

COMP4434 Big Data Analytics

Lecture 9

Recommender Systems

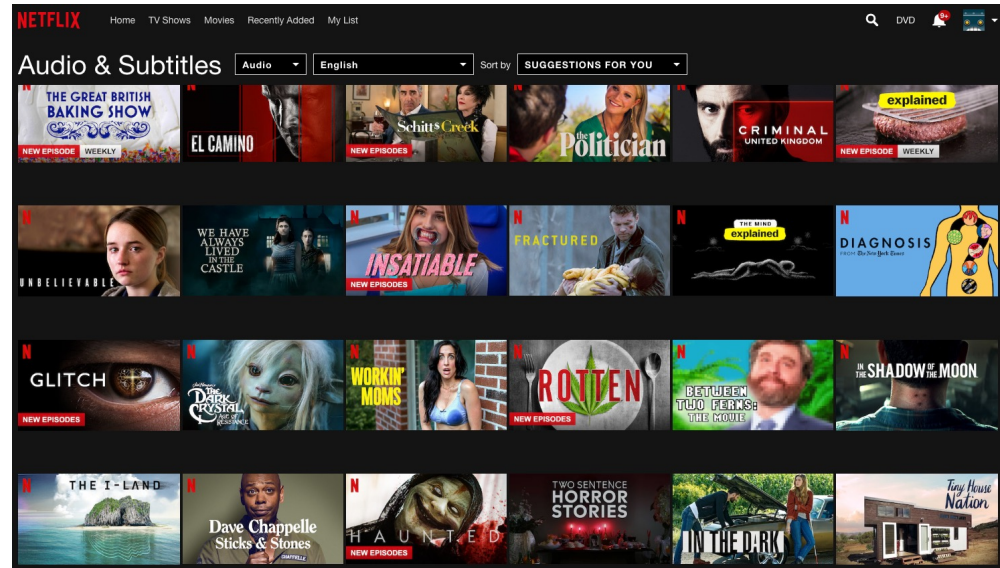
HUANG Xiao

xiaohuang@comp.polyu.edu.hk



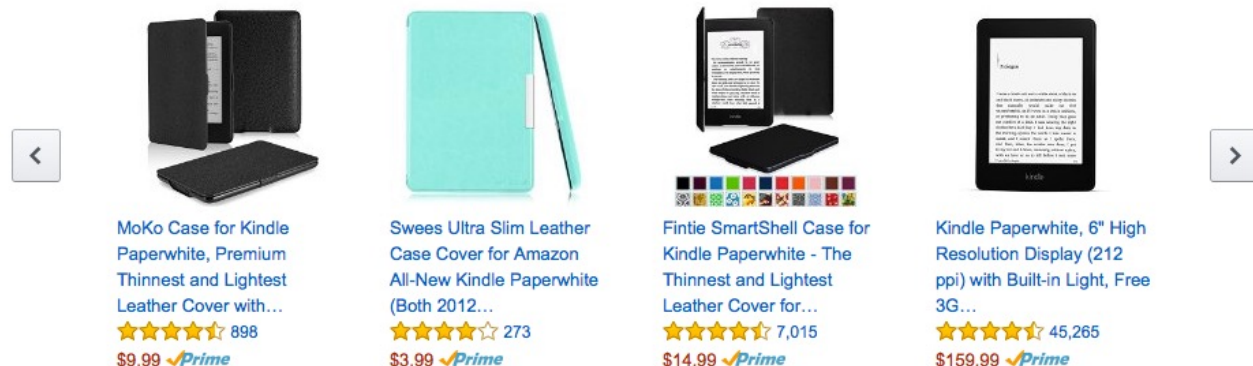
Recommender System Examples

- Amazon, YouTube, Netflix, ...
- How to improve users' satisfaction?
- What item for what people?
 - E.g., Recommend movies based on the predictions of user's movie ratings



