CSCE 670 - Information Storage and Retrieval

Lecture 12: Recommender Systems (Matrix Factorization)

Yu Zhang

yuzhang@tamu.edu

October 2, 2025

Course Website: https://yuzhang-teaching.github.io/CSCE670-F25.html

Adapted from the slides by Prof. Jure Leskovec (Stanford)

Recap: (Item-Item) Collaborative Filtering

- Some Users have rated some Items (e.g., CDs, movies).
- Derive unknown User-Item ratings from those of "similar" Items

- Step 1: For item i, find other similar Items $\mathcal{N}$ (e.g., using the Pearson Correlation Coefficient)
- Step 2: Estimate rating for item i

$$U_{xi} = \frac{\sum_{j \in \mathcal{N}} sim(i,j) \cdot U_{xj}}{\sum_{j \in \mathcal{N}} sim(i,j)}$$

$sim(i,j)$: Pearson Correlation Coefficient between item i and item j

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: RMSE
    - $\sqrt{\frac{1}{M}\sum_{x,i}\left(U_{xi} - U_{xi}^{*}\right)^2}$ where U_{xi} is predicted, and U_{xi}^{*} is the true rating of x on i; M is the number of testing samples
    - Netflix's system RMSE: 0.9514
- Competition
  - 2,700+ teams
  - $1 million prize for 10% improvement on Netflix

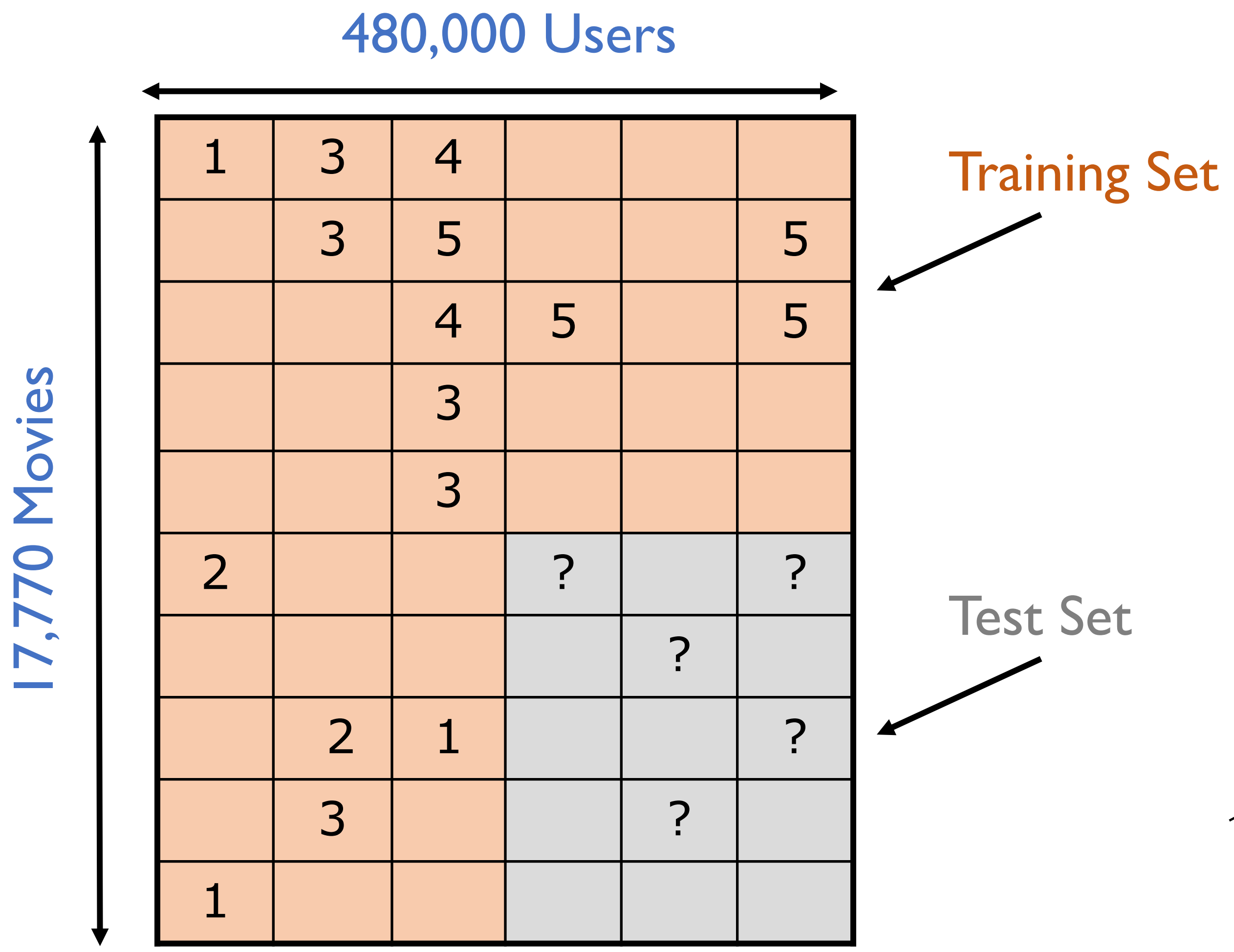
Recap: RMSE

480,000 Users

17,770 Movies

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

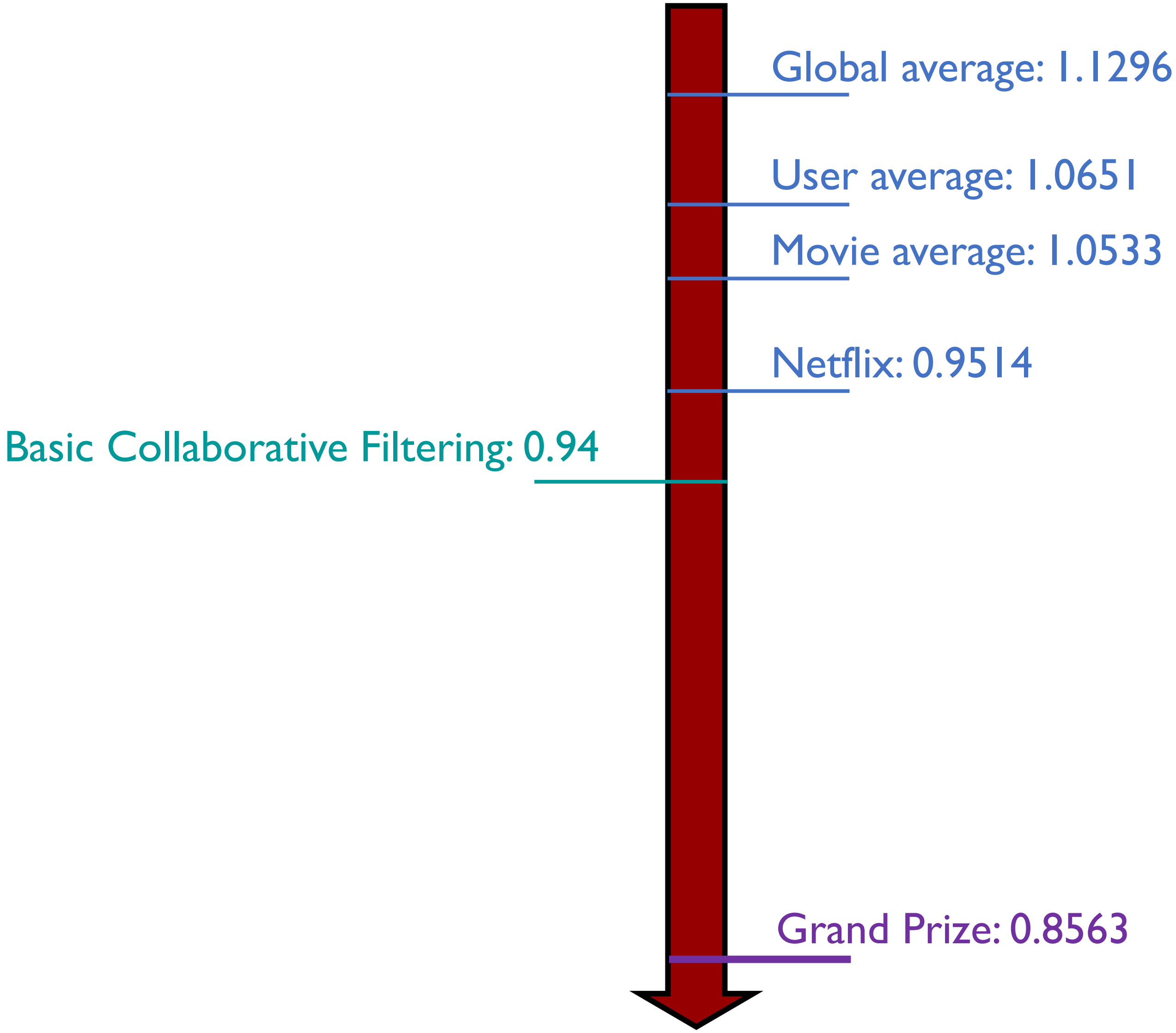
Recap: RMSE



RMSE:

$$\sqrt{\frac{1}{M}\sum_{x,i}\left(U_{xi} - U^*_{xi}\right)^2}$$

Performance of Various Models



Recap: Modeling Deviations

- Basic Collaborative Filtering:

$$U_{xi} = \frac{\sum_{j \in \mathcal{N}} sim(i,j) \cdot U_{xj}}{\sum_{j \in \mathcal{N}} sim(i,j)}$$

- In practice,

$$U_{xi} = b_{xi} + \frac{\sum_{j \in \mathcal{N}} sim(i,j) \cdot (U_{xj} - b_{xj})}{\sum_{j \in \mathcal{N}} sim(i,j)}$$

  - b_{xi}: baseline estimate for U_{xi} $\quad (b_{xi} = \mu + b_x + b_i)$
  - μ: overall mean movie rating
  - b_x: rating deviation of user x, which is the (avg. rating given by user x) – μ
  - b_i: rating deviation of item i, which is the (avg. rating given to item i) – μ

One Step Further: Learning the Weight

$$U_{xi} = b_{xi} + \sum_{j \in \mathcal{N}} w_{ij} \, (U_{xj} - b_{xj})$$

- w_{ij} is learned from training data
  - We allow $\sum_{j \in \mathcal{N}} w_{ij} \neq 1$.
- w_{ij} models the interaction between pairs of movies.
  - It does not depend on user x.

