华中科技大学计算机科学与技术学院

School of Computer Science &Technology,HUST

Chapter 6: Recommender Systems---Latent Factor Models

崔金华
电子邮箱: jhcui@hust.edu.cn
个人主页: https://csjhcui.github.io/

致谢:课件主要参考《 Mining of Massive Datasets》(Second Edition) J. Leskovec, A. Rajaraman, J. Ullman的相关课件, 特此致谢!!!

The Netflix Prize

❑Training data

- 100 million ratings(1亿条打分数据), 480,000 users, 17,770 movies
- 6 years of data: 2000-2005

❑Test data

- Last few ratings of each user (280万条数据)
- **Evaluation criterion:** Root Mean Square Error (RMSE) (均方误差) $= \frac{1}{|R|}\sqrt{\sum_{(i,x)\in R}(\hat{r}_{xi} - r_{xi})^2}$
- **Netflix’s CineMath system RMSE: 0.9514**

❑Competition

- 2,700+ teams
- **$1 million** prize (100万美元大奖) for 10% improvement on Netflix

The Netflix Utility Matrix *R*

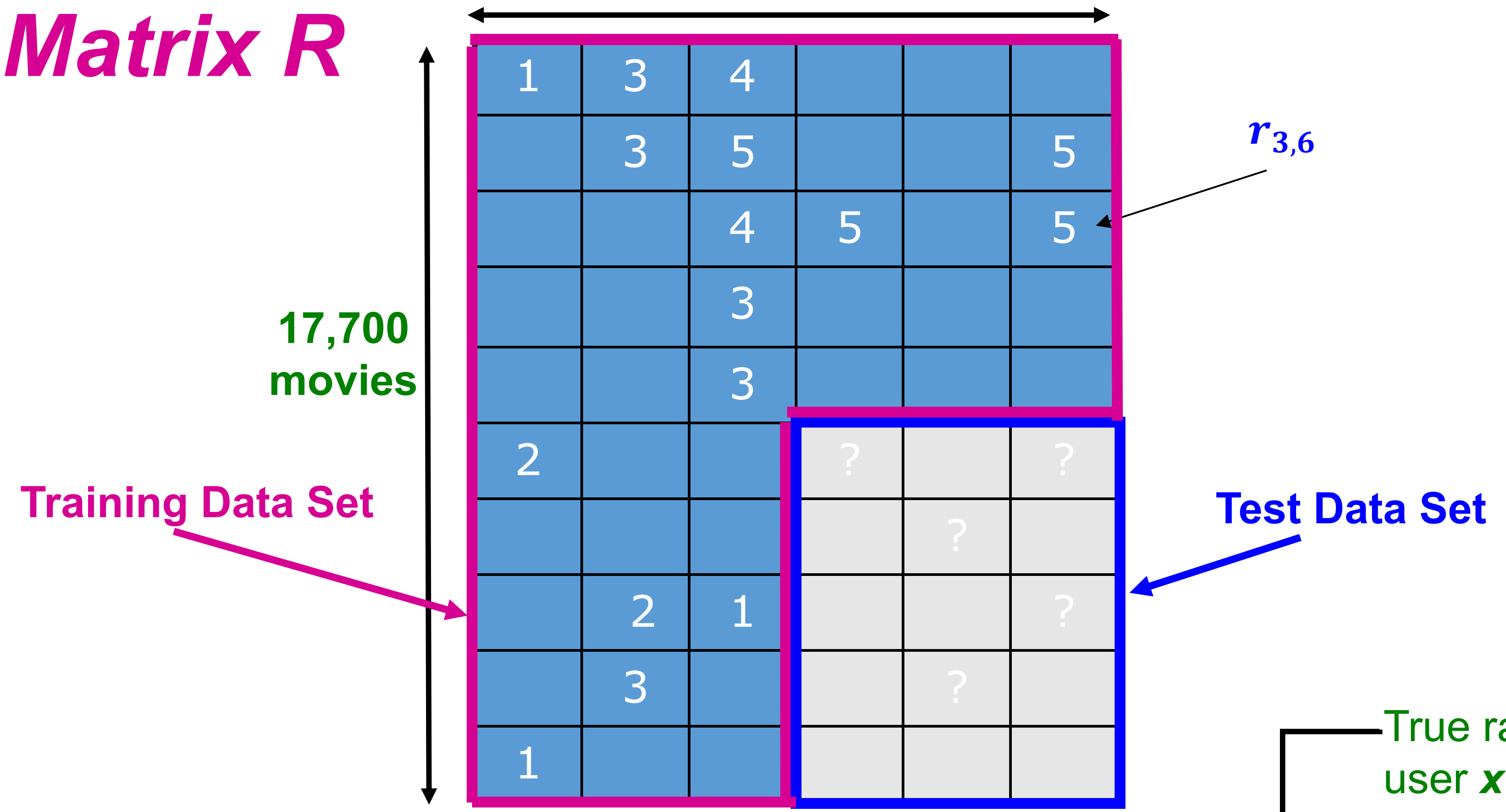
Matrix R

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					

Utility Matrix *R*: Evaluation



Matrix R

17,700 movies