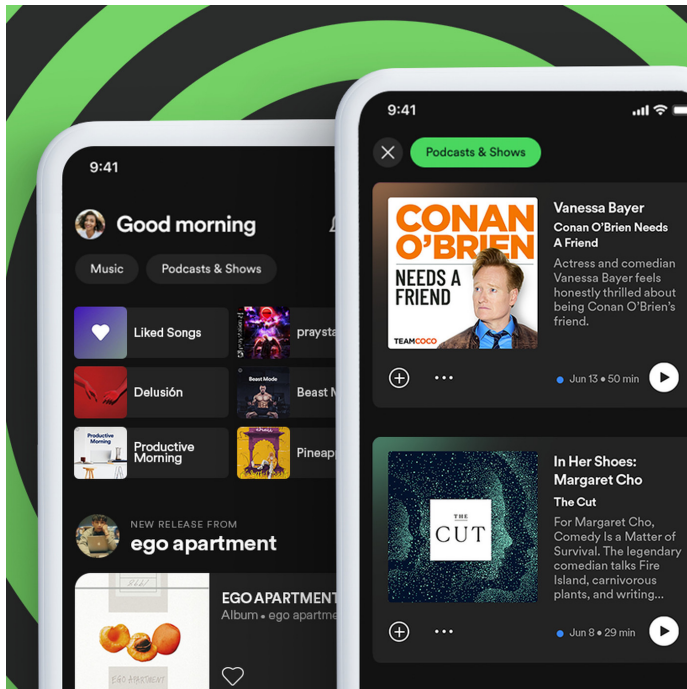
Recommended for You Based on Kindle Paperwhite, 6" High Resolution Display w...

