

Recommendation and Advertising

Shannon Quinn

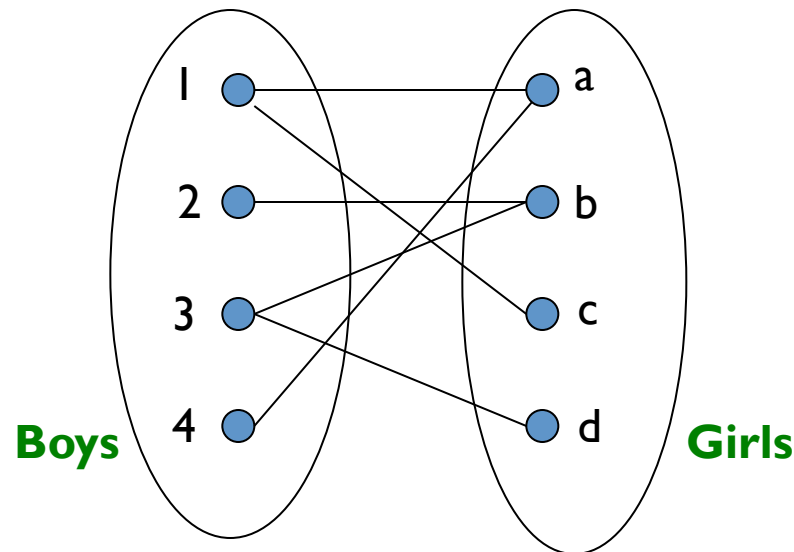
(with thanks to J. Leskovec, A.
Rajaraman, and J. Ullman of Stanford
University)

Lecture breakdown

- Part 1: Advertising
 - Bipartite Matching
 - AdWords
- Part 2: Recommendation
 - Collaborative Filtering
 - Latent Factor Models

I:Advertising on the Web

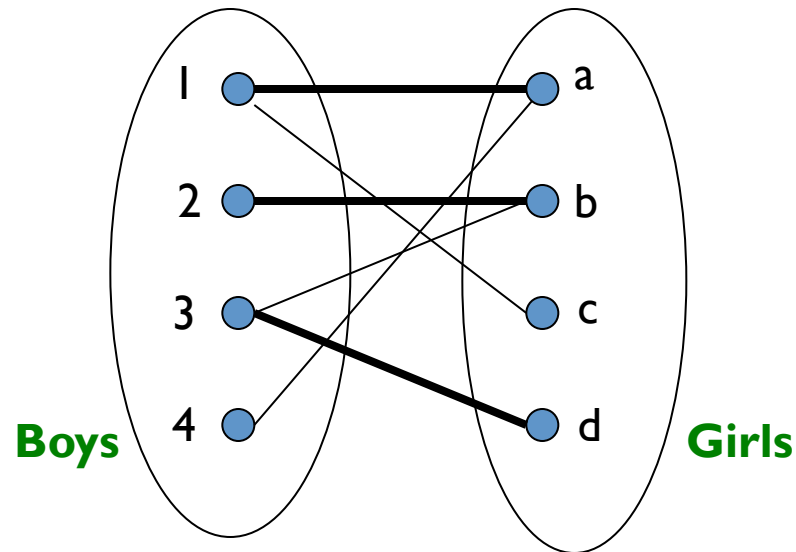
Example: Bipartite Matching



Nodes: Boys and Girls; Edges: Preferences

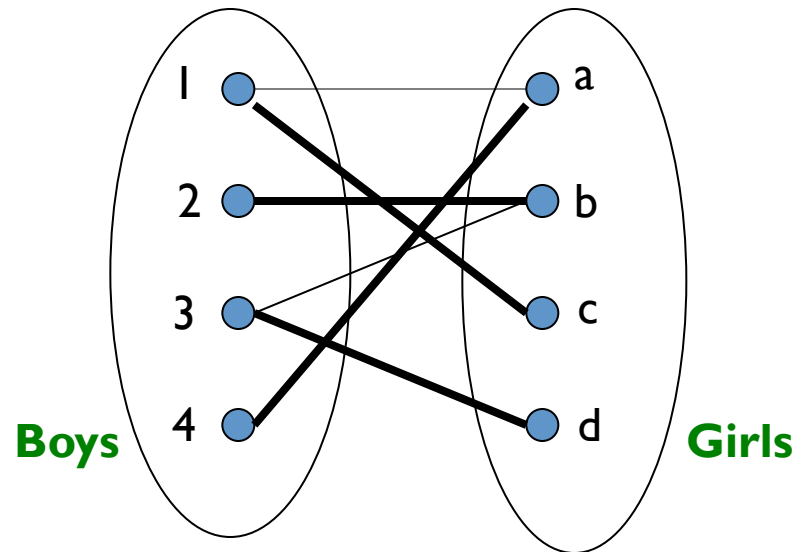
Goal: Match boys to girls so that maximum number of preferences is satisfied

Example: Bipartite Matching



$M = \{(1,a), (2,b), (3,d)\}$ is a **matching**
Cardinality of matching = $|M| = 3$

Example: Bipartite Matching



$M = \{(1,c), (2,b), (3,d), (4,a)\}$ is a
perfect matching

Perfect matching ... all vertices of the graph are matched

Maximum matching ... a matching that contains the largest possible number of matches

Matching Algorithm

- **Problem:** Find a maximum matching for a given bipartite graph
 - A perfect one if it exists
- There is a polynomial-time offline algorithm based on augmenting paths (Hopcroft & Karp 1973, see http://en.wikipedia.org/wiki/Hopcroft-Karp_algorithm)
- But what if we do not know the entire graph upfront?

Online Graph Matching Problem

- Initially, we are given the set boys
- In each round, one girl's choices are revealed
 - That is, girl's edges are revealed
- At that time, we have to decide to either:
 - Pair the girl with a boy
 - Do not pair the girl with any boy
- Example of application:
Assigning tasks to servers

Greedy Algorithm

- Greedy algorithm for the online graph matching problem:
 - Pair the new girl with **any** eligible boy
 - If there is none, do not pair girl
- How good is the algorithm?

Competitive Ratio

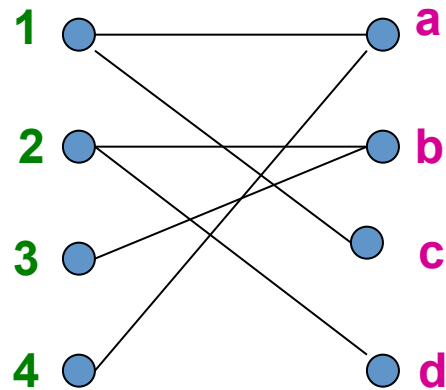
- For input I , suppose greedy produces matching M_{greedy} while an optimal matching is M_{opt}

Competitive ratio =

$$\min_{\text{all possible inputs } I} (|M_{greedy}|/|M_{opt}|)$$

(what is greedy's worst performance over all possible inputs I)

Worst-case Scenario



(1,a)
(2,b)

History of Web Advertising

- **Banner ads (1995-2001)**

- Initial form of web advertising

- Popular websites charged X\$ for every 1,000

- “impressions” of the ad

- Called “CPM” rate
(Cost per thousand impressions)

- Modeled similar to TV, magazine ads

- From untargeted to demographically targeted

- Low click-through rates

- Low ROI for advertisers



CPM...cost per mille
Mille...thousand in Latin

Performance-based Advertising

- Introduced by Overture around 2000
 - Advertisers **bid on search keywords**
 - When someone searches for that keyword, the **highest bidder's ad is shown**
 - Advertiser is charged only if the ad is clicked on
- Similar model adopted by Google with some changes around 2002
 - Called **Adwords**

Ads vs. Search Results

Web

Results 1 - 10 of about 2,230,000 for **geico**. (0.04 sec)

[GEICO](#) Car Insurance. Get an auto insurance quote and save today ...

GEICO auto insurance, online car insurance quote, motorcycle insurance quote, online insurance sales and service from a leading insurance company.