$r_{3,6}$

Training Data Set

Test Data Set

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

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

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

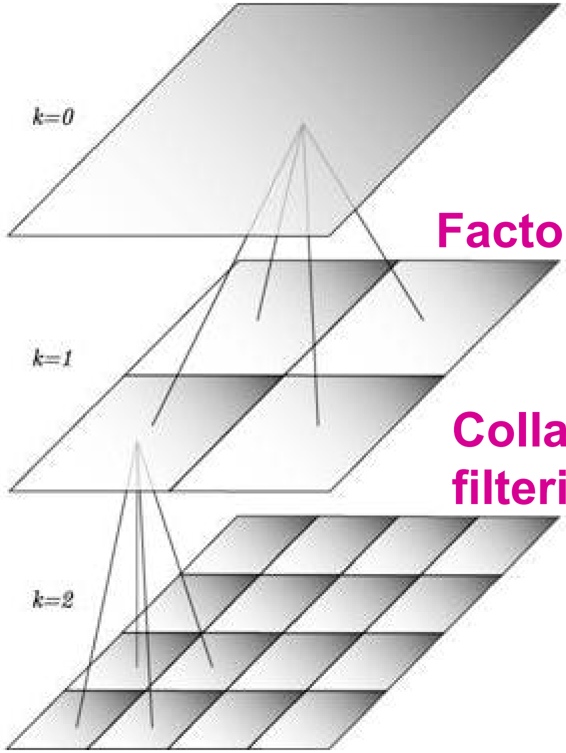
Predicted rating

BellKor Recommender System

□**BellKor’ s Pragmatic Chaos(BPC), 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(矩阵分解):**
  - Addressing “regional” effects
- **Collaborative filtering:**
  - Extract local patterns



Global effects

Factorization

Collaborative filtering

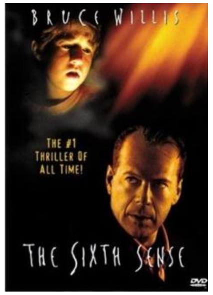
Papers: 《The BellKor Solution to the Netflix Grand Prize》
《The Pragmatic Theory Solution to the Netflix Grand Prize》
《The BitCHaos Solution to the Netflix Grand Prize》

Modeling Local & Global Effects

❑Global:

- ➢Mean movie rating(电影的平均打分): **3.7 stars**
- ➢*The Sixth Sense* 《第六感》 is **0.5** stars above avg.
- ➢Joe rates **0.2** stars below avg.

