Data Intelligence

Recommendation-2
Latent Factor Model

U Kang
Seoul National University

In This Lecture

- Learn the weight learning approach for collaborative filtering
- Understand the main idea of latent factor model
- Learn the advanced techniques for latent factor model, including regularization and bias extension

Outline

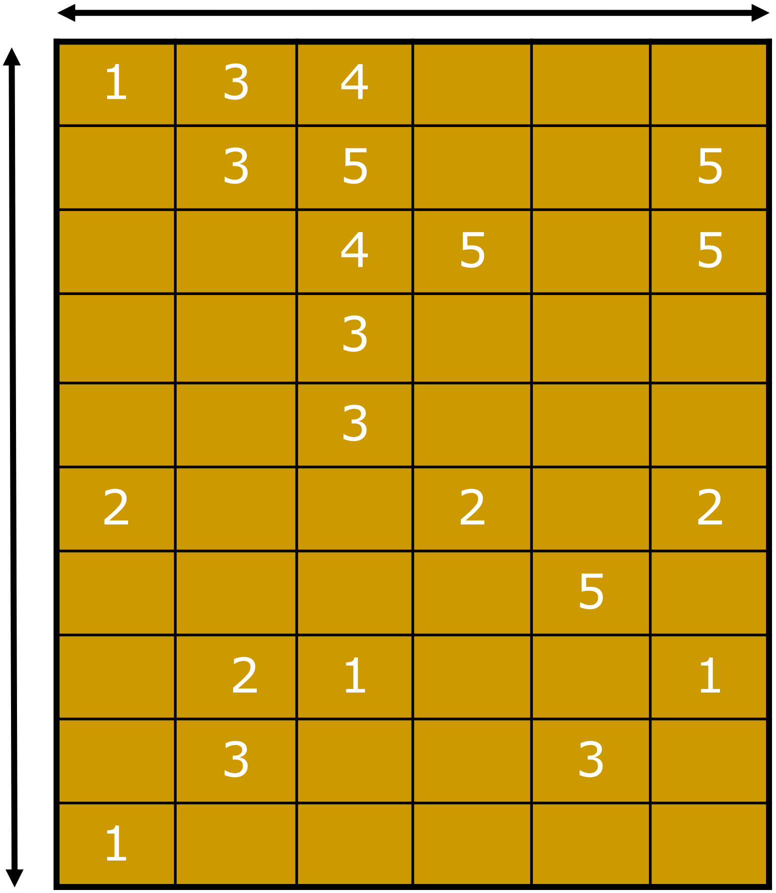
- ➡ ☐ **Netflix Prize; Weight Learning in CF**
- ☐ Latent Factor Model
- ☐ Regularization for LF
- ☐ Bias Extension for LF
- ☐ Netflix Challenge

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



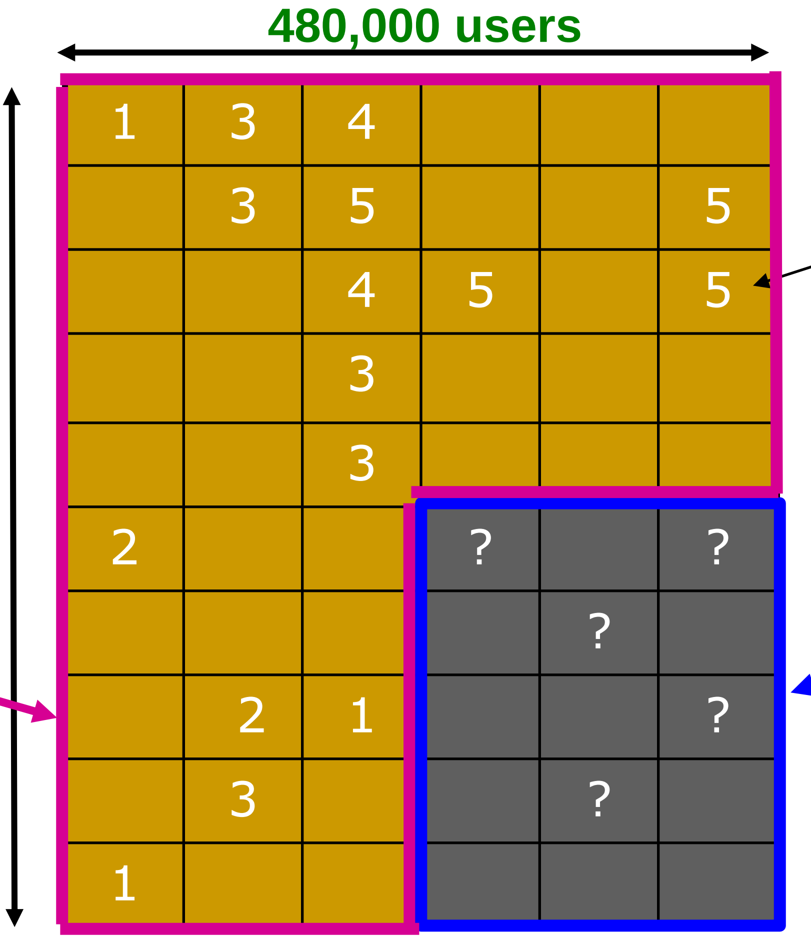
480,000 users

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

Matrix R



480,000 users

17,700 movies

$r_{3,6}$

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

Training Data Set

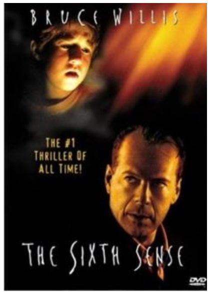
Test Data Set (hidden)

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

BellKor Recommender System

- **The winner of the Netflix Challenge!**
- **Multi-scale modeling of the data:** Combine global modeling of the data, with a refined, local view:
  - **Global:**
    - Overall deviations of users/movies
  - **Local:**
    - CF

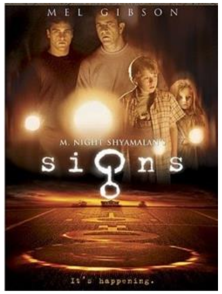
Modeling Local & Global Effects



- **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**

Recap: Collaborative Filtering (CF)

- **Earliest and most popular collaborative filtering method**
  - Infer 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) Taking a weighted average can be restricting

Solution: Instead of s_{ij} use w_{ij} that we learn 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)
    - We allow: $\boldsymbol{\sum_{j \in N(i,x)} w_{ij} \neq 1}$
  - $\boldsymbol{w_{ij}}$ models interaction between pairs of movies (it does not depend on user ***x***)

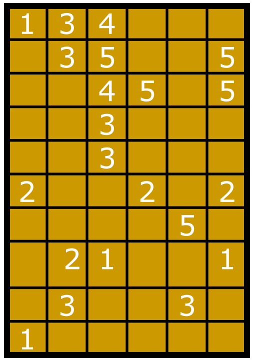
Idea: Interpolation Weights w_{ij}

- $\widehat{r_{xi}} = b_{xi} + \sum_{j \in N(i,x)} w_{ij}\left(r_{xj} - b_{xj}\right)$
- **How to set *w_{ij}*?**
  - Remember, error metric is: $\sqrt{\frac{\sum_{(i,x) \in R}(\hat{r}_{xi} - r_{xi})^2}{|R|}}$ 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

- **Goal:** Make good recommendations
  - ❑ Quantify goodness using **RMSE:**
    Lower RMSE ⇒ better recommendations
  - ❑ Want to make good recommendations on items that user has not yet seen. Very difficult task!
  - ❑ **Let's 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

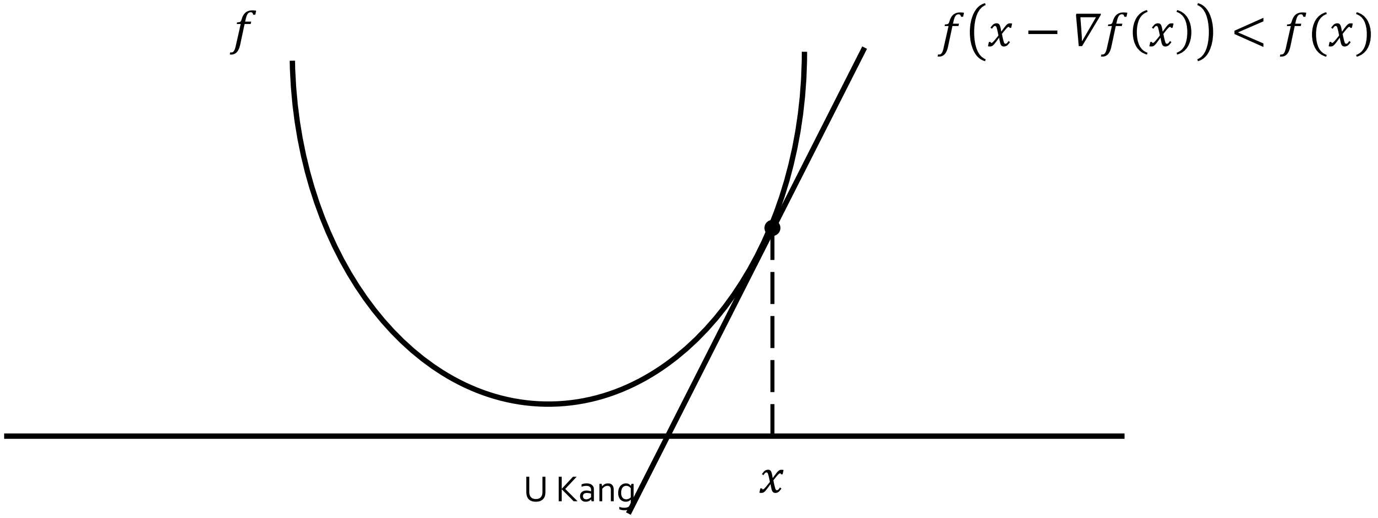
- Find ***w*$_{ij}$** that minimize **SSE** on **training data**!

