Recommender Systems: Latent Factor Models

Mining of Massive Datasets

Jure Leskovec, Anand Rajaraman, Jeff Ullman

Stanford University

http://www.mmds.org

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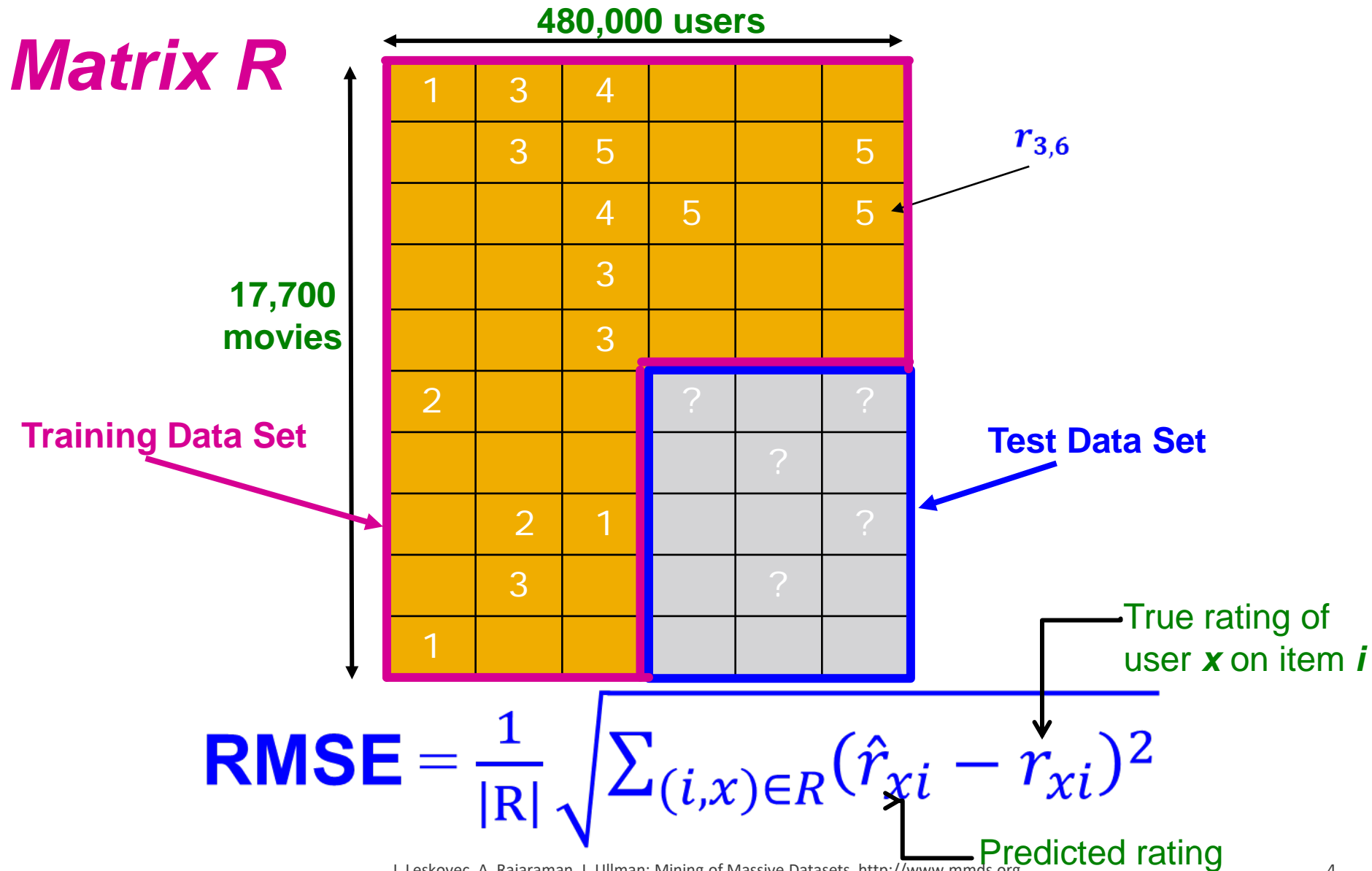
The Netflix Utility Matrix *R*

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					

Matrix R

480,000 users

17,700 movies

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

$r_{3,6}$

Training Data Set

Test Data Set

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

Predicted rating

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

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:
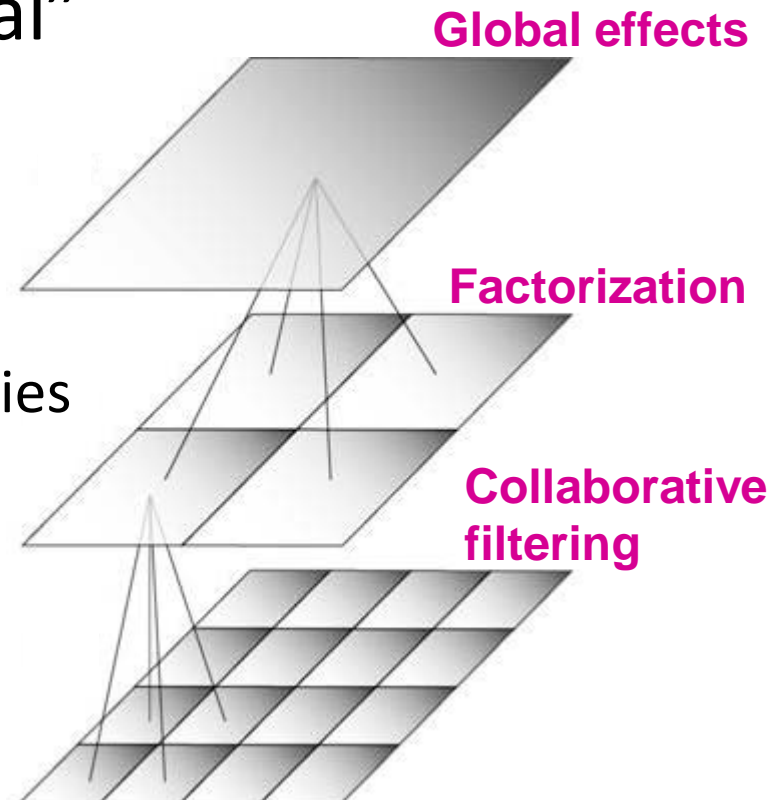
  - **Global:**
    - Overall deviations of users/movies
  - **Factorization:**
    - Addressing "regional" effects
  - **Collaborative filtering:**
    - Extract local patterns

Global effects

Factorization

Collaborative filtering

Modeling Local & Global Effects

- ## **Global:**

  - Mean movie rating: **3.7 stars**

  - *The Sixth Sense* is **0.5** stars above avg.

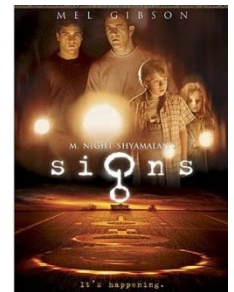 - Joe rates **0.2** stars below avg.