  ⇒ Baseline estimation:
  ***Joe* will rate *The Sixth Sense* 3.7+0.5-0.2=4 stars**



❑Local neighborhood (e.g.,CF基于协同过滤, NN基于神经网络):

- ➢*Joe* didn’t like related movie *Signs* 《天兆》
- ➢**⇒ Final estimate:**
  ***Joe* will rate *The Sixth Sense* 3.8 stars**

Recap: Collaborative Filtering (CF)

- □ **Earliest and most popular collaborative filtering method (基于协同过滤的推荐方法)**
- □ Derive unknown ratings from those of “**similar**” movies (item-item variant)
- □ Define **similarity measure** ***s***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:**

$$\hat{r}_{xi} = b_{xi} + \frac{\sum_{j \in N(i;x)} s_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i;x)} s_{ij}}$$

e.g., Item-item CF r_{xi}, then N r_{xj}

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***

$\widehat{r_{xi}}$: *predicted rating of user **x** on item **i***

r_{xj}: *rating of user **x** on item **j***

S_{ij}: *similarity of item **i** and **j***

N(i;x): *set of items similar to item **i** that were rated by **x***

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.**:

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

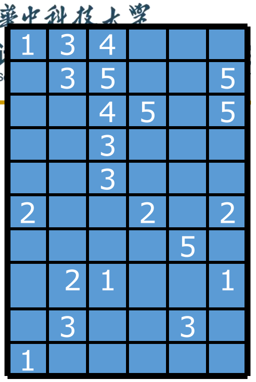
- $\boldsymbol{N(i;x)}$ … set of movies rated by user $\boldsymbol{x}$ that are similar to movie $\boldsymbol{i}$
- $\boldsymbol{w_{ij}}$ is the interpolation weight (插值权值, some real number)
  - We allow: $\sum_{j \in N(i,x)} w_{ij} \neq 1$
- $\boldsymbol{w_{ij}}$ models interaction between pairs of movies (it does not depend on user ***x***)

Idea: Interpolation Weights *w_{ij}*

- ❑ $\widehat{r_{xi}} = b_{xi} + \sum_{j \in N(i;x)} w_{ij}(r_{xj} - b_{xj})$
- ❑ **How to set *w_{ij}*?**
  - ➢ Remember, error metric **RMSE** 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 user ***x*** and all other users that rated ***i***

Why is this a good idea?

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 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***

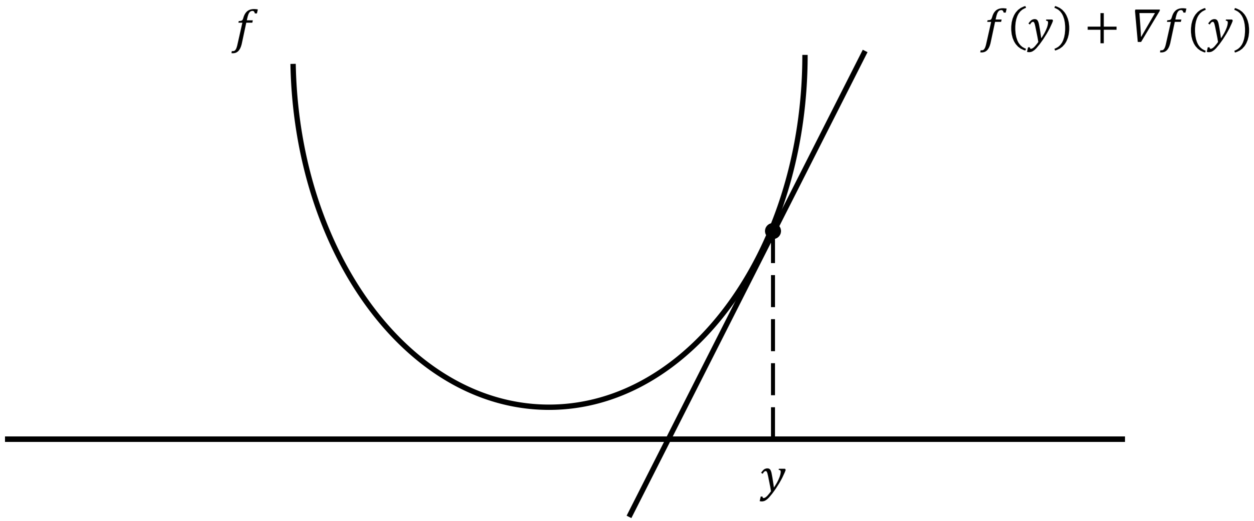
- Find $\boldsymbol{w_{ij}}$ that minimize **SSE** on **training data**!