- What is the objective?
  - RMSE! $\sqrt{\frac{1}{M} \sum_{x,i} \left(U_{xi} - U_{xi}^*\right)^2}$
  - Or equivalently: $\sum_{x,i} (U_{xi} - U_{xi}^*)^2$

Recommendations via Optimization

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

- How to find the values of w_{ij}?
  - Gradient descent!

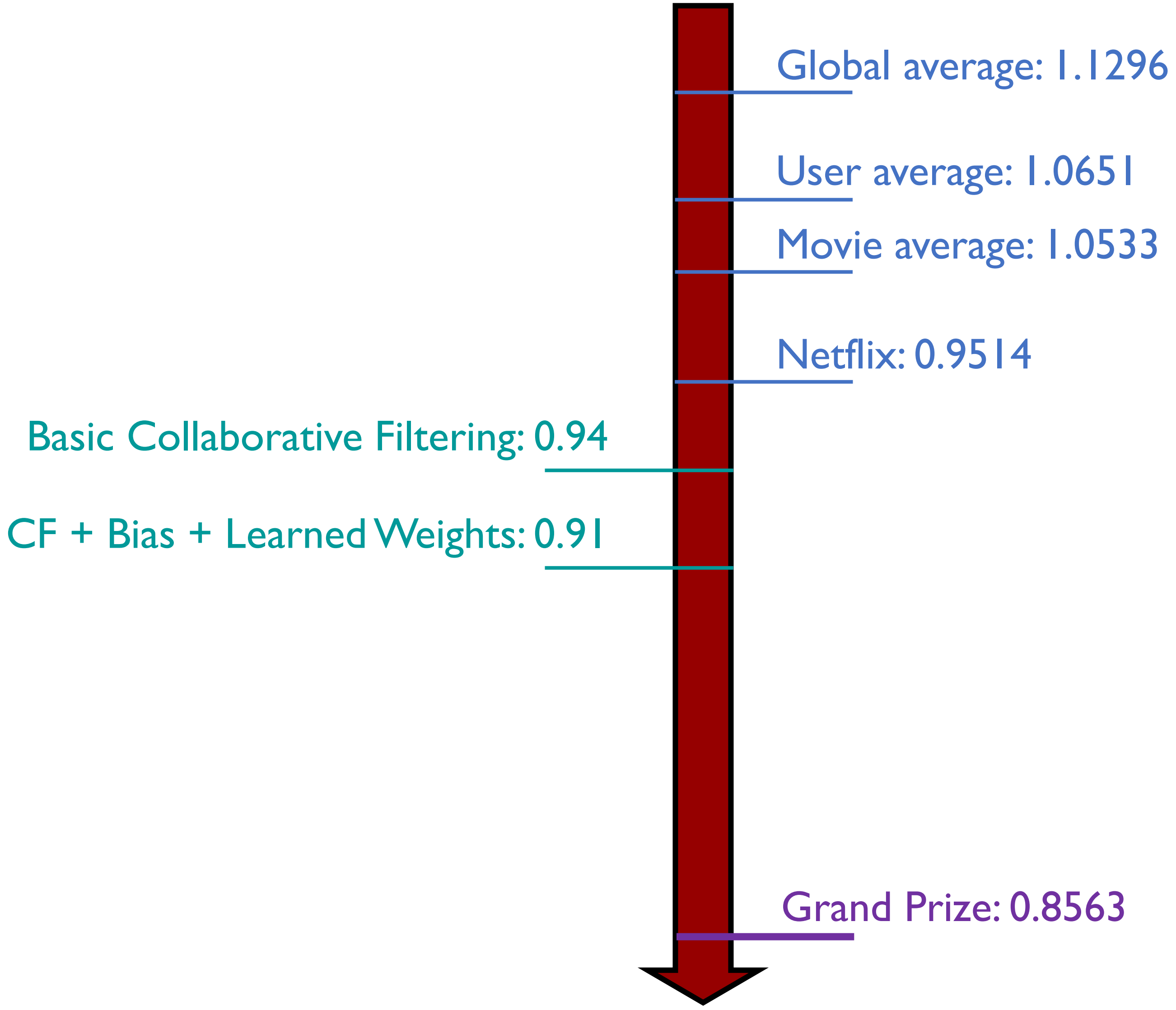
$$\frac{\partial J}{\partial w_{ij}} = 2 \sum_{x,i} \left(\left[b_{xi} + \sum_{j \in \mathcal{N}} w_{ij} \left(U_{xj} - b_{xj} \right) \right] - U_{xi}^* \right) (U_{xj} - b_{xj})$$

(for $j \in \mathcal{N}$)

$$\frac{\partial J}{\partial w_{ij}} = 0$$

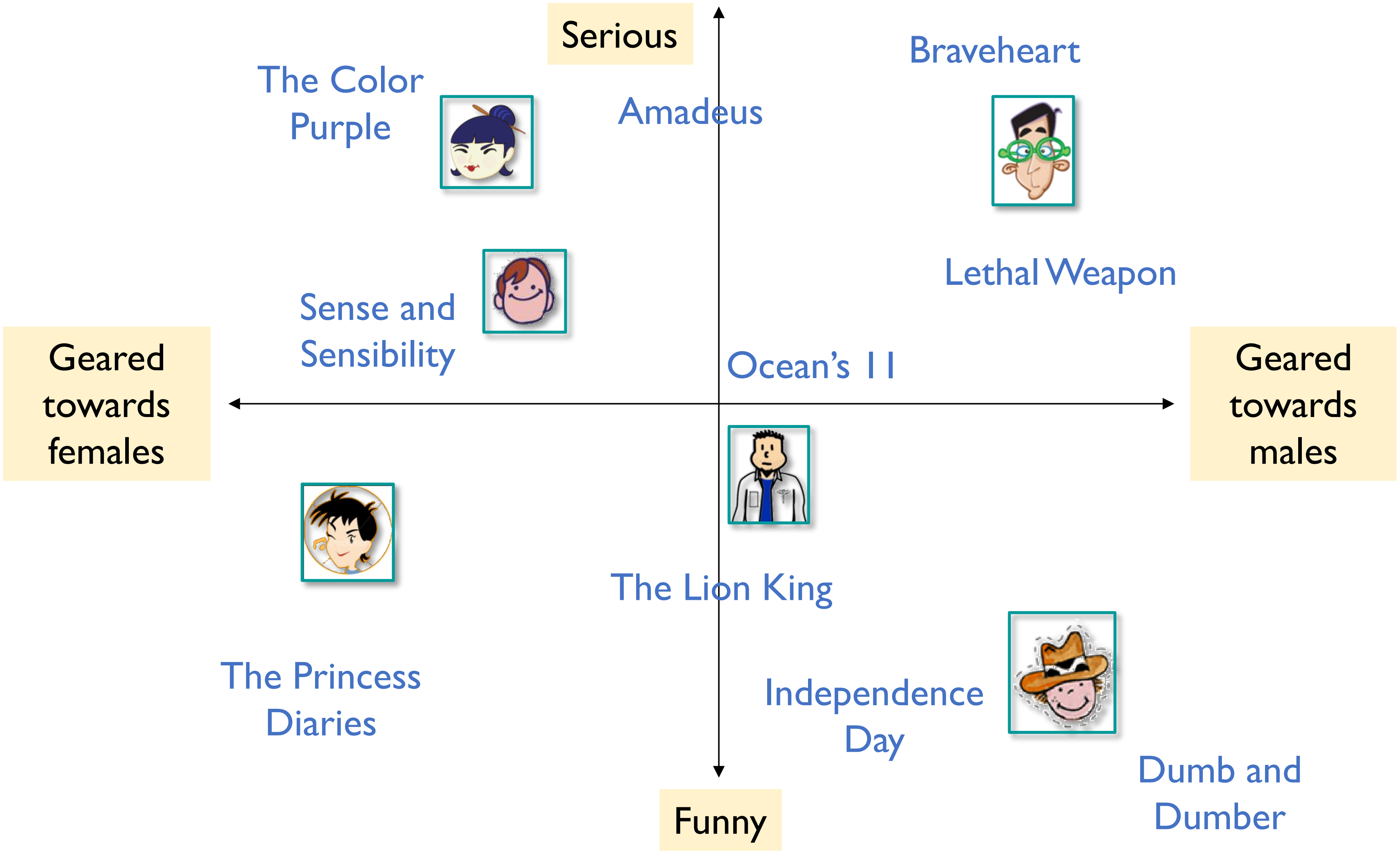
(for $j \notin \mathcal{N}$)

Performance of Various Models



Latent-Factor Models
(Matrix Factorization)

There are certain latent factors that influence users' ratings.



Latent-Factor Models

- $U \approx QP^T$

The number of factors is small.
In other words, Q and P are "thin".

users (columns) × **items** (rows):

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	

U

$\approx$

factors (columns) × **items** (rows):

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

Q

$\times$

users (columns) × **factors** (rows):

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

- For now, let's assume this is mathematically doable.
  - U has missing entries but let's first ignore that!
  - 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.

How to interpret $\boldsymbol{Q}$ and $\boldsymbol{P}$?

$\boldsymbol{Q}$ (rows: items; columns: factors)

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

$\times$

$\boldsymbol{P}^T$ (rows: factors; columns: 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

- Let's assume that the first factor is *the level of seriousness*.

How to interpret $\boldsymbol{Q}$ and $\boldsymbol{P}$?

factors

items

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

$\boldsymbol{Q}$

$\times$

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

$\boldsymbol{P}^T$

- Let's assume that the first factor is *the level of seriousness*.
  - The seriousness of User 1 is 1.1
  - The seriousness of Movie 4 is 1.1
  - So, User 1 may like Movie 4

How to interpret $\boldsymbol{Q}$ and $\boldsymbol{P}$?

factors

items

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

$\boldsymbol{Q}$

$\times$

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

$\boldsymbol{P}^T$

- Let's assume that the first factor is *the level of seriousness*.
  - The seriousness of User 1 is 1.1
  - The seriousness of Movie 5 is -0.7
  - So, User 1 may NOT like Movie 5

Of course, we need to consider all the factors.

