Information Retrieval

CS 547/DS 547
Worcester Polytechnic Institute
Department of Computer Science
Instructor: Prof. Kyumin Lee

Midterm Exam

max	78
min	29
avg	62

Upcoming Schedule

- March 17: due date of project proposal
- March 21: due date of proposal presentation slides

Recommenders

Collaborative Recommendations

Collaborative Recommendations

- User-based recommendation

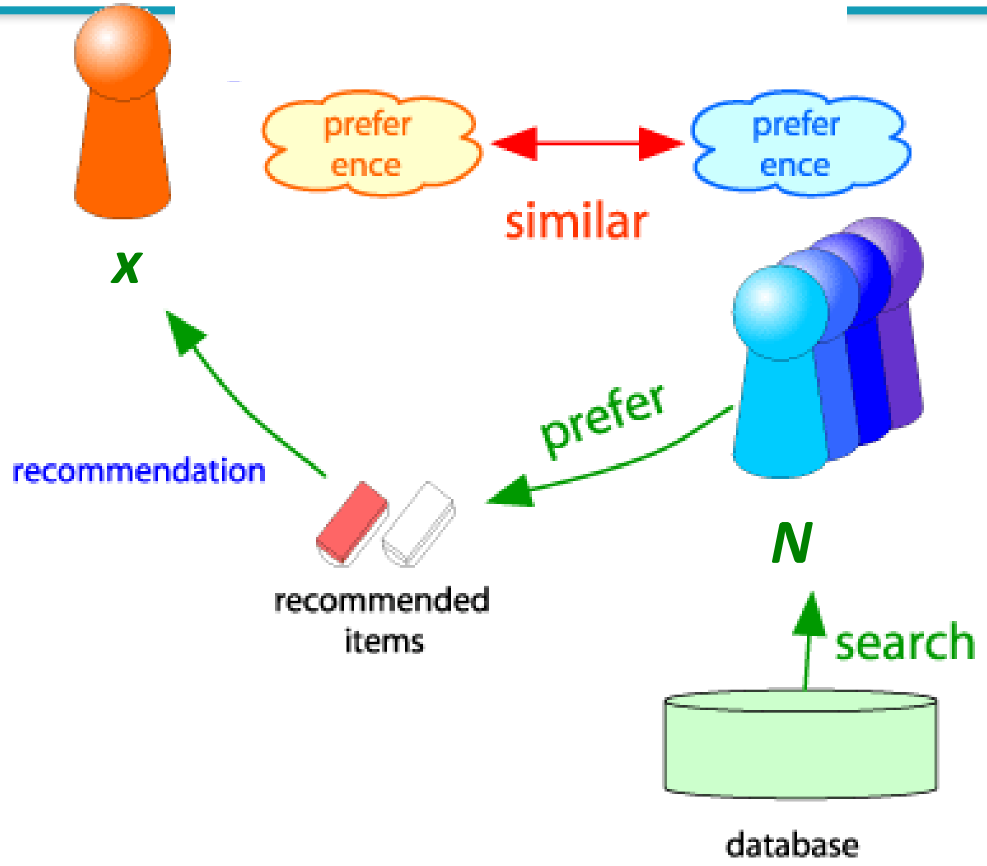
- Item-based recommendation

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***



User-User CF (|N|=2)

users

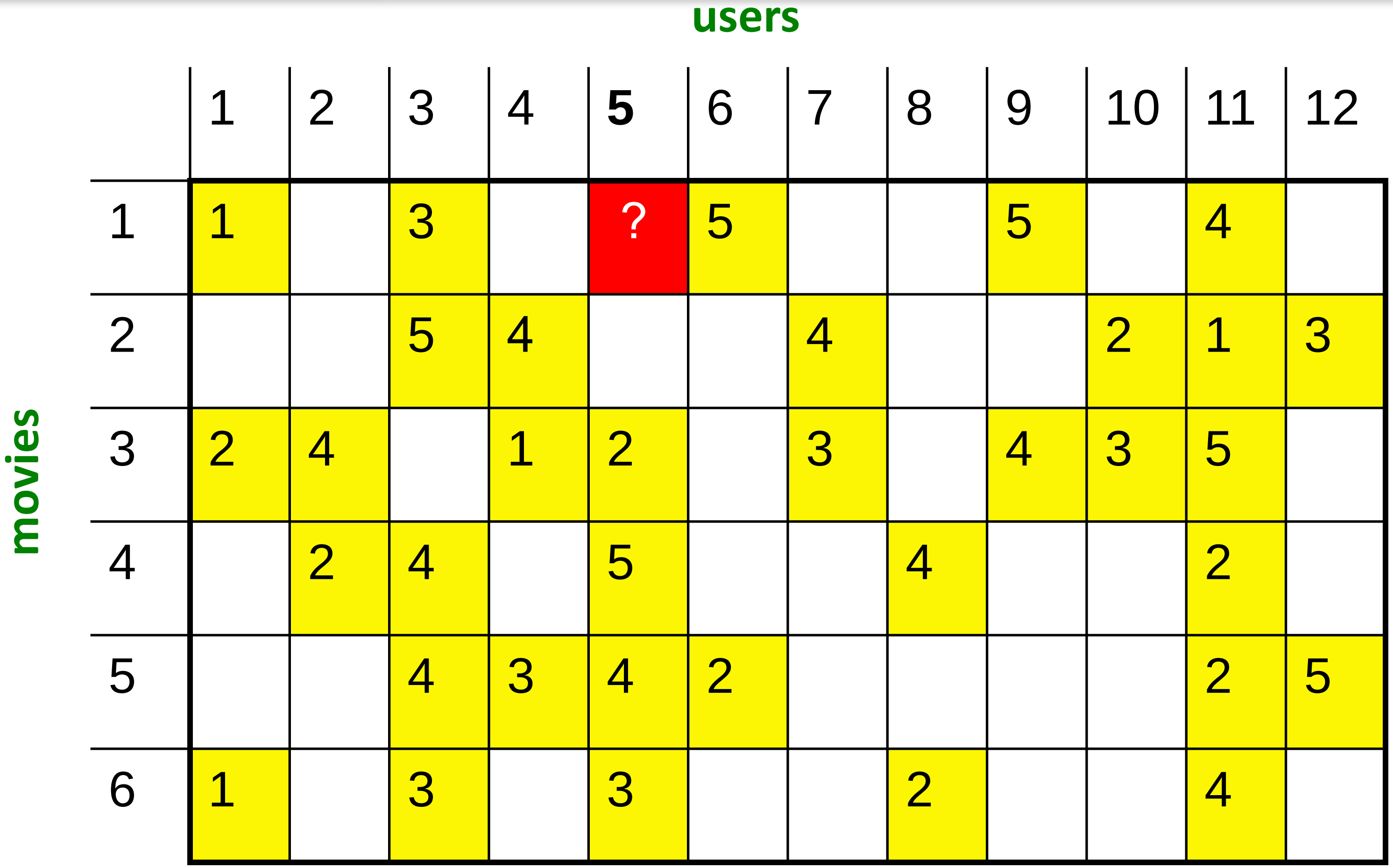
movies

	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

User-User CF (|N|=2)