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

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

Detour: Minimizing a function

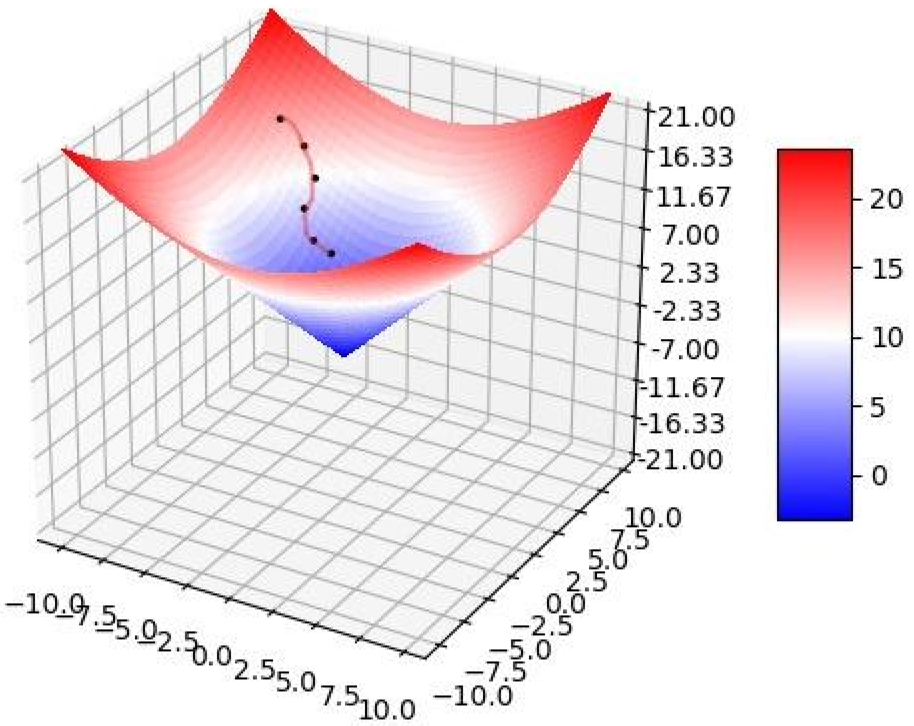
- **A simple way to minimize a function $f(x)$:**
  - Compute the take a derivative ∇f
  - **Start at some point y and evaluate $\nabla f(y)$**
  - **Make a step in the reverse direction of the gradient: $y = y - \nabla f(y)$**
  - **Repeat until converged**



Detour: Minimizing a function

- **A simple way to minimize a function $f(x)$:**
  - Compute t
  - **Start at so**
  - **Make a ste** **nt: $y = y - \nabla f(y)$**
  - **Repeat un**

(y)



/ y

Interpolation Weights

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

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

- **Gradient decent(梯度下降):**
  - **Iterate until convergence:** $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla_{\boldsymbol{w}} \boldsymbol{J}$

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

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

for $\boldsymbol{j} \in \{\boldsymbol{N}(\boldsymbol{i};\boldsymbol{x}),\ \forall \boldsymbol{i}, \forall \boldsymbol{x}\ \}$

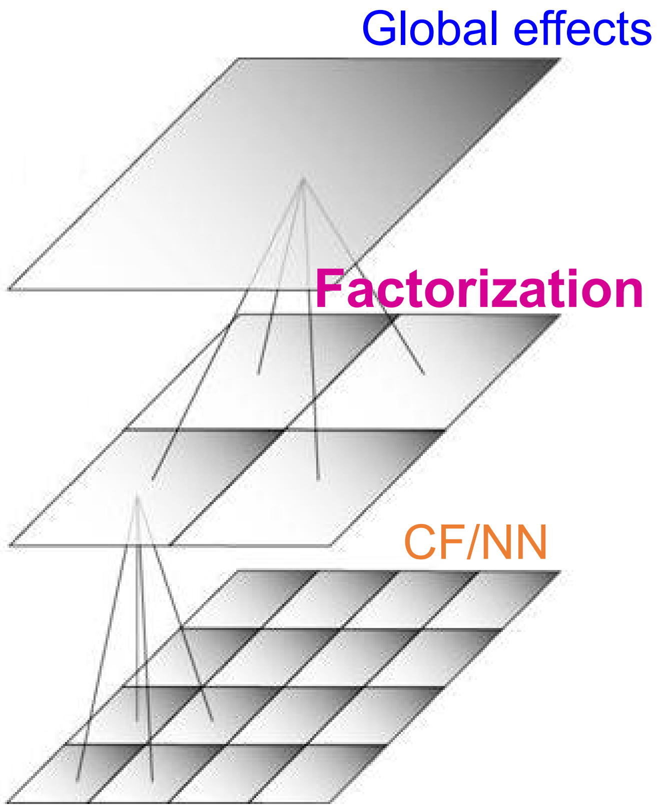
else $\frac{\partial J(w)}{\partial w_{ij}} = \mathbf{0}$

