Lec02 Neighborhood-based Collaborative Filtering

Contents

---

1. Collaborative Filtering Basics

Wisdom of Crowds

- The collected opinion of a group of individuals surpasses that of a single expert.

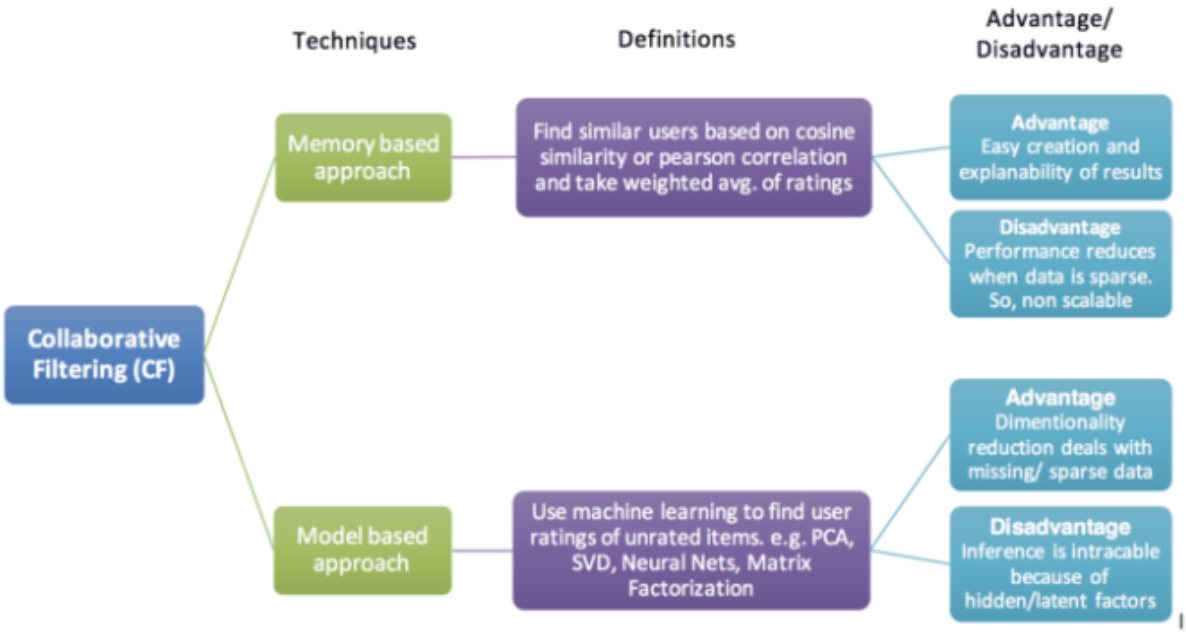
What is Collaborative Filtering (CF) ?

- Use the wisdom of crowds for recommendations.

핵심 : **많은 사용자들** 로 부터 얻은 취향 정보를 활용!!

- The most prominent approach for recommender systems
  - Various algorithms and their variations exist.
  - It is widely used by large, commercial e-commerce sites.
  - It is applicable in many domains, e.g., books, movies, and music.

협업 필터링의 종류



1. Memory based Approach(Neightborhood -based)

- 유사한 사용자(user)나 아이템(item)을 사용
- 특징 : 최적화 방법이나, 매개변수를 학습하지 않음. 단순한 산술 연산만 사용
- 방법 : Cosine Similarity나 Pearson Correlation을 사용함(KNN 방법에서 유사도 측정이 왼쪽과 같은 것들을 사용함)
- 장점
  - 쉽게 만들 수 있음
  - 결과의 설명력이 좋음
  - 도메인에 의존적이지 않음
- 단점

    - 데이터가 축적 X or Sparse한 경우 성능이 낮음
    - Not Scalability

2. Model-based Approach

- 기계학습을 통해 추천
- 특징 : 최적화 방법이나, 매개변수를 학습
- 방법 : 행렬 분해(Matrix Factorization), SVD, NN
- 장점
    - Sparse한 데이터도 처리 가능
- 단점
    - 결과의 설명력이 낮음

2주차는 memory기반 CF를 알아본다~

Neighborhood-based CF

- It is also referred to as **memory-based algorithms.**
- Neighborhood-based CF is mainly inspired by k**-nearest-neighbor methods.**
- Two types of neighborhood-based CF
    - **User-based neighborhood method** : The prediction is computed by **the weighted average of peer group ratings for a terget user.**
    - **Item-based neighborhood method** : The prediction is computed by **the weighted average rating of similar items for a target user.**

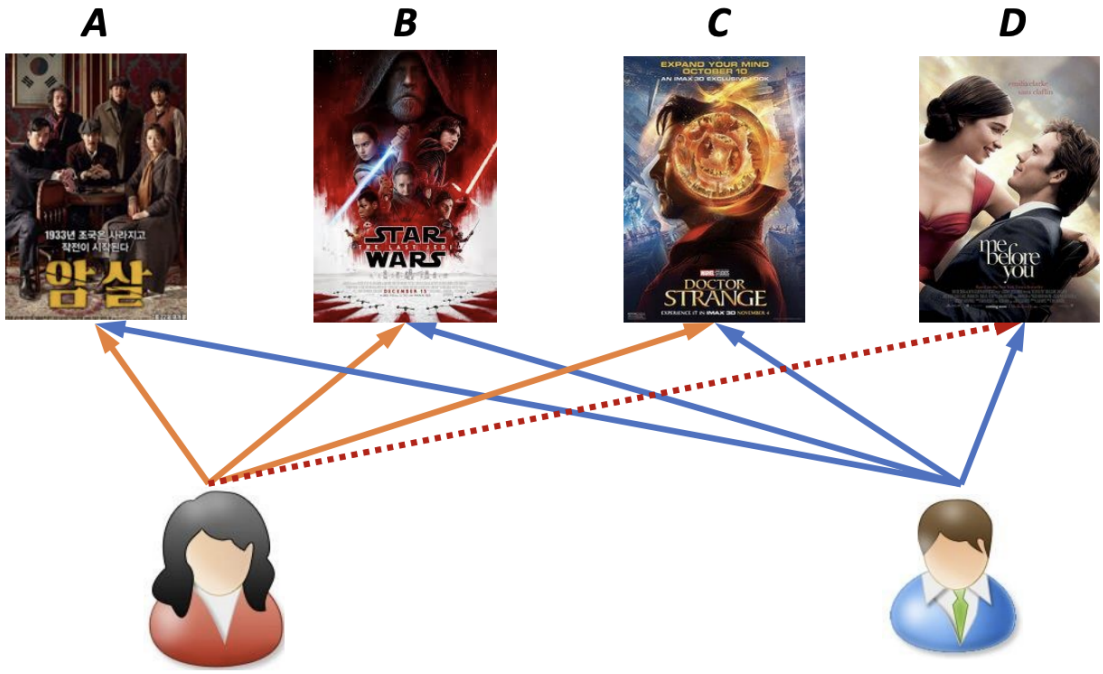
이웃기반 협업 필터링을 설명하기에 앞서 한 번 더 CF의 Key idea를 되짚어보자

Key Idea for Collaborative Filtering

Tell me what my peers like.

- Alice likes three movices A, B, and C.
- Bob also likes three movies A, B, and C. Bob also likes D.

- Then, Alice is most likely to prefer D.



Key Assumptions in CF

- Users have given the ratings to items(**implicitly** or **explicitly**).
- The users with **similar tastes in the past** will have **similar tastes in the future.**
- User preferences remain **stable** and c**onsistent** over time.

User preferences do not change over time.

