20기 정규세션
ToBig's 19기 강의자

김민서

Recommender System Basic

Contents

Unit 01 | Intro to RS

Unit 02 | Memory based CF

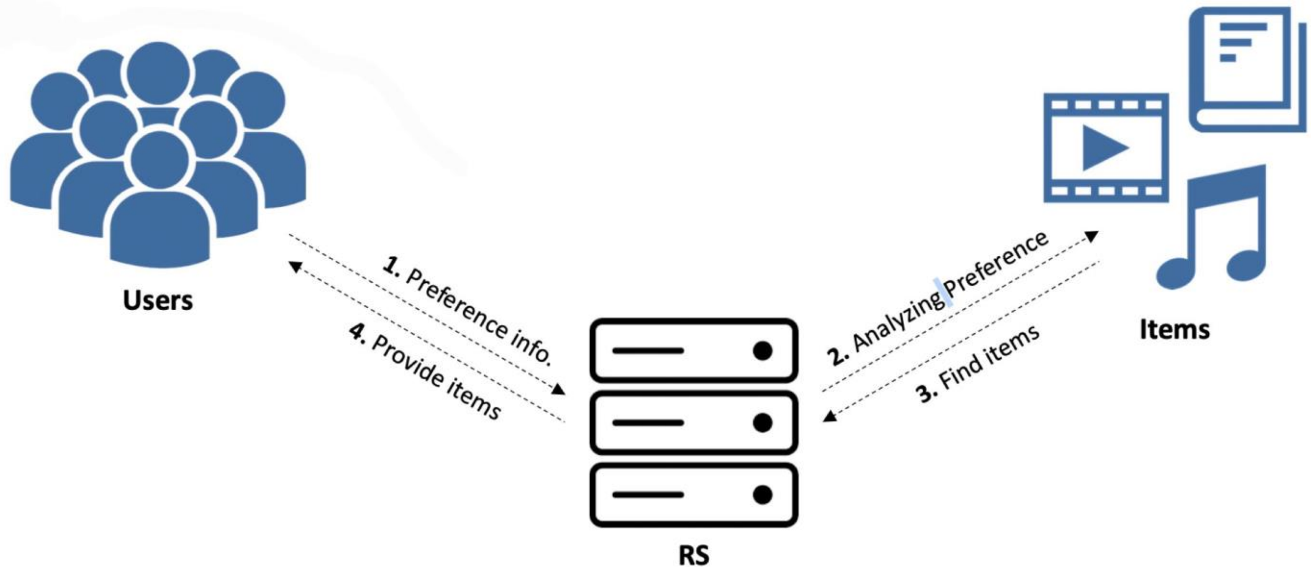
Unit 03 | Model based CF

Unit 04 | Evaluation

Main goal of RS

: To provide appropriate items to users according to their demands and preferences

- Items: movies, music, news, books, research articles, restaurants, historic place, grocery stores, and shopping malls etc.



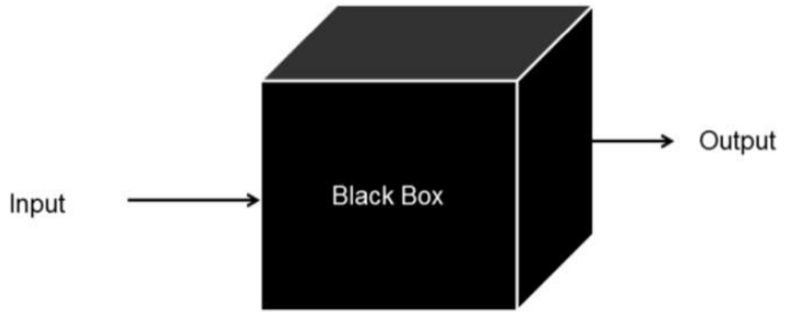
Why do we need RS?

- **The main problem users face on the web has gone from lack of information to information overload**
  - Information overload on the web, but users face lack of information
  - Users need help filtering out the noise (unnecessary information)

- **RS is a subclass of information filtering system**
  - RS provides the help by creating a filter

Which one you choose?

Model of RS



- RS seen as a function
- Given
  - User models
    : Explicit ratings, hidden user preferences, situational context
  - Item models
    : Descriptions of items, characteristics of items
- Find
  - Rating prediction. : Predict the ratings of unrated items.
  - Top-N recommendation. : Rank top-N items among unrated items.

Types of Data

- **Items info.**

  : the group of all items users have access to in the system

  - I (i_1,i_2, … , i_n)
- **Users info.**

  : the group of all available users in a recommender system

  - U (u_1,u_2, … , u_m)
- **Preference info. (Transactions)**

  : interaction or preference between a user and an item

  - f: $U \times I \rightarrow R$, where R indicates an entirely ordered set be a utility function such that $f(u_m, i_n)$ computes the usefulness of item i_n to user u_m.

  - E.g., Explicit feedback and Implicit feedback

Preference information

- **Preference information can be acquired explicitly or implicitly**
  - **Explicit info.** : typically by collecting users' ratings
    E.g., Explicit scale ratings (i.e., ranging from 1 to 5) and binary ratings (i.e., positive or negative)
  - **Implicit info.** : typically by monitoring users' behavior
    E.g., Songs heard, application downloaded, web sites visited and books read
  - **Tag** : a keyword assigned to a piece of information, and metadata to describe an item
    E.g., Bookmark and GPS locations

A user-item rating matrix

- **A user-item rating matrix Indicates the degree of preference of a user for items**
  - Ex. Rating r, Rating scale [0 – 5], 4 users $\{u_1, u_2, u_3, u_4\}$, 4 items $\{ i_1, i_2, i_3, i_4\}$
  - Null rating is 0
  - Positive ratings (r ≥ 4)

	i1	***i2***	***i3***	***i4***
u1	4	1	1	4
u2	1	4	2	0
u3	2	1	4	5
u4	1	4	0	1

Example of a user-item rating matrix

A user-item rating matrix

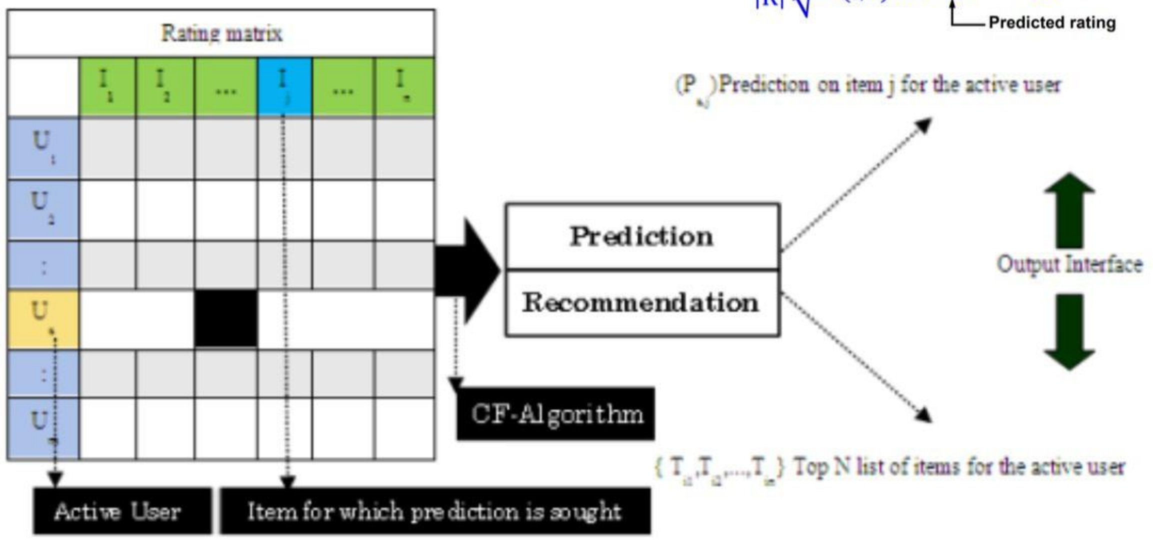
items (i_1 … i_8), users (u_1 … u_6)

	i_1							i_8
u_1	1		5			5		3
	5						4	
		3		3	4			
		2					4	5
	2		4	1		3		
u_6		3			2		5	

