COLLABORATIVE FILTERING

Readings

- Book Chapters
  - Chapter 10 Information Retrieval in Practice   or
  - Chapter 9 in Mining of Massive Datasets

- (Netflix) http://www.netflixprize.com/assets/ProgressPrize2008_BellKor.pdf

- (Netflix article) http://www.nytimes.com/2008/11/23/magazine/23Netflix-t.html?pagewanted=all&_r=1

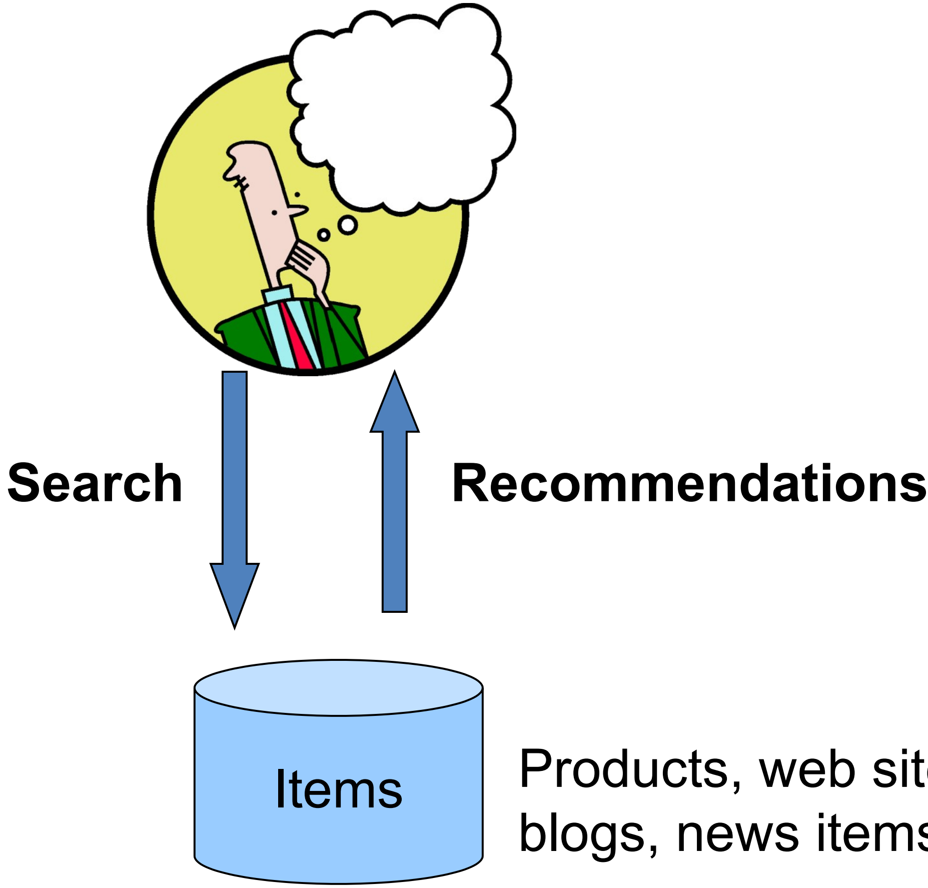
Collaborative Filtering

- Similar users are likely to have similar preferences
- *Collaborative filtering* exploits relationships between users to improve how items (documents) are matched to users (profiles)

Recommender Systems

- Recommender systems recommend items that a user may be interested in
- Examples
  - Amazon.com
  - NetFlix
- Recommender systems use collaborative filtering to recommend items to users

Recommendations



Products, web sites, blogs, news items, …

A class of problems which involves predicting users preferences / response.

Examples:

amazon.com

PANDORA

StumbleUpon

del.icio.us

NETFLIX

movielens
helping you find the *right* movies

last.fm
the social music revolution

Google News

YouTube

XBOX LIVE

Recommender Systems

- Systems for recommending items (e.g. books, movies, CD's, web pages, newsgroup messages) to users based on examples of their preferences.
- Many on-line stores provide recommendations (e.g. Amazon, Pandora, Netflix, Google).
- Recommenders have been shown to substantially increase sales at on-line stores.