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

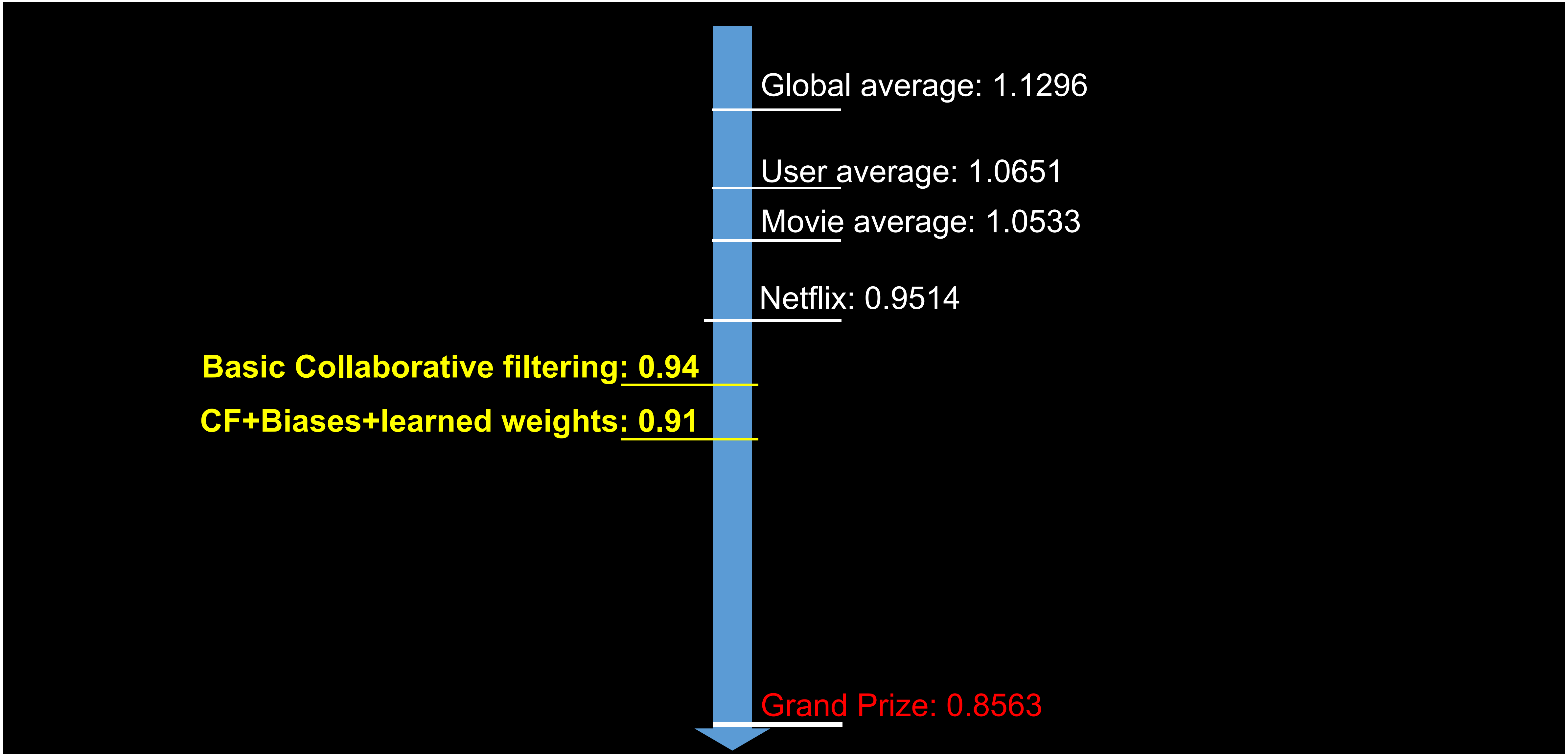
```
while $|w_{new} - w_{old}| > \varepsilon$:
    $w_{old} = w_{new}$
    $w_{new} = w_{old} - \eta \cdot \nabla w_{old}$
```

