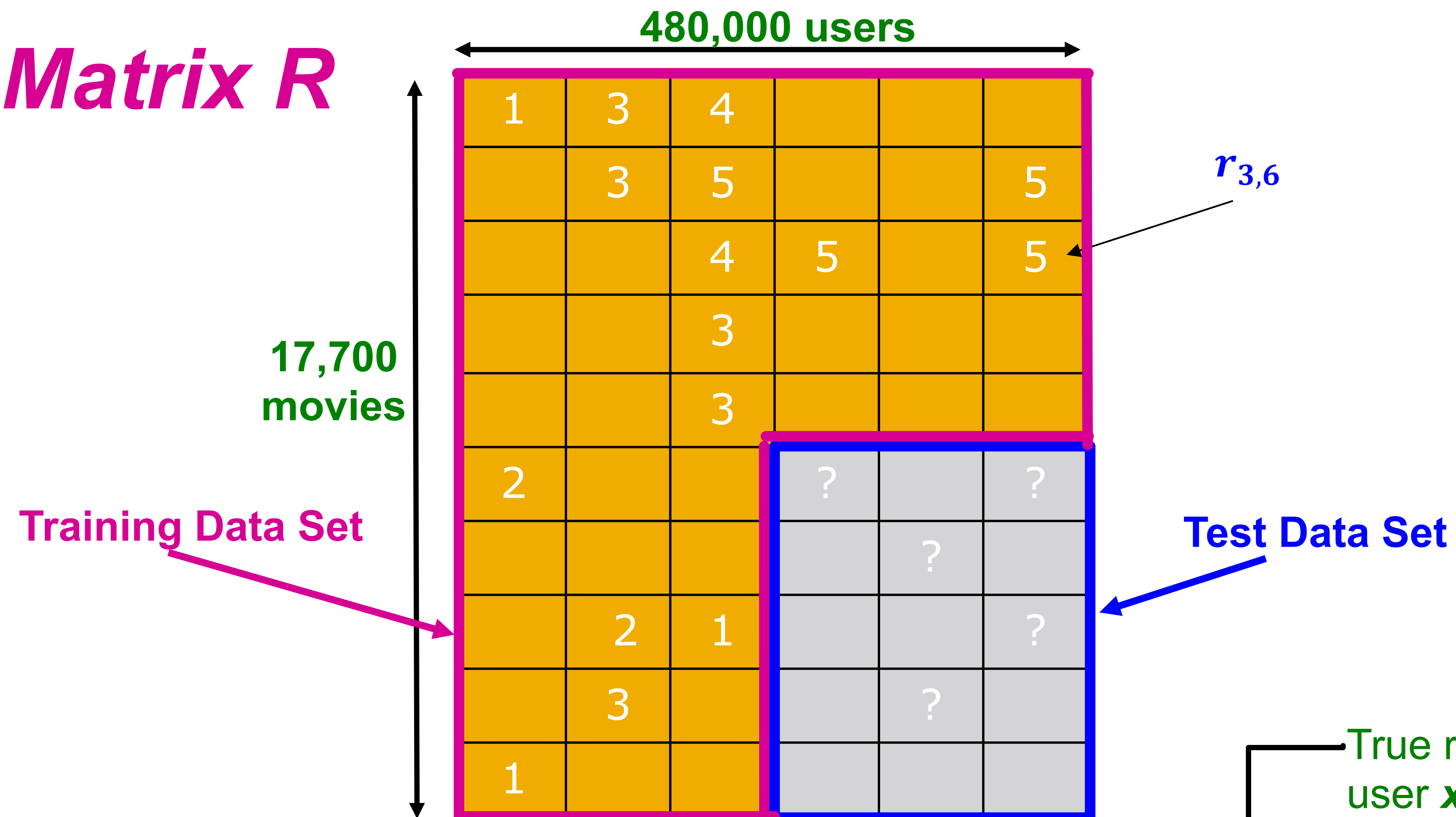
Matrix R

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					

Utility Matrix *R*: Evaluation



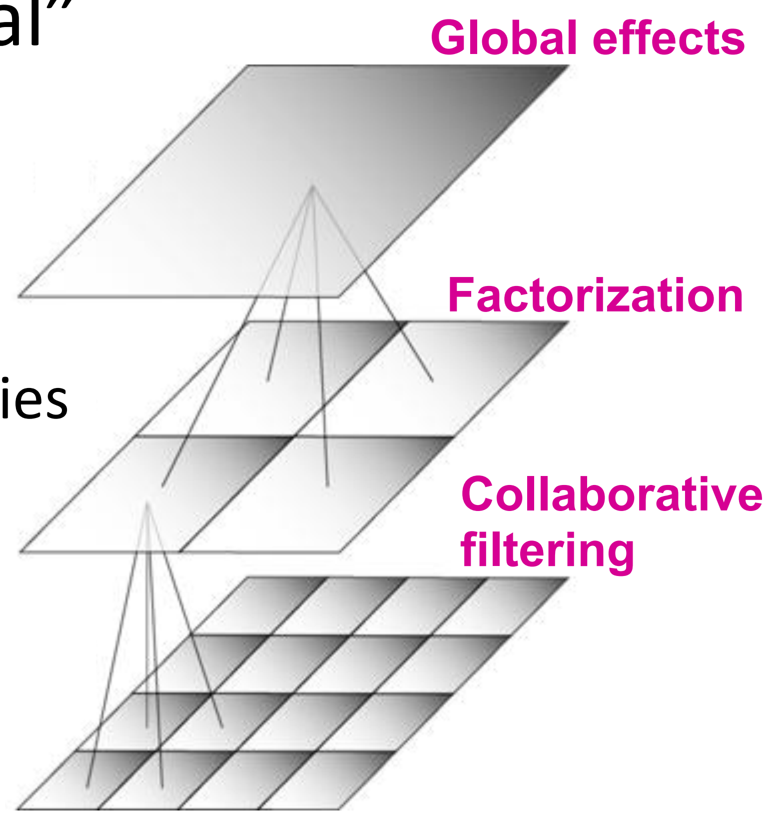
True rating of user ***x*** on item ***i***

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

Predicted rating

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

- **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**
    - **That is 4 = 3.7+0.5-0.2**

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

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

Recap: Collaborative Filtering (CF)

- **The earliest and the most popular collaborative filtering method**
- Derive unknown ratings from those of "**similar**" movies (item-item variant)
- Define **similarity measure** s_{ij} of items ***i*** and ***j***
- Select ***k***-nearest neighbors, compute the rating
  - ***N(i; x):*** items most similar to ***i*** that were rated by ***x***

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

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

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}

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

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** rather than **weighted avg.**:

$$\widehat{r_{xi}} = b_{xi} + \sum_{j \in N(i;x)} w_{ij}\left(r_{xj} - b_{xj}\right)$$

- **A few notes:**
  - $\boldsymbol{N(i;x)}$ … set of movies rated by user ***x*** that are similar to movie ***i***
  - $\boldsymbol{w_{ij}}$ is the **interpolation weight** (some real number)
    - Note, we allow: $\sum_{j \in N(i;x)} \boldsymbol{w_{ij}} \neq \mathbf{1}$
  - $\boldsymbol{w_{ij}}$ models interaction 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:** $\sum_{(i,x) \in R} (\hat{r}_{xi} - r_{xi})^2$
  - Find w_{ij} that minimize **SSE** on **training data!**
    - Models relationships between item ***i*** and its neighbors ***j***
  - w_{ij} can be **learned/estimated** based on ***x*** and all other users that rated ***i***

Why is this a good idea?

Recommendations via Optimization

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

- **Goal:** Make good recommendations
  - Quantify goodness using **RMSE:**
    Lower RMSE $\Rightarrow$ better recommendations
  - Want to make good recommendations on items that user has not yet seen. Can't really do this!
  - **Let's set build a system such that it works well on known (user, item) ratings**
    And **hope** the system will also predict well the **unknown ratings**

Recommendations via Optimization

- **Idea:** **Let's set values *w* such that they work well on known (user, item) ratings**
- **How to find such values *w*?**
- **Idea:** Define an objective function and solve the optimization problem

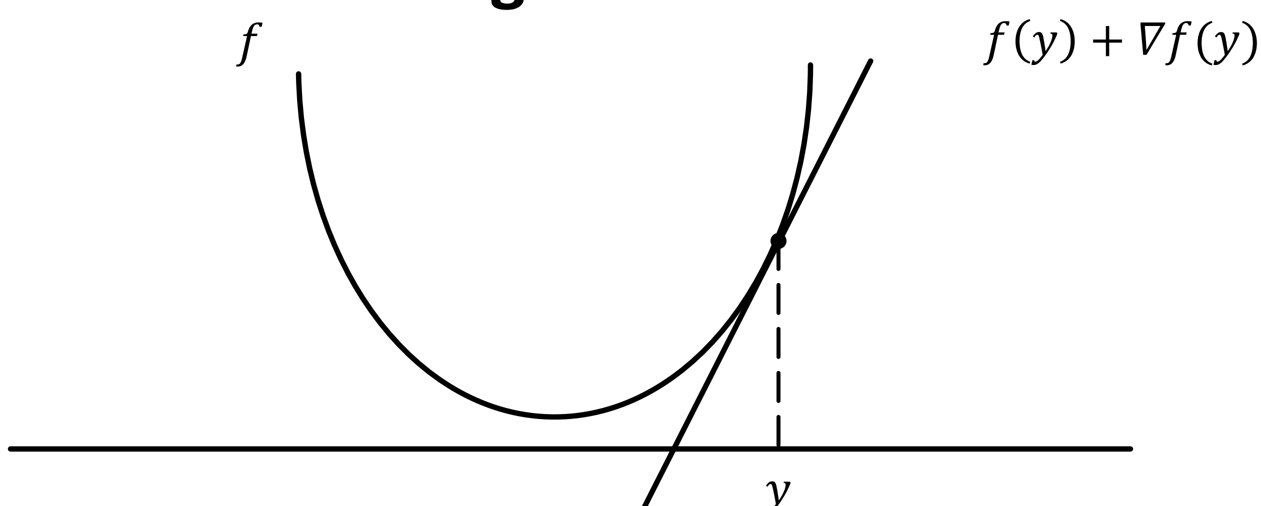
- Find $\boldsymbol{w_{ij}}$ that minimize **SSE** on **training data**!