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

True rating of user x on item i

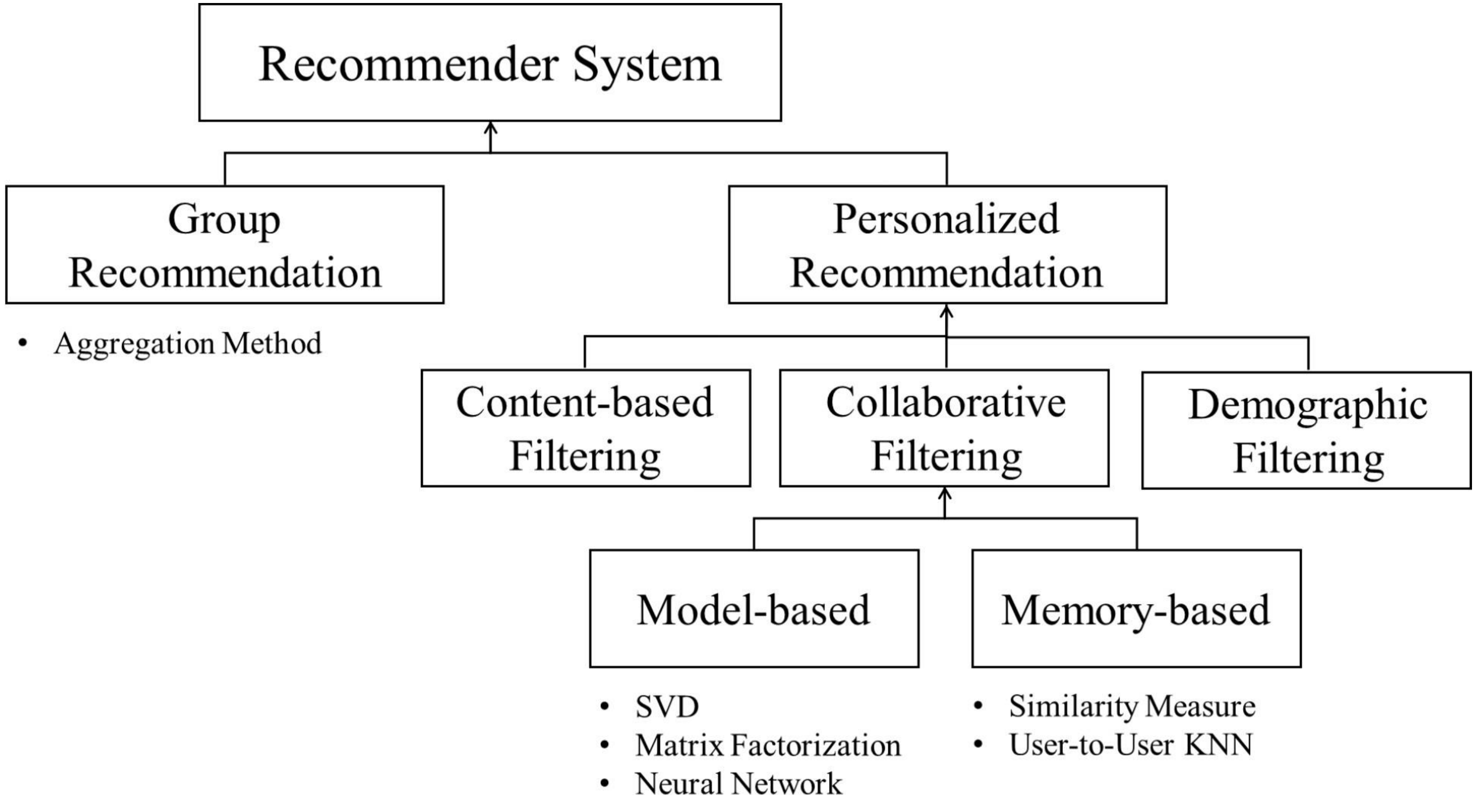
Predicted rating



The CF processes

https://www.researchgate.net/publication/287952023_Collaborative_Filtering_Recommender_Systems

Types of RS


Recommender System
Group Recommendation
• Aggregation Method
Personalized Recommendation
Content-based Filtering
Collaborative Filtering
Demographic Filtering
Model-based
Memory-based
• SVD
• Matrix Factorization
• Neural Network
• Similarity Measure
• User-to-User KNN

Group RS & Personalized RS

- **RS is largely classified into group and personalized recommendations**
- **Personalized RS** : Each user gets different recommendation results
- **Group RS** : A group that is a set of users with similar tendency gets different recommendation results

Group RS

- **The first thing to do**

  - To cluster a set of users having similar tastes into a group

- **Aggregation Method (AM)**

  - Most popular method in group RS

  - To aggregate the ratings of all users in a group

Users	Items				
	i_1	i_2	i_3	i_4	i_5
u_1	2	3	5	4	■
u_2	3	2	5	4	3
u_3	2	■	1	3	4

The users- to-items rating matrix

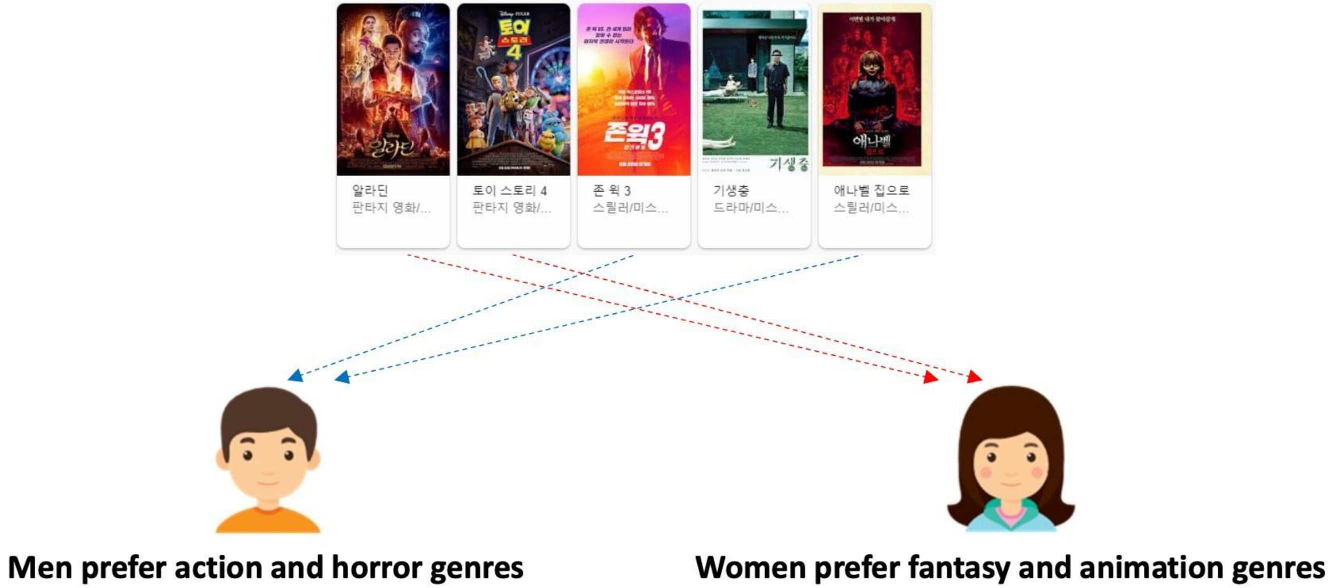
	items				
	i_1	i_2	i_3	i_4	i_5
AU	7	5	11	11	7
Mu	12	6	25	48	12
Avg	2.33	2.5	3.67	3.67	3.5

Examples of AMs

Personalized RS

- **Demographic Filtering**

  : To categorize the users based on common personal attributes (e.g., sex, age, country, etc.) and make recommendations based on demographic classes



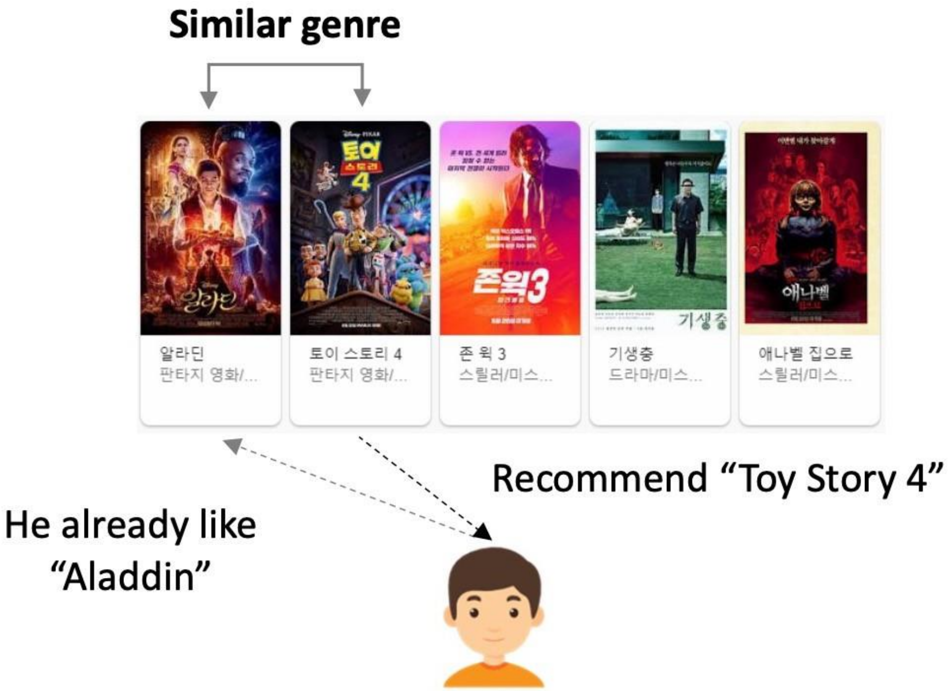
Personalized RS

- **Content-based Filtering**

  : Recommending items

  (1) with similar characteristics

  (2) that are similar to those in which the user has shown interest in the past



Personalized RS

- **Collaborative Filtering**

  : Recommending items to the user based on other individuals who are found to have similar preferences