users

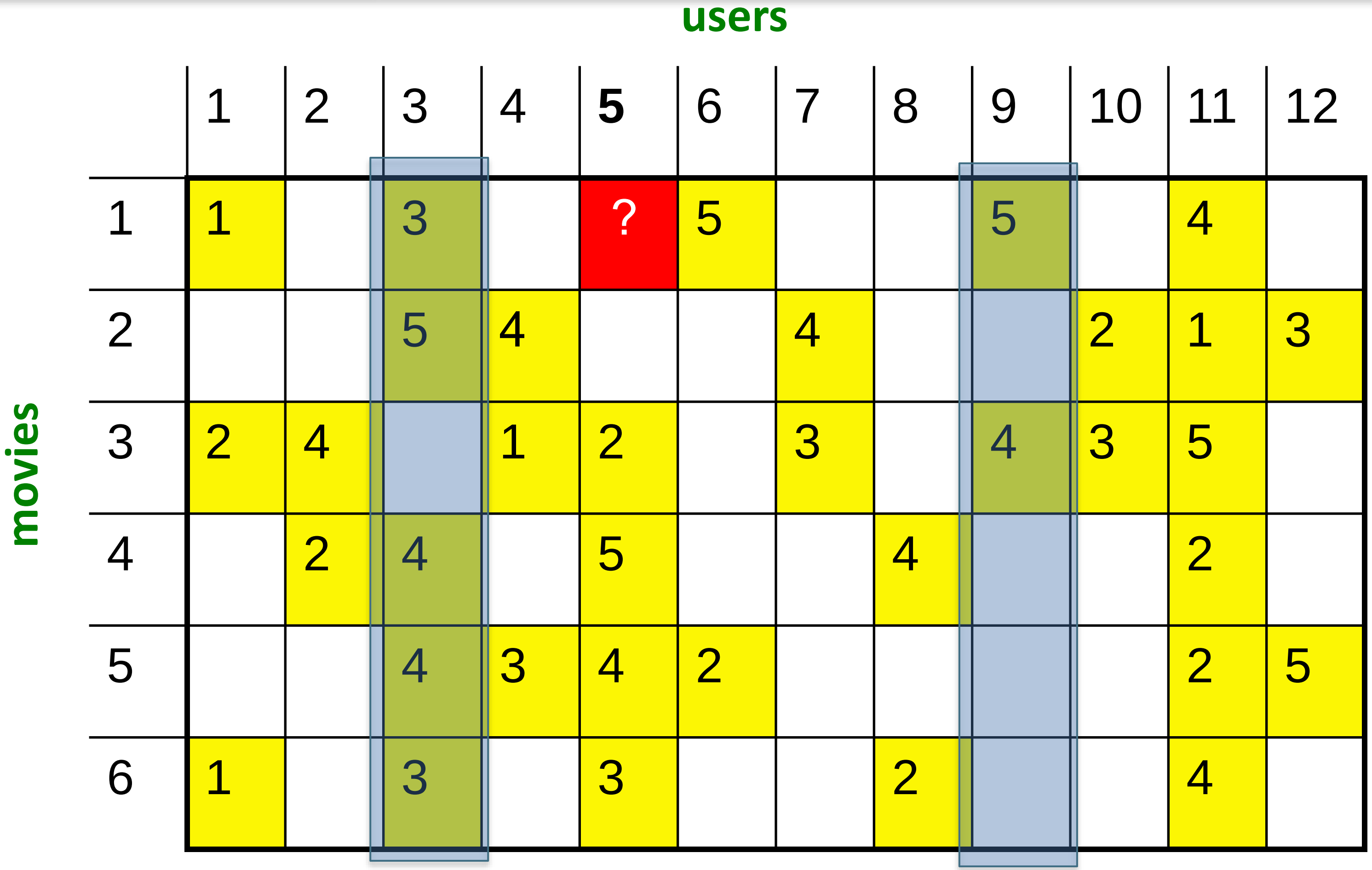
movies

	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	

Neighbor selection:
Identify users similar to user **5**, and rated item **1**

🟥 - estimate rating of movie **1** by user **5**

User-User CF (|N|=2)



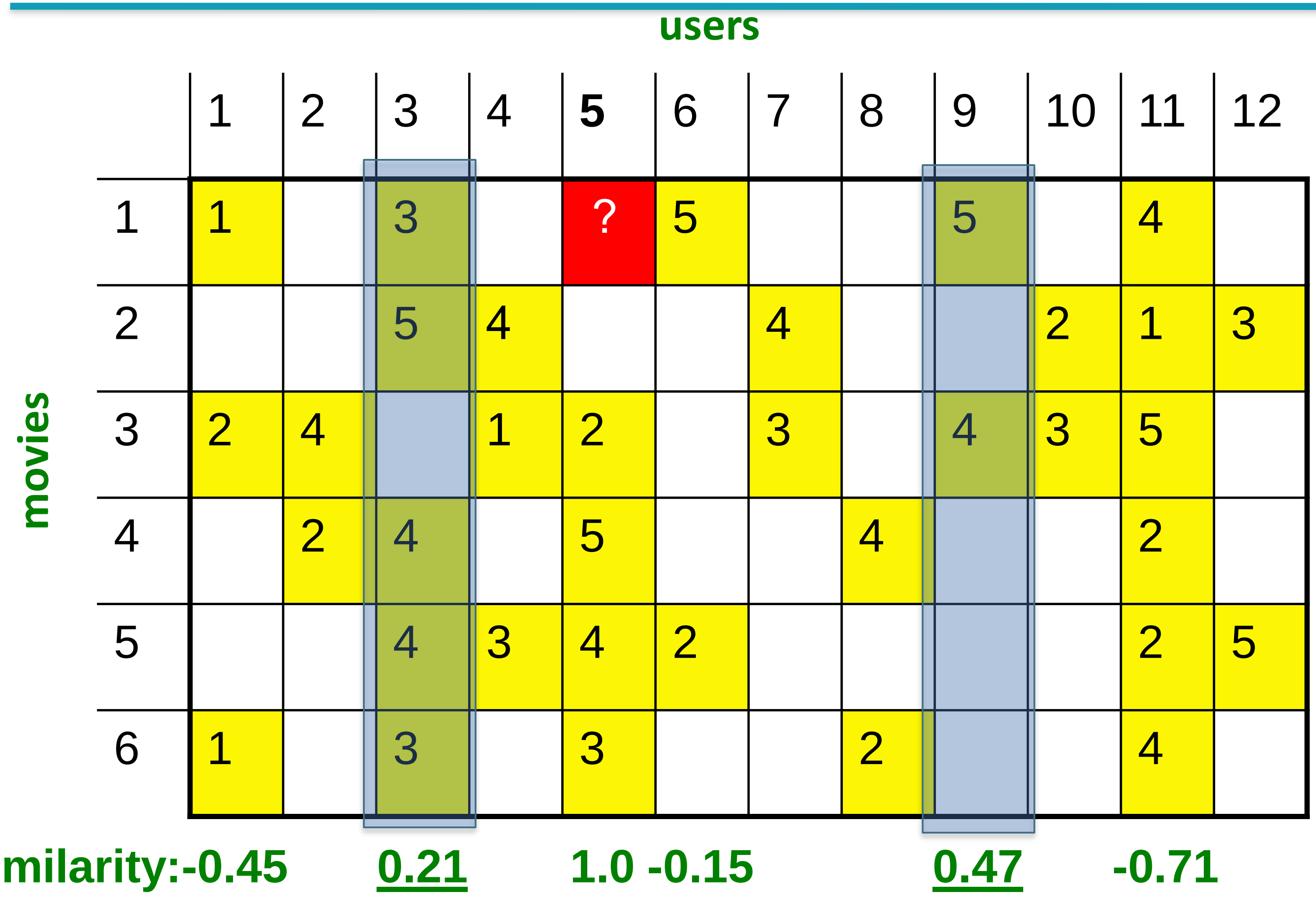
users

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	
Similarity:	**-0.45**		**0.21**		**1.0**	**-0.15**			**0.47**		**-0.71**	

Neighbor selection:
Identify users similar to user **5**, and rated item **1**

User-based CF (|K|=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	

Similarity:-0.45 0.21 1.0 -0.15 0.47 -0.71

Compute similarity weights:

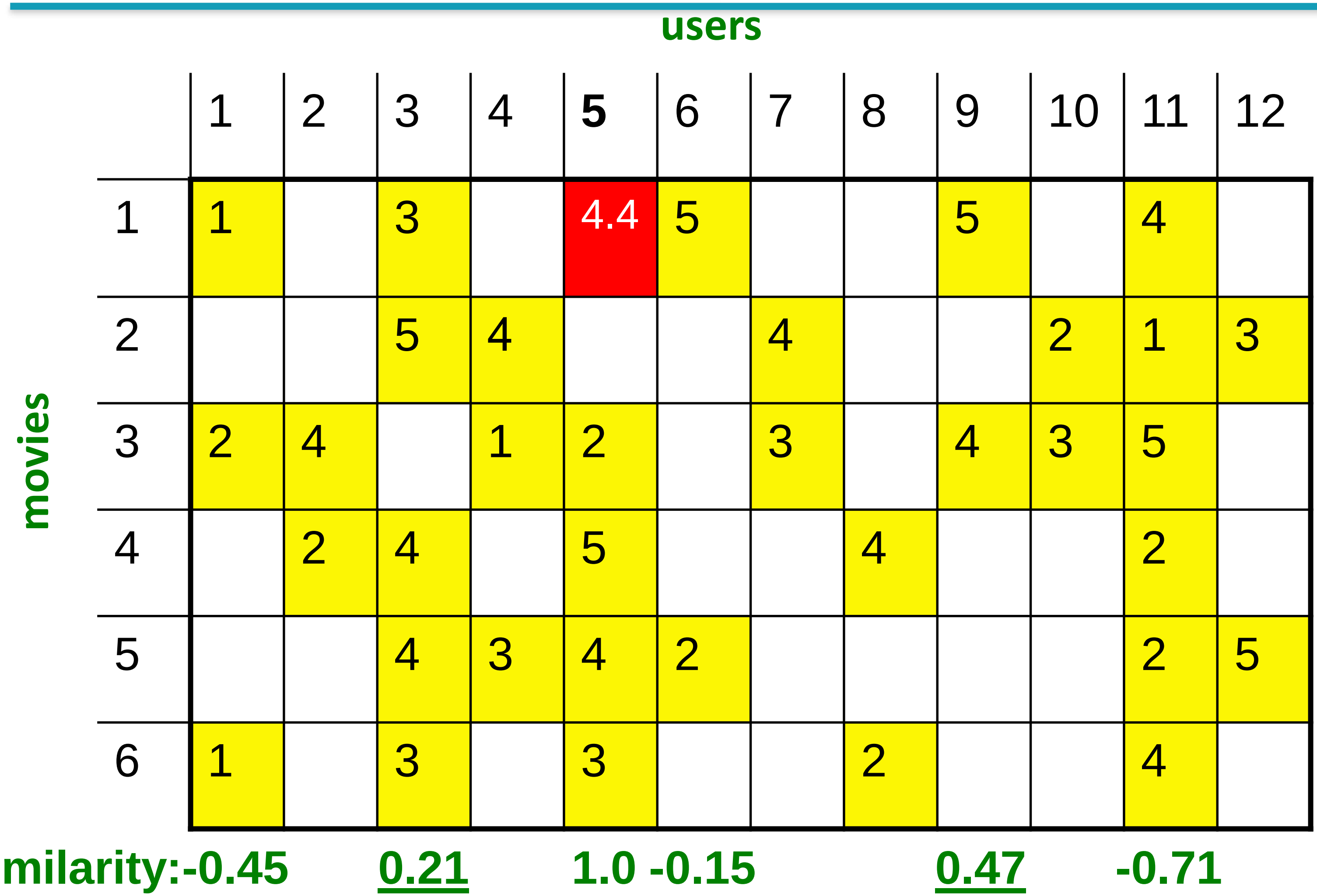
$s_{5,3}=0.21$, $s_{5,9}=0.47$

Predict by taking weighted average:

$r_{1,5} = (0.21*3 + 0.47*5) / (0.21+0.47) = 4.4$

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

User-based CF (|K|=2)



users

movies

	1	2	3	4	**5**	6	7	8	9	10	11	12
1	1		3		4.4	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	
Similarity:	-0.45		0.21		1.0	-0.15			0.47		-0.71	

Compute similarity weights:

$s_{5,3}=0.21$, $s_{5,9}=0.47$

Predict by taking weighted average:

$r_{1,5} = (0.21*3 + 0.47*5) / (0.21+0.47) = 4.4$

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

Item-based CF

- After computing the similarity between items we select a set of *k* most similar items to the target item and generate a predicted value of user x's rating

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

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

Item-Item CF (|N|=2)

users

movies

	1	2	3	4	5	6	7	8	9	10	11	12
1	1		3			5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

☐ - unknown rating 🟨 - rating between 1 to 5

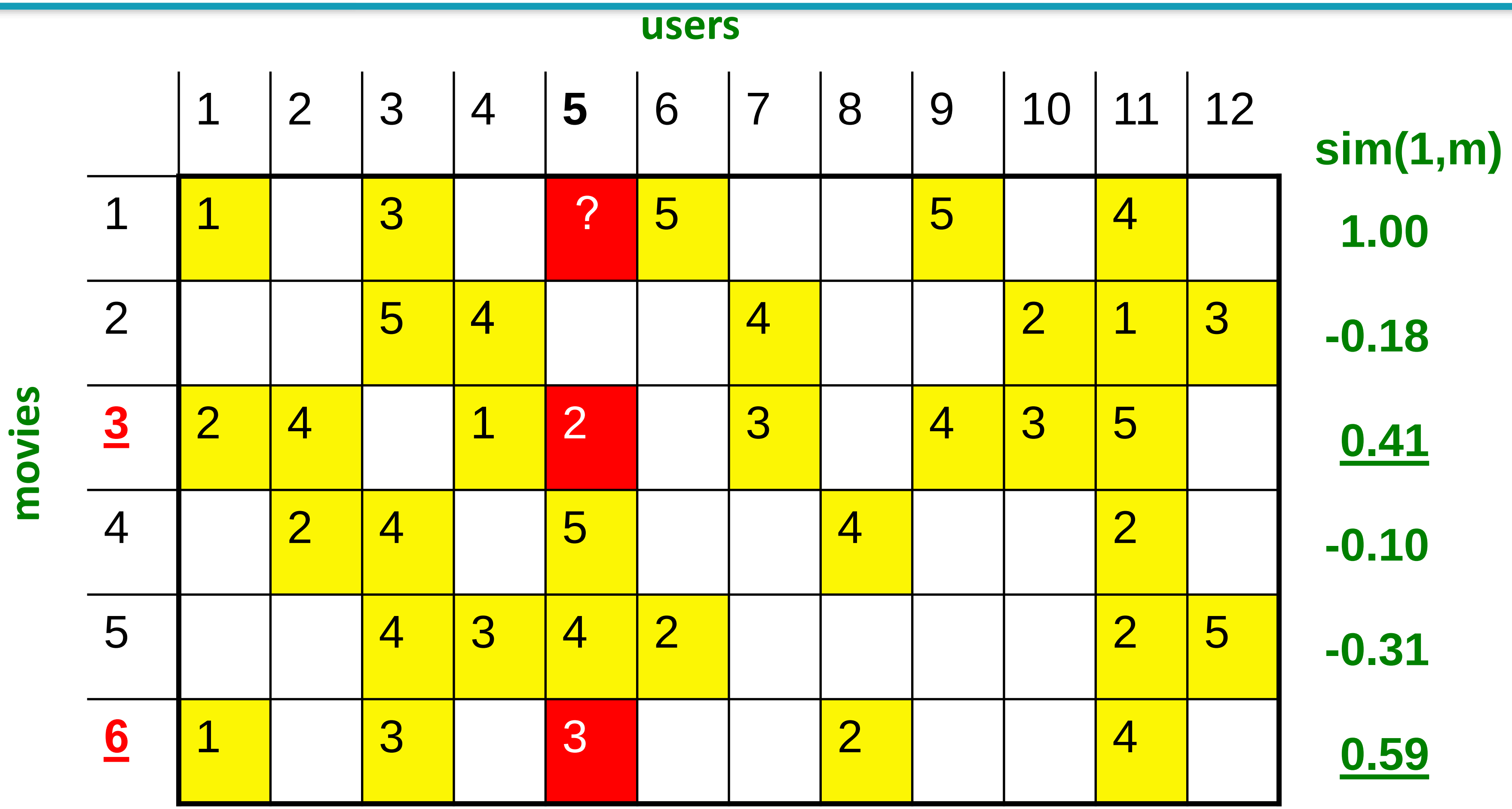
Item-Item CF (|N|=2)

users

movies	1	2	3	4	**5**	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

🟥 - estimate rating of movie **1** by user **5**

Item-Item CF (|N|=2)



users

movies

	1	2	3	4	**5**	6	7	8	9	10	11	12	sim(1,m)
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	-0.18
3	2	4		1	2		3		4	3	5		0.41
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
6	1		3		3			2			4		0.59

Neighbor selection:
Identify movies similar to movie **1**, **rated by user 5**

Here we use Pearson correlation as similarity:

1) Subtract mean rating m_i from each movie i

$m_1 = (1+3+5+5+4)/5 = \mathbf{3.6}$

row 1: *[-2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4, 0, 0.4, 0]*

2) Compute cosine similarities between rows

Item-Item CF (|N|=2)

users

movies

	1	2	3	4	5	6	7	8	9	10	11	12	sim(1,m)
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	-0.18
3	2	4		1	2		3		4	3	5		**0.41**
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
6	1		3		3			2			4		**0.59**

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	**sim(1,m)**
1	1		3		**2.6**	5			5		4		**1.00**
2			5	4			4			2	1	3	**-0.18**
3	2	4		1	2		3		4	3	5		**0.41**
4		2	4		5			4			2		**-0.10**
5			4	3	4	2					2	5	**-0.31**
6	1		3		3			2			4		**0.59**

Predict by taking weighted average:

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

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

Combining Global Baseline with CF

- **Global Baseline estimate:**
  - *Joe* will give *The Sixth Sense* 4 stars

- **Local neighborhood (CF/NN):**
  - *Joe* didn't like related movie *Signs*
  - Rated it 1 star below his average rating

- **Final estimate**
  - *Joe* will rate *The Sixth Sense* 4 – 1 = 3 stars

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)* – μ
- b_i = rating deviation of movie i

Latent Factor Models

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

Performance of Various Methods

Global average: 1.1296

User average: 1.0651

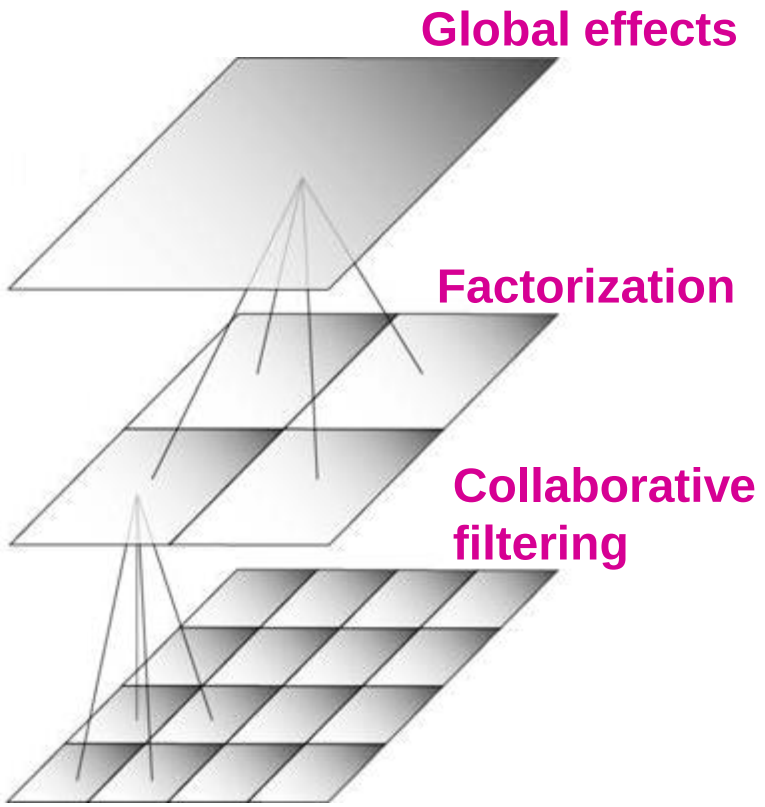
Movie average: 1.0533

Netflix: 0.9514

Grand Prize: 0.8563

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*
  - Rated it 1 star below his average rating

- **Final estimate**
  - *Joe* will rate *The Sixth Sense* 4 – 1 = 3 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 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.**:

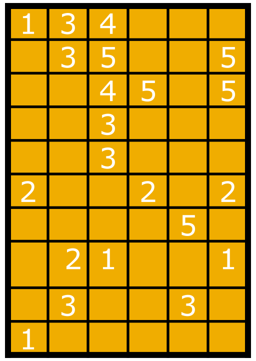
$$\widehat{r_{xi}} = b_{xi} + \sum_{j \in N(i;x)} w_{ij}\left(r_{xj} - b_{xj}\right)$$

- **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: $\boldsymbol{\sum_{j \in N(i;x)} w_{ij} \neq 1}$
  - $\boldsymbol{w_{ij}}$ 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: $\sqrt{\frac{1}{|R|} \sum_{(i,x) \in R} (\hat{r}_{xi} - r_{xi})^2}$ or equivalently Sum of Squared Error (**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***

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 ⇒ better recommendations
  - Want to make good recommendations on items that user has not yet seen. Can't really do this!
  - **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**