    ⇒ **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

- **Earliest and 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}

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

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.**:

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

- **A few notes:**
  - $N(i; x)$ … set of movies rated by user x that are similar to movie i

  - w_{ij} is the interpolation weight (some real number)
    - We allow: $\sum_{j \in N(i,x)} w_{ij} \neq 1$

  - w_{ij} models interaction between pairs of movies (it does not depend on user x)

Idea: Interpolation Weights w_{ij}

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

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

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

Predicted rating

True rating

- Think of *w* as a vector of numbers

Detour: Minimizing a function

- **A simple way to minimize a function $f(x)$:**
  - Compute the take a derivative ∇f
  - **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**

f $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad f(y) + \nabla f(y)$

y
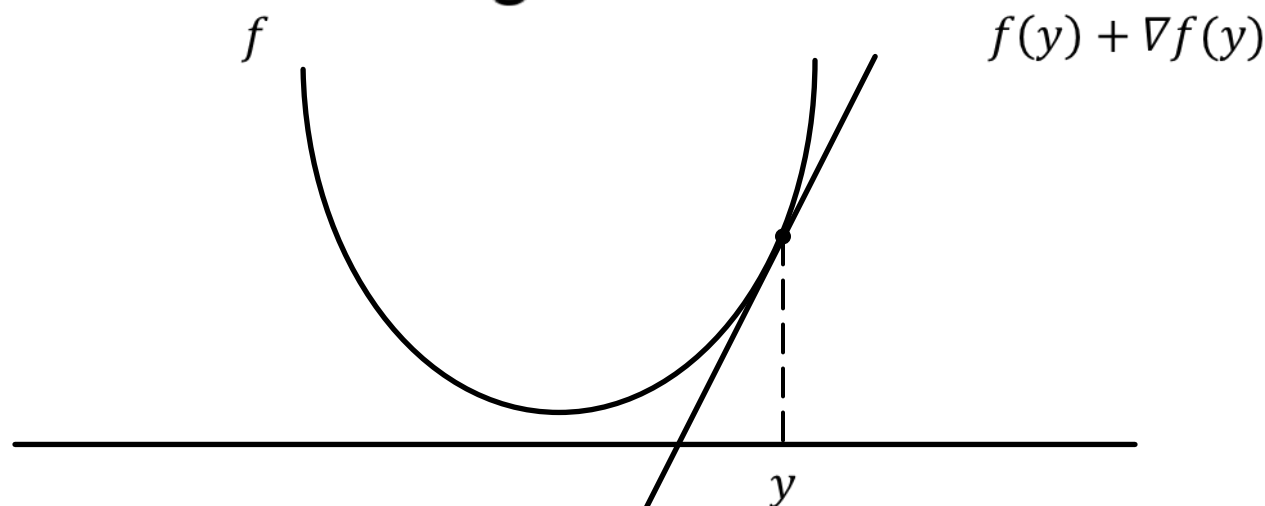
Interpolation Weights

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

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

- **Gradient decent:**

  - **Iterate until convergence:** $w \leftarrow w - \eta \nabla_w J$ η … learning rate

  - **where $\nabla_w J$ is the gradient (derivative evaluated on data):**

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

$$\text{for } j \in \{N(i;x), \forall i, \forall x\}$$
$$\text{else } \frac{\partial J(w)}{\partial w_{ij}} = 0$$

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

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

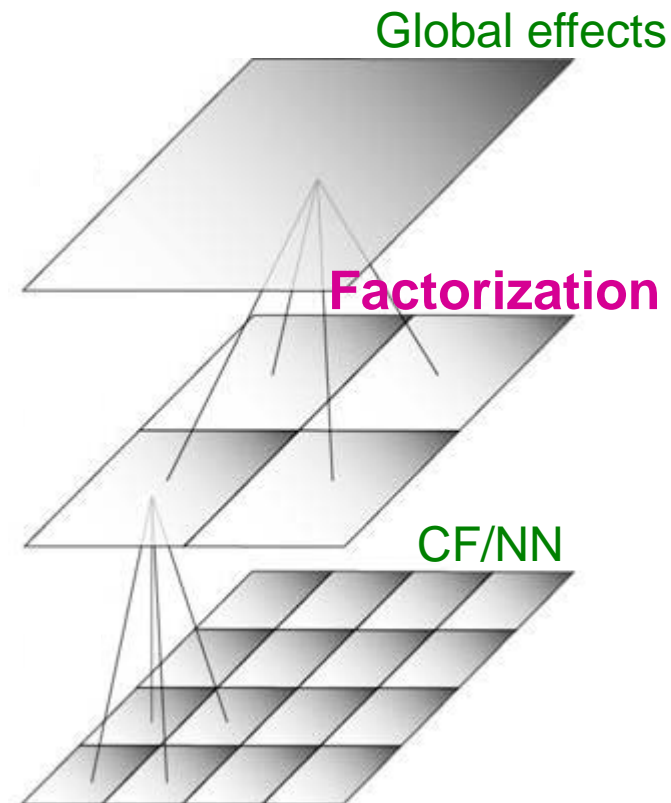# Interpolation Weights

- **So far:** $\widehat{r_{xi}} = b_{xi} + \sum_{j \in N(i;x)} w_{ij}(r_{xj} - b_{xj})$

