THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

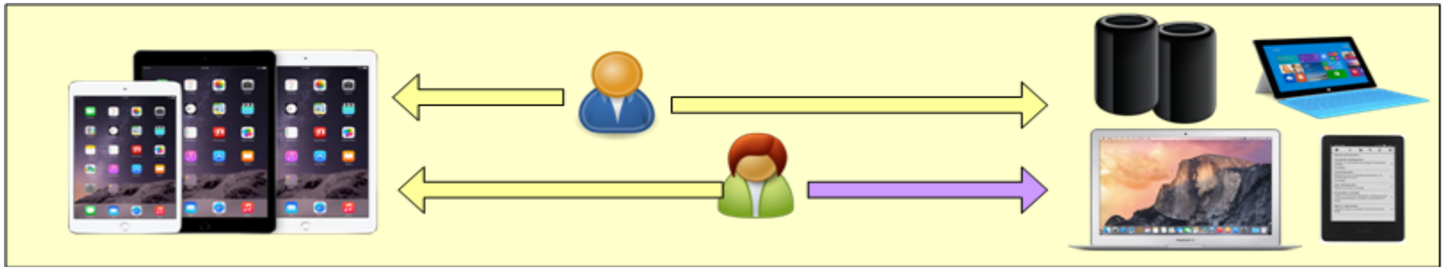
COMP4434 Big Data Analytics

Lecture 6
Collaborative Filtering

HUANG Xiao
xiaohuang@comp.polyu.edu.hk

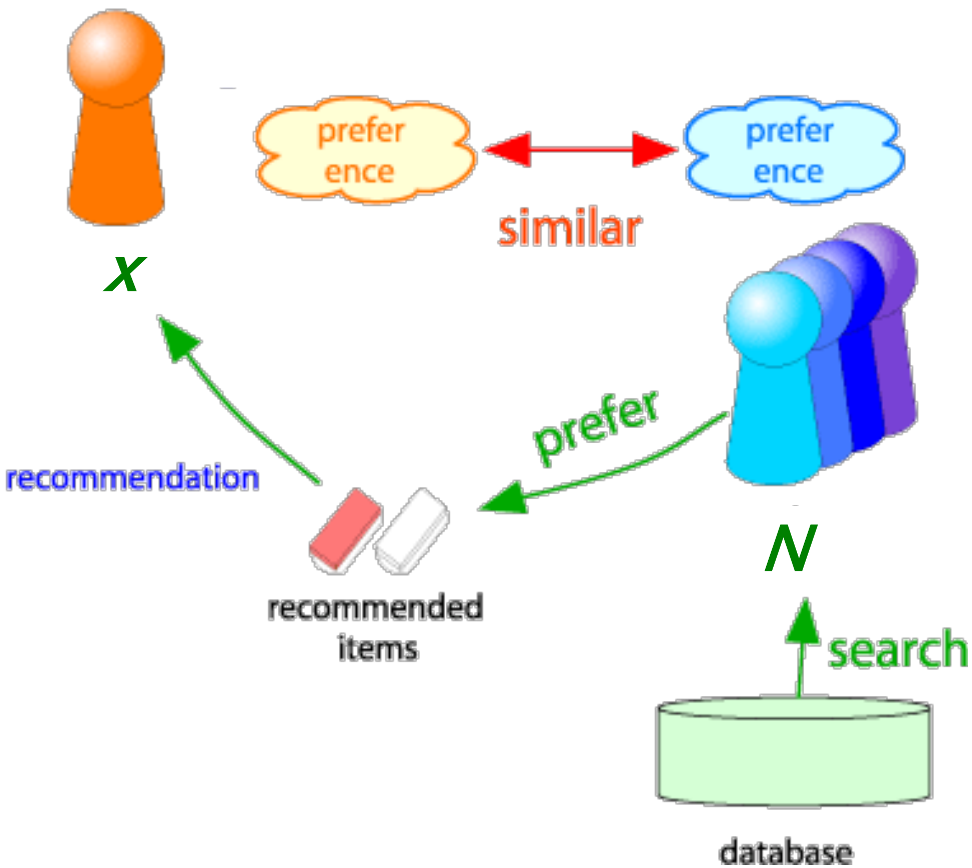
Collaborative Filtering Recommender Systems