Interpolation Weights

- **So far:** $\widehat{r_{xi}} = b_{xi} + \sum_{j \in N(i;x)} w_{ij}(r_{xj} - b_{xj})$
  - Weights ***w*** derived based on their role; **no use of an arbitrary similarity measure** (***w**ij*≠ ***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

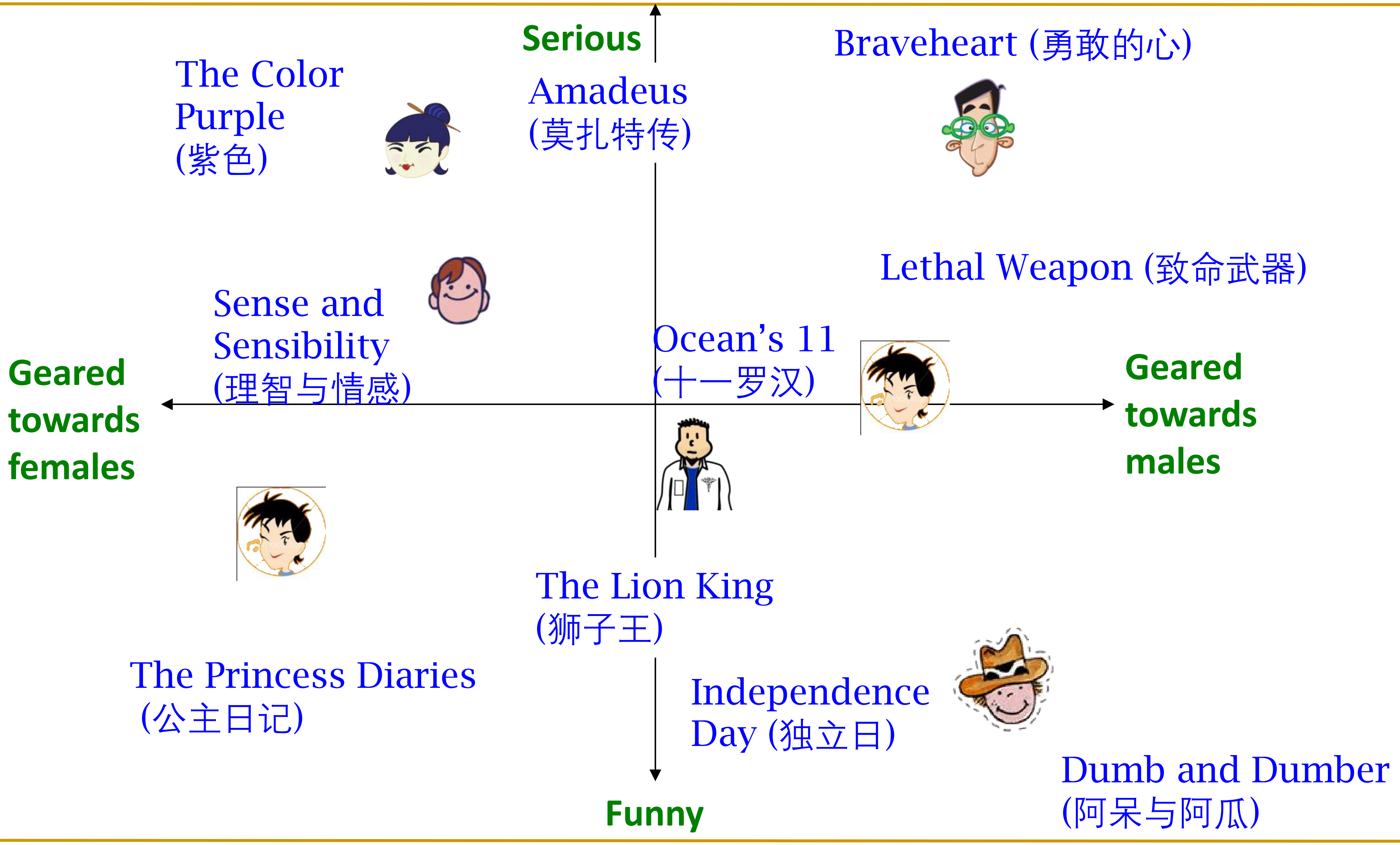
Netflix: 0.9514

Basic Collaborative filtering: 0.94

CF+Biases+learned weights: 0.91

Grand Prize: 0.8563

Latent Factor Models (e.g., SVD)



Serious

The Color Purple (紫色)

Amadeus (莫扎特传)

Braveheart (勇敢的心)

Lethal Weapon (致命武器)

Sense and Sensibility (理智与情感)

Ocean’s 11 (十一罗汉)

Geared towards females

Geared towards males