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

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

Detour: Minimizing a function

- **A simple way to minimize a function $f()$:**
  - Take a gradient $\boldsymbol{\nabla f}$
  - Start at some point $\boldsymbol{x}$ and evaluate $\boldsymbol{\nabla f(x)}$
  - Make a step in the reverse direction of the gradient: $\boldsymbol{x = x - \eta \nabla f(x)}$. This is called *gradient descent*.
  - Repeat until converged



Interpolation Weights

- **We have the optimization problem, now what?**

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

- **Gradient descent:**
  - **Iterate until convergence:** $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \boldsymbol{\nabla_w J}$ η … learning rate
  - **where $\boldsymbol{\nabla_w J}$ is the gradient:**

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

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

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

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

```
while $|w_{new} - w_{old}| > \varepsilon$:
    $w_{old} = w_{new}$
    $w_{new} = w_{old} - \eta \cdot \nabla w_{old}$
```

Interpolation Weights

- **So far:** $\widehat{r_{xi}} = b_{xi} + \sum_{j \in N(i;x)} w_{ij}\left(r_{xj} - b_{xj}\right)$
  - Weights $\boldsymbol{w_{ij}}$ learned based on their role; **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**

Performance of Various Methods

Global average: 1.1296

User average: 1.0651

Movie average: 1.0533

Netflix: 0.9514

Basic Collaborative filtering: 0.94

CF+Biases+learned weights: 0.91

(Collaborative filtering ++)

Grand Prize: 0.8563

Outline

☑ Netflix Prize; Weight Learning in CF

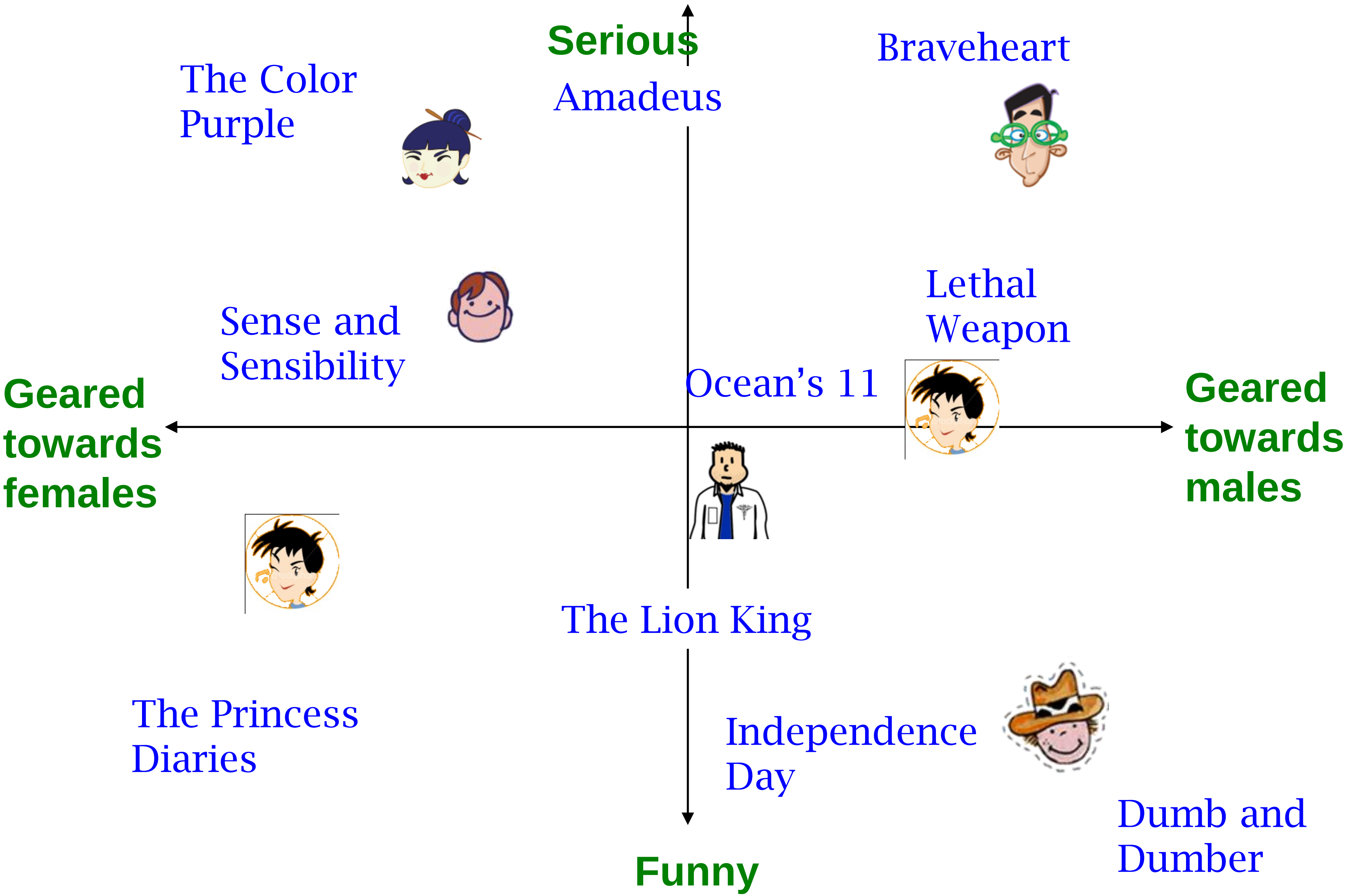
➡ ☐ **Latent Factor Model**

☐ Regularization for LF

☐ Bias Extension for LF

☐ Netflix Challenge

Latent Factor Models (e.g., SVD)



Serious

The Color Purple

Amadeus

Braveheart

Sense and Sensibility

Lethal Weapon

Geared towards females

Ocean's 11

Geared towards males

The Lion King

The Princess Diaries

Independence Day

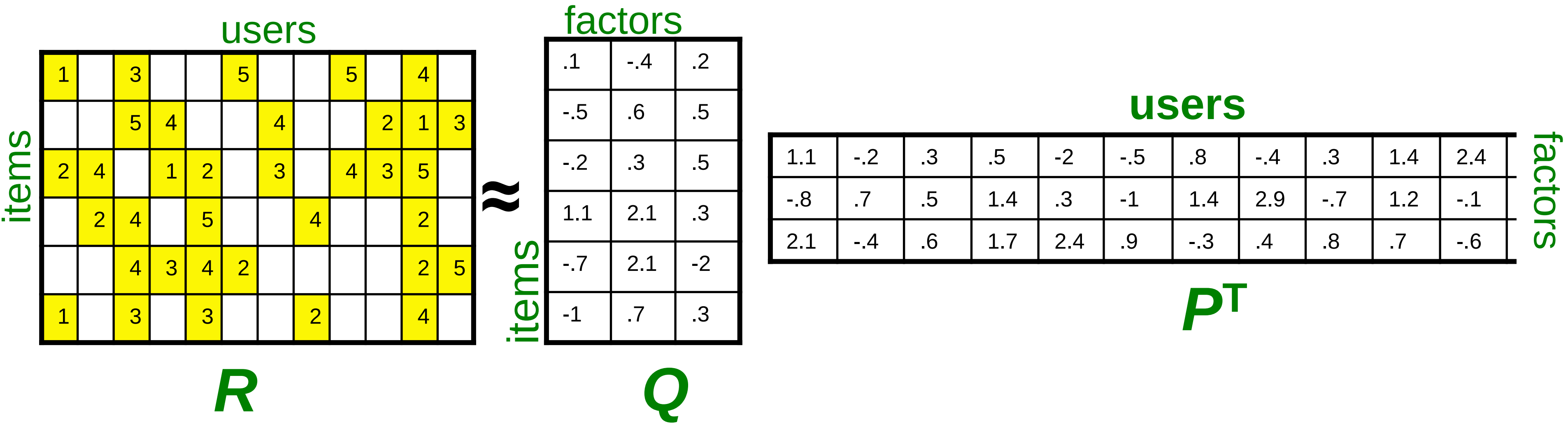
Dumb and Dumber

Funny

Latent Factor Models

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

- "SVD" on Netflix data: $\mathbf{R} \approx \boldsymbol{Q} \cdot \boldsymbol{P}^T$



R

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

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

$\boldsymbol{P}^{\mathbf{T}}$

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

- For now let's assume we can approximate the rating matrix $\boldsymbol{R}$ as a product of "thin" $\boldsymbol{Q} \cdot \boldsymbol{P}^{\mathbf{T}}$
  - ❑ ***R*** has missing entries but let's ignore that for now!
    - Basically, we will 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

