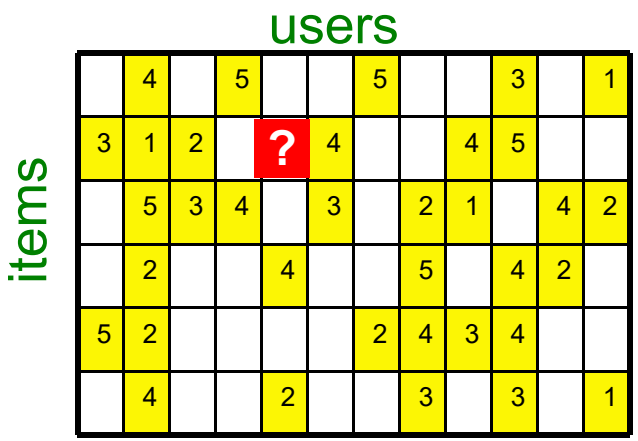
Ratings as Products of Factors

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



$$\hat{r}_{xi} = q_i \cdot p_x^{\mathrm{T}} = \sum_f q_{if} \cdot p_{xf}$$

q_i = row i of Q
p_x = column x of P^{T}

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

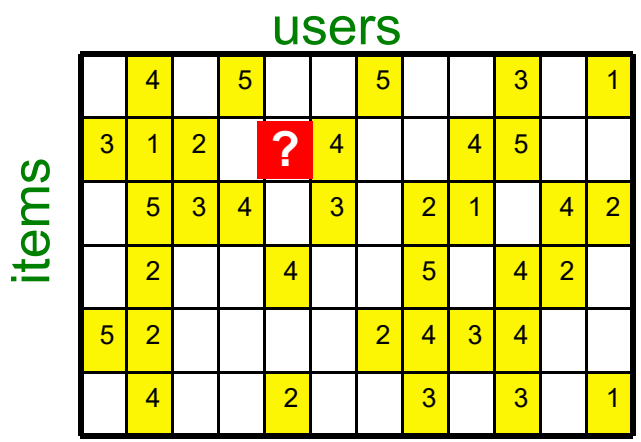
items × factors = Q

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

factors × users = P^{T}

Ratings as Products of Factors

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



$$\hat{r}_{xi} = q_i \cdot p_x^{\mathrm{T}} = \sum_f q_{if} \cdot p_{xf}$$

q_i = row i of Q
p_x = column x of P^{T}

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

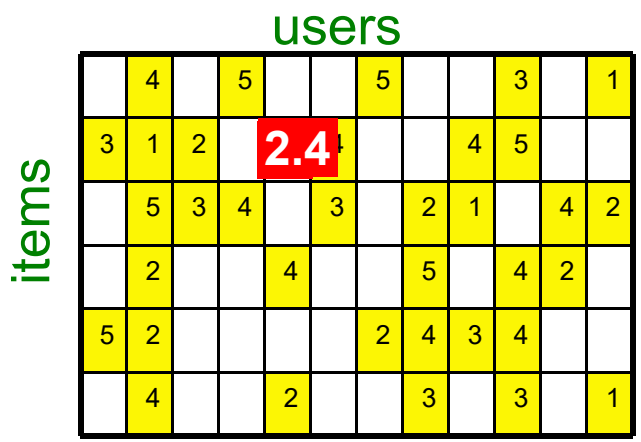
items, factors — Q