The NetFlix Challenge

- NetFlix offered a prize of 1 million to the approach that can beat their own recommendation algorithm called CineMatch by at least 10%.
  - The prize was awarded in 2009.
- Actually, NetFlix's algorithm was not very good because a simple algorithm that predicts a user's rating based on KNN was comparable to CineMatch.
- An approach utilizing Alternating Least Squares provided 7% improvement.
- The wining entry was a combination of several algorithms.
  - Time of rating turned out to be a useful metric while other features such as genre did not have a significant contribution to the recommendation.

Methods of Recommendation Systems

- **Content-based systems** – examine the properties of the items liked by the user and recommend similar items.
  - Similarity of items is determined by measuring the similarity in their properties.

- **Collaborative filtering systems** – recommend items based on similarity between users and/or items.
  - i.e. the items recommended to a user, are those preferred by 'similar' users.

Utility Matrix

- In a recommendation-system, there are two entities that are captured, referred to as users and items.

- Users have preferences for certain items, and this is captured as a utility matrix.
  - Each entry is user-item pair, a value that represents what is known about the degree of preference of that user for that item.
  - We assume that the matrix is sparse, meaning that most entries are "unknown"… i.e. we don't know the user's perefence for that item.

Utility Matrix (Example)

items

users

	HP1	HP2	HP3	SW1	SW2	SW3	LOR1
A	4			5	1		
B	5	5	4		4	3	
C				2			2
D		3					

- The utility matrix represents the user's ratings of movies on a scale of 1-5.
- Blank entries mean that the user has not rated the movie, and as you can see the matrix is mostly spare.
- The goal of a recommender system is to predict the blanks in the utility matrix.
- Ex. Would user A like the movie LOR1?

- **Content-based Recommender System:**
  - We can take into account the properties of movies, such as their producer, director, stars, genre, year, etc.
  - So, we might find that SW1 is similar to SW2, hence then conclude that user B will like SW1 because its similar to SW2.

- **Collaborative-based Recommender System:**
  - Given a lot more data, we may notice that users who rated HP1 highly, also rated HP2 highly.
  - Hence, we can conclude that user A who rated HP1 high will also rate HP2 high.

items

users

	HP1	HP2	HP3	SW1	SW2	SW3	LOR1
A	4			5	1		
B	5	5	4		4	3	
C				2			2
D		3					

We don't have to predict all unknown entries, just discover some entries are likely to be high… so we can recommend that entry.

Content-based Recommendations

- In a content-based recommendation system we must generate user and item profiles.

- **Item Profile**
  - The 'profile' is a vector of the important properties of an item.
  - Examples :
    - For movies : set of actors, directors, year, genre.
    - For news-paper articles: topic, diction, word frequency
- **User Profile**
  - The 'profile' of the user is a vector describing the user's preferences based on the item properties.

Content-based Example

- How do we represent the item's profile?
  - Suppose the properties of a movie are the set of actors who stared in the movie and the average rating.
  - Then the movie is a vector of 0's and 1's depending if a set of actors stared in that movie or not.

- How do we represent the user profile?
  - If 20% of the movies that user U likes have Julia Roberts as one of the actors, then the user profile for U will have 0.2 in the component for Julia Roberts.

Content-based Example (Cont.)