Page 1 of 5



More Recommender System Examples

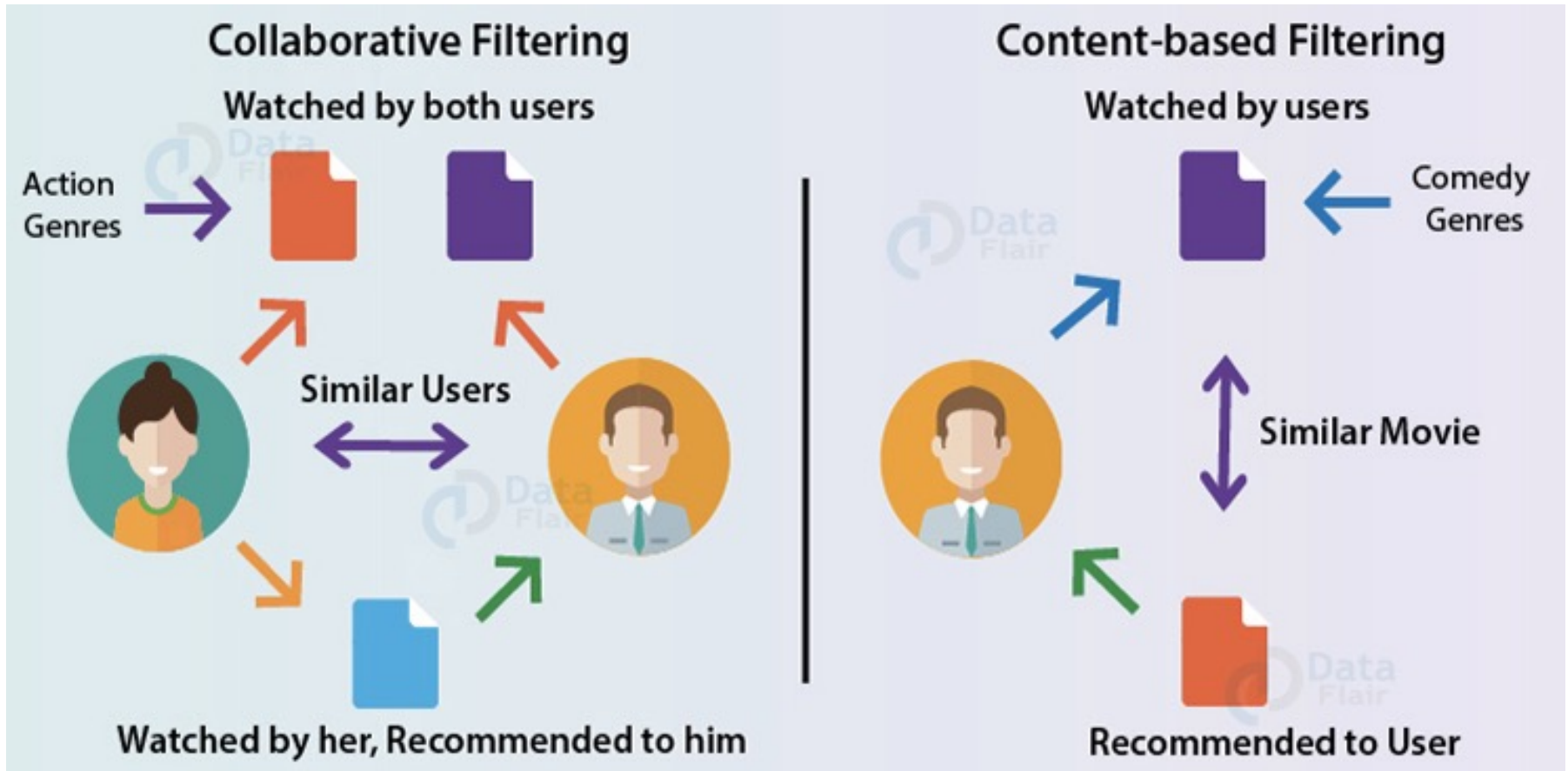
- News feed
- Music feed
- Twitter feed



Recommender System Types

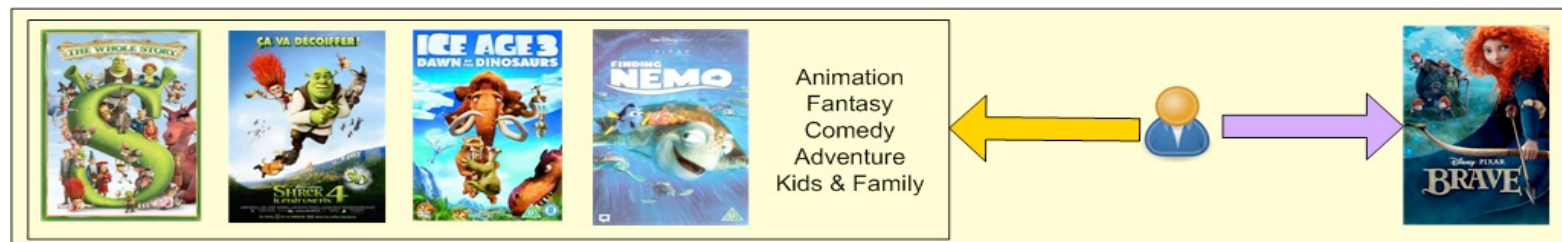
- **Content Based (CB):** recommendations are based on the assumption that if in the past a user liked a set of items with particular features, she/he will likely go for the items with similar characteristics.
- **Collaborative Filtering (CF):** recommendations are based on the assumption that users having similar history are more likely to have similar tastes/needs.