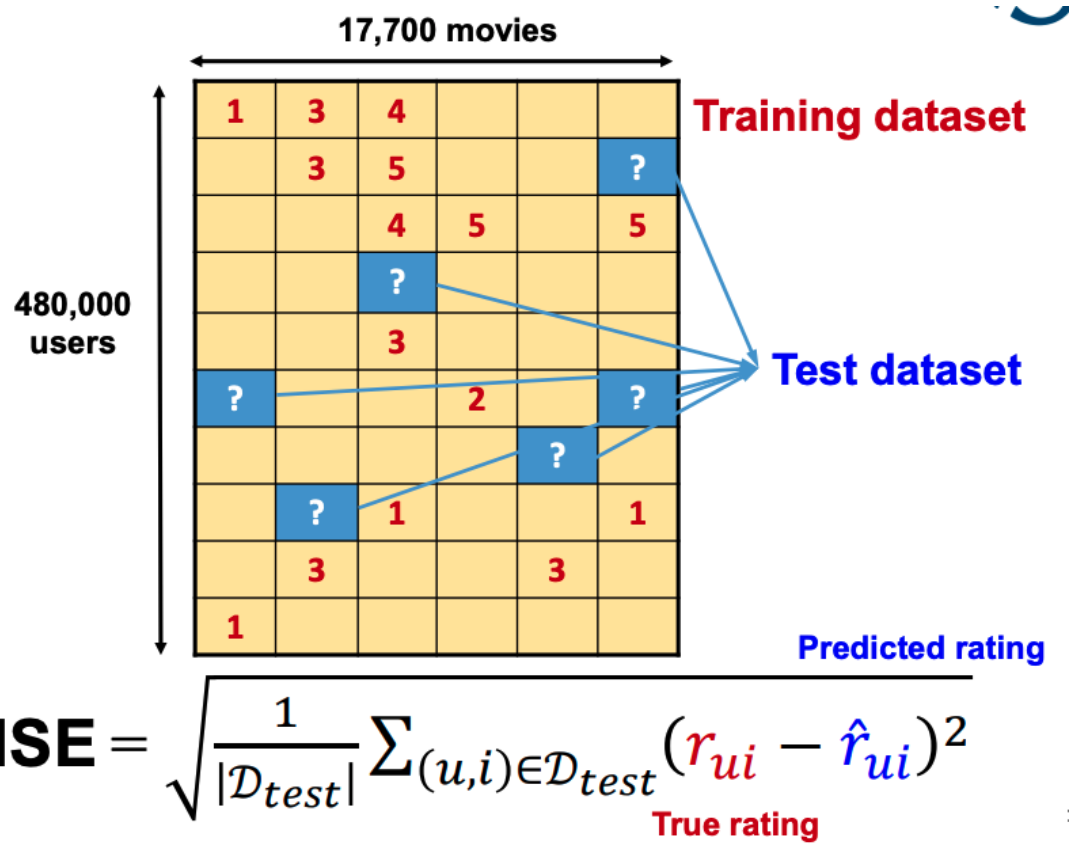
User-Item Rating Matrix

- We are given a user-item rating matrix $R \in R^{m*n}$.
  - Let R denote a user-item m x n rating matrix.
- Predict missing (or unobserved) user-item ratings.

Various User Ratings

- Continuous ratings
  - It is specified on a continuous scale, corresponding to the level of like/dislike of the item.
- Interval ratings
  - It is commonly drawn from a 5-point or 7-point scale.
  - It is also ordered by categorial values, i.e., ordinal ratings.
- Binary ratings : two options are present.
- Unary ratings
  - Implicit feedback : it only specifies a positive preference.
  - It often occurs in many real-world applications.

Example : A User-Item Matrix in Netflix Prize



---

2. User-based Collaborative Filtering

User-Based Collaborative Filtering (CF)

- We are given an **explicit user-item rating matrix** and solve the **rating prediction problem.**
- How to predict the rating of **Item5** for a target user, Bob?





	Item1	Item2	Item3	Item4	Item5
Bob	5	3	4	4	???
User1	3	1	2	2	2
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Three Key Questions in CF

- **Similarity measurement**
  - **How to calculate the similarity between users or items?**
- **Neighborhood selection**
  - **How to select similar users or items?**
- **Prediction function**
  - **How to predict the rating based on similar users or items?**

Measuring User Similarities

- **Zero-centering : the mean of ratings lies on zero.**
  - Normalize the ratings by substracting the average value.

$$\mu_u = \frac{\sum_{k \in \mathcal{I}_u} r_{uk}}{|\mathcal{I}_u|}, \qquad u \in \{1, \dots, m\}$$

$\mathcal{I}_u$: a set of items **rated** by user u

$$\mu_u = \frac{\sum_{k \in \mathcal{I}_u \cap \mathcal{I}_v} r_{uk}}{|\mathcal{I}_u \cap \mathcal{I}_v|}, \qquad u, v \in \{1, \dots, m\}$$

$\mathcal{I}_u \cap \mathcal{I}_v$: a set of items **co-rated** by two users u and v

- Numbers in brackets indicate normalized ratings.





	Item1	Item2	Item3	Item4	Item5	Average
Bob	5 (+1.0)	3 (-1.0)	4 (0.0)	4 (0.0)	???	4.0
User1	3 (+1.0)	1 (-1.0)	2 (0.0)	2 (0.0)	2 (0.0)	2.0
User2	4 (+0.2)	3 (-0.8)	4 (+0.2)	3 (-0.8)	5 (+1.2)	3.8
User3	3 (-0.2)	3 (-0.2)	1 (-2.2)	5 (+1.8)	4 (0.8)	3.2
User4	1 (-1.8)	5 (+2.2)	5 (+2.2)	2 (-0.8)	1 (-1.8)	2.8

- Adjusted cosine similarity

- $s_{uk} = r_{uk} - \hat{\mu}_u$ is the **normalized rating** for item k by user u.
- $\hat{\mu}_u$ is the average of **all the items** rated by user u.

$$sim(u,v) = AdjustedCosine(u,v) = \frac{\sum_{k \in \mathcal{I}_u \cap \mathcal{I}_v} s_{uk} \cdot s_{vk}}{\sqrt{\sum_{k \in \mathcal{I}_u \cap \mathcal{I}_v} s_{uk}^2} \sqrt{\sum_{k \in \mathcal{I}_u \cap \mathcal{I}_v} s_{vk}^2}}$$

$$-1 \leq sim(u,v) \leq 1$$

	Item1	Item2	Item3	Item4	Item5
Bob	5 (+1.0)	3 (-1.0)	4 (0.0)	4 (0.0)	???
User1	3 (+1.0)	1 (-1.0)	2 (0.0)	2 (0.0)	2 (0.0)
User2	4 (+0.2)	3 (-0.8)	4 (+0.2)	3 (-0.8)	5 (+1.2)
User3	3 (-0.2)	3 (-0.2)	1 (-2.2)	5 (+1.8)	4 (0.8)
User4	1 (-1.8)	5 (+2.2)	5 (+2.2)	2 (-0.8)	1 (-1.8)

$sim(Bob, U1)$	1.00
$sim(Bob, U2)$	0.60
$sim(Bob, U3)$	0.00
$sim(Bob, U4)$	-0.77

- Pearson correlation coefficient (PCC)

- $\mathcal{I}_u \cap \mathcal{I}_v$ is **a set of items co-rated** by two users u and v.
- μ_u is the average of items **co-rated** by two users u and v.

$$sim(u,v) = Pearson(u,v) = \frac{\sum_{i \in \mathcal{I}_u \cap \mathcal{I}_v} (r_{ui} - \mu_u)(r_{vi} - \mu_v)}{\sqrt{\sum_{i \in \mathcal{I}_u \cap \mathcal{I}_v} (r_{ui} - \mu_u)^2} \sqrt{\sum_{i \in \mathcal{I}_u \cap \mathcal{I}_v} (r_{vi} - \mu_v)^2}}$$

$$-1 \leq sim(u,v) \leq 1$$

	Item1	Item2	Item3	Item4	Item5
Bob	5 (+1.0)	3 (-1.0)	4 (0.0)	4 (0.0)	???
User1	3 (+1.0)	1 (-1.0)	2 (0.0)	2 (0.0)	2
User2	4 (+0.5)	3 (-0.5)	4 (+0.5)	3 (-0.5)	5
User3	3 (0.0)	3 (0.0)	1 (-2.0)	5 (+2.0)	4
User4	1 (-2.25)	5 (+1.75)	5 (+1.75)	2 (-1.25)	1

$sim(Bob, U1)$	1.00
$sim(Bob, U2)$	0.71
$sim(Bob, U3)$	0.00
$sim(Bob, U4)$	-0.79

Variant of Similarity Functions

- **Cosine similarity : use the cosine function of the raw ratings than mean-centered ratings.**

$$RawCosine(u,v) = \frac{\sum_{k \in \mathcal{I}_u \cap \mathcal{I}_v} r_{uk} \cdot r_{vk}}{\sqrt{\sum_{k \in \mathcal{I}_u \cap \mathcal{I}_v} r_{uk}^2} \sqrt{\sum_{k \in \mathcal{I}_u \cap \mathcal{I}_v} r_{vk}^2}}$$

- **The normalized factors in the denomiator are computed by all the rated items and not the mutually rated items.**

$$RawCosine(u,v) = \frac{\sum_{k \in \mathcal{I}_u \cap \mathcal{I}_v} r_{uk} \cdot r_{vk}}{\sqrt{\sum_{k \in \mathcal{I}_u} r_{uk}^2} \sqrt{\sum_{k \in \mathcal{I}_v} r_{vk}^2}}$$

Selecting Neighbors

- All the neighbors
- Random neighbors
- #of neighborhoods vs. thresholds
  - Select neighbors with positive similarities.
  - **Select top-K neighbors sorted by similarity values.**

	Item1	Item2	Item3	Item4	Item5
Bob	5 (+1.0)	3 (-1.0)	4 (0.0)	4 (0.0)	???
User1	3 (+1.0)	1 (-1.0)	2 (0.0)	2 (0.0)	2 (0.0)
User2	4 (+0.2)	3 (-0.8)	4 (+0.2)	3 (-0.8)	5 (+1.2)
User3	3 (-0.2)	3 (-0.2)	1 (-2.2)	5 (+1.8)	4 (0.8)
User4	1 (-1.8)	5 (+2.2)	5 (+2.2)	2 (-0.8)	1 (-1.8)

$sim(Bob, U1)$	1.00
$sim(Bob, U2)$	0.60
$sim(Bob, U3)$	0.00
$sim(Bob, U4)$	-0.77

How to Choose Neighbors?