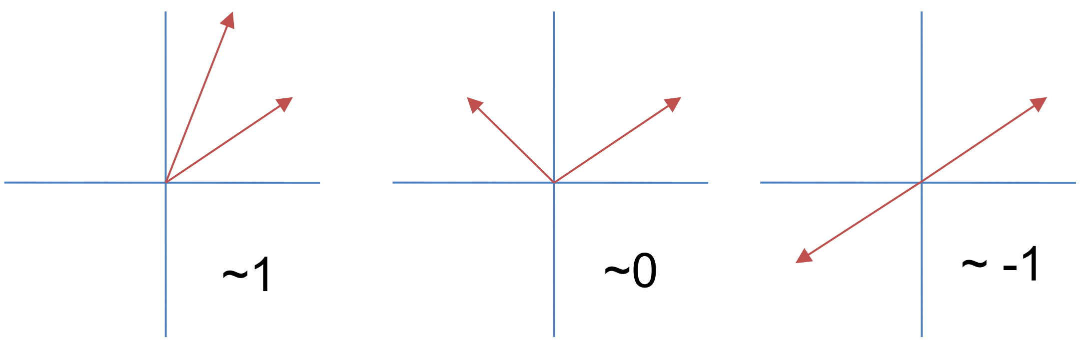
- Now suppose the utility matrix has a rating for each movie a user has seen, e.g., ratings 1–5, then we can weight the vectors representing the profiles of items by the utility value.
- Suppose user U, for all movies reviewed, gave a 3 on average.
- Suppose U has reviewed three movies with Julia Roberts, and gave ratings of 3, 4, and 5.
- Then, the user profile of U, for the feature Julia Roberts, will have the value of 1…. Essentially the average of (3-3), (4-3), and (5-3).
- Thus, the features will have a positive value for items with above-average ratings and negative values for items with below-average rating.

Content-based Example (Cont.)

- Given the profile vector for both the user and the movie
- We can estimate the degree to which a user would prefer an movie by computing the cosine distance between the user's and movie's vectors.
  - If a movie has many actors the user likes, and a few or no actors the they don't like, then a small cosine distance or a large positive fraction.
  - If a movie has many actors the user likes and a equal number of actors the user doesn't like, then cosine angle is 90 degrees.
  - If a movie has many actors the user doesn't like, the cosine will be a negative fraction and the angle between the vectors will be close to 180.

Cosine Distance

- Cosine similarity between two vectors is a measure that calculates the cosine of the angle between them.
- This metric is a measurement of orientation of the vectors.



$$\vec{a} \cdot \vec{b} = |\vec{a}| \cdot |\vec{b}| cos\theta$$

$$cos\theta = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| \cdot |\vec{b}|}$$

cosdist({a,b,c}, {x,y,z}) = $cos\theta = \frac{ax+by+cz}{\sqrt{|a^2|+|b^2|+|c^2|}+\sqrt{|x|+|y^2|+|z^2|}}$

We can treat blanks as 0, but this has the effect that lack of rating is considered more similar to disliking the movie than liking it.

Items Profile

Properties (actors that appeared in movies)

movies

	P1	P2	P3	P4	P5	P6	P7
M1	1	1			1	1	
M2		1	1	1			1
M3	1			1	1	1	
M4		1		1	1		

User Profile

	M1	M2	M3	M4
User	5	1	2	

Average movie rating is 2.66

CosineDist (User, M4) = -0.0687

	P1	P2	P3	P4	P5	P6	P7
User	1/3	0	-1.66	-1.66	0.83	0.33	0.38

Pros: Content-based Approach

- No need for data on other users
  - No **cold-start** or sparsity problems

- Able to recommend to users with unique tastes

- Able to recommend new & unpopular items
  - No first-rater problem

- 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? Must decide how to do this…
- **Overspecialization**
  - Will never recommends items outside user's content profile
  - People might have multiple interests
  - Unable to exploit quality judgments of other users