$$J(w) = \sum_{x,i \in R} \left(\underbrace{\left[b_{xi} + \sum_{j \in N(i;x)} w_{ij}\left(r_{xj} - b_{xj}\right) \right]}_{\text{Predicted rating}} - \underbrace{r_{xi}}_{\text{True rating}} \right)^2$$

- Think of ***w*** as a vector of numbers

Detour: Minimizing a function

- **A simple way to minimize a function $f(x)$:**
  - Compute the derivative $\nabla f(x)$
  - **Start at some point y and evaluate $\nabla f(y)$**
  - **Make a step in the reverse direction of the gradient: $y = y - \nabla f(y)$**
  - **Repeat until convergence**



Interpolation Weights

$$J(w) = \sum_{x,i \in R} \left(\left[b_{xi} + \sum_{j \in N(i;x)} w_{ij} \left(r_{xj} - b_{xj} \right) \right] - r_{xi} \right)^2$$

- **We have the optimization problem, now what?**
- **Gradient descent:**
  - **Iterate until convergence:** $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \boldsymbol{\nabla_w J}$ $\quad$ η … learning rate

where $\boldsymbol{\nabla_w J}$ is the gradient (derivative evaluated on data):

$$\nabla_w J = \left[\frac{\partial J(w)}{\partial w_{ij}} \right] = 2 \sum_{x,i \in R} \left(\left[b_{xi} + \sum_{k \in N(i;x)} w_{ik} \left(r_{xk} - b_{xk} \right) \right] - r_{xi} \right) \left(r_{xj} - b_{xj} \right)$$

for $\boldsymbol{j} \in \{ \boldsymbol{N(i;x)}, \forall \boldsymbol{i}, \forall \boldsymbol{x} \}$

else $\frac{\partial J(w)}{\partial w_{ij}} = \mathbf{0}$

  - **Note:** We fix movie $\boldsymbol{i}$, go over all $\boldsymbol{r_{xi}}$, for every movie $\boldsymbol{j} \in \boldsymbol{N(i;x)}$, we compute $\frac{\boldsymbol{\partial J(w)}}{\boldsymbol{\partial w_{ij}}}$

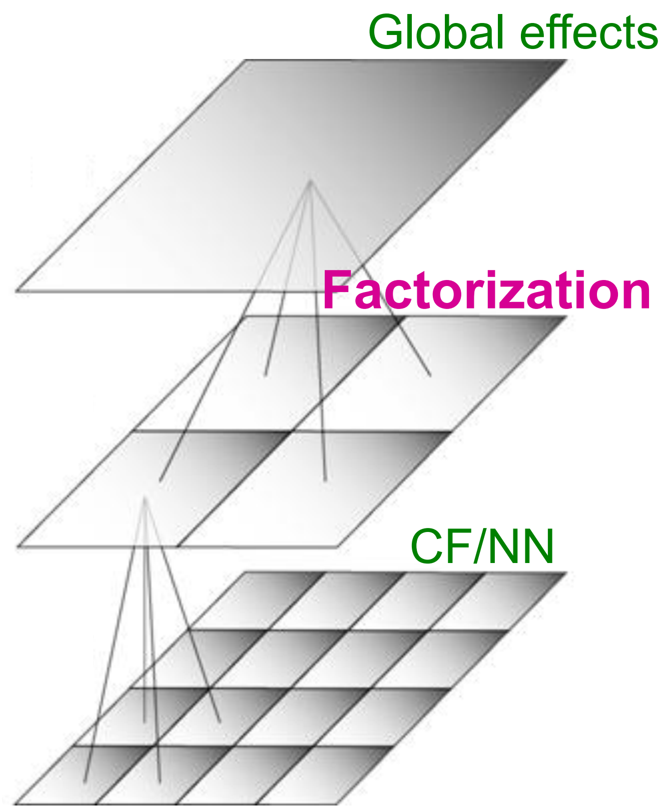
while $|w_{new} - w_{old}| > \varepsilon$:

$\quad w_{old} = w_{new}$

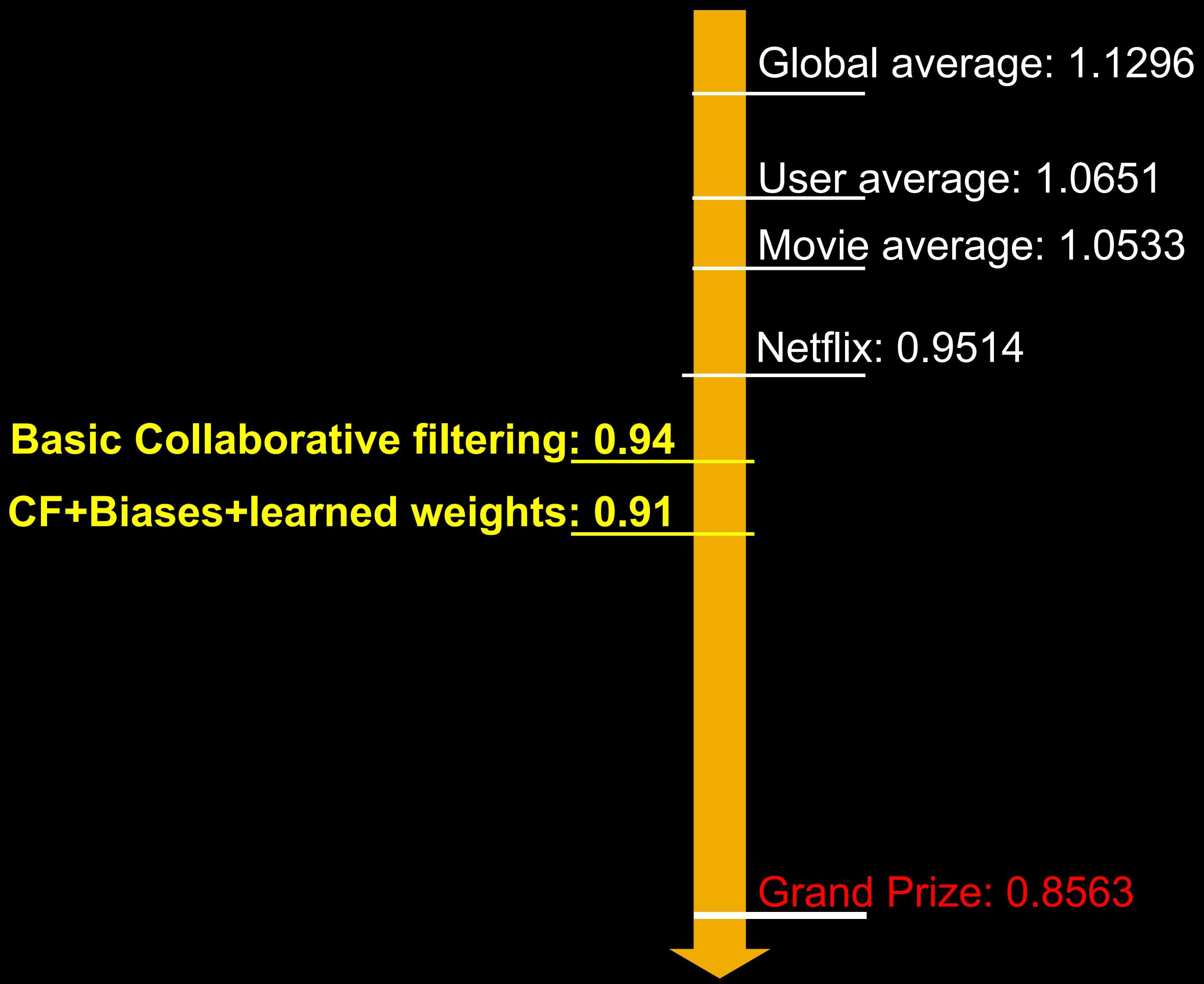
$\quad w_{new} = w_{old} - \eta \cdot \nabla w_{old}$