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

factors, users — P^{T}

Ratings as Products of Factors

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



$\approx$

$$\hat{r}_{xi} = q_i \cdot p_x^{\mathrm{T}} = \sum_f q_{if} \cdot p_{xf}$$

q_i = row i of Q
p_x = column x of P^{T}

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

items, *f* factors

Q

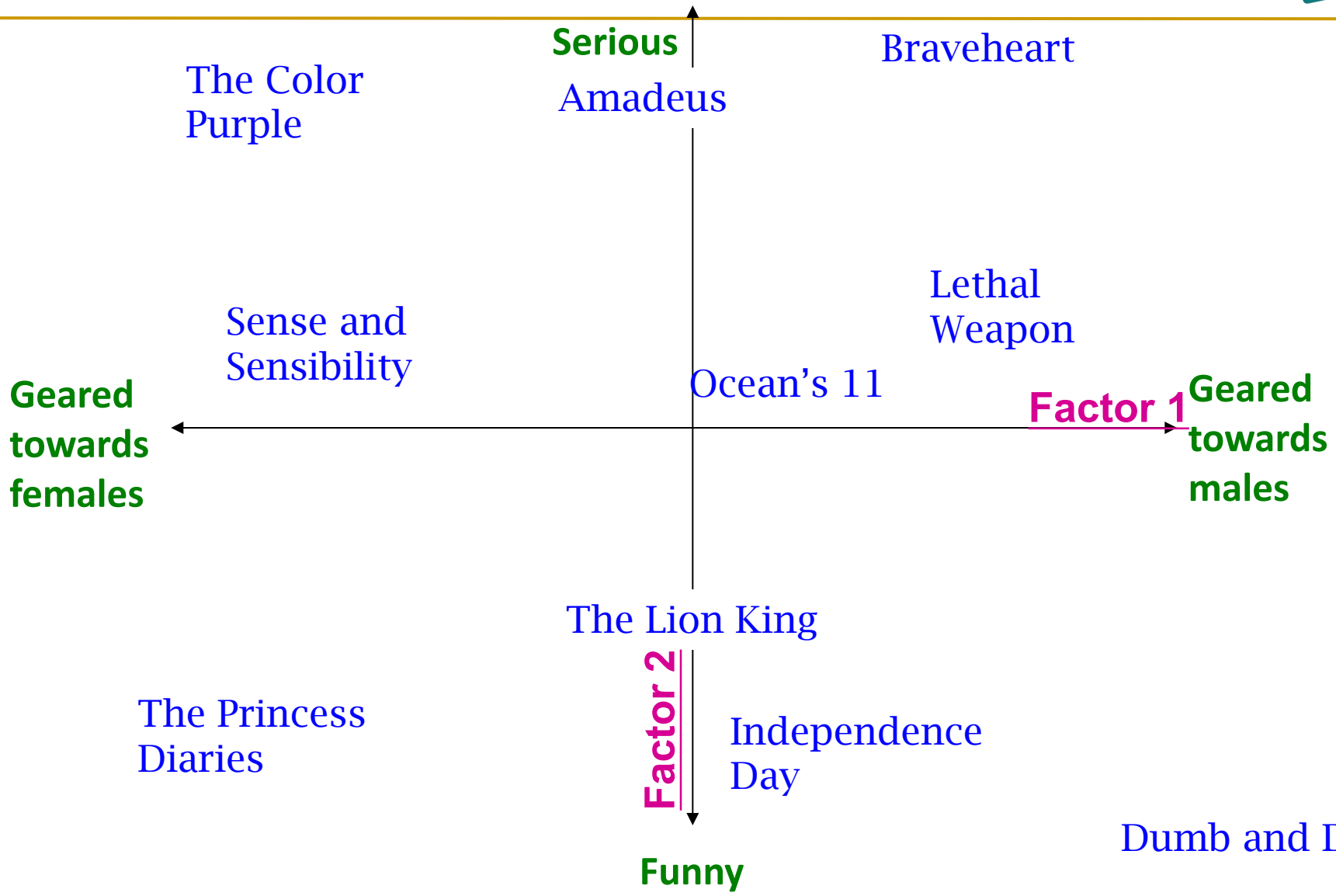
$\cdot$

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

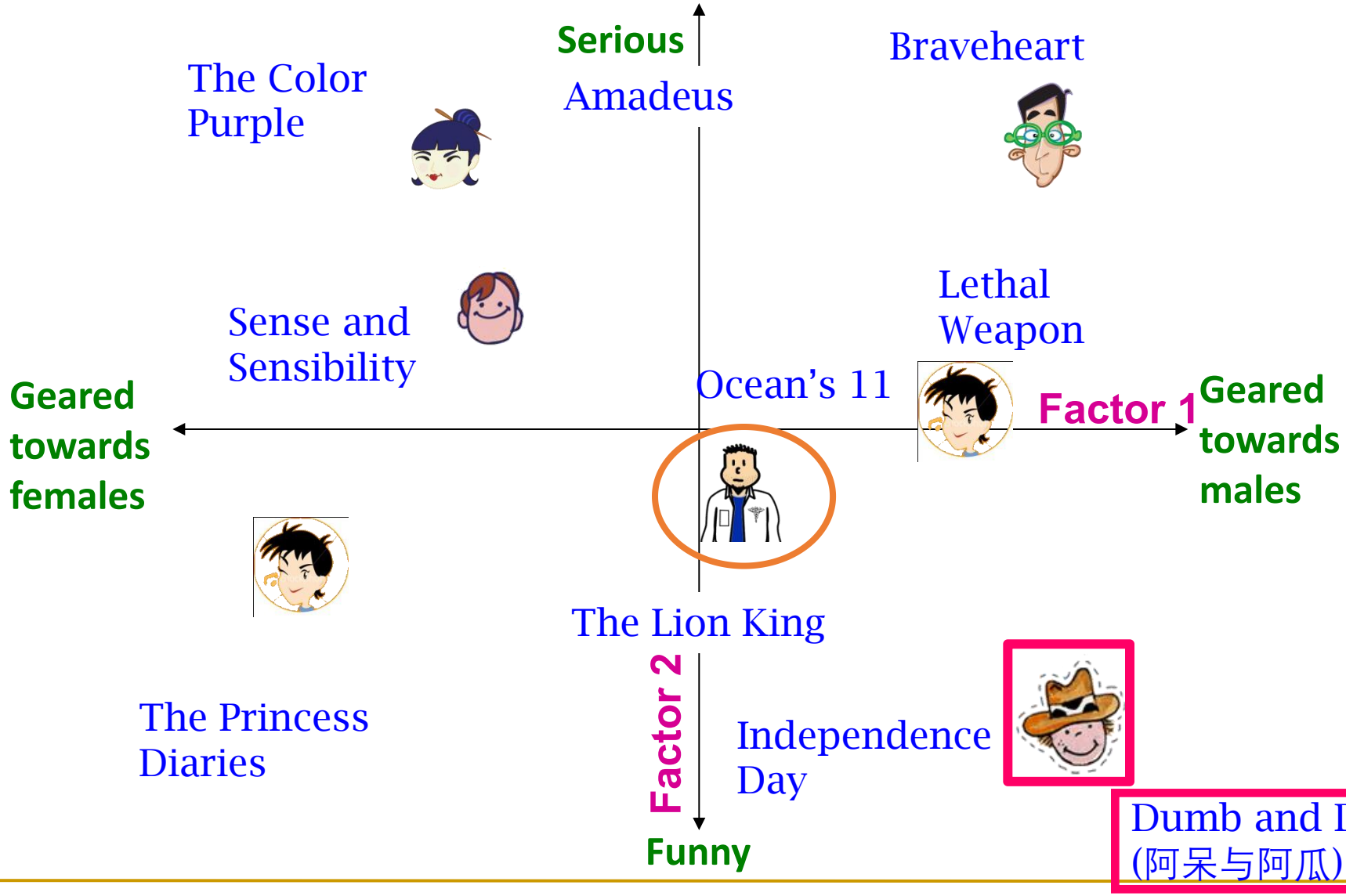
f factors, users

P^{T}

Latent Factor Models



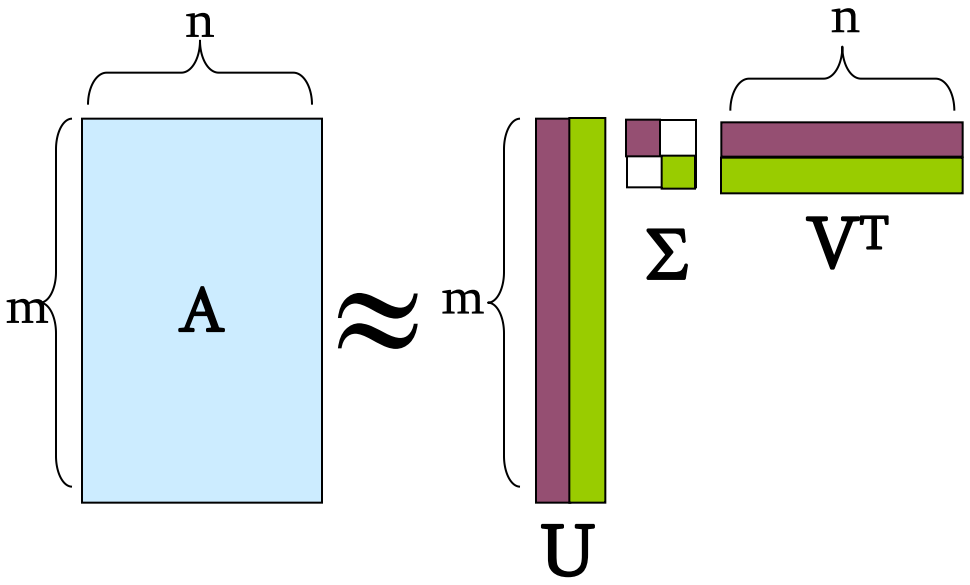
Latent Factor Models



SVD

- **SVD** (下个章节会详细讲解):
  - **A**: Input data matrix
  - **U**: Left singular vecs
  - **V**: Right singular vecs
  - Σ: Singular values