Recommender System Types

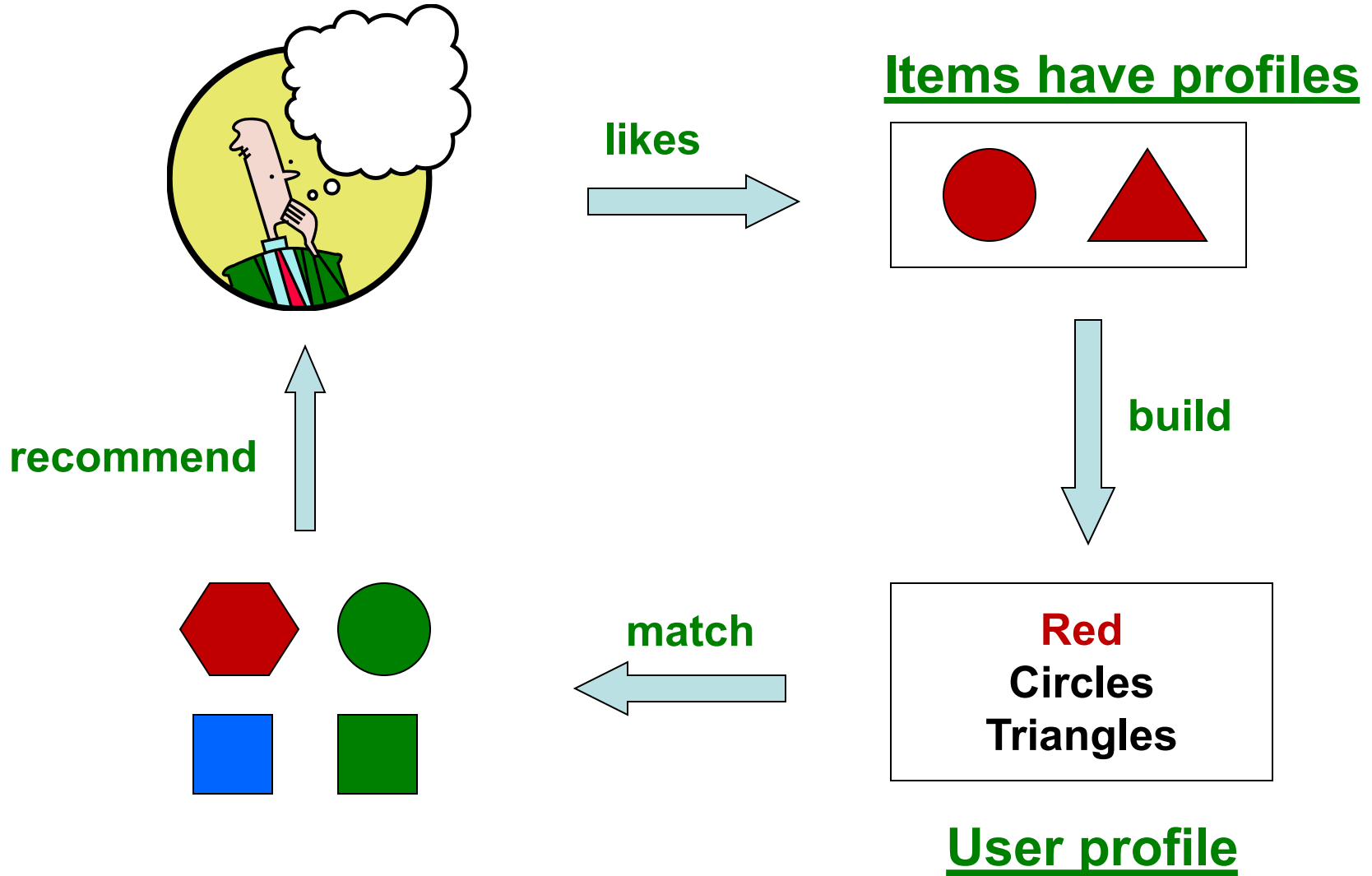


Content-based Recommender Systems

- Give recommendations to a user based on items with “similar” content in user’s profile
- Recommendation is only dependent on particular user’s historical data
- Besides user-item interactions (i.e., ratings), we also have the item feature vectors as the inputs



Plan of Action



Example

Movie	Alice	Bob	Carol	Dave	X1 (Romance)	X2 (KungFu)
Love letter	5	5	0	0	0.9	0
Romancer	5	?	?	0	1	0
Stay with me	?	4	0	?	0.89	0
KungFu Panda	0	0	5	4	0.2	0.9
FightFightFight	0	0	5	?	0.1	1

? not rated yet

- For each item, create an item profile (a set of features)
 - E.g., each movie has genre, author, title, actor, director,...

Symbols: Table

$n_m = 5$: number of movies

Movie	Alice	Bob	Carol	Dave	X1 (Romance)	X2 (KungFu)
Love letter	5	5	0	0	0.9	0
Romancer	5	?	?	0	1	0
Stay with me	?	4	0	?	0.89	0
KungFu Panda	0	0	5	4	0.2	0.9
FightFightFight	0	0	5	?	0.1	1

$n_u = 4$: number of users

$n = 2$: number of movie features

Symbols: Rating

$$m^{(1)} = 4$$

$$r(1,2) = 1, y^{(1,2)} = 5$$

Movie	Alice	Bob	Carol	Dave
Love letter	5	5	0	0
Romancer	5	?	?	0
Stay with me	?	4	0	?
KungFu Panda	0	0	5	4
FightFightFight	0	0	5	?

$$r(3,4) = 0$$

$r(i,j) = 1$ if user j has rated movie i ; $y^{(i,j)}$ is the rating
 $m^{(j)}$: number of rated movies rated by user j

RMSE

- Compare predictions with known ratings
- My system predicted you would rate
 - The Shawshank Redemption as 4.3 stars
 - In reality, you gave it 5 stars
 - The Matrix with 3.9 stars
 - In reality, you gave it 4 stars
- $\text{RMSE} = \text{sqrt}(1/2 * (4.3 - 5)^2 + (3.9 - 4)^2)$

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{j=1}^n (y_j - \hat{y}_j)^2}$$

How to solve the problem for Alice?