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

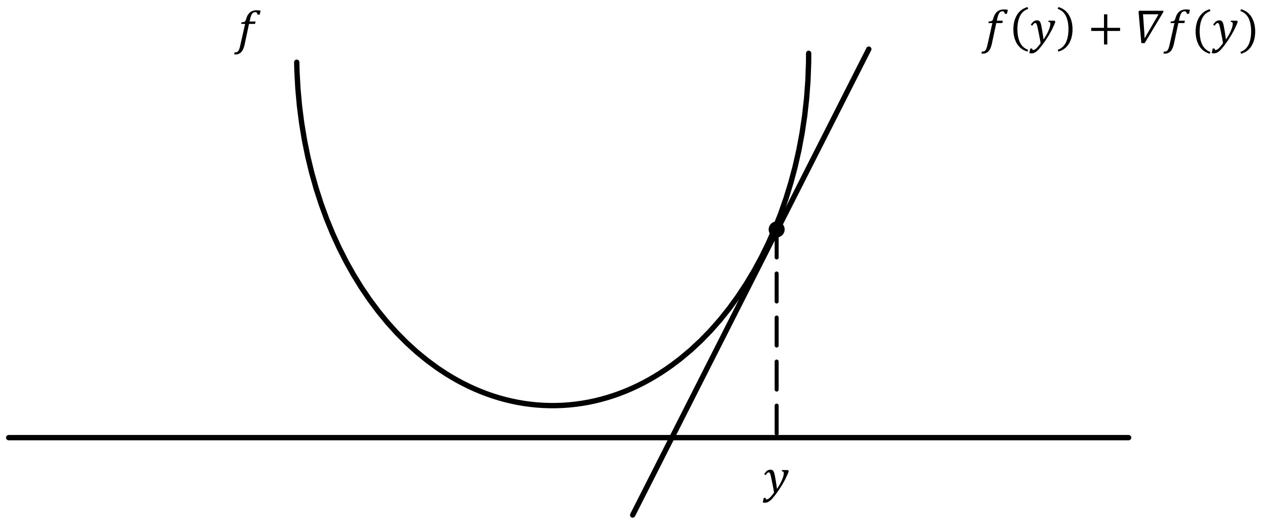
- Find w_{ij} that minimize **SSE** on **training data**!

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

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

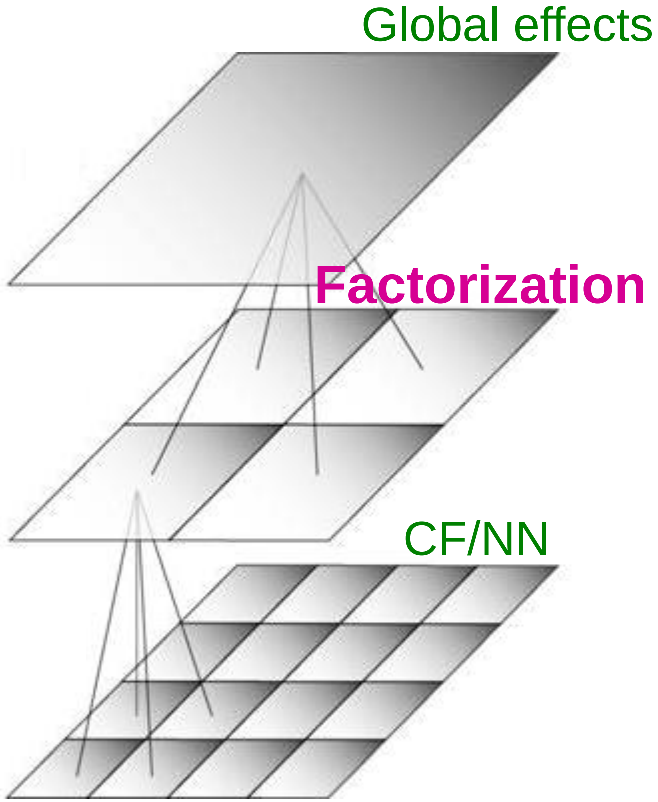
Detour: Minimizing a function

- **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 converged**

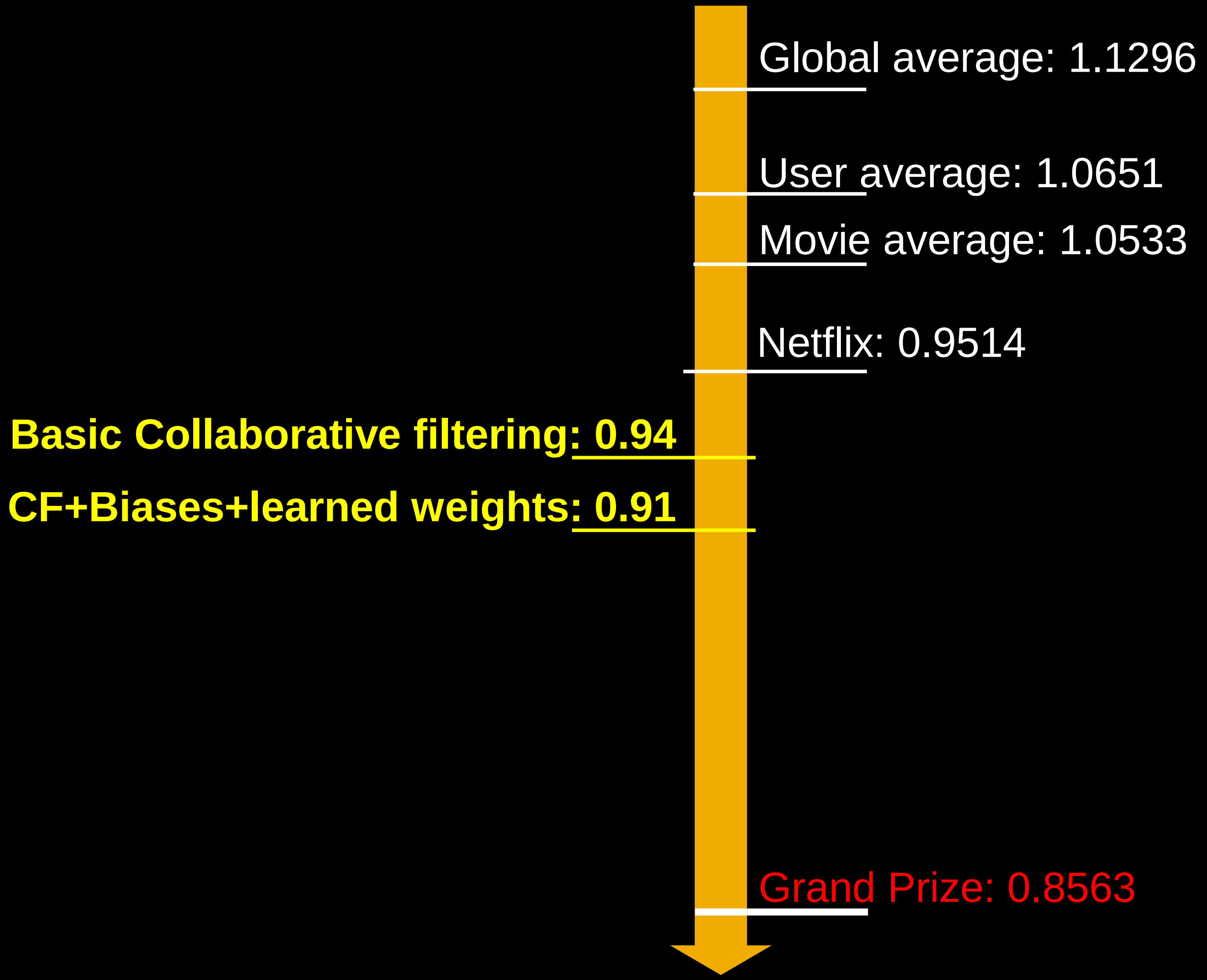


Interpolation Weights

- **So far:** $\widehat{r_{xi}} = b_{xi} + \sum_{j \in N(i;x)} w_{ij}(r_{xj} - b_{xj})$
  - Weights $\boldsymbol{w_{ij}}$ derived 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**
  - Extract "regional" correlations