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



- **So in our case:**

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

  $A = R,\ Q = U,\ P^T = \Sigma V^T$

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

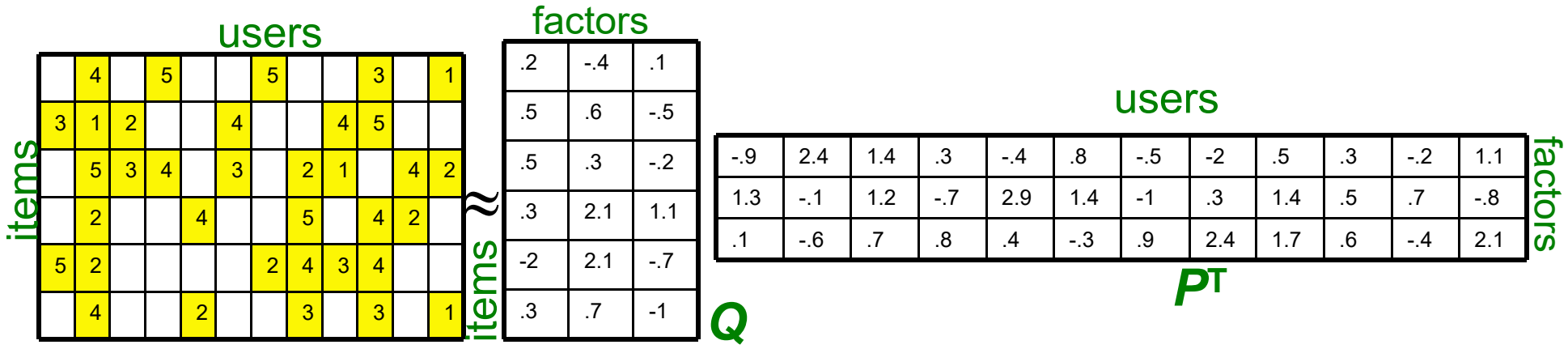
SVD: More good stuff

- **SVD gives minimum reconstruction error (Sum of Squared Errors, SSE):**

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

- **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!

Latent Factor Models



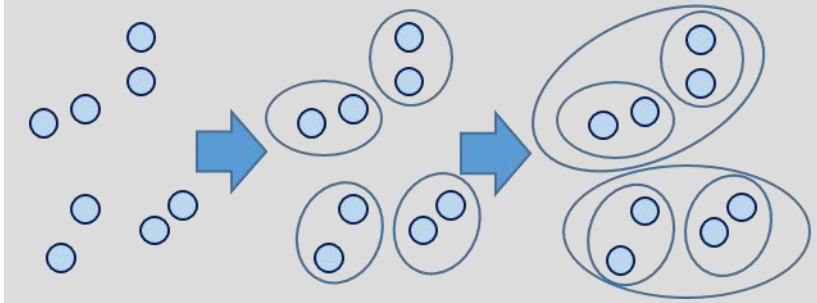
	4		5			5			3		1
3	1	2			4			4	5		
	5	3	4		3		2	1		4	2
	2			4			5		4	2	
5	2					2	4	3	4		
	4			2			3		3		1

≈

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

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

- **Use specialized methods to find *P, Q***
  - $\min_{P,Q} \sum_{(i,x)\in R} (r_{xi} - q_i \cdot p_x^T)^2$ SSE(平方误差和)
  - **Note:**
    - We don’ t require cols of ***P, Q*** to be orthogonal/unit length
    - ***P, Q*** map users/movies to a latent space

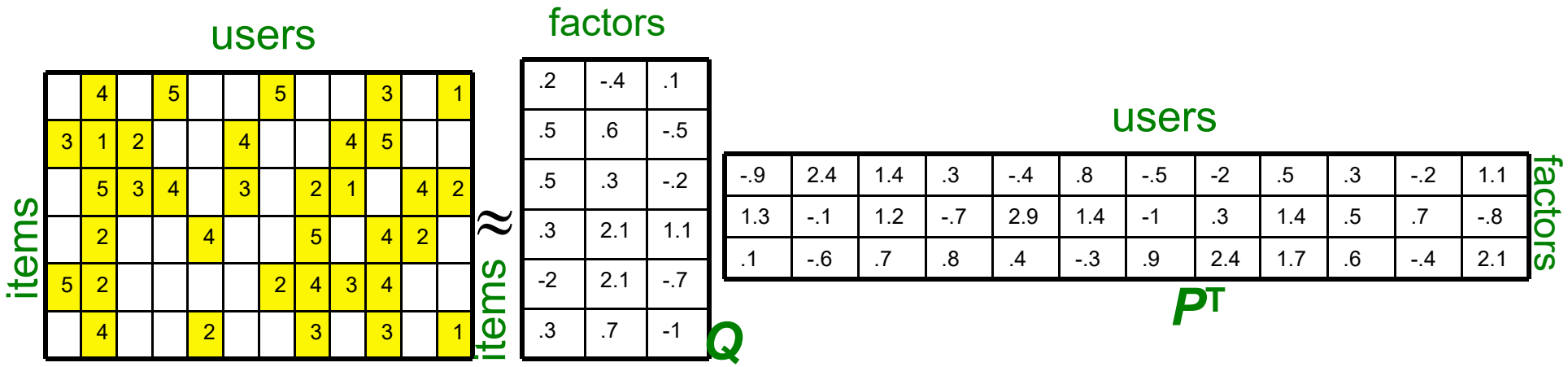


Finding the Latent Factors

Latent Factor Models

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

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



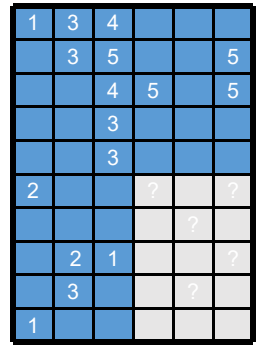
	4		5			5			3		1
3	1	2			4			4	5		
	5	3	4		3		2	1		4	2
	2			4			5		4	2	
5	2					2	4	3	4		
	4			2			3		3		1

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

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

Back to Our Problem

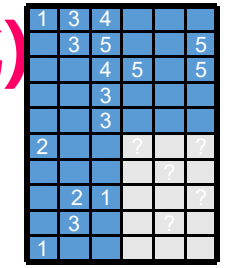
- **Want to minimize SSE for unseen test data**
- **Idea: Minimize SSE on training data**
  - Want large ***k*** (# of factors) to capture all the signals
  - But, **SSE** on **test data** begins to rise for ***k*** > 2