Movie	Alice	Bob	Carol	Dave	X1 (Romance)	X2 (KungFu)
Love letter	5	5	0	0	0.9	0
Romancer	5	?	?	0	1	0
Stay with me	?	4	0	?	0.89	0
KungFu Panda	0	0	5	4	0.2	0.9
FightFightFight	0	0	5	?	0.1	1

Hypothesis For Alice

- Learn parameter $\theta^{(1)} = [\theta_0^{(1)} \theta_1^{(1)} \theta_2^{(1)}]^T$ by solving a **Linear Regression** problem
- Hypothesis function

$$h_{\theta^{(1)}}(x) = \left(\theta^{(1)} \right)^T x = \theta_0^{(1)} x_0 + \theta_1^{(1)} x_1 + \theta_2^{(1)} x_2$$

- Cost Function

$$\begin{aligned} J(\theta^{(1)}) &= \frac{1}{2m^{(1)}} \sum_{i:r(i,1)=1} \left(\left(\theta^{(1)} \right)^T x^{(i)} - y^{(i,1)} \right)^2 + \frac{\lambda}{2m^{(1)}} \sum_{k=1}^n \left(\theta_k^{(1)} \right)^2 \\ &= \frac{1}{2m^{(1)}} \sum_{i:r(i,1)=1} \left(\sum_{k=0}^n \left(\theta_k^{(1)} x_k^{(i)} \right) - y^{(i,1)} \right)^2 + \frac{\lambda}{2m^{(1)}} \sum_{k=1}^n \left(\theta_k^{(1)} \right)^2 \end{aligned}$$

Iteration ...

```
class RegularizedLinearRegressionUsingGD:

    def __init__(self, eta=0.01, r= 0.009, n_iterations=10000):
        self.eta = eta
        self.r = r
        self.n_iterations = n_iterations

    def fit(self, x, y):
        self.cost_ = []
        self.w_ = np.array([0.0,0.0,0.0])
        m = x.shape[0]

        for _ in range(self.n_iterations):
            y_pred = np.dot(x, self.w_)
            residuals = y_pred - y


            gradient_vector_w_0 = np.sum(residuals) / m * self.eta
            gradient_vector_w_1 = (np.dot(x[:,1], residuals) + self.r * self.w_[1]) / m * self.eta
            gradient_vector_w_2 = (np.dot(x[:,2], residuals) + self.r * self.w_[2]) / m * self.eta
            self.w_[0] -= gradient_vector_w_0
            self.w_[1] -= gradient_vector_w_1
            self.w_[2] -= gradient_vector_w_2

            cost = np.sum((residuals ** 2)) / (2 * m) + self.r * ((self.w_[1] ** 2) + (self.w_[2] ** 2)) / (2 * m)
            self.cost_.append(cost)

        print('iter {}: w = {} \t cost ={}'.format(_, self.w_, cost))
        return self

    def predict(self, x):
        return np.dot(x, self.w_)
```

Gradient descent update



Alice's Model

```
iter 9999: w = [ 1.95158962  3.16948852 -2.52109055]      cost =0.045574863024676435  
predicted response: [ 4.80412929  5.12107814  0.31650583 -0.25255208]  
Root mean squared error:  0.05424593597454509  
R2 score:  0.9913206502440728
```

$$\theta^{(1)} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.025 \\ 0.0375 \\ 0 \end{bmatrix} = \dots = \begin{bmatrix} 1.95 \\ 3.17 \\ -2.52 \end{bmatrix}$$

$$h_{\theta^{(1)}}(x) = (\theta^{(1)})^T x = 1.95 + 3.17x_1 - 2.52x_2$$

Rating Prediction for Alice

Movie		X1 (Romance)	X2 (KungFu)	Alice $y^{(i,1)}$
Love letter	$x^{(1)}$	0.9	0	5
Romancer	$x^{(2)}$	1	0	5
Stay with me	$x^{(3)}$	0.89	0	4.77
KungFu Panda	$x^{(4)}$	0.2	0.9	0
FightFightFight	$x^{(5)}$	0.1	1	0

- Predict user j rating movie i with $(\theta^{(j)})^T x^{(i)}$
- E.g., $(\theta^{(1)})^T x^{(3)} = [1.95 \quad 3.17 \quad -2.52] \begin{bmatrix} 1 \\ 0.89 \\ 0 \end{bmatrix} = 4.77$