[www.geico.com/](#) - 21k - Sep 22, 2005 - [Cached](#) - [Similar pages](#)

[Auto Insurance](#) - [Buy Auto Insurance](#)

[Contact Us](#) - [Make a Payment](#)

[More results from www.geico.com »](#)

[Geico](#), Google Settle Trademark Dispute

The case was resolved out of court, so advertisers are still left without legal guidance on use of trademarks within ads or as keywords.

[www.clickz.com/news/article.php/3547356](#) - 44k - [Cached](#) - [Similar pages](#)

Google and [GEICO](#) settle AdWords dispute | The Register

Google and car insurance firm **GEICO** have settled a trade mark dispute over ... Car insurance firm **GEICO** sued both Google and Yahoo! subsidiary Overture in ...

[www.theregister.co.uk/2005/09/09/google_geico_settlement/](#) - 21k - [Cached](#) - [Similar pages](#)

[GEICO](#) v. Google

... involving a lawsuit filed by Government Employees Insurance Company (**GEICO**). **GEICO** has filed suit against two major Internet search engine operators, ...

[www.consumeraffairs.com/news04/geico_google.html](#) - 19k - [Cached](#) - [Similar pages](#)

Sponsored Links

[Great Car Insurance Rates](#)

Simplify Buying Insurance at Safeco

See Your Rate with an Instant Quote

[www.Safeco.com](#)

[Free Insurance Quotes](#)

Fill out one simple form to get multiple quotes from local agents.

[www.HometownQuotes.com](#)

[5 Free Quotes. 1 Form.](#)

Get 5 Free Quotes In Minutes!

You Have Nothing To Lose. It's Free

[sayyessoftware.com/Insurance](#)

Missouri

Web 2.0

- Performance-based advertising works!
 - Multi-billion-dollar industry
- Interesting problem:
What ads to show for a given query?
 - (Today's lecture)
- If I am an advertiser, which search terms should I bid on and how much should I bid?
 - (Not focus of today's lecture)

Adwords Problem

- **Given:**
 - 1. A set of bids by advertisers for search queries
 - 2. A click-through rate for each advertiser-query pair
 - 3. A budget for each advertiser (say for 1 month)
 - 4. A limit on the number of ads to be displayed with each search query
- **Respond to each search query with a set of advertisers such that:**
 - 1. The size of the set is no larger than the limit on the number of ads per query
 - 2. Each advertiser has bid on the search query
 - 3. Each advertiser has enough budget left to pay for the ad if it is clicked upon

Adwords Problem

- A stream of queries arrives at the search engine: q_1, q_2, \dots
- Several advertisers bid on each query
- When query q_i arrives, search engine must pick a subset of advertisers whose ads are shown
- **Goal:** Maximize search engine's revenues
 - **Simple solution:** Instead of raw bids, use the “expected revenue per click” (i.e., $\text{Bid} \times \text{CTR}$)
- **Clearly we need an online algorithm!**

The Adwords Innovation

Advertiser	Bid	CTR	Bid * CTR
A	\$1.00	1%	1 cent
B	\$0.75	2%	1.5 cents
C	\$0.50	2.5%	1.125 cents

Click through
rate

Expected
revenue

The Adwords Innovation

Advertiser	Bid	CTR	Bid * CTR
B	\$0.75	2%	1.5 cents
C	\$0.50	2.5%	1.125 cents
A	\$1.00	1%	1 cent

Complications: Budget

- Two complications:
 - Budget
 - CTR of an ad is unknown
- Each advertiser has a limited budget
 - Search engine guarantees that the advertiser will not be charged more than their daily budget

Complications: CTR

- CTR: Each ad has a different likelihood of being clicked
 - Advertiser 1 bids \$2, click probability = 0.1
 - Advertiser 2 bids \$1, click probability = 0.5
 - Clickthrough rate (CTR) is measured historically
 - Very hard problem: Exploration vs. exploitation
 - Exploit: Should we keep showing an ad for which we have good estimates of click-through rate
 - or
 - Explore: Shall we show a brand new ad to get a better sense of its click-through rate

BALANCE Algorithm [MSVV]

- **BALANCE** Algorithm by Mehta, Saberi, Vazirani, and Vazirani
 - For each query, pick the advertiser with the largest unspent budget
 - Break ties arbitrarily (but in a deterministic way)

Example: BALANCE

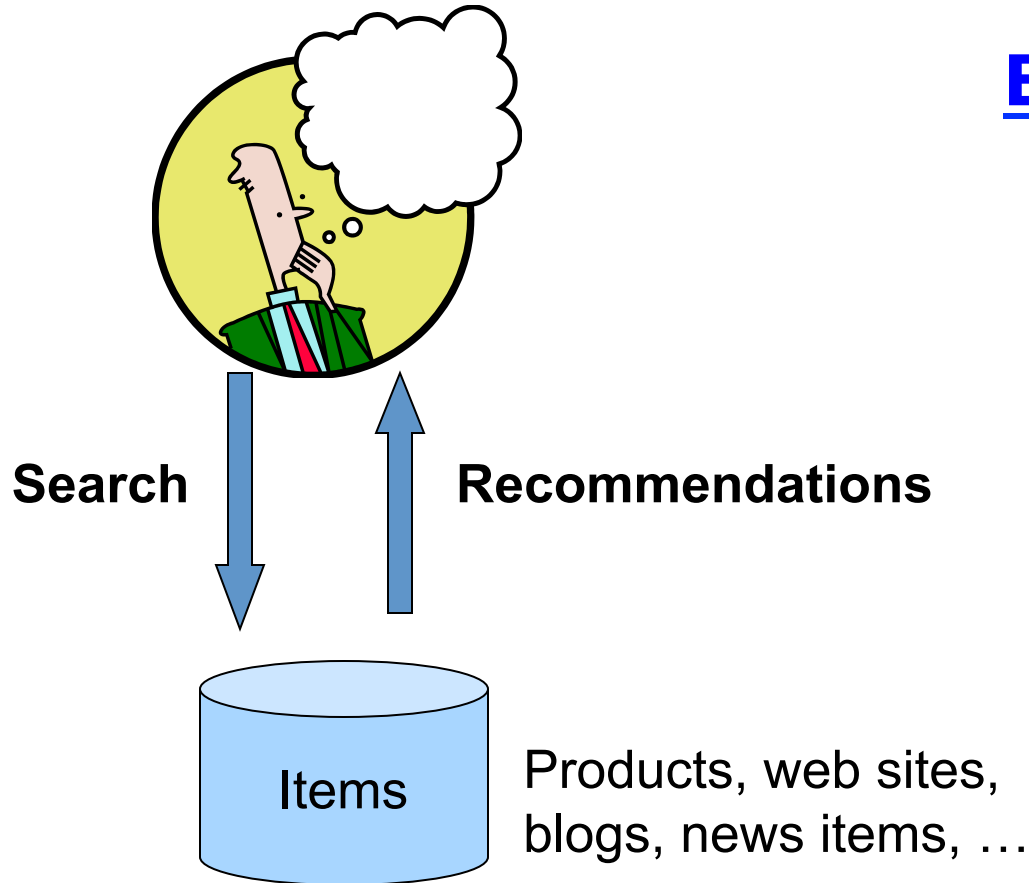
- Two advertisers A and B
 - A bids on query x , B bids on x and y
 - Both have budgets of \$4
- Query stream: $xxxxyyyy$
- BALANCE choice: A B A B B B _ _
 - Optimal: A A A A B B B B
- In general: For BALANCE on 2 advertisers
Competitive ratio = $\frac{3}{4}$

BALANCE: General Result

- In the general case, worst competitive ratio of BALANCE is $1 - 1/e = \text{approx. } 0.63$
 - Interestingly, no online algorithm has a better competitive ratio!