  - Weights w_{ij} derived based on their role; **no use of an arbitrary similarity measure** ($w_{ij} \neq s_{ij}$)
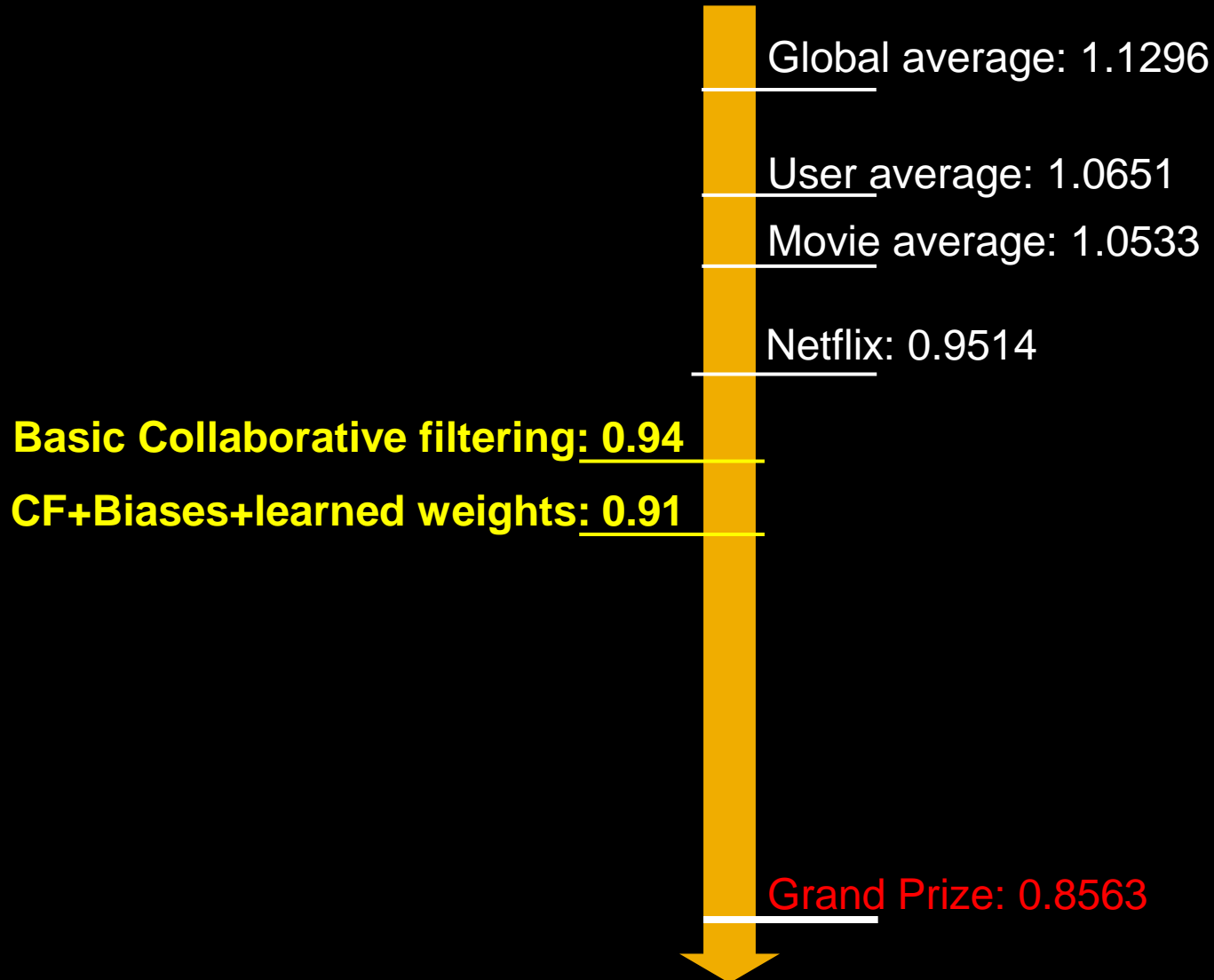
  - Explicitly account for interrelationships among the neighboring movies
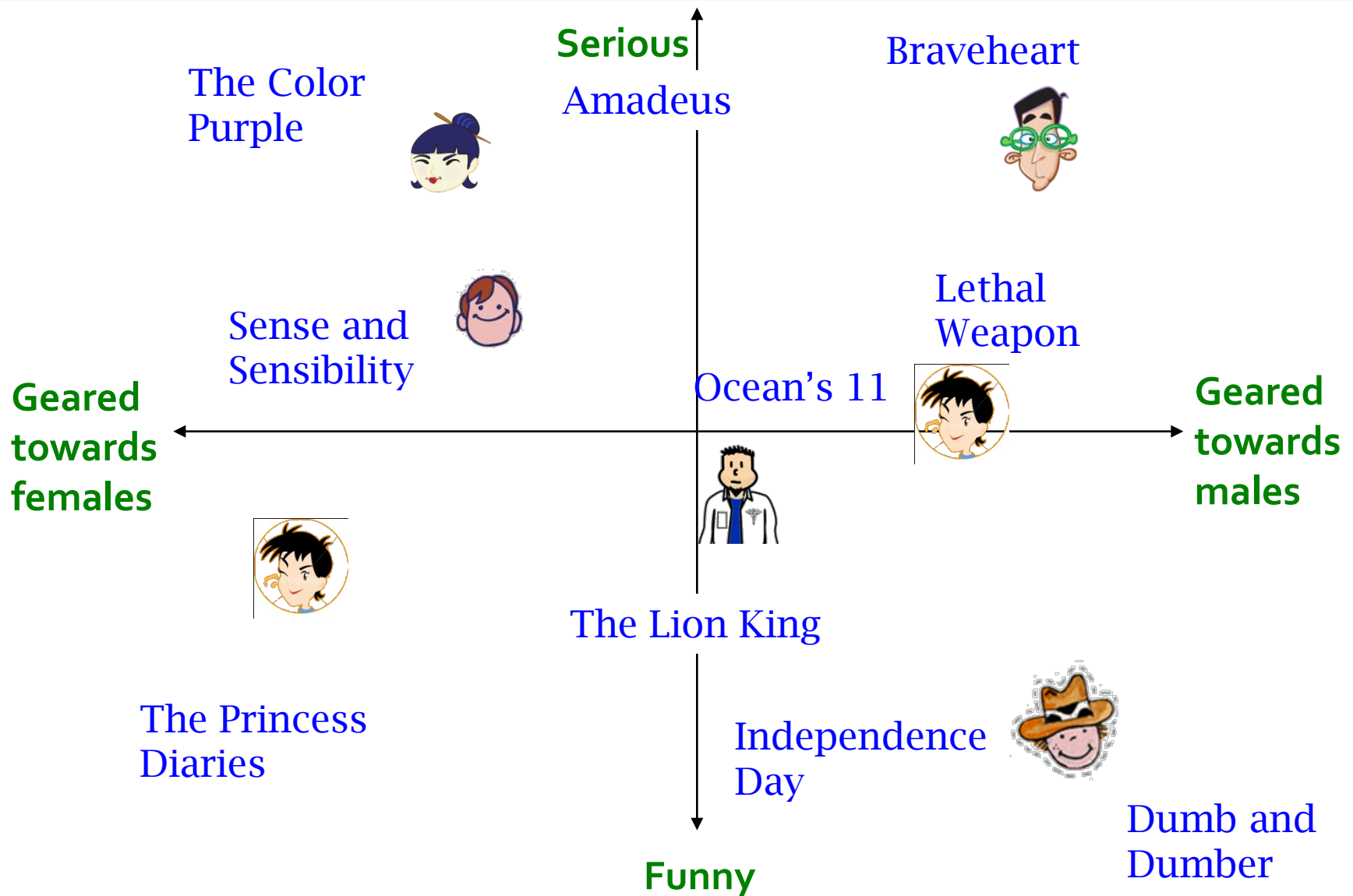
- **Next: Latent factor model**

  - Extract "regional" correlations

Global effects

Factorization

CF/NN

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 (e.g., SVD)

Latent Factor Models

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

- **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 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 / factors

.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

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

Latent Factor Models

Serious

The Color Purple

Braveheart

Amadeus

Lethal Weapon

Sense and Sensibility

Ocean's 11

Factor 1

Geared towards females

Geared towards males

The Lion King

Factor 2

The Princess Diaries

Independence Day

Dumb and Dumber

Funny

Latent Factor Models

Recap: SVD

- **Remember SVD:**
  - **A**: Input data matrix
  - **U**: Left singular vecs
  - **V**: Right singular vecs
  - Σ: Singular values
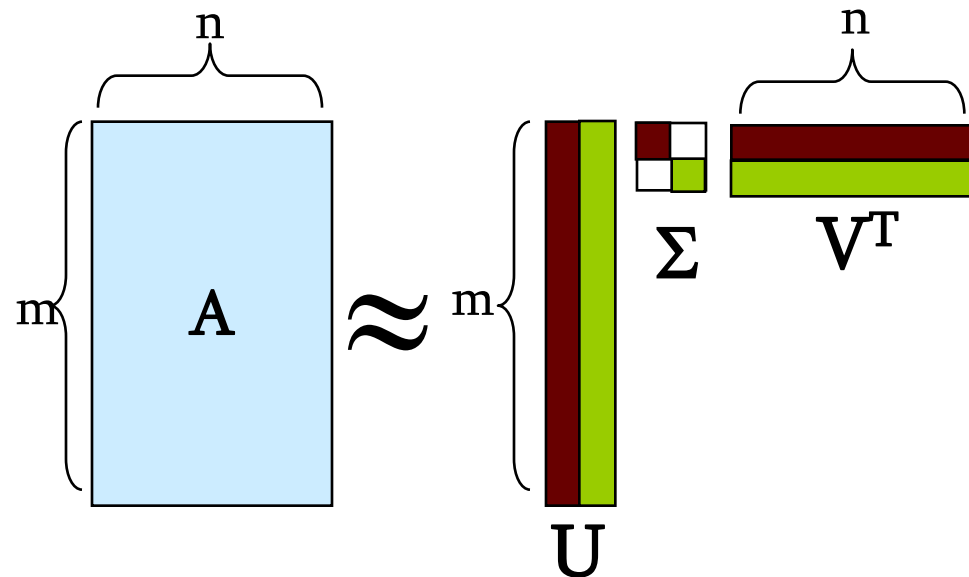
- **So in our case:**
  "SVD" on Netflix data: $R \approx Q \cdot P^T$
  $A = R, \; Q = U, \; P^T = \Sigma V^T$

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

SVD: More good stuff

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

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

- **Note two things:**

  - **SSE** and **RMSE** are monotonically related:

    - $RMSE = \frac{1}{c}\sqrt{SSE}$ **Great news: SVD is minimizing RMSE**

  - **Complication:** The sum in SVD error term is over all entries (no-rating in interpreted as zero-rating). But our **R** has missing entries!

users

factors

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

items (ratings matrix 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	

$\approx$

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

- **SVD isn't defined when entries are missing!**
- **Use specialized methods to find *P, Q***

$$\min_{P,Q} \sum_{(i,x)\in \mathrm{R}} (r_{xi} - q_i \cdot p_x)^2 \qquad \hat{r}_{xi} = q_i \cdot p_x$$

- **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

Finding the Latent Factors

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

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

users

factors

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

Q

items

≈

items

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

P^T
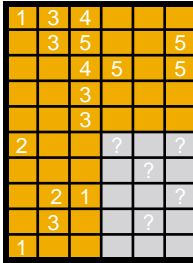
factors

Back to Our Problem

- **Want to minimize SSE for unseen test data**
- **Idea: Minimize SSE on <u>training</u> data**
  - Want large k (# of factors) to capture all the signals
  - But, **SSE** on <u>**test data**</u> 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