Interpolation Weights

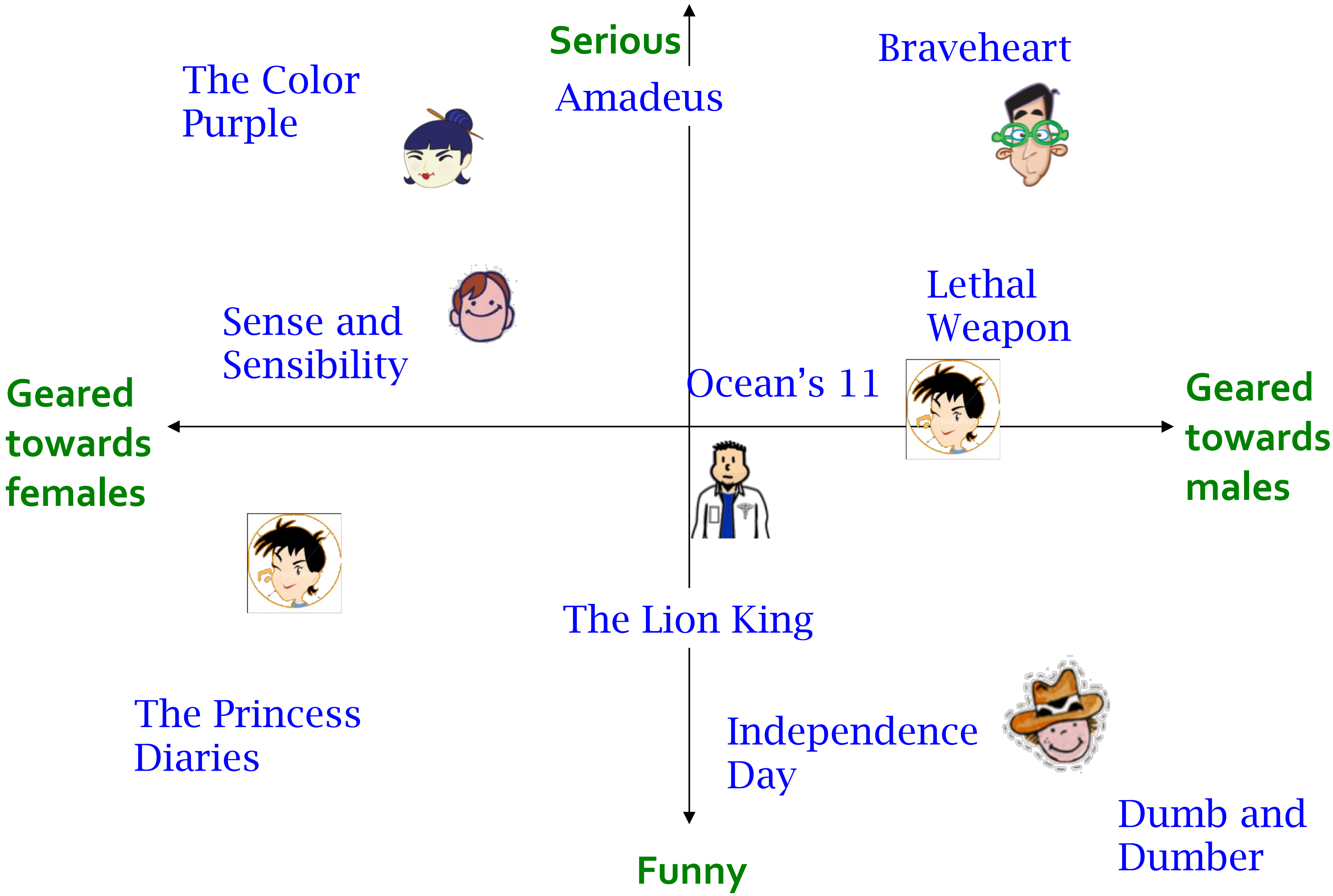
- **So far:** $\widehat{r_{xi}} = b_{xi} + \sum_{j \in N(i;x)} w_{ij}\left(r_{xj} - b_{xj}\right)$
  - Weights $\boldsymbol{w_{ij}}$ derived based on their roles; **no use of an arbitrary similarity measure** ($\boldsymbol{w_{ij}} \neq \boldsymbol{s_{ij}}$)
  - Explicitly account for interrelationships among the neighboring movies
- **Next: Latent factor model**
  - Extract "regional" correlations



Performance of Various Methods



Latent Factor Models (e.g., SVD)



Latent Factor Models

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

- "SVD" on Netflix data: **R** $\approx Q \cdot P^T$

R (items × users)

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

$\approx$

Q (items × factors)

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

P^T (factors × users)

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

- For now let's assume we can approximate the rating matrix ***R*** as a product of "thin" $Q \cdot P^T$
  - ***R*** has missing entries but let's ignore that for now!
    - Basically, we want the reconstruction error to be small on known ratings and we don't care about the values on the missing ones

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

users (columns) / items (rows)

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

≈

Q: items (rows) × factors (columns)

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

•

P^T: factors (rows) × users (columns)

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

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***

q_i = row i of Q

p_x = column x of P^T

users (columns) / items (rows)

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

≈

Q (rows: items; columns: factors)

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

•

P^T (rows: factors; columns: users)

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

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

Ratings matrix (items × users):

1		3			5			5		4	
		5	4	**2.4**		4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

≈

Q (items × *f* factors):

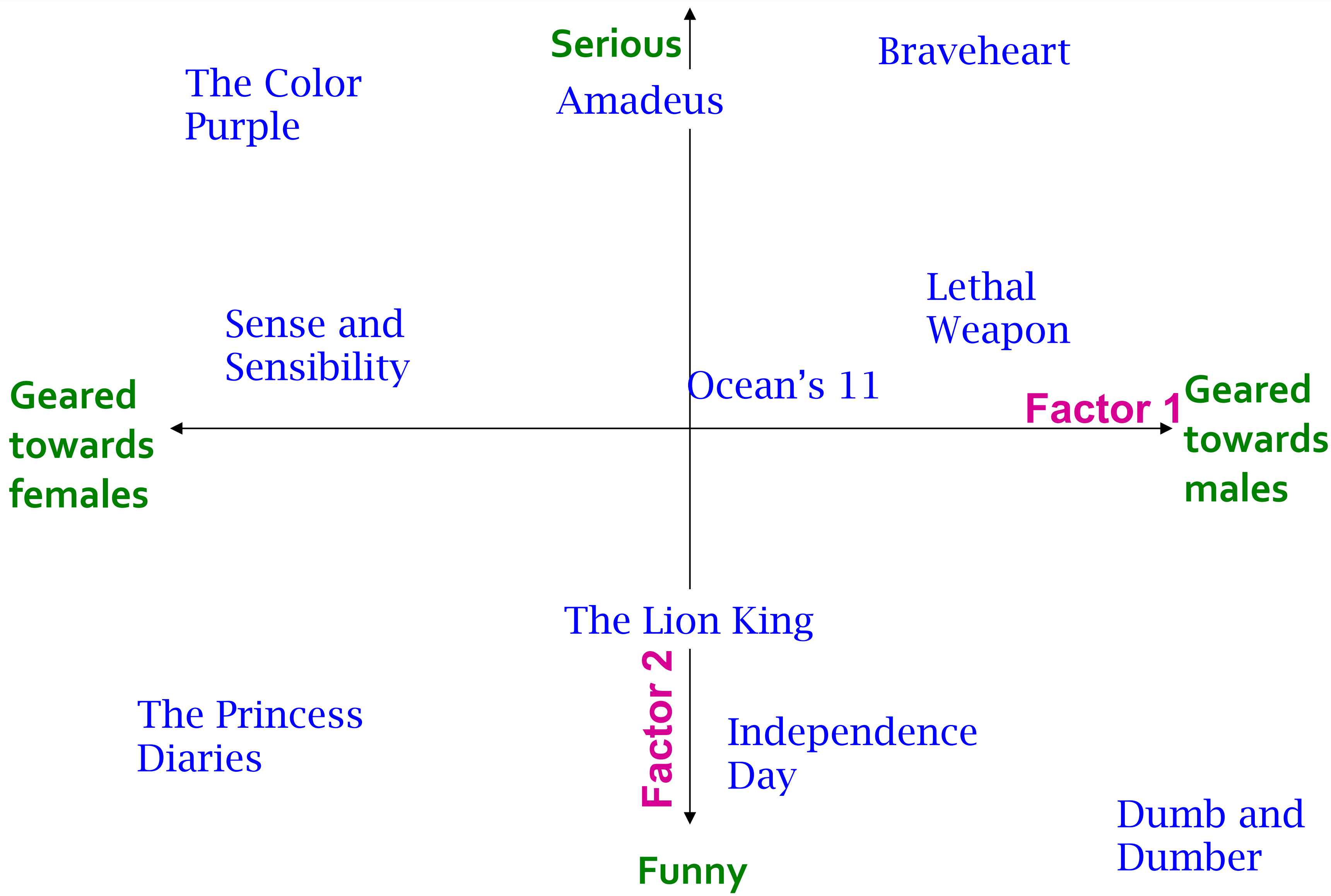
.1	-.4	.2
-.5	**.6**	**.5**
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

•

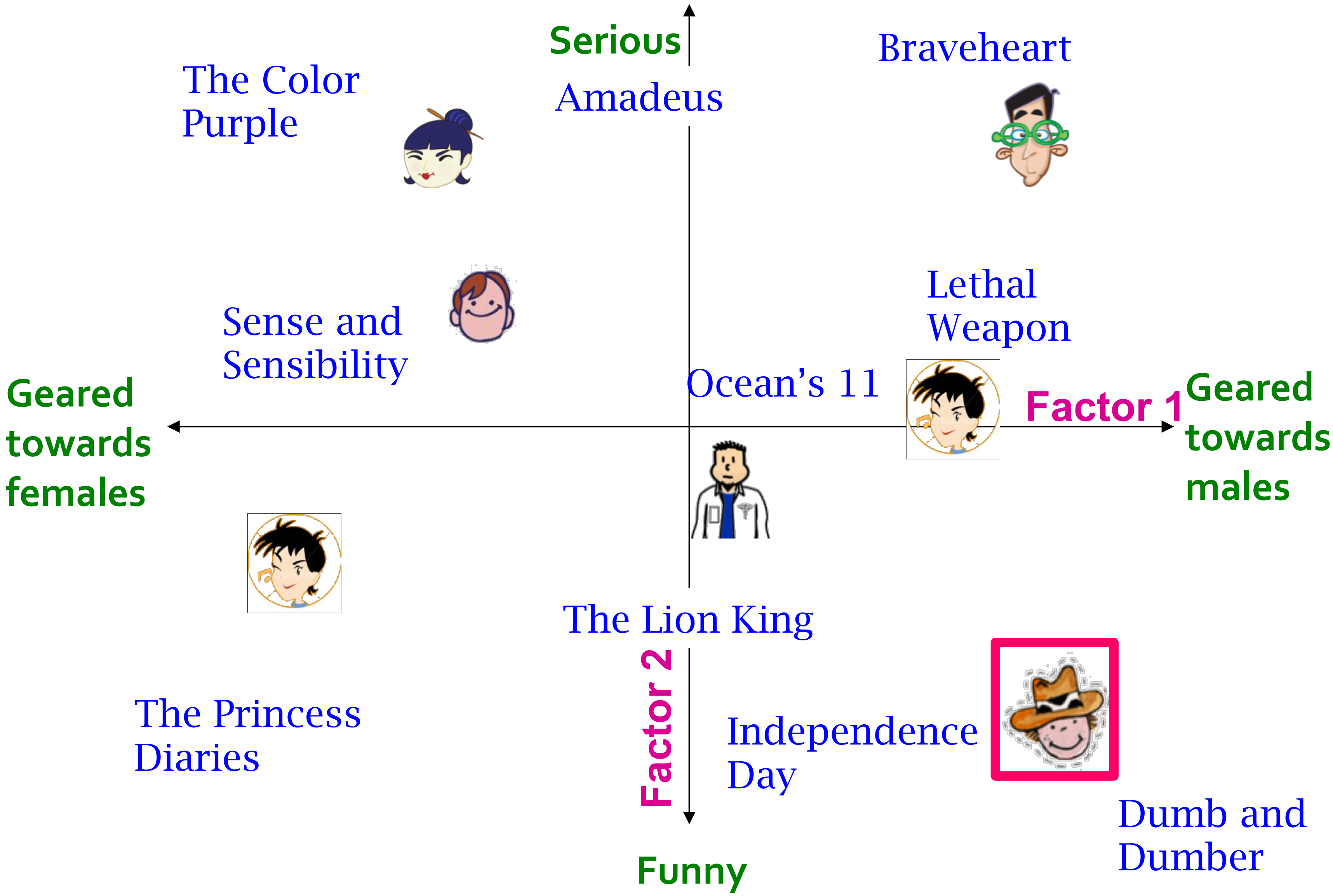
P^T (*f* factors × users):