- Give recommendations to a user based on the preferences of “similar” users
- Recommendation is dependent on other users’ historical data
- We only have user-item interactions (i.e., ratings) as the inputs



Collaborative Filtering

- Consider user ***x***
- Find set ***N*** of other users whose ratings are "**similar**" to ***x***'s ratings
- Estimate ***x***'s ratings based on ratings of users in ***N***



Example

Movie	Alice $\theta^{(1)}$	Bob $\theta^{(2)}$	Carol $\theta^{(3)}$	Dave $\theta^{(4)}$	X1	X2
Love letter $x^{(1)}$	5	5	0	0	?	?
Romancer $x^{(2)}$	5	?	?	0	?	?
Stay with me $x^{(3)}$	?	4	0	?	?	?
KungFu Panda $x^{(4)}$	0	0	5	4	?	?
FightFightFight $x^{(5)}$	0	0	5	?	?	?

- If both C and D like KungFu Panda and dislike Love Letter, then when C has rated a new movie FightFightFight as good, it will recommend the movie to D
- It learns feature itself - "Feature Learning"

Symbols

- $r(i, j) = 1$ if user j has rated movie i
- $r(i, j) = 0$ if user j has not rated movie i
- $y^{(i,j)}$: rating by user j on movie i if $r(i, j) = 1$
- n : number of features of a movie
- $\theta^{(j)} \in R^n$: parameter vector for user j
- $x^{(i)} \in R^n$: feature vector for movie i
- $m^{(j)}$: number of rated movies rated by user j
- n_u : number of users
- n_m : number of movies

Approach I: CF based on Linear Regression

- Learn parameter $x^{(1)}, x^{(2)}, \ldots, x^{(5)}$ and $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(4)}$ by solving a Linear Regression problem
- $x^{(i)} = [x_1^{(i)} x_2^{(i)}]^T, \theta^{(j)} = [\theta_1^{(j)} \theta_2^{(j)}]^T, m = 15, n = 2$
- Hypothesis function

$$h_{i,j}(x,\theta) = \left(\theta^{(j)}\right)^T x^{(i)} = \theta_1^{(j)} x_1^{(i)} + \theta_2^{(j)} x_2^{(i)}$$

- Cost Function

$$J(x,\theta) = \frac{1}{2m} \sum_{(i,j):r(i,j)=1} \left(\left(\theta^{(j)}\right)^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2m} \sum_{i=1}^{5} \sum_{k=1}^{n} \left(x_k^{(i)}\right)^2 + \frac{\lambda}{2m} \sum_{j=1}^{4} \sum_{k=1}^{n} \left(\theta_k^{(j)}\right)^2$$

Collaborative Filtering Algorithm

- Initialize $\theta^{(1)}, \theta^{(2)}, \cdots, \theta^{(n_u)}$ and $x^{(1)}, x^{(2)}, \cdots, x^{(n_m)}$ to small random values
- Minimize $J\left(x^{(1)}, x^{(2)}, \cdots, x^{(n_m)}, \theta^{(1)}, \theta^{(2)}, \cdots, \theta^{(n_u)}\right)$ using gradient decent
- For a user with parameters θ and a movie with learned features x, predict a rating of $\theta^T x$

CF Gradient $\partial J(x,\theta)/\partial\theta$ and $\partial J(x,\theta)/\partial\theta$

- Gradient for x

$$\frac{\partial J\left(x^{(i)},\theta^{(j)}\right)}{\partial x_k^{(i)}} = \frac{1}{m}\left(\sum_{j:r(i,j)=1}\left(\left(\theta^{(j)}\right)^T x^{(i)} - y^{(i,j)}\right)\theta_k^{(j)} + \lambda x_k^{(i)}\right)$$

