COLLABORATIVE FILTERING

Rule base

Recommendation

Rule based recommendation

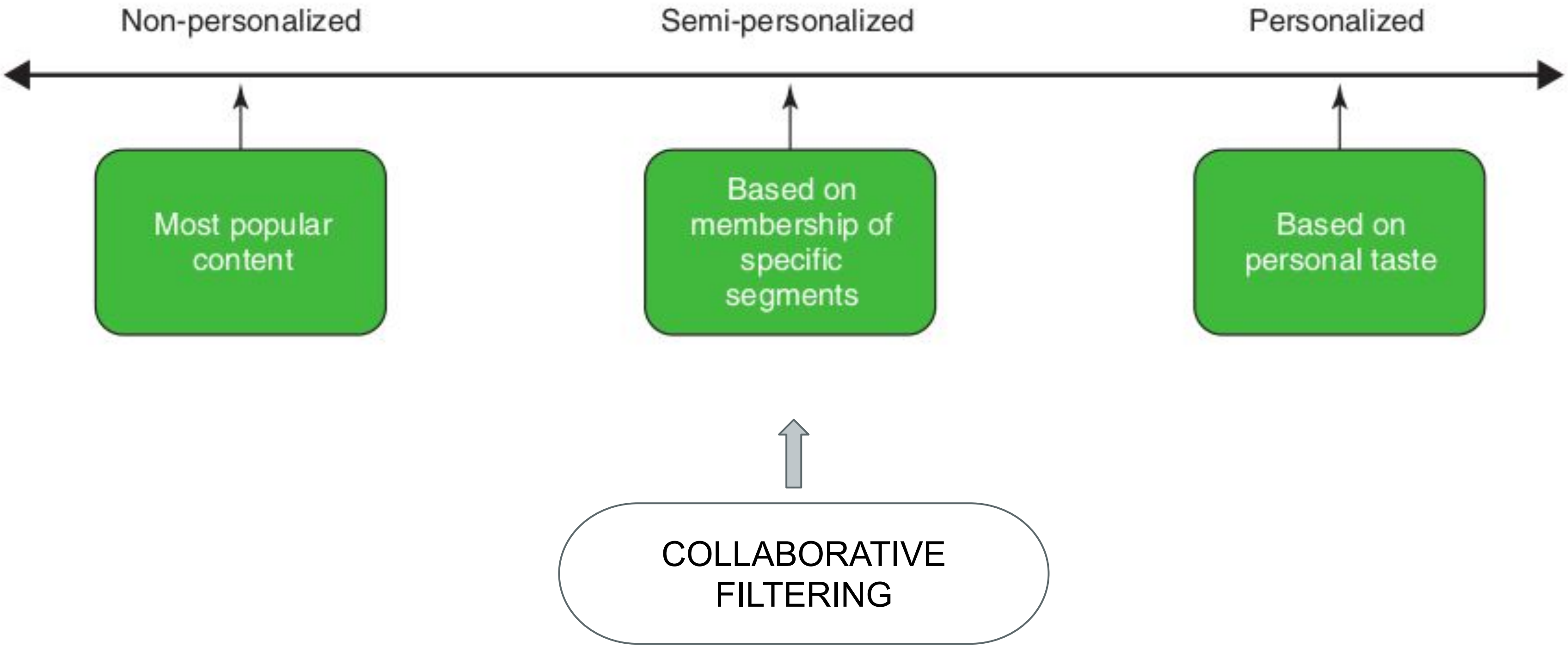
- Recommend base on fix rule apply on: popularity, categories,...
- Can be presented on:
    - Condition: if - then - else -then.
    - Formula:

ex:

$$\frac{f(popularity)}{g(age)}$$

- **Disadvantages**:
    + Lack of diversity.
    + No personalization.

Personalization level



COLLABORATIVE FILTERING

Collaborative filtering ideals

- **Helping each other:** The assumption on which collaborative filtering is based is that together we can be better, and together we'll better understand each other.

Find the others with similar to you

Ask them:
what ..... you like most?

User-based CF

Ask them:
I liked ..... what ..... that I may interest?

Item-based CF

You are content provider that have data about the interaction between users and items!

