Special Topic: Recommender Systems

Dong-Kyu Chae

PI of the Data Intelligence Lab @HYU
Department of Computer Science & Data Science
Hanyang University

Data
Intelligence
LAB

Latent Factor Models

❑ So far…

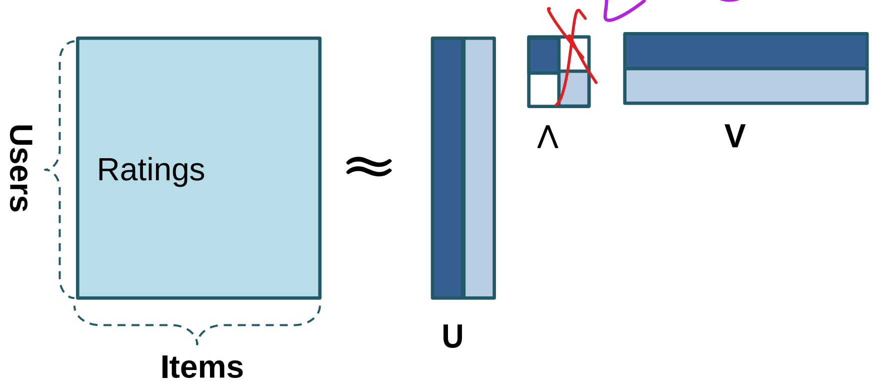
- ❑ We learned KNN-based methods for recommender systems
- ❑ However, the methods are heuristic-based, using hand-crafted functions

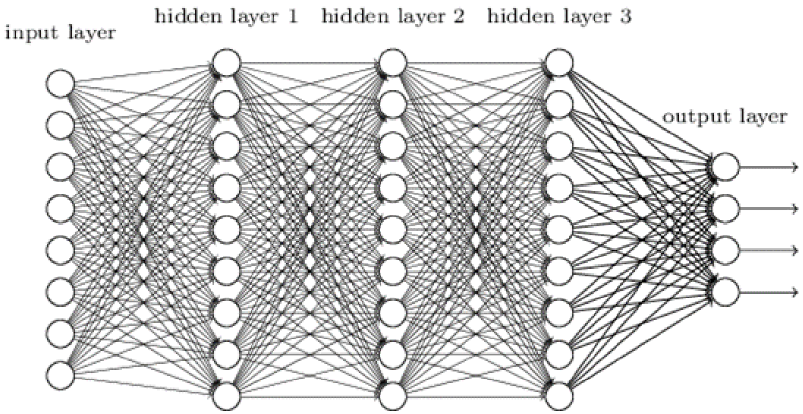
❑ Model-based methods

- ❑ Latent factor models
  - Linear models: matrix factorization, SVD, …
  - Non-linear models: Autoencoder, deep neural networks, …