- Gradient for θ

$$\frac{\partial J\left(x^{(i)},\theta^{(j)}\right)}{\partial \theta_k^{(j)}} = \frac{1}{m}\left(\sum_{i:r(i,j)=1}\left(\left(\theta^{(j)}\right)^T x^{(i)} - y^{(i,j)}\right)x_k^{(i)} + \lambda \theta_k^{(j)}\right)$$

Collaborative Filtering Gradient Descent Update

- Gradient Descent Update for x

$$x_k^{(i)} = x_k^{(i)} - \frac{\alpha}{m}\left(\sum_{j:r(i,j)=1} \left(\left(\theta^{(j)} \right)^T x^{(i)} - y^{(i,j)} \right) \theta_k^{(j)} + \lambda x_k^{(i)} \right)$$

- Gradient Descent Update for θ

$$\theta_k^{(j)} = \theta_k^{(j)} - \frac{\alpha}{m}\left(\sum_{i:r(i,j)=1} \left(\left(\theta^{(j)} \right)^T x^{(i)} - y^{(i,j)} \right) x_k^{(i)} + \lambda \theta_k^{(j)} \right)$$

Collaborative Filtering Optimization Objective

- Learn $\theta^{(1)}, \theta^{(2)}, \cdots, \theta^{(n_u)}$ and $x^{(1)}, x^{(2)}, \cdots, x^{(n_m)}$:

$$J\left(x^{(1)}, x^{(2)}, \cdots, x^{(n_m)}, \theta^{(1)}, \theta^{(2)}, \cdots, \theta^{(n_u)}\right)$$
$$= \frac{1}{2m} \sum_{(i,j):r(i,j)=1} \left(\left(\theta^{(j)} \right)^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2m} \sum_{j=1}^{n_u} \sum_{k=1}^{n} \left(\theta_k^{(j)} \right)^2 + \frac{\lambda}{2m} \sum_{i=1}^{n_m} \sum_{k=1}^{n} \left(x_k^{(i)} \right)^2$$

Movie	$x_1^{(i)}$	$x_2^{(i)}$	User 1 $y^{(i,1)}$	…	User n_u $y^{(i,n_u)}$
Love letter $x^{(1)}$	?	?	5		0
Romancer $x^{(2)}$	?	?	5		0
Stay with me $x^{(3)}$	?	?	**?**		?
KungFu Panda $x^{(4)}$	?	?	0		4
FightFightFight $x^{(5)}$	?	?	0		?

Content-based Optimization Objective

- Given $x^{(1)}, x^{(2)}, \cdots, x^{(n_m)}$, to learn $\theta^{(1)}, \theta^{(2)}, \cdots, \theta^{(n_u)}$:

$$J\left(\theta^{(1)}, \theta^{(2)}, \cdots, \theta^{(n_u)}\right) = \frac{1}{2m} \sum_{(i,j):r(i,j)=1} \left(\left(\theta^{(j)} \right)^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2m} \sum_{j=1}^{n_u} \sum_{k=1}^{n} \left(\theta_k^{(j)} \right)^2$$

Movie	$x_1^{(i)}$	$x_2^{(i)}$	User 1 $y^{(i,1)}$	…	User n_u $y^{(i,n_u)}$
Love letter $x^{(1)}$	0.9	0	5		0
Romancer $x^{(2)}$	1	0	5		0
Stay with me $x^{(3)}$	.89	0	**?**		?
KungFu Panda $x^{(4)}$	0.2	0.9	0		4
FightFightFight $x^{(5)}$	0.1	1	0		?

Approach II: User-user collaborative filtering

From similarity metric to recommendations:

- Let $\boldsymbol{r_x}$ be the vector of user $\boldsymbol{x}$'s ratings
- Let $\boldsymbol{N}$ be the set of $\boldsymbol{k}$ users most similar to $\boldsymbol{x}$ who have rated item $\boldsymbol{i}$
- **Prediction for item *i* of *user x*:**
  - $r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$
  - $r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$