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

Latent Factor Models

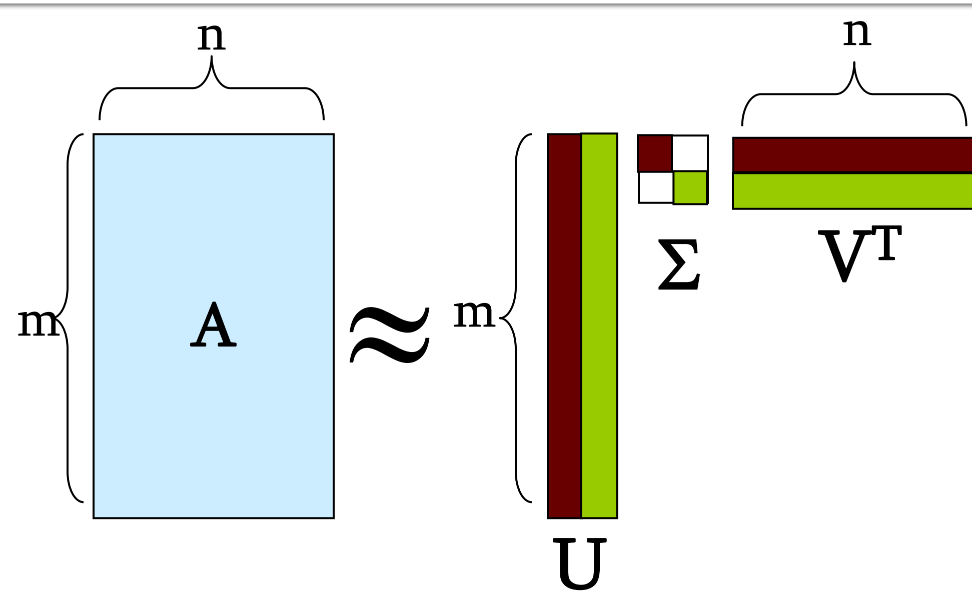


Latent Factor Models



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: $R \approx Q \cdot P^T$
  $A = R, \;\; Q = U, \;\; P^T = \Sigma\, V^T$

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

SVD: More good stuff

- **We already know that SVD gives minimum reconstruction error (Sum of Squared Errors):**

$$\min_{U,\mathrm{V},\Sigma} \sum_{ij \in A} \left(A_{ij} - [U \Sigma V^{\mathrm{T}}]_{ij} \right)^2$$

- **Note two things:**
  - **SSE** and **RMSE** are monotonically related:
    - $\boldsymbol{RMSE = \frac{1}{c}\sqrt{SSE}}$ **Great news: SVD is minimizing RMSE!**
  - **Complication:** The sum in SVD error term is over all entries (no-rating is interpreted as zero-rating). But our ***R*** has missing entries!

Latent Factor Models

users (columns) × items (rows):

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

≈

Q: items (rows) × factors (columns)

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

P^T: factors (rows) × users (columns)

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

- **SVD isn't defined when entries are missing!**
- **Use specialized methods to find *P, Q***
  - $\min_{P,Q} \sum_{(i,x)\in R} \left(r_{xi} - q_i \cdot p_x\right)^2$ $\qquad \hat{r}_{xi} = q_i \cdot p_x$
  - **Note:**
    - We don't require cols of ***P, Q*** to be orthogonal/unit length
    - ***P, Q*** map users/movies to a latent space
    - This was the most popular model among Netflix contestants

Finding the Latent Factors

Latent Factor Models

- **Our goal is to find P and Q such that:**

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

users (columns), items (rows):

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

≈

Q — factors (columns), items (rows):

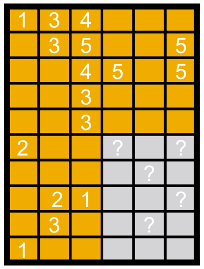
.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

P^T — users (columns), factors (rows):

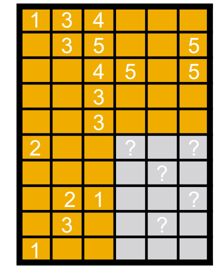
1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

Back to Our Problem

- **Want to minimize SSE for unseen test data**
- **Idea: Minimize SSE on training data**
  - Want large ***k*** (# of factors) to capture all the signals
  - But, **SSE** on **test data** begins to rise for ***k*** > 2
- This is a classical example of **overfitting:**
  - With too much freedom (too many free parameters) the model starts fitting noise
    - That is, the model fits too well the training data and is thus **not generalizing** well to unseen test data



Dealing with Missing Entries



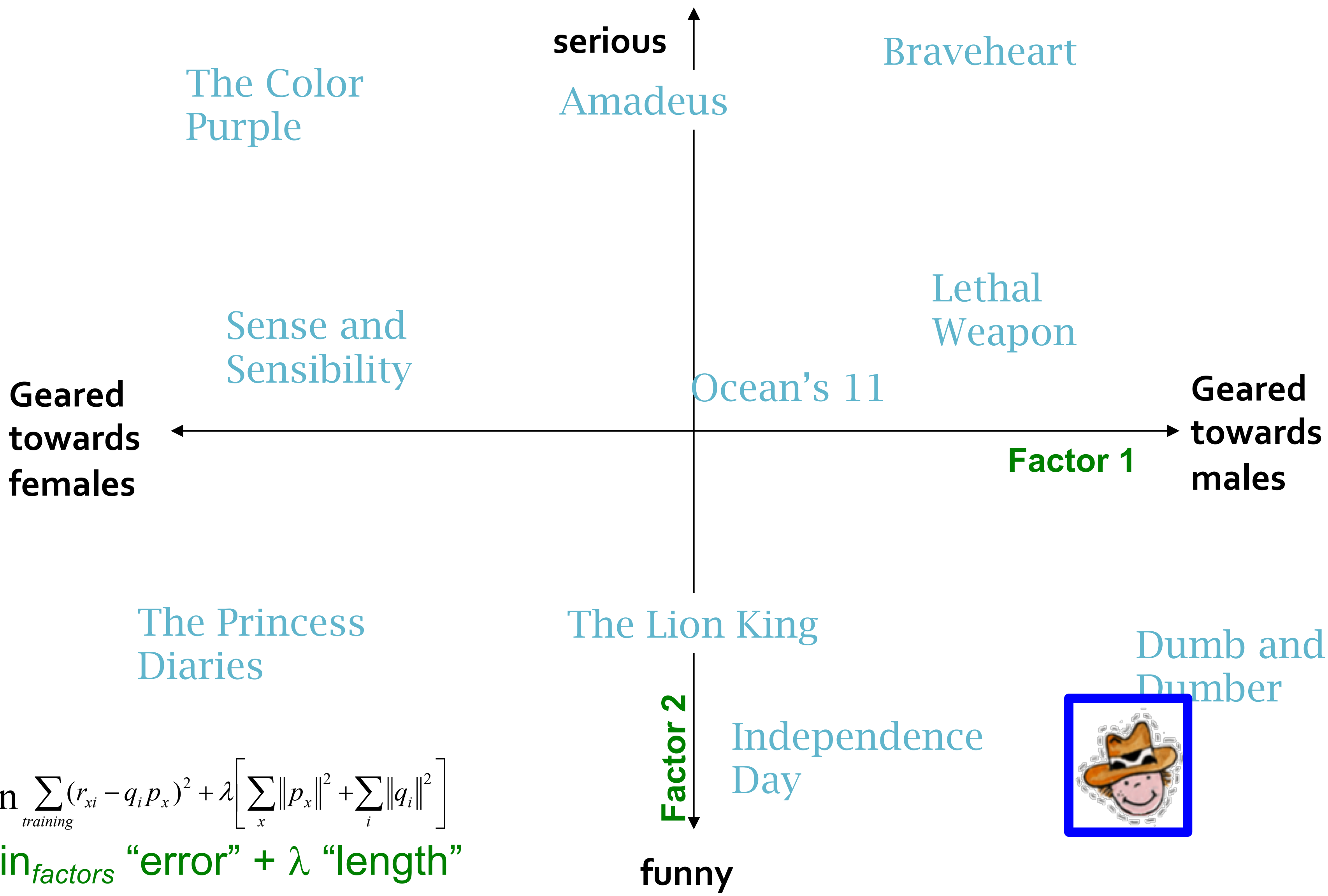
- **To solve overfitting we introduce regularization:**
  - Allow rich model where there is sufficient data
  - Shrink aggressively where data is scarce

$$\min_{P,Q} \underbrace{\sum_{training} (r_{xi} - q_i p_x)^2}_{\text{"error"}} + \left[\underbrace{\lambda_1 \sum_x \|p_x\|^2 + \lambda_2 \sum_i \|q_i\|^2}_{\text{"length"}} \right]$$