2: Recommender Systems

Recommendations



Examples:

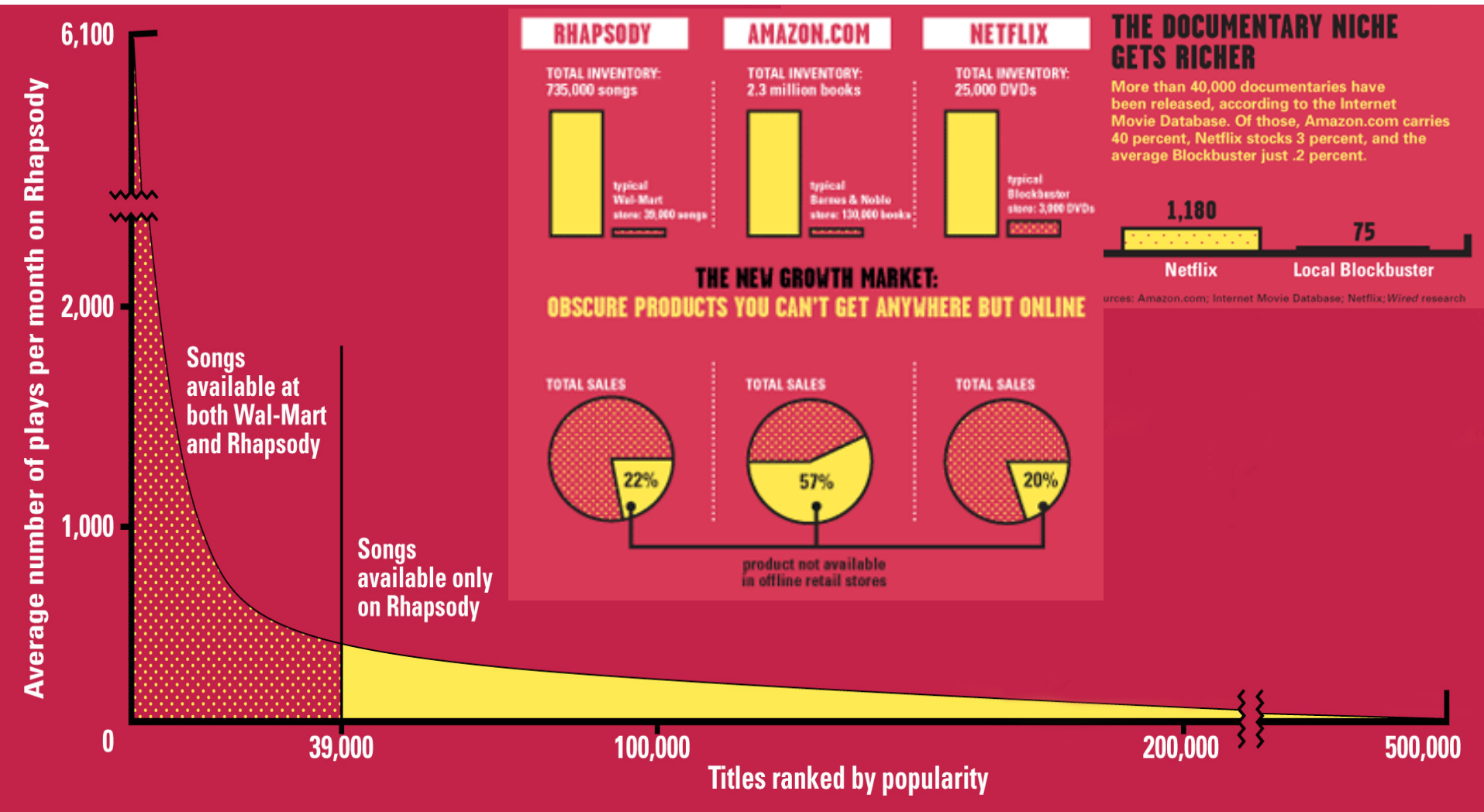
amazon.com.



movielens
helping you find the *right* movies



Sidenote: The Long Tail



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

Source: Chris Anderson (2004).

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

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., 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