- **Choose top-K neighbors sorted by similarity values.**
  - **N_u: a set of top-K neighbors for user u**
- **The neighbors should have the rating for a target item j.**
  - **U_j: a set of user u who rates item j**

Making Prediction

- Calculate the **weighted average** from neighbors.
- Add the **average of an active user** from **neighbors' bias.**

$$\hat{r}_{uj} = \mu_u + \frac{\sum_{v \in \mathcal{N}_u \cap \mathcal{U}_j} sim(u,v) \cdot (r_{vj} - \mu_v)}{\sum_{v \in \mathcal{N}_u \cap \mathcal{U}_j} |sim(u,v)|}$$

$\mathcal{N}_u$: a set of top-K neighbors for user u

$\mathcal{U}_j$: a set of user u who rates item j

	Item1	Item2	Item3	Item4	Item5
Bob	5 (+1.0)	3 (-1.0)	4 (0.0)	4 (0.0)	???
User1	3 (+1.0)	1 (-1.0)	2 (0.0)	2 (0.0)	2 (0.0)
User2	4 (+0.2)	3 (-0.8)	4 (+0.2)	3 (-0.8)	5 (+1.2)
User3	3 (-0.2)	3 (-0.2)	1 (-2.2)	5 (+1.8)	4 (0.8)
User4	1 (-1.8)	5 (+2.2)	5 (+2.2)	2 (-0.8)	1 (-1.8)

$\mathcal{U}_j = \{U1, U2, U3, U4\}$

$sim(Bob, U1)$	1.00
$sim(Bob, U2)$	0.60
$sim(Bob, U3)$	0.00
$sim(Bob, U4)$	-0.77

$$pred(\text{Bob}, \text{Item5}) = 4.0 + \frac{1.0 \times 0.0 + 0.6 \times 1.2}{1.0 + 0.6} = 4.45$$

Bob's average Neighbors' bias

	Item1	Item2	Item3	Item4	Item5
Bob	5 (+1.0)	3 (-1.0)	4 (0.0)	4 (0.0)	???
User1	3 (+1.0)	1 (-1.0)	2 (0.0)	2 (0.0)	2 (0.0)
User2	4 (+0.2)	3 (-0.8)	4 (+0.2)	3 (-0.8)	5 (+1.2)
User3	3 (-0.2)	3 (-0.2)	1 (-2.2)	5 (+1.8)	4 (0.8)
User4	1 (-1.8)	5 (+2.2)	5 (+2.2)	2 (-0.8)	1 (-1.8)

$\mathcal{N}_u \cap \mathcal{U}_j = \{U1, U2\}$

$sim(Bob, U1)$	1.00
$sim(Bob, U2)$	0.60
$sim(Bob, U3)$	0.00
$sim(Bob, U4)$	-0.77

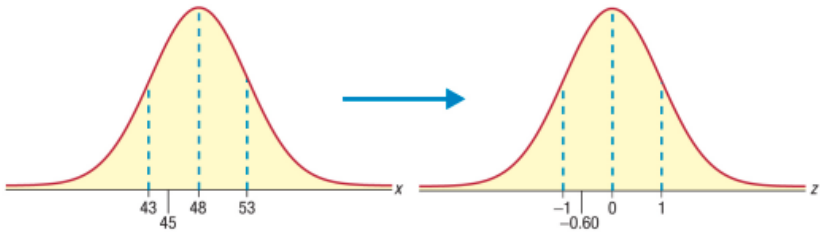
Variants of the Prediction Function

- Use the **Z-score normalization** instead of the mean-centered normalization.

$$\sigma_u = \sqrt{\frac{\sum_{j \in \mathcal{J}_u} (r_{uj} - \mu_u)^2}{|\mathcal{J}_u| - 1}}$$

$$z_{uj} = \frac{r_{uj} - \mu_u}{\sigma_u} = \frac{s_{uj}}{\sigma_u}$$

$$\Rightarrow \quad \hat{r}_{uj} = \mu_u + \sigma_u \frac{\sum_{v \in \mathcal{N}_u \cap \mathcal{U}_j} sim(u, v) \cdot z_{vj}}{\sum_{v \in \mathcal{N}_u \cap \mathcal{U}_j} |sim(u, v)|}$$



J_u : a set of items rated by user u

z-score normalization은 전체 데이터의 평균을 0, 표준편차를 1로 만드는 정규화 방법이다.

z-score normalization은 비교적 이상치에 영향을 적게 받는다.

- Amplify the similarity by exponentiating it to the power of α.

$$sim(u,v) = Pearson(u,v)^{\alpha}$$

- By choosing $\alpha > 1$, it is possible to amplify the importance of the similarity in the prediction function.

$$\hat{r}_{uj} = \mu_u + \frac{\sum_{v \in \mathcal{N}_u \cap \mathcal{U}_j} sim(u,v) \cdot r_{vj}}{\sum_{v \in \mathcal{N}_u \cap \mathcal{U}_j} |sim(u,v)|}$$

Example: Zero-centered Normalization

Users \ Movies	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12	Average
U1	3		3		?	5			5				4
U2			4	4			4			2	1	3	3
U3	2	4		1	2		3		4	3	5		3
U4		2	4		5			4	5				4
U5			4	3	4	1					1	5	3
U6	1		3		3			5			3		3

- Estimate the rating of movie 5 by user 1.

34

Example: Zero-centered Normalization

Users \ Movies	$I1$	$I2$	$I3$	$I4$	$I5$	$I6$	$I7$	$I8$	$I9$	$I10$	$I11$	$I12$	Average
$U1$	-1		-1		?	+1			+1				4
$U2$			+1	+1			+1			-1	-2	0	3
$U3$	-1	+1		-2	-1		0		+1	0	+2		3
$U4$		-2	0		+1			0	+1				4
$U5$			+1	0	+1	-2					-2	+2	3
$U6$	-2		0		0			+2			0		3

Example: User Similarity

- **How to calculate user similarity?**
  - Calculate the similarity for co-rated items $\{I3\}$ for $U2$.
  - Calculate the similarity for co-rated items $\{I1, I9\}$ for $U3$.
  - Calculate the similarity for co-rated items $\{I3, I9\}$ for $U4$.

Users \ Movies	$I1$	$I2$	$I3$	$I4$	$I5$	$I6$	$I7$	$I8$	$I9$	$I10$	$I11$	$I12$
$U1$	-1		-1		?	+1			+1			
$U2$			+1	+1			+1			-1	-2	0
$U3$	-1	+1		-2	-1		0		+1	0	+2	
$U4$		-2	0		+1			0	+1			
$U5$			+1	0	+1	-2					-2	+2
$U6$	-2		0		0			+2			0	

36

Example: User Neighbor Selection

- The users who do not have the predicted rating are not regarded as neighbors.
  - E.g., $U2$ is not regarded as the neighbor.

Users \ Movies	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12
U1	-1		-1		?	+1			+1			
U2			+1	+1			+1			-1	-2	0
U3	-1	+1		-2	-1		0		+1	0	+2	
U4		-2	0		+1			0	+1			
U5			+1	0	+1	-2					-2	+2
U6	-2		0		0			+2			0	

37

Improving Similarity Measures

- When the number of co-rated items is too small, the similarity between users may be amplified.

	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12
U1	4	2	5		5	4			3			
U2			3	4			4	5		2	1	3

The number of co-rated items is only one.