Personalized RS

- **Collaborative Filtering**
  - Most personalized RSs are focusing on CF
  - Can be divided into memory-based and model-based approaches
  - **Memory-based** : mainly uses similarity metric between two users or items
    - **Item-based vs User-based**
    - E.g., Cosine similarity, Pearson correlation coefficient (PCC), and Jaccard
  - **Model-based** : uses a specific model to find the hidden features of user preference
    - E.g., Singular value decomposition (SVD), Matrix factorization (MF), and deep learning

Other RS

- **Context-based Recommendation**
  - Context-aware Recommendation System
  - Location-based Recommendation System
  - Real-time or Time-Sensitive Recommendation System
- **Community-based Recommendation**
  - Recommend based on the preferences of user's friends or community utilizing SNS network data
  - Several statistical and graph-based approaches have been applied
- **Knowledge-based Recommendation**
  - Recommend items based on particular domain knowledge
  - Cased-based Recommendation § Constraint-based Recommendation

Contents

Unit 01 | Intro to RS

Unit 02 | Memory based CF

Unit 03 | Model based CF

Unit 04 | Evaluation

Memory-based CF

- **Memory-based CF is mainly based on similarity measures**
  - First, must find K-nearest neighbor (KNN)
  - Ex. If we set K=5, then find 5-nearest neighbors (u1, u2, u4, u5, u10)

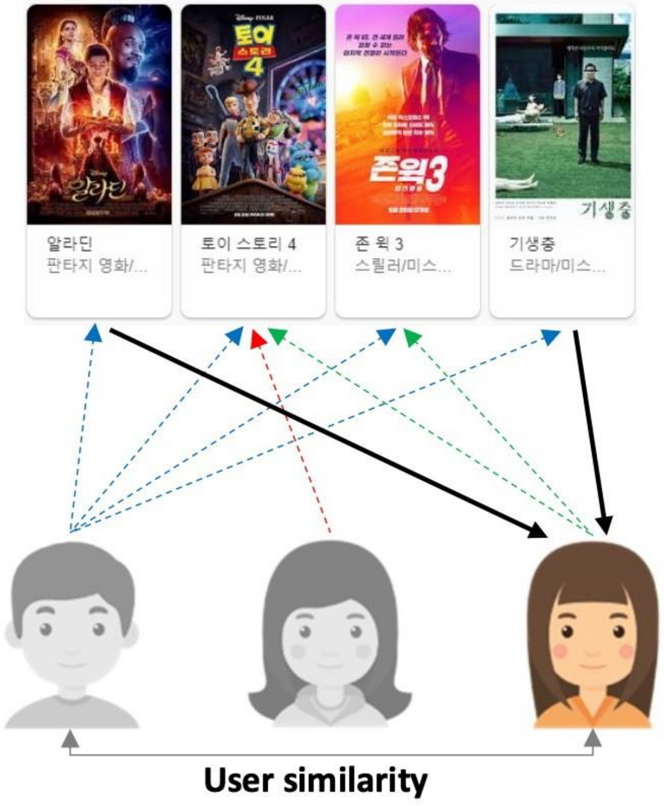
	u1	*u2*	*u3*	*u4*	*u5*	*u6*	*u7*	*u8*	*u9*	*u10*
u1	0.85	0.75	0.54	0.96	0.71	0.63	0.42	0.52	0.36	0.87

Example of user similarity

- Memory-based CF algorithm is calculated by their information
- Therefore, memory-based CF is also called **Neighborhood-based CF** or **KNN-based CF**

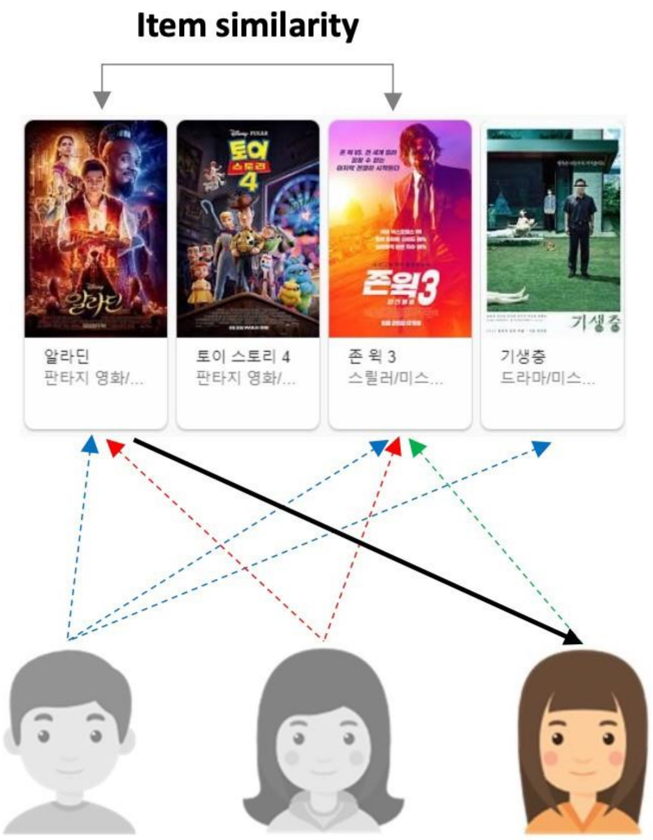
Memory-based CF

- **Consist of 2 methods**
  - **User-based CF** vs Item-based CF
- **In user-based CF**
  - Similar users which have similar ratings for similar items are found
  - Target user's rating for the item which target user has never interacted is predicted



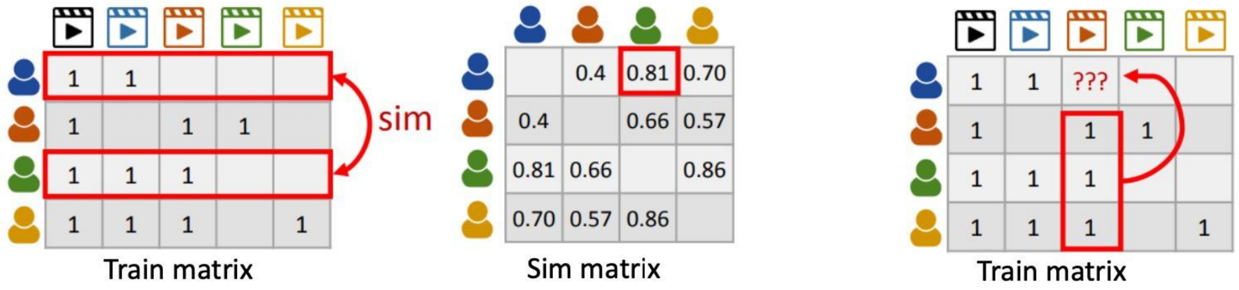
Memory-based CF

- **Consist of 2 methods**
  - User-based CF vs **Item-based CF**
- **In item-based CF**
  - uses the rating of co-rated item to predict the rating on specific item



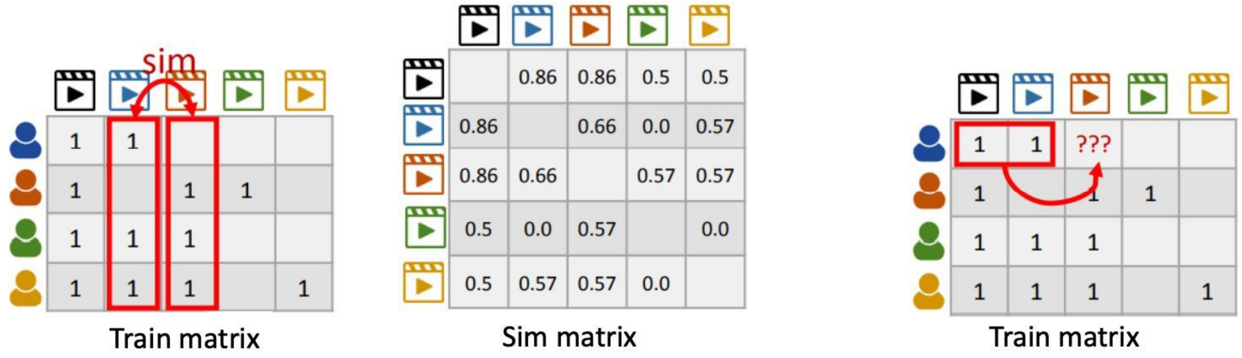
User-based CF

1. Similarity calculation using user viewing information
2. Predict the score using other users



Item-based CF

1. Similarity calculation using item viewing information
2. Predict the score using other items



Tasks of User-based CF

- **We have a database of ratings of the current user, Alice, and some other users.**
  - How do we measure the similarity between users?
  - How many neighbors do we consider?
  - How do we make a prediction from neighbors' ratings?