Collaborative-based Recommendations

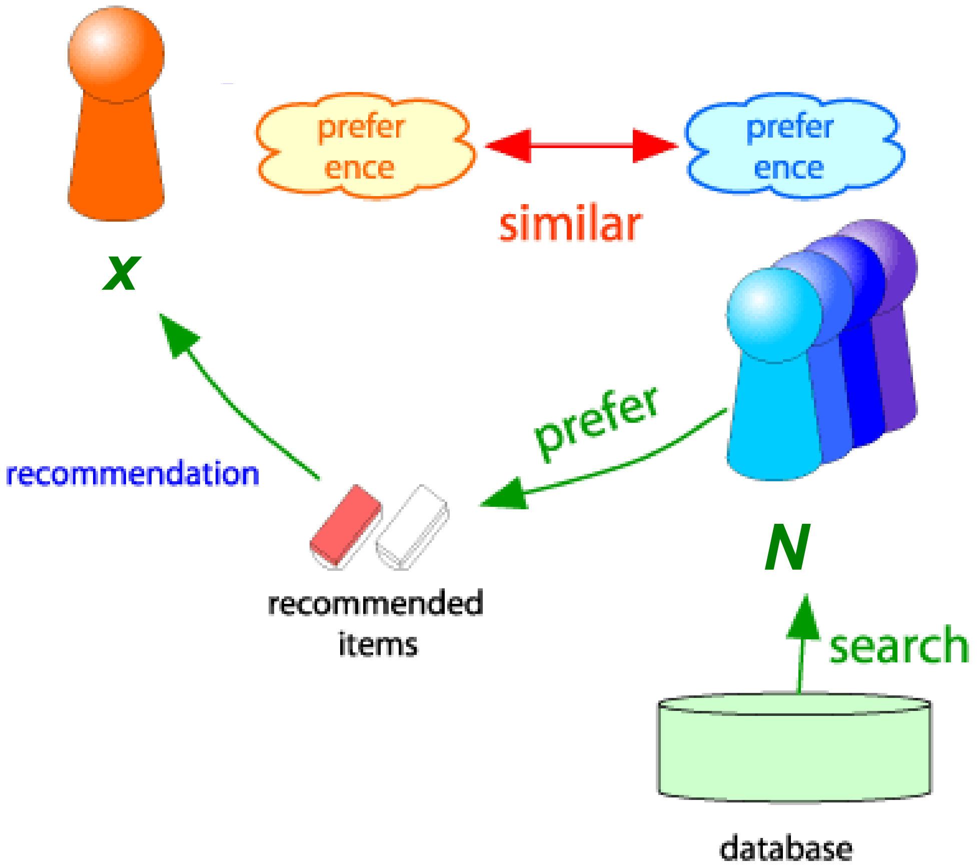
- In a collaborative-based recommendation system we must have a lot more data about users .. But we don't need any information about the item's properties.

- **User Profile**
  - The 'profile' of the user is a vector describing the user's preferences / ratings on items (movies in our case).

- Recommendation for a user U is made by looking at users that most similar to U and recommending items that these users like.

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***



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

User-User

movies

users

	M1	M2	M3	M4	M5	M6	M7
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Intuitively we want sim(A, B) > sim (A, C). Use cosine similarity, and subtract the row mean

For Row A, B, C, mean is 10/3 , 14/3, and 11/3 respectively. Hence, we get

	M1	M2	M3	M4	M5	M6	M7
A	2/3			5/3	-7/3		
B	1/3	1/3	-2/3				
C				-5/3	1/3	4/3	
D		0					0

CosineDist(A,B) =0.09245

CosineDist(A,C) = -0.559

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
  - Item-item filtering is prefered.

Item-Item (|N|=2)

users

movies

L	K	J	I	H	G	F	E	D	C	B	A	
	4		5			5			3		1	1
3	1	2			4			4	5			2
	5	3	4		3		2	1		4	2	3
	2			4			5		4	2		4
5	2					2	4	3	4			5
	4			2			3		3		1	6

- unknown rating

- rating between 1 to 5

Item-Item (|N|=2)

users

movies

L	K	J	I	H	G	F	E	D	C	B	A	
	4		5			5	?		3		1	1
3	1	2			4			4	5			2
	5	3	4		3		2	1		4	2	3
	2			4			5		4	2		4
5	2					2	4	3	4			5
	4			2			3		3		1	6