General Problem

Movie		Alice $\theta^{(1)}$	Bob $\theta^{(2)}$	Carol $\theta^{(3)}$	Dave $\theta^{(4)}$	X1 (Romance)	X2 (KungFu)
Love letter	$x^{(1)}$	5	5	0	0	0.9	0
Romancer	$x^{(2)}$	5	?	?	0	1	0
Stay with me	$x^{(3)}$?	4	0	?	0.89	0
KungFu Panda	$x^{(4)}$	0	0	5	4	0.2	0.9
FightFightFight	$x^{(5)}$	0	0	5	?	0.1	1

- For each user j , learn parameter $\theta^{(j)} \in R^{n+1}$

Problem Formulation

- $r(i, j) = 1$ 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
- $\theta^{(j)} \in R^{n+1}$: parameter vector for user j
- $x^{(i)} \in R^{n+1}$: 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

CB Optimization Objective

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

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

Movie	$x_1^{(i)}$ (Romance)	$x_2^{(i)}$ (KungFu)	User j $y^{(i,j)}$
Love letter $x^{(1)}$	0.9	0	5
Romancer $x^{(2)}$	1	0	5
Stay with me $x^{(3)}$	0.89	0	?
KungFu Panda $x^{(4)}$	0.2	0.9	0
FightFightFight $x^{(5)}$	0.1	1	0

$$x^{(i)} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0.9 & 1 & 0.89 & 0.2 & 0.1 \\ 0 & 0 & 0 & 0.9 & 1 \end{bmatrix}$$

$$\theta^{(j)} = \begin{bmatrix} \theta_0^{(j)} \\ \theta_1^{(j)} \\ \theta_2^{(j)} \end{bmatrix} = ?$$

$$\theta^{(j)} \in R^{n+1}; x^{(i)} \in R^{n+1}, x_0^{(i)} = 1$$

Optimization Objectives

- To learn $\theta^{(j)}$ (parameter for user j):

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

- To learn $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(n_u)}$

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

CB Gradient Decent Update

- $k = 0$

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

- $k \neq 0$

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

Pros: Content-based Approach

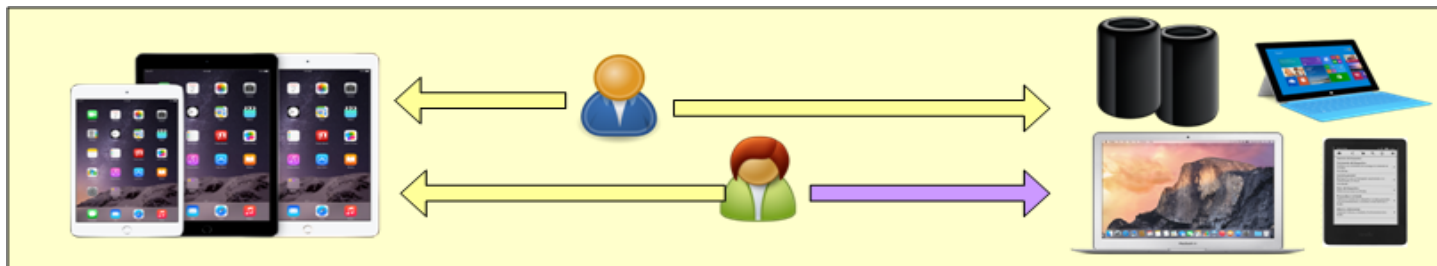
- **+: No need for data on other users**
- **+: Able to recommend to users with unique tastes**
- **+: Able to recommend new & unpopular items**
 - No cold-start item problems
- **+: Able to provide explanations**
 - Can provide explanations of recommended items by listing content-features that caused an item to be recommended

Cons: Content-based Approach

- —: Finding the appropriate features is hard
 - E.g., images, movies, music
- —: Recommendations for new users
 - How to build a user profile?
- —: Overspecialization
 - Never recommends items outside user's content profile
 - People might have multiple interests
 - Unable to exploit quality judgments of other users