	Item1	Item2	Item3	Item4	Item5
Alice	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

Similarity Measure

Euclidean Distance

- **The ordinary straight-line distance between two vectors in Euclidean space**
  - $EUC(u,v) = \sqrt{(r_{u,1} - r_{v,1})^2 + (r_{u,2} - r_{v,2})^2 + \cdots + (r_{u,n} - r_{v,n})^2} = \sqrt{\sum_{i=1}^{n}(r_{u,i} - r_{v,i})^2}$
  - Null value is 0
  - $u = (4,1,1,4), \quad v = (2,0,4,5)$
  - $EUC(u,v) = \sqrt{(4-2)^2 + (1-0)^2 + (1-4)^2 + (4-5)^2} = \sqrt{4+1+9+1} = \sqrt{15} = 3.873$

	i1	*i2*	*i3*	*i4*
u1	4	1	1	4
u2	.	4	2	.
u3	2	.	4	5
u4	1	4	.	1

Jaccard Measure

- **The number of items in which both one user and the other have made a rating regarding the total number of items which have been rated between the two users**
  - $u = (4, 1, 1, 4) \rightarrow (1, 1, 1, 1), \quad v = (2, 0, 4, 5) \rightarrow (1, 0, 1, 1)$
  - $JAC(u, v) = {}^{|U \cap V|}/_{|U \cup V|} = {}^{|\{i_1, i_3, i_4\}|}/_{|\{i_1, i_2, i_3, i_4\}|} = {}^{3}/_{4}$
  - $u + v = (2, 1, 2, 2)$ means that #num (> 1) is intersection and #num (> 0) is union

	i1	***i2***	***i3***	***i4***
u1	4	1	1	4
u2	.	4	2	.
u3	2	.	4	5
u4	1	4	.	1

Mean-squared Distance

- **Similar to EUC, but Null values are excluded from the calculation**
  - $MSD(u,v) = {\sum_{\forall i \mid r_{u,i} \neq null \wedge r_{v,i} \neq null} (r_{u,i} - r_{v,i})^2} \Big/ {|\{\forall i \mid r_{u,i} \neq null \wedge r_{v,i} \neq null\}|}$
  - If one rating is Null value for two users u and v, then do not calculate that rating
  - The smaller the value, the closer the relationship
  - $u = (4,1,1,4),\ \ v = (2,0,4,5)$
  - $MSD(u,v) = \{(4-2)^2 + (1-4)^2 + (4-5)^2\}/3 = 14/3 = 4.667$

	i1	*i2*	*i3*	*i4*
u1	4	1	1	4
u2	.	4	2	.
u3	2	.	4	5
u4	1	4	.	1

Cosine Similarity

- **Most common similarity measure which calculates the cosine of the angle between two vectors**
  - $COS(u,v) = {}^{u \cdot v}/_{\|u\|\|v\|} = {}^{\sum_{i=1}^{n} r_{u,i} \cdot r_{v,i}} \Big/ _{\sqrt{\sum_{i=1}^{n} (r_{u,i})^2} \sqrt{\sum_{i=1}^{n} (r_{v,i})^2}}$
  - $u = (4, 1, 1, 4),\ v = (2, 0, 4, 5)$
  - $COS(u,v) = {}^{(4\times 2\ +\ 1\times 0\ +\ 1\times 4\ +\ 4\times 5)} \Big/ _{\sqrt{16+1+1+16}\sqrt{4+0+16+25}} = {}^{32}/_{39.1152144312159} = 0.8181$

	i1	*i2*	*i3*	*i4*
u1	4	1	1	4
u2	.	4	2	.
u3	2	.	4	5
u4	1	4	.	1

Pearson Correlation Coefficient

- **PCC is the most commonly used metric in RS**
  - $PCC(u,v) = \sum_{\forall i \mid r_{u,i} \neq null \wedge r_{v,i} \neq null} (r_{u,i} - \overline{r_u}) \cdot (r_{v,i} - \overline{r_v}) \Big/ \sqrt{\sum_{\forall i \mid r_{u,i} \neq null \wedge r_{v,i} \neq null} (r_{u,i} - \overline{r_u})^2} \sqrt{\sum_{\forall i \mid r_{u,i} \neq null \wedge r_{v,i} \neq null} (r_{v,i} - \overline{r_v})^2}$
  - Basic COS in item-based case has one important drawback-the difference in rating scale between different users are not taken into account
  - $\overline{r_u}$: the average of user u
  - $u' = (4 - 2.5, 1 - 2.5, 4 - 2.5) = (1.5, -1.5, 1.5),\ v' = (2 - 3.67, 4 - 3.67, 5 - 3.67) = (-1.67, 0.33, 1.33)$
  - $PCC(u,v) = {(1.5 \times (-1.67) + (-1.5) \times 0.33 + 1.5 \times 1.33)} \Big/ {\sqrt{1.5^2 + (-1.5)^2 + 1.5^2} \sqrt{(-1.67)^2 + 0.33^2 + 1.33^2}} = -0.1782$

	i1	*i2*	*i3*	*i4*	*avg*
u1	4	1	1	4	2.5
u2	.	4	2	.	3
u3	2	.	4	5	3.6667
u4	1	4	.	1	2

Memory-based CF Algorithm

CF Algorithm (KNNBasic)

$$p_{u,i} = \frac{\sum_{v \in N_i^k(u)} sim(u,v)\, r_{v,i}}{\sum_{v \in N_i^k(u)} sim(u,v)}$$

- u is a target user
- $r_{v,i}$: the true rating of a user v for an item i
- $p_{u,i}$: the predicted rating of a target user u for an item i
- $N_i^k(u)$: the k nearest neighbors of user u that have rated item i

CF Algorithm with Mean (KNNwithMeans)

- **A basic CF algorithm, taking into account the mean ratings of each user**

$$p_{u,i} = \mu_u + \frac{\sum_{v \in N_i^k(u)} sim(u,v)\,(r_{v,i} - \mu_v)}{\sum_{v \in N_i^k(u)} sim(u,v)}$$

- u is a target user
- $r_{v,i}$: the true rating of a user v for an item i
- $p_{u,i}$: the predicted rating of a target user u for an item i
- $N_i^k(u)$: the k nearest neighbors of user u that have rated item i
- μ_u: the mean of all ratings given by user u

CF Algorithm with z-score(KNNwithZScore)

- **A basic CF algorithm, taking into account the z-score normalization of each user**

$$p_{u,i} = \mu_u + \sigma_u \frac{\sum_{v \in N_i^k(u)} sim(u,v)\,(r_{v,i} - \mu_v)/\sigma_v}{\sum_{v \in N_i^k(u)} sim(u,v)}$$

- u is a target user
- $r_{v,i}$: the true rating of a user v for an item i
- $p_{u,i}$: the predicted rating of a target user u for an item i
- $N_i^k(u)$: the k nearest neighbors of user u that have rated item i
- μ_u: the mean of all ratings given by user u
- σ_u: the standard deviation of all ratings given by user u

CF Algorithm with Baseline Rating (KNNBaseline)

- **A basic CF algorithm, taking into account a baseline rating**

$$p_{u,i} = b_{u,i} + \frac{\sum_{v \in N_i^k(u)} sim(u,v)(r_{v,i} - b_{v,i})}{\sum_{v \in N_i^k(u)} sim(u,v)} \text{ or } p_{u,i} = b_{u,i} + \frac{\sum_{j \in N_u^k(i)} sim(i,j)(r_{u,j} - b_{u,j})}{\sum_{j \in N_u^k(i)} sim(i,j)}$$

$$b_{u,i} = \mu + b_u + b_i$$

- μ: the mean of all ratings
- $b_u = \mu_u - \mu$
- $b_i = \mu_i - \mu$

Summary

1.Measuring Similarity between Users

- **Mean-centering**

  - Normalize rated items with average.