The Lion King (狮子王)

The Princess Diaries (公主日记)

Independence Day (独立日)

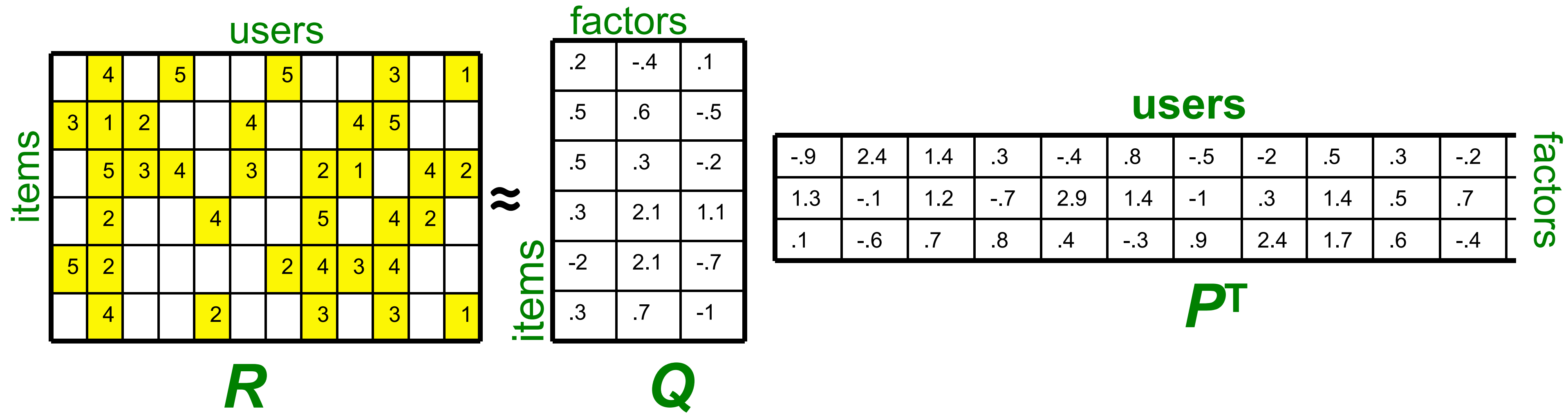
Dumb and Dumber (阿呆与阿瓜)

Funny

Latent Factor Models

- “SVD” on Netflix data: **R ≈ $\boldsymbol{Q \cdot P^T}$**

SVD: $A = U \Sigma V^T$
(下个章节将会详细讲解)



R:

	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

Q:

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

P^T:

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

- For now let’ s assume we can approximate the rating matrix ***R*** as a product of “thin” $\boldsymbol{Q \cdot P^T}$
  - ***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