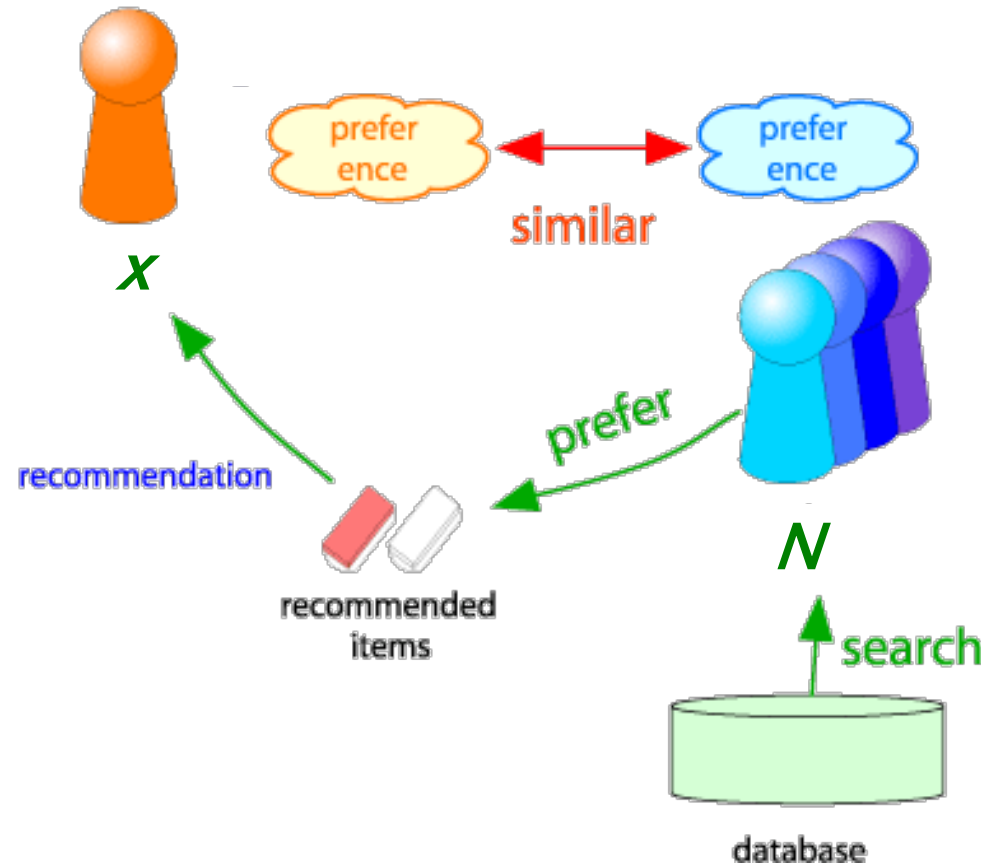
Collaborative Filtering Recommender Systems

- 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 $x^{(1)}, x^{(2)}, \dots, x^{(5)}$ and $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(4)}$ 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

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

- Cost Function

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

Collaborative Filtering Algorithm

- Initialize $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(n_u)}$ and $x^{(1)}, x^{(2)}, \dots, x^{(n_m)}$ to small random values
- Minimize $J(x^{(1)}, x^{(2)}, \dots, x^{(n_m)}, \theta^{(1)}, \theta^{(2)}, \dots, \theta^{(n_u)})$ 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

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

- Gradient for θ

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

CF Gradient Descent Update

- Gradient Descent Update for x

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

- Gradient Descent Update for θ

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

CF Optimization Objective

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

$$J(x^{(1)}, x^{(2)}, \dots, x^{(n_m)}, \theta^{(1)}, \theta^{(2)}, \dots, \theta^{(n_u)})$$

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

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		?

CB Optimization Objective

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

$$J(\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(n_u)})$$

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

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: Finding “Similar” Users

- Let r_x be the vector of user x 's ratings
- Jaccard similarity measure**
 - Problem:** Ignores the value of the rating

$$r_x = [* , _ , _ , * , ***]$$

$$r_y = [* , _ , ** , ** , _]$$

- Cosine similarity measure**

- $$\text{sim}(x, y) = \cos(r_x, r_y) = \frac{r_x \cdot r_y}{||r_x|| \cdot ||r_y||}$$

r_x, r_y as sets:

$$r_x = \{1, 4, 5\}$$

$$r_y = \{1, 3, 4\}$$

- Problem:** Treats missing ratings as “negative”