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

$$\min_{P,Q} \underbrace{\sum_{training} (r_{xi} - q_i p_x)^2}_{\text{"error"}} + \left[\underbrace{\lambda_1 \sum_x \|p_x\|^2 + \lambda_2 \sum_i \|q_i\|^2}_{\text{"length"}} \right]$$
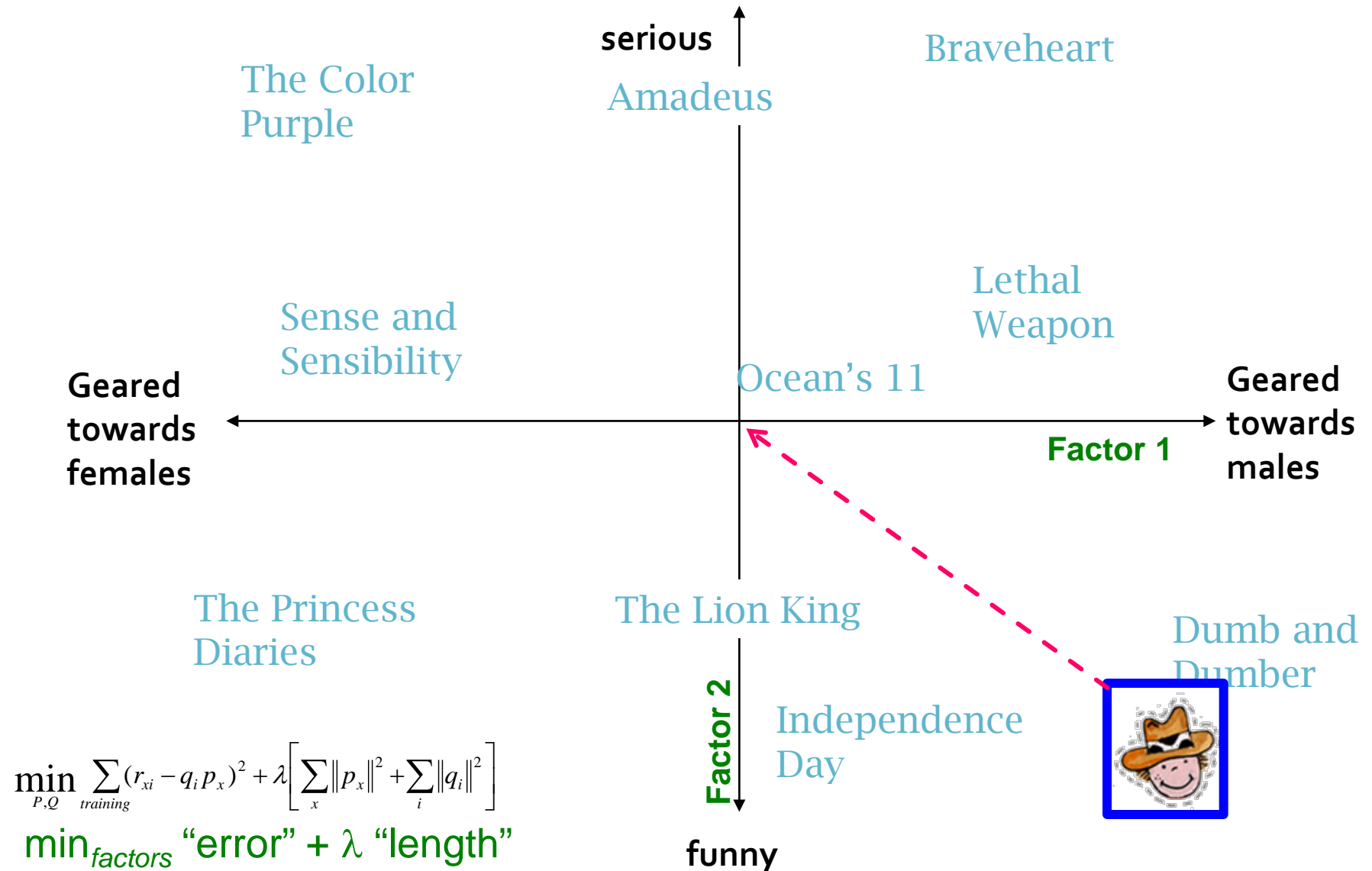
λ_1, λ_2 … user set regularization parameters

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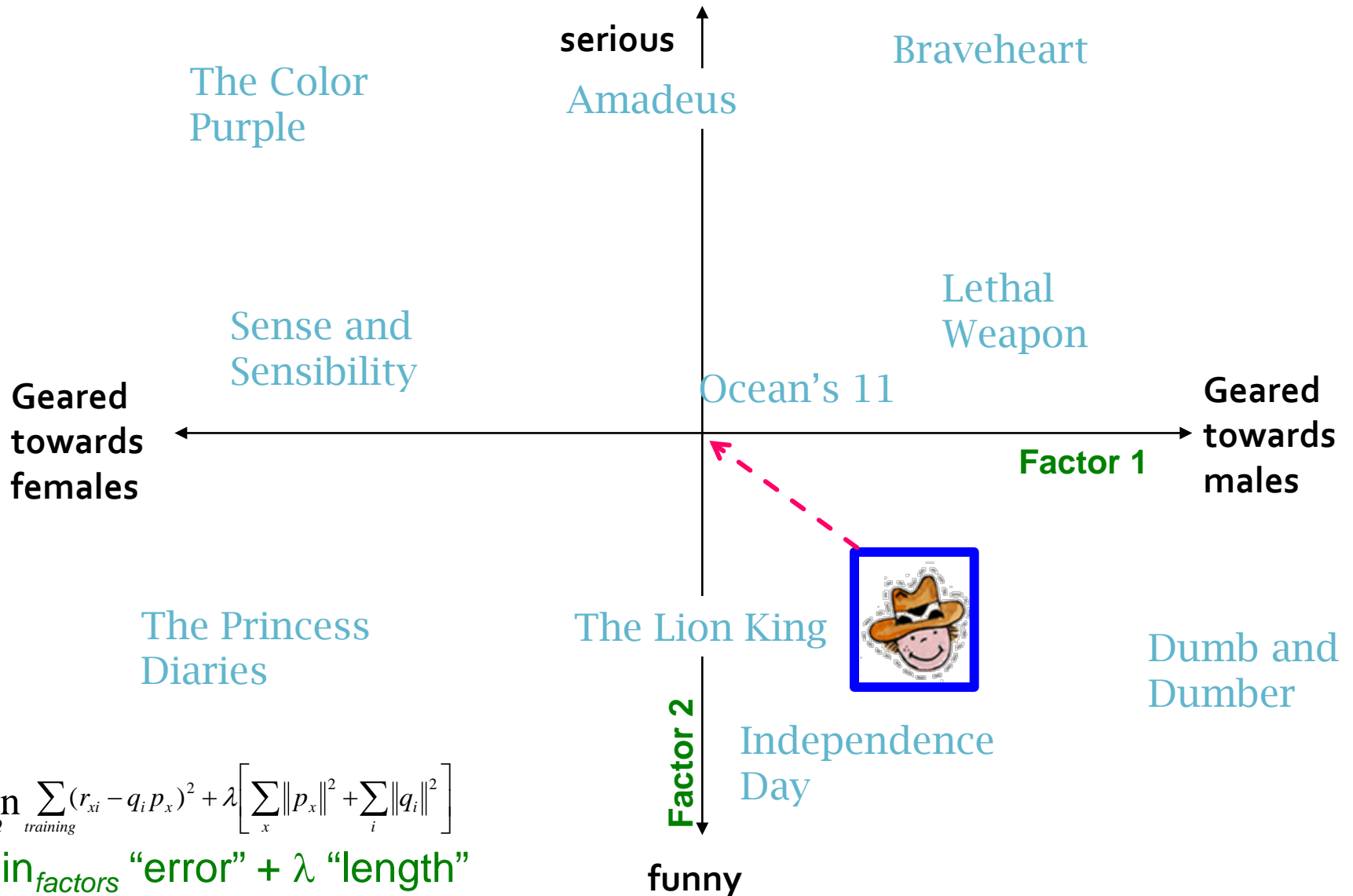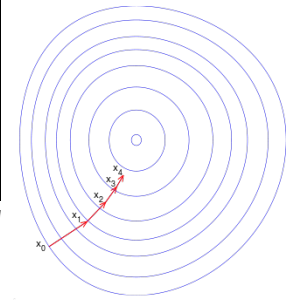
The Effect of Regularization