- This is a classical example of **overfitting(过度拟合, 过拟合):**
  - 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

- **To solve overfitting we introduce regularization(正则化)**
  - Allow rich model where there are sufficient data
  - Shrink aggressively where data are scarce



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

"error" "length"

λ_1, λ_2 … user set regularization parameters(正则化参数);
||A||, Frobenius norm (F范式) $\|A\|_F \equiv \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$

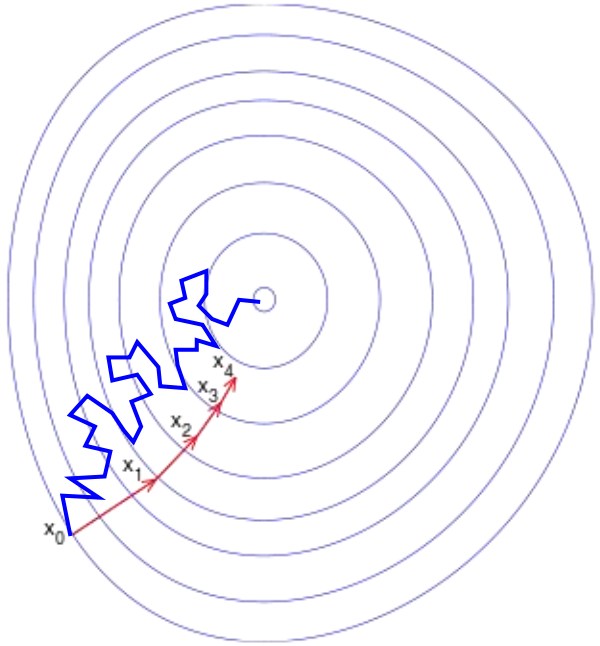
Note: We do not care about the "raw" value of the objective function, but we care in P,Q that achieve the minimum of the objective

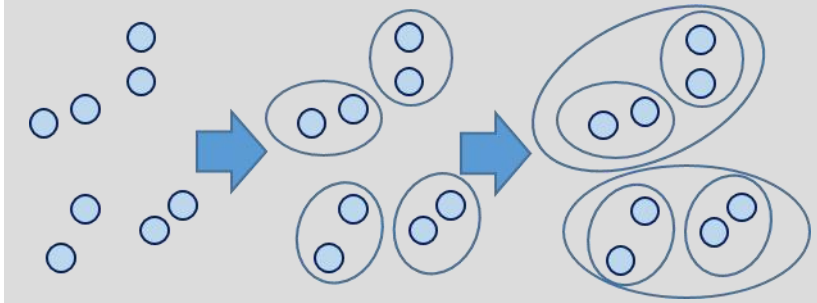
Stochastic Gradient Descent

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

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

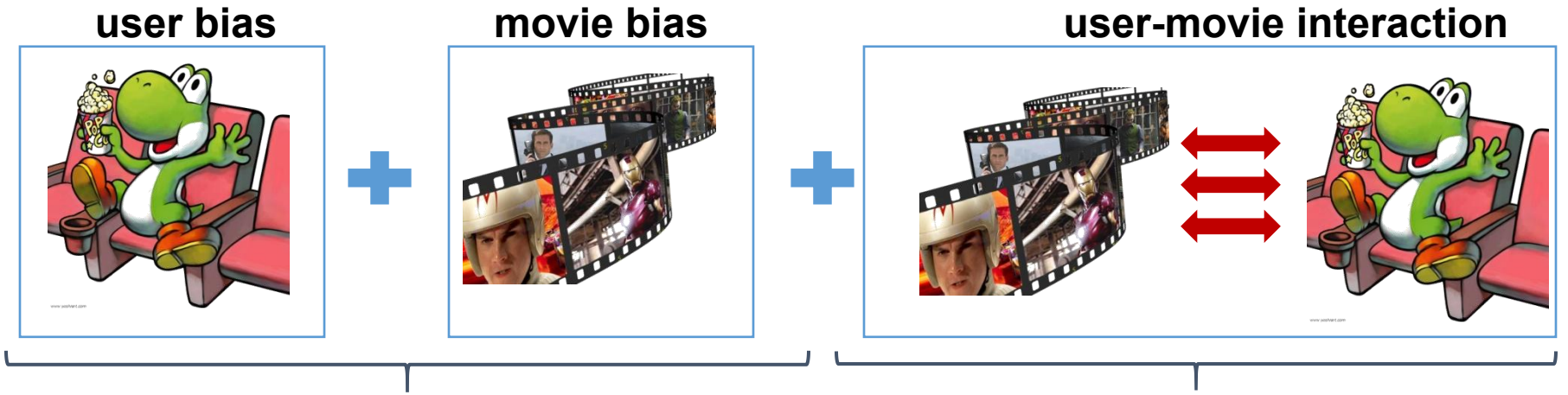
- **Ans: Stochastic gradient decent(SGD)**
  - SGD, 随机梯度下降, 感兴趣的同学自学





Extending Latent Factor Model to Include Biases

Modeling Biases and Interactions



Baseline predictor

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

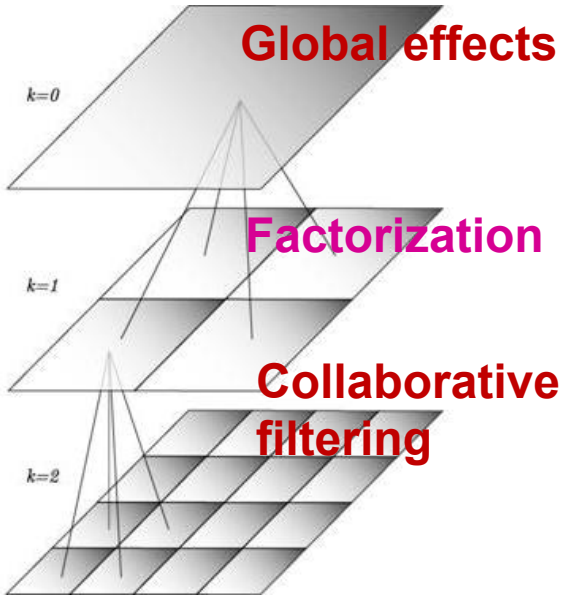
User-Movie interaction

- 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

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

Overall mean rating

Bias for user ***x***

Bias for movie ***i***

User-Movie interaction

- **Example:**
  - Mean rating: $\mu = 3.7$
  - You are a critical reviewer: your ratings are 1 star lower than the mean: $b_x = -1$
  - Star Wars (星球大战) 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