Improving Similiarity Measures

- Use significance weighting by linearly reducing the weight when the number of co-rated items is small.
  - If $|J_u \cap J_v| \geq \gamma$, $sim'(u,v)$ is equal to $sim(u,v)$
  - Otherwise, $sim(u,v)$ is penalized.

$$sim'(u,v) = \frac{\min(|\mathcal{I}_u \cap \mathcal{I}_v|, \gamma)}{\gamma} \cdot sim(u,v)$$

$\mathcal{I}_u$: a set of items rated by the user u

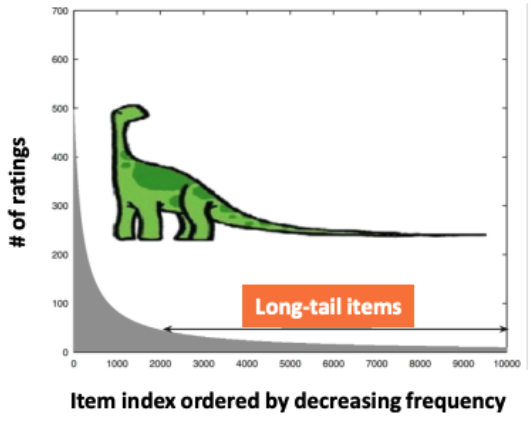
$|\mathcal{I}_u \cap \mathcal{I}_v|$: the number of co-rated items for two users u and v

γ: a threshold

- This change bounds the similarity to the interval [0, 1].

Long-tail Item Distribution

- Users tend to rate more popular items.
  - **Missing entries** in the rating matrix are **Not random**.



- The rating distribution is **missing-not-at-random.**
  - When an item is popular, it is liekly to be chosen as a preferred item.
  - It incurs a **selection bias.**

Impact of Long-tail Items

- Popular items heavily affect measuring the similarity.
- Inverse user frequency
  - Adopt the notion of **inverse document frequency (IDF)** used in IR.
  - The rarer the item is, the higher the weight is.

$$w_j = \log\left(\frac{m}{m_j}\right)$$

m: # of total users
m_j: # of users who have rated item j

➢ **Pearson correlation coefficient (PCC)**

$$Pearson(u,v) = \frac{\sum_{j \in \mathcal{I}_u \cap \mathcal{I}_v} w_j (r_{uj} - \mu_u)(r_{vj} - \mu_v)}{\sqrt{\sum_{j \in \mathcal{I}_u \cap \mathcal{I}_v} w_j (r_{uj} - \mu_u)^2} \sqrt{\sum_{j \in \mathcal{I}_u \cap \mathcal{I}_v} w_j (r_{vj} - \mu_v)^2}}$$

---

3. item-Based Collaborative Filtering

Item-Based CF

- Basic idea : use the **similarity between items.**
  - **It is just like transposing** a user-item rating matrix.
  - Note : **the normalization is the same for user-based CF.**
- Step 1 : look for items that are similar to Item 5.
- Step 2 : take Bob's ratings to predict the rating for Item 5.

	Item1	Item2	Item3	Item4	Item5	Average
Bob	5 (+1.0)	3 (-1.0)	4 (0.0)	4 (0.0)	???	4.0
User1	3 (+1.0)	1 (-1.0)	2 (0.0)	2 (0.0)	2 (0.0)	2.0
User2	4 (+0.2)	3 (-0.8)	4 (+0.2)	3 (-0.8)	5 (+1.2)	3.8
User3	3 (-0.2)	3 (-0.2)	1 (-2.2)	5 (+1.8)	4 (0.8)	3.2
User4	1 (-1.8)	5 (+2.2)	5 (+2.2)	2 (-0.8)	1 (-1.8)	2.8

Measuring Item Similarities

- Adjusted cosine similarity
  - **S_{ui} = r_{ui} - $\hat{\mu}_u$ is the normalized rating for item i by user u.**
  - $\hat{\mu}_u$ is the average of all the items rated by user u.

$$sim(i,j) = AdjustedCosine(i,j) = \frac{\sum_{u \in \mathcal{U}_i \cap \mathcal{U}_j} s_{ui} \cdot s_{uj}}{\sqrt{\sum_{u \in \mathcal{U}_i \cap \mathcal{U}_j} s_{ui}^2} \sqrt{\sum_{u \in \mathcal{U}_i \cap \mathcal{U}_j} s_{uj}^2}}$$

$$-1 \leq sim(i,j) \leq 1$$

	Item1	Item2	Item3	Item4	Item5
Bob	5 (+1.0)	3 (-1.0)	4 (0.0)	4 (0.0)	???
User1	3 (+1.0)	1 (-1.0)	2 (0.0)	2 (0.0)	2 (0.0)
User2	4 (+0.2)	3 (-0.8)	4 (+0.2)	3 (-0.8)	5 (+1.2)
User3	3 (-0.2)	3 (-0.2)	1 (-2.2)	5 (+1.8)	4 (0.8)
User4	1 (-1.8)	5 (+2.2)	5 (+2.2)	2 (-0.8)	1 (-1.8)

$sim(I5, I1)$	0.693
$sim(I5, I2)$	−0.863
$sim(I5, I3)$	−0.762
$sim(I5, I4)$	0.392

44

Making Predictions

- A prediction function is

$$\hat{r}_{uj} = \frac{\sum_{k \in \mathcal{N}_j \cap \mathcal{I}_u} sim(k,j) * r_{uk}}{\sum_{k \in \mathcal{N}_j \cap \mathcal{I}_u} |sim(k,j)|}$$

$\mathcal{N}_j$: a set of neighbors for item j

$\mathcal{I}_u$: a set of items rated by user u

- Why not do we consider the average rating for user u?

- Calculate the weighted average from neighbors of Item5.

$$pred(\text{Bob}, \text{Item5}) = \frac{0.693 \times 5 + 0.392 \times 4}{(0.693 + 0.392)} = 4.639$$

	Item1	Item2	Item3	Item4	Item5
Bob	5 (+1.0)	3 (-1.0)	4 (0.0)	4 (0.0)	???
User1	3 (+1.0)	1 (-1.0)	2 (0.0)	2 (0.0)	2 (0.0)
User2	4 (+0.2)	3 (-0.8)	4 (+0.2)	3 (-0.8)	5 (+1.2)
User3	3 (-0.2)	3 (-0.2)	1 (-2.2)	5 (+1.8)	4 (0.8)
User4	1 (-1.8)	5 (+2.2)	5 (+2.2)	2 (-0.8)	1 (-1.8)

$\mathcal{N}_j \cap \mathcal{I}_u = \{I1, I4\}$

$sim(I5, I1)$	0.693
$sim(I5, I2)$	−0.863
$sim(I5, I3)$	−0.762
$sim(I5, I4)$	0.392

Example: Item Neighbor Selection

➢ **Among the items rated by user U1, choose the top-K neighbors.**

- $\mathcal{I}_u$: a set of items rated by user u
  - $\mathcal{I}_1 = \{I1, I3, I6, I9\}$
- $\mathcal{N}_j$: a set of top−K neighbors for item j.

Movies

Users	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12
U1	3		3		?	5			5			
U2			4	4			4			2	1	3
U3	2	4		1	2		3		4	3	5	
U4		2	4		5			4	5			
U5			4	3	4	1					1	5
U6	1		3		3			5			3	

47

4. User-based vs. Item-based CF

Neighborhood-based Methods

- Advantages
  - It is **easy to implement and debug.**
  - The **interpretability** of the item-based method is **notable.**
  - It is possible for i**ncremental approximations** for **new items/users.**
- Disadvangates
  - It is impratical in large-scale settings. (m >> n)
    - User-based methods requires $O(m^2)$.
  - It is limited in addressing the **data sparsity.**
    - **Top-K neighbors** are only used.
- How do we complement the drawback of neighborhood-based methods?

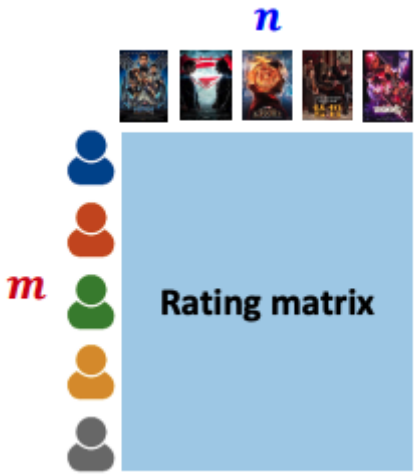
Data Sparsity Problem

- **Cold start problem(CF의 근본적인 한계)**
  - cold start란, 추천 시스템이 새로운 또는 어떤 유저들에 대한 충분한 정보가 수집된 상태가 아니라서 해당 유저들에게 적절한 제품을 추천해주지 못하는 문제를 말한다.
  - How to recommend **new items?**
  - What to recommend to **new users?**
- **Straightforward approaches**
  - Ask/force new users to rate a set of items.
  - Use another method, e.g., content-based, demographic or simply non-personalized, in the initial phase.

Scalability for User-based CF

- Given **m** users and **n** items,

- The correlation between two users is $O(n)$.
- All correlation for a user is $O(mn)$.
- All pair-wise correlation is $O(m^2n)$.
- For each user, recommendation is $O(mn)$.



- What is the bottleneck of computing user-based CF?
- Which is larger : # of users vs. # of items ?
  - # of users is larger than # of items.