  - 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
User1	3 (+1.0)	1 (-1.0)	2 (0.0)	2 (0.0)	2 (0.0)	2
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

1.Measuring Similarity between Users

- **Pearson correlation coefficient**

S_{xy} is a set of items that are co-rated by users x and y.

$\bar{r_x}$ is the average of rated items for user x.

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

	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)

$sim(Bob,U1)$	1.0
$sim(Bob,U2)$	0.60
$sim(Bob,U3)$	0.0
$sim(Bob,U4)$	-0.77

2. Selecting Neighbors

- **How to select neighborhoods**
  - All the neighbors
  - Random neighbors
  - # of neighborhoods vs. thresholds
    - Select neighbors with positive similarities.
    - Select top-N 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)	$sim(Bob, U1)$	1.0
User2	4 (+0.2)	3 (-0.8)	4 (+0.2)	3 (-0.8)	5 (+1.2)	$sim(Bob, U2)$	0.60
User3	3 (-0.2)	3 (-0.2)	1 (-2.2)	5 (+1.8)	4 (0.8)	$sim(Bob, U3)$	0.0
User4	1 (-1.8)	5 (+2.2)	5 (+2.2)	2 (-0.8)	1 (-1.8)	$sim(Bob, U4)$	-0.77

3. Making Prediction

- **How to predict the ratings of an item**
  - Calculate the weighted average from neighbors.
  - Add the average of active user.

$$pred(a,p) = \overline{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{b \in N} sim(a,b)}$$

	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)	$sim(Bob, U1)$	1.0
User2	4 (+0.2)	3 (-0.8)	4 (+0.2)	3 (-0.8)	5 (+1.2)	$sim(Bob, U2)$	0.60
User3	3 (-0.2)	3 (-0.2)	1 (-2.2)	5 (+1.8)	4 (0.8)	$sim(Bob, U3)$	0.0
User4	1 (-1.8)	5 (+2.2)	5 (+2.2)	2 (-0.8)	1 (-1.8)	$sim(Bob, U4)$	-0.77

3. Making Prediction

- **How to predict the ratings of an item**
  - Calculate the weighted average from neighbors.
  - Add the average of active user.

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

	Item1	Item2	Item3	Item4	Item5		
Bob	5 (+1.0)	3 (-1.0)	4 (0.0)	4 (0.0)	???	Top-2 = {U1, U2}	
User1	3 (+1.0)	1 (-1.0)	2 (0.0)	2 (0.0)	2 (0.0)	$sim(Bob, U1)$	1.0
User2	4 (+0.2)	3 (-0.8)	4 (+0.2)	3 (-0.8)	5 (+1.2)	$sim(Bob, U2)$	0.60
User3	3 (-0.2)	3 (-0.2)	1 (-2.2)	5 (+1.8)	4 (0.8)	$sim(Bob, U3)$	0.0
User4	1 (-1.8)	5 (+2.2)	5 (+2.2)	2 (-0.8)	1 (-1.8)	$sim(Bob, U4)$	-0.77

Pros/Cons of Memory-based CF

- **Advantages**
  - The explainability of the results
  - Easy creation and use
  - Easy facilitation of new data
  - Content-independence of the items
  - Good scaling with co-rated items

- **Disadvantages**
  - Performance decreases when data gets sparse
  - Low scalability for large datasets
  - Much time to calculate similarity
  - Lower performance than model-based CF

Contents

Unit 01 | Intro to RS

Unit 02 | Memory based CF

Unit 03 | Model based CF

Unit 04 | Evaluation

Model-based CF

- **Also called latent factor model**

  - Representing users and items as vectors (or features) of the same dimension

    - User vector & Item vector

    - Dimension is lower than both the number of users and items

Latent Factor

영화 \ 사용자	Alice	Bob	Carol	Dave
뷰티인사이드	5	5	1	1
라라랜드	5	4	1	1
러브스토리	5	5	1	1
매트릭스	1	1	5	5
스타워즈	1	1	5	4

영화 \ 특징	로맨틱 영화	공상과학 영화
뷰티인사이드	높은 점수	낮은 점수
라라랜드	높은 점수	낮은 점수
러브스토리	높은 점수	낮은 점수
매트릭스	낮은 점수	높은 점수
스타워즈	낮은 점수	높은 점수

특징 \ 사용자	Alice	Bob	Carol	Dave
로맨틱한 사람	높은 점수	높은 점수	낮은 점수	낮은 점수
상상력이 풍부한 사람	낮은 점수	낮은 점수	높은 점수	높은 점수

Model-based CF

- **The most successful realizations of model-based CF methods are based on matrix factorization (MF)**

  - Characterizing both users and items by vectors (i.e., features) inferred from item rating patterns

  - High correspondence between item and user vectors leads to a recommendation

  - Good scalability with predictive accuracy

  - Faster than memory-based CF (because of the lower dimension of the users and items vectors)

  - Good flexibility

  - Allowing incorporation of additional information

    - Utilizing implicit information (e.g., purchase history, browsing history, or search patterns) is easier than memory-based CF

Model-based CF methods

$$AA^T = U(\Sigma\Sigma^T)U^T$$
$$A^TA = V(\Sigma^T\Sigma)V^T$$

SVD

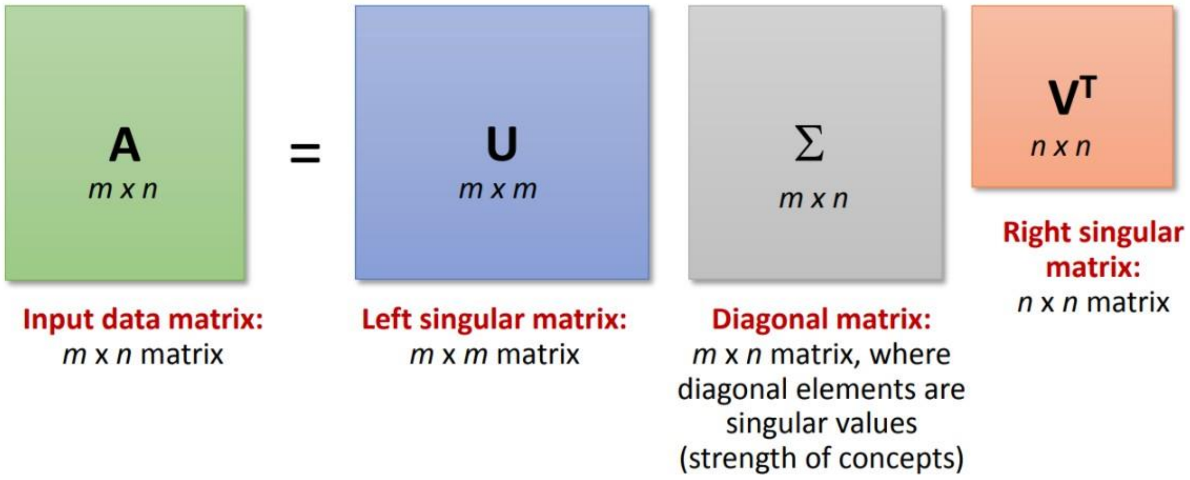
- **MF is closely related to singular value decomposition (SVD)**

- Well-established technique for identifying latent semantic factors in information retrieval

$$A_{n\times m} = U_{n\times n}\Sigma_{n\times m}V^T_{m\times m}$$

$$\Sigma=\begin{pmatrix}\sigma_1 & & \\ & \ddots & \\ & & \sigma_s \\ & & 0\end{pmatrix}\ (m>n)\ \ or\ \ \Sigma=\begin{pmatrix}\sigma_1 & & & \\ & \ddots & & \\ & & \sigma_s & 0\end{pmatrix}\ (m<n)$$

- $A_{n\times m}$: original matrix (n users & m items)
- $U_{n\times n}$: left singular vector of A (orthogonal matrix)
- $V^T_{m\times m}$: right singular vector of A (orthogonal matrix)
- $\Sigma_{n\times m}$: singular values of A (diagonal matrix)



Properties of SVD

- **Always possible to decompose a matrix A into A = $\mathbf{U\Sigma}\boldsymbol{VT}$**

  - U,Σ,V : unique

  - U, V : column **orthonormal**

    - $u^T u = I, V^T V = I$ (I : identity matrix)

    - Columns are orthogonal unit vectors.

  - Σ : **diagonal**

    - Entries (singular values) are positive.