- Ratings as "sum of products"

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	

$\boldsymbol{U}$

$\approx$

factors / items

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

$\boldsymbol{Q}$

$\times$

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

$\boldsymbol{P}^T$

$$U_{xi} = \sum_{\phi:\text{ all factors}} Q_{i\phi} \cdot P_{x\phi}$$

Estimating the Missing Rating

- Ratings as "sum of products"

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	

$\boldsymbol{U}$

$\approx$

factors

items

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

$\boldsymbol{Q}$

$\times$

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

$\boldsymbol{P}^T$

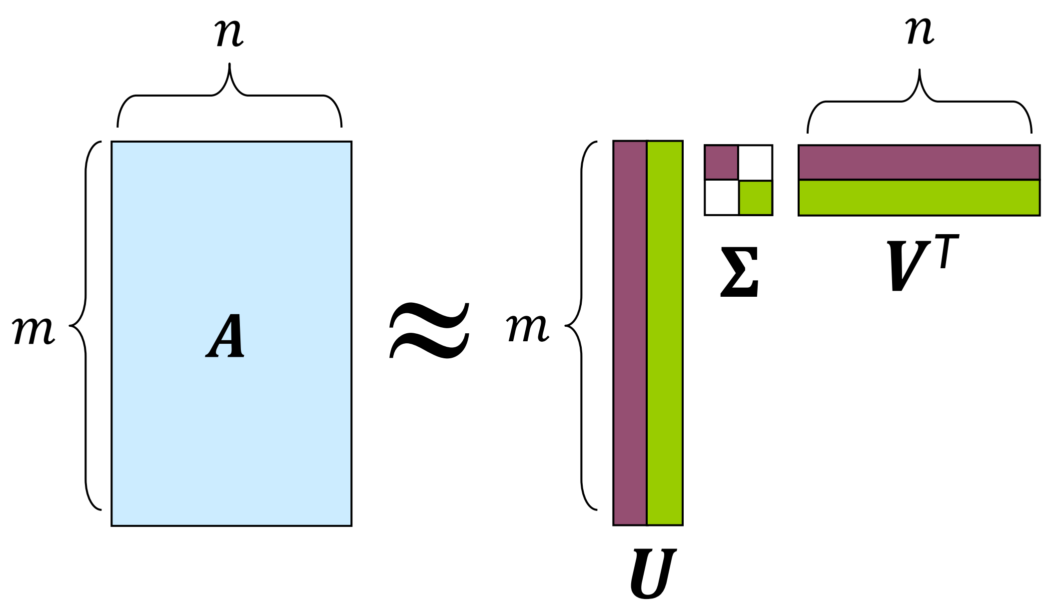
$$U_{xi} = \sum_{\phi:\text{ all factors}} Q_{i\phi} \cdot P_{x\phi} = (-0.5) \times (-2) + 0.6 \times 0.3 + 0.5 \times 2.4 = 2.38$$

How to find $\boldsymbol{Q}$ and $\boldsymbol{P}$?

Singular Value Decomposition (SVD)

$$\boldsymbol{A} \approx \boldsymbol{U\Sigma V}^T$$

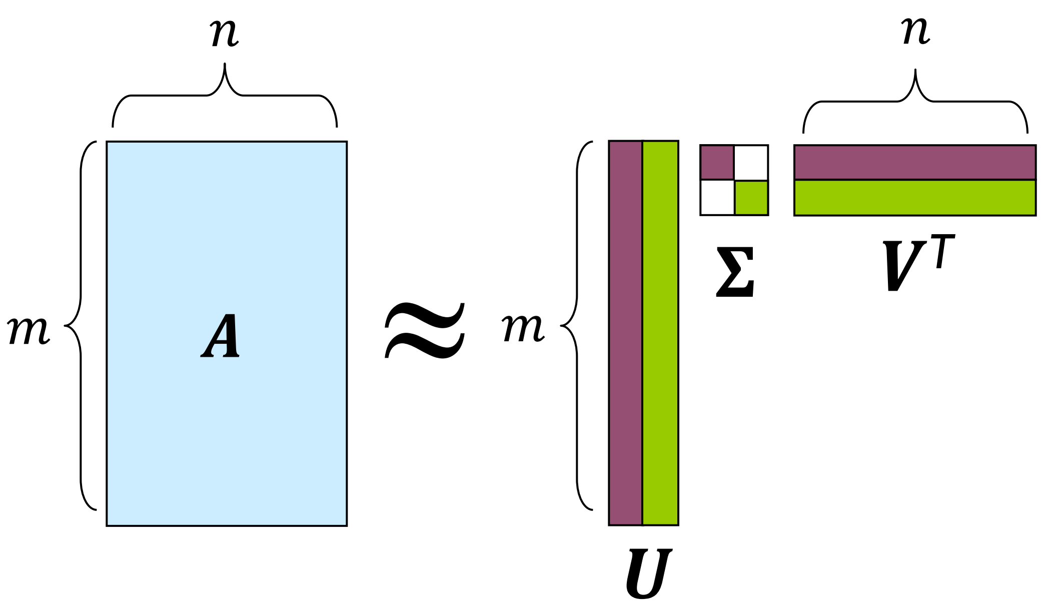
- Input matrix: $\boldsymbol{A}$
- Step 1: Compute $\boldsymbol{A}^T\boldsymbol{A}$
- Step 2: Find the eigenvalues of eigenvectors of $\boldsymbol{A}^T\boldsymbol{A}$
  - Eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$
  - Eigenvectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$
- Step 3: Consider the largest k eigenvalues and their corresponding eigenvectors only. (The choice of k depends on how closely you wish to approximate)
  - $\boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k]$



Singular Value Decomposition (SVD)

$$\boldsymbol{A} \approx \boldsymbol{U\Sigma V}^T$$

- Step 2: Find the eigenvalues of eigenvectors of $\boldsymbol{A}^T\boldsymbol{A}$
  - Eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$
  - Eigenvectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$
- Step 3: Consider the largest k eigenvalues and their corresponding eigenvectors only.
  - $\boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k]$
  - $\boldsymbol{\Sigma} = \mathrm{diag}\{\sqrt{\lambda_1}, \sqrt{\lambda_2}, \ldots, \sqrt{\lambda_k}\}$
- Step 4: $\boldsymbol{U} = \boldsymbol{AV\Sigma}^{-1}$
  - Or you can do Steps 1-3 again for $\boldsymbol{AA}^T$ (rather than $\boldsymbol{A}^T\boldsymbol{A}$) to get $\boldsymbol{U}$



SVD is good, but …

- SVD gives the minimum reconstruction error if we know all entries in $\boldsymbol{A}$.

$$\min_{\boldsymbol{U},\boldsymbol{\Sigma},\boldsymbol{V}} \sum_{i,j} \left(A_{ij} - [\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T]_{ij}\right)^2$$

  - Exactly our objective!
  - Using SVD for our matrix factorization task?

Latent Factor Model	User-Item Matrix: $\boldsymbol{U}$	User-Factor Matrix: $\boldsymbol{Q}$	Factor-Item Matrix: $\boldsymbol{P}^T$
SVD	Input Matrix: $\boldsymbol{A}$	$\boldsymbol{U}$	$\boldsymbol{\Sigma}\boldsymbol{V}^T$

- BUT, our user-item matrix $\boldsymbol{U}$ has missing values!
  - How to interpret missing values? (as 0? a bad idea)
  - Does the property of minimum reconstruction error still hold if there are missing values? (we don't know)

Factorizing a Matrix with Missing Values

$$\min_{\boldsymbol{Q},\boldsymbol{P}} \sum_{(x,i)\text{ known}} (U_{xi} - [\boldsymbol{Q}\boldsymbol{P}^T]_{xi})^2 = \sum_{(x,i)\text{ known}} (U_{xi} - \boldsymbol{q}_i \boldsymbol{p}_x^T)^2$$

- $\boldsymbol{q}_i$ (item vector): the row corresponding to item i in $\boldsymbol{Q}$
- $\boldsymbol{p}_x^T$ (user vector): the column corresponding to user x in $\boldsymbol{P}^T$

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	

$\boldsymbol{U}$

$\approx$

factors

items

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

$\boldsymbol{Q}$

$\times$

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

$\boldsymbol{P}^T$

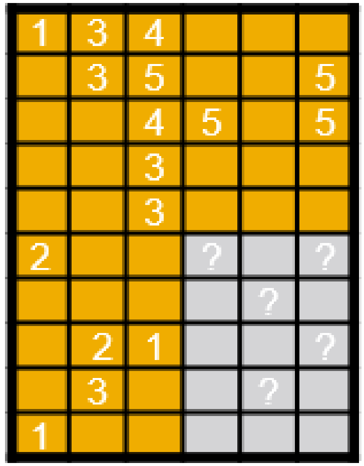
Overfitting

$$\min_{\boldsymbol{Q},\boldsymbol{P}} \sum_{(x,i)\text{ known}} (U_{xi} - \boldsymbol{q}_i \boldsymbol{p}_x^T)^2$$

- $\boldsymbol{q}_i$ (item vector): the row corresponding to item i in $\boldsymbol{Q}$
- $\boldsymbol{p}_x^T$ (user vector): the column corresponding to user x in $\boldsymbol{P}^T$

- No closed form solution.
- All item vectors and user vectors are parameters to be learned!

- Overfitting: With too much freedom (too many free parameters) the model starts fitting noise in the training data, thus not generalizing well to unseen test data.