U랑 V만

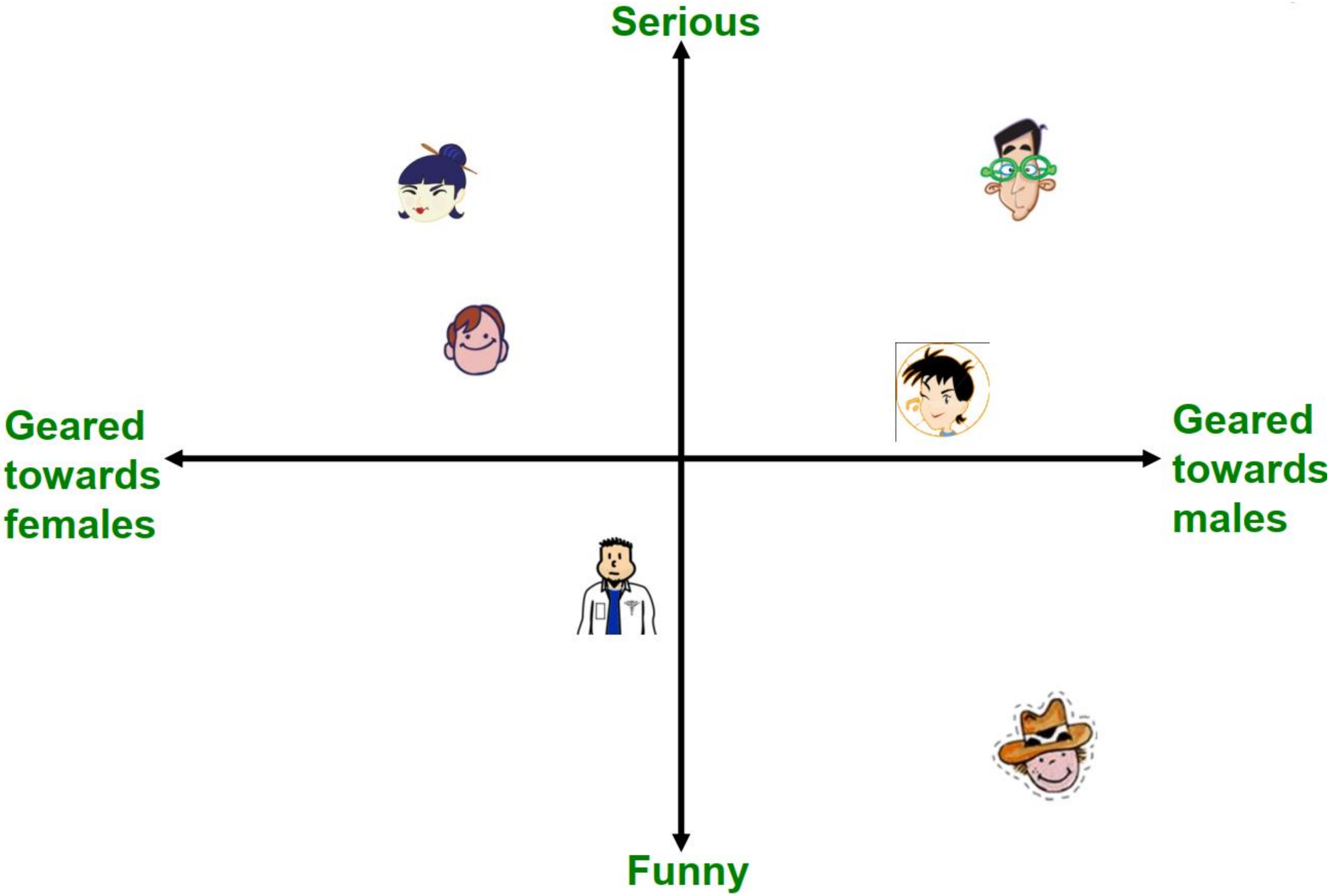
얘는 3개





Latent Factor Models

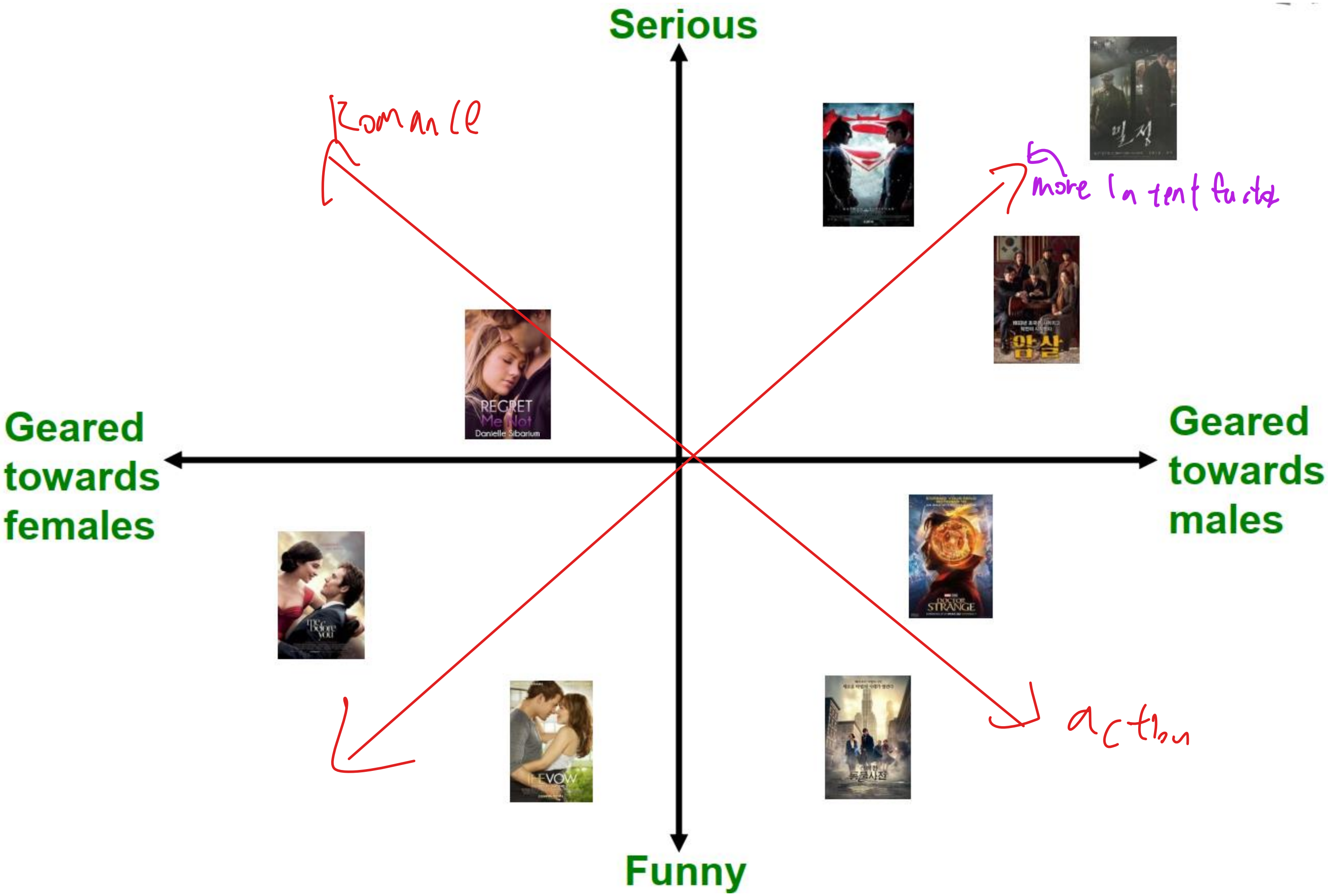
- **What is latent factor?**
  - A feature that describe characteristics of users and items **hidden** in data



Latent Factor Models

❑ What is latent factor?

- ❑ A feature that describe characteristics of users and items **hidden** in data

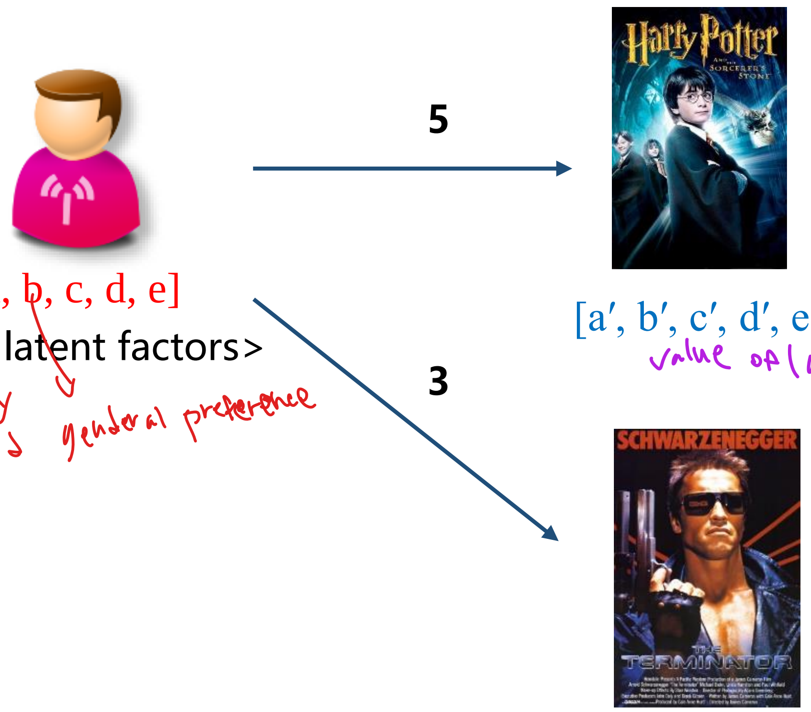


laten factor는 behind the rating matrix에 있다.