items

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	

≈

items

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

factors

Q

•

users

factors

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

P^T

Ratings as Products of Factors

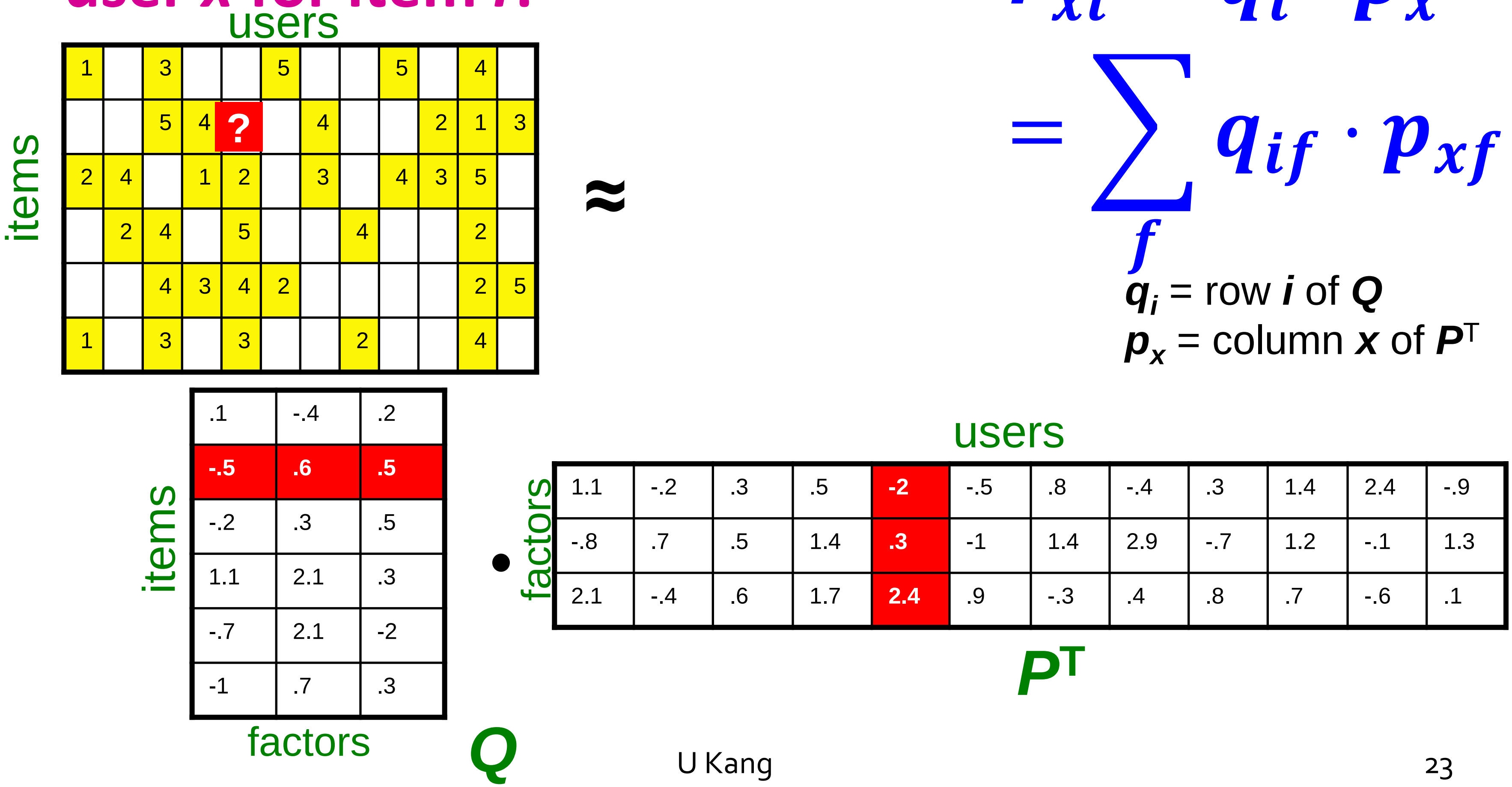
- **How to estimate the missing rating of user *x* for item *i*?**