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

Item-Item (|N|=2)

users

movies

L	K	J	I	H	G	F	E	D	C	B	A		sim(1,m)
	4		5			5	?		3		1	1	1.00
3	1	2			4			4	5			2	-0.18
	5	3	4		3		2	1		4	2	**3**	**0.41**
	2			4			5		4	2		4	-0.10
5	2					2	4	3	4			5	-0.31
	4			2			3		3		1	**6**	**0.59**

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

Here we use Pearson correlation as similarity:

1) Subtract mean rating m_i from each movie i

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

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

2) Compute cosine similarities between rows

Item-Item (|N|=2)

users

movies

L	K	J	I	H	G	F	E	D	C	B	A		sim(1,m)
	4		5			5	?		3		1	1	1.00
3	1	2			4			4	5			2	-0.18
	5	3	4		3		2	1		4	2	3	0.41
	2			4			5		4	2		4	-0.10
5	2					2	4	3	4			5	-0.31
	4			2			3		3		1	6	0.59

Compute similarity weights:

$s_{1,3}=0.41$, $s_{1,6}=0.59$

Item-Item (|N|=2)

users

movies

L	K	J	I	H	G	F	E	D	C	B	A	
	4		5			5	**2.6**		3		1	1
3	1	2			4			4	5			2
	5	3	4		3		2	1		4	2	3
	2			4			5		4	2		4
5	2					2	4	3	4			5
	4			2			3		3		1	6

Predict by taking weighted average:

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

Recommendation System

- **Problem:**
  - Given the utility matrix, with *user* and *item* profiles, and given a new user U; predict whether this user likes a given item T.

- Approaches
  - **K-Nearest Neighbor**
  - Alternating Least Squares

Approach 1 – Use KNN (content-based)

- **Apply K-Nearest Neighbors over movies**
  - Kind **K** movies whose properties is most similar to the movie we are trying to predict.
  - These K movies must have been reviewed by user U.
  - Combine U's opinions about these K movies (either simple sum or weighted sum by closeness).

- **Example:**
  - **Given:** A utility matrix that expresses the user's movie ratings, and user U's profile (ratings for some movies).
  - **Goal:** Predict the user's U rating for unrated item T.
  - **Approach:**
    - Compare the properties of each movie to those of T's properties….choose top K closest movies.
    - Compute U's opinion about T based on their opinion over these K movies.

Approach 2 – Use Kmeans (content-based)

- **Apply Kmeans over items**
  - Divide items into clusters based on their fan-list
  - Average users U's votes over item's T cluster.

- **Example:**
  - **Given:** A utility matrix that expresses the user's movie ratings, and user U's profile (ratings for some movies).
  - **Goal:** Predict the user's U rating for unrated item T.
  - **Approach:**
    - Cluster movies based on their fan-list.
    - Find C, the cluster movie T most likely belong to.
    - Then predict U's rating for movie T based on U's rating for the other movies in the cluster C.

Collaborative Filtering

- Content-based filtering uses the features of items to determine their similarity, and hence then determine the rating of the ‘unseen’ item.

- Collaborative filtering however uses the user’s profile vector.
  - It compares the similarity between two users based on the users vectors.
  - Recommendation for a user U is then made by looking at the users that are most similar to U in this sense, and recommending items that these users like

Approach 1 – Use KNN

- **Apply K-Nearest Neighbors over users**
  - Find **K** users whose taste in items is closest to user **U**.
  - These **K** users need to have expressed interested in item **T** (either reviewed, viewed, bought, etc. the item)
  - Combine their vote about item **T** (this can be a simple sum or weighted sum by closeness).

- **Example:**
  - **Given:** A utility matrix that expresses the user's movie ratings, and user U's profile (ratings for some movies).
  - **Goal:** Predict the user's U rating for unrated item T.
  - **Approach:**
    - Compare each user expressed in utility matrix to user U and find the KNN based on the profile.
    - Then predict U's rating for unrated item T based on the average score of the KNN for movie T.