(I) 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

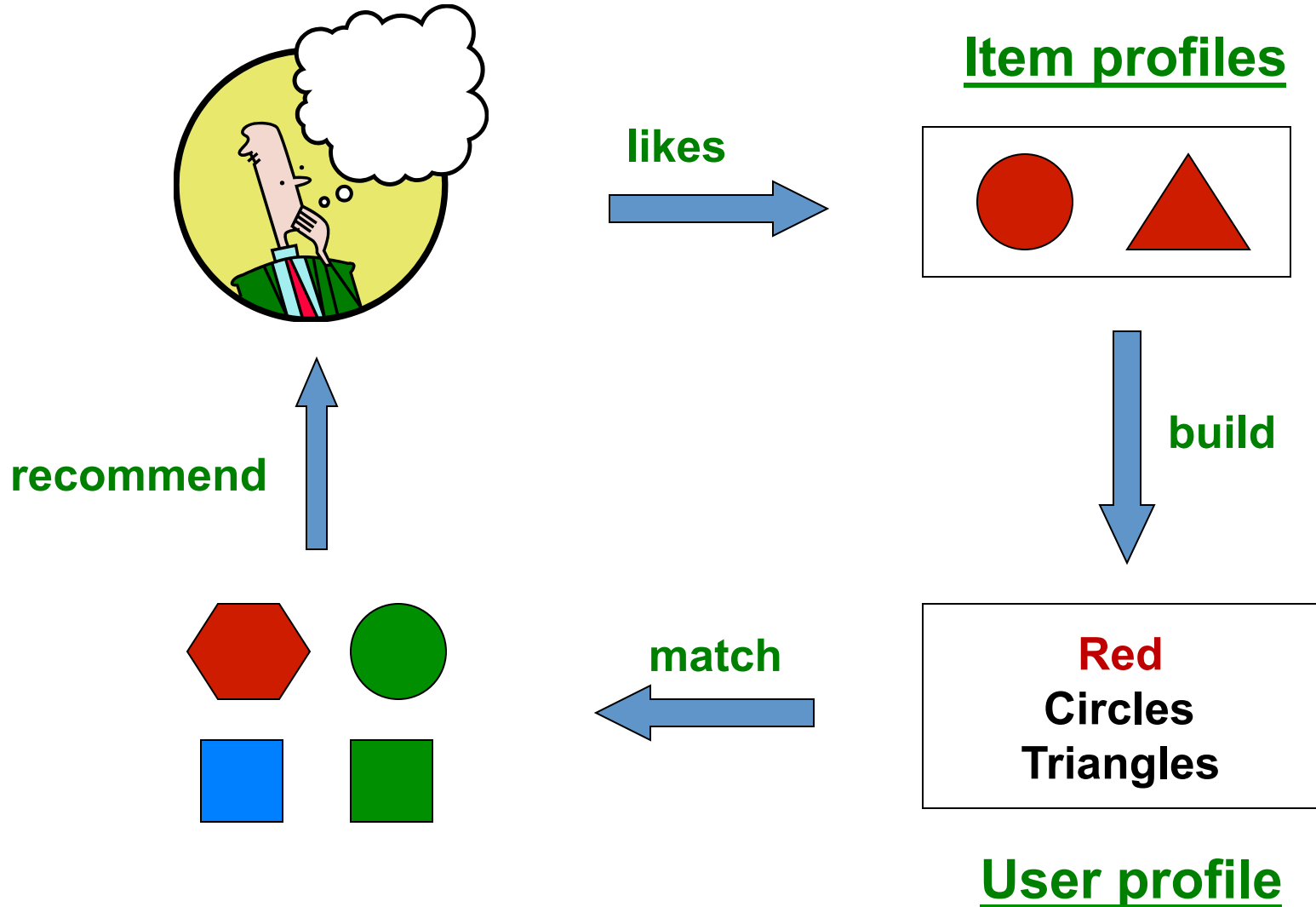
Content-based Recommendations

- **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**

Pros: Content-based Approach

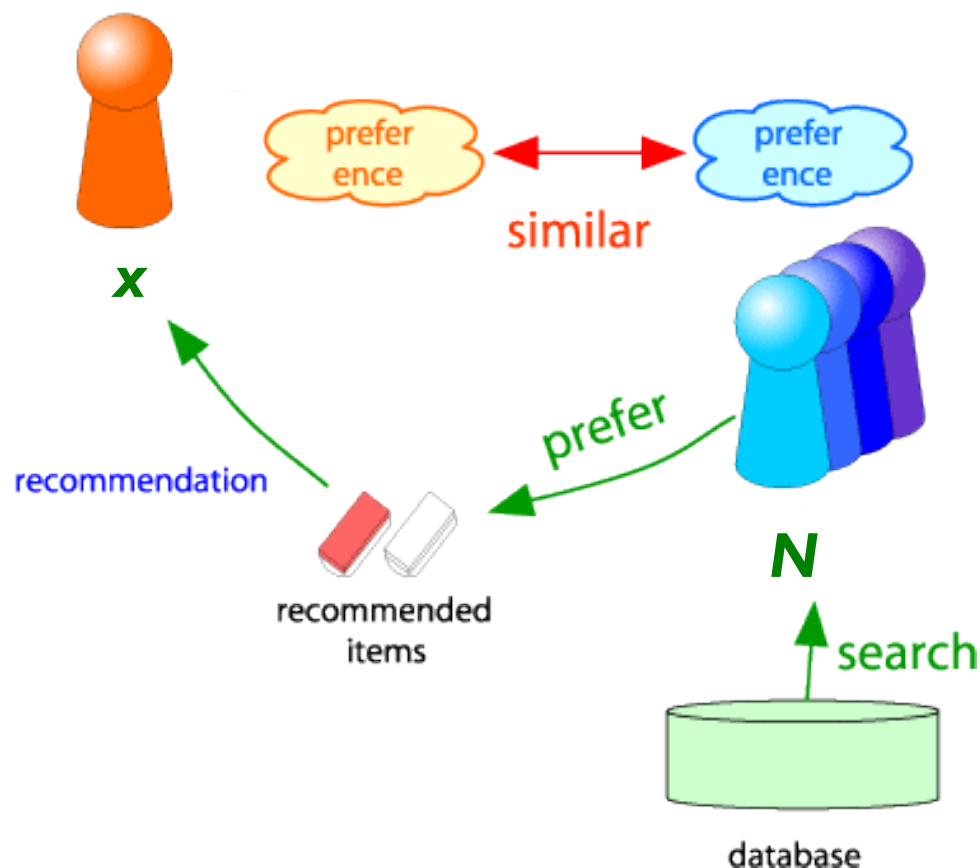
- **+: No need for data on other users**
 - No cold-start or sparsity problems
- **+: Able to recommend to users with unique tastes**
- **+: Able to recommend new & unpopular items**
 - No first-rater problem
- **+: Able to provide explanations**
 - Can provide explanations of recommended items by listing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

- 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

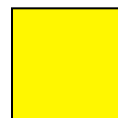


Item-Item CF ($|N|=2$)

		users											
		1	2	3	4	5	6	7	8	9	10	11	12
movies	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											
		1	2	3	4	5	6	7	8	9	10	11	12
movies	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												
		1	2	3	4	5	6	7	8	9	10	11	12	sim(1,m)
movies	1	1		3		?	5			5		4		1.00
	2			5	4			4			2	1	3	-0.18
	<u>3</u>	2	4		1	2		3		4	3	5		<u>0.41</u>
	4		2	4		5			4			2		-0.10
	5			4	3	4	2					2	5	-0.31
	<u>6</u>	1		3		3			2			4		<u>0.59</u>

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 CF ($|N|=2$)

		users												sim(1,m)
		1	2	3	4	5	6	7	8	9	10	11	12	
movies	1	1		3		?	5			5		4		1.00
	2			5	4			4			2	1	3	-0.18
	<u>3</u>	2	4		1	2		3		4	3	5		<u>0.41</u>
	4		2	4		5			4			2		-0.10
	5			4	3	4	2					2	5	-0.31
	<u>6</u>	1		3		3			2			4		<u>0.59</u>

Compute similarity weights:

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

Item-Item CF ($|N|=2$)

		users											
		1	2	3	4	5	6	7	8	9	10	11	12
movies	1	1		3		2.6	5			5		4	
	2			5	4			4			2	1	3
	<u>3</u>	2	4		1	2		3		4	3	5	
	4		2	4		5			4			2	
	5			4	3	4	2					2	5
	<u>6</u>	1		3		3			2			4	

Predict by taking weighted average:

$$r_{i,x} = \frac{\sum_{j \in N(i,x)} s_{ij} \cdot r_j}{\sum_{j \in N(i,x)} s_{ij}}$$

$$r_{1,5} = (0.41 \cdot 2 + 0.59 \cdot 3) / (0.41 + 0.59) = 2.6$$

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

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

- 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

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 item-item 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

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 $O(k \cdot |X|)$
 - 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**

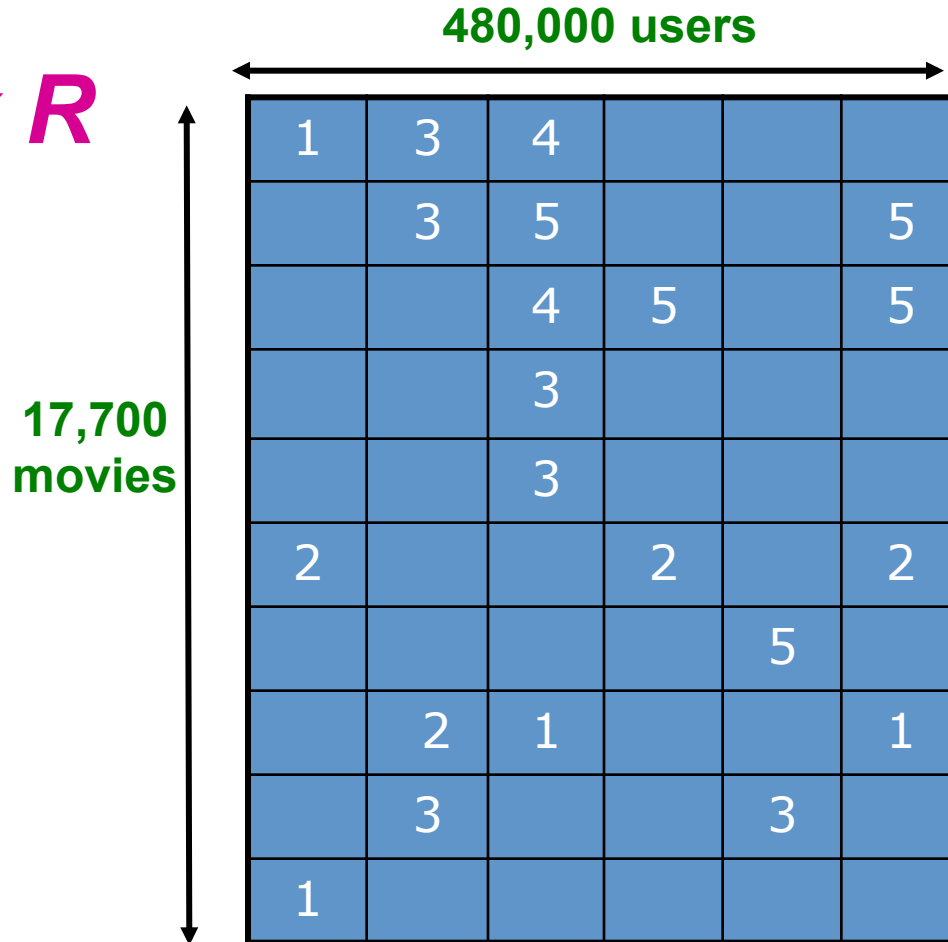
<http://anand.typepad.com/datawocky/2008/03/more-data-usual.html>

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)
 - Netflix's system RMSE: 0.9514
- **Competition**
 - 2,700+ teams
 - **\$1 million** prize for 10% improvement on Netflix