Performance of Various Methods



Global average: 1.1296

User average: 1.0651

Movie average: 1.0533

Netflix: 0.9514

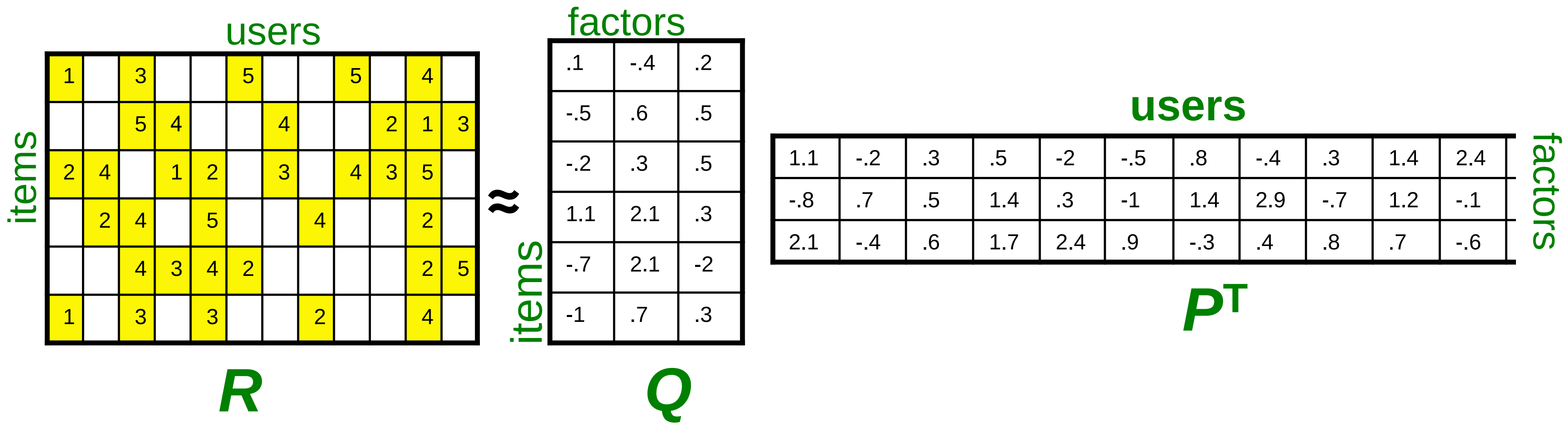
Basic Collaborative filtering: 0.94

CF+Biases+learned weights: 0.91

Grand Prize: 0.8563

Latent Factor Models

- Latent Factor Model 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

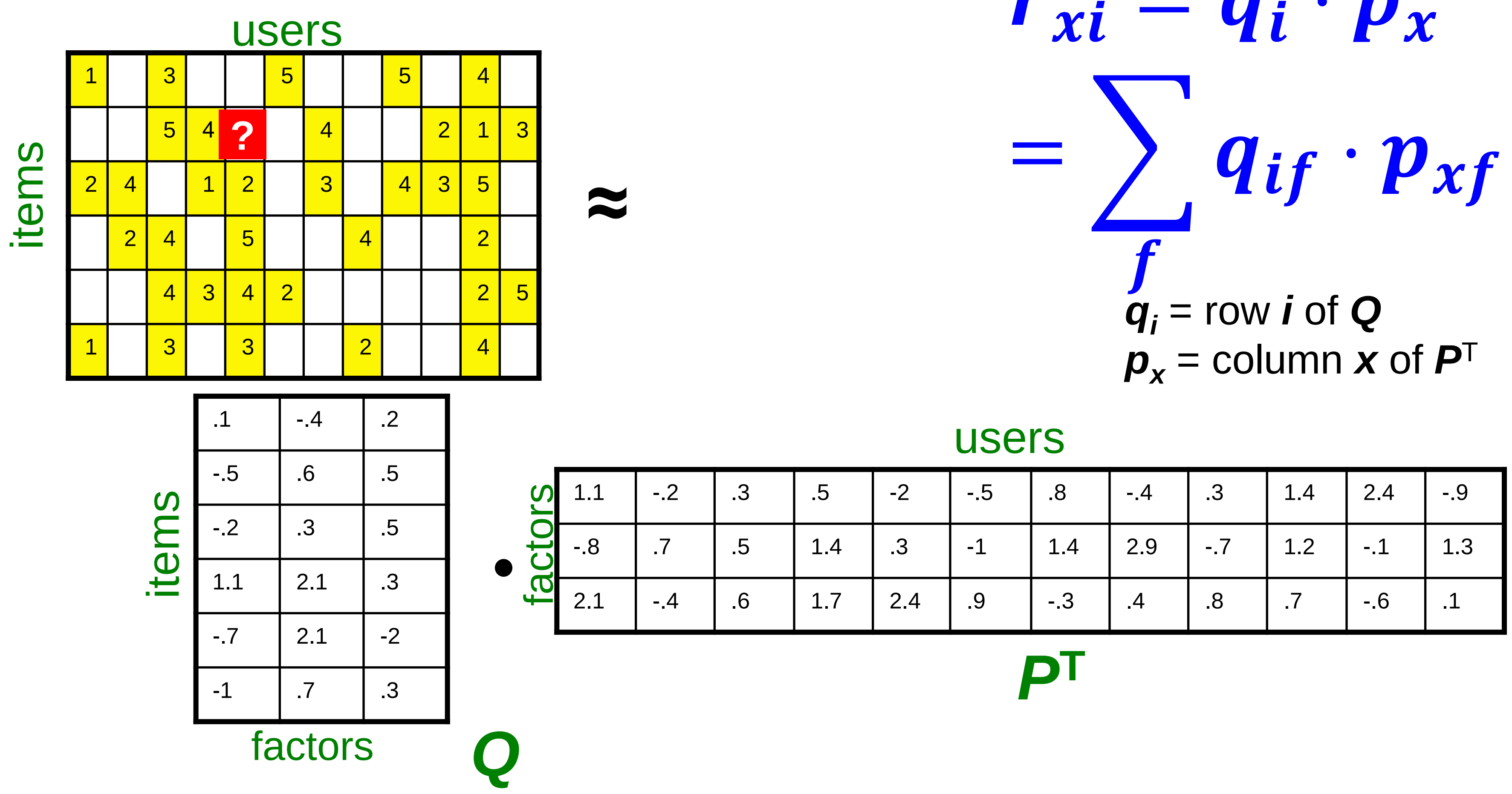
P^T

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 ***R*** as a product of "thin" $\boldsymbol{Q} \cdot \boldsymbol{P}^T$
  - ***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*?



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

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

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

q_i = row ***i*** of ***Q***

p_x = column ***x*** of P^T

Ratings 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 $\boldsymbol{P}^{\mathrm{T}}$