Matrix Factorization

❑ Relationship between the latent factors and ratings

- ❑ Assumption: a rating is a result of **interaction between user latent factors** and **item latent factors**



[a, b, c, d, e]

<user latent factors>

mood

genderal preference

[a′, b′, c′, d′, e′]

value of latent factor

[a′, b′, c′, d′, e′]

<item latent factors>

aa′ + bb′ + ·· ee′

[a, b, c, d, e] **x** [a′, b′, c′, d′, e′] **=** **5**

[a, b, c, d, e] **x** [a′, b′, c′, d′, e′] **=** **3**

<linear interaction> <ratings>

Matrix Factorization

❑ Latent factor examples

- ❑ We don’t know the explicit meaning of each factor in the latent matrices
- ❑ But there must be “some” meaning

3

factors

users

user1	0.1	-0.4	0.2
user2	-0.5	0.6	0.5
	-0.2	0.3	0.5
	1.1	2.1	0.3
	-0.7	2.1	-0.2
	-1.0	0.7	0.3

U

x

item1 item2 item3 . . . - -

items

factors

3

1.1	-0.2	0.3	0.5	-2.0	-0.5	0.8	-0.4	0.3	1.4	2.4	-0.9
-0.8	0.7	0.5	1.4	0.3	-1.0	1.4	2.9	-0.7	1.2	-0.1	1.3
2.1	-0.4	0.6	1.7	2.4	0.9	-0.3	0.4	0.8	0.7	-0.6	0.1

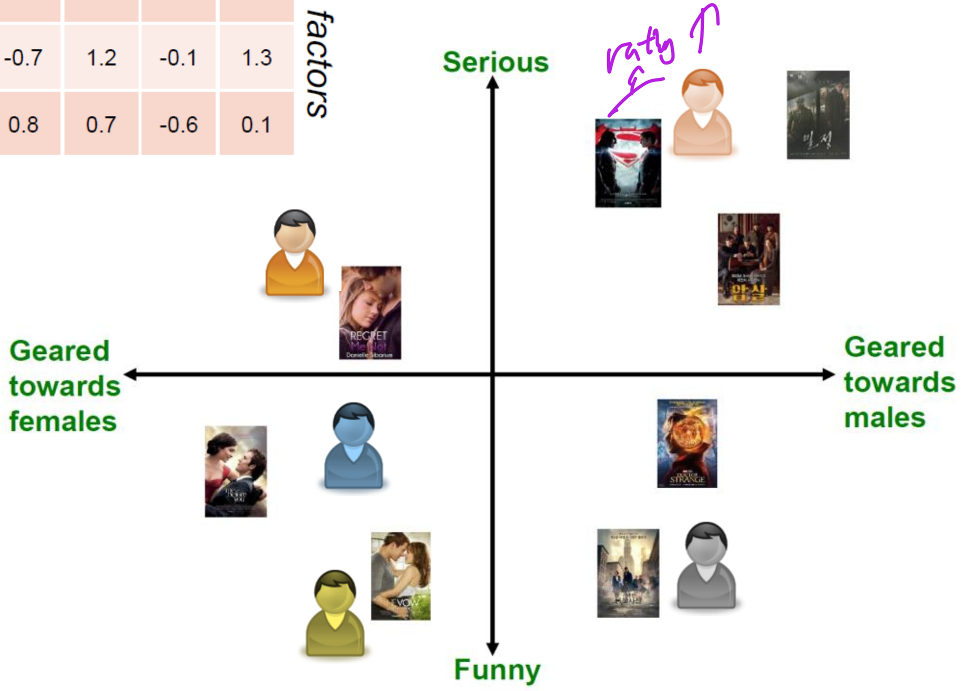
$\mathbf{V^T}$

///

R

User 4's value on the second latent factor

Item 1's value of the second latent factor



Matrix Factorization

❑ Approach

- ❑ "**Learn**" the latent factors, U and V, that may originate the ratings
- ❑ All the latent factors in U and V are model parameters
- ❑ It can be seen as factorization of the rating matrix into two "thin" matrices U and V

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}
u_1	1		3			5			5		4	
u_2			5	4			4			2	1	3
u_3	2	4		1	2		3		4	3	5	
u_4		2	4		5			4			2	
u_5			4	3	4	2					2	5
u_6	1		3		3			2			4	

R

≈

factors

users

0.1	-0.4	0.2
-0.5	0.6	0.5
-0.2	0.3	0.5
1.1	2.1	0.3
-0.7	2.1	-0.2
-1.0	0.7	0.3

U

처음에 random인데

이거를 통해서 update함

items

factors

1.1	-0.2	0.3	0.5	-2.0	-0.5	0.8	-0.4	0.3	1.4	2.4	-0.9
-0.8	0.7	0.5	1.4	0.3	-1.0	1.4	2.9	-0.7	1.2	-0.1	1.3
2.1	-0.4	0.6	1.7	2.4	0.9	-0.3	0.4	0.8	0.7	-0.6	0.1

V^T

Objective:

MSE

$$\min_{U,V} \sum_{u=1}^{\overset{6}{m}} \sum_{i=1}^{\overset{12}{n}} y_{ui}\left(r_{ui} - U_u V_i^T\right)^2$$

acture rating

user

item

$$y_{ui} \begin{cases} 1 & if\ r_{ui} \text{ exists} \\ 0 & \text{otherwise} \end{cases}$$

Random value를
처음에 넣고
계속 update 한다..