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

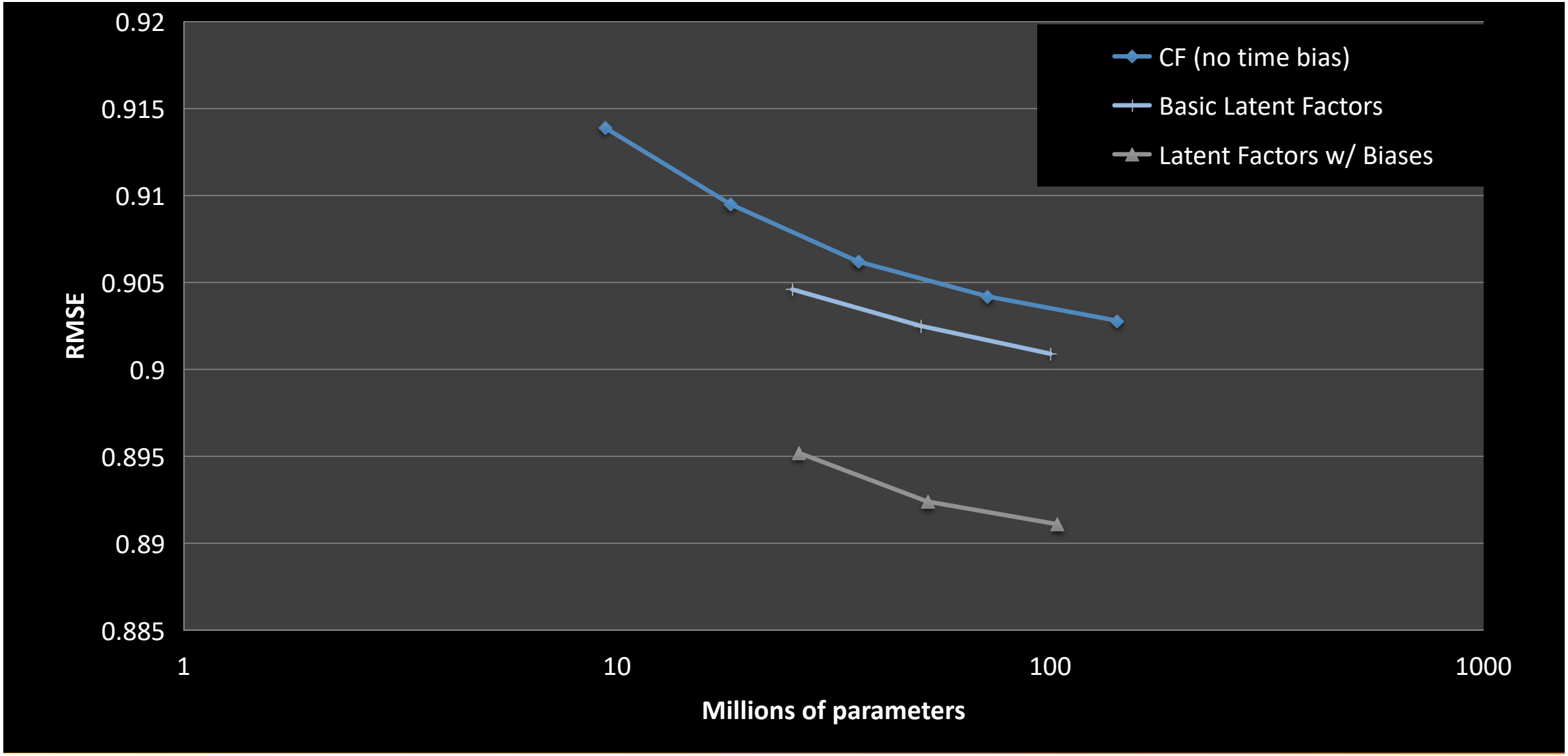
regularization

λ is selected via grid-search on a validation set

- **Stochastic gradient decent (SGD) 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)