serious

The Color
Purple

Braveheart

Amadeus

Lethal
Weapon

Sense and
Sensibility

Ocean's 11

**Geared
towards
females**

Factor 1

**Geared
towards
males**

The Princess
Diaries

The Lion King

Dumb and
Dumber

Factor 2

Independence
Day

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

$\min_{factors}$ "error" + λ "length"

funny

The Effect of Regularization

serious

Braveheart

The Color Purple

Amadeus

Lethal Weapon

Sense and Sensibility

Ocean's 11

Geared towards females

Factor 1

Geared towards males

The Princess Diaries

The Lion King

Dumb and Dumber

Factor 2

Independence Day

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

$\min_{factors}$ "error" + λ "length"

funny

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

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

$\min_{factors}$ "error" + λ "length"

funny

The Effect of Regularization

serious

Braveheart

The Color
Purple

Amadeus

Lethal
Weapon

Sense and
Sensibility

Ocean's 11

**Geared
towards
females**

Factor 1

**Geared
towards
males**

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"

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 decent:**
  - Initialize **P** and **Q** (using SVD, pretend missing ratings are 0)
  - Do gradient descent:

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

    - $P \leftarrow P - \eta \cdot \nabla P$
    - $Q \leftarrow Q - \eta \cdot \nabla Q$
    - 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 q_{if} is entry f of row q_i of matrix Q
  - **Observation: Computing gradients is slow!**

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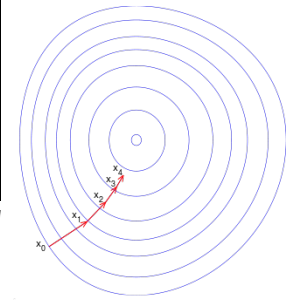
- **Gradient Descent (GD) vs. Stochastic GD**
  - **Observation:** $\nabla Q = [\nabla q_{if}]$ where

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

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

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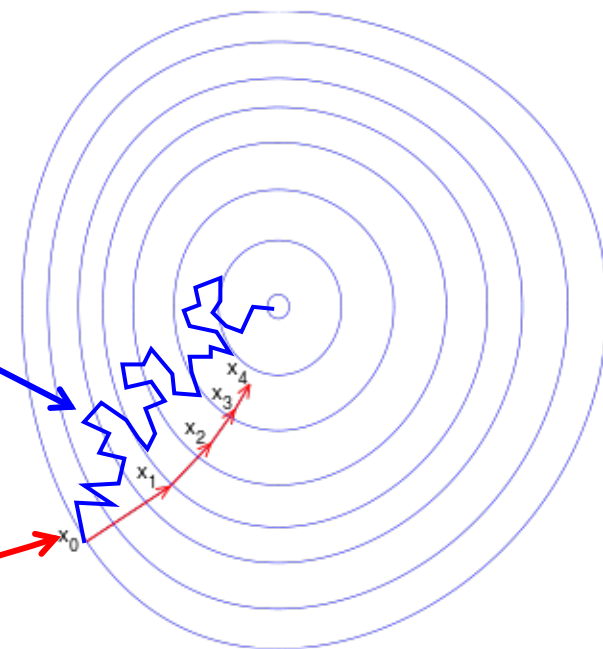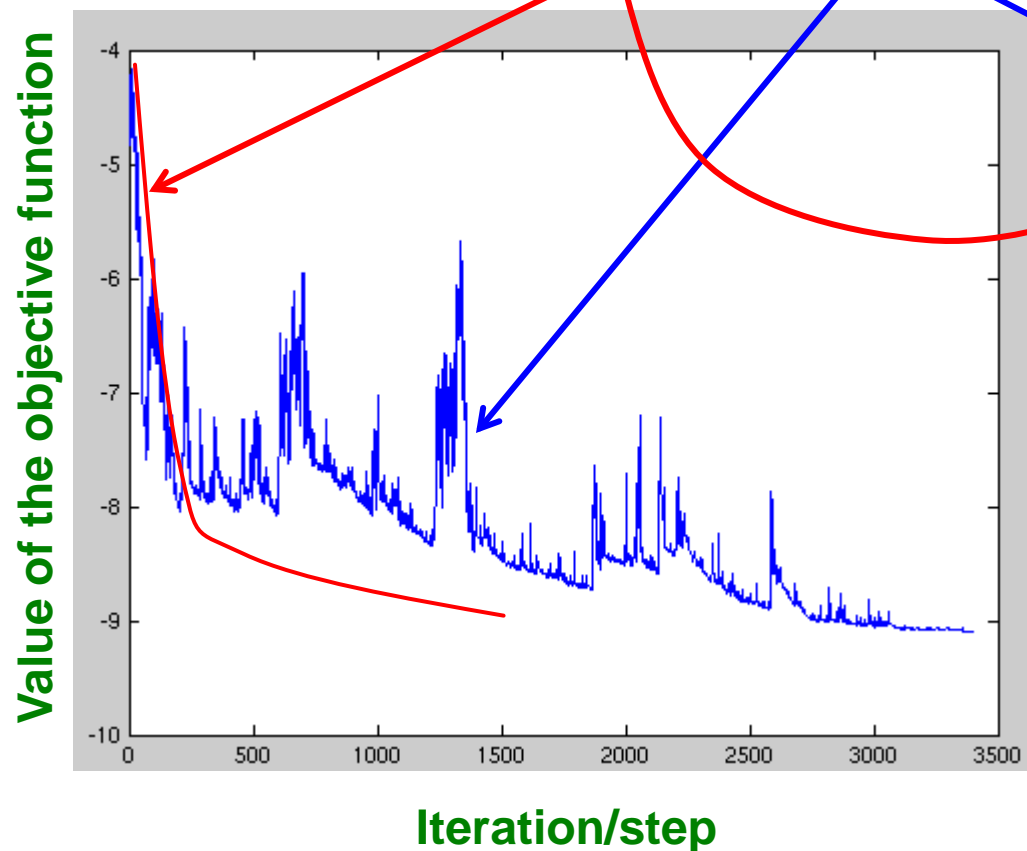
GD improves the value of the objective function at every step.
SGD improves the value but in a "noisy" way.
GD takes fewer steps to converge but each step takes much longer to compute.
In practice, **SGD** is much faster!

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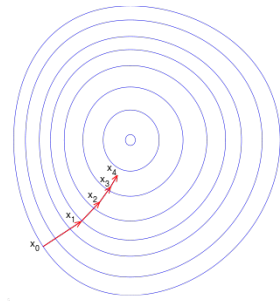
- **Stochastic gradient decent:**
  - Initialize **P** and **Q** (using SVD, pretend missing ratings are 0)
  - Then iterate over the ratings (multiple times if necessary) and update factors:

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

                                        μ ... learning rate

- **2 for loops:**
  - For until convergence:
    - For each r_{xi}
      - Compute gradient, do a "step"
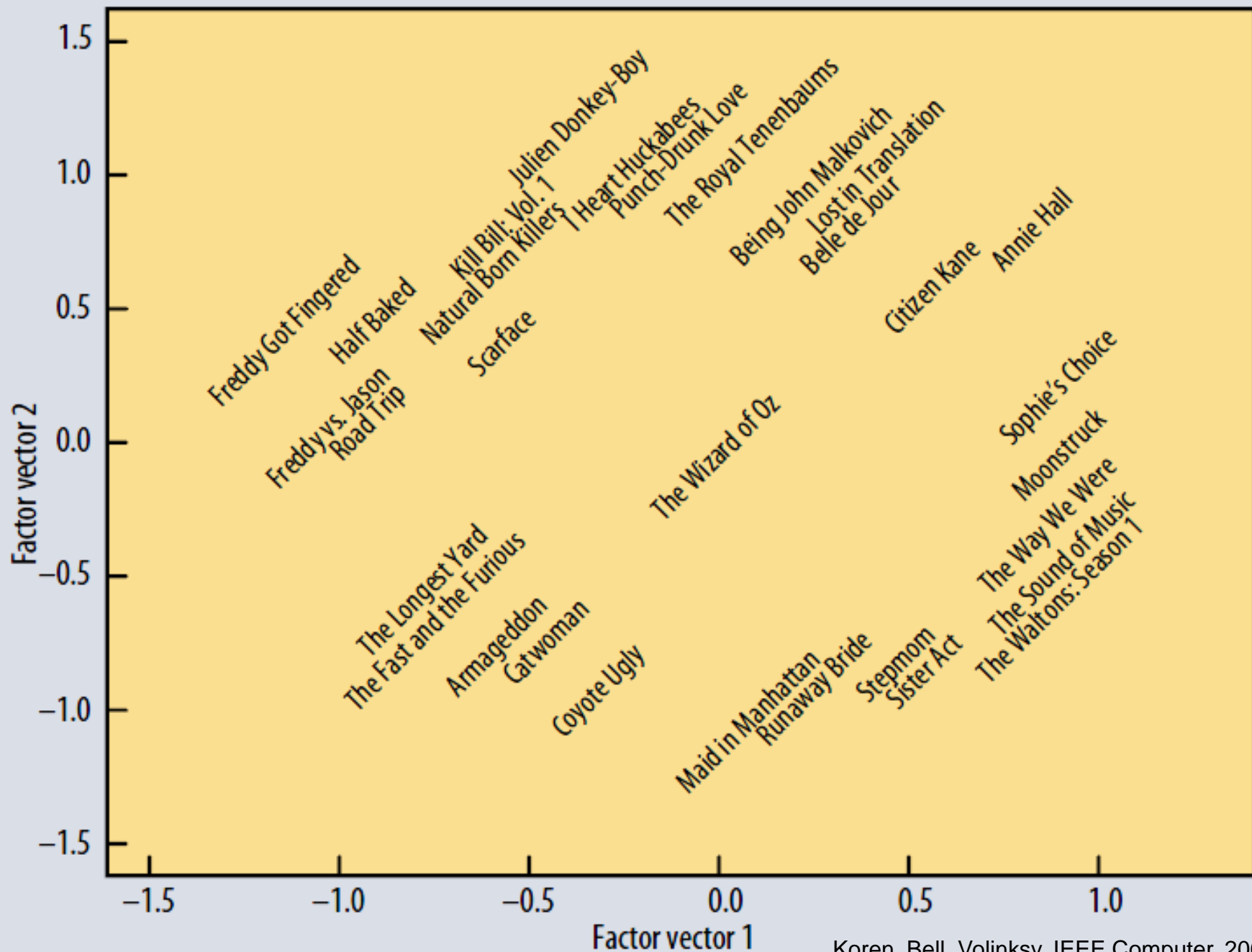
Koren, Bell, Volinksy, IEEE Computer, 2009

Extending Latent Factor Model to Include Biases

Modeling Biases and Interactions

user bias   **movie bias**   **user-movie interaction**
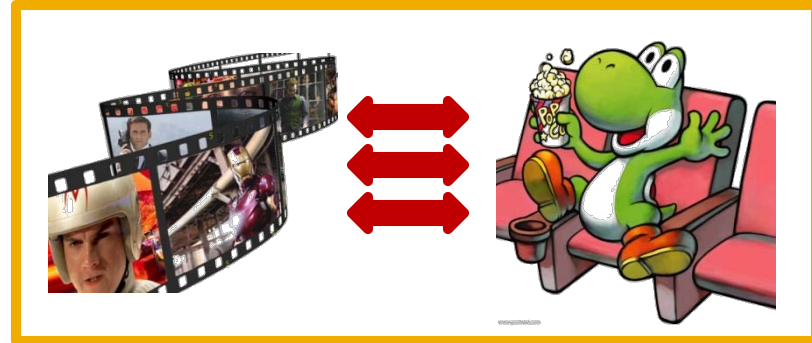
Baseline predictor

- 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

- 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 (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

$$r_{xi} = \mu + b_x + b_i + q_i \cdot p_x$$

Overall mean rating   Bias for user x Bias for movie i User-Movie interaction

- **Example:**
  - Mean rating: μ = **3.7**
  - You are a critical reviewer: your ratings are 1 star lower than the mean: b_x = **-1**
  - Star Wars gets a mean rating of *0.5* higher than average movie: b_i = **+ 0.5**
  - Predicted rating for you on Star Wars:
    = **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 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