λ_1, λ_2 … user set regularization parameters

Note: We do not care about the "raw" value of the objective function, but we care about P,Q that achieve the minimum of the objective

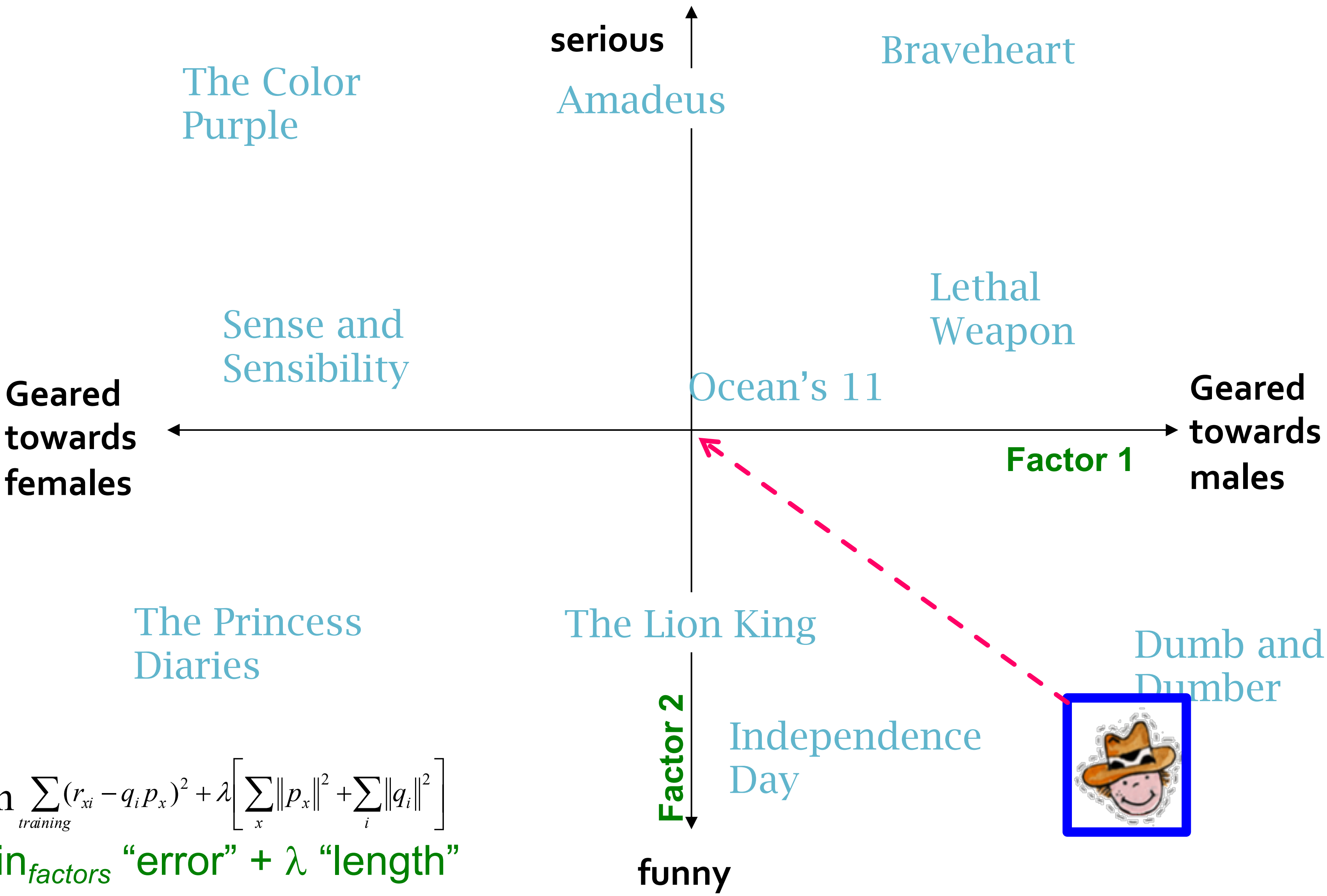
The Effect of Regularization



$$\min_{P,Q} \sum_{training} (r_{xi} - q_i p_x)^2 + \lambda \left[\sum_x \|p_x\|^2 + \sum_i \|q_i\|^2 \right]$$

$\min_{factors}$ "error" + λ "length"

The Effect of Regularization



$$\min_{P,Q} \sum_{training} (r_{xi} - q_i p_x)^2 + \lambda \left[\sum_x \|p_x\|^2 + \sum_i \|q_i\|^2 \right]$$

$\min_{factors}$ "error" + λ "length"

The Effect of Regularization

serious

The Color Purple

Amadeus

Braveheart

Lethal Weapon

Sense and Sensibility

Ocean's 11

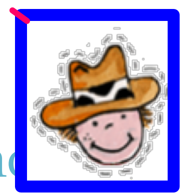
Geared towards females

Geared towards males

Factor 1

The Princess Diaries

The Lion King



Dumb and Dumber

Factor 2

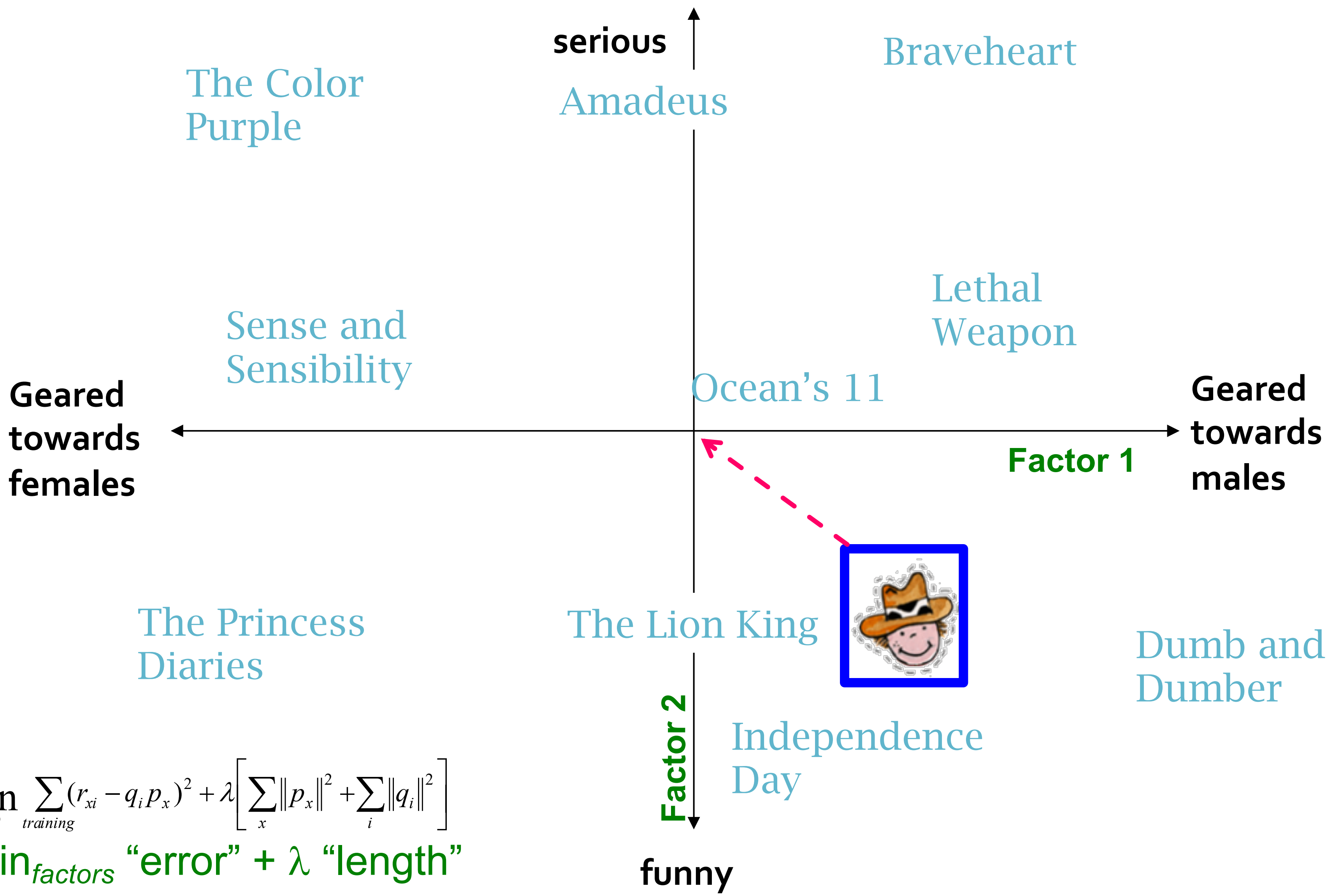
Independence Day

funny

$$\min_{P,Q} \sum_{training} (r_{xi} - q_i p_x)^2 + \lambda \left[\sum_x \|p_x\|^2 + \sum_i \|q_i\|^2 \right]$$

$\min_{factors}$ "error" + λ "length"

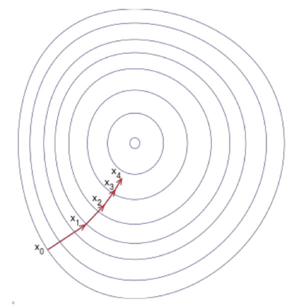
The Effect of Regularization



$$\min_{P,Q} \sum_{training} (r_{xi} - q_i p_x)^2 + \lambda \left[\sum_x \|p_x\|^2 + \sum_i \|q_i\|^2 \right]$$