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

$$= \sum_f q_{if} \cdot p_{xf}$$

q_i = row *i* of *Q*

p_x = column *x* of P^T



users

items

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	

≈

items

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

factors

Q

•

users

factors

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

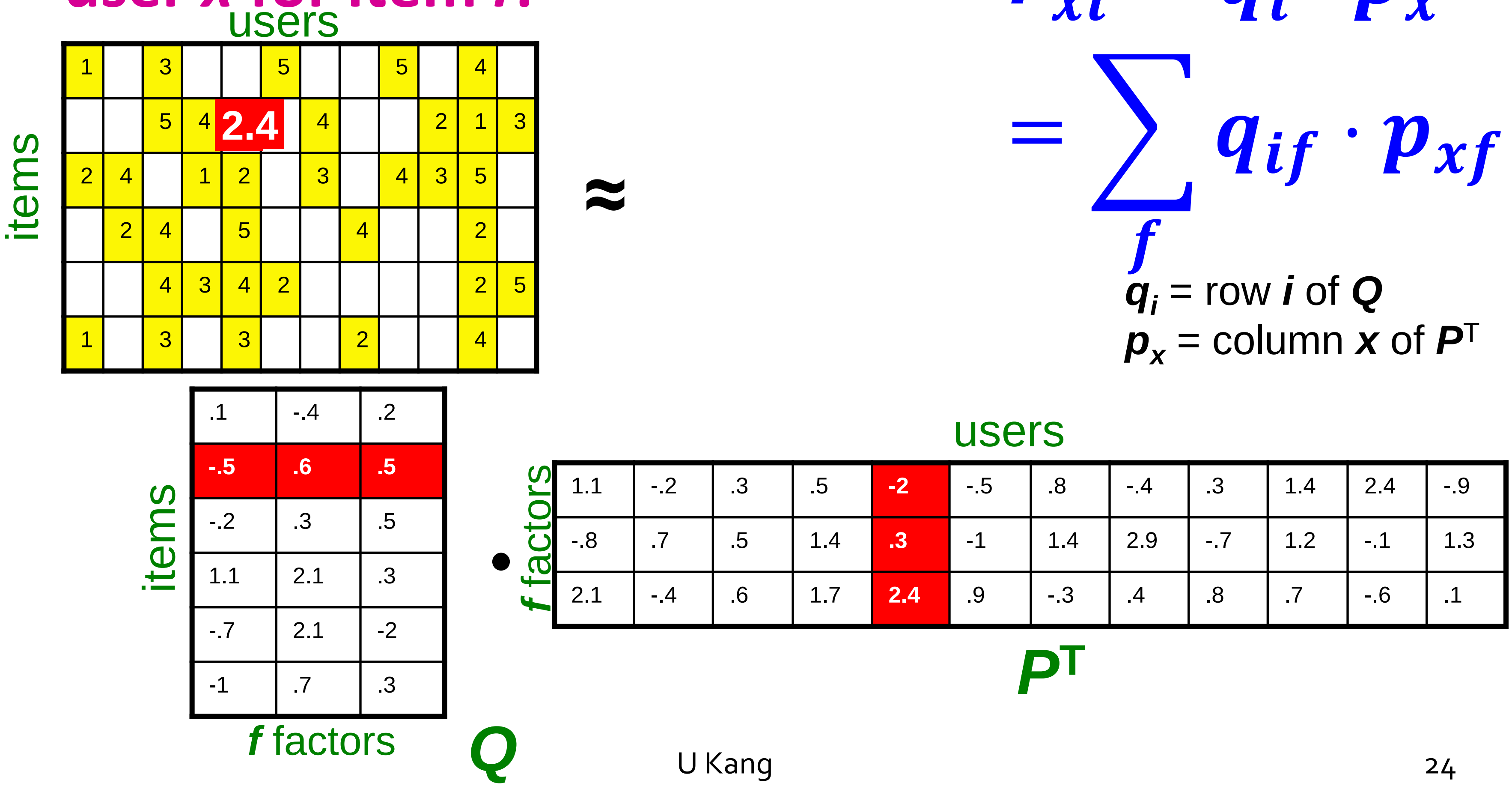
P^T

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

items

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	

≈

items

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

f factors

Q

•

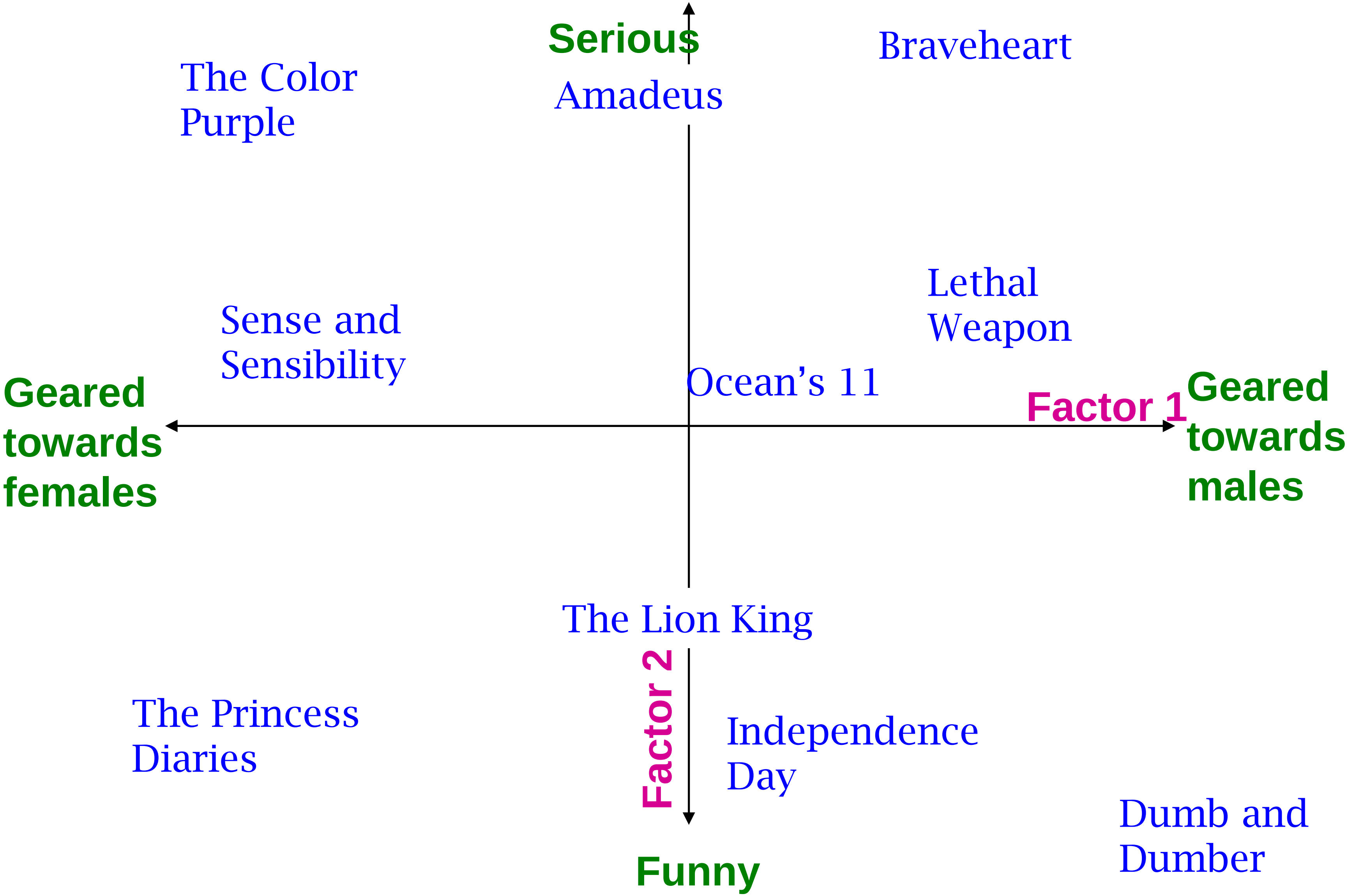
users

f factors

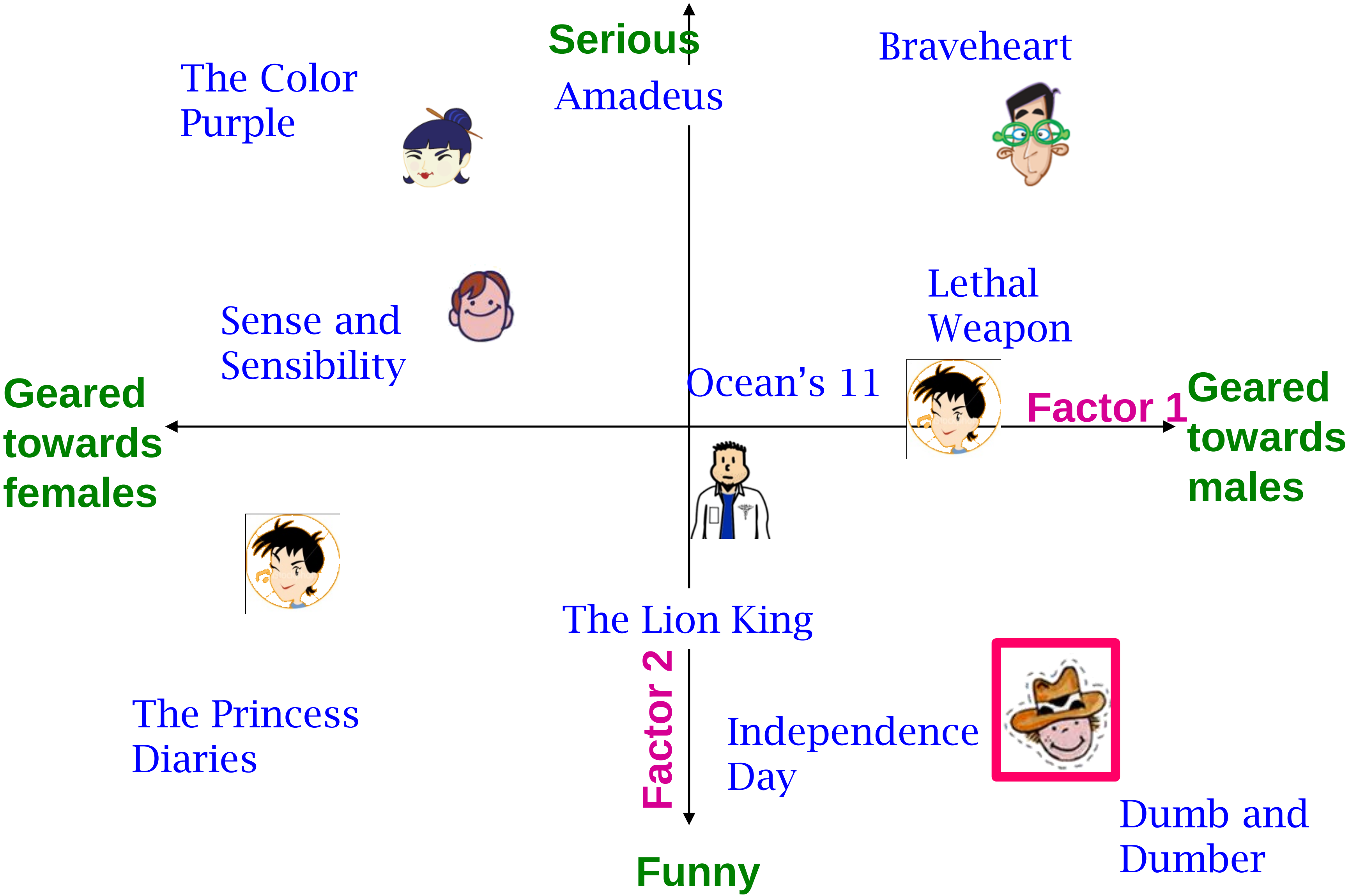
1.1	-.2	.3	.5	**-2**	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	**.3**	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	**2.4**	.9	-.3	.4	.8	.7	-.6	.1

P^T

Latent Factor Models



Latent Factor Models



Serious

The Color Purple

Amadeus

Braveheart

Sense and Sensibility

Lethal Weapon

Ocean's 11

Geared towards females

Factor 1

Geared towards males

The Lion King

Factor 2

The Princess Diaries

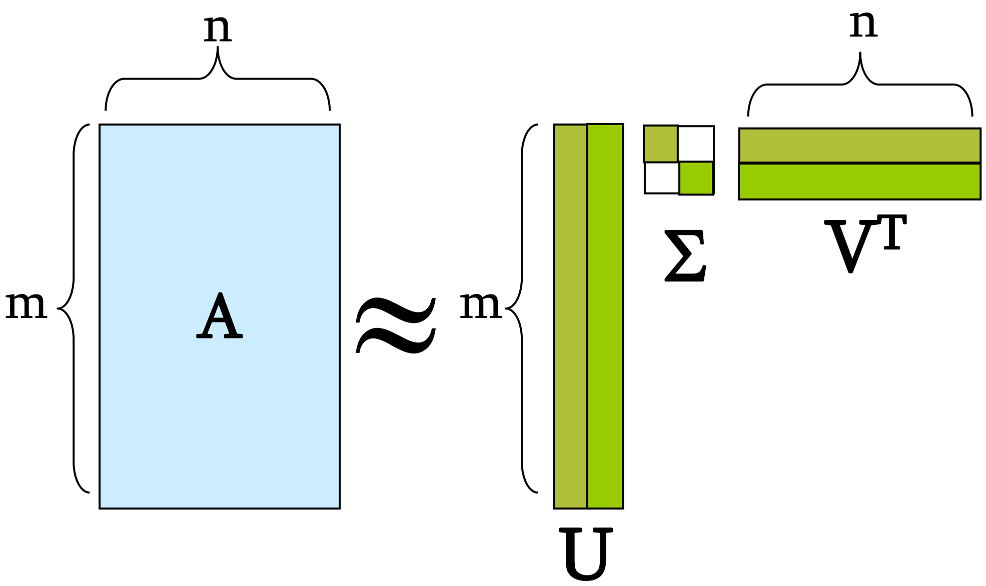
Independence Day

Dumb and Dumber

Funny

Singular Value Decomposition(SVD)

- **SVD:**
  - **A**: Input data matrix
  - **U**: Left singular vecs
  - **V**: Right singular vecs
  - Σ: Singular values



- **So in our case:**
  "SVD" on Netflix data: $R \approx Q \cdot P^T$
  $A = R,\ \ Q = U,\ \ P^T = \Sigma\, V^T$

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

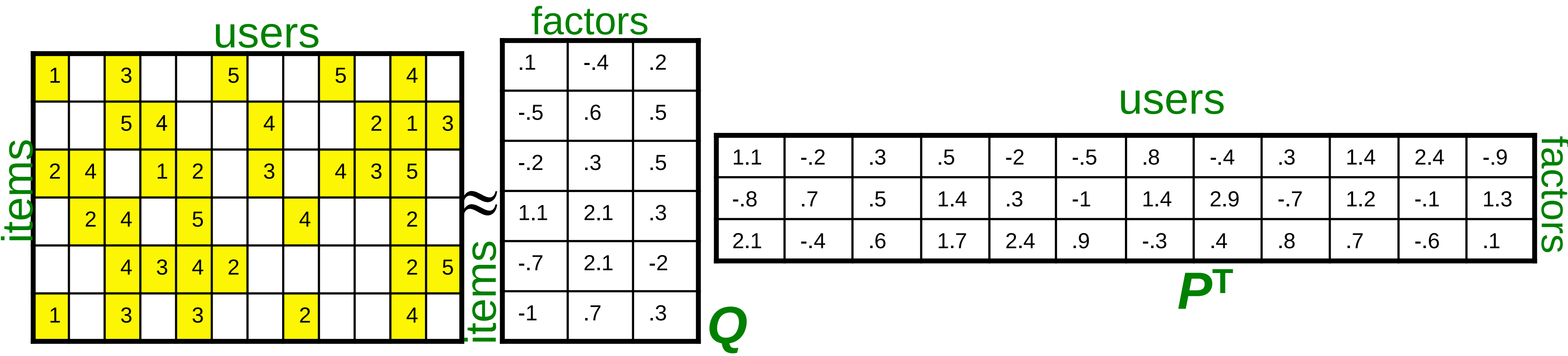
SVD: More Good Stuff

- 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



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	

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

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***
  - $$\min_{P,Q} \sum_{(i,x)\in \mathrm{R}} \left(r_{xi} - q_i \cdot p_x\right)^2 \qquad \hat{r}_{xi} = q_i \cdot p_x$$
  - **Note:**
    - We don't require cols of ***P, Q*** to be orthogonal/unit length
    - ***P, Q*** map users/movies to a latent space
    - The most popular model among Netflix contestants

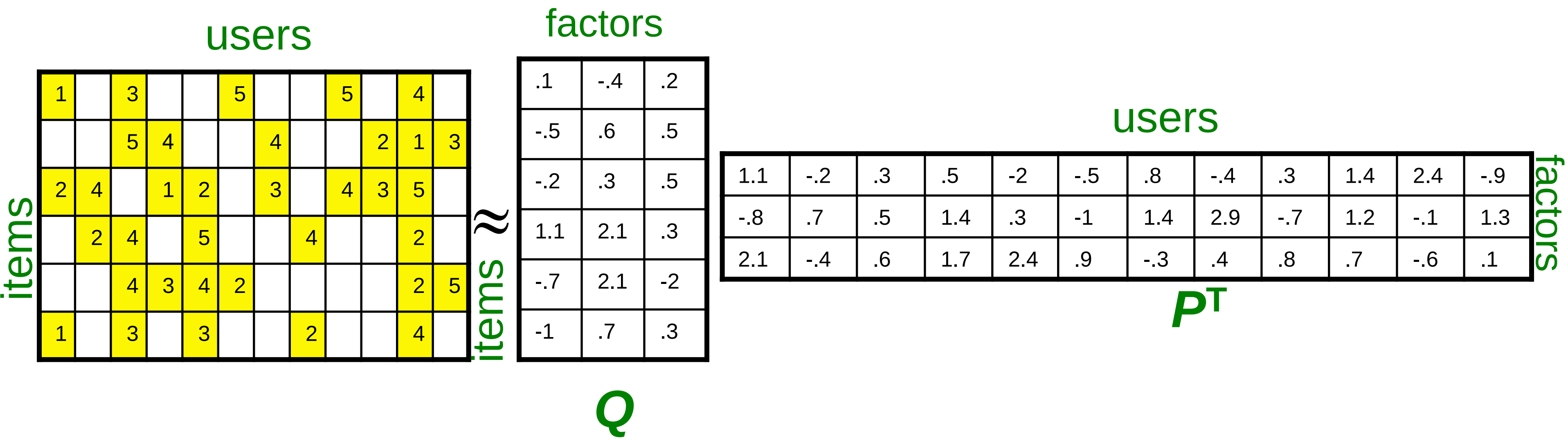
Outline

- ☑ Netflix Prize; Weight Learning in CF
- ☑ Latent Factor Model
- ➡ ☐ **Regularization for LF**
- ☐ Bias Extension for LF
- ☐ Netflix Challenge