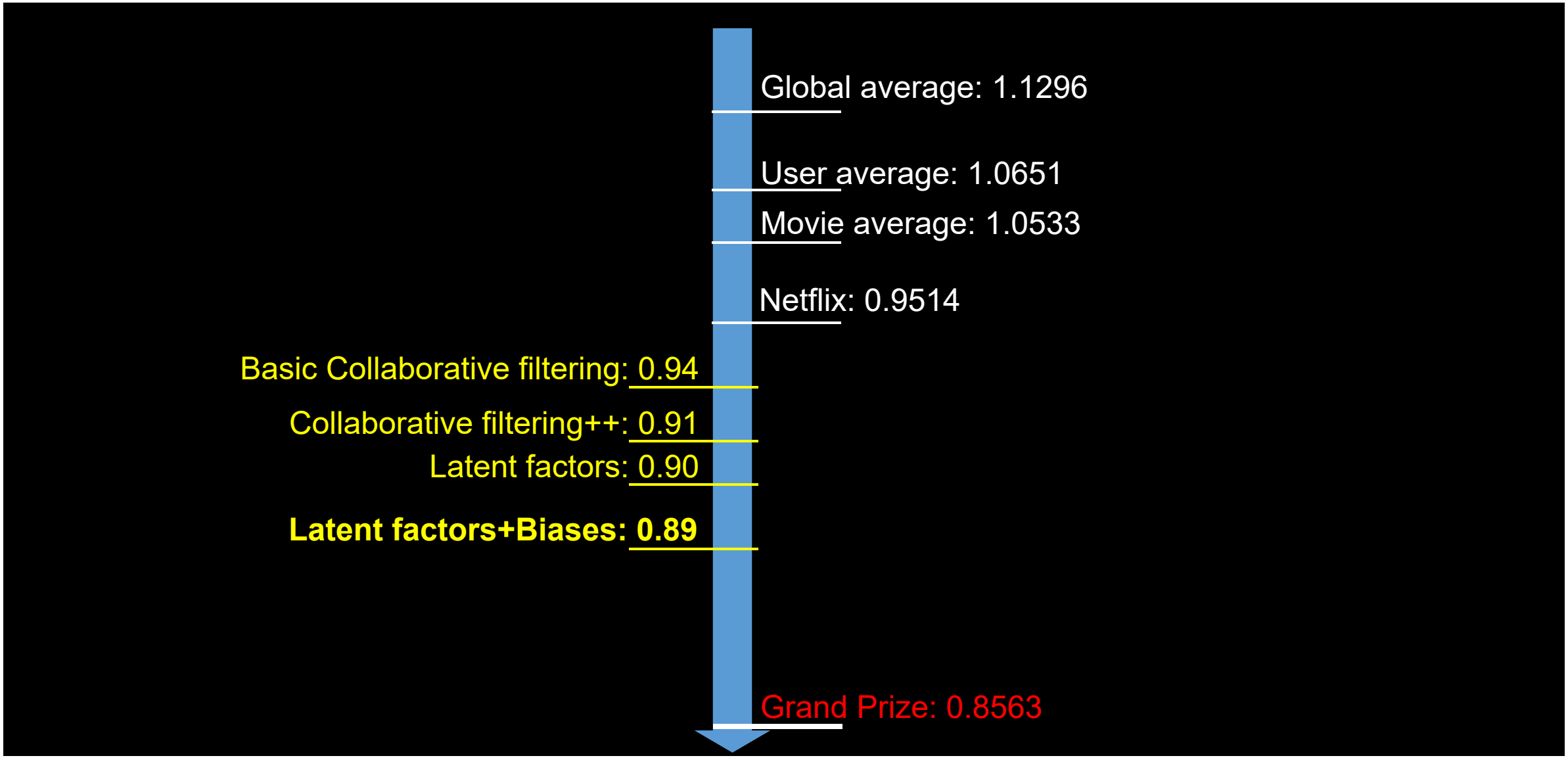
Performance of Various Methods

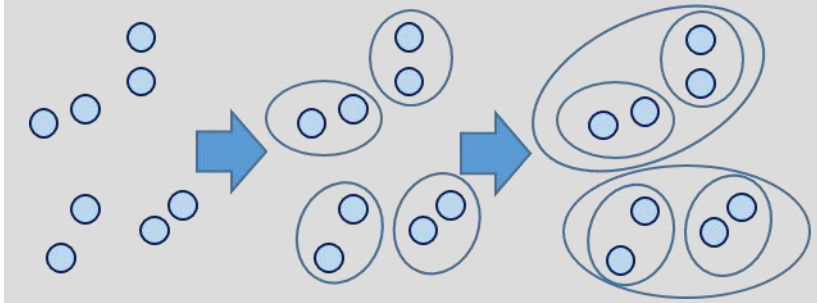
華中科技大學
计算机科学与技术学院
School of Computer Science &Technology，HUST