The Netflix Utility Matrix R

Matrix R



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:

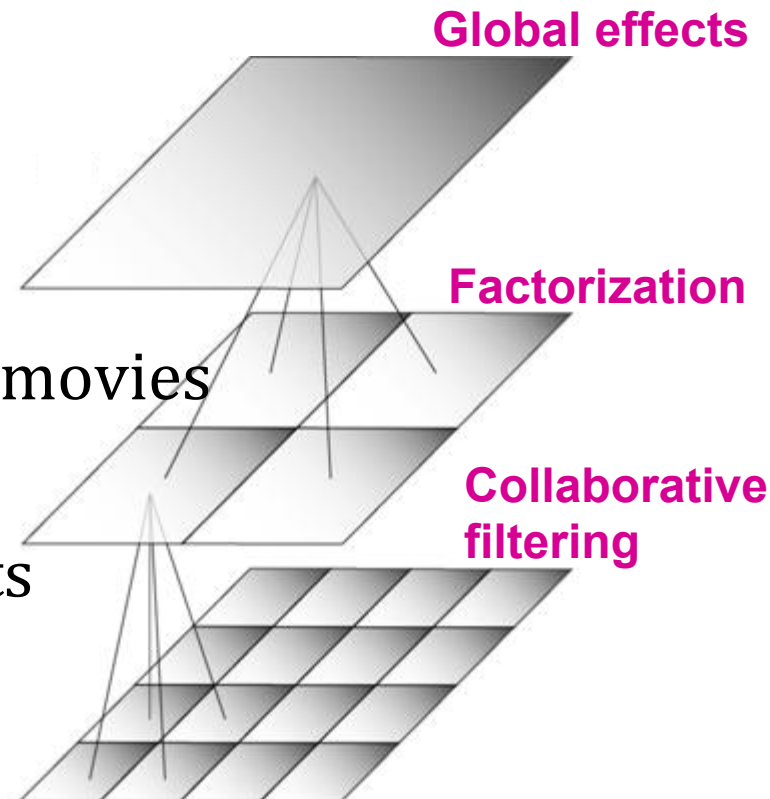
- Overall deviations of users/movies

- 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

- Local neighborhood (CF/NN):

- Joe didn't like related movie *Signs*

⇒ Final estimate:

Joe will rate The Sixth Sense 3.8 stars



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

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

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

Recommendations via Optimization

- **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

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

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

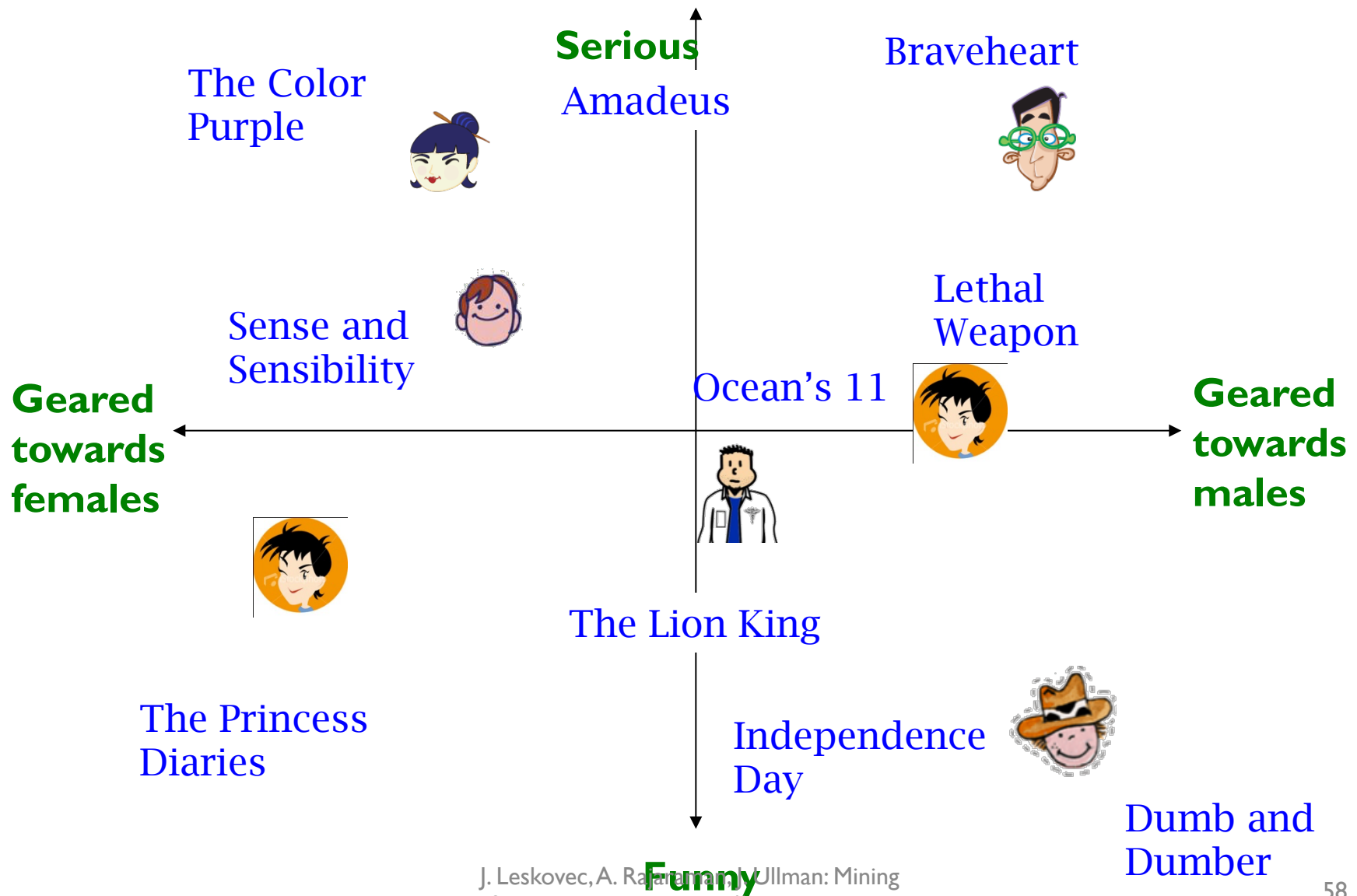
- Find w_{ij} that minimize **SSE on training data!**