Approach 2 – Use Kmeans

- **Apply Kmeans over users**
  - Divide users into clusters, based on their item preferences.
  - Given user U, find its closest cluster C.
  - Take vote on unrated item T, based on cluster C… the cluster that U most likely matches.

- **Example:**
  - **Given:** A utility matrix that expresses the user's movie ratings, and user U's profile (ratings for some movies).
  - **Goal:** Predict the user's U rating for unrated item T.
  - **Approach:**
    - Cluster users based on their movie preferences.
    - Find C, the cluster user U most likely fits in.
    - Then predict U's rating for unwatched movie T based on the cluster's average vote.

Yet Another Approach

- Our goal is to estimate the blank (unknown) entries in the utility matrix, which represents the user / item profiles.

- Another approach to estimating the blank entries is to express the utility matrix as the *product* of two long thin matrices.
  - If there is actually a ‘smaller’ set of features of items and users that determine the behavior of most users to most items.

- Example:
  - Originally we expressed the utility matrix where each row is a user's profile and the columns are the movies the user has rated.
  - But do we really need to maintain all the movies as features?
  - Most users respond to a small number of features.
    - certain genres
    - certain actors or directors
    - certain plot lines

UV-Decomposition

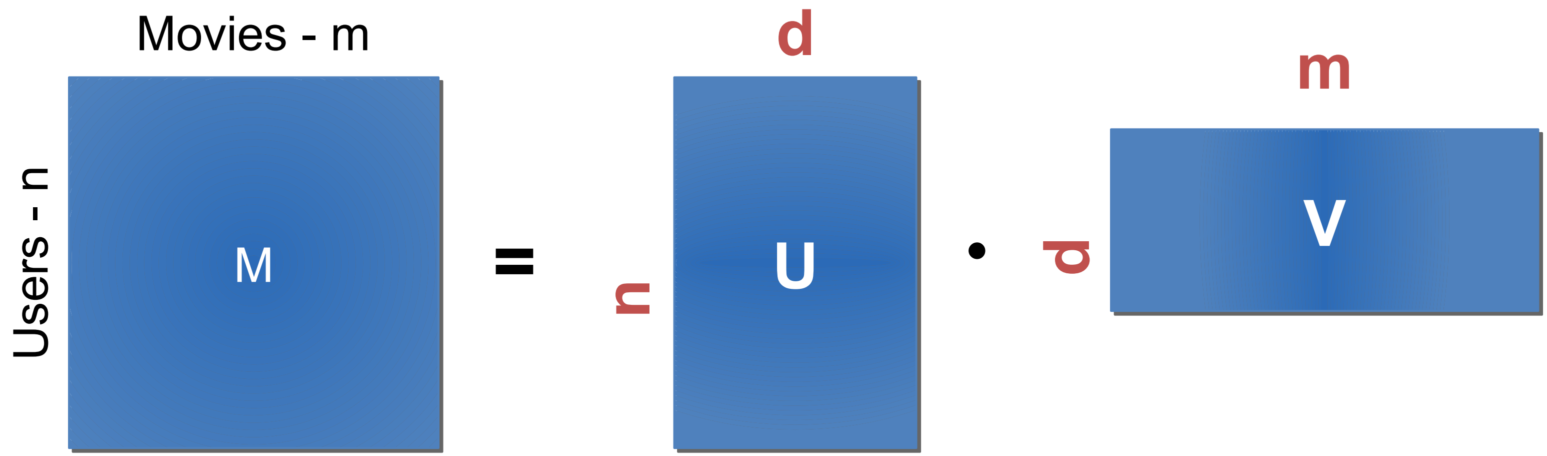
- Given utility matrix ***M***, with ***n*** rows and **m** columns.

- Find matrix ***U*** with ***n*** rows and **d** columns and a matrix V with ***d*** rows and ***m*** columns, such that ***U•V*** closely approximates ***M*** in those entries where ***M*** is non-blank.