Shorthand: $s_{xy} = sim(x, y)$

Step 1: Finding "Similar" Users

- Let $\boldsymbol{r_x}$ be the vector of user $\boldsymbol{x}$'s ratings
- **Jaccard similarity measure**
  - **Problem:** Ignores the value of the rating
- **Cosine similarity measure**
  - sim($\boldsymbol{x}$, $\boldsymbol{y}$) = cos($\boldsymbol{r_x}$, $\boldsymbol{r_y}$) = $\frac{r_x \cdot r_y}{||r_x|| \cdot ||r_y||}$
  - **Problem:** Treats missing ratings as "negative"
- **Pearson correlation coefficient**
  - S_{xy} = items rated by both users $\boldsymbol{x}$ and $\boldsymbol{y}$
  - S_x = items rated by user $\boldsymbol{x}$
  - S_y = items rated by both user $\boldsymbol{y}$

$\boldsymbol{r_x}$ = [*, _, _, *, ***]
$\boldsymbol{r_y}$ = [*, _, **, **, _]

r_x, r_y as sets:
$\boldsymbol{r_x}$ = {1, 4, 5}
$\boldsymbol{r_y}$ = {1, 3, 4}

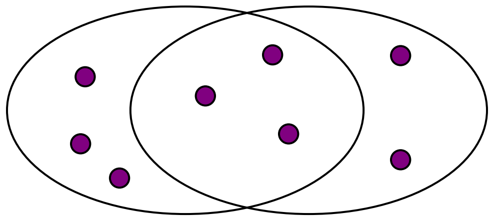
r_x, r_y as points:
$\boldsymbol{r_x}$ = {1, 0, 0, 1, 3}
$\boldsymbol{r_y}$ = {1, 0, 2, 2, 0}

$$sim(x,y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})(r_{ys} - \overline{r_y})}{\sqrt{\sum_{s \in S_x} (r_{xs} - \overline{r_x})^2} \sqrt{\sum_{s \in S_y} (r_{ys} - \overline{r_y})^2}}$$

$\overline{r_x}$, $\overline{r_y}$ … avg. rating of **x**, **y**

Jaccard similarity

- The **Jaccard similarity** of two **sets** is the size of their intersection divided by the size of their union:
  $sim(C_1, C_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$
- **Jaccard distance:** $d(C_1, C_2) = 1 - |C_1 \cap C_2| / |C_1 \cup C_2|$



3 in intersection
8 in union
Jaccard similarity= 3/8
Jaccard distance = 5/8

- **Cosine similarity** $$sim(x,y) = \frac{\sum_i r_{xi} \cdot r_{yi}}{\sqrt{\sum_i r_{xi}^2} \cdot \sqrt{\sum_i r_{yi}^2}}$$

An Example

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- **Intuitively we want: sim(*A*, *B*) > sim(*A*, *C*)**
- **Jaccard similarity:** 1/5 < 2/4
- **Cosine similarity:** $cossim(x, y) = \frac{\sum_i r_{xi} \cdot r_{yi}}{\sqrt{\sum_i r_{xi}^2} \cdot \sqrt{\sum_i r_{yi}^2}}$
  - $\frac{20}{\left(\sqrt{(16+25+1)} \cdot \sqrt{(25+25+16)}\right)}$ > $\frac{(10+4)}{\left(\sqrt{(16+25+1)} \cdot \sqrt{(4+16+25)}\right)}$
  - 0.380 > 0.322
  - Considers missing ratings as "negative"

An Example

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

- **Pearson correlation coefficient**
  - S_{xy} = items rated by both users $\boldsymbol{x}$ and $\boldsymbol{y}$

$$\boldsymbol{sim(x,y)} = \frac{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})(r_{ys} - \overline{r_y})}{\sqrt{\sum_{s \in S_x} (r_{xs} - \overline{r_x})^2} \sqrt{\sum_{s \in S_y} (r_{ys} - \overline{r_y})^2}}$$

$\overline{\mathbf{r}}_{\mathbf{x}}$, $\overline{\mathbf{r}}_{\mathbf{y}}$ … avg. rating of **x**, **y**