Regularization

- Model parameters can be "complicated" where there are sufficient training data
- Model parameters should be "simple" where training data are scarce

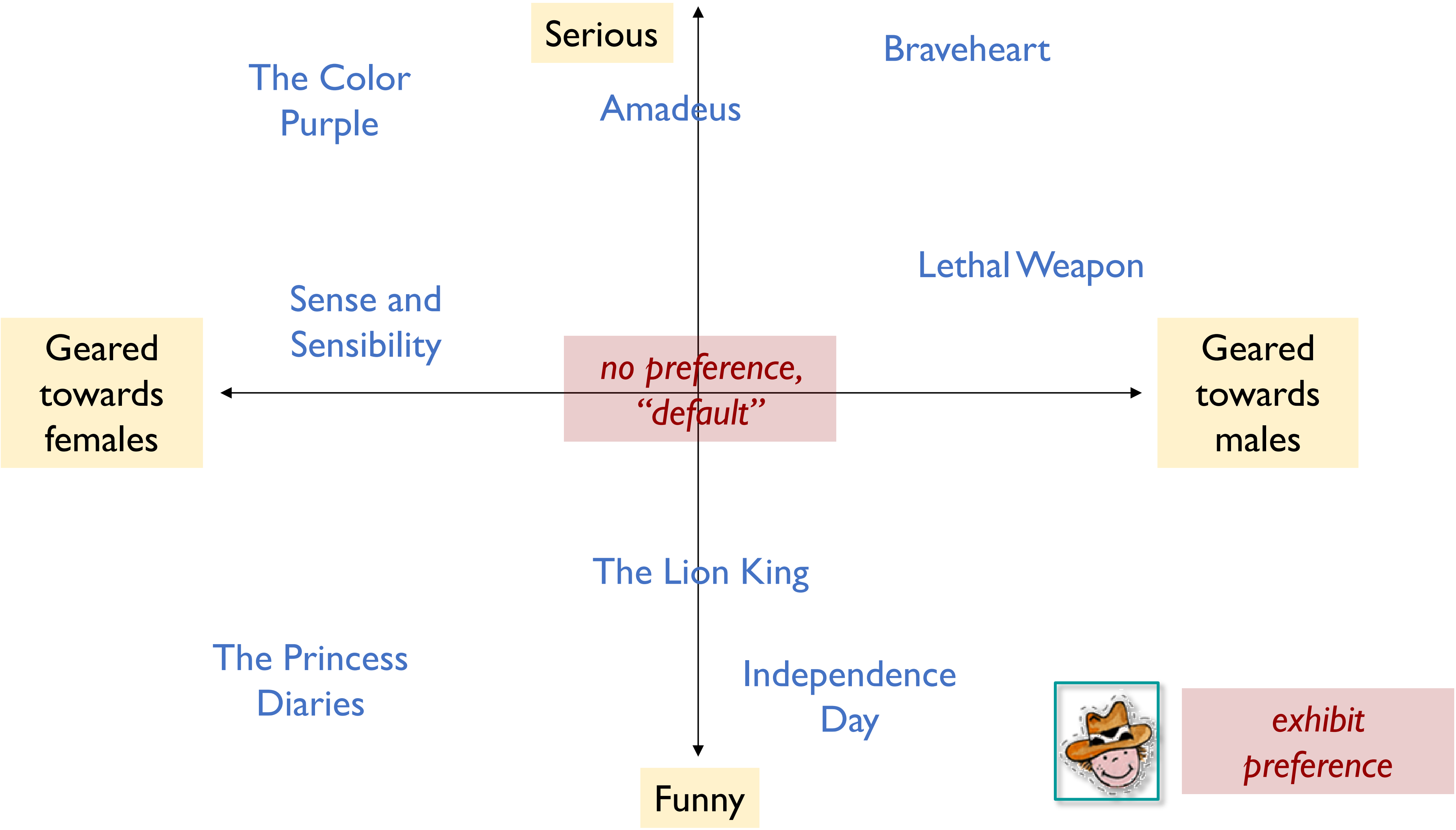
$$\min_{\boldsymbol{Q},\boldsymbol{P}} \sum_{(x,i)\ \text{known}} (U_{xi} - \boldsymbol{q}_i \boldsymbol{p}_x^T)^2 + \left[c_1 \sum_x \|\boldsymbol{p}_x\|^2 + c_2 \sum_i \|\boldsymbol{q}_i\|^2 \right]$$

Original Objective | Regularization Term

(c_1 and c_2 are hyperparameters)

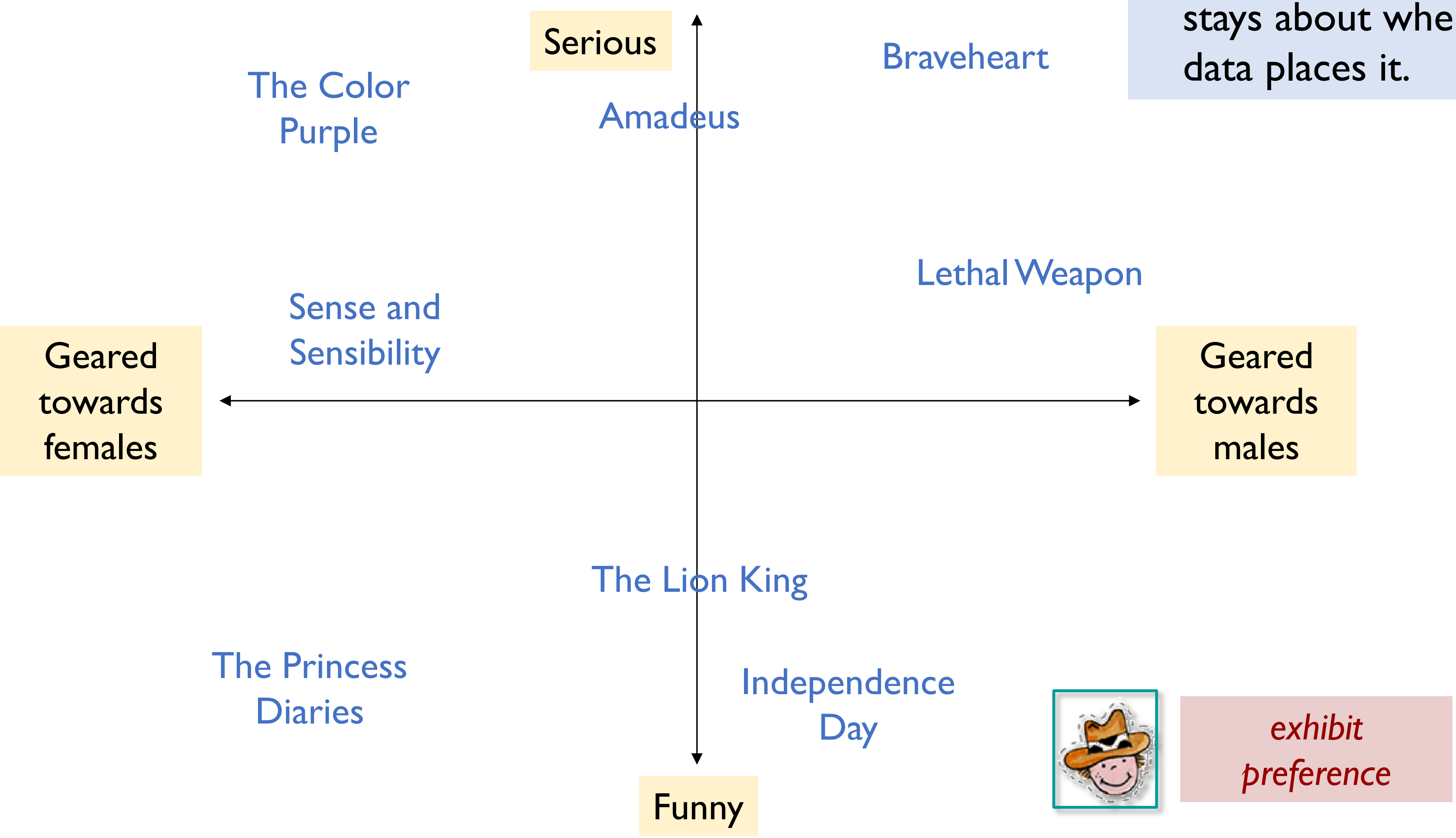
- How to understand the Regularization Term?

The Effect of Regularization



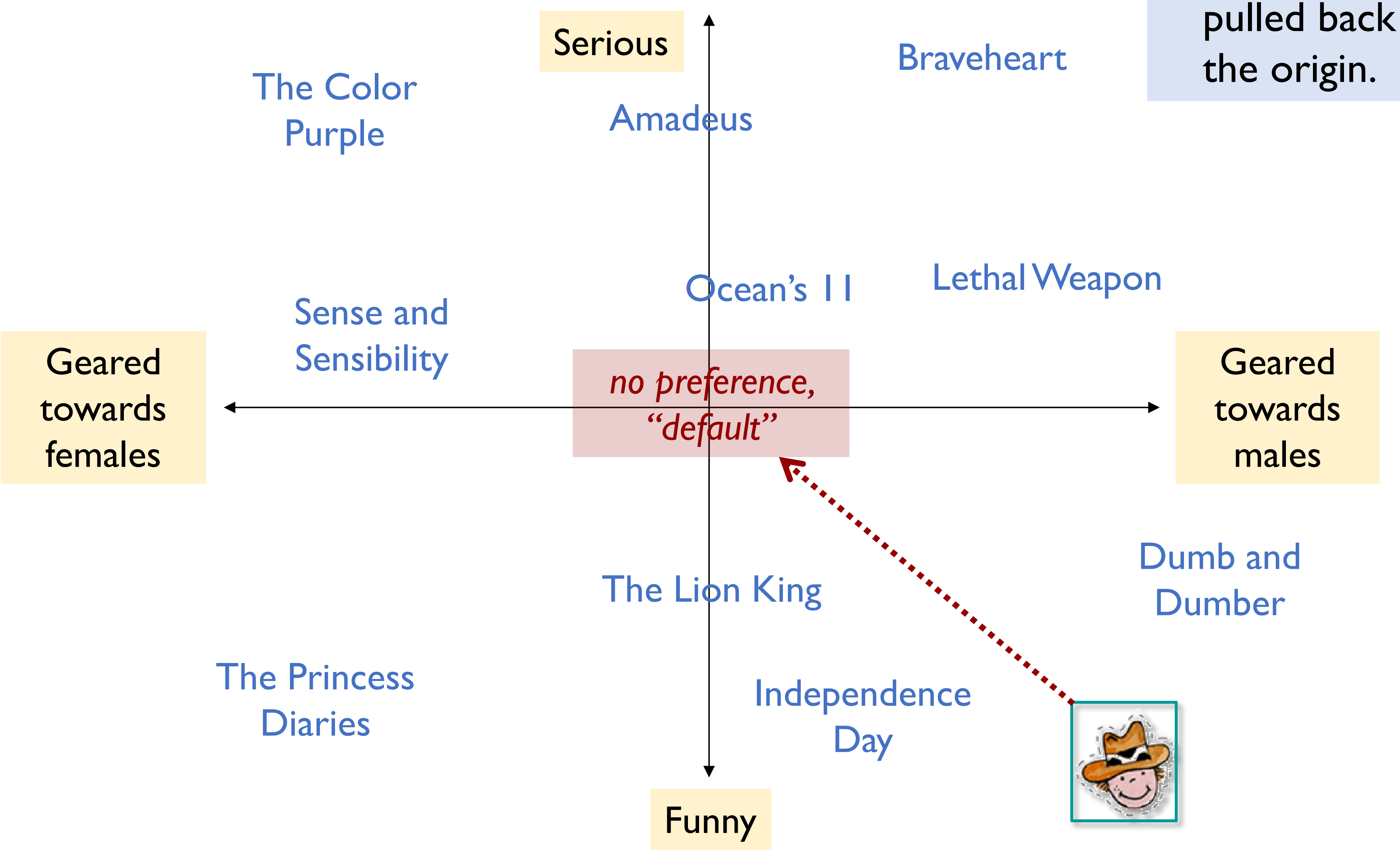
The Effect of Regularization

- If the user has rated hundreds of movies, it stays about where the data places it.