$\min_{factors}$ "error" + λ "length"

Stochastic Gradient Descent



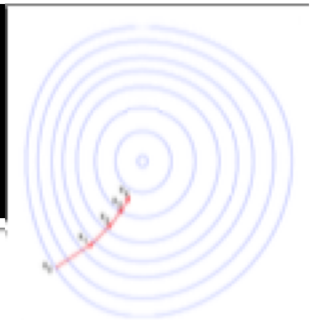
- **Want to find matrices *P* and *Q*:**

$$\min_{P,Q} \sum_{training} (r_{xi} - q_i p_x)^2 + \left[\lambda_1 \sum_x \|p_x\|^2 + \lambda_2 \sum_i \|q_i\|^2 \right]$$

- **Gradient descent:**
  - Initialize ***P*** and ***Q*** (using SVD, pretend missing ratings are 0)
  - Do gradient descent:
    - $\boldsymbol{P} \leftarrow \boldsymbol{P} - \eta \cdot \nabla \mathbf{P}$
    - $\boldsymbol{Q} \leftarrow \boldsymbol{Q} - \eta \cdot \nabla \mathbf{Q}$
    - where ***∇Q*** is gradient/derivative of matrix ***Q***:
      $\nabla Q = [\nabla q_{if}]$ and $\nabla q_{if} = \sum_{x,i} -2(r_{xi} - q_i p_x) p_{xf} + 2\lambda_2 q_{if}$
      - Here $\boldsymbol{q_{if}}$ is entry ***f*** of row $\boldsymbol{q_i}$ of matrix ***Q***
  - **Observation: Computing gradients is slow!**

How to compute gradient of a matrix?
Compute gradient of every element independently!

Stochastic Gradient Descent



- **Gradient Descent (GD) vs. Stochastic GD**
  - **Observation:** $\nabla Q = [\nabla q_{if}]$ where

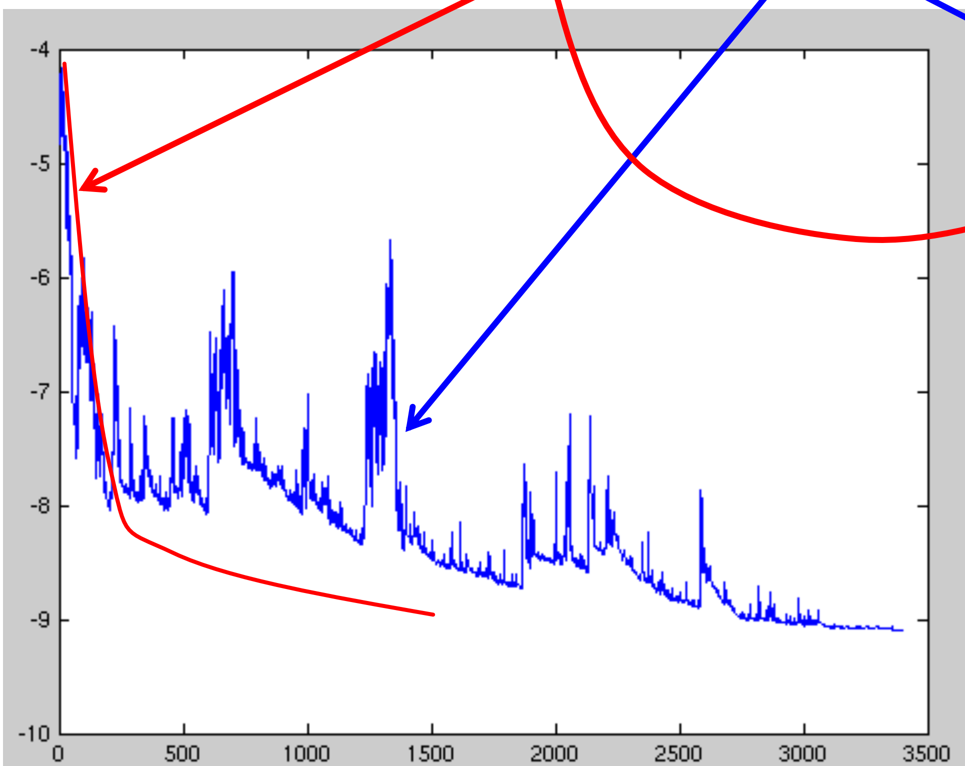
$$\nabla q_{if} = \sum_{x,i} -2(r_{xi} - q_{if} p_{xf}) p_{xf} + 2\lambda q_{if} = \sum_{\boldsymbol{x,i}} \nabla \boldsymbol{Q}\,(\boldsymbol{r_{xi}})$$

    - Here $\boldsymbol{q_{if}}$ is entry $\boldsymbol{f}$ of row $\boldsymbol{q_i}$ of matrix $\boldsymbol{Q}$
  - $\boldsymbol{Q} \leftarrow \boldsymbol{Q} - \eta \nabla \boldsymbol{Q} = \boldsymbol{Q} - \eta\left[\sum_{\boldsymbol{x,i}} \nabla \boldsymbol{Q}\,(\boldsymbol{r_{xi}})\right]$
  - **Idea:** Instead of evaluating gradient over all ratings evaluate it for each individual rating and make a step
- **GD:** $\boldsymbol{Q} \leftarrow \boldsymbol{Q} - \eta\left[\sum_{\boldsymbol{r_{xi}}} \nabla \boldsymbol{Q}(\boldsymbol{r_{xi}})\right]$
- **SGD:** $\boldsymbol{Q} \leftarrow \boldsymbol{Q} - \mu \nabla \boldsymbol{Q}(\boldsymbol{r_{xi}})$
  - **Faster convergence!**
    - Need more steps but each step is computed much faster

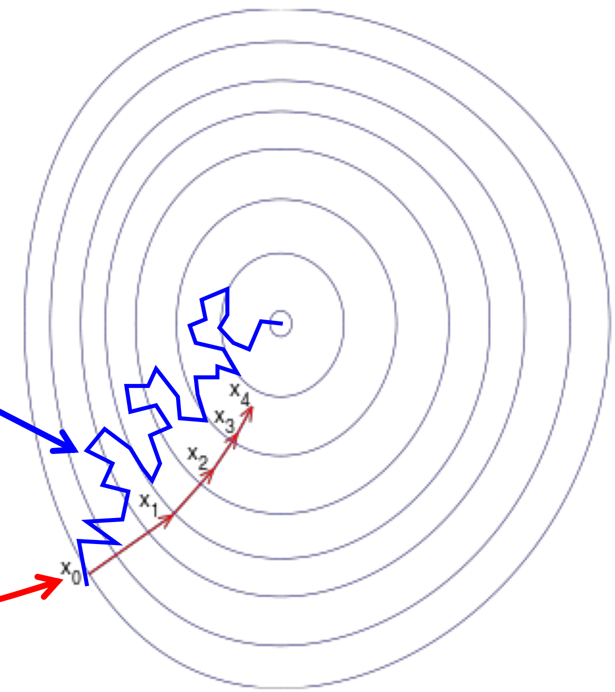
SGD vs. GD

- **Convergence of GD vs. SGD**

Value of the objective function

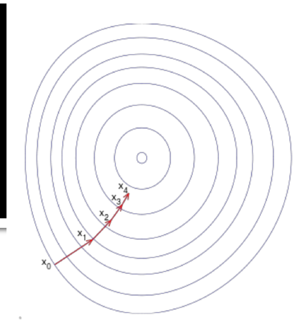


Iteration/step



GD improves the value of the objective function at every step.
SGD improves the value but in a "noisy" way.
GD takes fewer steps to converge but each step takes much longer to compute.
In practice, **SGD** is much faster!

Stochastic Gradient Descent



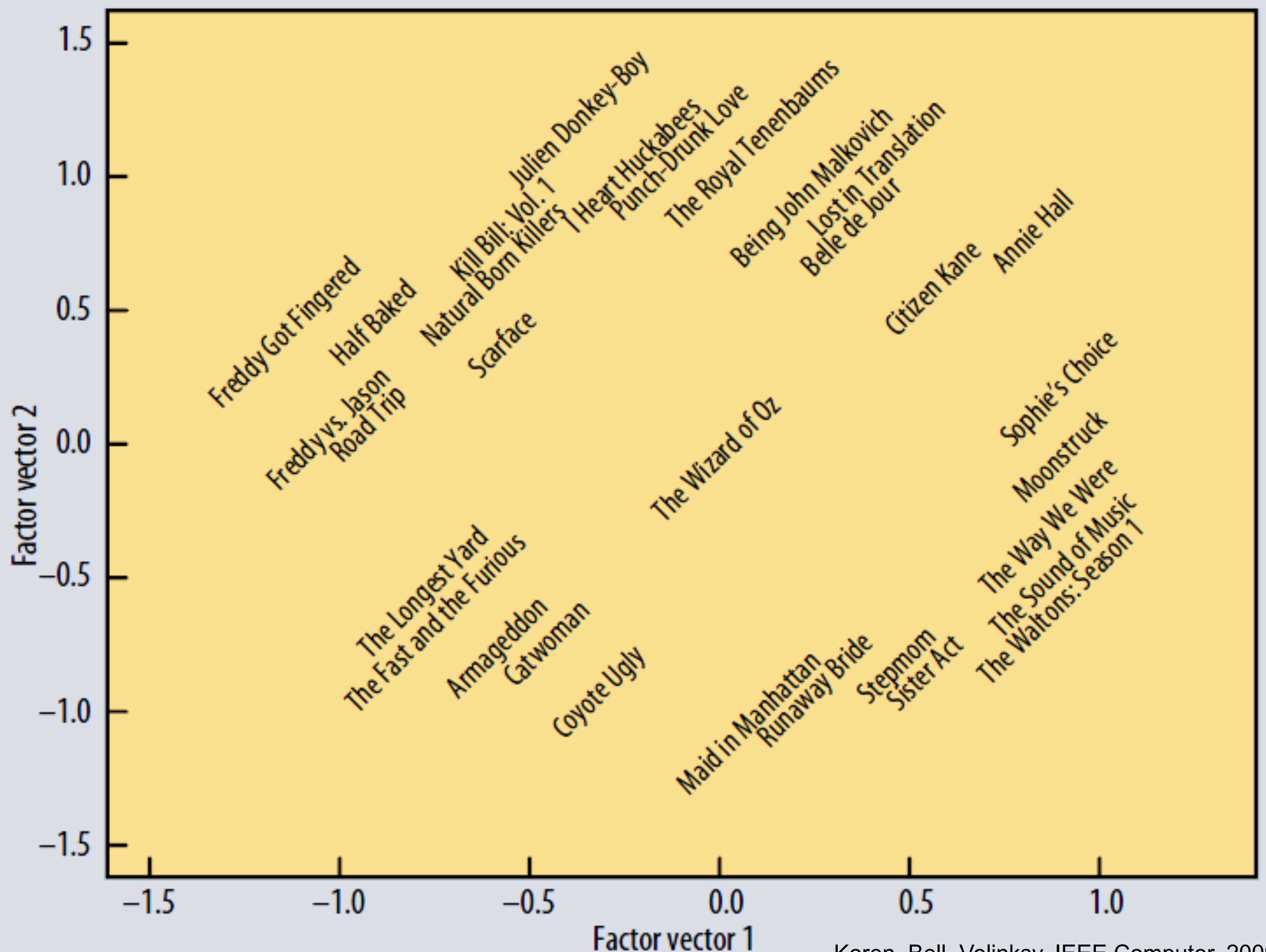
- **Stochastic gradient descent:**
  - Initialize ***P*** and ***Q*** (using SVD, pretend missing ratings are 0)
  - Then iterate over the ratings (multiple times if necessary) and update factors:

  For each r_{xi}:

    - $\varepsilon_{xi} = 2(r_{xi} - q_i \cdot p_x)$ (derivative of the "error")
    - $q_i \leftarrow q_i + \mu_1 \left(\varepsilon_{xi}\, p_x - 2\lambda_2\, q_i\right)$ (update equation)
    - $p_x \leftarrow p_x + \mu_2 \left(\varepsilon_{xi}\, q_i - 2\lambda_1\, p_x\right)$ (update equation)

  μ … learning rate

- **Two For loops:**
  - For until convergence:
    - For each $\mathbf{r}_{xi}$
      - Compute gradient, do a "step" as above



Koren, Bell, Volinksy, IEEE Computer, 2009

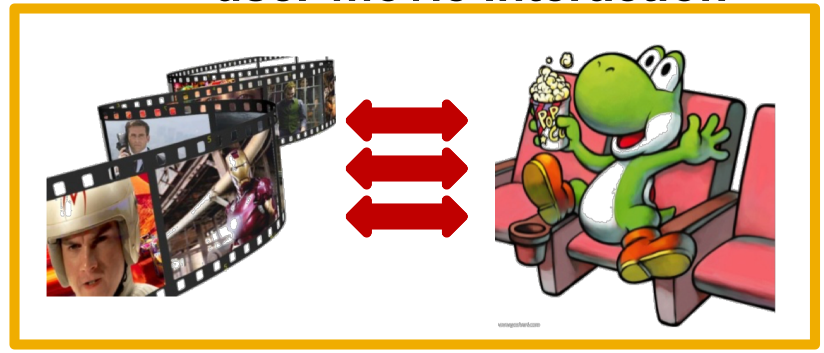
Extending Latent Factor Model to Include Biases

Modeling Biases and Interactions

user bias

movie bias

user-movie interaction



Baseline predictor
- Separates users and movies
- Benefits from insights into user's behavior
- Among the main practical contributions of the competition

User-Movie interaction
- Characterizes the matching between users and movies
- Attracts most research in the field
- Benefits from algorithmic and mathematical innovations

- $\boldsymbol{\mu}$ = overall mean rating
- $\boldsymbol{b_x}$ = bias of user $\boldsymbol{x}$
- $\boldsymbol{b_i}$ = bias of movie $\boldsymbol{i}$

Baseline Predictor

- We have expectations on the rating by user ***x*** of movie ***i***, even without estimating ***x***'s attitude towards movies like ***i***

- Rating scale of user ***x***
- Values of other ratings user gave recently (day-specific mood, anchoring, multi-user accounts)

- (Recent) popularity of movie ***i***
- Selection bias; related to number of ratings user gave on the same day ("frequency")

Putting It All Together

$$r_{xi} = \underset{\text{Overall mean rating}}{\mu} + \underset{\text{Bias for user } \boldsymbol{x}}{b_x} + \underset{\text{Bias for movie } \boldsymbol{i}}{b_i} + \underset{\text{User-Movie interaction}}{q_i \cdot p_x}$$

- **Example:**
  - Mean rating: $\boldsymbol{\mu = 3.7}$
  - You are a critical reviewer: your mean rating is 1 star lower than the mean: $\boldsymbol{b_x = -1}$
  - Star Wars gets a mean rating of *0.5* higher than average movie: $\boldsymbol{b_i = +0.5}$
  - Predicted rating for you on Star Wars:
    $\boldsymbol{= 3.7 - 1 + 0.5 = 3.2}$

Fitting the New Model

- **Solve:**

$$\min_{Q,P} \sum_{(x,i)\in R} \left(r_{xi} - (\mu + b_x + b_i + q_i\, p_x)\right)^2$$

goodness of fit

$$+\left(\lambda_1 \sum_i \|q_i\|^2 + \lambda_2 \sum_x \|p_x\|^2 + \lambda_3 \sum_x \|b_x\|^2 + \lambda_4 \sum_i \|b_i\|^2\right)$$

regularization

λ is selected via grid-search on a validation set

- **Stochastic gradient decent to find parameters**
  - **Note:** Both biases $\boldsymbol{b_x}$, $\boldsymbol{b_i}$ as well as interactions $\boldsymbol{q_i}$, $\boldsymbol{p_x}$ are treated as parameters (and we learn them)

Performance of Various Methods

Global average: 1.1296

User average: 1.0651

Movie average: 1.0533

Netflix: 0.9514

Basic Collaborative filtering: 0.94

Collaborative filtering++: 0.91

Latent factors: 0.90

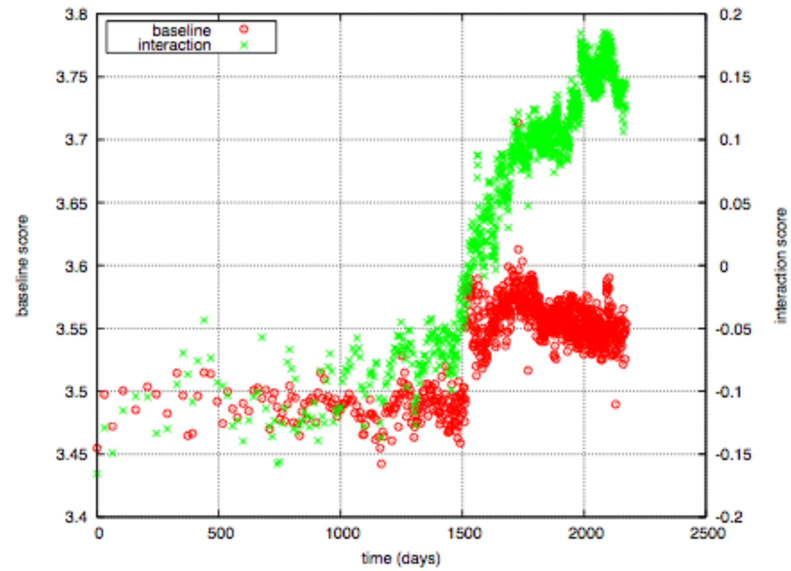
Latent factors+Biases: 0.89

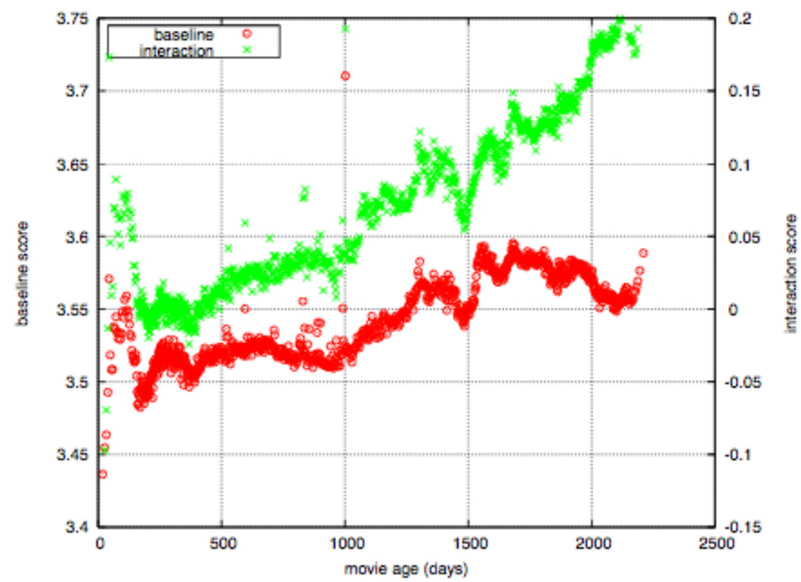
Grand Prize: 0.8563

The Netflix Challenge: 2006-09

Temporal Biases Of Users

- **Sudden rise in the average movie rating** (early 2004)
  - Improvements in Netflix
  - GUI improvements
  - Meaning of rating changed
- **Movie age**
  - Users prefer new movies without any reasons
  - Older movies are just inherently better than newer ones





[Y. Koren, Collaborative filtering with temporal dynamics, KDD ’09]