Latent Factor Models

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

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



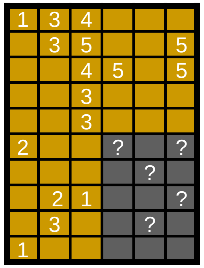
1		3			5			5		4	
		5	4			4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

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

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

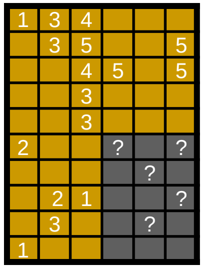
Back to Our Problem

- 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 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



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

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

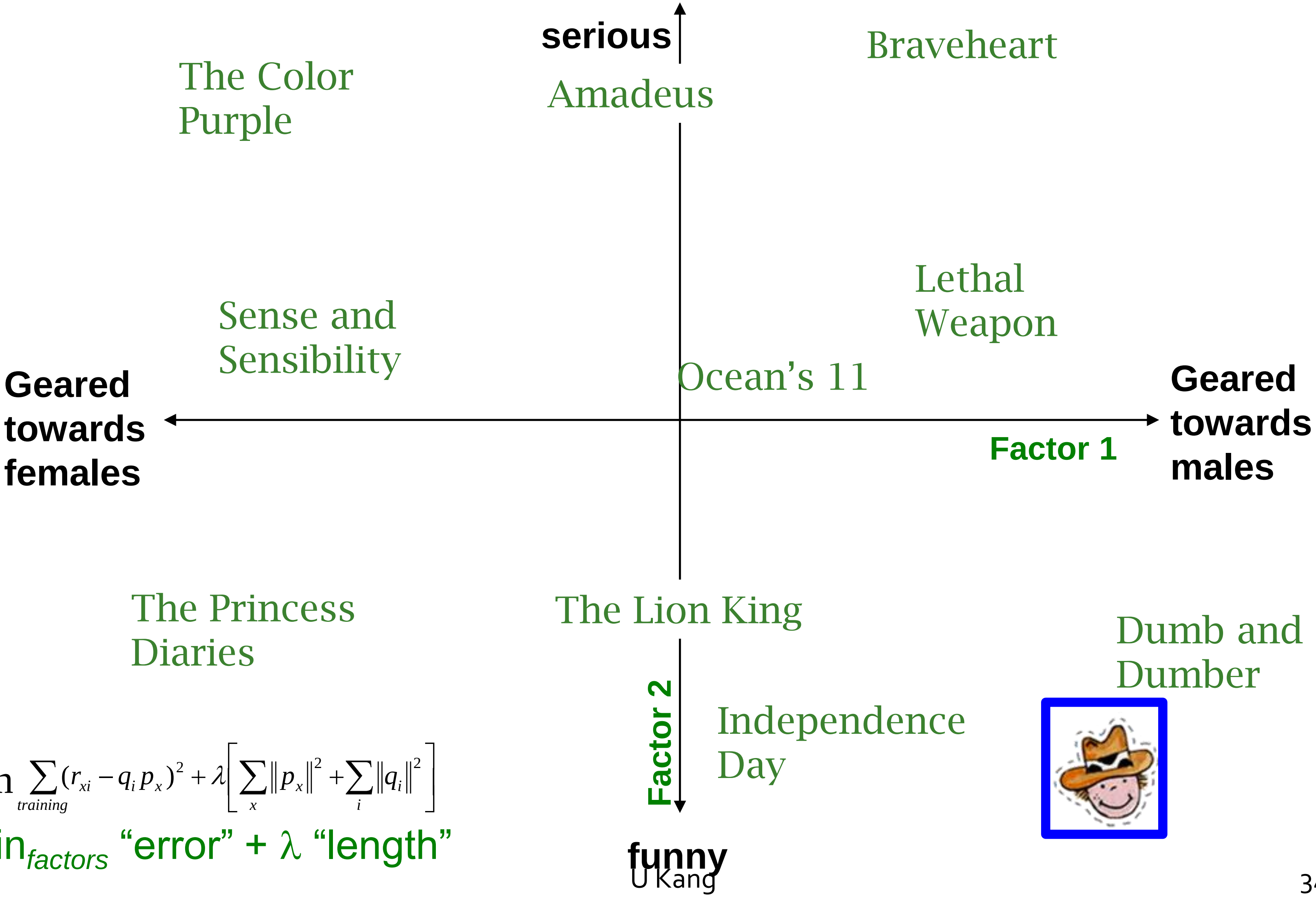
$$\min_{P,Q} \underbrace{\sum_{training} (r_{xi} - q_i p_x)^2}_{\text{“error”}} + \underbrace{\left[\lambda_1 \sum_x \left\| p_x \right\|^2 + \lambda_2 \sum_i \left\| q_i \right\|^2 \right]}_{\text{“length”}}$$

λ_1, λ_2 … user set regularization parameters (≥ 0)

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

U Kang

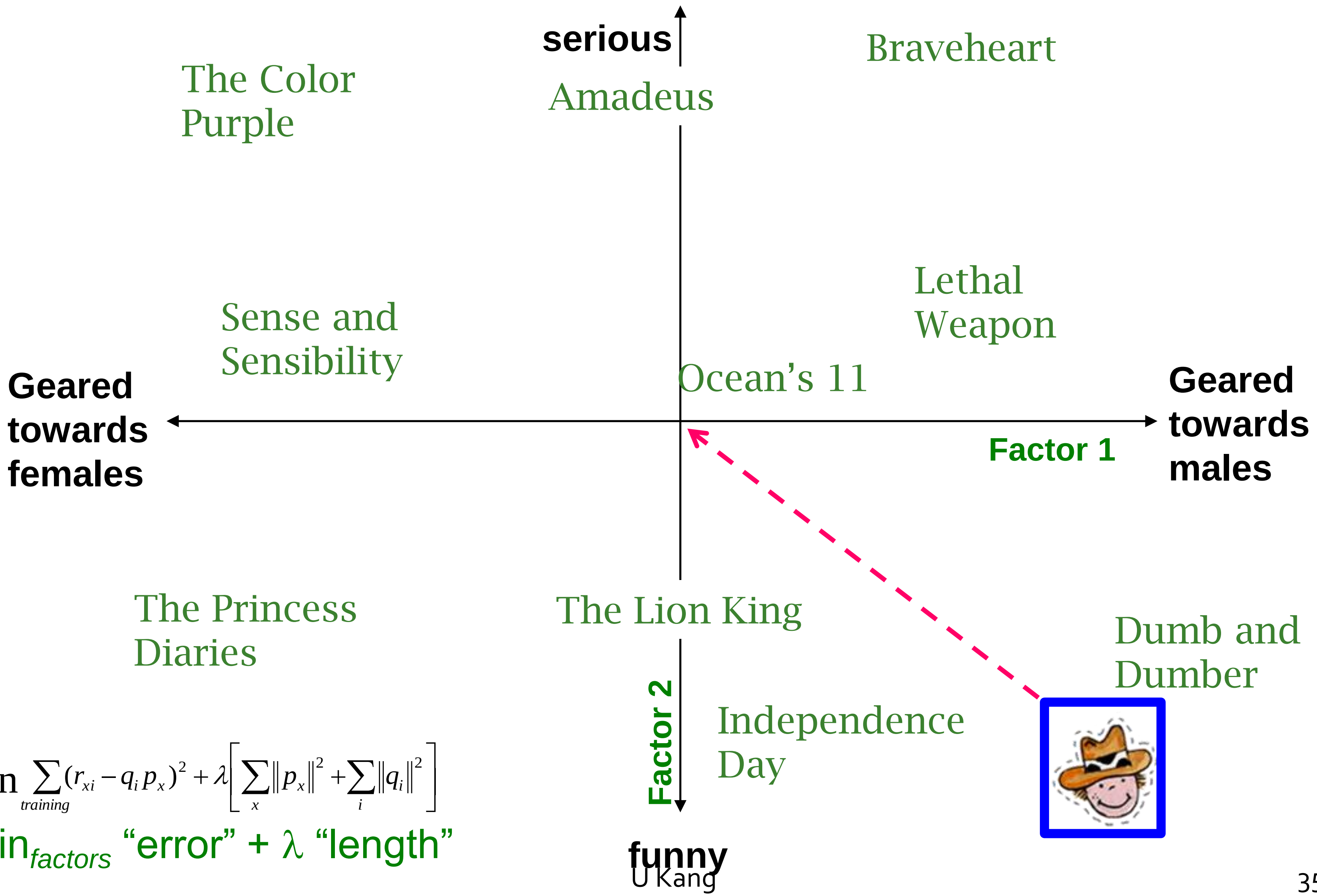
The Effect of Regularization



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

$\min_{factors}$ “error” + λ “length”