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

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

Ratings as Products of Factors

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

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

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

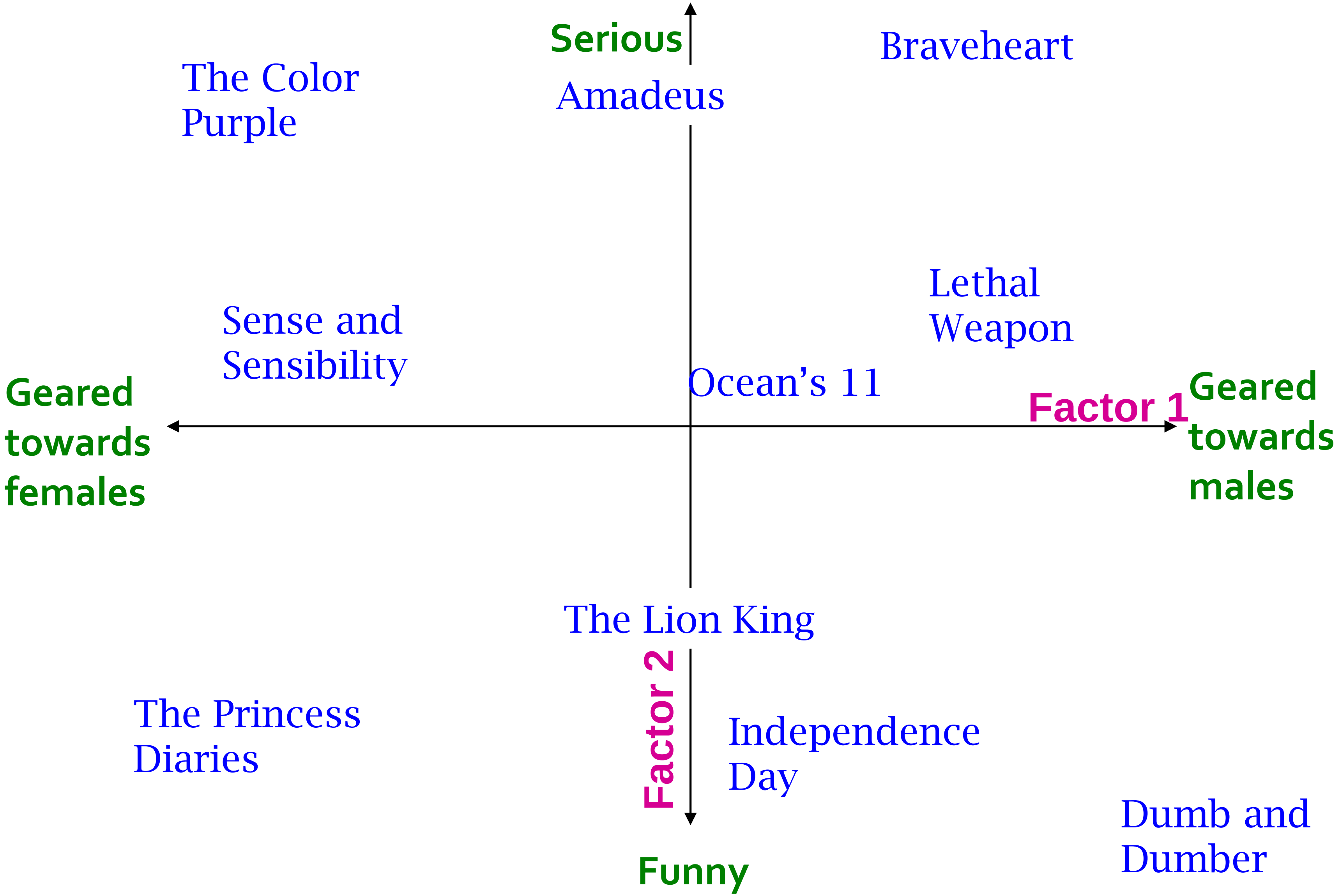
$$\hat{r}_{xi} = q_i \cdot p_x$$

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

$\boldsymbol{q_i}$ = row $\boldsymbol{i}$ of $\boldsymbol{Q}$

$\boldsymbol{p_x}$ = column $\boldsymbol{x}$ of $\boldsymbol{P}^{\mathrm{T}}$

Latent Factor Models



Serious

Braveheart

The Color Purple

Amadeus

Lethal Weapon

Sense and Sensibility

Ocean's 11

Geared towards females

Factor 1

Geared towards males

The Lion King

Factor 2

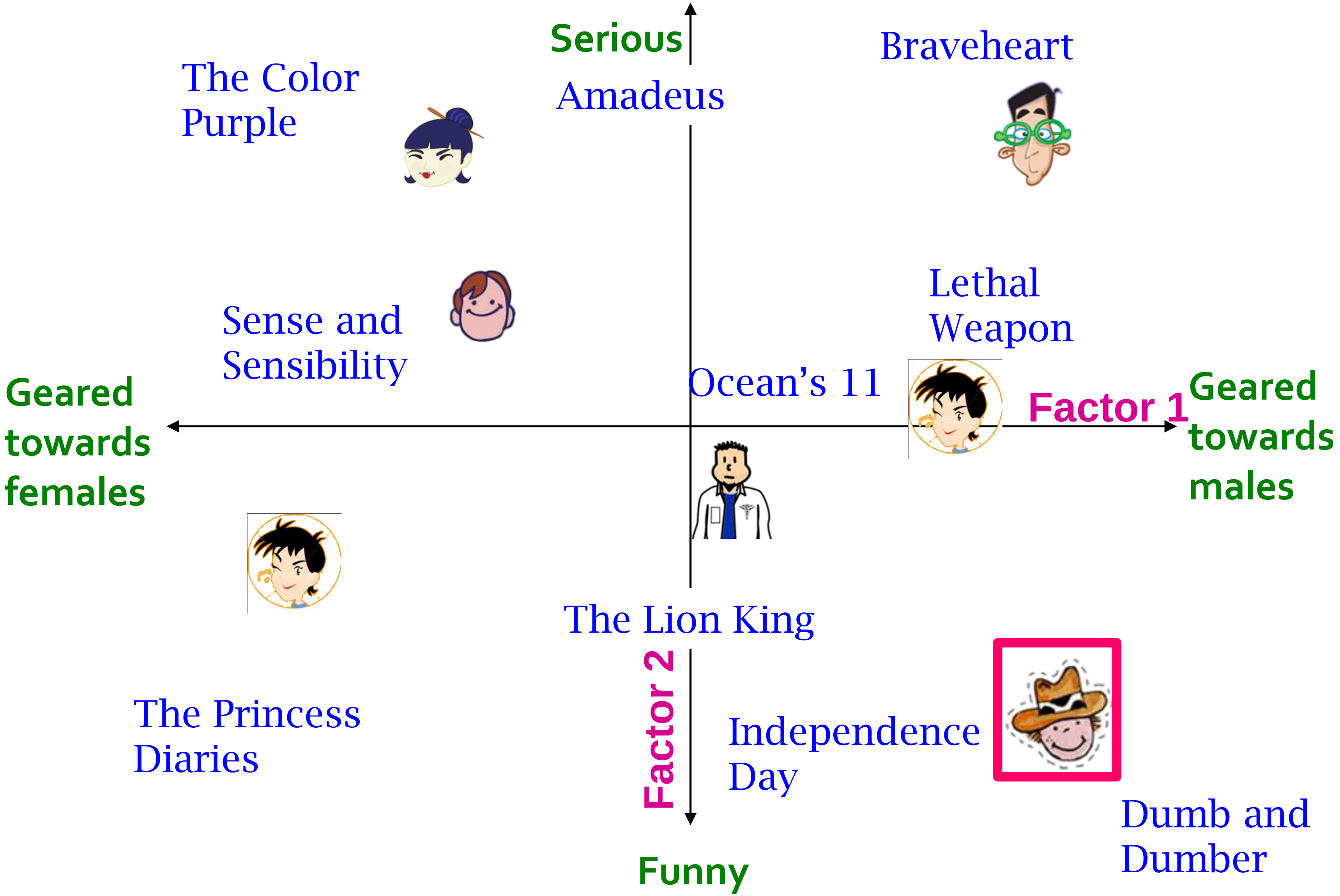
The Princess Diaries

Independence Day

Dumb and Dumber

Funny

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

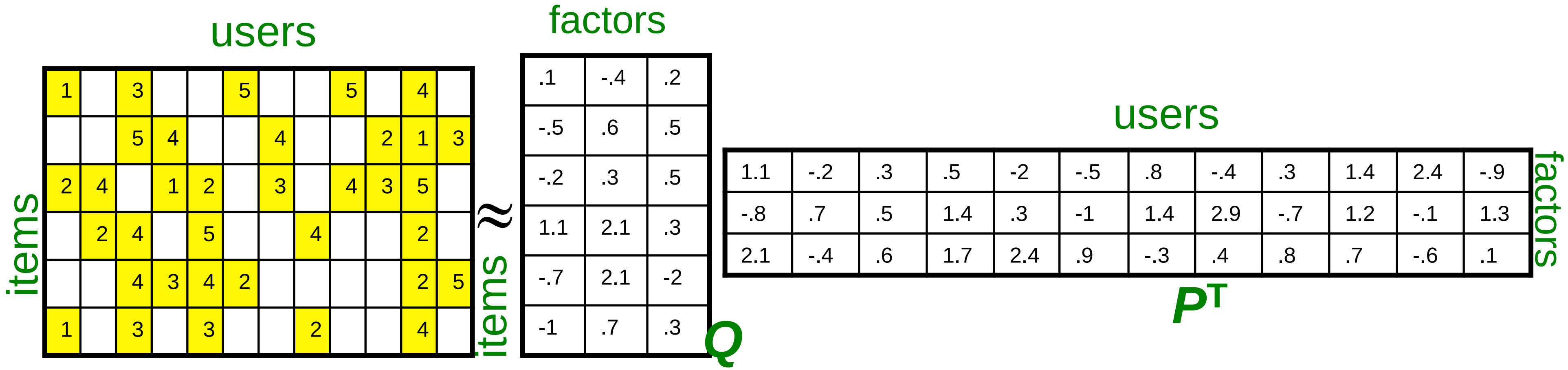
Funny

Finding the Latent Factors

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	

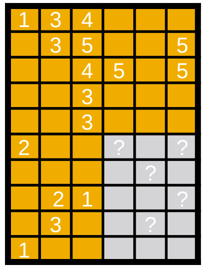
≈

.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

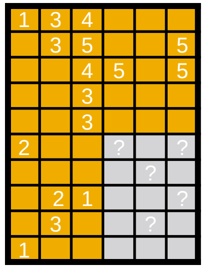
- Want to minimize sum of the squared errors (SSE) for unseen test data
- **Idea:** Minimize SSE on training data
  - Want large ***k*** (# of factors) to capture all the signals
  - But, **SSE** on test data begins to rise for ***k*** > 2



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

- 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

Dealing with Missing Entries



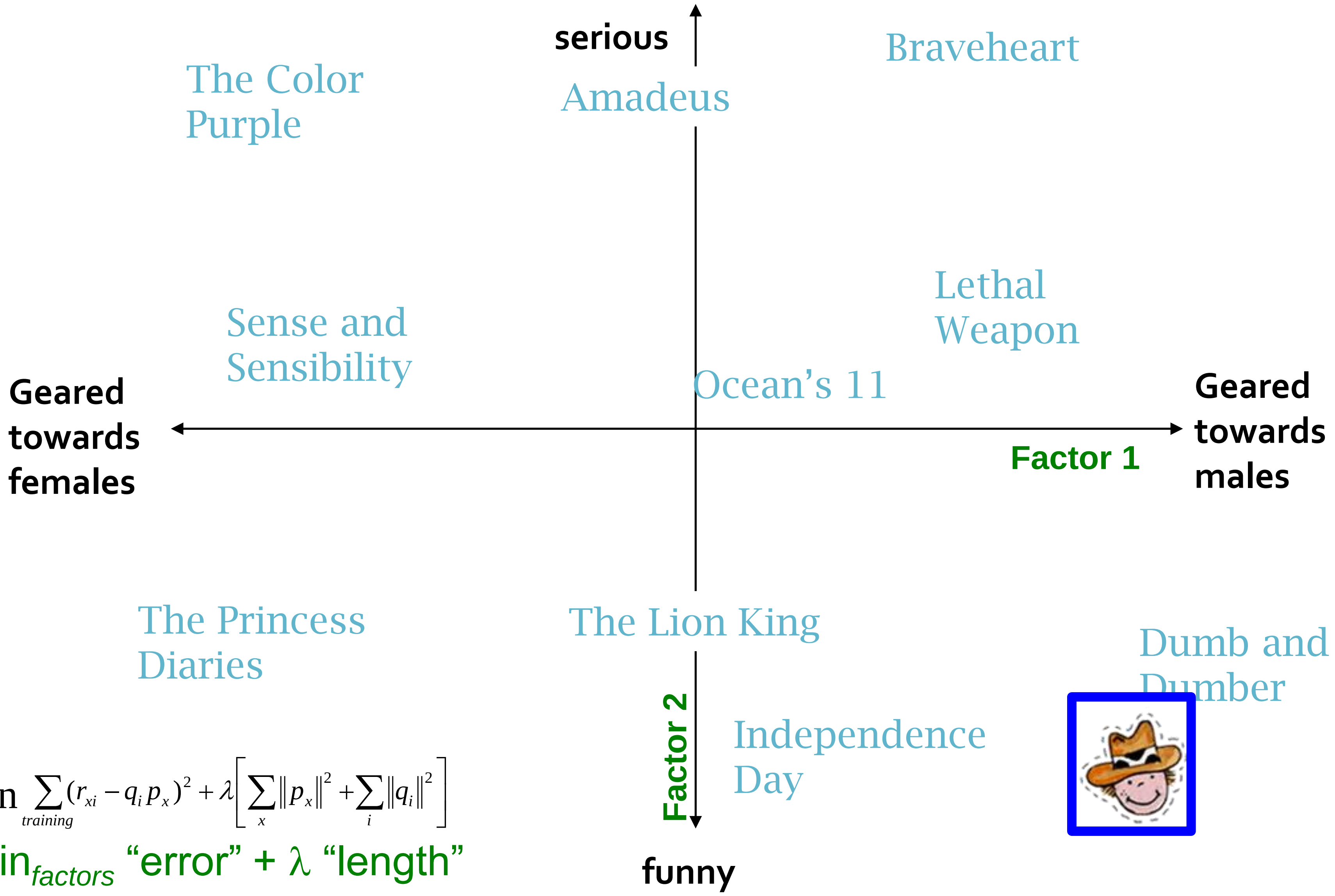
1	3	4			
	3	5			5
		4	5		5
		3			
		3			
2			?		?
				?	
	2	1			?
	3			?	
1					