Matrix Factorization

❑ Rating prediction with the trained U and V

- ❑ We can reconstruct "**dense**" rating matrix through $U \cdot V^T$

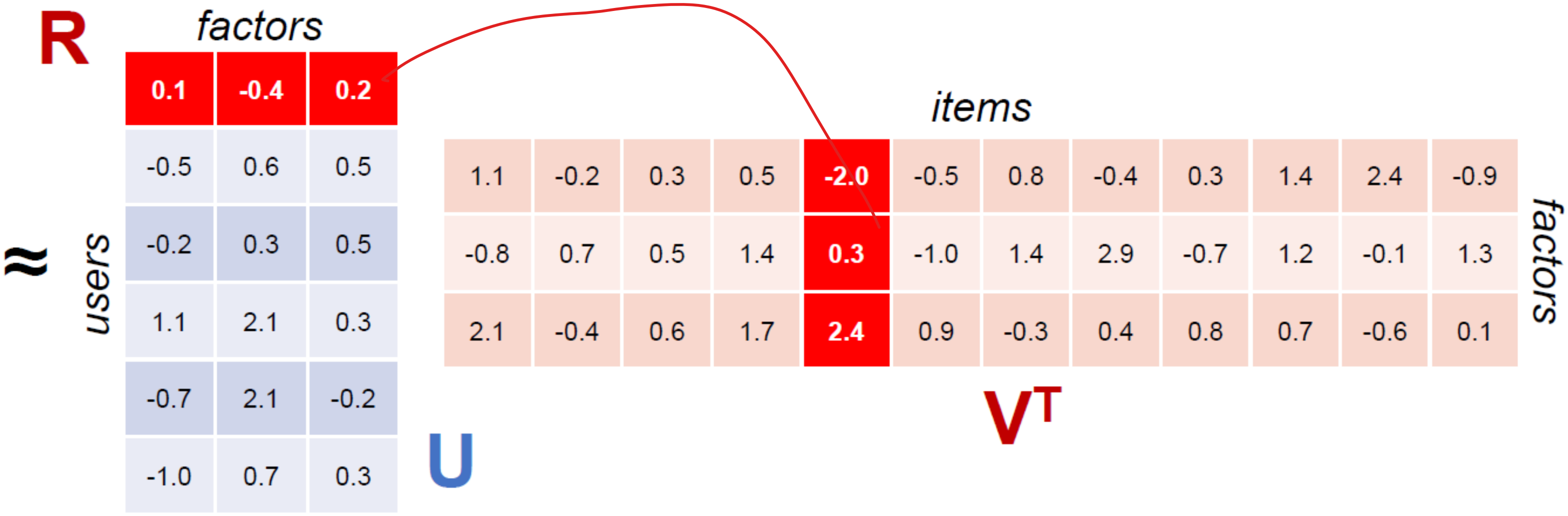
↘ all the values

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}
u_1	1		3		??	5			5		4	
u_2			5	4			4			2	1	3
u_3	2	4		1	2		3		4	3	5	
u_4		2	4		5			4			2	
u_5			4	3	4	2					2	5
u_6	1		3		3			2			4	

$$\tilde{r}_{ui} = U_u V_i^T = \sum_{f=1}^{k} U_{uf} V_{fi}^T$$

U_u = row u of U
V_i^T = column i of V

이걸 predict한다.



U (users × factors):

0.1	-0.4	0.2
-0.5	0.6	0.5
-0.2	0.3	0.5
1.1	2.1	0.3
-0.7	2.1	-0.2
-1.0	0.7	0.3

V^T (factors × items):

1.1	-0.2	0.3	0.5	-2.0	-0.5	0.8	-0.4	0.3	1.4	2.4	-0.9
-0.8	0.7	0.5	1.4	0.3	-1.0	1.4	2.9	-0.7	1.2	-0.1	1.3
2.1	-0.4	0.6	1.7	2.4	0.9	-0.3	0.4	0.8	0.7	-0.6	0.1

Matrix Factorization

- **Objective function: minimizing the sum of squared error**

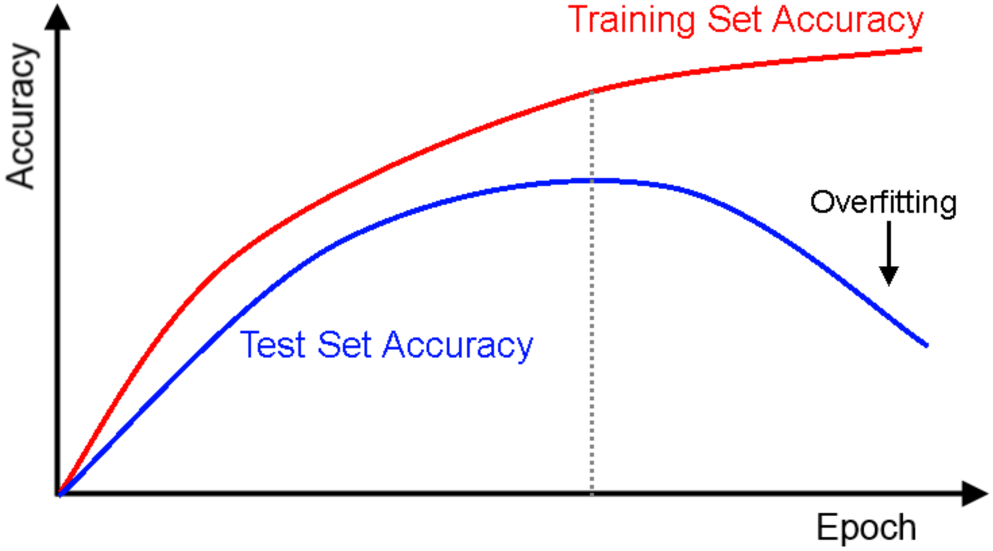
$$\underset{U,V}{\operatorname{argmin}} \sum_{u=1}^{m} \sum_{i=1}^{n} y_{ui}\left(r_{ui} - U_u V_i^T\right)^2$$

- **Practically: there is the overfitting issue**

0.1	-0.4	0.2
-0.5	0.6	0.5
-0.2	0.3 100	0.5
1.1	2.1	0.3
-0.7	2.1	-0.2
-1.0	0.7	0.3

U

1.1	-0.2	0.3	0.5	-2.0
-0.8	0.7	0.5	1.4	0.3 -81
2.1	-0.4	0.6	1.7	2.4



Matrix Factorization

❑ Objective function with the regularization term

Goodness of fit Regularization

$$E = \underset{U,V}{\operatorname{argmin}} \sum_{u=1}^{m} \sum_{i=1}^{n} y_{ui}(r_{ui} - U_u V_i^T)^2 + \lambda(||U||^2 + ||V||^2)$$

0.02
0.01
0.1

overfitting을 막기위해

→ 패널티

& value itself of U & V

- ❑ The goodness of fit is to reduce the prediction error
- ❑ The regularization term is used to alleviate the overfitting problem

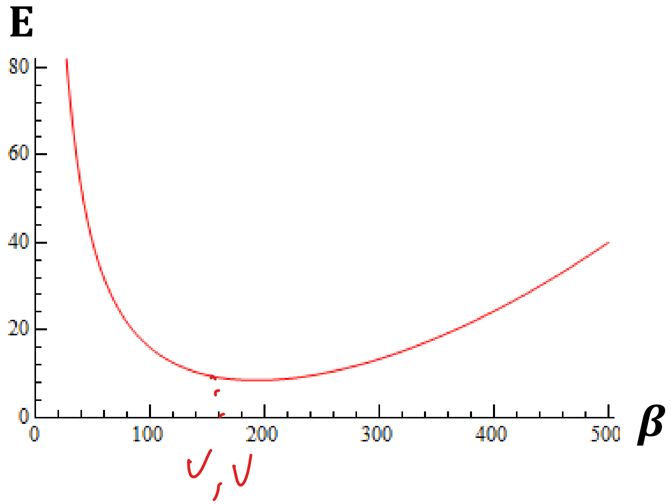
❑ Two ways to training the MF model

- ❑ Stochastic gradient descent (**SGD**)
  - Training U and V simultaneously
- ❑ Alternating least squares (**ALS**)
  - Fix one of U and V, and then optimize the other