The Effect of Regularization

- If the user has rated only a handful, it is pulled back towards the origin.

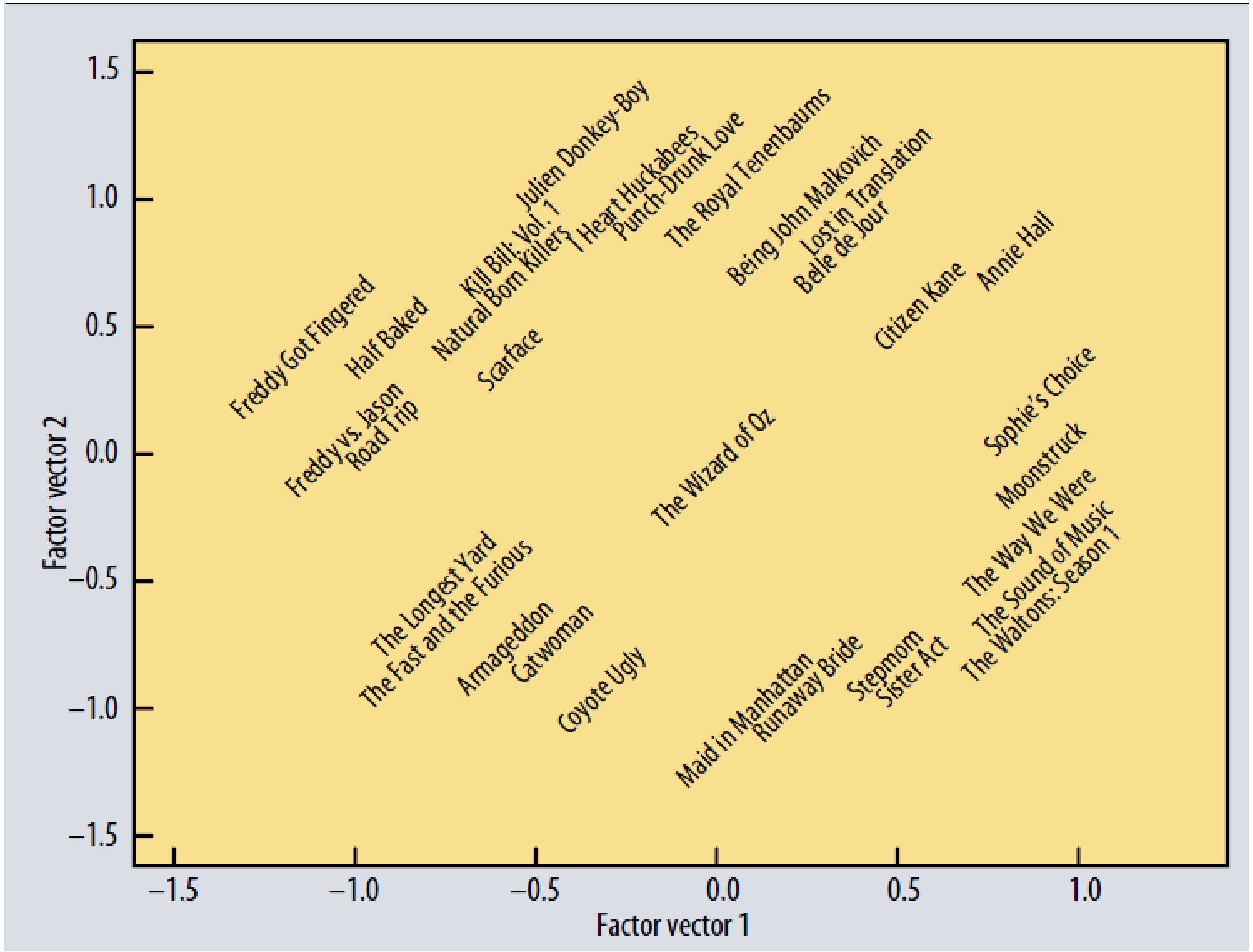


Gradient Descent

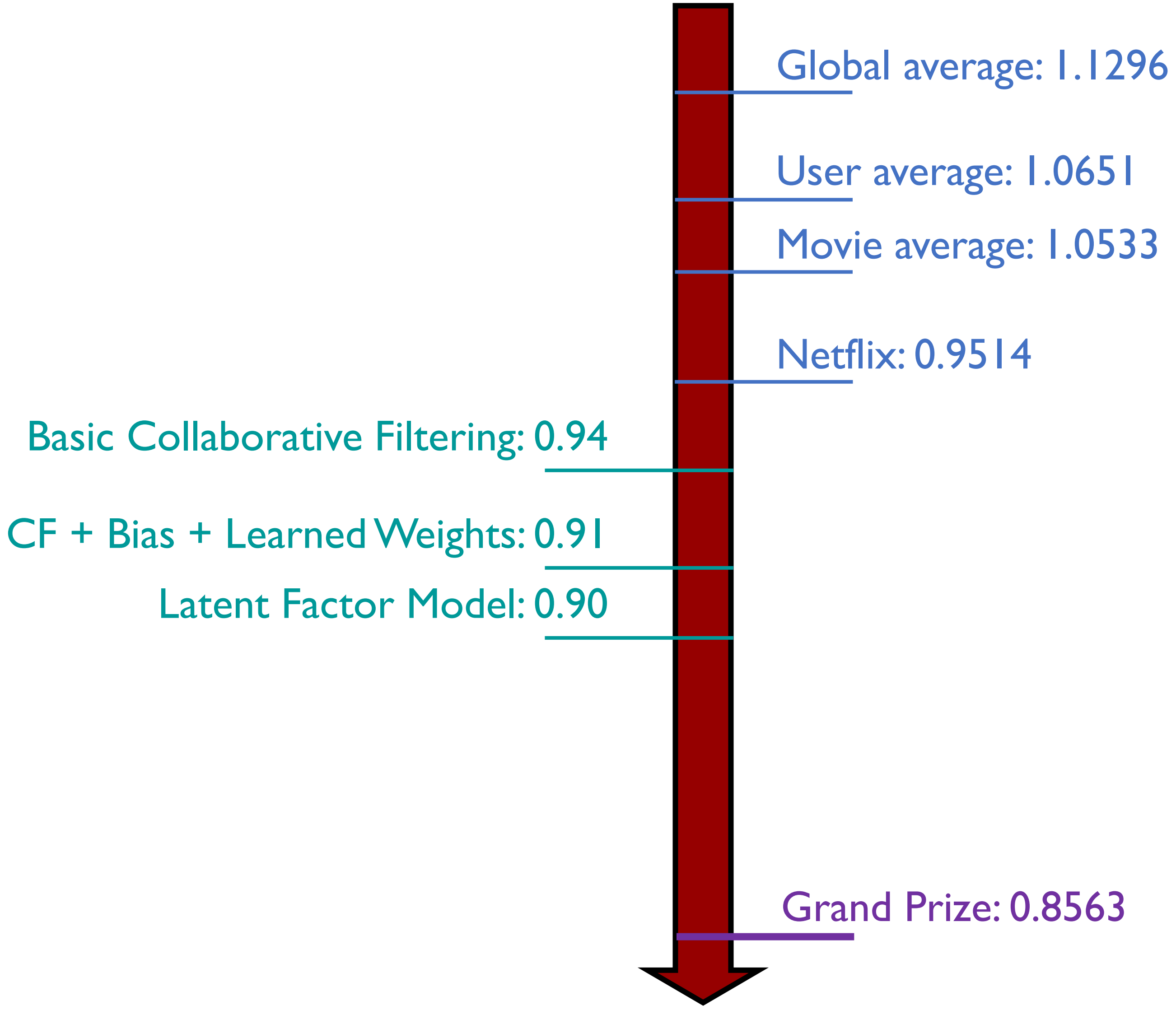
$$\min_{\boldsymbol{Q},\boldsymbol{P}} J = \sum_{(x,i)\text{ known}} (U_{xi} - \boldsymbol{q}_i \boldsymbol{p}_x^T)^2 + \left[c_1 \sum_x \|\boldsymbol{p}_x\|^2 + c_2 \sum_i \|\boldsymbol{q}_i\|^2 \right]$$

- Step 1: Initialize $\boldsymbol{Q}$ and $\boldsymbol{P}$ using SVD (pretend missing ratings are 0)
- Step 2: Gradient descent
  - $P_{x\phi} = P_{x\phi} - \eta \frac{\partial J}{\partial P_{x\phi}}$
    - $\frac{\partial J}{\partial P_{x\phi}} = \sum_{(x,i)\text{ known}} \left(-2(U_{xi} - \boldsymbol{q}_i \boldsymbol{p}_x^T) Q_{i\phi} + 2c_1 P_{x\phi} \right)$
  - $Q_{i\phi} = Q_{i\phi} - \eta \frac{\partial J}{\partial Q_{i\phi}}$
    - $\frac{\partial J}{\partial Q_{i\phi}} = \sum_{(x,i)\text{ known}} \left(-2(U_{xi} - \boldsymbol{q}_i \boldsymbol{p}_x^T) P_{x\phi} + 2c_2 Q_{i\phi} \right)$