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

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

λ_1, λ_2 … user set regularization parameters

The Effect of Regularization



serious

The Color Purple

Amadeus

Braveheart

Sense and Sensibility

Lethal Weapon

Ocean’s 11

Geared towards females

Geared towards males

Factor 1

The Princess Diaries

The Lion King

Dumb and Dumber

Factor 2

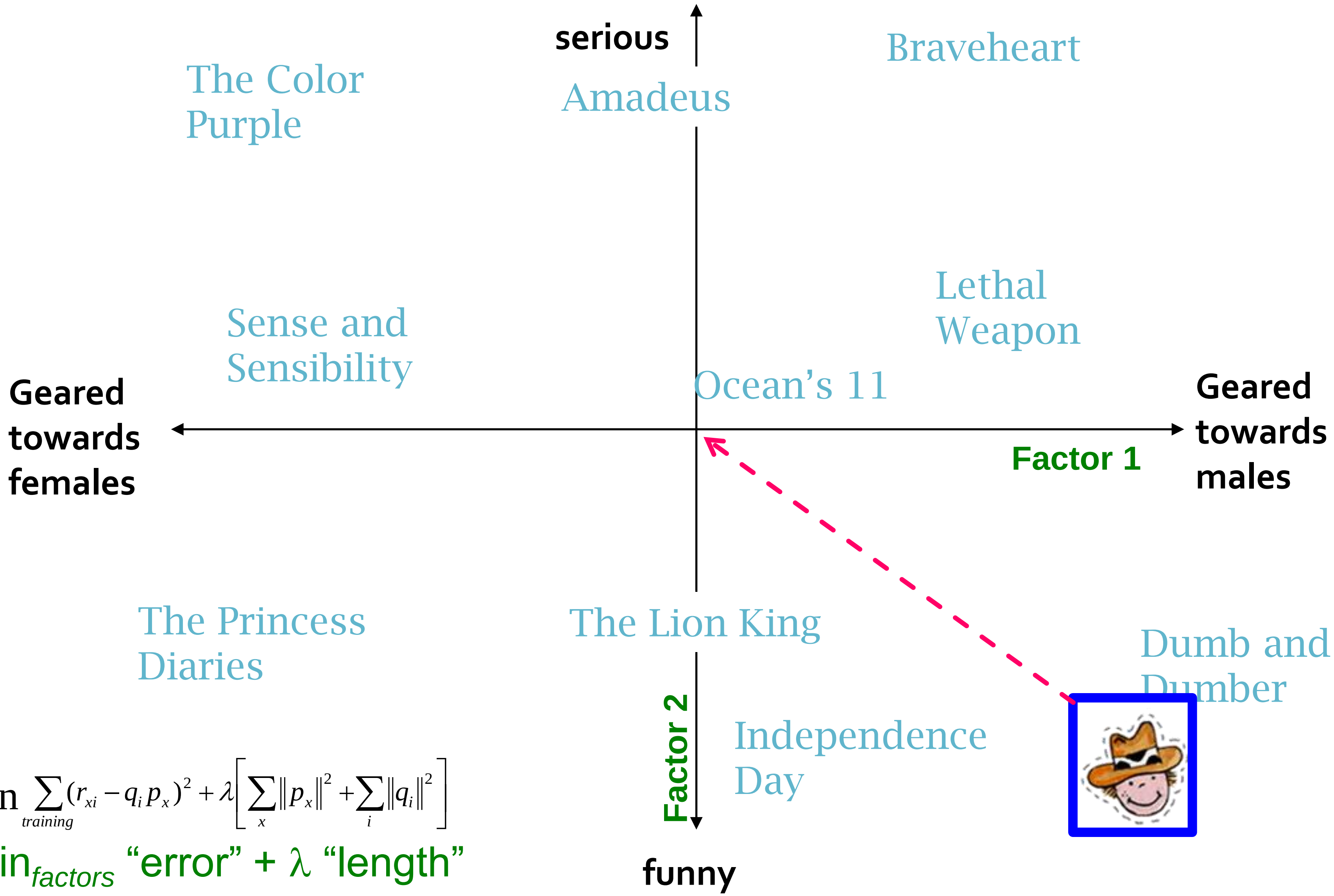
Independence Day

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”

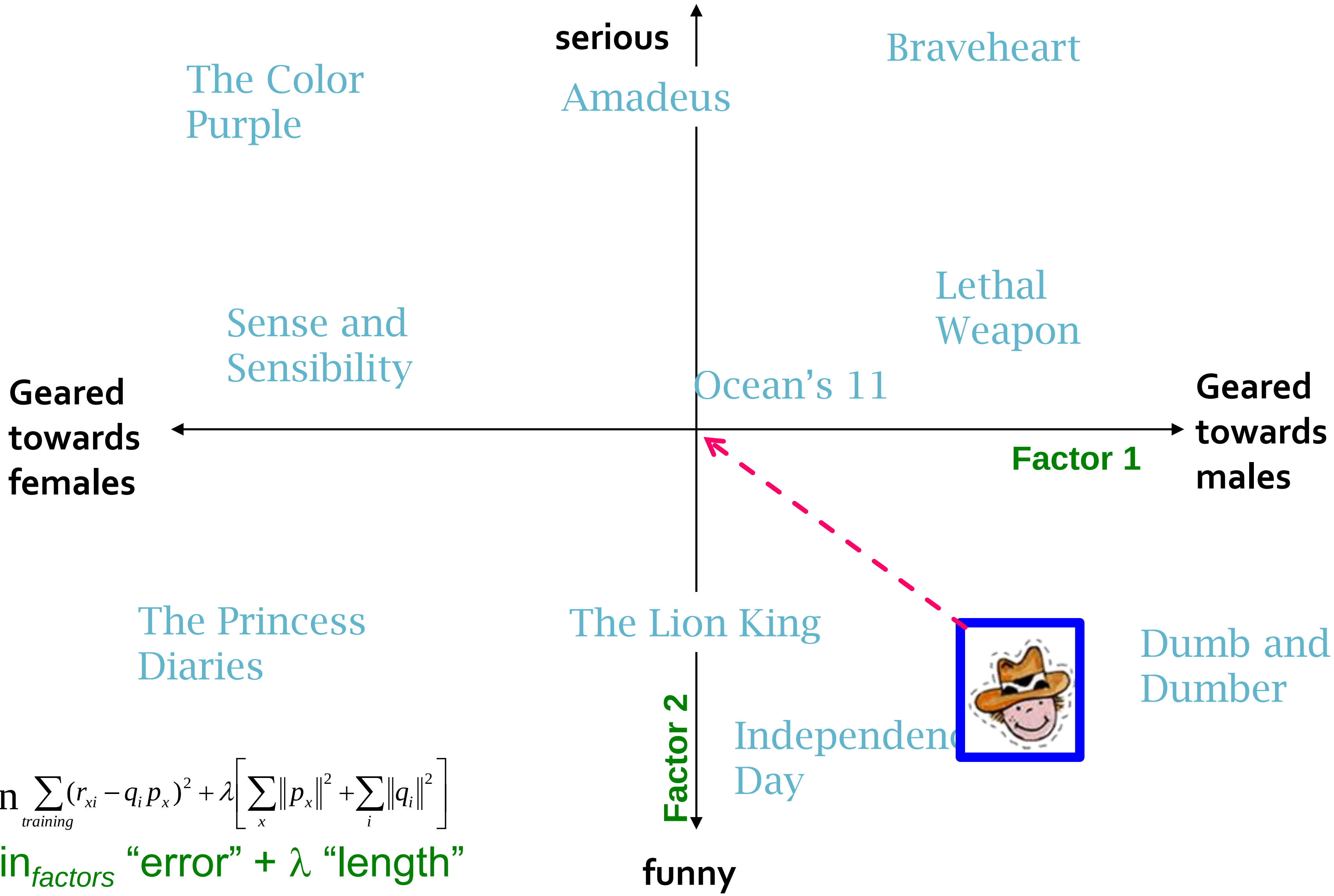
The Effect of Regularization



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

$\min_{factors}$ “error” + λ “length”