  - **sim(*A*, *B*)** = 0.09 > **sim(*A*, *C*)** = -0.559
    - A: [2/3, , , 5/3, -7/3, ,]
    - B: [1/3, 1/3, -2/3, , , ,]
    - C: [, , , -5/3, 1/3, 4/3,]
    - (2/9)/(sqrt(4/9+25/9+49/9) * sqrt(1/9+1/9+4/9))
    - (-25/9-7/9)/(sqrt(4/9+25/9+49/9) * sqrt(25/9+1/9+16/9))

Step 2: Rating Predictions

From similarity metric to recommendations:

- Let $\boldsymbol{r_x}$ be the vector of user $\boldsymbol{x}$'s ratings
- Let $\boldsymbol{N}$ be the set of $\boldsymbol{k}$ users most similar to $\boldsymbol{x}$ who have rated item $\boldsymbol{i}$
- **Prediction for item *i* of *user x*:**
  - $r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$
  - $r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$
  - Other options?

Shorthand: $s_{xy} = sim(x, y)$

Item-Item Collaborative Filtering

- **So far: User-user collaborative filtering**
- **Another view: Item-item**
  - For item ***i***, find other similar items
  - Estimate rating for item ***i*** based on ratings for similar items
  - Can use same similarity metrics and prediction functions as in user-user model

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

s_{ij}… similarity of items ***i*** and ***j***
r_{xj}…rating of user ***x*** on item ***j***
N(i;x)… set items rated by ***x*** similar to ***i***

Exercise

- Consider an Item-Item Collaborative Filtering recommendation system. For a given item A, the system has identified the top 3 most similar items to A, namely B, C, and D. The cosine similarities between item A and the corresponding items are 0.4, 0.5, and 0.6, respectively.
- Assuming that the system has collected user ratings for items B, C, and D from a specific user x, and the ratings are as follows:
- Rating for item B: 5.0
- Rating for item C: 4.0
- Rating for item D: 3.0
- Compute the predicted rating for user x on item A using the weighted average
- $(0.4*5+0.5*4+0.6*3)/(0.4+0.5+0.6) \approx 3.87$

Item-Item CF (|N|=2)

users

movies	1	2	3	4	**5**	6	7	8	9	10	11	12
1	1		3		?	5			5		4	
2			5	4			4			2	1	3
3	2	4		1	2		3		4	3	5	
4		2	4		5			4			2	
5			4	3	4	2					2	5
6	1		3		3			2			4	

- unknown rating

- rating between 1 to 5

- estimate rating of movie **1** by user **5**

Item-Item CF (|N|=2)

users

movies	1	2	3	4	**5**	6	7	8	9	10	11	12	sim(1,m)
1	1		3		?	5			5		4		1.00
2			5	4			4			2	1	3	-0.18
3	2	4		1	2		3		4	3	5		**0.41**
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
6	1		3		3			2			4		**0.59**

Neighbor selection:
Identify movies similar to movie **1**, **rated by user 5**

Here we use Pearson correlation as similarity:

1) Subtract mean rating m_i from each movie i

$m_1 = (1+3+5+5+4)/5 = \mathbf{3.6}$

row 1: *[-2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4, 0, 0.4, 0]*

2) Compute similarities between rows

Details

- The mean of movie 1 is (1+3+5+5+4)/5=3.6, so it becomes row 1: [-2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4, 0, 0.4, 0]
- The mean of movie 3 is (2+4+1+2+3+4+3+5)/8 = 3, so it becomes row 3: [-1, 1, 0, -2, -1, 0, 0, 0, 1, 0, 2, 0]
- The Pearson correlation coefficient is (2.6+1.4+0.4*2)/(sqrt(2.6^2 + 0.6^2 + 1.4^2 + 1.4^2 + 0.4^2) * sqrt(1+1+4+1+1+4)) = 4.8/(3.346*3.464) = 0.41.