Learned Item Vectors in the Latent Factor Space



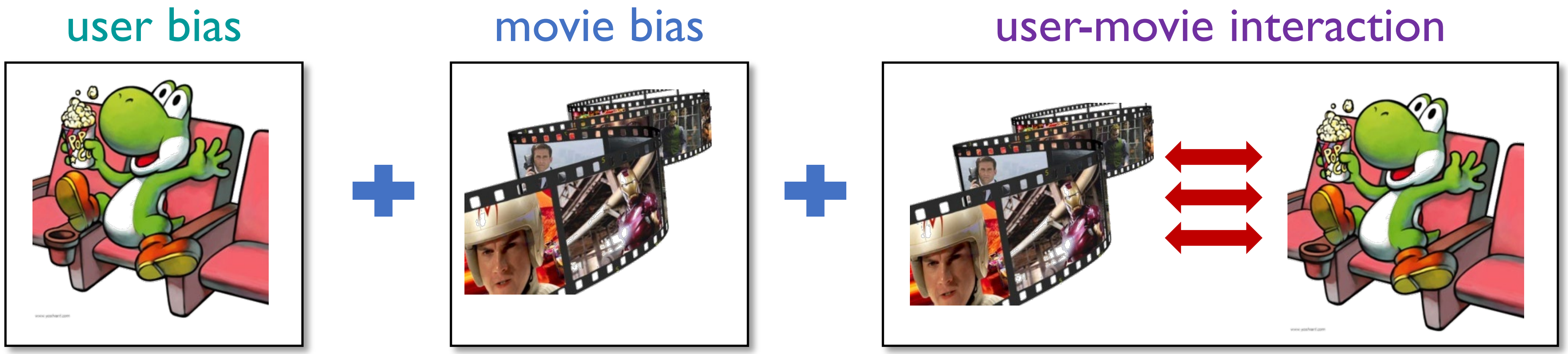
Performance of Various Models



Extending Latent Factor Models to Include Bias

Bias, Again

- Basic Latent Factor Model: $U_{xi} = \boldsymbol{q}_i \boldsymbol{p}_x^T$
- Latent Factor Model with Bias: $U_{xi} = \mu + b_x + b_i + \boldsymbol{q}_i \boldsymbol{p}_x^T$
    - μ: overall mean movie rating
    - b_x: rating deviation of user x
    - b_i: rating deviation of item i



Bias, Again

- Latent Factor Model with Bias:

$$U_{xi} = \mu + b_x + b_i + \boldsymbol{q}_i \boldsymbol{p}_x^T$$

  - μ: overall mean movie rating
    - E.g., $\mu = 2.7$
  - b_x: rating deviation of user x (*to be learned*)
    - E.g., Bob is a critical reviewer. Based on the training data, his rating will be 0.7 star lower than the mean $\Rightarrow b_x = -0.7$.
  - b_i: rating deviation of item i (*to be learned*)
    - E.g., Star Wars will get a mean rating of 0.5 higher than the average $\Rightarrow b_i = 0.5$
  - $\boldsymbol{q}_i$ and $\boldsymbol{p}_x$: vector of user x and item i in the latent factor space (*to be learned*)
    - E.g., based on the genre, Bob likes Star Wars $\Rightarrow \boldsymbol{q}_i \boldsymbol{p}_x^T = 0.3$
  - $U_{xi} = 2.7 - 0.7 + 0.5 + 0.3 = 2.8$

Fitting the New Model

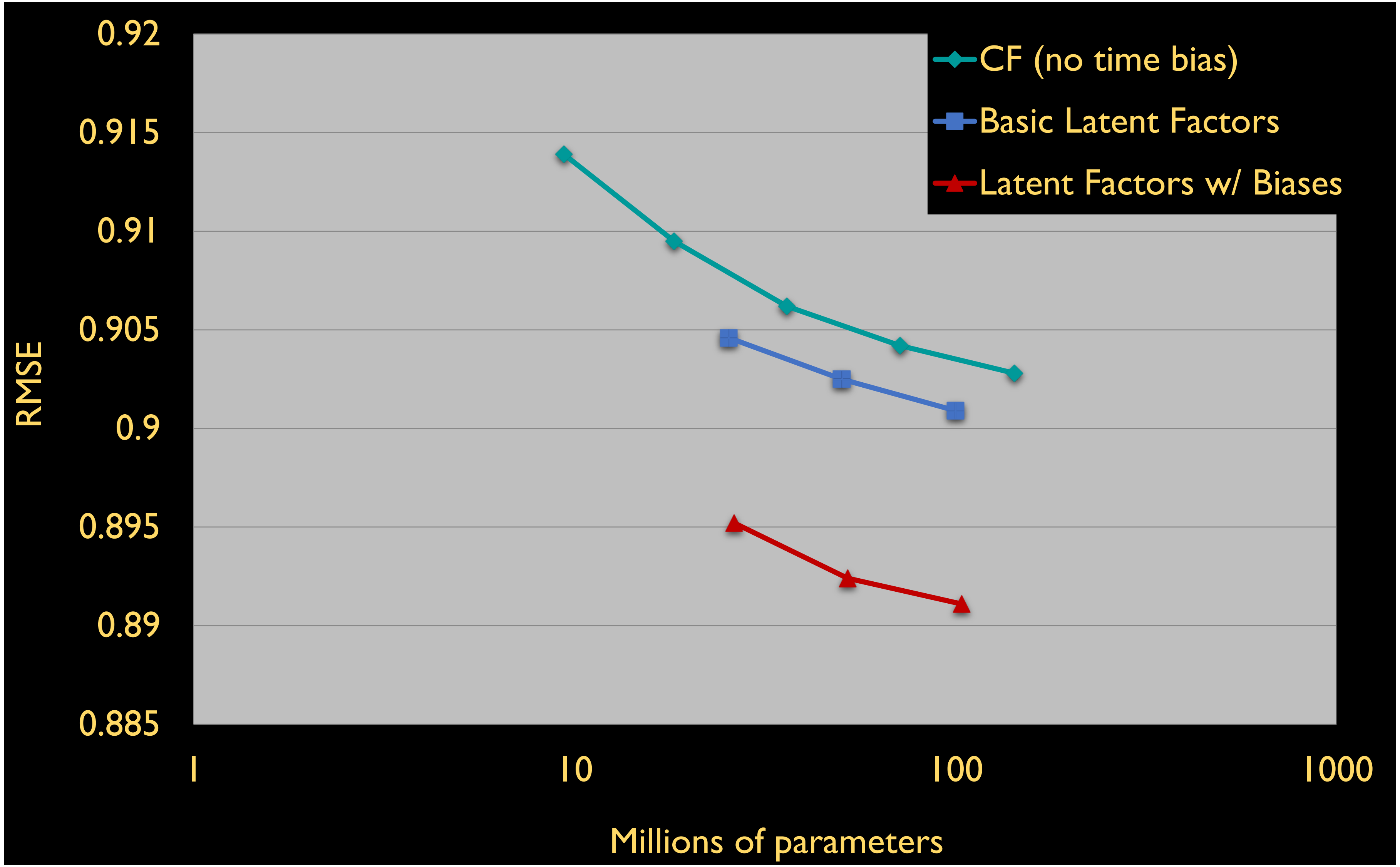
$$\min_{\boldsymbol{Q},\boldsymbol{P},b_x,b_i} J = \sum_{(x,i)\text{ known}} (U_{xi} - (\mu + b_x + b_i + \boldsymbol{q}_i \boldsymbol{p}_x^T))^2$$

$$+ \left[c_1 \sum_x \|\boldsymbol{p}_x\|^2 + c_2 \sum_i \|\boldsymbol{q}_i\|^2 + c_3 \sum_x \|b_x\|^2 + c_4 \sum_i \|b_i\|^2 \right]$$

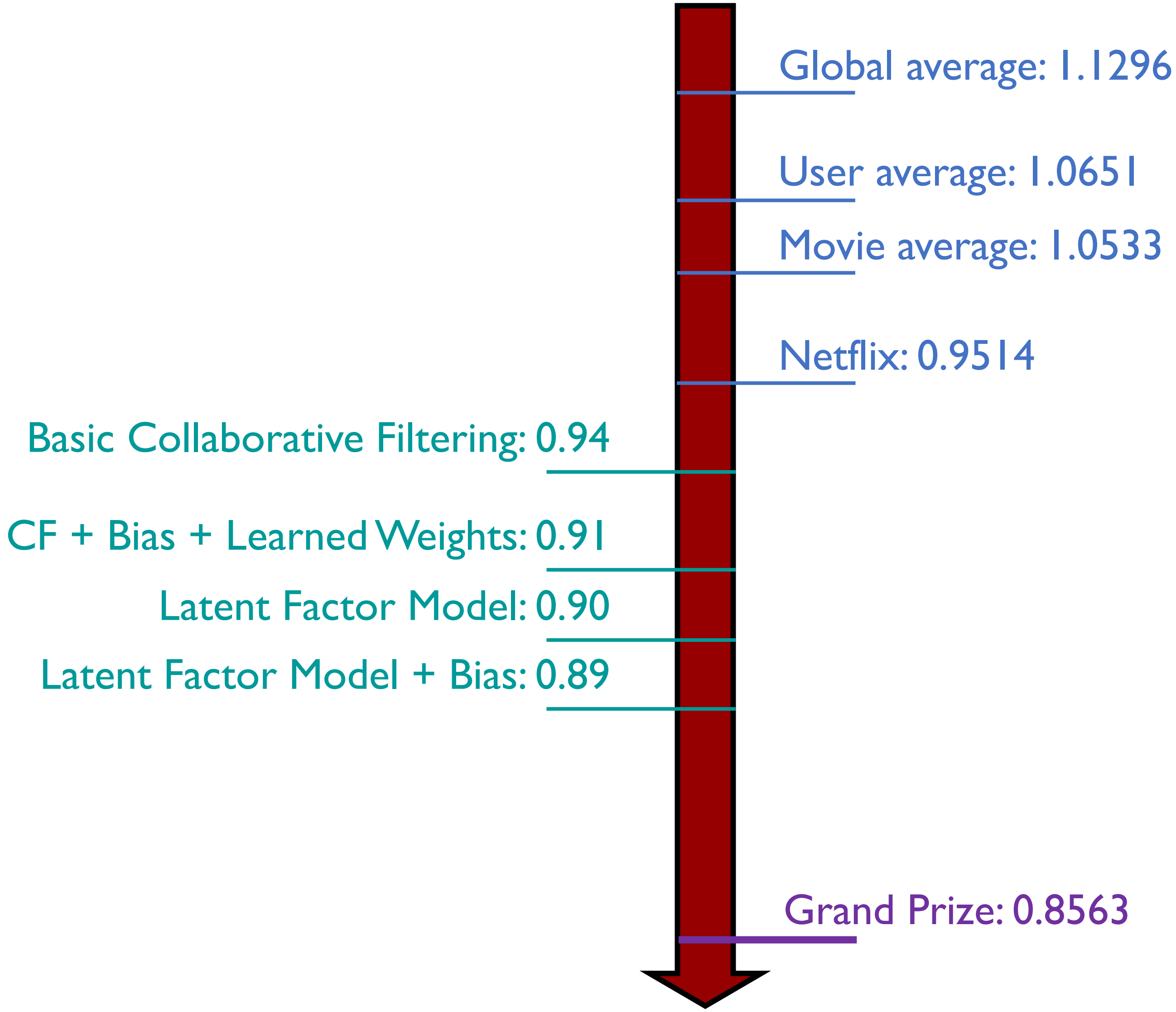
- Both biases b_x, b_i as well as interactions $\boldsymbol{q}_i, \boldsymbol{p}_x$ are treated as parameters to be learned via gradient descent
  - $P_{x\phi} = P_{x\phi} - \eta \frac{\partial J}{\partial P_{x\phi}}, \qquad Q_{i\phi} = Q_{i\phi} - \eta \frac{\partial J}{\partial Q_{i\phi}}$
  - $b_x = b_x - \eta \frac{\partial J}{\partial b_x}, \qquad b_i = b_i - \eta \frac{\partial J}{\partial b_i}$