Performance of Various Methods

華中科技大學
计算机科学与技术学院
School of Computer Science &Technology，HUST

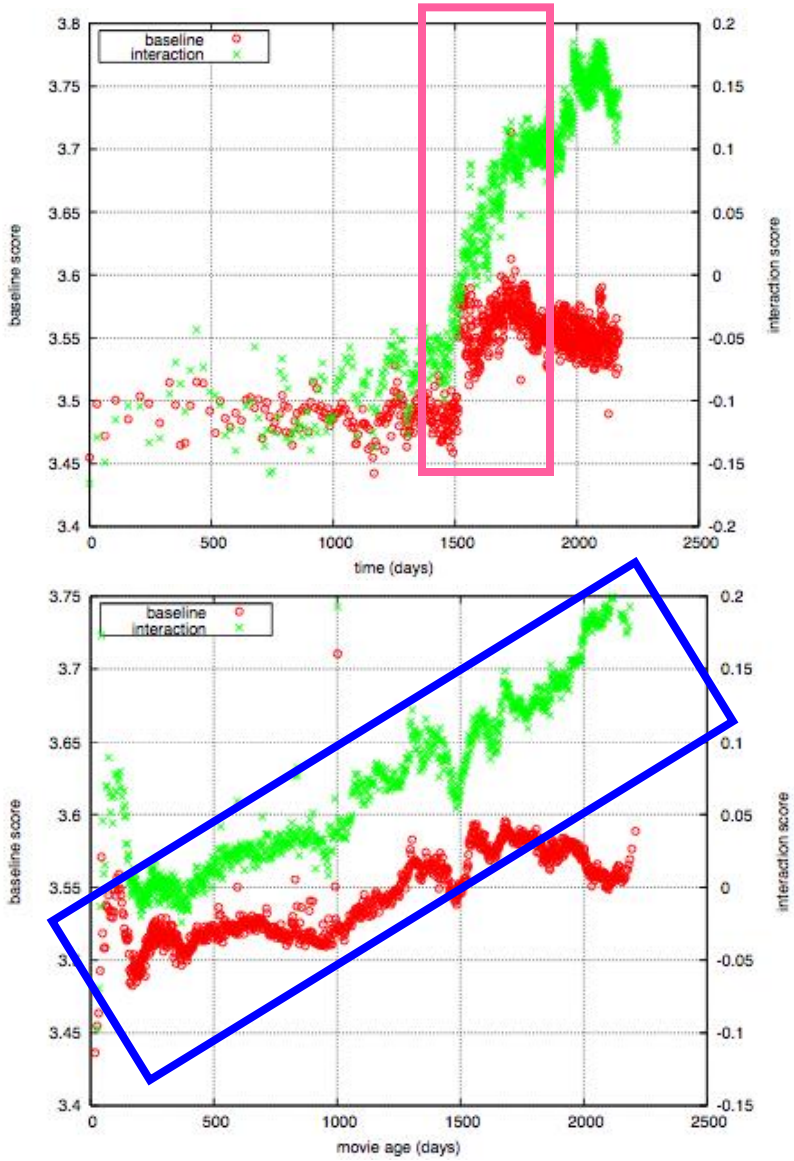




Add Temporal Biases

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:**

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

b_x = bias of user x
b_i = bias of movie i

- **Add time dependence to biases:**

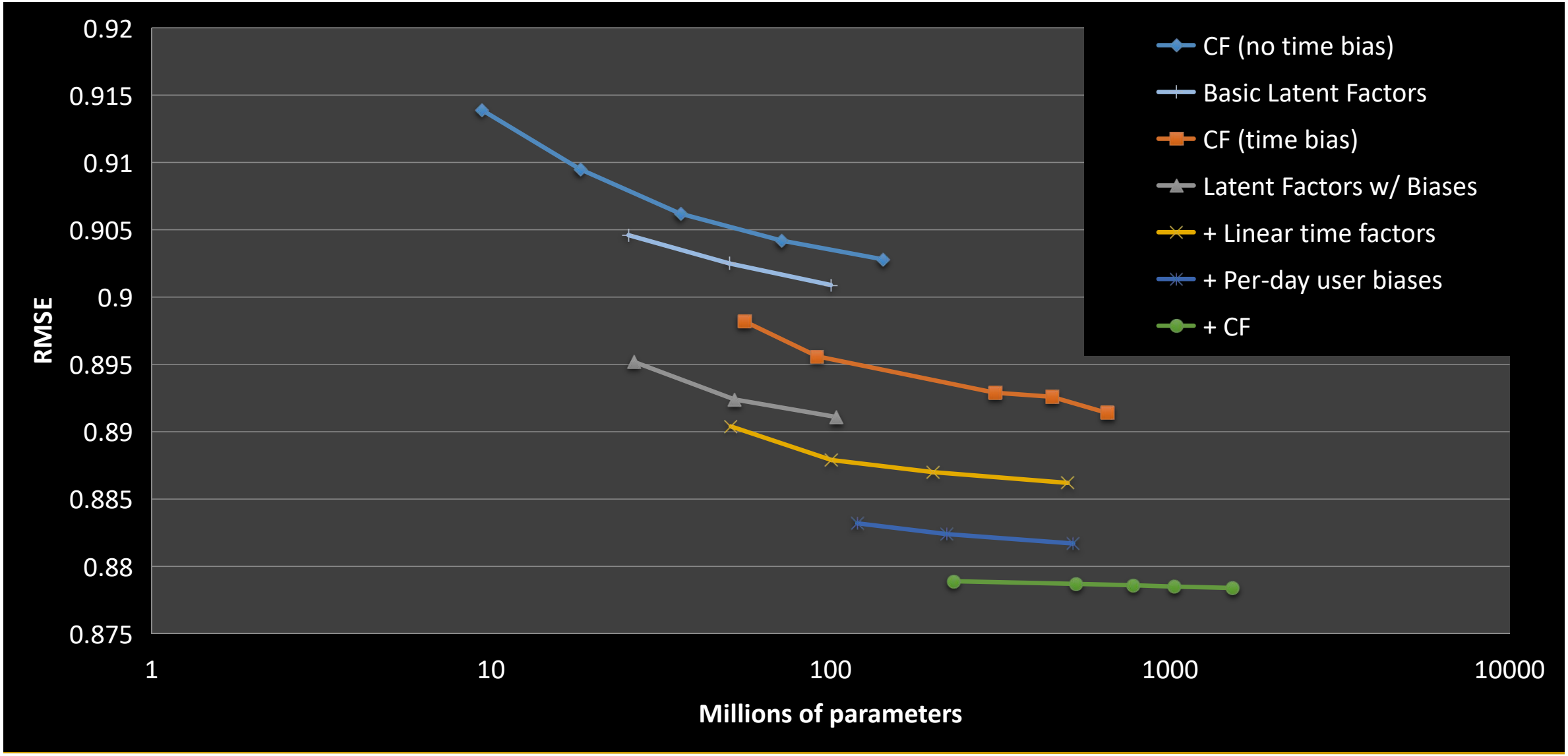
$$r_{xi} = \mu + b_x(t) + b_i(t) + q_i \cdot p_x$$

  - 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

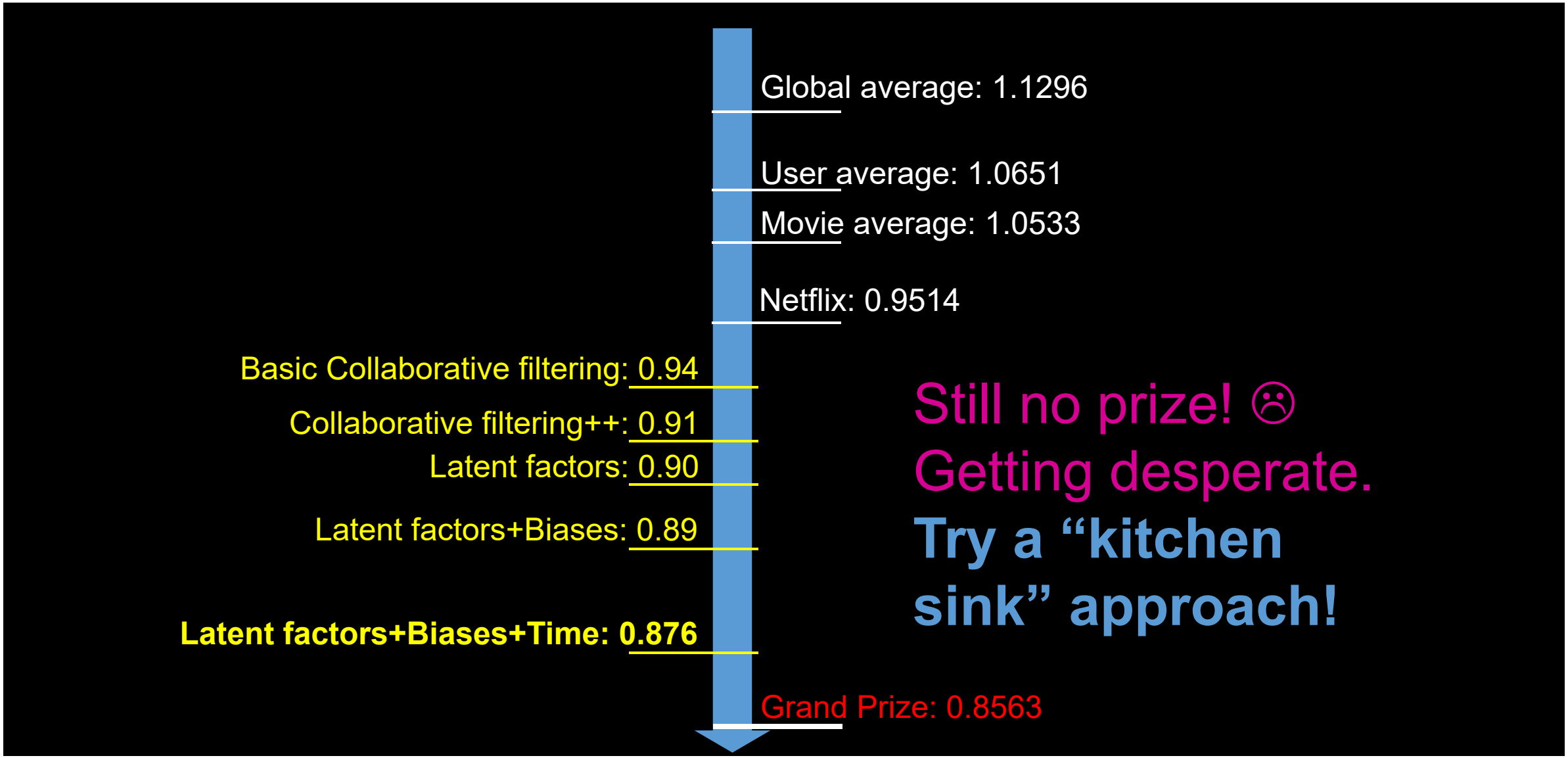
$$b_i(t) = b_i + b_{i,\mathrm{Bin}(t)}$$