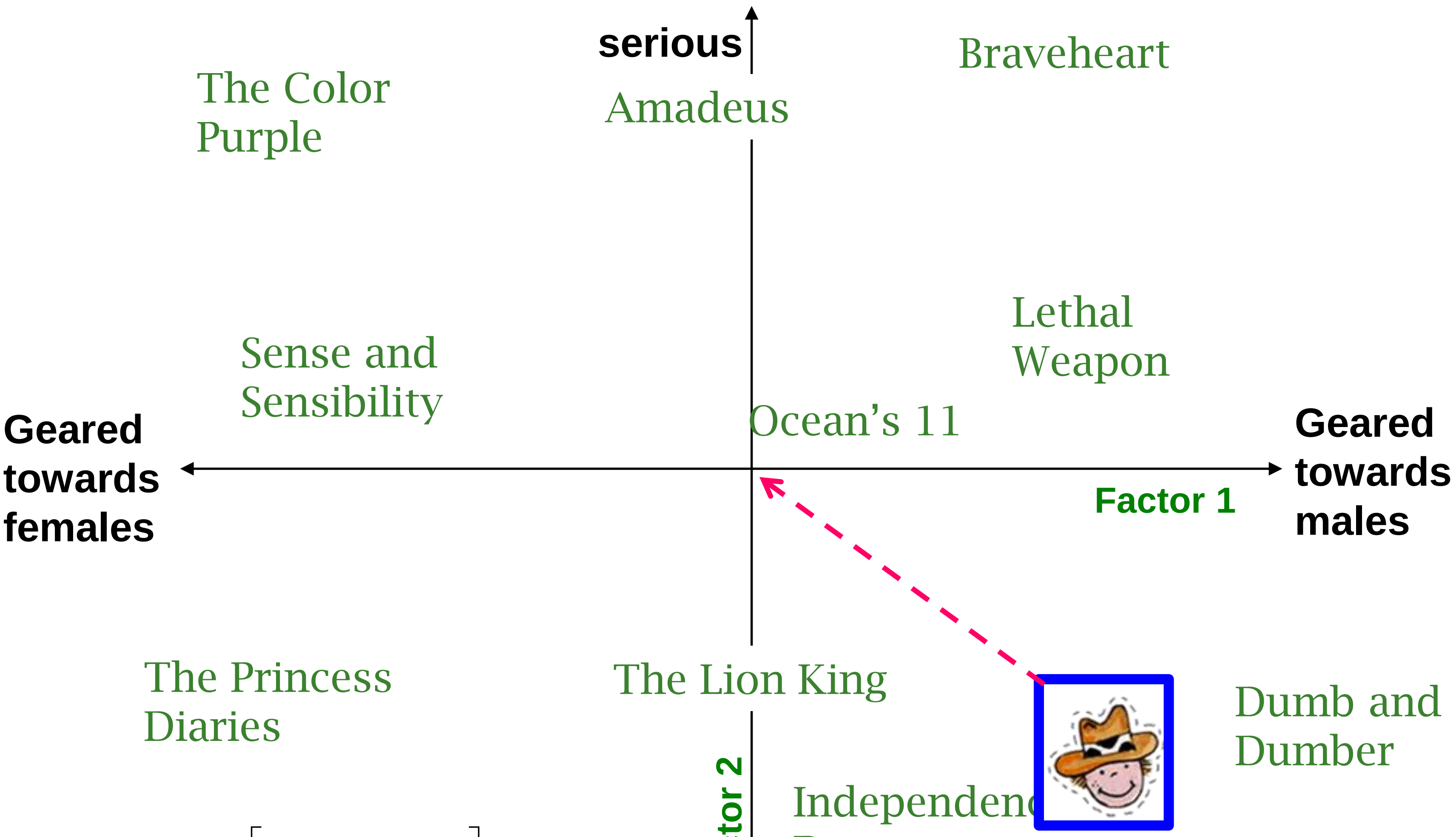
The Effect of Regularization



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

$\min_{factors}$ “error” + λ “length”

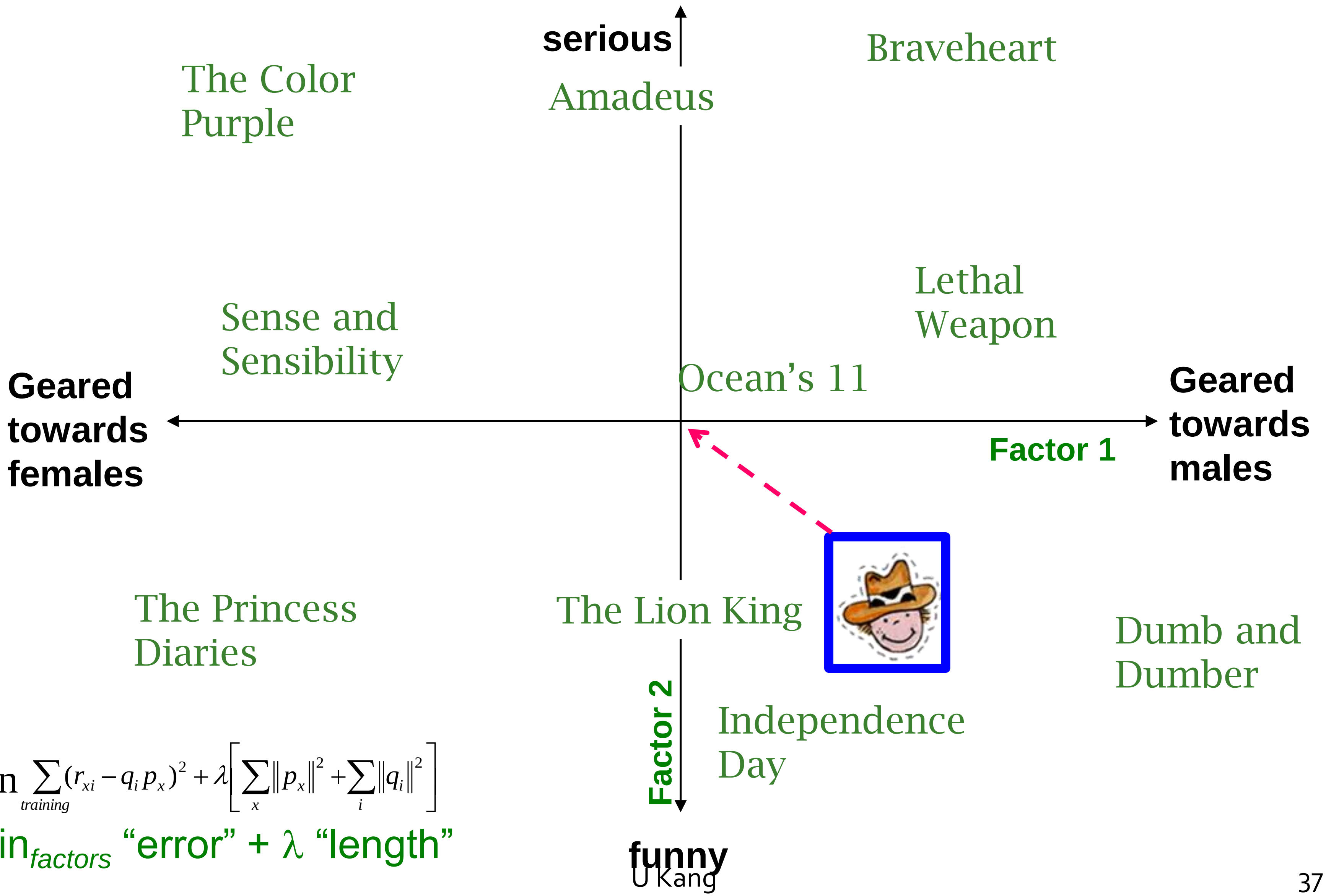
The Effect of Regularization



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

$\min_{factors}$ "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

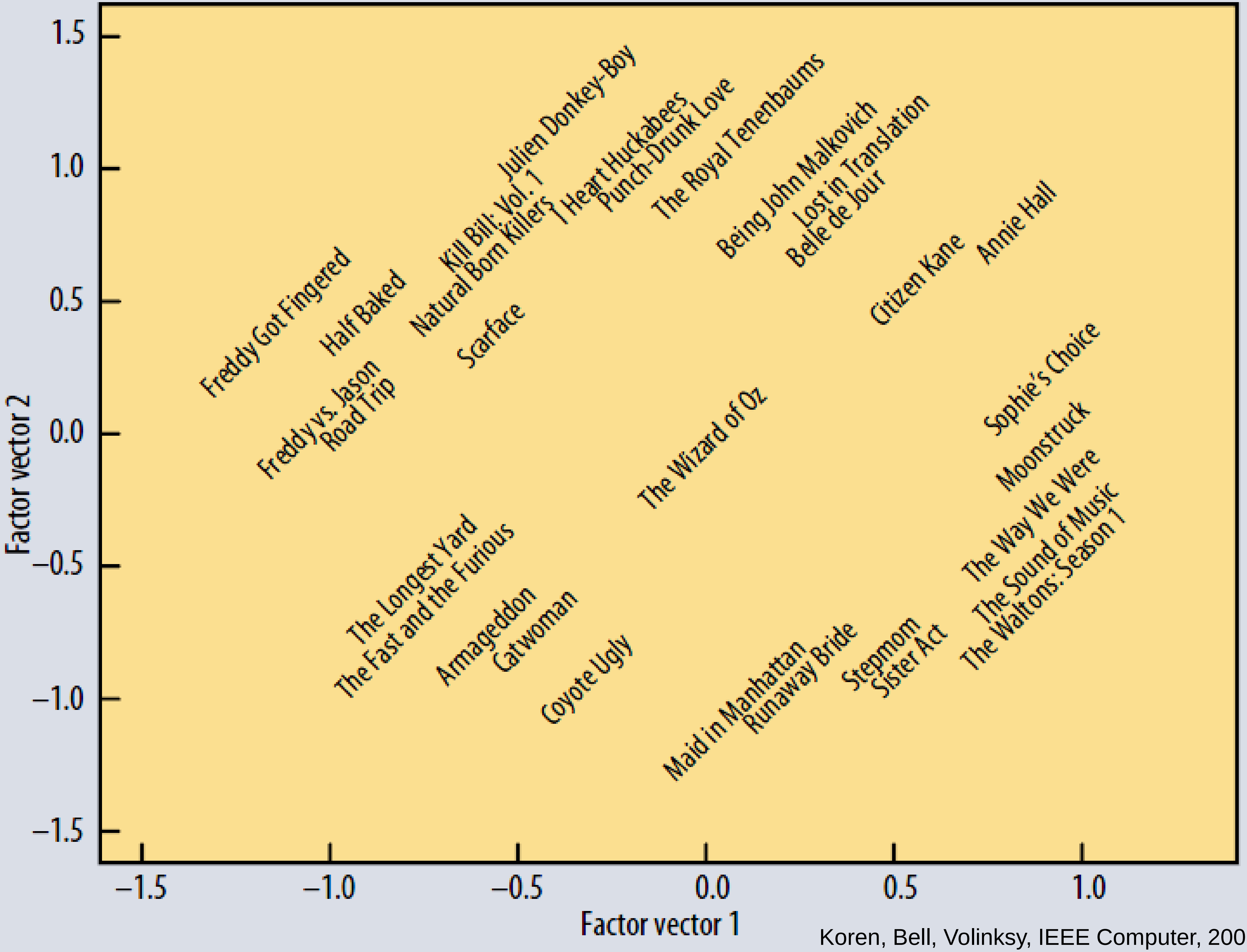
Factor 2

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”