(Review) Optimization

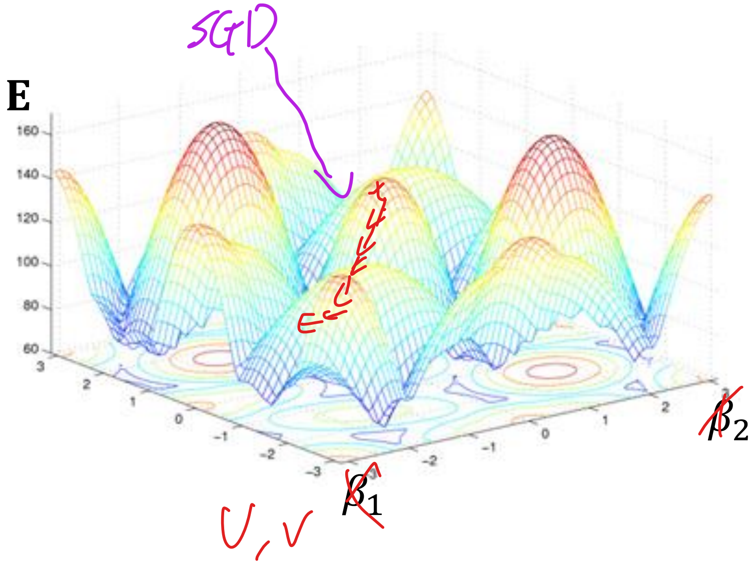
- **If the cost function E is simple:**
  - You can directly find the optimal parameters via computing the points where the **derivative** of E becomes zero

$$\frac{\partial E}{\partial \cancel{\beta}\ U,V} = 0$$



- **If the cost function E is complex:**
  - Multiple model parameters…
  - Model parameters are aligned with each other…
  - We cannot directly solve the problem via

$$\frac{\partial E}{\partial \cancel{\beta}\ U,V} = 0$$



Stochastic Gradient Descent

❑ Objective:

Goodness of fit Regularization

$$\underset{U,V}{\operatorname{argmin}} \sum_{u=1}^{m} \sum_{i=1}^{n} y_{ui}(r_{ui} - U_u V_i^T)^2 + \lambda(||U||^2 + ||V||^2)$$

❑ Process:

1. Initialize U and V randomly Starting position
2. Repeat:
   1. Choose a pair user item (u, i) randomly
   2. Then the loss on the chosen (u, i) is:

$$E = \frac{1}{2}((r_{ui} - U_u V_i^T)^2 + \lambda(||U||^2 + ||V||^2))$$

loss

   3. Update via

$\frac{\partial E}{\partial U}$

difference

- $e_{ui} = r_{ui} - U_u V_i^T$
- $U_u \leftarrow U_u + \eta(e_{ui} V_i^T - \lambda U_u)$
- $V_i \leftarrow V_i + \eta(e_{ui} U_u - \lambda V_i)$
- Update U_u and V_i iteratively.

$\frac{\partial E}{\partial V}$

$V_{t+1} = V_t - \eta \frac{\partial E}{\partial v}$

They are the partial derivative of U_u and V_i.

detail skip

0.1	-0.4	0.2
-0.5	0.6	0.5
-0.2	0.3	0.5
1.1	2.1	0.3
-0.7	2.1	-0.2
-1.0	0.7	0.3

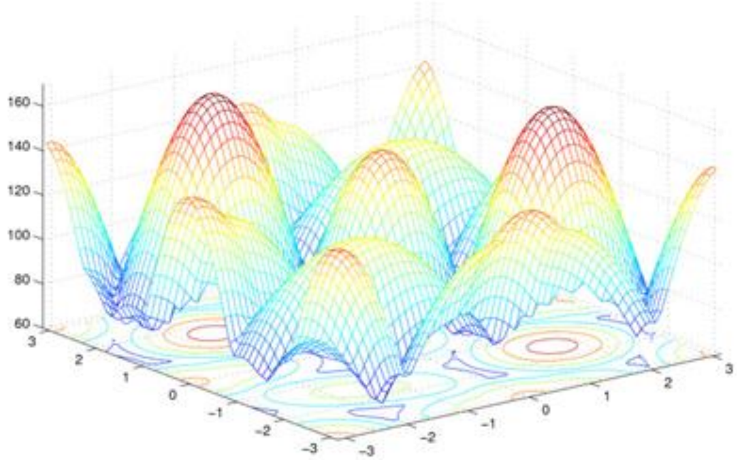
1.1	-0.2	0.3	0.5	-2.0
-0.8	0.7	0.5	1.4	0.3
2.1	-0.4	0.6	1.7	2.4

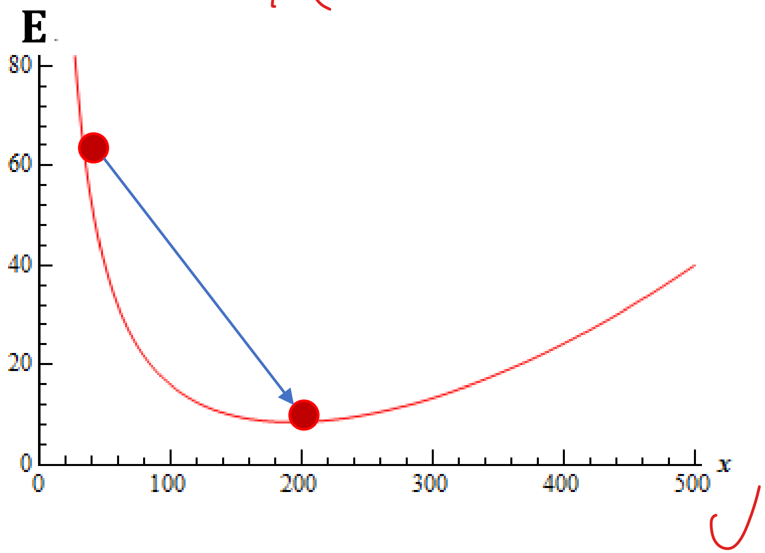
Alternating Least Squares

- Fix one, then the loss surface becomes simple

Goodness of fit   Regularization

$$\underset{U,V}{\operatorname{argmin}} \sum_{u=1}^{m} \sum_{i=1}^{n} y_{ui}\left(r_{ui} - U_u \underset{\downarrow \text{fix}}{\boxed{V_i^T}}\right)^2 + \lambda\left(\|U\|^2 + \|\underset{\downarrow \text{fix}}{\boxed{V}}\|^2\right)$$





$$\frac{\partial E}{\partial U_i} = 0$$

Alternating Least Squares

ALS

❑ Objective:

Goodness of fit | Regularization

$$\underset{U,V}{\operatorname{argmin}} \sum_{u=1}^{m}\sum_{i=1}^{n} y_{ui}\left(r_{ui} - \boxed{U}_u V_i^T\right)^2 + \lambda\left(||\boxed{U}||^2 + ||V||^2\right)$$

❑ Procedure

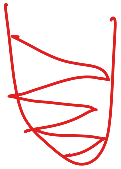
1. Initialize the user latent factors U randomly
2. For each item i, let r_i be the vector of ratings of that item. Compute:

$$\frac{\partial E}{\partial V} = 0 \qquad \underbrace{V_i = (U^T U + \lambda I)^{-1} U^T r_i}_{V_1\ V_2\ \cdots\ V_n} \quad \text{(here, } U \text{ is constant)}$$

3. For each user u, let r_u be the vector of ratings of that ~~item~~ user. Compute:

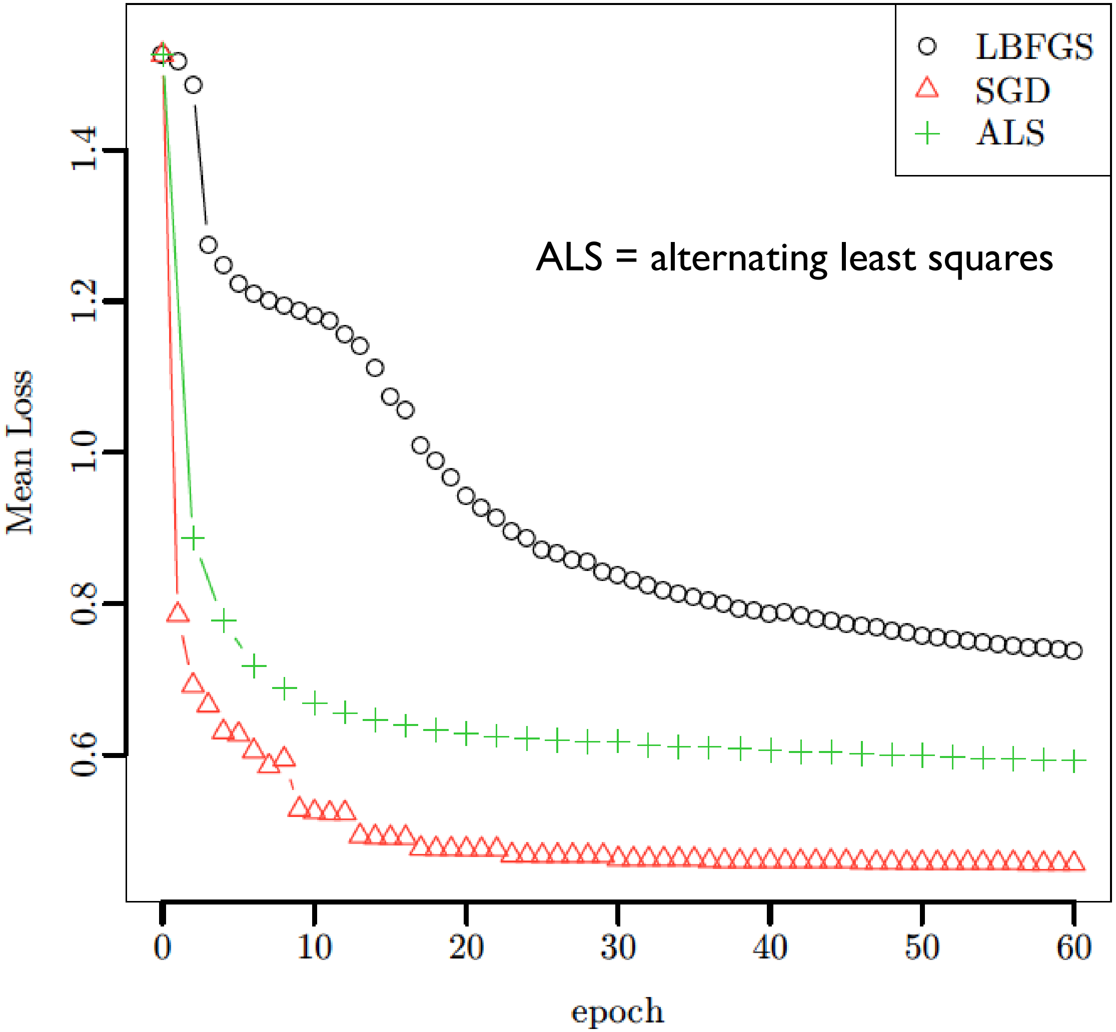
$$\frac{\partial E}{\partial U} \longleftarrow \qquad \underbrace{U_u = (V^T V + \lambda I)^{-1} V^T r_u}_{U_1\ U_2\ \cdots\ U_m \rightarrow U} \quad \text{(here, } V \text{ is constant)}$$

4. Repeat 2), 3) until convergence



Matrix Factorization

❑ SGD VS ALS



ALS = alternating least squares

Putting Bias to Prediction

- **Idea: The user rating consists of four parts:**
  - μ : Global average for all ratings
  - b_u : User bias for ratings
  - b_i : Item bias for ratings
  - $U_u V_i^T$: Interaction between user u and item i

$$r_{ui} = \mu + b_u + b_i + U_u V_i^T$$

Overall mean rating

Bias for user *u*

Bias for movie *i*

User-Movie interaction

matrix fatorization

- **Example**
  - Global average μ = 3.7
  - You are a critical reviewer: Your ratings are 1 star lower than the mean: b_u = -1
  - Star Wars is the popular movie: This gets a mean rating of 0.5 higher than average movie: b_i = +0.5