- Hence, we are concluding that there are ***d*** dimensions that allow us to characterize ‘topics’.
  - Users usually like certain ‘topics’ / ‘classes’ of movies.
  - Movies usually fall under certain ‘topics’ / ‘classes’

- We can then use the entry in the product ***U•V*** to estimate the corresponding blank entry in utility matrix ***M***.

UV-Decomposition



Example: Given a 5-by-5 matrix, with two its entries are unknown.

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & ? & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & ? \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \\ u_{31} & u_{32} \\ u_{41} & u_{42} \\ u_{51} & u_{52} \end{bmatrix} \bullet \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} & v_{15} \\ v_{21} & v_{22} & v_{23} & v_{24} & v_{25} \end{bmatrix}$$

Root-Mean-Square Error

- There are many solutions to the UV dot product, thus, we choose a solution that minimizes the root-mean-square-error (RMSE).
- How to compute RMSE:
  - Sum over all non-blank entries in M then square of the difference between that entry and the corresponding entry in the product UV.
  - Take then mean (Average) of these square by dividing by the number of terms in the sum (the number of nonblank entries in M).
  - Take the square root of the mean.
- Minimizing the sum of the squares is the same as minimizing the square root of the average square.

- If we guess U and V are all 1's, then this yields a matrix consisting of all 2's.
- This is a bad guess obviously, because the RMSE is high.

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & ? & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & ? \end{bmatrix}$$

Ignore unknown in computation of RMSE

Row 1 : $(5-2)^2 + (2-2)^2 + (4-2)^2 + (4-2)^2 + (3-2)^2 = 18$
Row 2 : $(3-2)^2 + (1-2)^2 + (2-2)^2 + (4-2)^2 + (1-2)^2 = 7$
Row 3 : $(2-2)^2 + (3-2)^2 + (1-2)^2 + (4-2)^2 = 6$
Row 4 : $(2-2)^2 + (5-2)^2 + (4-2)^2 + (3-2)^2 + (5-2)^2 = 23$
Row 5 : $(4-2)^2 + (4-2)^2 + (5-2)^2 + (4-2)^2 = 21$

The sum of all 5 row is 75.
To compute the true RMSE, we divide by 23 (the number of nonblank entries in M) and take the square root. In this case sqrt(75/23)= 1.806 is the RMSE.

Computing UV-Decomposition

- Incremental UV-Decomposition - Start by arbitrarily chosen U and V , and repeatedly adjusting U and V to make the RMSE smaller .

- SGD (Stochastic Gradient descent)
  - For each example in the dataset you compute the error and then you update the parameters by a factor in the opposite direction of the gradient.

- Alternating Least Squares (ALS)
  - A different approach to optimizing the loss function. The key insight is that you can turn the non-convex optimization problem into an "easy" quadratic problem if you fix either U or V.
  - ALS fixes each one of those alternatively, hence, when one is fixed, the other one is computed, and vice versa.

  - There are two main benefits of this approach.
    - First, this is very easy to parallelize.
    - Second, whenever dealing with implicit datasets, which are usually not sparse, SGD is not practical (users times items can easily be in the order of billions). ALS is a much more efficient optimization technique in these cases.

UV Decomposition

- Select a U and V matrix
- While RMSE > threshold:
  - Select an element of U or V
  - Update element such that it minimizes RMSE

$$\begin{bmatrix} x & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} x+1 & x+1 & x+1 & x+1 & x+1 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

$(5-(x+1))^2 + (2- (x+1))^2 + (4\ (x+1))^2 + (4\ (x+1))^2 + (3-(x+1))^2 \rightarrow$

We want x that minimizes the sum, so we take the derivative and set it equal to 0

$-2((4-x)+(1-x)+(3-x)+(3-x)+(2-x)) = 0 \rightarrow$ Solve for x, you get $x = 2.6$