- Pearson correlation coefficient**

r_x, r_y as points:

$$r_x = \{1, 0, 0, 1, 3\}$$

$$r_y = \{1, 0, 2, 2, 0\}$$

- S_{xy} = items rated by both users x and y
 - S_x = items rated by user x
 - S_y = items rated by both user y

$$\text{sim}(x, y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \bar{r}_x)(r_{ys} - \bar{r}_y)}{\sqrt{\sum_{s \in S_x} (r_{xs} - \bar{r}_x)^2} \sqrt{\sum_{s \in S_y} (r_{ys} - \bar{r}_y)^2}}$$

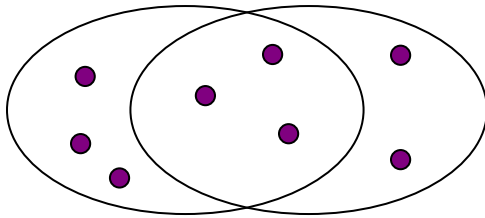
$\bar{r}_x, \bar{r}_y \dots$ avg.
rating of x, y

Jaccard similarity

- The **Jaccard similarity** of two **sets** is the size of their intersection divided by the size of their union:

$$\text{sim}(\mathbf{C}_1, \mathbf{C}_2) = |\mathbf{C}_1 \cap \mathbf{C}_2| / |\mathbf{C}_1 \cup \mathbf{C}_2|$$

- **Jaccard distance:** $d(\mathbf{C}_1, \mathbf{C}_2) = 1 - |\mathbf{C}_1 \cap \mathbf{C}_2| / |\mathbf{C}_1 \cup \mathbf{C}_2|$



3 in intersection

8 in union

Jaccard similarity = 3/8

Jaccard distance = 5/8

- **Cosine similarity**

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

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: $\text{sim}(A, B) > \text{sim}(A, C)$

- Jaccard similarity: $1/5 < 2/4$

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

$$\frac{20}{\left(\sqrt{(16 + 25 + 1)} \cdot \sqrt{(25 + 25 + 16)}\right)} > \frac{(10 + 4)}{\left(\sqrt{(16 + 25 + 1)} \cdot \sqrt{(4 + 16 + 25)}\right)}$$

- $0.380 > 0.322$

- 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 x and y

$$sim(x, y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \bar{r}_x)(r_{ys} - \bar{r}_y)}{\sqrt{\sum_{s \in S_x} (r_{xs} - \bar{r}_x)^2} \sqrt{\sum_{s \in S_y} (r_{ys} - \bar{r}_y)^2}}$$

$\bar{r}_x, \bar{r}_y \dots$ 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,]

- $(2/9)/(\sqrt{4/9+25/9+49/9} * \sqrt{1/9+1/9+4/9})$

- $(-25/9-7/9)/(\sqrt{4/9+25/9+49/9} * \sqrt{25/9+1/9+16/9})$

Rating Predictions

From similarity metric to recommendations:

- Let r_x be the vector of user x 's ratings
- Let N be the set of k users most similar to x who have rated item 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: $s_{xy} = \text{sim}(x, y)$

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 u on item j
 $N(i;x)$... set items rated by x similar to i

Exercise

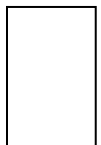
- Consider an Item-Item Collaborative Filtering recommendation system. For a given item A, the system has identified the top 3 most similar items to A, namely B, C, and D. The cosine similarities between item A and the corresponding items are 0.4, 0.5, and 0.6, respectively.
- Assuming that the system 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	$\text{sim}(1,m)$
1	1		3		?	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>

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

$$m_1 = (1+3+5+5+4)/5 = 3.6$$

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
 $(2.6+1.4+0.4*2)/(\text{sqrt}(2.6^2 + 0.6^2 + 1.4^2 + 1.4^2 + 0.4^2) * \text{sqrt}(1+1+4+1+1+4)) = 4.8/(3.346*3.464) = 0.41.$

Item-Item CF ($|N|=2$)

		users												
		1	2	3	4	5	6	7	8	9	10	11	12	
movies	1	1		3		2.6	5			5		4		sim(1,m) 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:

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

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

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:

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

baseline estimate for r_{xi}

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

Before:

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

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

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





Netflix Prize

COMPLETED

[Home](#) [Rules](#) [Leaderboard](#) [Update](#) [Download](#)

Leaderboard

Showing Test Score. [Click here to show quiz score](#)Display top 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	0.8562	9.98	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

Item-Item vs. User-User

- In practice, it has been observed that item-item 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.

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