Temporal Biases & Factors

- **Original model:**
  $r_{xi} = \mu + b_x + b_i + q_i \cdot p_x$
- **Add time dependence to biases:**
  $r_{xi} = \mu + b_x(t) + b_i(t) + q_i \cdot p_x$
  - Make parameters $\boldsymbol{b_x}$ and $\boldsymbol{b_i}$ to depend on time
  - **(1)** Parameterize time-dependence by linear trends
    (2) Each bin corresponds to 10 consecutive weeks

$$b_i(t) = b_i + b_{i,\mathrm{Bin}(t)}$$

- **Add temporal dependence to factors**
  - $\boldsymbol{p_x(t)}$… user preference vector on day $\boldsymbol{t}$

Y. Koren, Collaborative filtering with temporal dynamics, KDD '09

Performance of Various Methods

Global average: 1.1296

User average: 1.0651

Movie average: 1.0533

Netflix: 0.9514

Basic Collaborative filtering: 0.94

Collaborative filtering++: 0.91

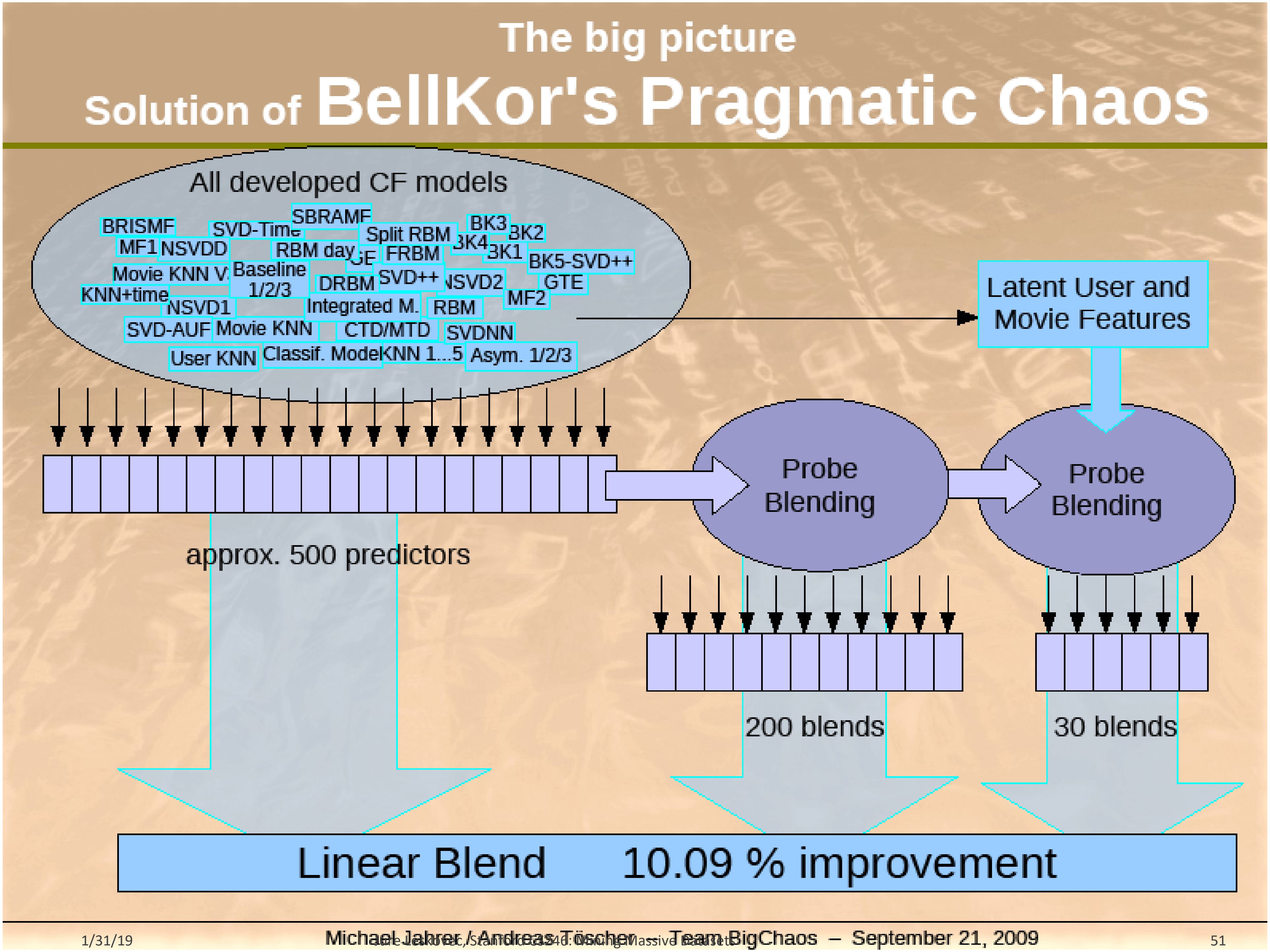
Latent factors: 0.90

Latent factors+Biases: 0.89

Latent factors+Biases+Time: 0.876

Grand Prize: 0.8563

Still no prize! ☹
Getting desperate.
Try a "kitchen sink" approach!


The big picture
Solution of BellKor's Pragmatic Chaos
All developed CF models
BRISMF
SVD-Time
SBRAMF
Split RBM
BK3
BK2
MF1
NSVDD
RBM day
GE
FRBM
BK4
BK1
BK5-SVD++
Movie KNN V
Baseline 1/2/3
DRBM
SVD++
NSVD2
GTE
KNN+time
NSVD1
Integrated M.
RBM
MF2
SVD-AUF
Movie KNN
CTD/MTD
SVDNN
User KNN
Classif. Model
KNN 1...5
Asym. 1/2/3
Latent User and Movie Features
approx. 500 predictors
Probe Blending
Probe Blending
200 blends
30 blends
Linear Blend
10.09 % improvement
Michael Jahrer / Andreas Töscher – Team BigChaos – September 21, 2009

Standing on June 26th 2009

NETFLIX

Netflix Prize

Home | Rules | Leaderboard | Register | Update | Submit | Download

Leaderboard

Display top 20 leaders.

Rank	Team Name	Best Score	% Improvement	Last Submit Time
1	BellKor's Pragmatic Chaos	0.8558	10.05	2009-06-26 18:42:37
Grand Prize - RMSE <= 0.8563				
2	PragmaticTheory	0.8582	9.80	2009-06-25 22:15:51
3	BellKor in BigChaos	0.8590	9.71	2009-05-13 08:14:09
4	Grand Prize Team	0.8593	9.68	2009-06-12 08:20:24
5	Dace	0.8604	9.56	2009-04-22 05:57:03
6	BigChaos	0.8613	9.47	2009-06-23 23:06:52
Progress Prize 2008 - RMSE = 0.8616 - Winning Team: BellKor in BigChaos				
7	BellKor	0.8620	9.40	2009-06-24 07:16:02
8	Gravity	0.8634	9.25	2009-04-22 18:31:32
9	Opera Solutions	0.8638	9.21	2009-06-26 23:18:13
10	BruceDengDaoCiYiYou	0.8638	9.21	2009-06-27 00:55:55
11	pengpengzhou	0.8638	9.21	2009-06-27 01:06:43
12	xlvector	0.8639	9.20	2009-06-26 13:49:04
13	xiangliang	0.8639	9.20	2009-06-26 07:47:34

June 26th submission triggers 30-day "last call"

The Last 30 Days

- **Ensemble team formed**
  - Group of other teams on leaderboard forms a new team
  - Relies on combining their models
  - Quickly also get a qualifying score over 10%
- **BellKor**
  - Continue to get small improvements in their scores
  - Realize they are in direct competition with team **Ensemble**
- **Strategy**
  - Both teams carefully monitoring the leader board
  - Only sure way to check for improvement is to submit a set of predictions
    - This alerts the other team of your latest score

24 Hours from the Deadline

- **Submissions limited to 1 a day**
  - Only 1 final submission could be made in the last 24h
- **24 hours before deadline…**
  - **BellKor** team member in Austria notices (by chance) that **Ensemble** posts a score that is slightly better than BellKor's
- **Frantic last 24 hours for both teams**
  - Much computer time on final optimization
  - Carefully calibrated to end about **an hour before deadline**
- **Final submissions**
  - **BellKor** submits a little early (on purpose), 40 mins before deadline
  - **Ensemble** submits their final entry 20 mins later
  - **….and everyone waits….**

NETFLIX

Netflix Prize

COMPLETED

Home | Rules | Leaderboard | Update | Download

Leaderboard

Showing Test Score. Click here to show quiz score

Display top 20 leaders.

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	[illegible]	[illegible]	[illegible]
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace_	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11
Progress Prize 2008 - RMSE = 0.8627 - Winning Team: BellKor in BigChaos				
13	xiangliang	0.8642	9.27	2009-07-15 14:53:22
14	Gravity	0.8643	9.26	2009-04-22 18:31:32
15	Ces	0.8651	9.18	2009-06-21 19:24:53
16	Invisible Ideas	0.8653	9.15	2009-07-15 15:53:04
17	Just a guy in a garage	0.8662	9.06	2009-05-24 10:02:54
18	J Dennis Su	0.8666	9.02	2009-03-07 17:16:17
19	Craig Carmichael	0.8666	9.02	2009-07-25 16:00:54
20	acmehill	0.8668	9.00	2009-03-21 16:20:50
Progress Prize 2007 [redacted]				

Million $ Awarded Sept 21st 2009

What’s the moral of the story?

Submit early! ☺

Acknowledgments

- Some slides and plots borrowed from Yehuda Koren, Robert Bell and Padhraic Smyth
- **Further reading:**
  - Y. Koren, Collaborative filtering with temporal dynamics, KDD '09
- https://web.archive.org/web/20141130213501/http://www2.research.att.com/~volinsky/netflix/bpc.html
- https://web.archive.org/web/20141227110702/http://www.the-ensemble.com/

Follow me on LinkedIn for more:
Steve Nouri
https://www.linkedin.com/in/stevenouri/