User-based vs. Item-based CF

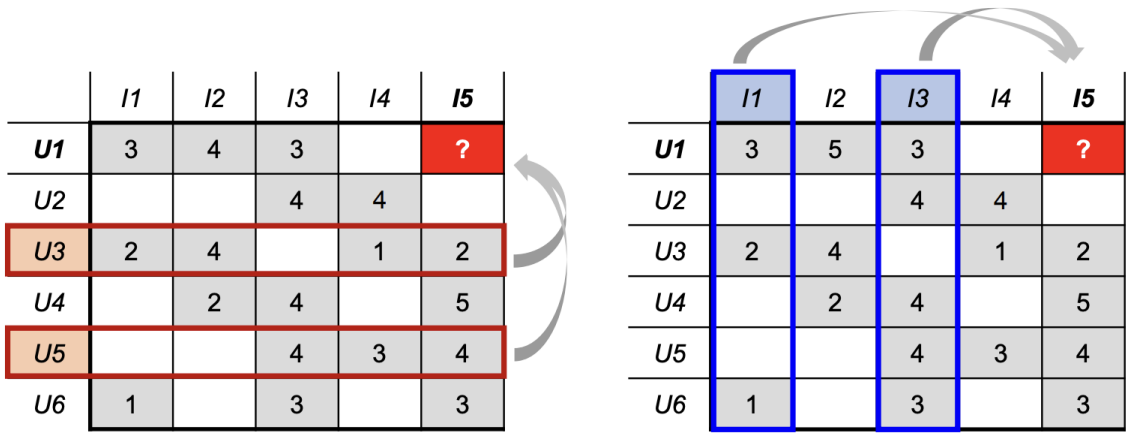
- Item-based CF often is better than user-based CF.
  - In item-based CF, **user's own ratings** are used for the recommendation.
    - It is more robust to **shilling attacks** in recommender systems.
  - In user-based CF, the ratings are **extrapolated from other users.**
- Item-based CF can provide a concrete reason as follows.
  - The **item neighbors** can be used for explanations.

> Because you watched "*Terminator*," [the recommendations are] ⟨*lists*⟩.

- Item-based CF is more **stable** with changes to the ratings.
  - # users is generally larger than # of items.
  - New users are more liekly to be added more frequently.

Unifying Two Methods

- Drawback of existing methods
  - User-based method **ignores the similarity between items.**
  - Item-based method **ignores the similarity between users.**
- How to combine the similarity between users and items?



	I1	I2	I3	I4	I5
U1	3	4	3		?
U2			4	4	
U3	2	4		1	2
U4		2	4		5
U5			4	3	4
U6	1		3		3

	I1	I2	I3	I4	I5
U1	3	5	3		?
U2			4	4	
U3	2	4		1	2
U4		2	4		5
U5			4	3	4
U6	1		3		3

- Step 1 : The rows of the rating matrix are mean-centered.
- Step 2 : For a target entry (u, j), determine the most similar rows/columns by the cosine coefficient.

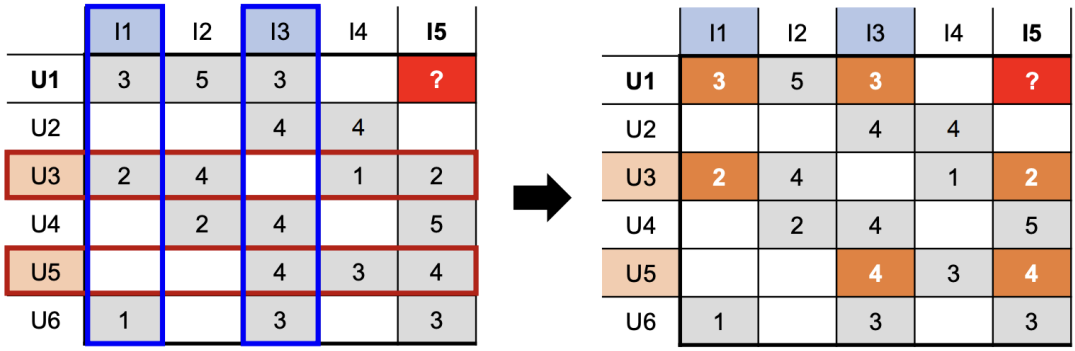
$$\hat{r}_{uj} = \mu_u + \frac{\sum_{v \in N_u \cap U_j} sim(u,v) \cdot s_{vj} + \sum_{k \in N_j \cap S_u} sim(k,j) \cdot s_{uk}}{\sum_{v \in N_u \cap U_j} |sim(u,v)| + \sum_{k \in N_j \cap S_u} |sim(k,j)|}$$

	I1	I2	I3	I4	I5
U1	3	4	3		?
U2			4	4	
U3	2	4		1	2
U4		2	4		5
U5			4	3	4
U6	1		3		3

- Step3 : Predict it by **a weighted combination by using similar rows/columns.**

- Generalizing similarity computation in an **entry-wise manner**

- For a target entry, **determine the most similar entries** with the combination function of the smilarity between users and columns.
- Predict it using **a weighted combination of the most similar entries.**



5. Discussion on Other Methods

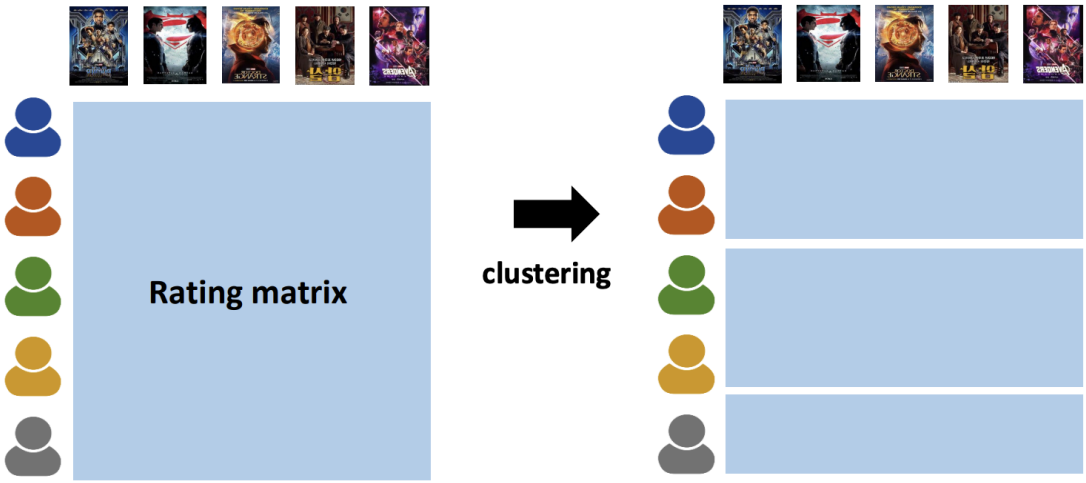
How to Reduce Offline Computation?

- The main problem with neighborhood-based methods is **the complexity of the offline phase.**
- Solution : replace the offline **nearest-neighbor computation phase** with an offline clustering phase.
  - Top-k closest peers **within the same cluster** are used for prediction.
- Because pairwise similarity computation is performed **within the same cluster**, it reduces comptational costs.
  - When the rating matrix is very large, it provides a practical alternative at a small cost.
  - It has a trade-off between efficiency and accuracy.

Clustering For Neighborhood Methods

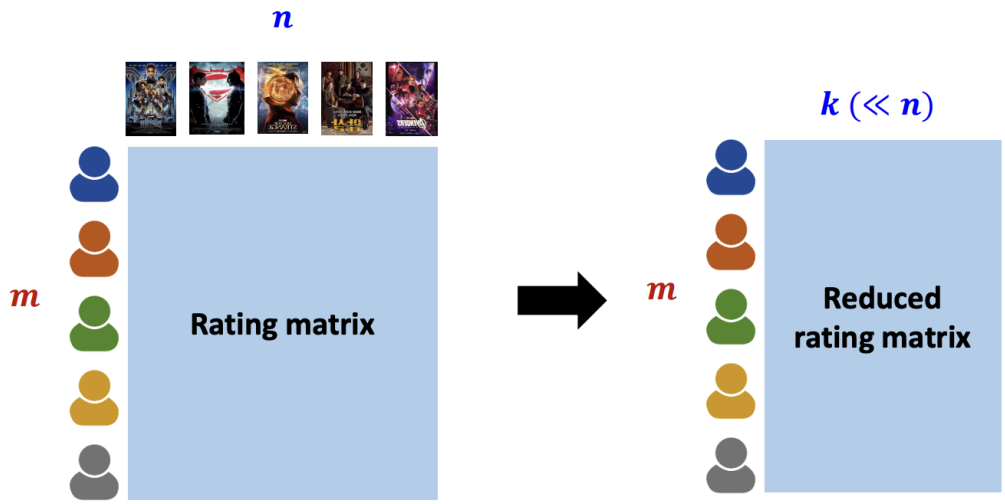
- An entire set of users is split into several user groups.
  - Typically, k-means clustering is used.