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

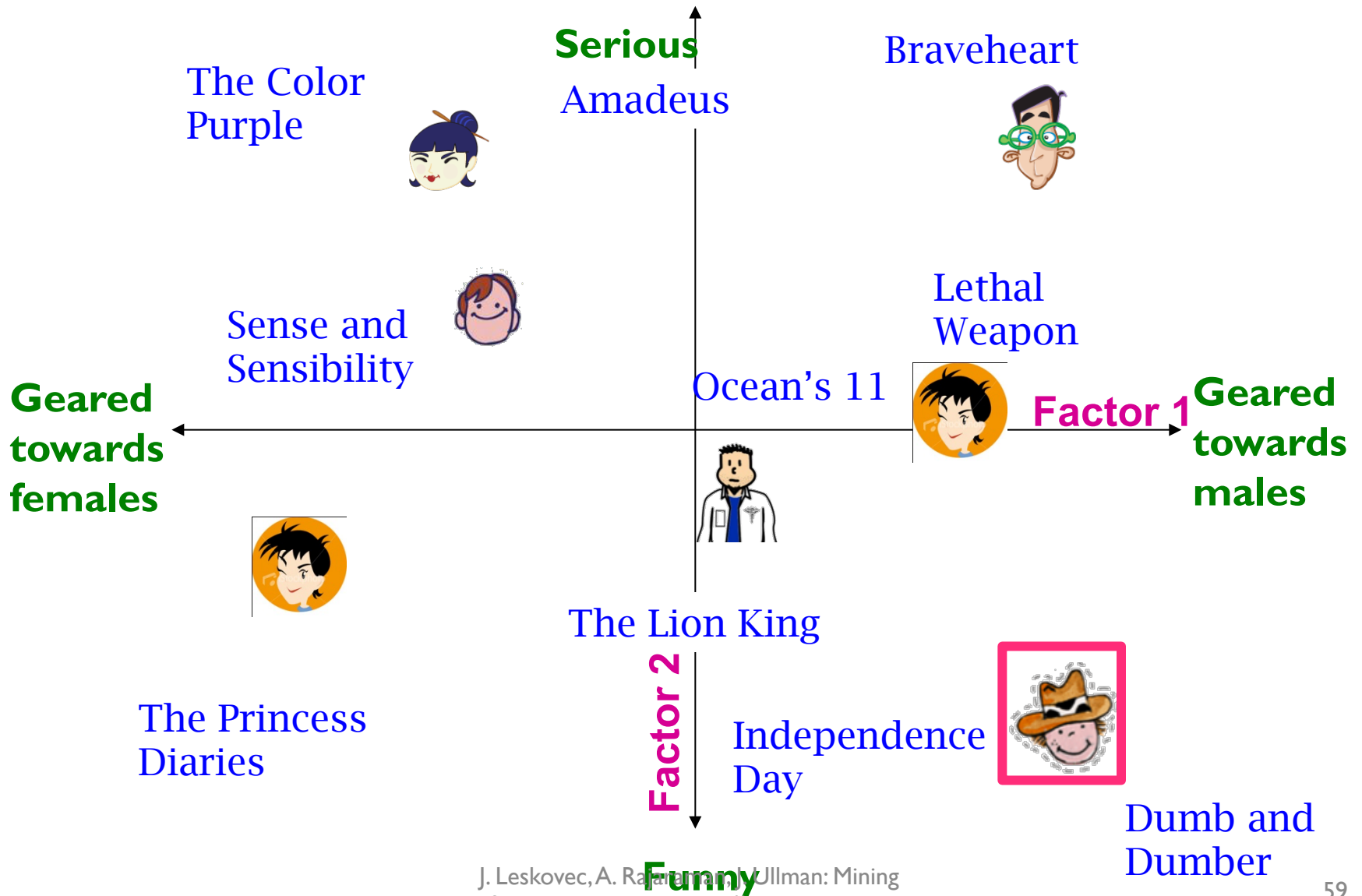
True
rating

- Think of w as a vector of numbers

Latent Factor Models (e.g., SVD)

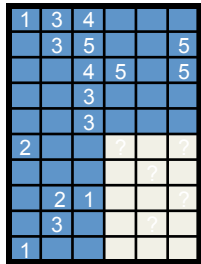


Latent Factor Models



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 it fits too well the training data and thus **not generalizing** well to unseen test data



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

Dealing with Missing Entries

- To solve overfitting we introduce **regularization:**

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

- Allow rich model where there are sufficient data

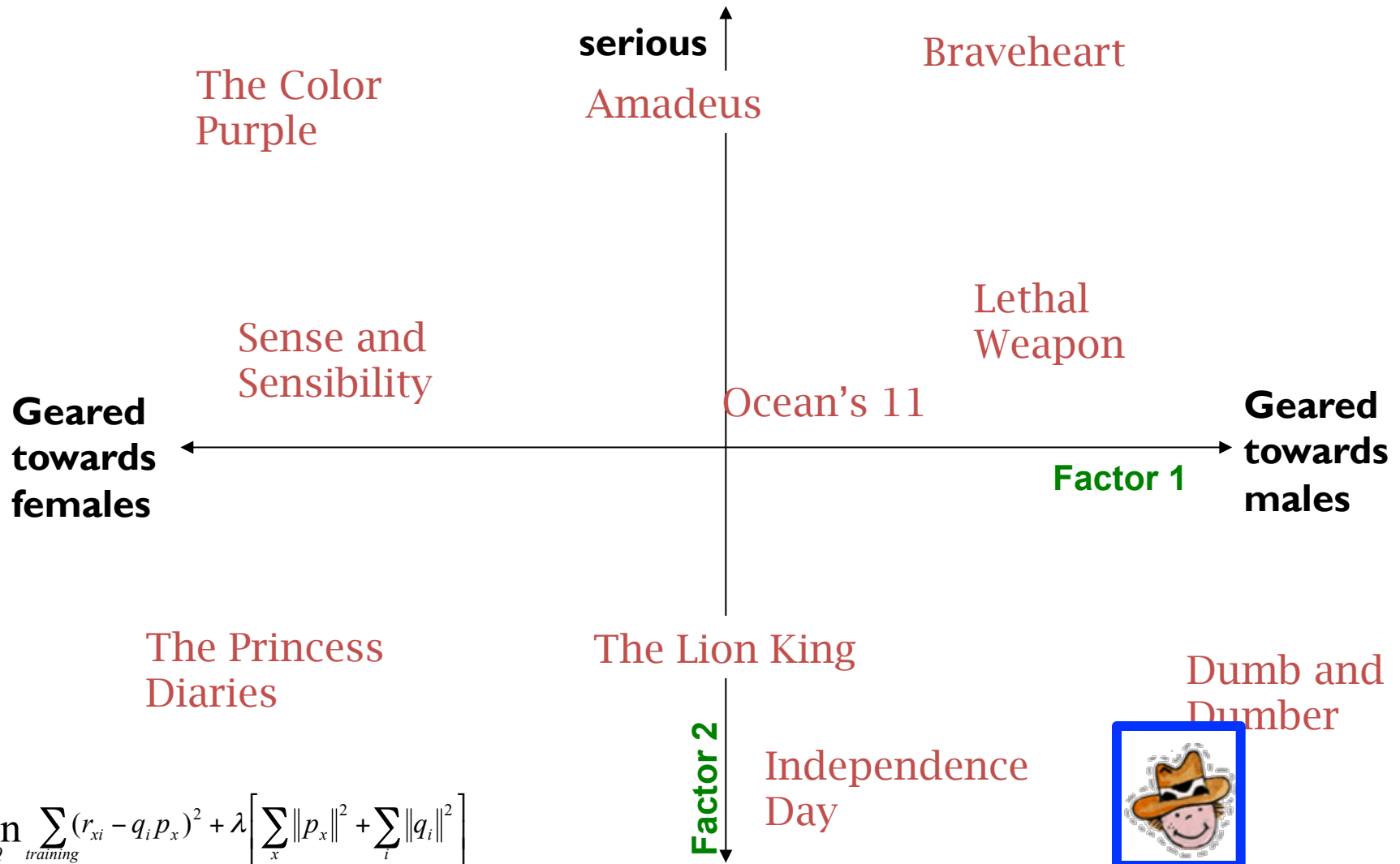
- Shrink aggressively where data are scarce