Item-Item CF (|N|=2)

users

movies	1	2	3	4	5	6	7	8	9	10	11	12	sim(1,m)
1	1		3		**2.6**	5			5		4		1.00
2			5	4			4			2	1	3	-0.18
<u>3</u>	2	4		1	2		3		4	3	5		<u>0.41</u>
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
<u>6</u>	1		3		3			2			4		<u>0.59</u>

Predict by taking weighted average:

$r_{1.5}$ = **(0.41*2 + 0.59*3) / (0.41+0.59) = 2.6**

$$r_{ix} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{jx}}{\sum s_{ij}}$$

CF: Common Practice

- Define **similarity** s_{ij} of items i and j
- Select k nearest neighbors $N(i; x)$
  - Items most similar to i, that were rated by x
- Estimate rating r_{xi} as the weighted average:

Before:

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

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

baseline estimate for r_{xi}

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

- μ = overall mean movie rating
- b_x = rating deviation of user x
  = *(avg. rating of user* x*)* $- \mu$
- b_i = rating deviation of movie i
  = *(avg. rating of movie* i*)* $- \mu$

Item-Item vs. User-User

- **In practice, it has been observed that <u>item-item</u> often works better than user-user**

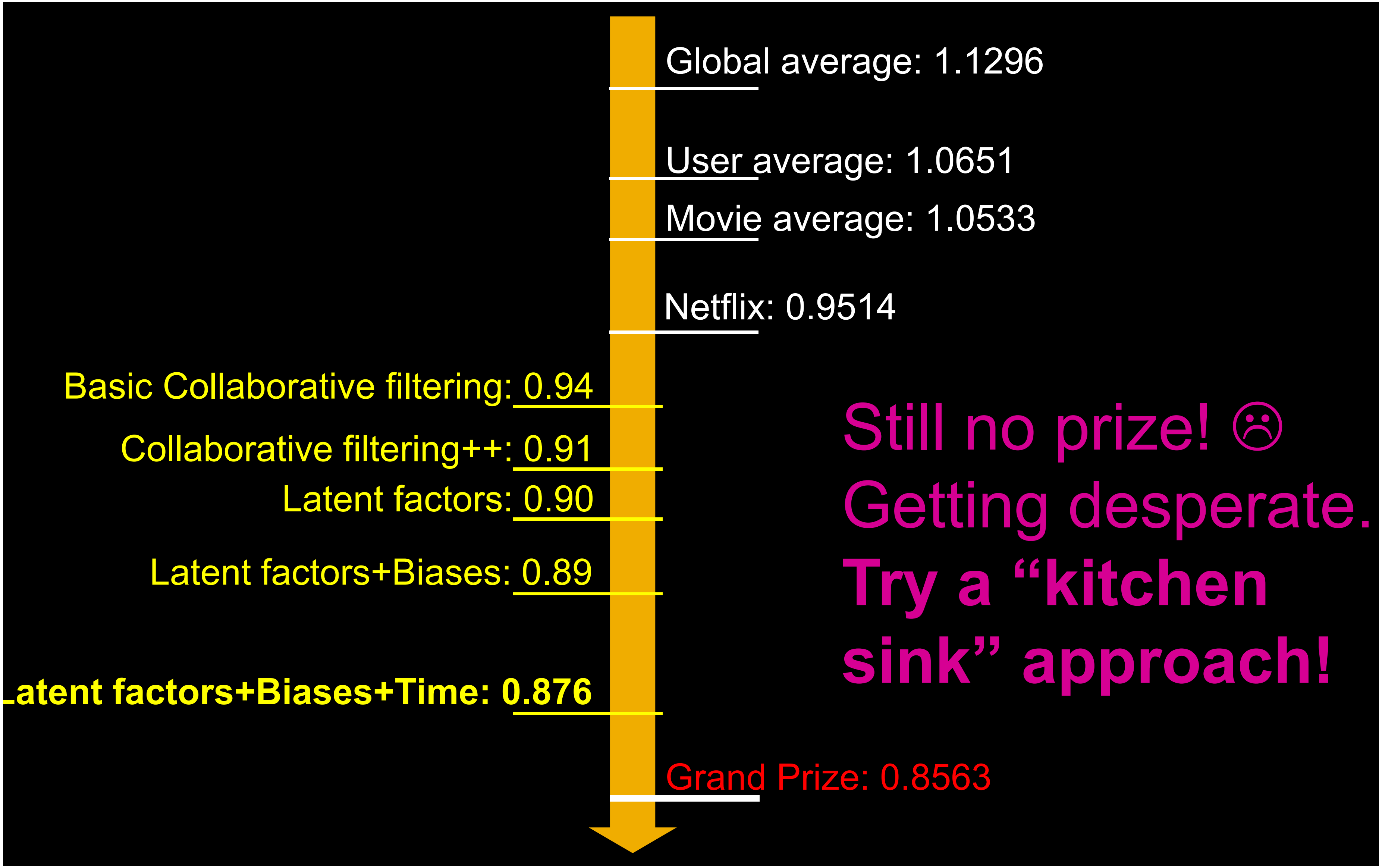
- **Why?** Items are simpler, users have multiple tastes