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
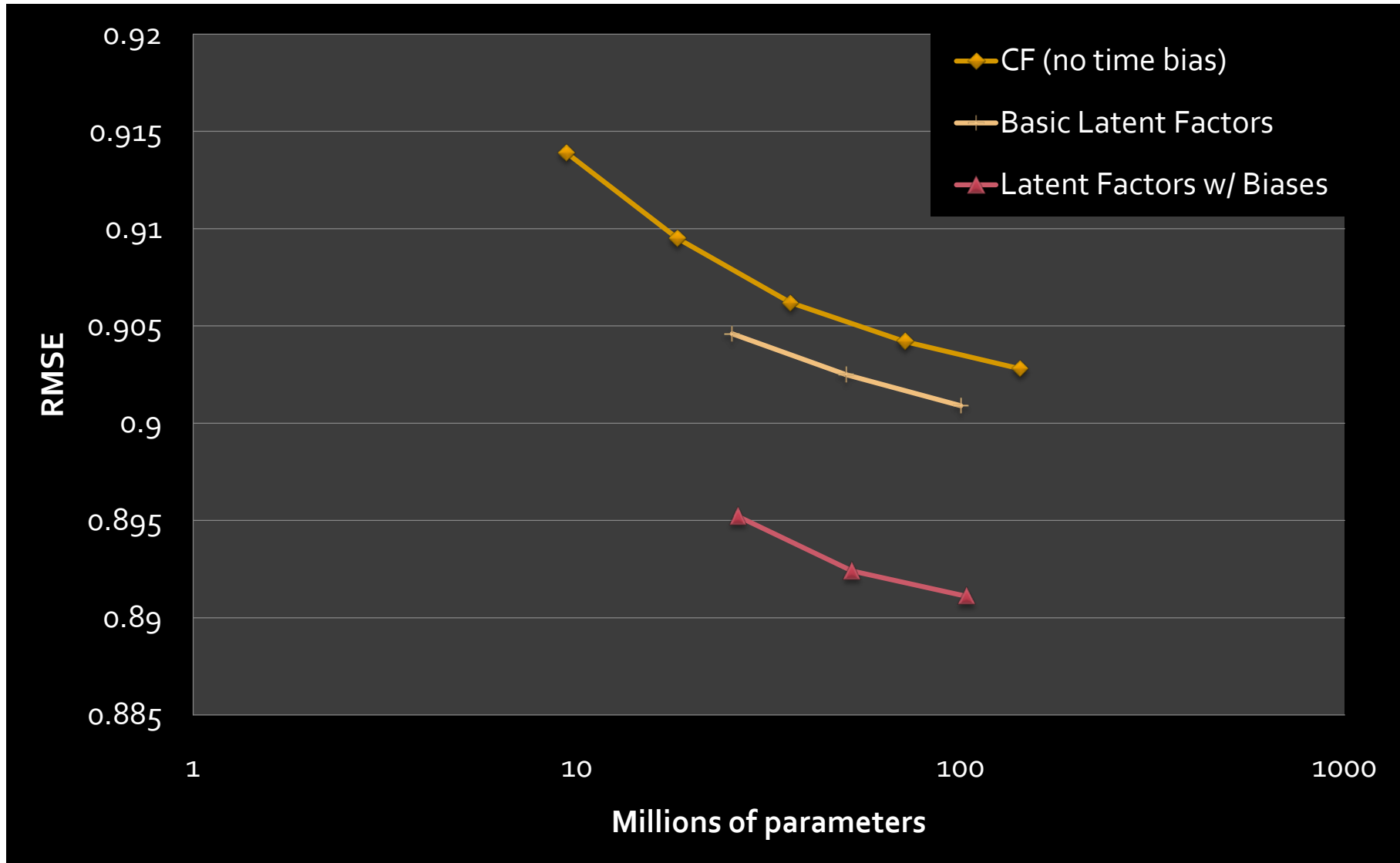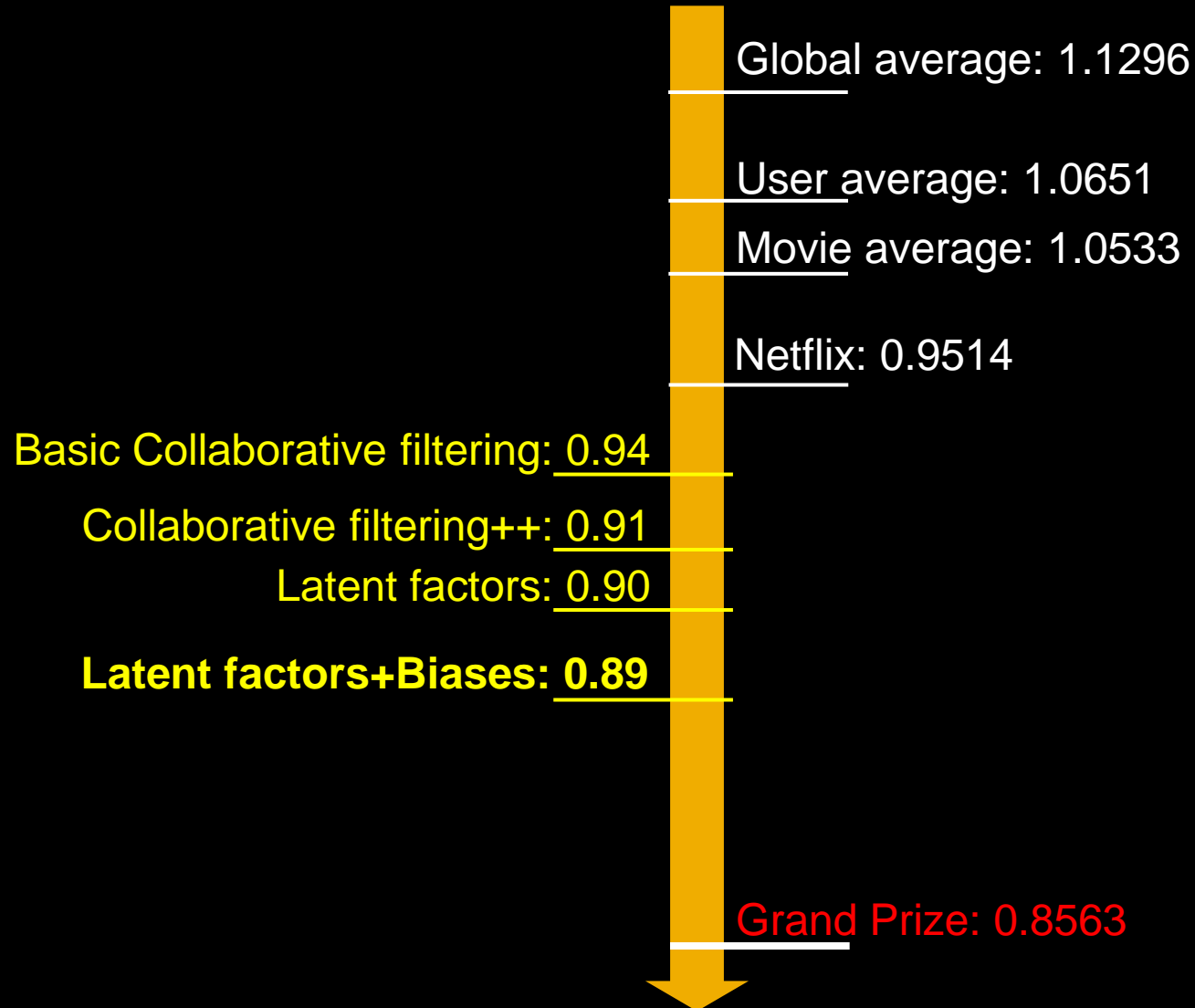
The Netflix Challenge: 2006-09

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
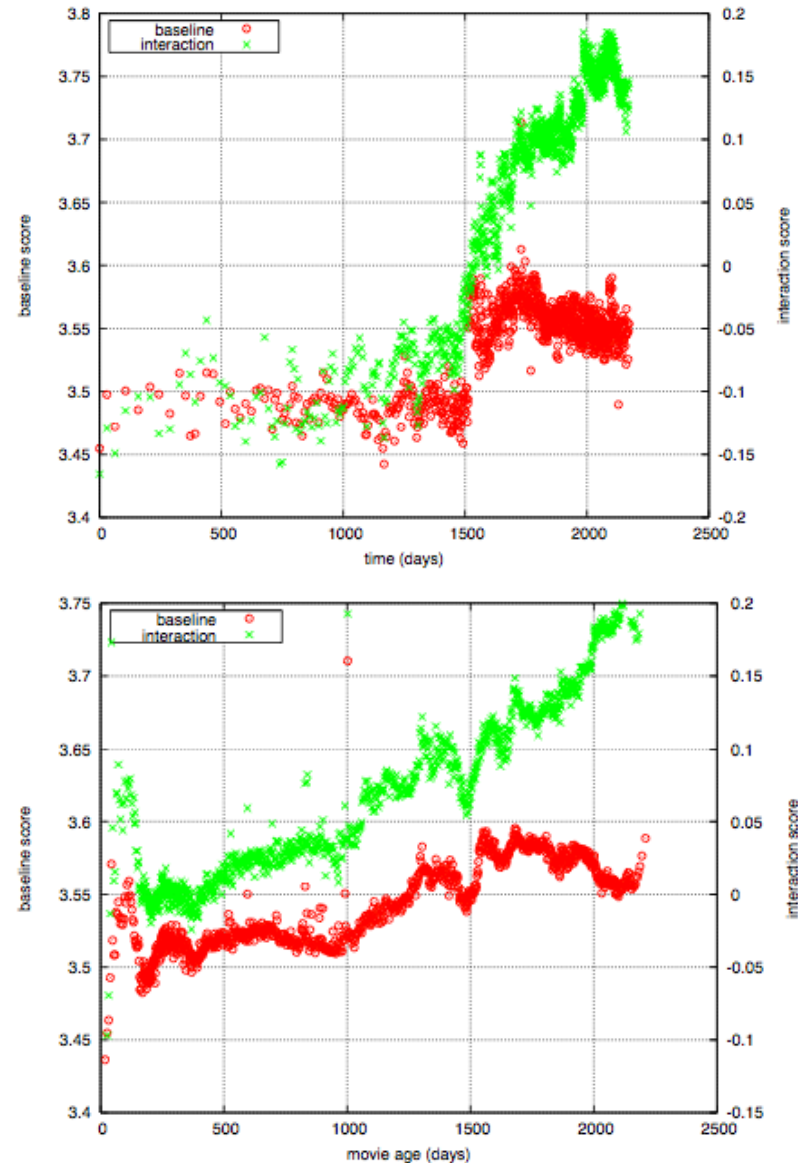
Temporal Biases & Factors

- **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 b_x and 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