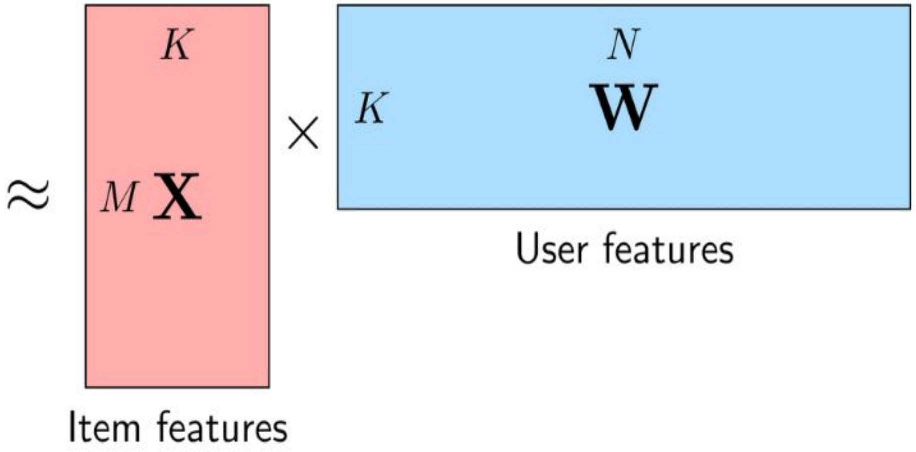
How to make it on computer?

Rating matrix

Building Rating matrix.

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	5	5	2	0	1	1.68	2.70
i_1	4	3.23	2.33	0	1.67	2	3.38
i_2	4.15	4	1	-0.5	0.71	1	1
i_3	2	2	3	4	4	2.10	4
i_4	2	0	4	2.9	4.06	3.10	5

Neighborhood method



Matrix factorization

Do not have rating data?

- Almost user never or rarely rating item.

- Rating is just unit that reflect the interest of user in item.

=> Implicit rating:

+ User behavior: Time to watch an item, number of click or purchased on an item,....

+ Implicit data is easy to collect and have diverse of type.

How to make money with implicit data?

- Use formula to derive implicit to explicit.

Ex:

$$AP(u,i) = \ln\left(\frac{\text{The number of transactions of user } u \text{ including item } i}{\text{The number of transactions of user } u} + 1\right) \quad (1)$$

$$\downarrow$$

$$RP(u,i) = \frac{AP(u,i)}{\underset{c \in U}{\mathrm{Max}}(AP(c,i))}$$

$$\downarrow$$

$$\text{Implicit rating}(u,i) = \text{Round up } (5 \times RP(u,i))$$

Use matrix factorization

- Matrix factorization can be applied on implicit data.

- Finding latent factor about relationship of user and item.

- Rating matrix will be decomposed to item-features vector (matrix) and user weighted vector (matrix)