The Effect of Regularization



serious

Braveheart

The Color Purple

Amadeus

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

Independence Day

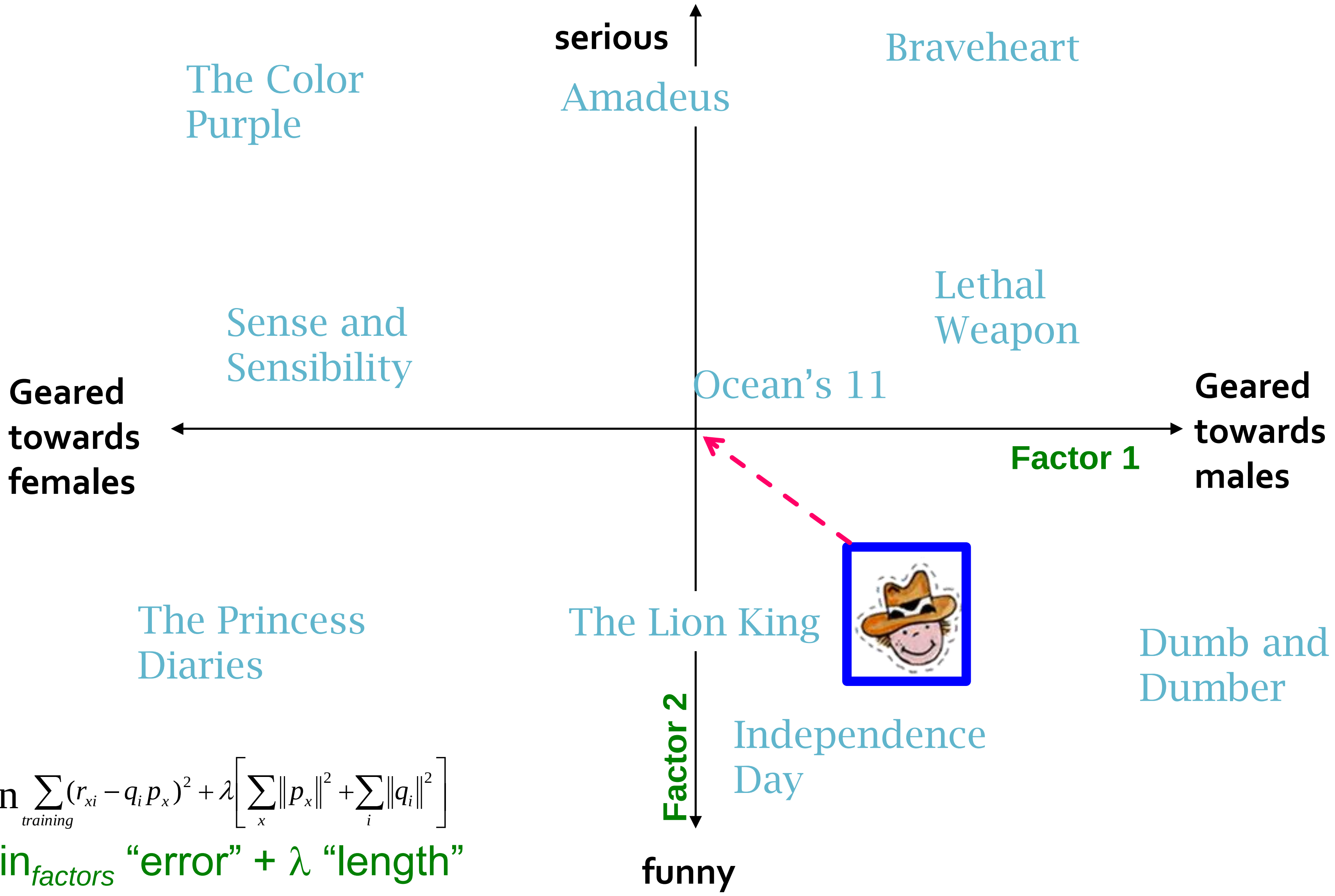
Factor 2

funny

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

$\text{min}_{factors}$ “error” + λ “length”

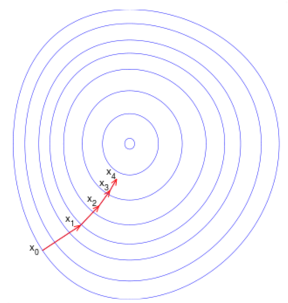
The Effect of Regularization



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

$\min_{factors}$ “error” + λ “length”

Stochastic Gradient Descent



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

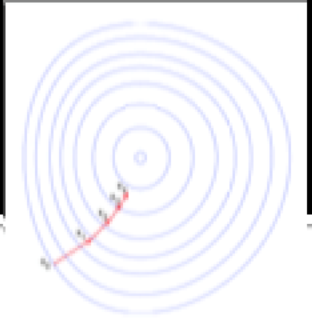
$$\min_{P,Q} \sum_{training} (r_{xi} - q_i p_x)^2 + \left[\lambda_1 \sum_x \|p_x\|^2 + \lambda_2 \sum_i \|q_i\|^2 \right]$$

- **Gradient descent:**
  - Initialize ***P*** and ***Q*** (using SVD, pretend missing ratings are 0)
  - Do gradient descent:
    - $\boldsymbol{P \leftarrow P - \eta \cdot \nabla P}$
    - $\boldsymbol{Q \leftarrow Q - \eta \cdot \nabla Q}$
    - 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***

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

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

Stochastic Gradient Descent



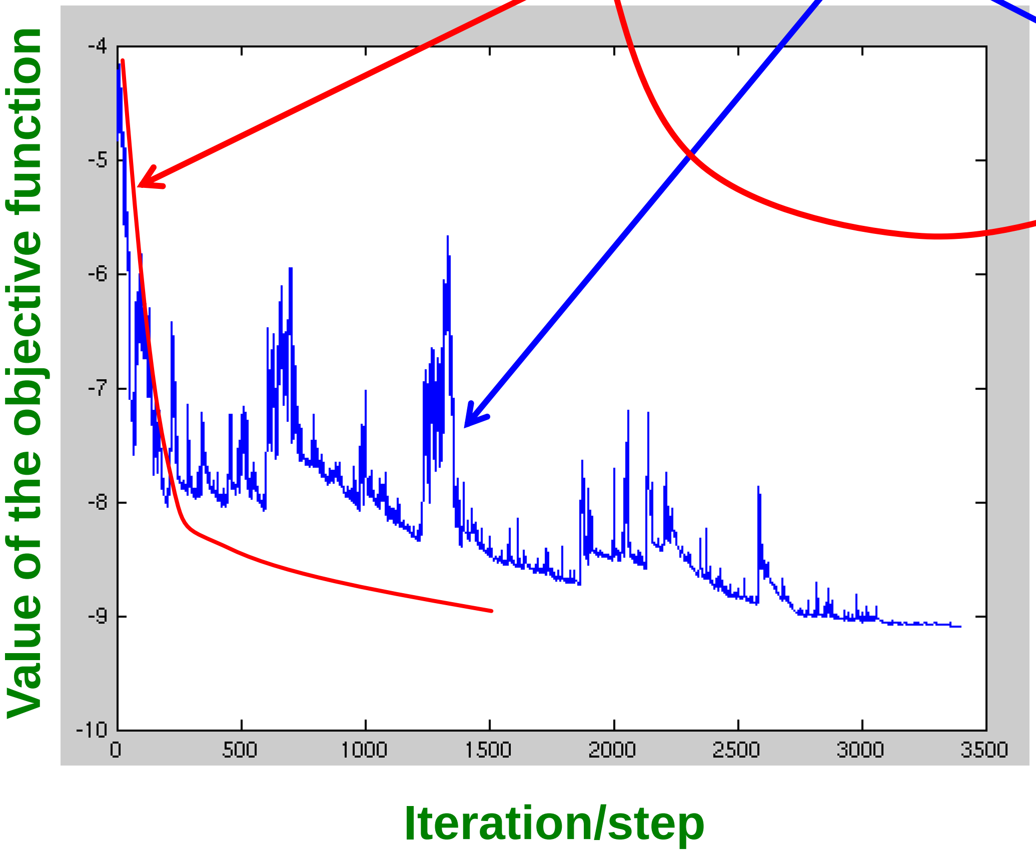
- **Gradient Descent (GD) vs. Stochastic GD**
  - **Observation:** $\nabla Q = [\nabla q_{if}]$ where

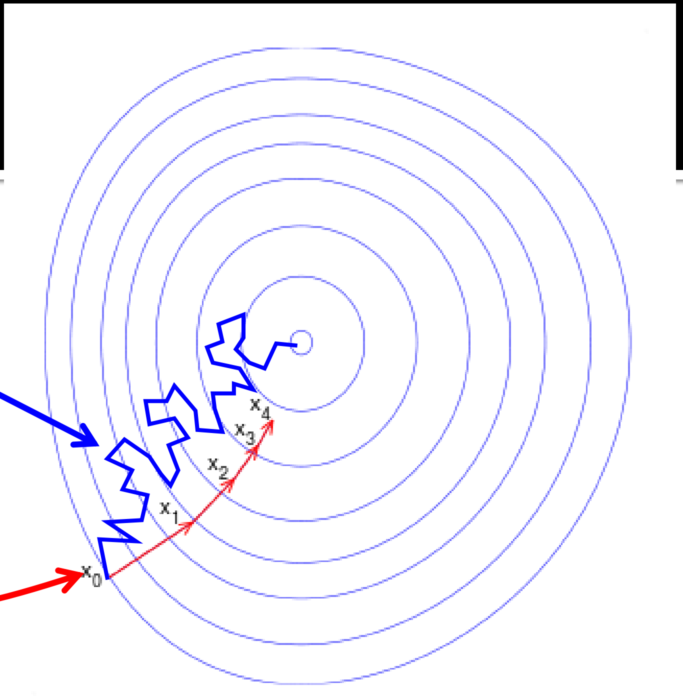
    $$\nabla q_{if} = \sum_{x,i} -2(r_{xi} - q_{if} p_{xf}) p_{xf} + 2\lambda q_{if} = \sum_{x,i} \nabla \boldsymbol{Q}\,(\boldsymbol{r_{xi}})$$

    - Here $\boldsymbol{q_{if}}$ is entry $\boldsymbol{f}$ of row $\boldsymbol{q_i}$ of matrix $\boldsymbol{Q}$
  - $\boldsymbol{Q} = \boldsymbol{Q} - \eta \nabla \boldsymbol{Q} = \boldsymbol{Q} - \eta \left[\sum_{\boldsymbol{x,i}} \nabla \boldsymbol{Q}\,(\boldsymbol{r_{xi}})\right]$
  - **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**





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!

Pytorch Tutorial + Matrix Factorization