- Challenge : **the rating matrix is incomplete.**



Dimensionality Reduction

- It provides **dense low-dimensional** representations.
  - A reduced representation can be created in terms of row-wise or column-wise **latent factors.**
  - Transform the m x n matrix R into a low-dimensional space k.



- Overall process using the reduced matrix

- Eacn sparse n-dimensional vector is transformed into a dense low-dimensional vector
- After the k-dimensional representation of each user is determined, the similarity is computed from the target user.

- It is more robust since low-dimensional vectors are dense.
- It is more efficient because of using low dimensionality.
  - A simple cosine or dot product is used in the low dimensionality.

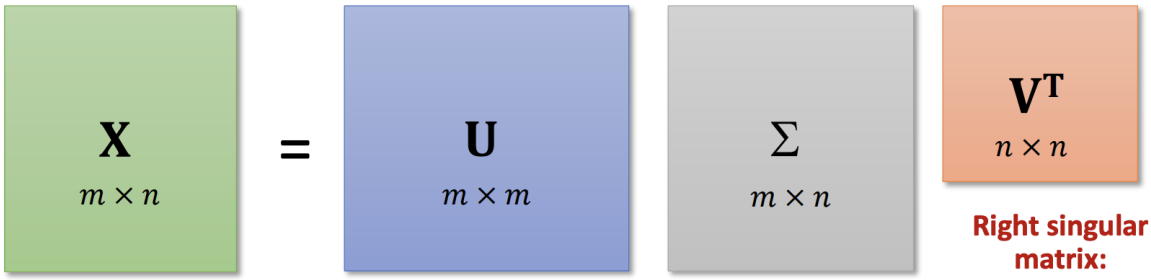
Singular Value Decomposition (SVD)

➢ **The rating matrix R can be decomposed into three matrices.**

$$\mathbf{X}_{[m\times n]} = \mathbf{U}_{[m\times m]}\Sigma_{[m\times n]}(\mathbf{V}_{n\times n})^{\mathbf{T}}$$

- **U, Σ, V: unique**
- **U, V: column orthonormal**
  - $\mathbf{U}^{\mathbf{T}}\mathbf{U} = \mathbf{I}$, $\mathbf{V}^{\mathbf{T}}\mathbf{V} = \mathbf{I}$ ($\mathbf{I}$: identity matrix)
  - Columns are orthogonal unit vectors.
- **Σ: diagonal**
  - Entries (**singular values**) are **positive**.
  - Single values are sorted in decreasing order ($\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$).

$$\mathbf{X}_{[m\times n]} = \mathbf{U}_{[m\times m]}\Sigma_{[m\times n]}(\mathbf{V}_{n\times n})^{\mathbf{T}}$$



Input matrix: $m \times n$ matrix

Left singular matrix: $m \times m$ matrix

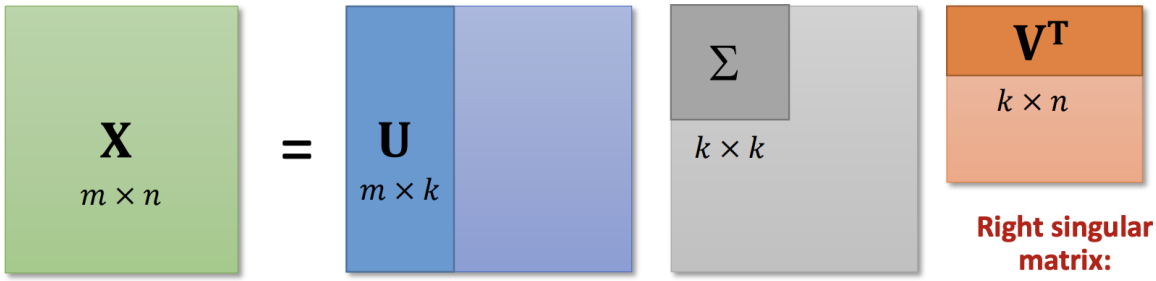
Diagonal matrix: $m \times n$ matrix, where diagonal elements are singular values (strength of concepts)

Right singular matrix: $n \times n$ matrix

62

Dimensionality Reduction using SVD

$$\mathbf{X}_{[m\times n]} \approx \mathbf{U}_{[m\times k]}\Sigma_{[k\times k]}(\mathbf{V}_{n\times k})^{\mathbf{T}}$$



Input matrix: $m \times n$ matrix

Left singular matrix: $m \times k$ matrix

Diagonal matrix: $k \times k$ matrix, where diagonal elements are singular values **(strength of concepts)**

Right singular matrix: $k \times n$ matrix

6

- **Augment an incomplete matrix by filling in missing entries.**
  - The missing entry is estimated as the mean of the corresponding row in the matrix.
  - $\mathbf{R}_{full}$: augmented full matrix
- **Compute the $n \times n$ similarity matrix $\mathbf{S} = \mathbf{R}_{full}^{\mathbf{T}}\mathbf{R}_{full}$.**
- **Determine the dominant basis vector of $\mathbf{R}_{full}$ for SVD.**
  - $\mathbf{S} = \mathbf{P}\Delta\mathbf{P}^{\mathbf{T}}$, where $\mathbf{P}$ is $n \times n$ matrix and Δ is a diagonal matrix.
  - Let $\mathbf{P}_k$ denote the $n \times k$ matrix with the columns of $\mathbf{P}$ corresponding to **the largest k eigenvectors**.
  - PCA can also be used for dimensionality reduction.
- **Build the reduced $m \times k$ matrix by $\mathbf{R}_{full}\mathbf{P}_k$.**

S is co-occurrence matrix

Example : Augmented Matrix

- **Missing ratings are filled by the average for each user.**

	I1	I2	I3
U1	1	1	1
U2	7	7	7
U3	3	1	1
U4	5	7	7
U5	3	1	?
U6	5	7	?
U7	3	1	?
U8	5	7	?

	I1	I2	I3
U1	1	1	1
U2	7	7	7
U3	3	1	1
U4	5	7	7
U5	3	1	**2**
U6	5	7	**6**
U7	3	1	**2**
U8	5	7	**6**

Bias Problem

- The full matrix is derived from the **incomplete matrix** by filling in unspecified entries.
- It may distort the covariance between items, i.e., the item with sparse ratings.
  - After filling missing ratings,(I1 and I2) are higher than (I1 and I3).

	I1	I2	I3
I1	3.0	**4.5**	**4.0**
I2	4.5	9.0	7.5
I3	4.0	7.5	6.5

Covariance matrix **with** filling missing ratings

	I1	I2	I3
I1	3.0	**4.5**	**6.0**
I2	4.5	9.0	9.0
I3	6.0	9.0	9.0

Covariance matrix **without** filling missing ratings

Example: SVD in Matlab

Items

Users	*I1*	*I2*	*I3*	*I4*	*I5*	*I6*	*I7*	*I8*	*I9*	*I10*	*I11*	*I12*	**Average**
U1	1		3			5			5		4		3.60
U2			5	4			4			2	1	3	3.16
U3	2	4		1	2		3		4	3	5		3.00
U4		2	4		5			4			2		4.40
U5			4	3	4	2					2	5	3.33
U6	1		3		3			2			4		2.60

```
R =

   -2.6000         0   -0.6000         0         0    1.4000         0         0    1.4000         0    0.4000         0
         0         0    1.8400    0.8400         0         0    0.8400         0         0   -1.1600   -2.1600   -0.1600
   -1.0000    1.0000         0   -2.0000   -1.0000         0         0         0    1.0000         0    2.0000         0
         0   -1.4000    0.6000         0    1.6000         0         0    0.6000         0         0   -1.4000         0
         0         0    0.6700   -0.3300    0.6700   -1.3300         0         0         0         0   -1.3300    1.6700
   -1.6000         0    0.4000         0    0.4000         0         0   -0.6000         0         0    0.4000         0
```

Example: SVD in Matlab

- **Building 3-ranked SVD**
- **Matlab code: [U, S, V] = svds(R, 3, 'L')**

```
>> [U, S, V] = svds(R, 3, 'L')

U =

    0.3911   -0.7692    0.1873
   -0.4821   -0.3917   -0.1735
    0.5985    0.0222   -0.5781
   -0.3765   -0.3183    0.0862
   -0.3127   -0.0910   -0.7601
    0.1301   -0.3805   -0.1242

S =

    5.0446         0         0
         0    3.3714         0
         0         0    2.5352

V =

   -0.3615    0.7672    0.1143
    0.2231    0.1388   -0.2756
   -0.2983   -0.1967   -0.3704
   -0.2971   -0.1019    0.4975
   -0.2693   -0.2209    0.0619
    0.1910   -0.2835    0.5022
   -0.0803   -0.0976   -0.0575
   -0.0602    0.0111    0.0498
    0.2272   -0.3128   -0.1246
    0.1109    0.1348    0.0794
    0.6719    0.2958    0.0529
   -0.0882   -0.0265   -0.4897
```