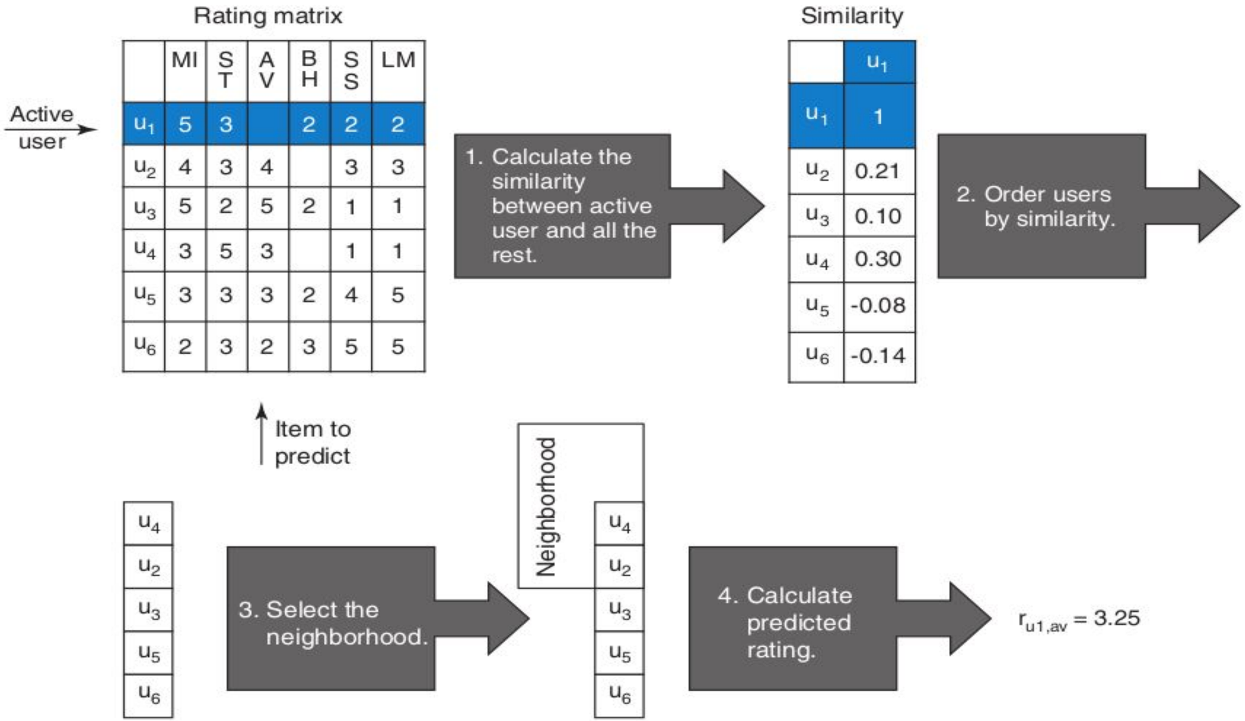
NEIGHBORHOOD
Collaborative Filtering

USER-USER

NEIGHBORHOOD

USER-USER NEIGHBORHOOD

- Use user similarity to fill missing values of rating matrix.
- The missing values can be calculated by mean of the others one.



USER-USER NEIGHBORHOOD

Compute similarity

- Distance-base similarity:

$$\frac{1}{1+\sqrt{\sum_{i=1}^{m}(R_{a,i}-R_{b,i})^2}}$$

- Cosin similarity:

$$\frac{\sum_{i=1}^{m}(R_{a,i})(R_{b,i})}{\sqrt{\sum_{i=1}^{m}(R_{a,i})^2}\sqrt{\sum_{i=1}^{m}(R_{b,i})^2}}$$

- Pearson similarity:

$$\frac{\sum_{i=1}^{m}(R_{ai}-\overline{R}_a)(R_{b,i}-\overline{R}_b)}{\sqrt{\sum_{i=1}^{m}(R_{a,i}-\overline{R}_a)^2}\sqrt{\sum_{i=1}^{m}(R_{b,i}-\overline{R}_b)^2}}$$

USER-USER NEIGHBORHOOD

- User vector will be very large and very sparse.
=> Use sparsed matrix and normalized it.

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	5	5	2	0	1	?	?
i_1	4	?	?	0	?	2	?
i_2	?	4	1	?	?	1	1
i_3	2	2	3	4	4	?	4
i_4	2	0	4	?	?	?	5
	↓	↓	↓	↓	↓	↓	↓
$\bar{u}_j$	3.25	2.75	2.5	1.33	2.5	1.5	3.33

a) Original utility matrix $\mathbf{Y}$ and mean user ratings.

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	1.75	2.25	-0.5	-1.33	-1.5	0	0
i_1	0.75	0	0	-1.33	0	0.5	0
i_2	0	1.25	-1.5	0	0	-0.5	-2.33
i_3	-1.25	-0.75	0.5	2.67	1.5	0	0.67
i_4	-1.25	-2.75	1.5	0	0	0	1.67

b) Normalized utility matrix $\bar{\mathbf{Y}}$.

USER-USER NEIGHBORHOOD

- What is sparsed matrix?
+ Coordinate sparsed matrix form: (COO)

$$\begin{bmatrix} 0 & 0 & 3 & 0 & 4 \\ 0 & 0 & 5 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 6 & 0 & 0 \end{bmatrix} \Rightarrow$$

Row	0	0	1	1	3	3
Column	2	4	2	3	1	2
Value	3	4	5	7	2	6

+ Compressed Sparse Column (CSC)
+ Compressed Sparse Row (CSR)