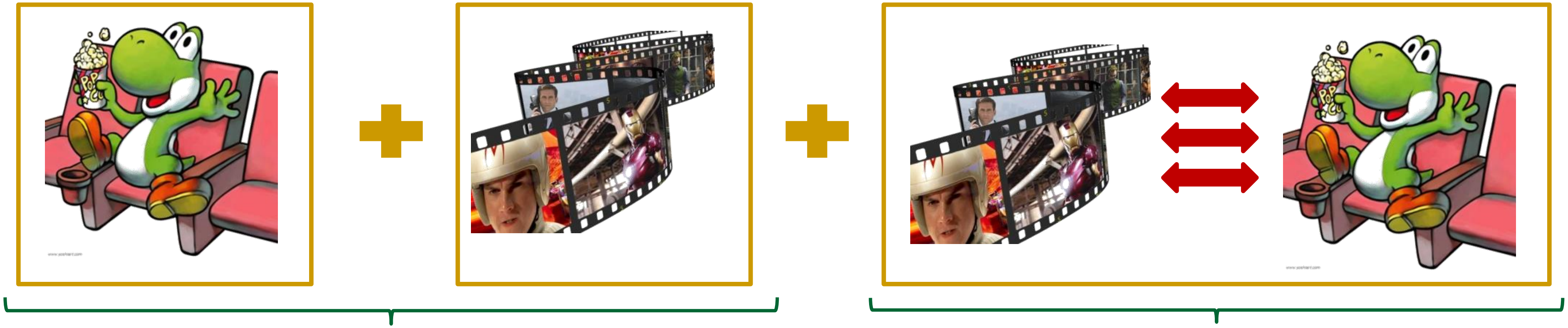


Koren, Bell, Volinksy, IEEE Computer, 2009

Outline

- ☑ Netflix Prize; Weight Learning in CF
- ☑ Latent Factor Model
- ☑ Regularization for LF
- ➡ ☐ **Bias Extension for LF**
- ☐ Netflix Challenge

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

- μ = overall mean rating
- b_x = bias of user x
- b_i = bias of movie 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

- (Recent) popularity of movie ***i***

Putting It All Together

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

- **Example:**
  - Mean rating: ***μ = 3.7***
  - You are a critical reviewer: your ratings are 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 (w/o interaction):
    = 3.7 - 1 + 0.5 = 3.2

Fitting the New Model

- **Solve:**

goodness of fit

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

regularization

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

λ is selected via cross-validation

- **Stochastic gradient descent 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 (we estimate 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

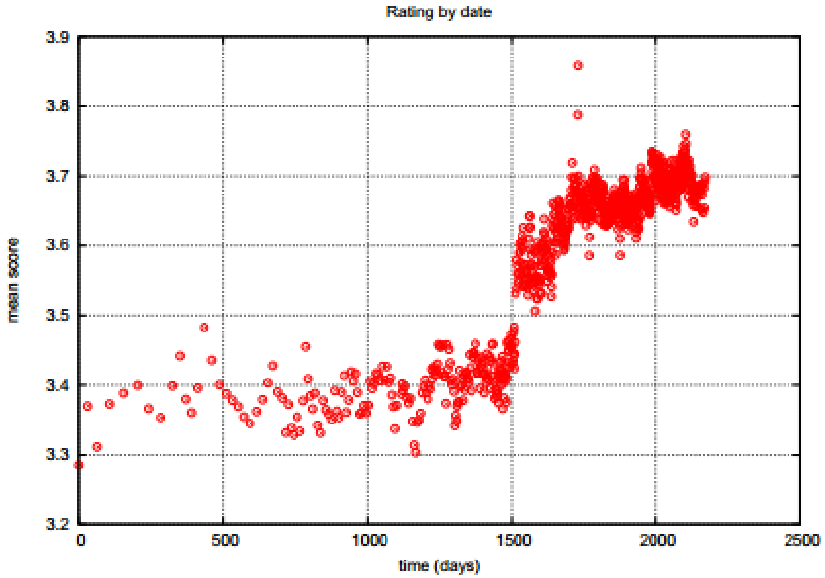
Latent factors: 0.90

Latent factors+Biases: 0.89

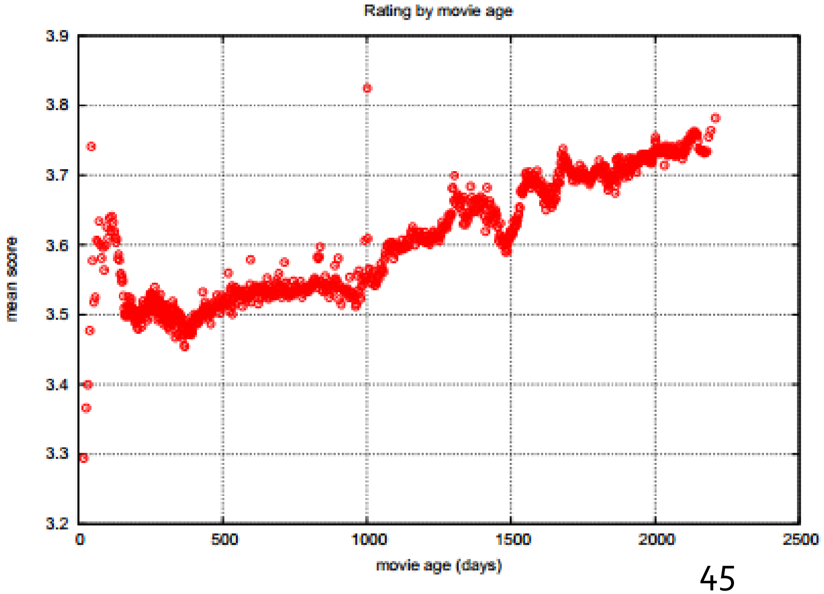
Grand Prize: 0.8563

Temporal Biases Of Users

- **Sudden rise in the average movie rating** (early 2004)
  - ❑ Improvements in Netflix
  - ❑ GUI improvements
  - ❑ Meaning of rating changed
- **Movie age**
  - ❑ Older movies receive higher ratings than newer ones



Rating by date



Rating by movie age

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

Temporal Biases & Factors

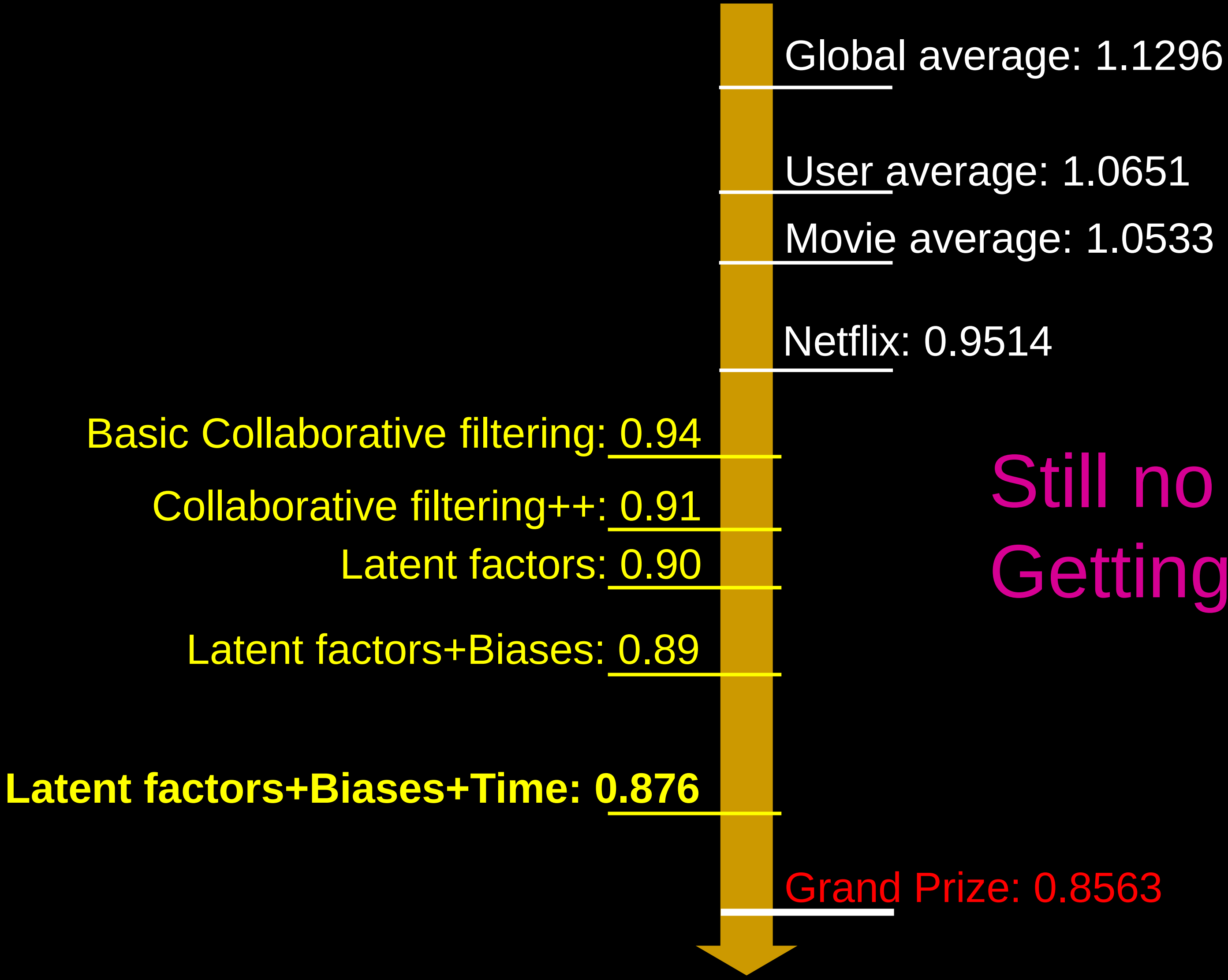
- **Original model:**
  $r_{xi} = \mu + b_x + b_i + q_i \cdot p_x$
- **Add time dependence to biases:**
  $r_{xi} = \mu + b_x(t) + b_i(t) + q_i \cdot p_x$
  - Make parameters $\boldsymbol{b_x}$ and $\boldsymbol{b_i}$ to depend on time
  - **(1)** Parameterize time-dependence by linear trends
    (2)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

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

- **Add temporal dependence to factors**
  - ***$p_x(t)$***… user preference vector on day ***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.