Performance of Various Models

- Which hyperparameter determines the number of parameters?
  - Number of factors



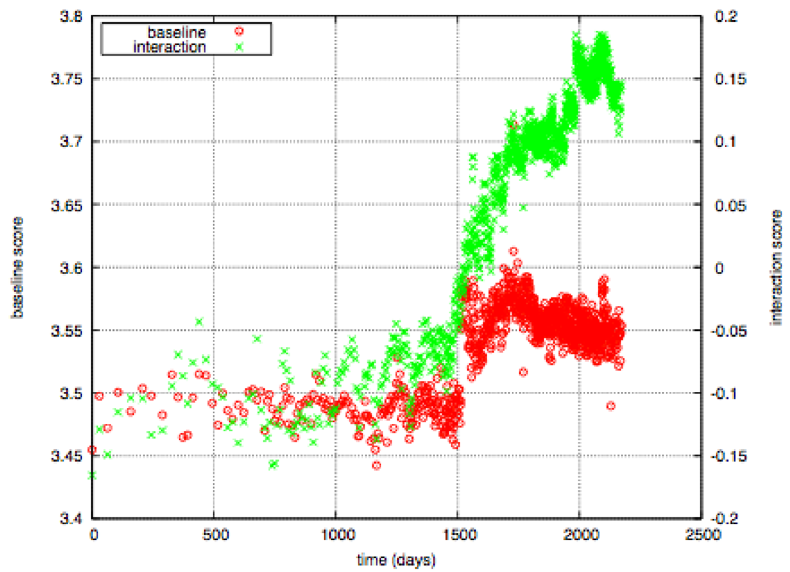
Performance of Various Models

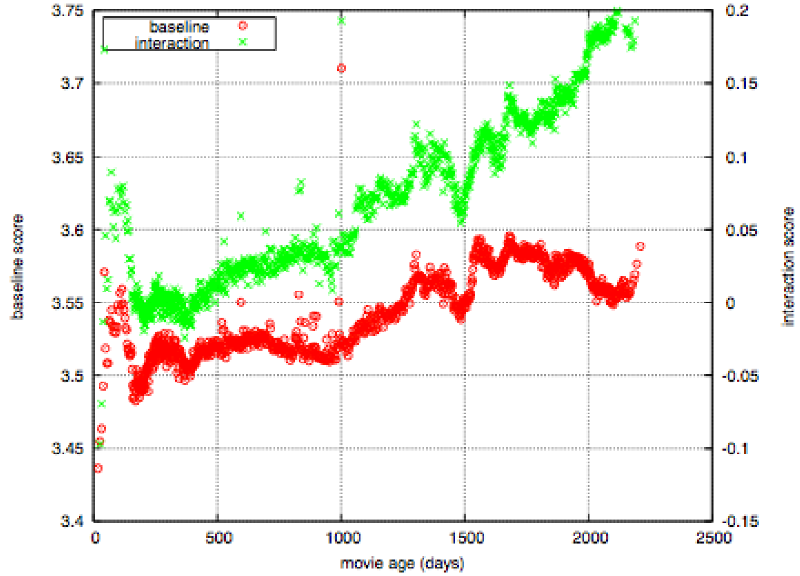


Extended Content: The Netflix Challenge 2006-2009
(will not appear in quizzes or the exam)

Temporal Biases Of Users [Koren, KDD 2009]

- A sudden surge in the average movie rating observed in early 2004.
  - Possible reasons:
    - Improvements in Netflix
    - GUI improvements
    - Meaning of rating changed
- For the rating of a single movie, its age is an important factor.
  - Users prefer the newest movies
  - For not that new movies, people believe even older movies are just inherently better