Example: SVD in Matlab

Users	Items												
		I1	***I2***	*I3*	*I4*	*I5*	*I6*	*I7*	*I8*	*I9*	*I10*	*I11*	*I12*	**Average**
	U1	1	?	3			5			5		4		3.60
	U2			5	4			4			2	1	3	3.16
	U3	2	4		1	2		3		4	3	5		3.00
	U4		2	4		5			4			2		4.40
	U5			4	3	4	2					2	5	3.33
	U6	1		3		3			2			4		2.60

```
>> U*S*V'

ans =

   -2.6487   -0.0505   -0.2544   -0.0858    0.0709    1.3507    0.0674   -0.1239    1.2004   -0.0931    0.5840   -0.3380
   -0.1843   -0.6046    1.1483    0.6381    0.9192   -0.3110    0.3494    0.1100   -0.0846   -0.4825   -2.0479    0.4650
   -1.2015    1.0880   -0.3727   -1.6337   -0.9203   -0.1806   -0.1654   -0.2541    0.8451    0.2284    1.9733    0.4494
   -0.1119   -0.6329    0.6968    0.7823    0.7619    0.0514    0.2446    0.1134   -0.1230   -0.3378   -1.5819    0.0889
    0.1147    0.1367    1.2446   -0.4588    0.3731   -1.1821    0.2673   -0.0043   -0.0224   -0.3692   -1.2525    1.0910
   -1.2575    0.0552    0.1733   -0.2210    0.0871    0.3309    0.0906   -0.0694    0.5897   -0.1252    0.0449    0.1303
```

The entry in reduced user-item matrix + user_average

$$= -0.0505 + 3.60000 = \mathbf{3.5495}$$

Limitations in SVD

- **Challenge #1: missing values**
  - Filling the missing values with the user average may not be accurate.
- **Challenge #2: scalability**
  - SVD computation is $O(m^2n + n^3)$.
    - It does not scale well to large datasets.
- **Challenge #3: lack of transparency**
  - It is non-trivial to interpret the semantics of latent features.

Solution : Direct Matrix Factorization

SVD를 그냥 추천 모델로 사용해보자!

- When the matrix is sparse, the covariance estimation is statistically unreliable. ☹

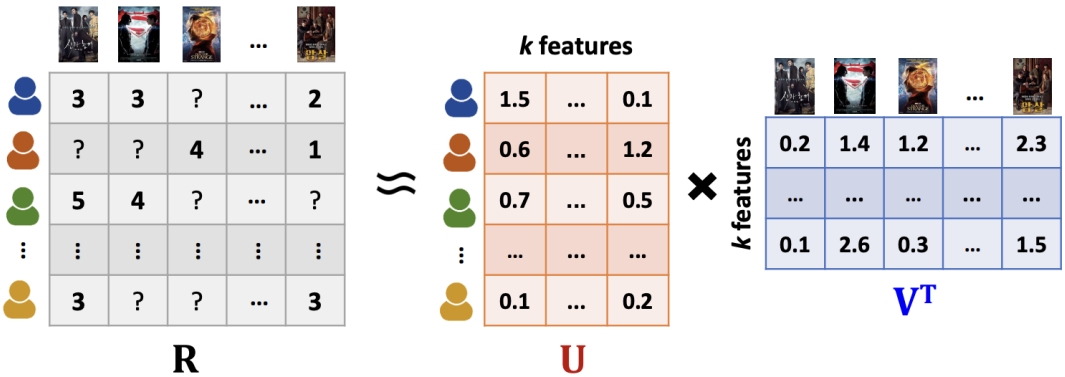
- What about directly applying matrix factorization for the rating matrix?

$$\mathbf{R}_{[m\times n]} \approx \mathbf{U}_{[m\times k]}\Sigma_{[k\times k]}(\mathbf{V}_{n\times k})^{\mathrm{T}}$$

- It provides the reduced basis as well as the representation of the ratings in the reduced basis.

Latent Factor Model

- The matrix $\mathbf{R}$ can be approximated as the linear combination of two latent matrices $\mathbf{U}$ and $\mathbf{V}$.
  - $\mathbf{R}$: user-item rating matrix ($m \times n$ matrix)
  - $\mathbf{U}$: latent user matrix ($m \times k$ matrix)
  - $\mathbf{V}$: latent item matrix ($n \times k$ matrix)
    - k: # of latent features



Arkadiusz Paterek, *Improving regularized singular value decomposition for collaborative filtering*, KDD Cup 2007

72

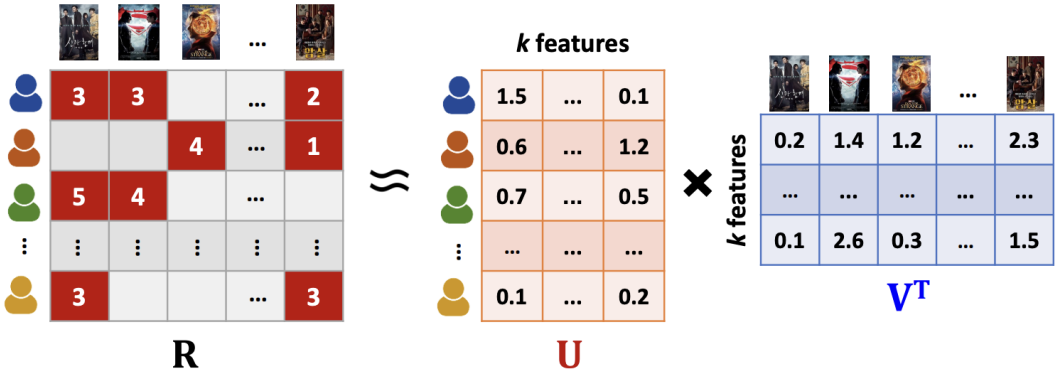
Latent Factor Model

- **Do not care about the values on the missing ones.**
- **Factorize a rating matrix into two latent matrices.**
  - The matrix **R** can be approximated as a product of **thin** $\mathbf{U}\mathbf{V}^{\mathbf{T}}$.

$$\min_{U,V} \sum_{u=1}^{m} \sum_{i=1}^{n} y_{ui} \left(r_{ui} - \mathbf{U}_u \mathbf{V}_i^{\mathbf{T}}\right)^2$$

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

U_u: u-th row of U V_i: i–th row of V

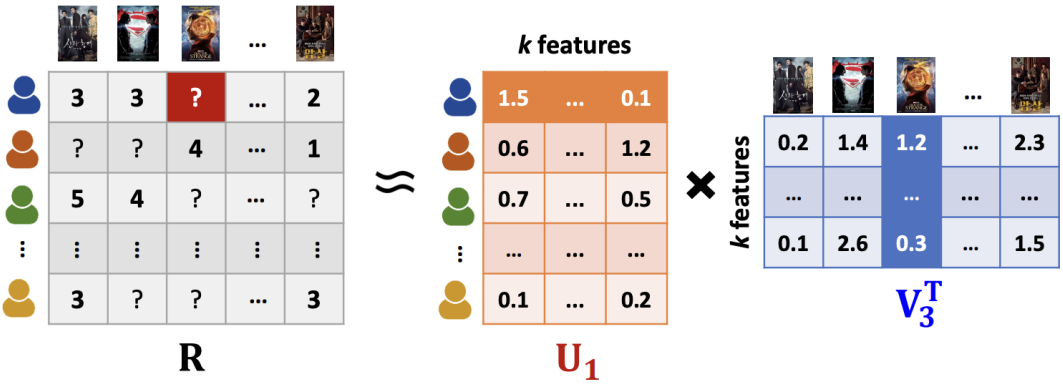


73

Predicting Unrated Items

- **After learning two matrices U and V, estimate the missing rating of user u from item i as $\mathbf{U}_u \mathbf{V}_i^{\mathbf{T}}$.**

$$\hat{r}_{ui} = \mathbf{U}_u \mathbf{V}_i^{\mathbf{T}}$$



Regression for Neighborhood Methods

- **It is possible to replace the similarity coefficient between users with the unknown parameter.**
  - Let $s_{vj} = r_{vj} - \mu_v$.