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

$\lambda_1, \lambda_2 \dots$ user set regularization parameters

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

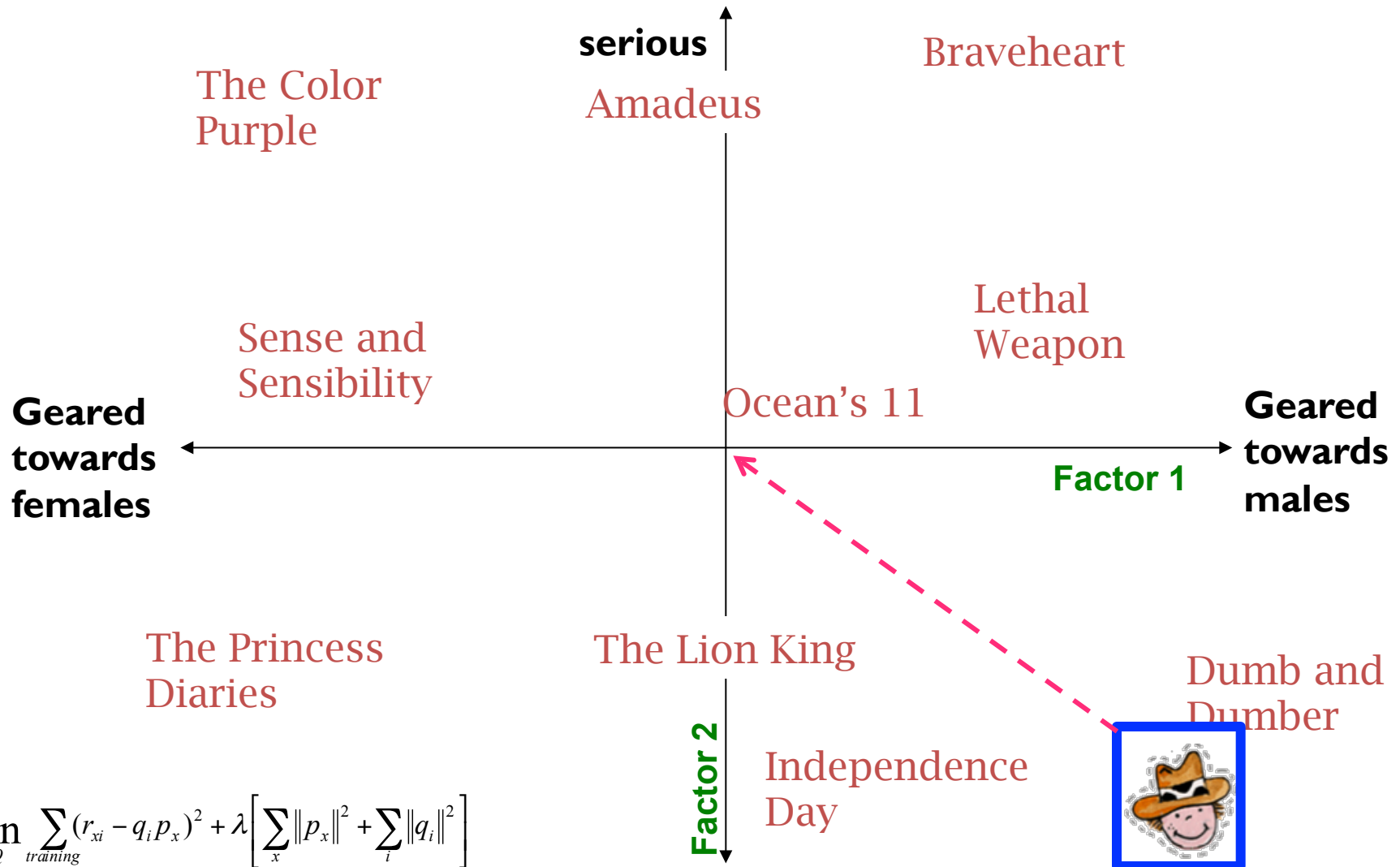
The Effect of Regularization



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

\min_{factors} “error” + λ “length”

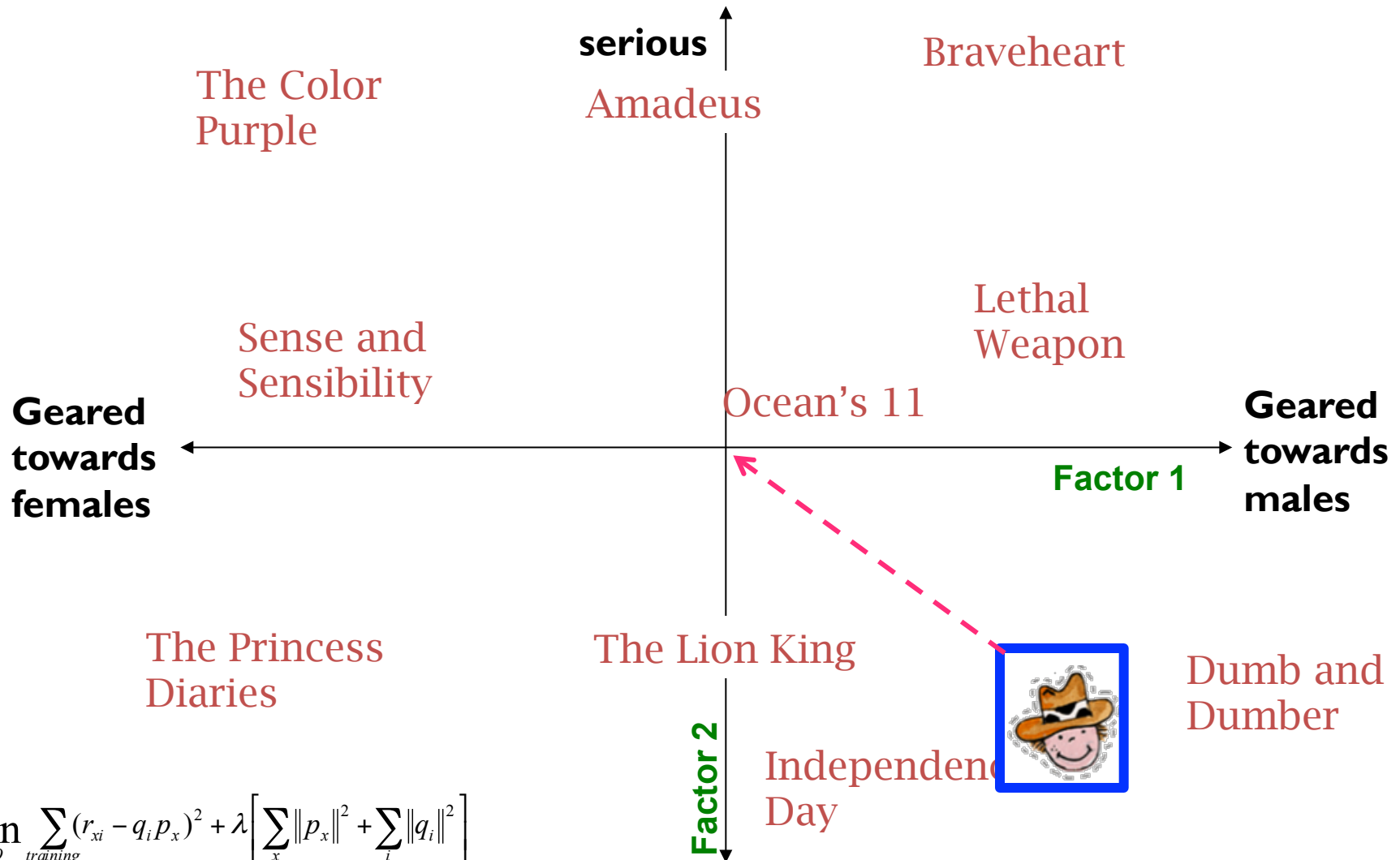
The Effect of Regularization



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

\min_{factors} "error" + λ "length"