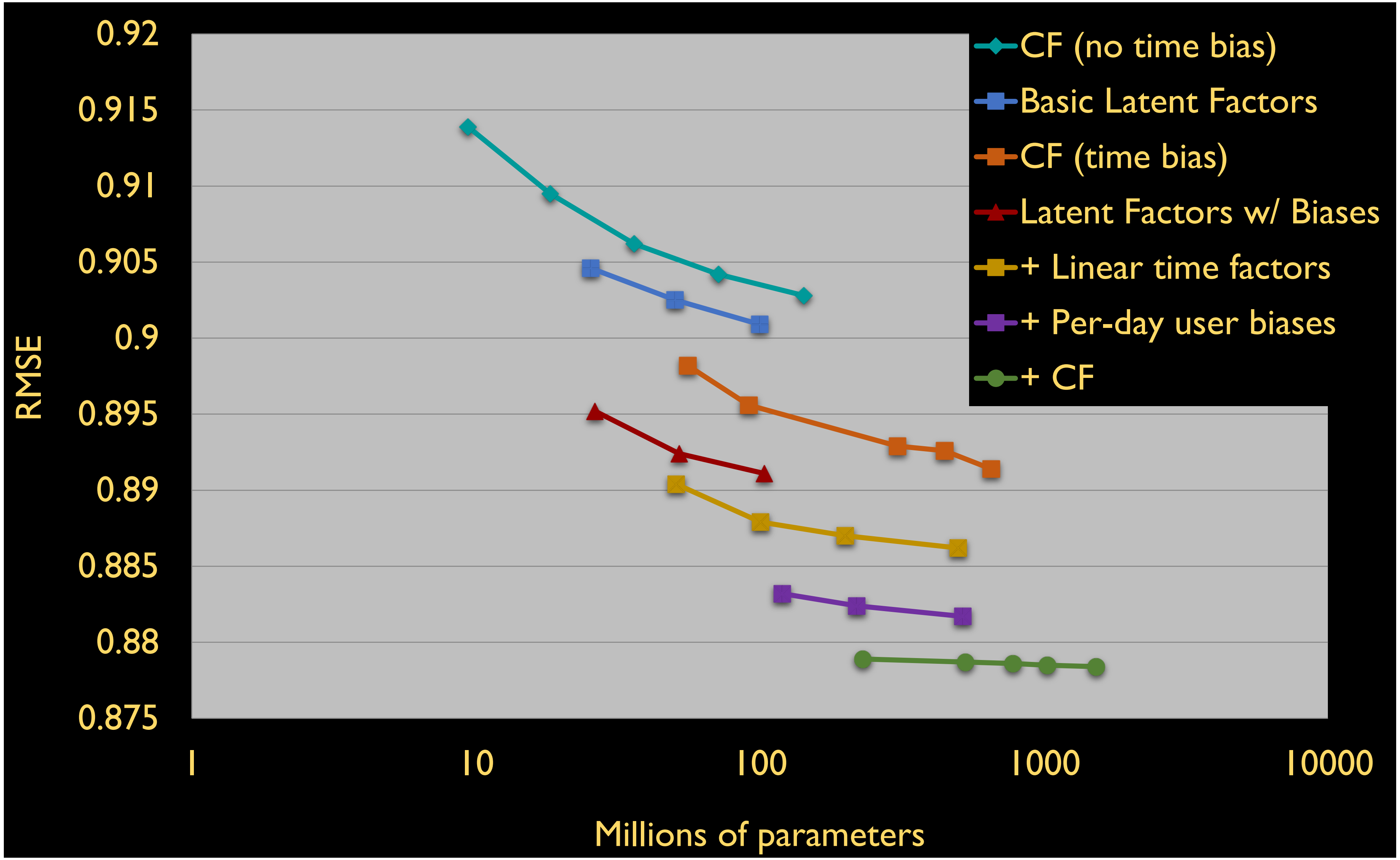




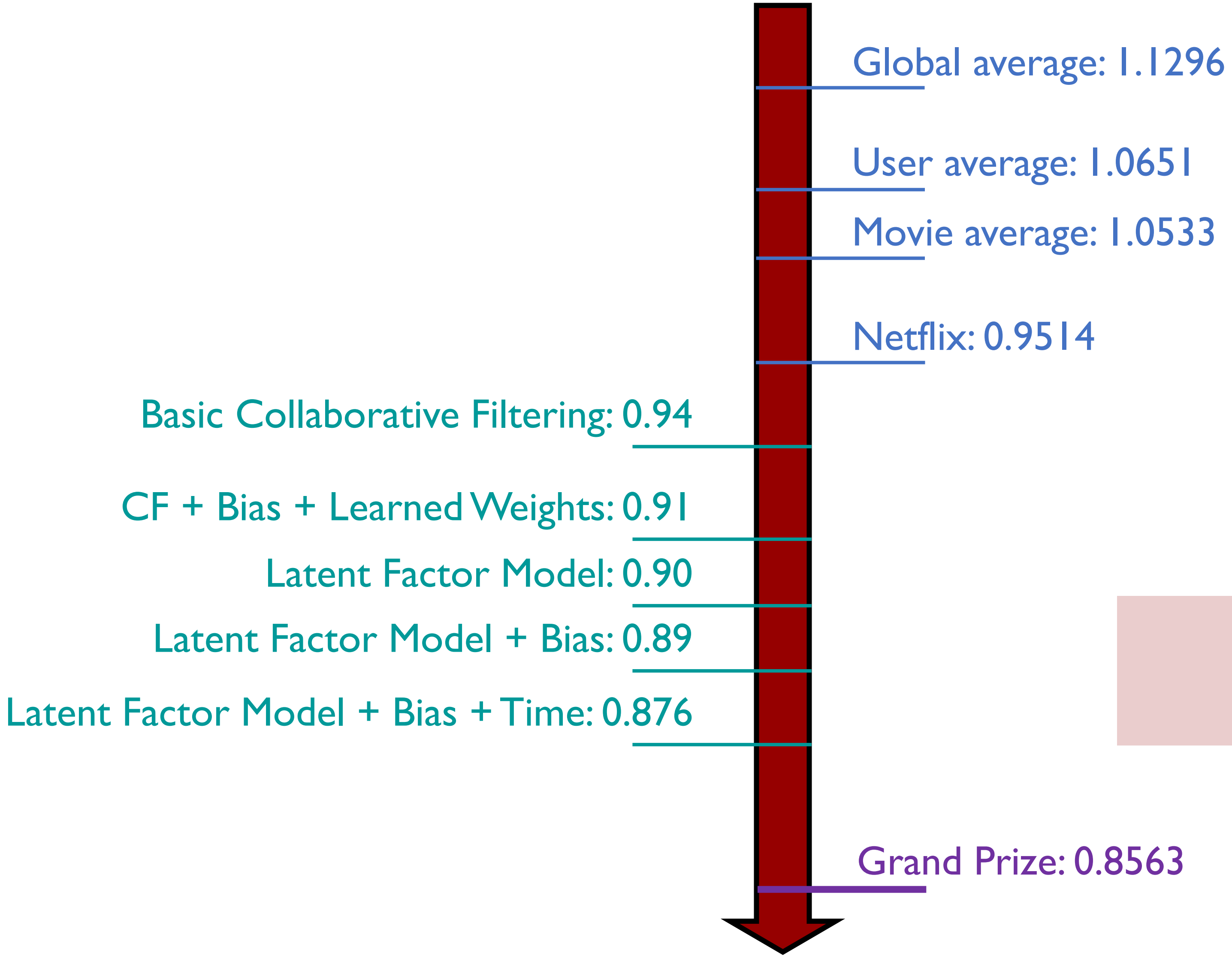
Temporal Biases and Factors

- Latent Factor Model with Constant Bias: $U_{xi} = \mu + b_x + b_i + \boldsymbol{q}_i \boldsymbol{p}_x^T$
- Latent Factor Model with Temporal Bias: $U_{xi} = \mu + b_x(t) + b_i(t) + \boldsymbol{q}_i \boldsymbol{p}_x^T$
  - Make parameters b_x and b_i to depend on time
    - Parameterize time-dependence by linear trends
    - Each bin corresponds to 10 consecutive weeks
      - $b_i(t) = b_i + b_{i,\mathrm{Bin}(t)}$
- One can further add temporal dependence to user/item vectors
  - $\boldsymbol{p}_x(t)$: user preference vector on day t

Adding Temporal Effects



Performance of Various Models



Still no prize!
Getting desperate.

BellKor Recommender System: Winner of the Netflix Challenge

NETFLIX

Netflix Prize

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

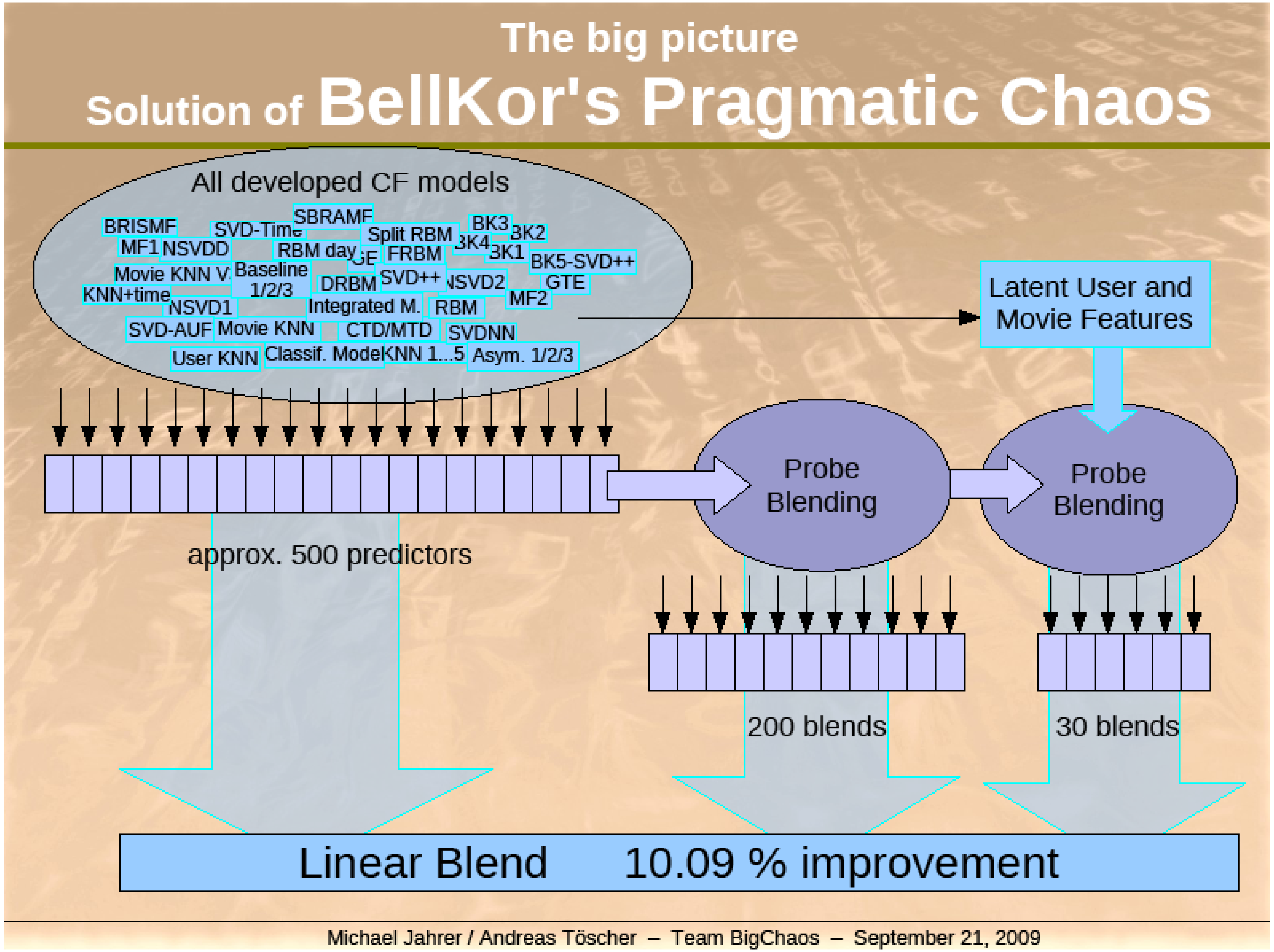
Leaderboard

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
15	Ces	0.8642	9.17	2009-06-24 14:34:14

June 26, 2009
RMSE = 0.8558

A "Kitchen Sink" Approach

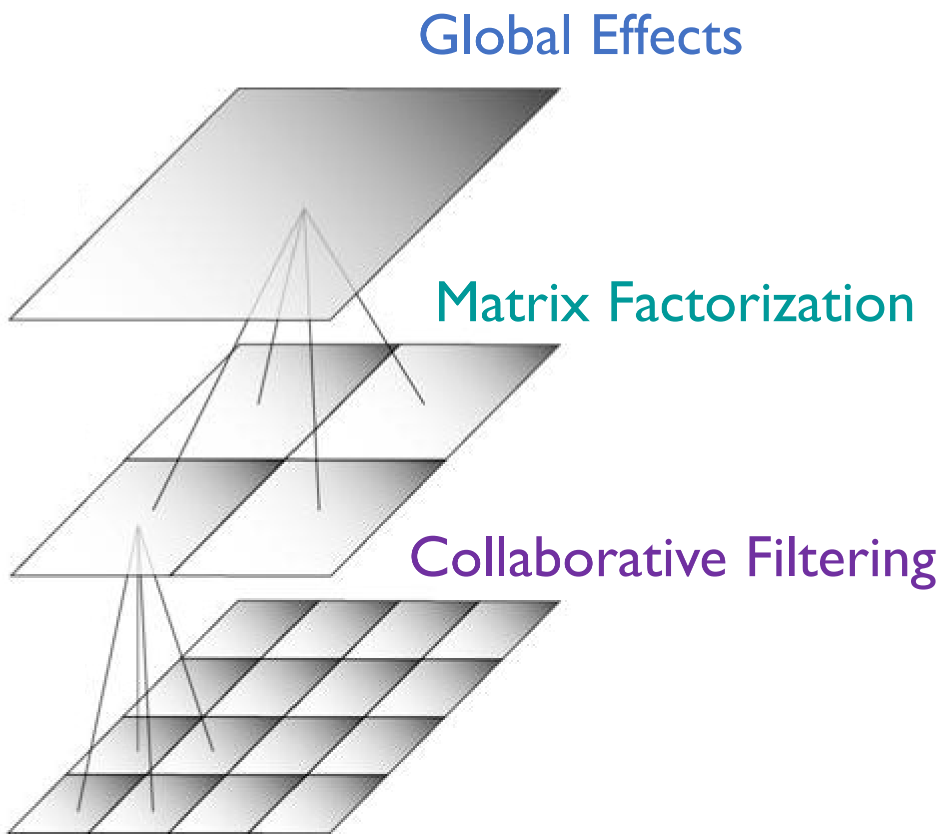


- For a *research project*, this is a very bad idea (since you don't know which part works or why).
- To *achieve a certain level of model performance* (and win a prize), this might be an unavoidable path to take.

BellKor Recommender System: Rough Idea

- 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
- Matrix Factorization:
  - Addressing "regional" effects
- Collaborative Filtering:
  - Extract local patterns



Next Lecture

- Finish the story of the Netflix Prize
- Quiz 2!
    - All policies are the same as Quiz 1 (number of questions, time limit, grading, etc.)
    - Scope:
        - Lecture 8 (Statistical Significance Test in IR Evaluation)
        - Lectures 9 & 10 (Learning to Rank)
        - Lecture 11 (Collaborative Filtering)
        - Lecture 12 (Matrix Factorization)
        - Homework 1

Thank You!

Course Website: https://yuzhang-teaching.github.io/CSCE670-F25.html