$$\hat{r}_{uj} = \mu_u + \frac{\sum_{v \in \mathcal{N}_u \cap \mathcal{U}_j} sim(u,v) \cdot s_{vj}}{\sum_{v \in \mathcal{N}_u \cap \mathcal{U}_j} |sim(u,v)|} \quad \Rightarrow \quad \hat{r}_{uj} = \mu_u + \sum_{v \in \mathcal{N}_u \cap \mathcal{U}_j} w_{uv} \cdot s_{vj}$$

w_{uv}: user similarity coefficient between two user u and v

- **Minimize the following objective function.**

$$\sum_{u=1}^{m} \sum_{j \in \mathcal{I}_u} \left(r_{uj} - \left[\sum_{v \in \mathcal{N}_u \cap \mathcal{U}_j} w_{uv} \cdot (r_{vj} - \mu_v) \right] \right)^2$$

Regression for Neighborhood Methods

- **The item-based approach learns item-item correlations.**

$$\hat{r}_{uj} = \frac{\sum_{k \in \mathcal{N}_j \cap \mathcal{I}_u} sim(k,j) * r_{uk}}{\sum_{k \in \mathcal{N}_j \cap \mathcal{I}_u} |sim(k,j)|} \quad \Rightarrow \quad \hat{r}_{uj} = \sum_{k \in \mathcal{N}_j \cap \mathcal{I}_u} w_{kj} \cdot r_{uk}$$

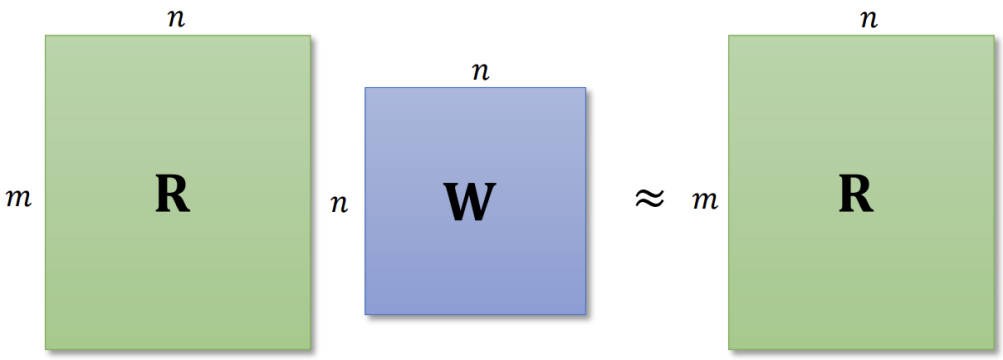
w_{kj}: item similarity coefficient between two items k and j

- **Minimize the following objective function.**

$$\sum_{j=1}^{n} \sum_{u \in \mathcal{U}_i} \left(r_{uj} - \left[\sum_{k \in \mathcal{N}_j \cap \mathcal{I}_u} w_{kj} \cdot r_{uk} \right] \right)^2$$

Sparse Linear Method(SLIM) → autoencoder모델로 발전

- Learning the **item-item similarity matrix W** by approximation the user-item rating matrix R



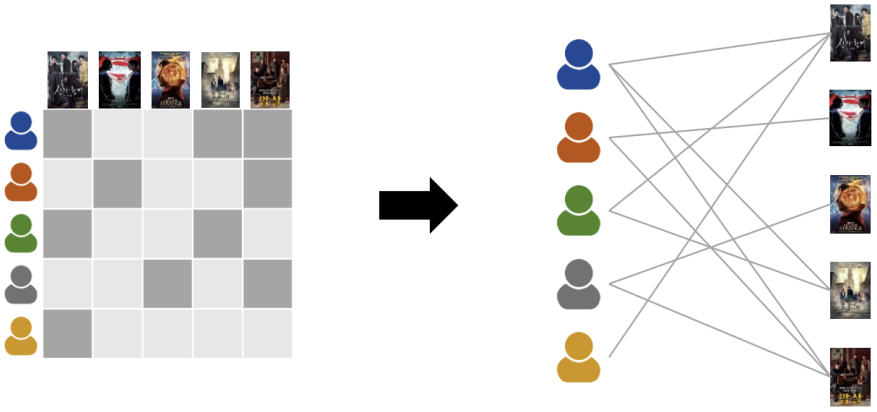
- However, without the constraint for diagonal elements of W, the solution can be trival.

Graph Models

- Various graph models are used to define the similarity with the use of either **structural transitivity** or **ranking methods.**
  - It can overcome the **data sparsity problem.**
- Graphs can be built on the users, on the items, or both.
  - E.g., user-item grpah, user-user graph, item-item graph.
- It exploits either **random-walk** or s**hortest-path** methods for recommendations.
- Recently, it has been widely used in neural recommender models, e.g., NGCF and LightGCN.

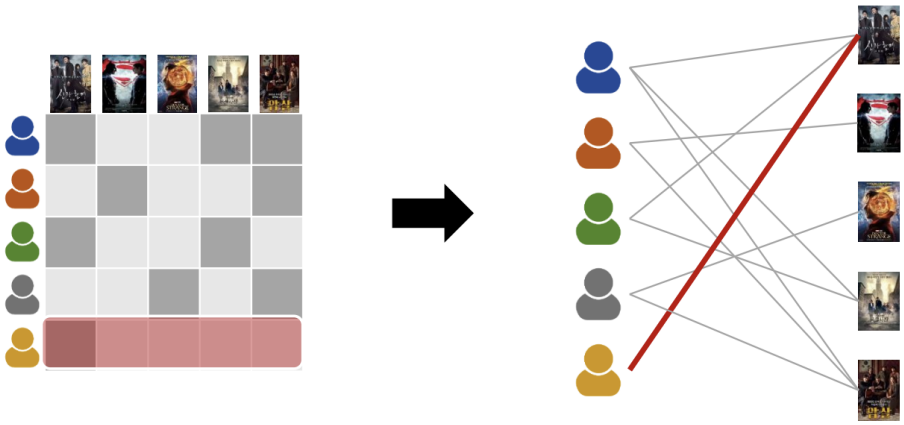
Example: Graph for User-Item Matrix

- A rating matrix can be represented by a bipartite graph.



- An edge represents user-item interaction.

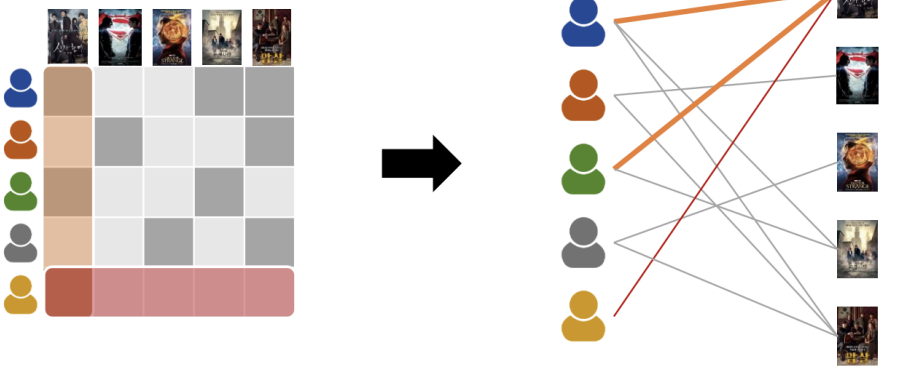
- For a user who has scarce feedback, the direct connectivity of the user provide insufficient information.



- To address this problem, we exploit high-order connectivity from user-item interactions.

High-order Connectivity

- The path reaches target node u from any node with the path length l larger than 1.

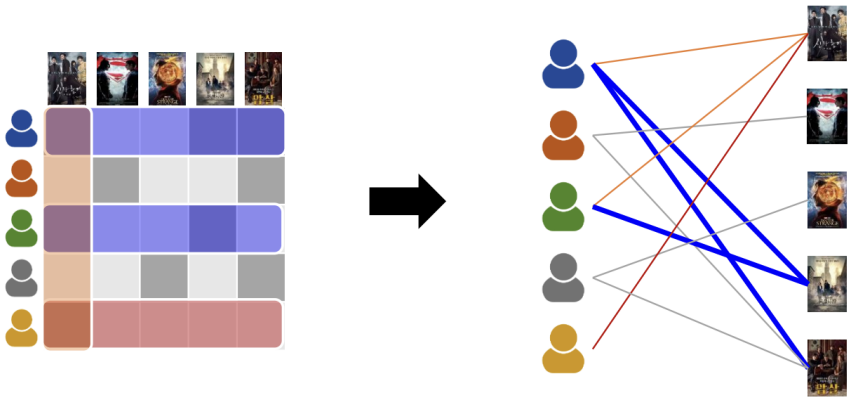


When l = 2, the target user also considers **other users**.

81

High-order Connectivity

- The path reaches target node u from any node with the path length l larger than 1.



When l = 3, the target user also considers **other items**.

82

MF(Latent Factor Model) → NCF → NGCF → LightGCN

SLIM → AutoEncoder → AutoRec → CDAE → RecVAE