- For example, it is easier to discover items that are similar because they belong to the same genre, than it is to detect that two users are similar because they prefer one genre in common, while each also likes some genres that the other doesn't care for.

The Netflix Prize (2 Oct 2006 – 21 Sept 2009)

- 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 squared error (RMSE)
  - Netflix Cinematch system RMSE: 0.9514
- Competition
  - 2700+ teams
  - $1 million grand prize for 10% improvement over Netflix

Million $ Awarded Sept 21st 2009



https://www.kaggle.com/netflix-inc/netflix-prize-data

NETFLIX

Netflix Prize

COMPLETED

Home | Rules | Leaderboard | Update | Download

Leaderboard

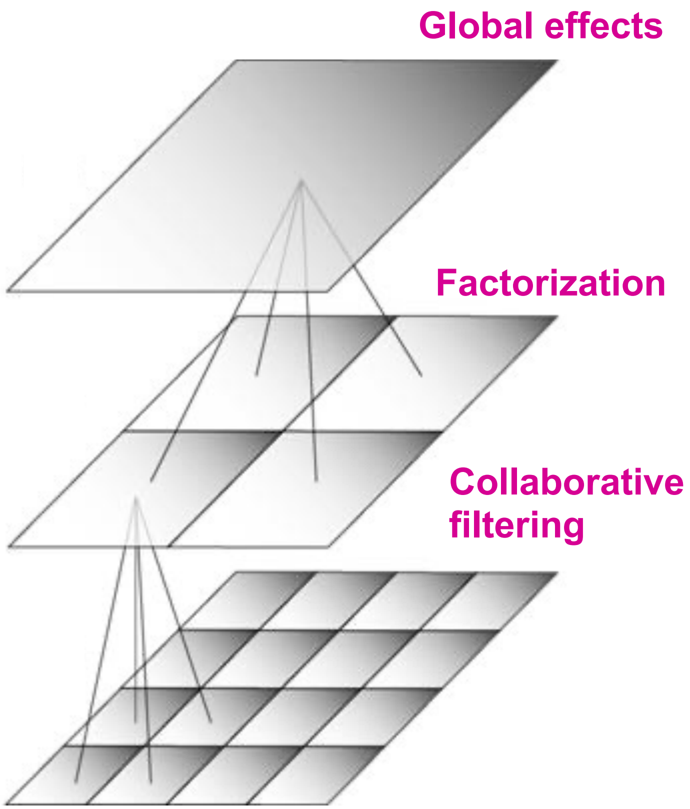
Showing Test Score. Click here to show quiz score

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	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				

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



Problems with Error Measures

- **Narrow focus on accuracy sometimes misses the point**
  - Prediction Diversity
  - Prediction Context
  - Order of predictions
- **In practice, we care only to predict high ratings:**
  - RMSE might penalize a method that does well for high ratings and badly for others

Pros/Cons of Collaborative Filtering

- **+ Works for any kind of item**
  - No feature selection needed
- **- Cold Start:**
  - Need enough users in the system to find a match
- **- Sparsity:**
  - The user/ratings matrix is sparse
  - Hard to find users that have rated the same items
- **- First rater:**
  - Cannot recommend an item that has not been previously rated
  - New items, Esoteric items
- **- Popularity bias:**
  - Cannot recommend items to someone with unique taste
  - Tends to recommend popular items