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

Adding Temporal Effects

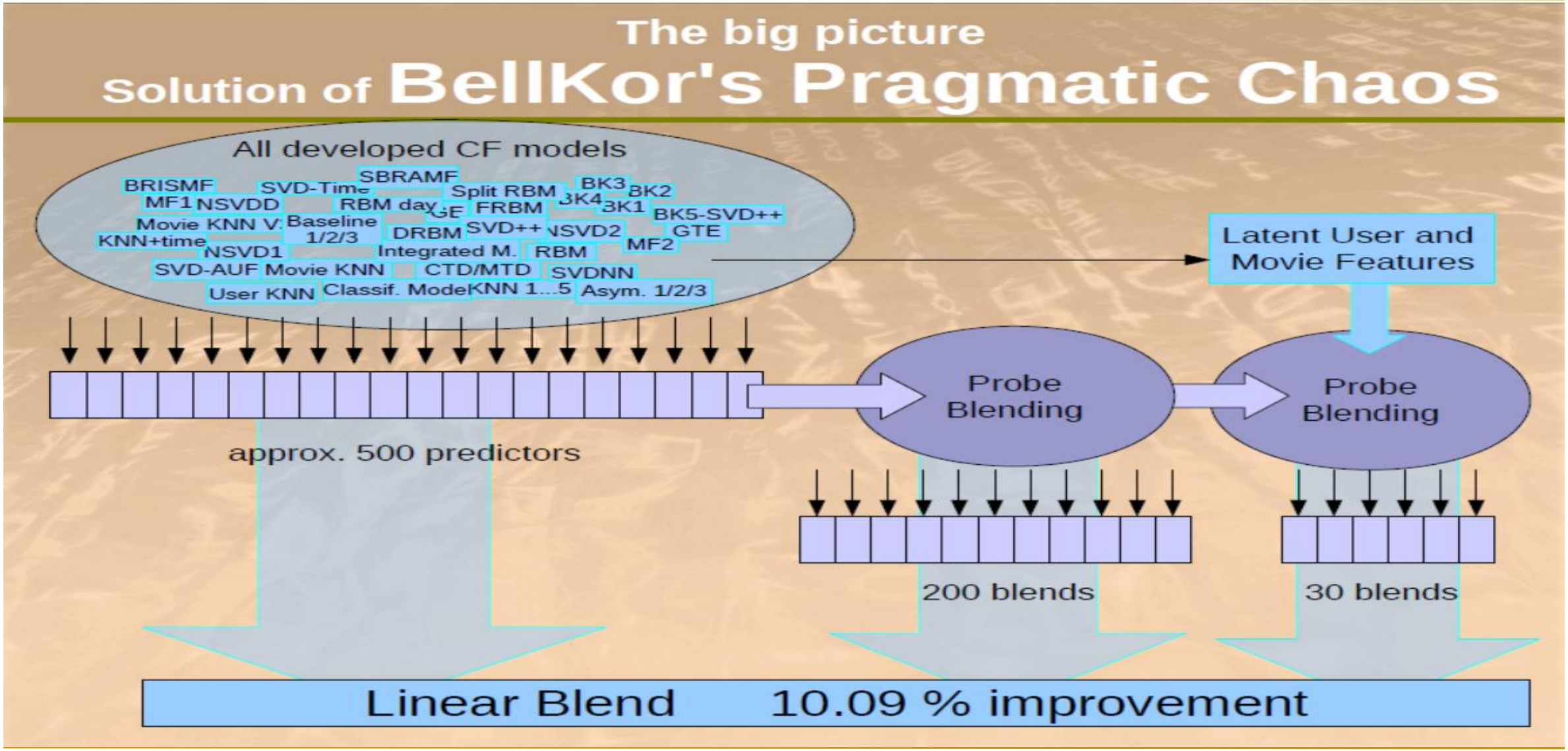


Performance of Various Methods



Performance of Various Methods

華中科技大學
计算机科学与技术学院
School of Computer Science &Technology，HUST



Standing on June 26th 2009

NETFLIX

Netflix Prize

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

Leaderboard

Display top 20 leaders.

Rank	Team Name	Best Score	% Improvement	Last Submit Time
1	BellKor's Pragmatic Chaos	0.8558	10.05	2009-06-26 18:42:37
Grand Prize - RMSE <= 0.8563				
2	PragmaticTheory	0.8582	9.80	2009-06-25 22:15:51
3	BellKor in BigChaos	0.8590	9.71	2009-05-13 08:14:09
4	Grand Prize Team	0.8593	9.68	2009-06-12 08:20:24
5	Dace	0.8604	9.56	2009-04-22 05:57:03
6	BigChaos	0.8613	9.47	2009-06-23 23:06:52
Progress Prize 2008 - RMSE = 0.8616 - Winning Team: BellKor in BigChaos				
7	BellKor	0.8620	9.40	2009-06-24 07:16:02
8	Gravity	0.8634	9.25	2009-04-22 18:31:32
9	Opera Solutions	0.8638	9.21	2009-06-26 23:18:13
10	BruceDengDaoCiYiYou	0.8638	9.21	2009-06-27 00:55:55
11	pengpengzhou	0.8638	9.21	2009-06-27 01:06:43
12	xlvector	0.8639	9.20	2009-06-26 13:49:04
13	xiangliang	0.8639	9.20	2009-06-26 07:47:34
14	Feeds2	0.8641	9.18	2009-06-26 22:51:55

BellKor's Pragmatic Chaos

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….**

NETFLIX

Netflix Prize

COMPLETED

Home | Rules | Leaderboard | Update | Download

Leaderboard

Showing Test Score. Click here to show quiz score

Display top 20 leaders.

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	[illegible]	[illegible]	[illegible]
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11
Progress Prize 2008 - RMSE = 0.8627 - Winning Team: BellKor in BigChaos				
13	xiangliang	0.8642	9.27	2009-07-15 14:53:22
14	Gravity	0.8643	9.26	2009-04-22 18:31:32
15	Ces	0.8651	9.18	2009-06-21 19:24:53
16	Invisible Ideas	0.8653	9.15	2009-07-15 15:53:04
17	Just a guy in a garage	0.8662	9.06	2009-05-24 10:02:54
18	J Dennis Su	0.8666	9.02	2009-03-07 17:16:17
19	Craig Carmichael	0.8666	9.02	2009-07-25 16:00:54
20	acmehill	0.8668	9.00	2009-03-21 16:20:50
Progress Prize 2007 - RMSE = 0.8723 - Winning Team: KorBell				

Million $ Awarded Sept 21st 2009

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/

Chapter 6 总结

- **Three Key Problems for Recommender systems:**
- **(1) Gathering “known” ratings for matrix**
- **(2) Extrapolate unknown ratings from the known ones**
  - **1)** Content-based
  - **2)** Collaborative-based
  - **3)** Latent factor based
- **(3) Evaluating extrapolation methods**