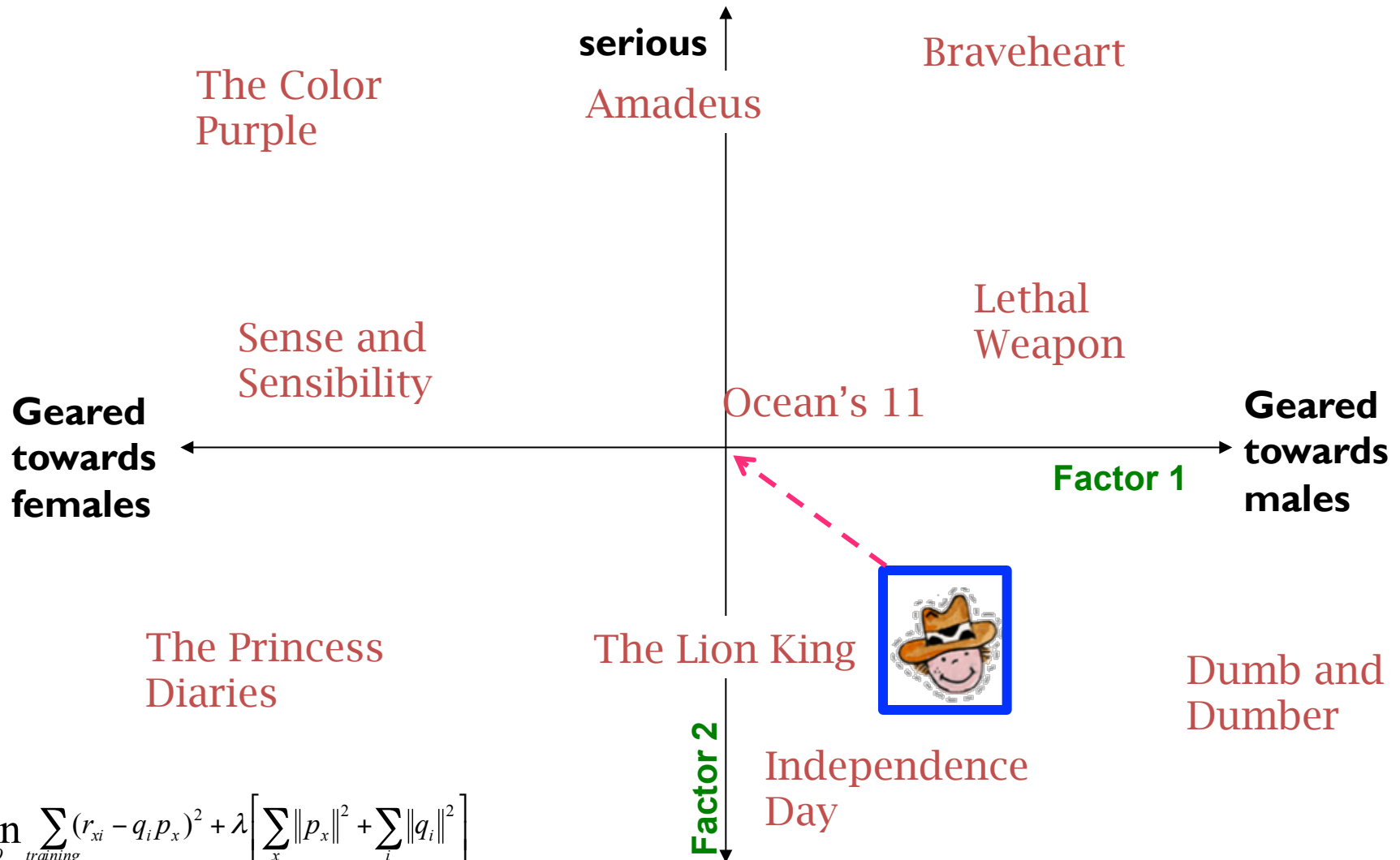
The Effect of Regularization



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

\min_{factors} "error" + λ "length"

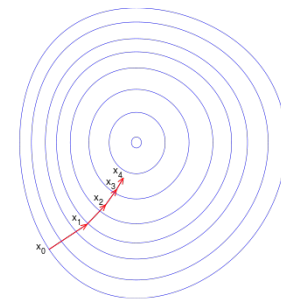
The Effect of Regularization



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

\min_{factors} "error" + λ "length"

Stochastic Gradient Descent



- Want to find matrices P and Q :

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

- **Gradient decent:**

- Initialize P and Q (using SVD, pretend missing ratings are 0)

- Do gradient descent:

- $P \leftarrow P - \eta \cdot \nabla P$
- $Q \leftarrow Q - \eta \cdot \nabla Q$

- where ∇Q is gradient/derivative of matrix Q :
and

- Here is entry f of row q_i of matrix Q

- **Observation: Computing gradients is slow!**

How to compute gradient of a matrix?

Compute gradient of every element independently!

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

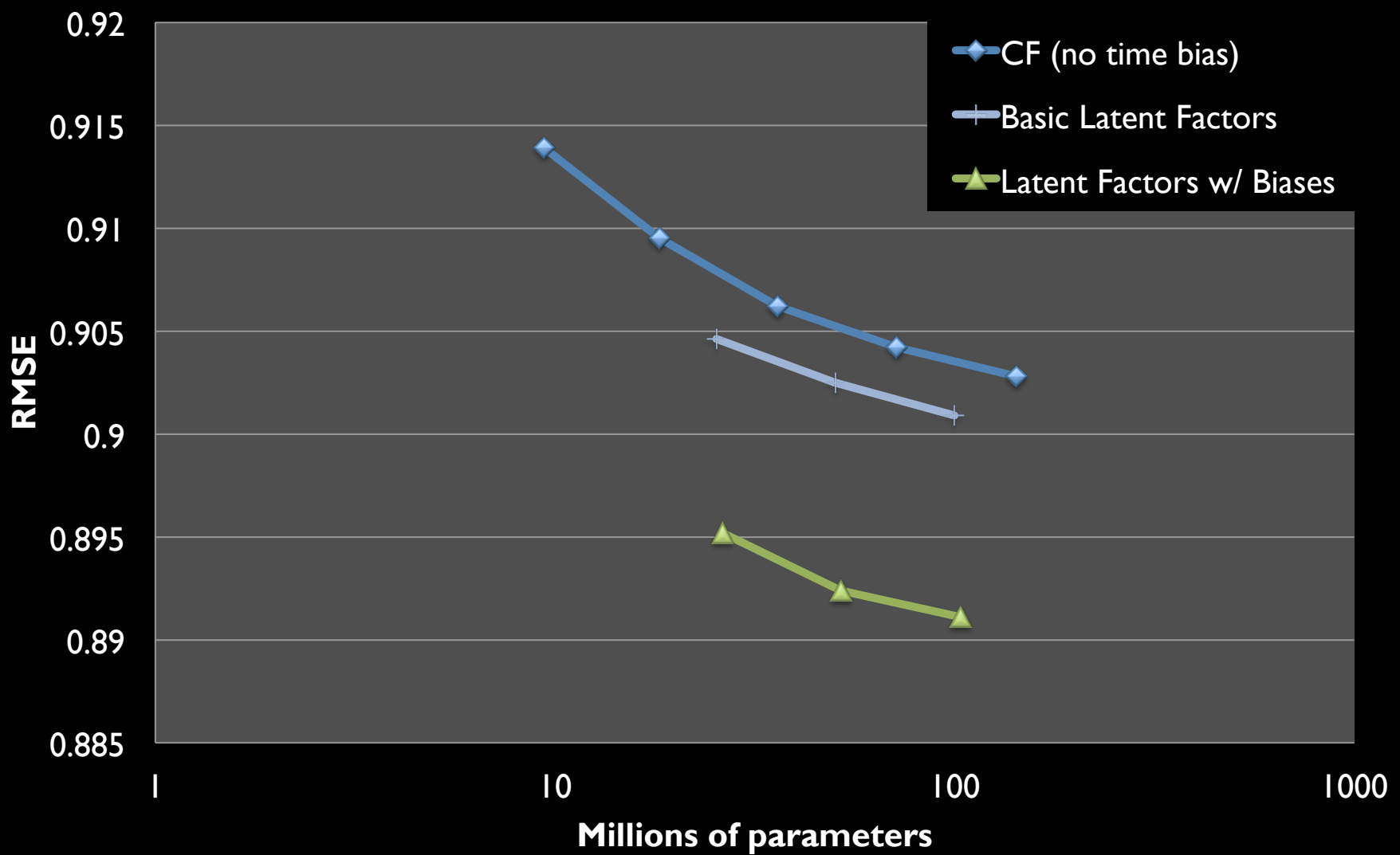
$$+ \left(\lambda_1 \sum_i \|q_i\|^2 + \lambda_2 \sum_x \|p_x\|^2 + \lambda_3 \sum_x \|b_x\|^2 + \lambda_4 \sum_i \|b_i\|^2 \right)$$

regularization

λ is selected via grid-search on a validation set

- Stochastic gradient decent to find parameters
 - **Note:** Both biases b_x, b_i as well as interactions q_i, p_x are treated as parameters (we estimate them)

Performance of Various Methods



Performance of Various Methods