Putting Bias to Prediction

- **Objective:**

Goodness of fit

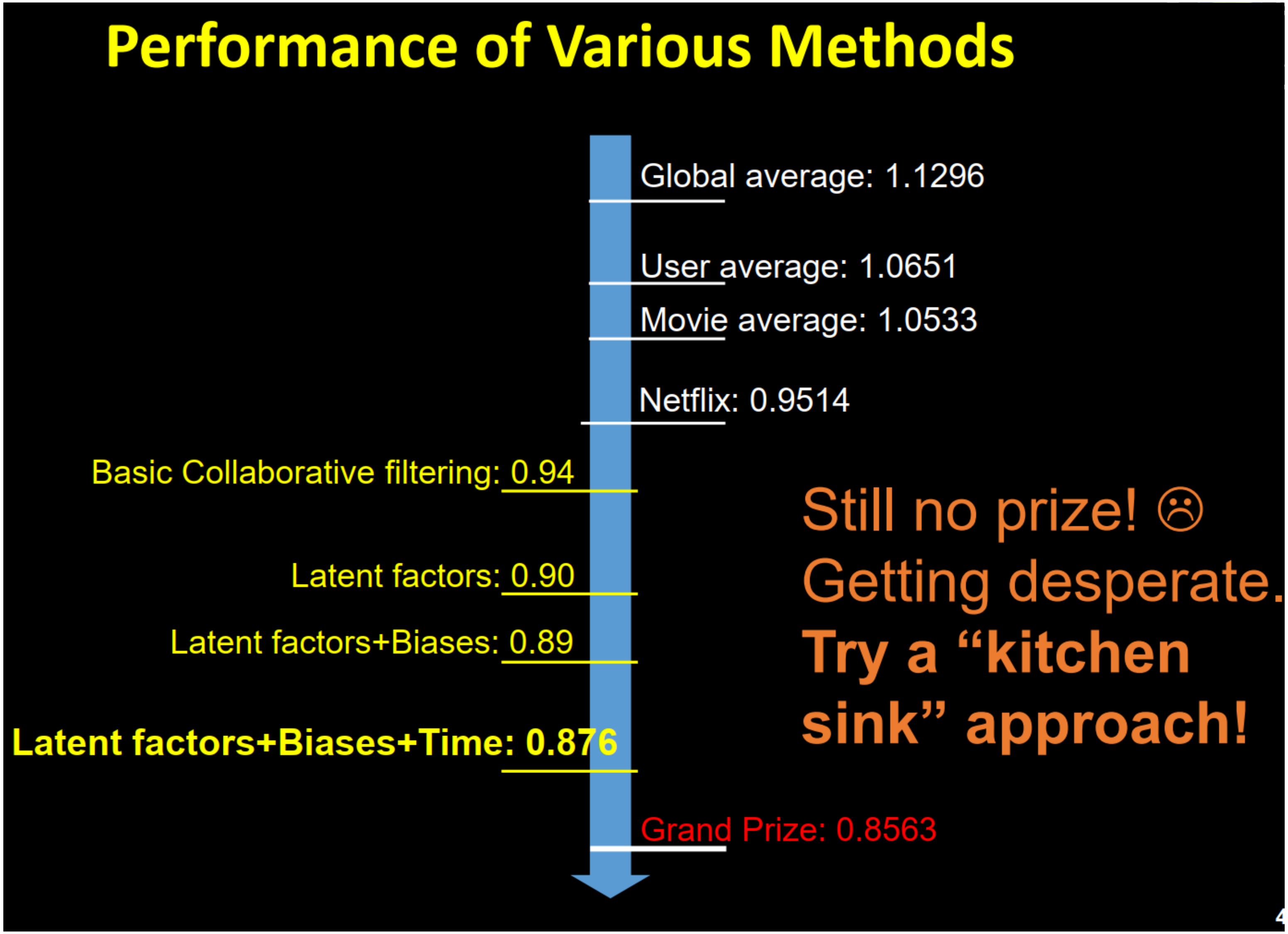
$$\underset{U,V,b_u,b_i}{\operatorname{argmin}} \sum_{u=1}^{m} \sum_{i=1}^{n} y_{ui} \left(r_{uj} - (\mu + b_u + b_i + U_u V_i^T) \right)^2$$

$$+\lambda \left(\sum_{u=1}^{m} ||U_u||^2 + \sum_{i=1}^{n} ||V_i||^2 + \sum_{u=1}^{m} ||b_u||^2 + \sum_{i=1}^{n} ||b_i||^2 + \right)$$