    - Sorted in decreasing order ($\sigma\# \geq \sigma\% \geq \cdots \geq 0$)

$$\Sigma = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_s \\ & & 0 \end{pmatrix} \; (m>n) \;\; or \;\; \Sigma = \begin{pmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_s & 0 \end{pmatrix} \; (m<n)$$

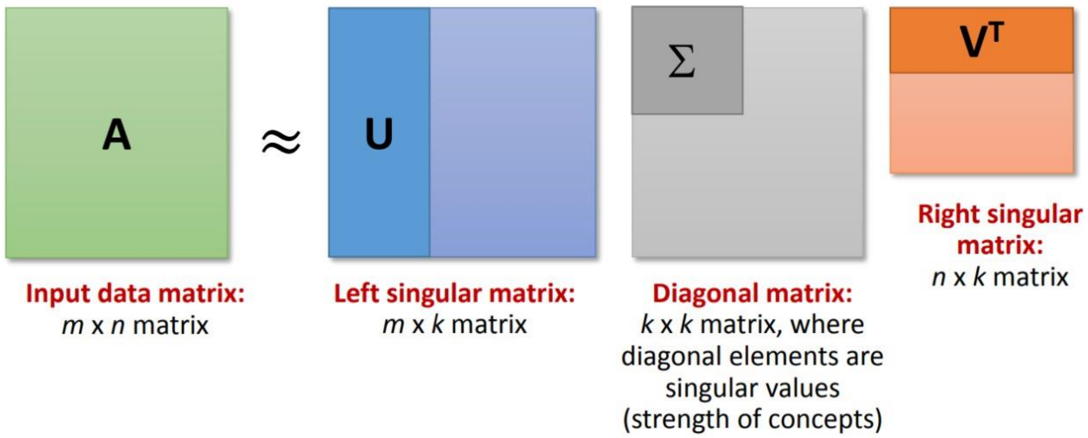
Reduced SVD

- **Thin / Compact SVD**
- **Truncated SVD**
- **Reduce the dimension of U and V**

$$\hat{A}_{n\times m} = U_{n\times c}\Sigma_{c\times c}V^T_{c\times m}$$

approximation matrix of A

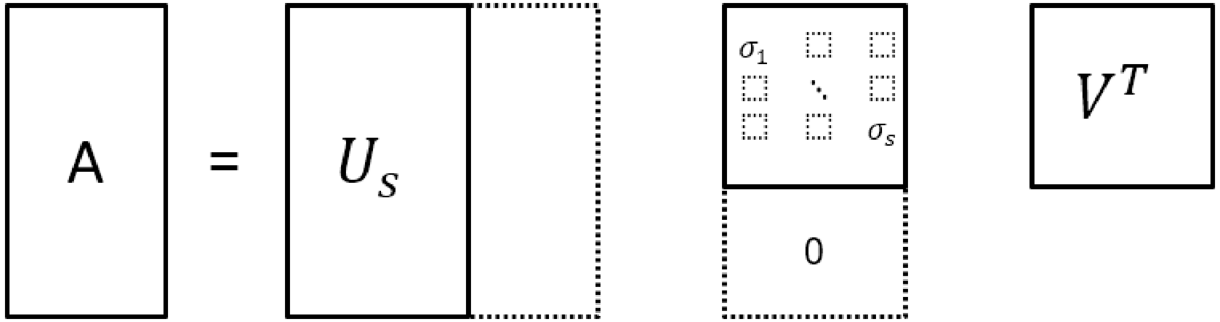
- $\hat{A}_{n\times m}$: approximation matrix of A
- $U_{n\times c}$: left singular vector of $\hat{A}$ (orthogonal matrix)
- $V^T_{c\times m}$: right singular vector of $\hat{A}$ (orthogonal matrix)
- $\Sigma_{c\times c}$: singular values of $\hat{A}$ (diagonal matrix)
- **Reduce n (original dimension of U) and m (original dimension of V) to lower dimension of c**



Reduced SVD

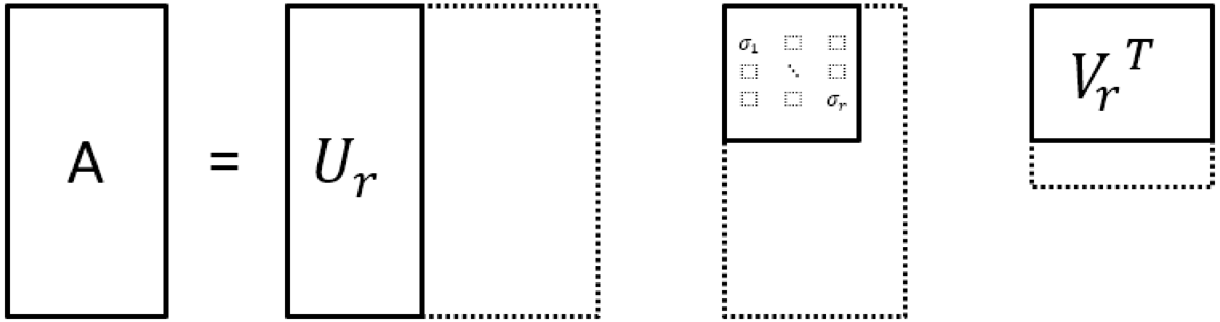
assume that among the s diagonal elements in Σ, the number of non-zero elements is r (s ≥ r)

Thin SVD



The form obtained by removing the non-diagonal parts composed of zeros from Σ and removing the column vectors corresponding to them from U.

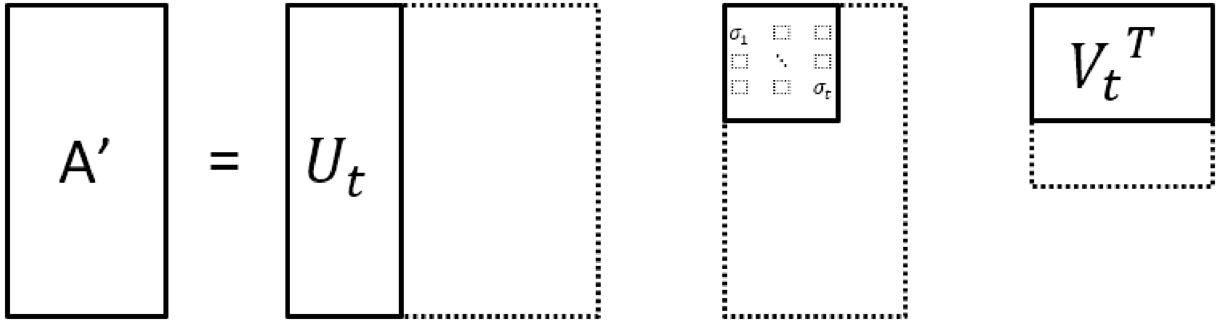
Compact SVD



The form obtained by removing all singular values that are equal to zero

Reduced SVD

Truncated SVD



An approximate matrix A’ for A, excluding singular values in the sigma matrix that are not equal to zero, which is not the complete A matrix.

Limitations in SVD

- **Challenge #1 : missing values**

  - Filling the missing values with user average may be not accurate

- **Challenge #2 : scalability**

  - SVD computation is O($m^n n + n^3$)

    : can't scale well to large data set

  - Reduce dimensionality of problems

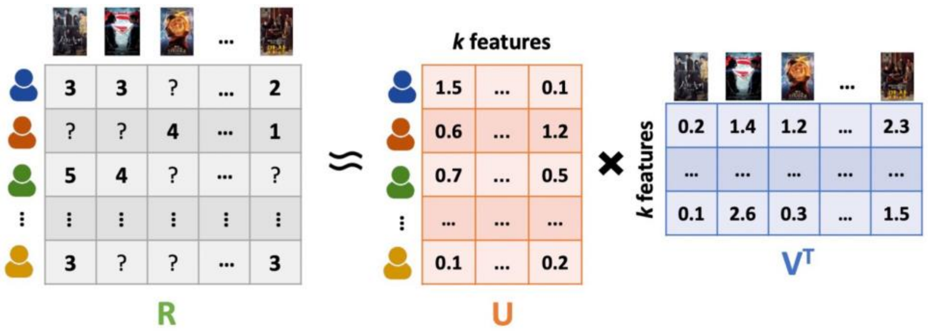
- **Challenge #3 : lack of transparency / explainability**