USER-USER NEIGHBORHOOD

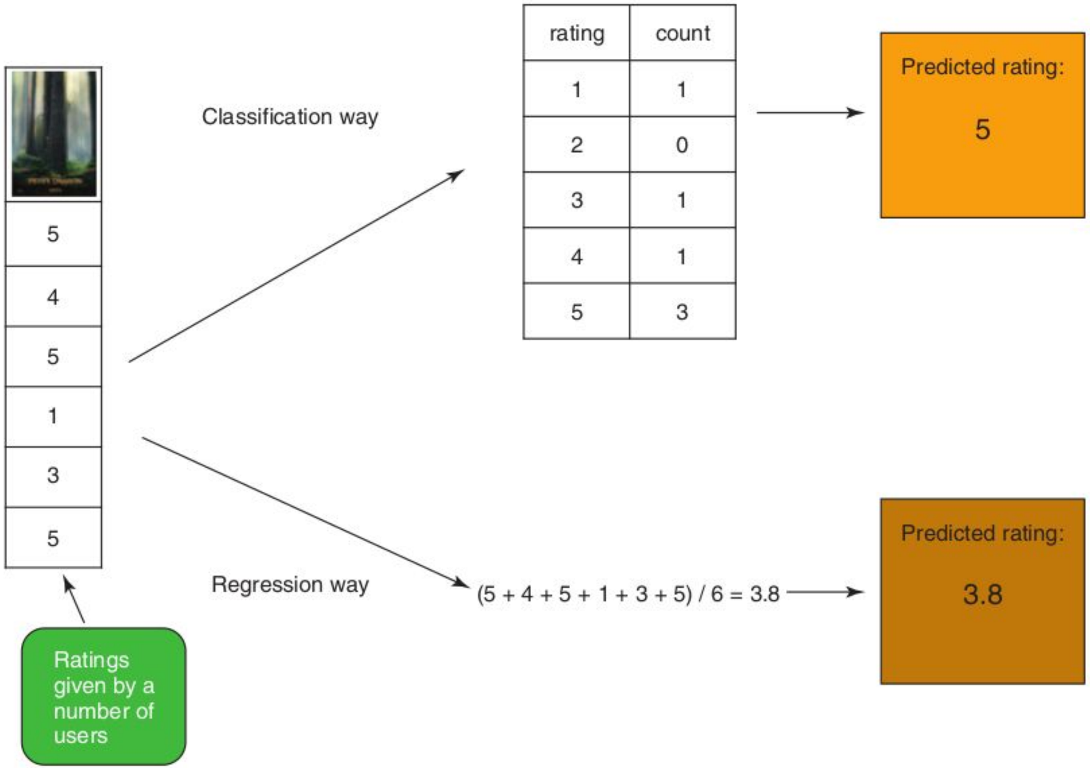
+ Each sparsed matrix form have their own the advantages and disadvantages.

+ Python has a library for present it: **scipy.sparse**

+ We can change their form easy and do some basic operation on it.

USER-USER NEIGHBORHOOD

- After compute similariy between the users. We choose **K neighboor** user to compute the missing values of the active user vector.



ITEM- ITEM

NEIGHBORHOOD

ITEM-ITEM NEIGHBORHOOD

In fact, the number of users is mostly greater than the number of items and everyone not always rates for items. So, Utility Y matrix contains many sparse cell.

That calculating on similarity item-item before recommending items to users who had the same connections with previous items reduces sparse cells.

The approach is called item-item collaborative filtering and used commonly.

ITEM-ITEM NEIGHBORHOOD

The process of item-items approach is similar to that of user-user, but calculating the mean user ratings, this way will find the mean item ratings.

	u_0	u_1	u_2	u_3	u_4	u_5	u_6		
i_0	5	5	2	0	1	?	?	$\rightarrow$	2.6
i_1	4	?	?	0	?	2	?	$\rightarrow$	2
i_2	?	4	1	?	?	1	1	$\rightarrow$	1.75
i_3	2	2	3	4	4	?	4	$\rightarrow$	3.17
i_4	2	0	4	?	?	?	5	$\rightarrow$	2.75

a) Original utility matrix $\mathbf{Y}$ and mean item ratings.

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	2.4	2.4	-.6	-2.6	-1.6	0	0
i_1	2	0	0	-2	0	0	0
i_2	0	2.25	-0.75	0	0	-0.75	-0.75
i_3	-1.17	-1.17	-0.17	0.83	0.83	0	0.83
i_4	-0.75	-2.75	1.25	0	0	0	2.25

b) Normalized utility matrix $\bar{\mathbf{Y}}$.

ITEM-ITEM NEIGHBORHOOD

It is interesting in Item similarity matrix (table c.) is 2 ranges covered blue and red squares including positive values only. So, there will be 2 groups of items (i0, i1, i2 and i3, i4) and this result is helpful to recommend kind of items to users.

	i_0	i_1	i_2	i_3	i_4
i_0	1	0.77	0.49	-0.89	-0.52
i_1	0.77	1	0	-0.64	-0.14
i_2	0.49	0	1	-0.55	-0.88
i_3	-0.89	-0.64	-0.55	1	0.68
i_4	-0.52	-0.14	-0.88	0.68	1

c) Item similarity matrix $\mathbf{S}$.

ITEM-ITEM NEIGHBORHOOD

The below formula is used to predict the each rating of item from user.

$$\widehat{y_{u,i}} = \frac{\sum_{i_j \in N(i,u)} \bar{y}_{u,i_j} sim(i, i_j)}{\sum_{i_j \in N(i,u)} |sim(i, i_j)|}$$

N(i,u) imply k items (neighborhood) have the greatest similarity in ratings from users

Given k =2, i0 and i2 have the greatest similarity to i1 in ratings (=0.77 ; 0)

To predict the rating i1 from u1, we find that u1 rated i0 and i2 at 2.4 and 2.24 respectively.