Outline

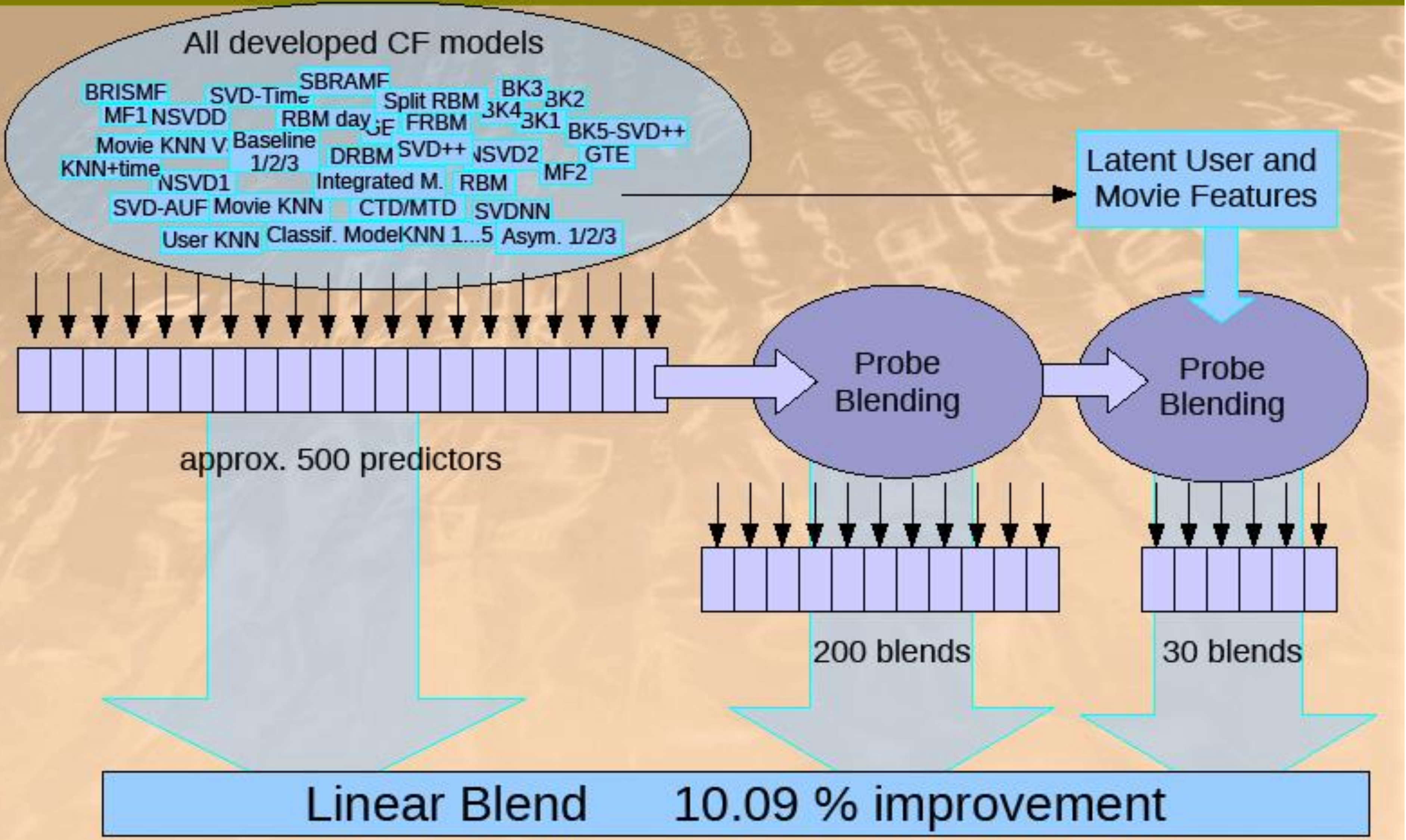
- ☑ Netflix Prize; Weight Learning in CF
- ☑ Latent Factor Model
- ☑ Regularization for LF
- ☑ Bias Extension for LF
- ➡ ☐ **Netflix Challenge**

Final Solution

- Many solutions proposed
  - Baseline
  - Basic collaborative filtering
  - Basic collaborative filtering w/ weight learning
  - Latent factor model
  - Latent factor w/ time bias
  - …
- ‘Blending’ the solutions leads to the best performance
  - Linear combination of N (≥ 500) predictors
  - $\widehat{r_{xi}} = \sum_{k=1}^{N} w_k \cdot pred_k(x,i)$ pred$_k$: k th predictor

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. Mode
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

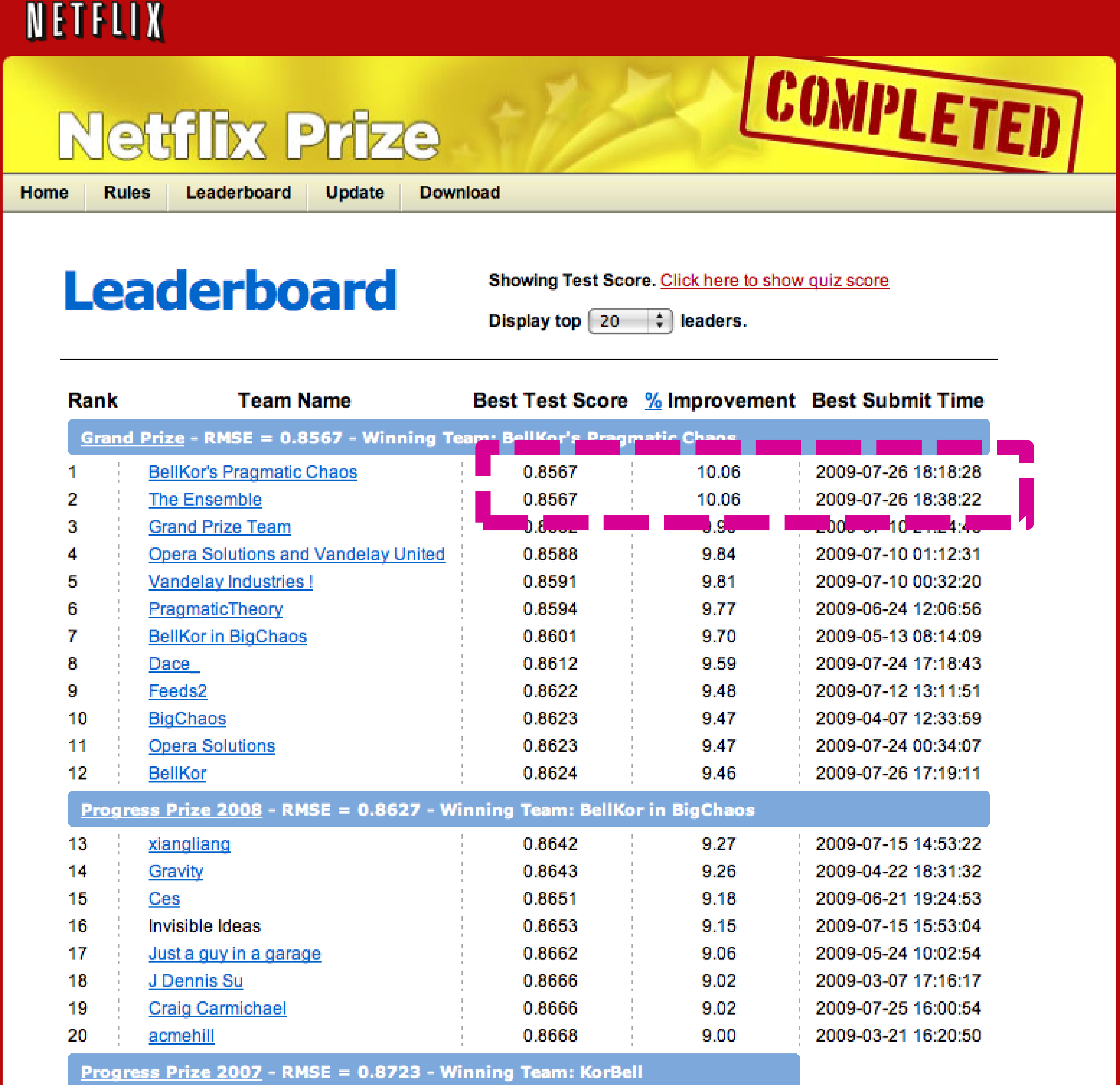
Standing on June 26th 2009

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"



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	0.8582	9.90	2009-07-10 21:24:40
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 - RMSE = 0.8723 - Winning Team: KorBell				

Million $ Awarded Sept 21^{st} 2009

DATE: 09.21.09

PAY TO THE ORDER OF: BellKor's Pragmatic Chaos

$ 1,000,000 00

AMOUNT: ONE MILLION 00/100

FOR: The Netflix Prize

Reed Hastings

What You Need to Know

- Weight learning approach for collaborative filtering
  - Learns optimal weights from data
- Latent factor model
  - Low dimensional embedding of users and items
  - Regularization: make the model generalize well
  - Bias extension

Questions?