  - Optimal dimensions do not correspond to latent features

Factorizing Two Latent Matrices

- **The user-item rating matrix R can be approximated as a linear combination of two latent matrices U and V.**
  - R: user-item rating matrix ($\boldsymbol{m} \times \boldsymbol{n}$ matrix)
  - U: latent user matrix ($\boldsymbol{m} \times \boldsymbol{k}$ matrix)
    - Each user is represented by a latent vector ($1 \times k$ vector).
  - V: latent item matrix ($\boldsymbol{n} \times \boldsymbol{k}$ matrix)
    - Each item is represented by a latent vector ($1 \times k$ vector).
    - k : # of latent features

$$\hat{r}_{u,i} = U_u V_i^T$$



Factorizing Two Latent Matrices

- **Factorize a rating matrix into two latent matrices.**
  - The matrix R can be approximated as a product of "thin" $U \cdot V^T$
  - We DO NOT care about the values on the missing ones

$$\hat{r}_{u,i} = U_u V_i^T$$

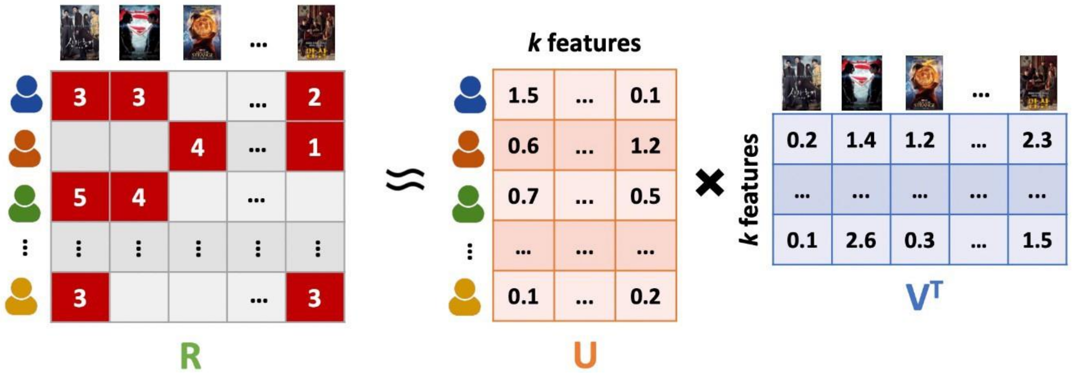
$$argmin_{U,V} \sum_{u=1}^{m} \sum_{i=1}^{n} y_{ui}(r_{ui} - U_u V_i^T)^2$$

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

$U_u : u - th\ row\ of\ U$

$V_i : i - th\ column\ of\ V$

$\hat{r}_{u,i}$: predicted rating of user u for item i



Learning method (Learning Two Latent Matrices)

- **By using stochastic gradient descent (SGD) or alternating least square (ALS), we learn U and V iteratively.**
  - The goodness of fit is to reduce the prediction error.
  - The regularization term is used to alleviate the overfitting problem

$$argmin_{U,V} \underbrace{\sum_{u=1}^{m}\sum_{i=1}^{n} y_{u,i}\left(r_{u,i} - \hat{r}_{u,i}\right)^2}_{\text{Goodness of fit}} + \underbrace{\lambda(\|U_u\|^2 + \|V_i\|^2)}_{\text{Regularization}}$$

- $where\ y_{ui} = \begin{cases} 1 & if\ r_{ui}\ exists \\ 0 & otherwise \end{cases}.$
- $U_u: u - th\ row\ of\ U$
- $V_i: i - th\ column\ of\ V$
- $\hat{r}_{u,i}$: predicted rating of user u for item i
- λ: regularization term

Overall procedure

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

They are the partial derivative of U_u and V_i.

Learning method (Learning Two Latent Matrices)

For $\hat{r}_{ij} = p_i^T q_j = \sum_{k=1}^{K} p_{ik} q_{kj}$,
Define $e_{ij} = (r_{ij} - \hat{r}_{ij})$

We can compute the gradient by taking partial derivatives with respect to both p and q.
▶ $\frac{\partial}{\partial p_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij}) q_{kj} = -2 e_{ij} q_{kj}$
▶ $\frac{\partial}{\partial q_{ik}} e_{ij}^2 = -2(r_{ij} - \hat{r}_{ij}) p_{ik} = -2 e_{ij} p_{ik}$
Now, update p and q using the gradient obtained through differentiation to reduce the error during the learning process

To preventing overfitting, a regularization term has been added

$$\boxed{argmin_{U,V} \sum_{u=1}^{m} \sum_{i=1}^{n} y_{u,i} \left(r_{u,i} - \hat{r}_{u,i}\right)^2 + \lambda(\|U_u\|^2 + \|V_i\|^2)}$$

Learning Method

- **By ALS**

: alternating between holding one matrix fixed and solves the least-squares problem

Algorithm 1 ALS for Matrix Completion

Initialize X, Y
repeat
  for $u = 1 \dots n$ **do**

$$x_u = \Big(\sum_{r_{ui} \in r_{u*}} y_i y_i^{\intercal} + \lambda I_k\Big)^{-1} \sum_{r_{ui} \in r_{u*}} r_{ui} y_i \tag{2}$$

  end for
  for $i = 1 \dots m$ **do**

$$y_i = \Big(\sum_{r_{ui} \in r_{*i}} x_u x_u^{\intercal} + \lambda I_k\Big)^{-1} \sum_{r_{ui} \in r_{*i}} r_{ui} x_u \tag{3}$$

  end for
until convergence

Learning Method

- **By ALS**

: alternating between holding one matrix fixed and solves the least-squares problem

$$\underset{U,V}{\operatorname{argmin}} \sum_{observed\, r_{u,i}} (r_{u,i} - U_u V_i^T)^2 + \lambda(\|U_u\|^2 + \|V_i\|^2)$$

$$\underset{p_u}{\operatorname{argmin}} \|r_u - Vp_u\|^2 + \lambda\|p_u\|^2$$

$$\underset{q_i}{\operatorname{argmin}} \|r_i - Uq_i\|^2 + \lambda\|q_i\|^2$$

→ becomes a linear regression problem

Modeling User and Item biases

- Not consider bias which is the observed deviations of users and items
- Bias affects the recommendation results a lot

$$\hat{r}_{u,i} = \mu + b_u + b_i + U_u V_i^T$$

$$argmin_{U,V} \sum_{u=1}^{m} \sum_{i=1}^{n} y_{u,i} \left(r_{u,i} - \hat{r}_{u,i}\right)^2 + \lambda \left({b_u}^2 + {b_i}^2 + \|U_u\|^2 + \|V_i\|^2\right)$$

- μ: global average for all ratings
- $b_u = \mu_u - \mu$: user bias for ratings
- $b_i = \mu_i - \mu$: item bias for ratings

SVD++

- **can apply implicit information to the MF.**
  - Taking into account implicit information

$$\hat{r}_{u,i} = \mu + b_u + b_i + q_i^T(p_u + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j)$$

$$argmin_{U,V} \sum_{u=1}^{m} \sum_{i=1}^{n} y_{u,i}(r_{u,i} - \hat{r}_{u,i})^2 + \lambda(b_u{}^2 + b_i{}^2 + \|U_u\|^2 + \|V_i\|^2)$$

  - I_u: the set of all items rated by user u
  - y_j: A new set of item factors that capture implicit information

$$y_j = \begin{cases} 1 & \boldsymbol{if}\ user\ u\ rated\ an\ item\ j \\ 0 & \boldsymbol{else} \end{cases}$$

Summary

Unit 03 | Model-based CF

- **Step 1 : Initialize U and V**

U

Bob	-0.008	0.021	0.029
$u1$	-0.001	0.039	0.022
$u2$	-0.008	0.029	-0.009
$u3$	-0.01	0.016	-0.012
$u4$	-0.035	0.012	-0.049

V^T

$i1$	$i2$	$i3$	$i4$	$i5$	$i6$	$i7$	$i8$
-0.02	-0.005	-0.003	0.005	-0.013	-0.016	-0.023	-0.017
0.022	0.028	0.028	0.018	0.005	-0.001	0.028	0.005
0.026	0.021	0.006	-0.005	-0.027	-0.023	-0.018	-0.026

- **Step 2 : Predict based MF**

UV^T

	$i1$	$i2$	$i3$	$i4$	$i5$	$i6$	$i7$	$i8$
Bob	0.001	0.001	0.0	0.0	0.0	0.0	0.0	0.0
$u1$	0.001	0.001	0.001	0.0	0.0	0.0	0.0	0.0
$u2$	0.0	0.0	0.0	0.0	0.0	0.0	0.001	0.0
$u3$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$u4$	0.0	0.0	0.0	0.0	0.001	0.001	0.002	0.001

=

U

Bob	-0.008	0.021	0.029
$u1$	-0.001	0.039	0.022
$u2$	-0.008	0.029	-0.009
$u3$	-0.01	0.016	-0.012
$u4$	-0.035	0.012	-0.049