Adding Temporal Effects

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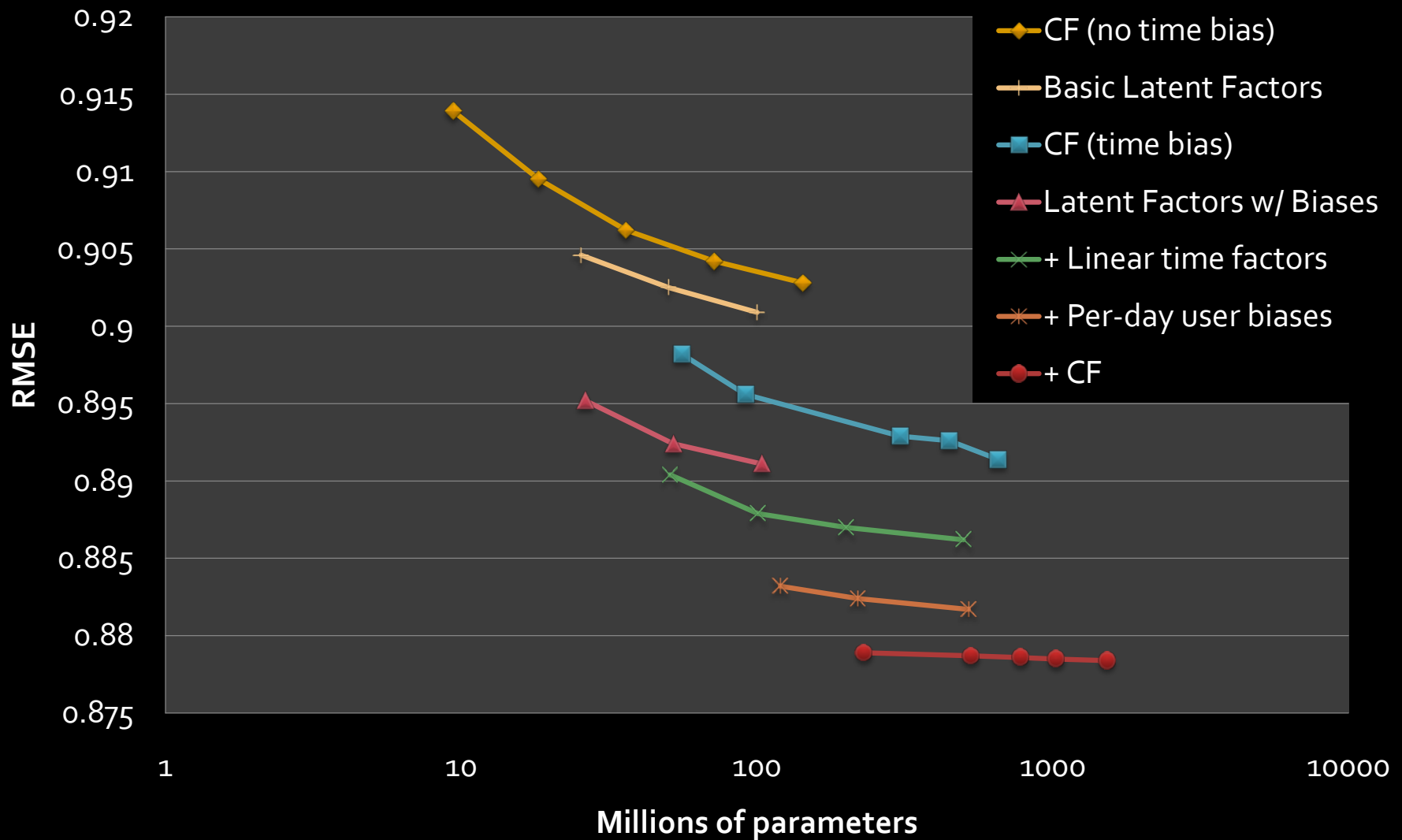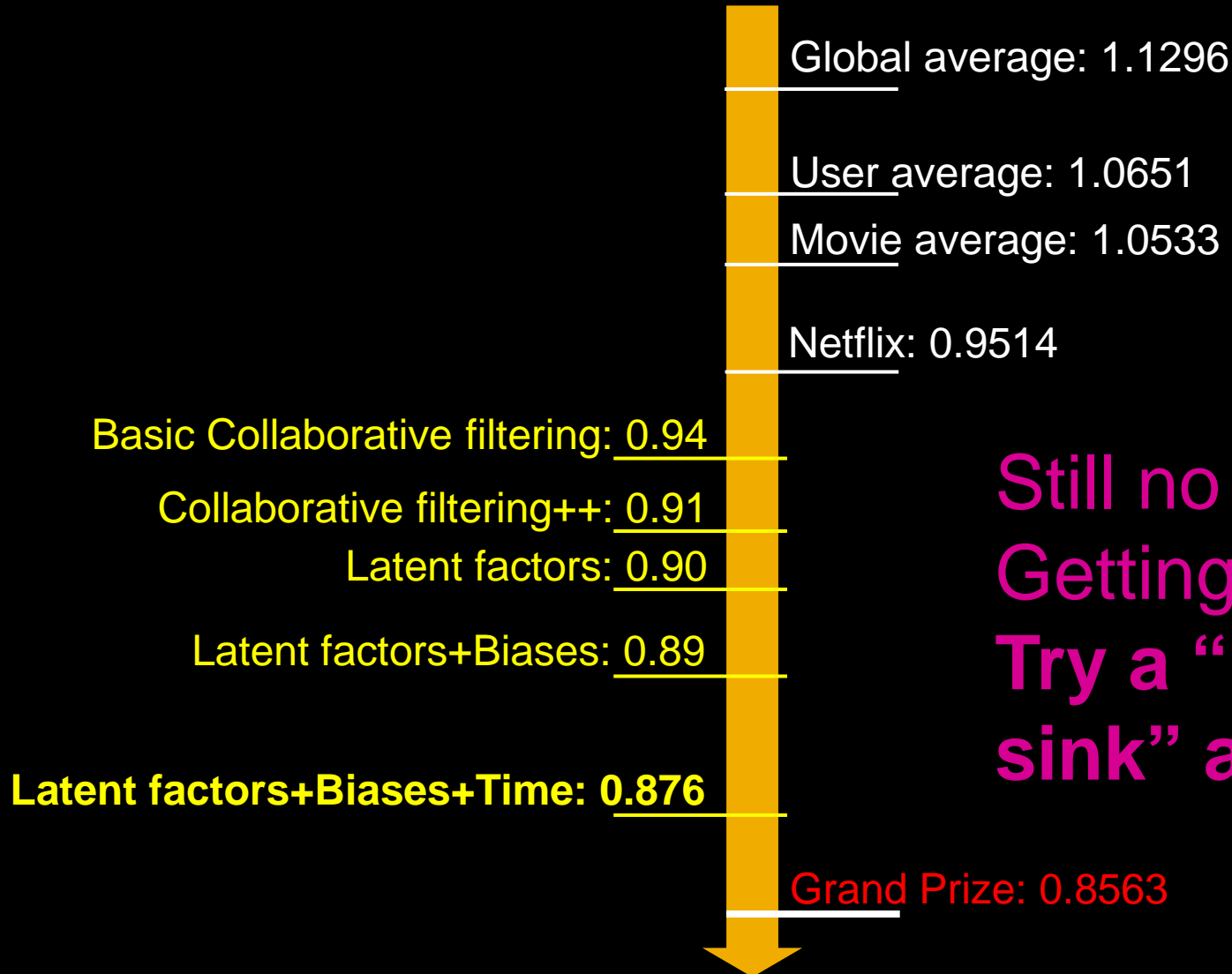
Still no prize! ☹
Getting desperate.
Try a "kitchen sink" approach!

The big picture
Solution of BellKor's Pragmatic Chaos

All developed CF models

BRISMF, SBRAMF, SVD-Time, MF1, NSVDD, RBM day, Split RBM, BK3, BK2, BK4, BK1, BK5-SVD++, Movie KNN V, Baseline 1/2/3, GE, FRBM, SVD++, NSVD2, GTE, KNN+time, DRBM, MF2, NSVD1, Integrated M., RBM, 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

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 that they are in direct competition with Ensemble

- **Strategy**
  - Both teams carefully monitoring the leaderboard
  - 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….**

Million $ Awarded Sept 21ˢᵗ 2009

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
  - http://www2.research.att.com/~volinsky/netflix/bpc.html
  - http://www.the-ensemble.com/