Regularization

- **Solve: stochastic gradient descent**

Results



Performance of Various Methods

Global average: 1.1296

User average: 1.0651

Movie average: 1.0533

Netflix: 0.9514

Basic Collaborative filtering: 0.94

Latent factors: 0.90

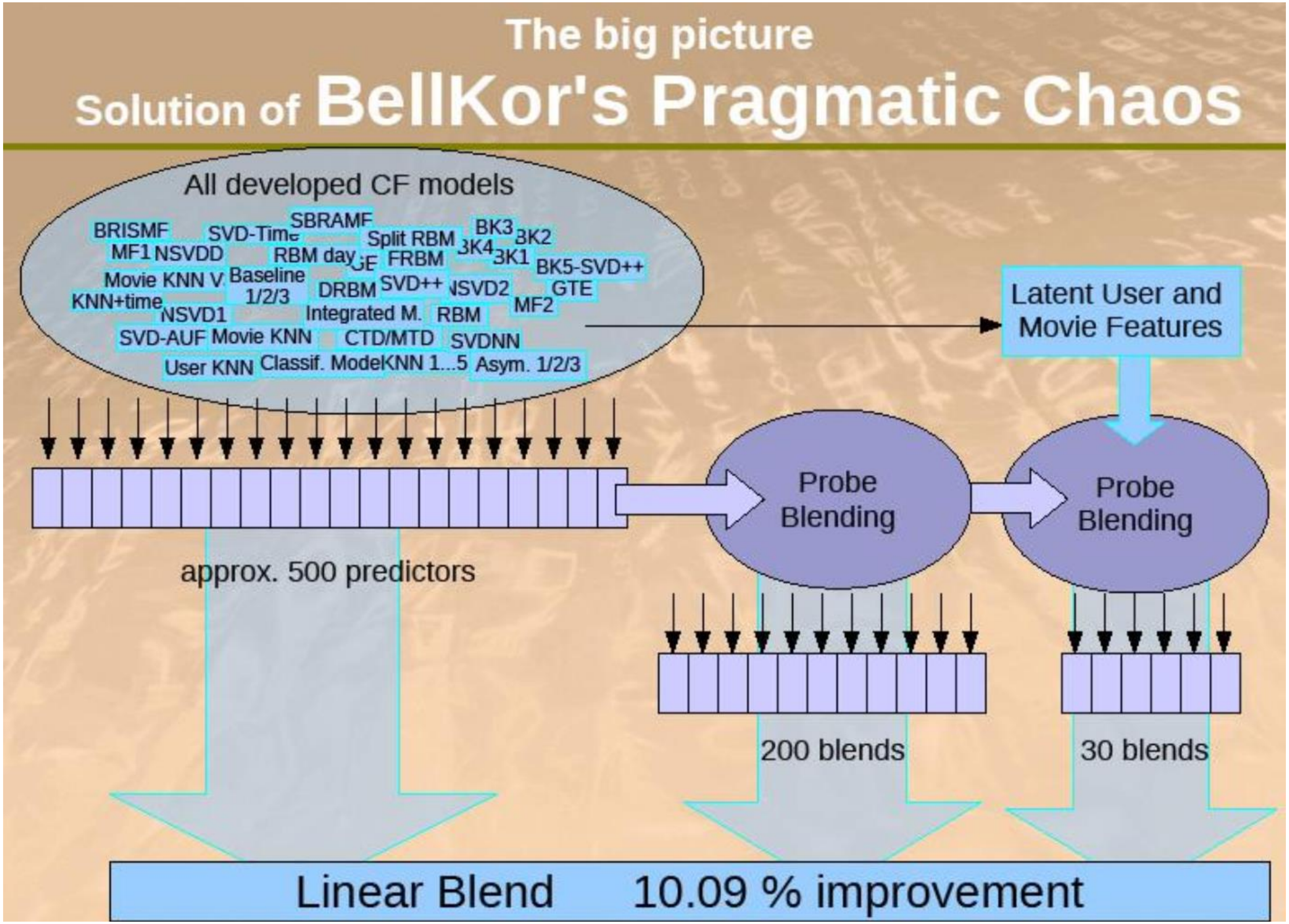
Latent factors+Biases: 0.89

Latent factors+Biases+Time: 0.876

Grand Prize: 0.8563

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

Finally....



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. ModeKNN 1...5 Asym. 1/2/3

Latent User and Movie Features

approx. 500 predictors

Probe Blending

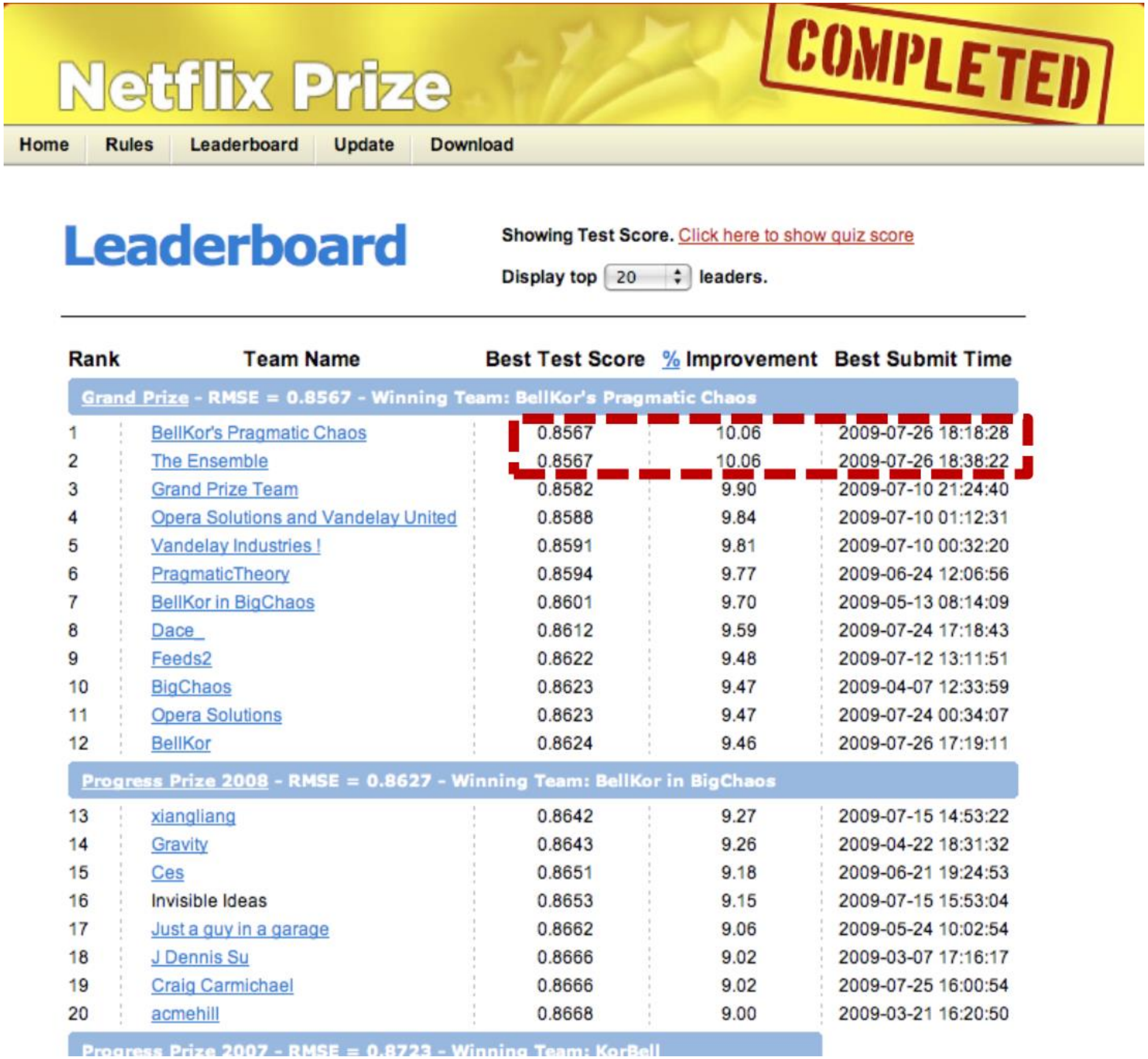
Probe Blending

200 blends

30 blends

Linear Blend 10.09 % improvement

Finally....



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				

Finally....

- $1 Million Awarded Sept 21st 2009

Thank You