×

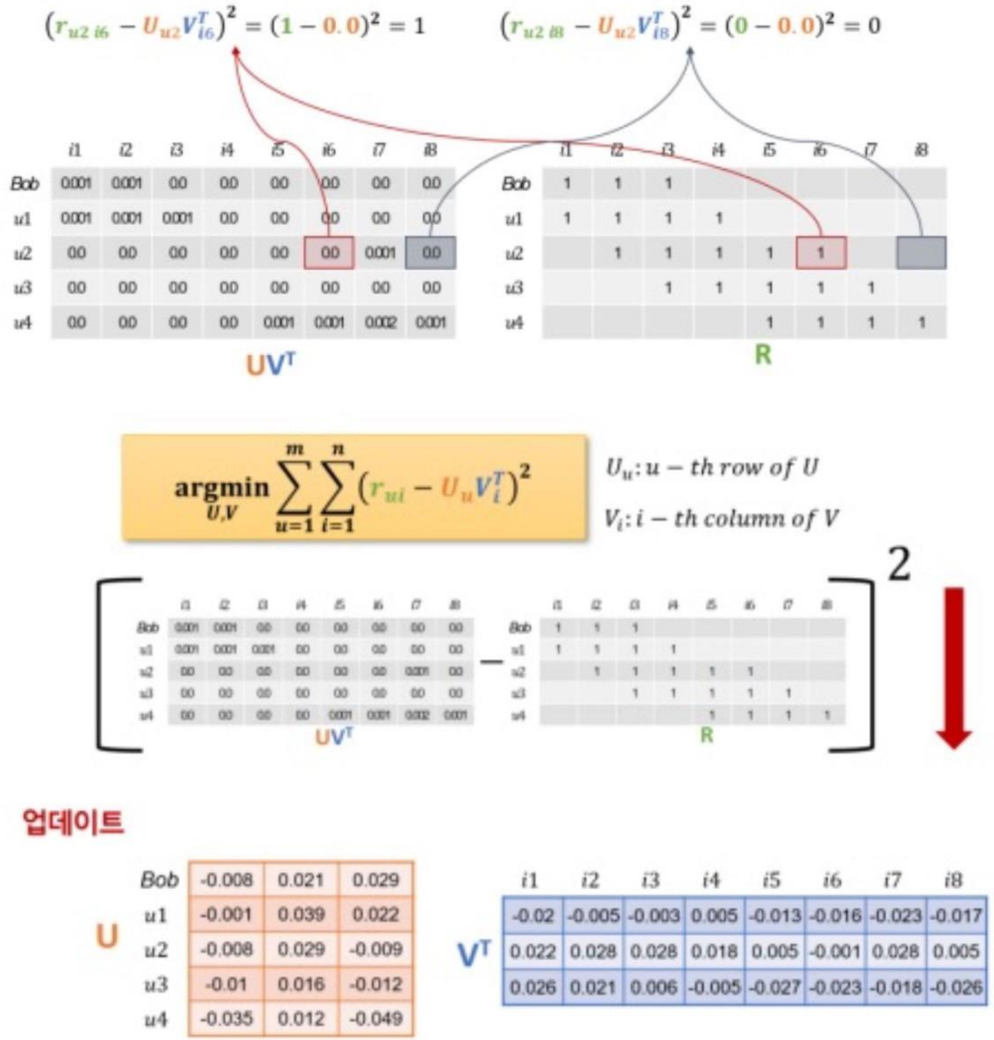
V^T

$i1$	$i2$	$i3$	$i4$	$i5$	$i6$	$i7$	$i8$
-0.02	-0.005	-0.003	0.005	-0.013	-0.016	-0.023	-0.017
0.022	0.028	0.028	0.018	0.005	-0.001	0.028	0.005
0.026	0.021	0.006	-0.005	-0.027	-0.023	-0.018	-0.026

Unit 03 | Model-based CF

- **Step 3 : Find the difference between the predicted and actual values**

- **Step 4 : Update U and V to reduce overall difference**



Contents

Unit 01 | Intro to RS

Unit 02 | Memory based CF

Unit 03 | Model based CF

Unit 04 | Evaluation

RMSE & MAE

- Root Mean Squared Error (RMSE)

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

- Mean Absolute Error (MAE)

$$MAE = \frac{1}{|\hat{R}|} \sum_{\hat{r}_{u,i} \in \hat{R}} \left|r_{u,i} - \hat{r}_{u,i}\right|$$

	i1	i2	i3	I4
$r_{u,i}$	2	1	4	5
$\hat{r}_{u,i}$	3.23	2.13	3.12	4.58
$\lvert r_{u,i} - \hat{r}_{u,i}\rvert$	1.23	1.13	0.88	0.42

Precision@K

- **The proportion of relevant recommended items from the total number of recommended items**

$$Precision = \frac{1}{|U|}\sum_{u\in U}\frac{\left|\{i \in Z_u | r_{u,i} \geq \theta\}\right|}{n}$$

Z_u: the set of n recommendations to user u

Recall@K

- **The proportion of relevant recommended items from the number of relevant items**

$$Recall = \frac{1}{|U|}\sum_{u\in U}\frac{\left|\{i \in Z_u | r_{u,i} \geq \theta\}\right|}{\left|\{i \in Z_u | r_{u,i} \geq \theta\}\right| + \left|\{i \in Z_u^c | r_{u,i} \geq \theta\}\right|}$$

NDCG (Normalized Discounted Cumulative Gain)

- **In ranking task**
- **DCG is the total gain accumulated at a particular rank k**

$$DCG_k = \frac{1}{|U|}\sum_{u\in U}(r_{u,p_1} + \sum_{i=2}^{k}\frac{r_{u,p_i}}{log_2 i})$$

$$NDCG_k = \frac{DCG_k}{IDCG_k}$$

- $p_1, p_2, \ldots, p_k$: a list of recommended items
- $r_{u,p_1}, r_{u,p_2}, \ldots, r_{u,p_k}$: the original rating of the user u for the p_i-st item
- $IDCG_k$: ideal DCG

NDCG

- **Example**

$$DCG_k = \frac{1}{|U|}\sum_{u \in U}(r_{u,\hat{r}_1} + \sum_{i=2}^{k}\frac{r_{u,\hat{r}_i}}{log_2 i})$$

$$NDCG_k = \frac{DCG_k}{IDCG_k}$$

- $p_1, p_2, \ldots, p_k$: a list of recommended items
- $r_{u,p_1}, r_{u,p_2}, \ldots, r_{u,p_k}$: the original rating of the user u for the p_i-st item
- $IDCG_k$: ideal DCG

	i1	i2	i3	i4
$r_{u,i}$	2	1	4	5
$\hat{r}_{u,i}$	3.23	2.13	3.12	4.58

- $p_1 = i_4,\ p_2 = i_1,\ p_3 = i_3,\ p_4 = i_2$
- $r_{u,p_1} = 5,\ r_{u,p_2} = 2,\ r_{u,p_3} = 4,\ r_{u,p_4} = 1$
- $DCG_4 = 5 + \frac{2}{log_2 2} + \frac{4}{log_2 3} + \frac{1}{log_2 4} = 10.0237$
- $IDCG_4 = 5 + \frac{4}{log_2 2} + \frac{2}{log_2 3} + \frac{1}{log_2 4} = 10.7619$
- $NDCG_4 = \frac{DCG_4}{IDCG_4} = 0.9314$

Referenced

참고자료

- 투빅스 16기 이승주님 강의자료
- 성균관대 이종욱 교수님 추천시스템 강의 자료
- 인하대 서영덕 교수님 추천시스템 강의 자료
- ToBig's 16기 정규세션 RS 강의 (강지우님)
- T아카데미, 추천시스템 분석 경진대회 참여하기, 김현우
- [선형대수학#4] 특이값 분해(Singular Value Decomposition, SVD)의 활용, 다크 프로그래머
- ALS Implicit Collaborative Filtering
- [논문 리뷰] BPR: Bayesian Personalized Ranking from Implicit Feedback , HyeJinLee
- BPR: Bayesian Personalized Ranking from Implicit Feedback, 기내식은수박바
- [추천시스템][paper review][구현] Factorization Machines, RISING FASTBALL
- https://velog.io/@sangyun/
- https://skettee.github.io/post/latent_factor_model/
- https://sungkee-book.tistory.com/11

Assignment

과제

https://grouplens.org/datasets/movielens/latest/

1. 파이썬을 이용하여 무비렌즈 데이터(ml-latest-small)를 전처리 및 EDA 진행해주세요.

- 유의미한 분석 5개 이상

2. Memory based CF 2개 이상, Model based CF 1개 이상 알고리즘을 사용해서 아래 결과를 출력해주세요

- 성능 평가
- 추천 결과 출력

+ Surprise library 사용 가능

`surprise` 라이브러리를 설치한다. `pip` 와 `conda` 모두 지원한다.

아나콘다를 사용할 경우 Anaconda Prompt에 다음 명령을 입력한다.

```
conda install -c conda-forge scikit-surprise
```

아나콘다를 사용하지 않거나, `conda` 가 작동하지 않을 경우 `pip` 로 설치한다.

```
pip install scikit-surprise
```

Q & A

들어주셔서 감사합니다.