ITEM-ITEM NEIGHBORHOOD

Example:

$$\widehat{y_{u_1,i_1}} = \frac{0.77 * 2.4 + 0 * 2.25}{0.77 + 0} = 2.4$$

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	2.4	2.4	-.6	-2.6	-1.6	0	0
i_1	2	0	0	-2	0	0	0
i_2	0	2.25	-0.75	0	0	-0.75	-0.75
i_3	-1.17	-1.17	-0.17	0.83	0.83	0	0.83
i_4	-0.75	-2.75	1.25	0	0	0	2.25

b) Normalized utility matrix $\bar{\mathbf{Y}}$.

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	2.4	2.4	-.6	-2.6	-1.6	-0.29	-1.52
i_1	2	2.4	-0.6	-2	-1.25	0	-2.25
i_2	2.4	2.25	-0.75	-2.6	-1.20	-0.75	-0.75
i_3	-1.17	-1.17	-0.17	0.83	0.83	0.34	0.83
i_4	-0.75	-2.75	1.25	1.03	1.16	0.65	2.25

d) Normalized utility matrix $\bar{\mathbf{Y}}$.

USER-USER or ITEM-ITEM NEIGHBORHOOD

The 2 approach results provide similar recommendations, but 2 cells in column u5 and u6. According to item similarity matrix, there 2 group, if someone likes i0, she or he will have tendency to choose i1 and i2 not i3 or i4.

So, the result of item-item way is more reasonable.

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	1.75	2.25	-0.5	-1.33	-1.5	0.18	-0.63
i_1	0.75	0.48	-0.17	-1.33	-1.33	0.5	0.05
i_2	0.91	1.25	-1.5	-1.84	-1.78	-0.5	-2.33
i_3	-1.25	-0.75	0.5	2.67	1.5	0.59	0.67
i_4	-1.25	-2.75	1.5	1.57	1.56	1.59	1.67

User-User

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	2.4	2.4	-.6	-2.6	-1.6	-0.29	-1.52
i_1	2	2.4	-0.6	-2	-1.25	0	-2.25
i_2	2.4	2.25	-0.75	-2.6	-1.20	-0.75	-0.75
i_3	-1.17	-1.17	-0.17	0.83	0.83	0.34	0.83
i_4	-0.75	-2.75	1.25	1.03	1.16	0.65	2.25

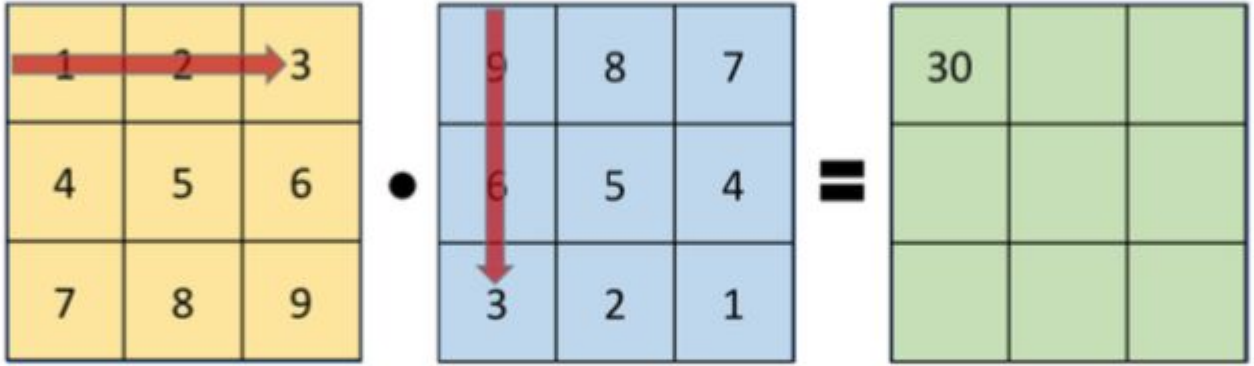
Item-Item

	i_0	i_1	i_2	i_3	i_4
i_0	1	0.77	0.49	-0.89	-0.52
i_1	0.77	1	0	-0.64	-0.14
i_2	0.49	0	1	-0.55	-0.88
i_3	-0.89	-0.64	-0.55	1	0.68
i_4	-0.52	-0.14	-0.88	0.68	1

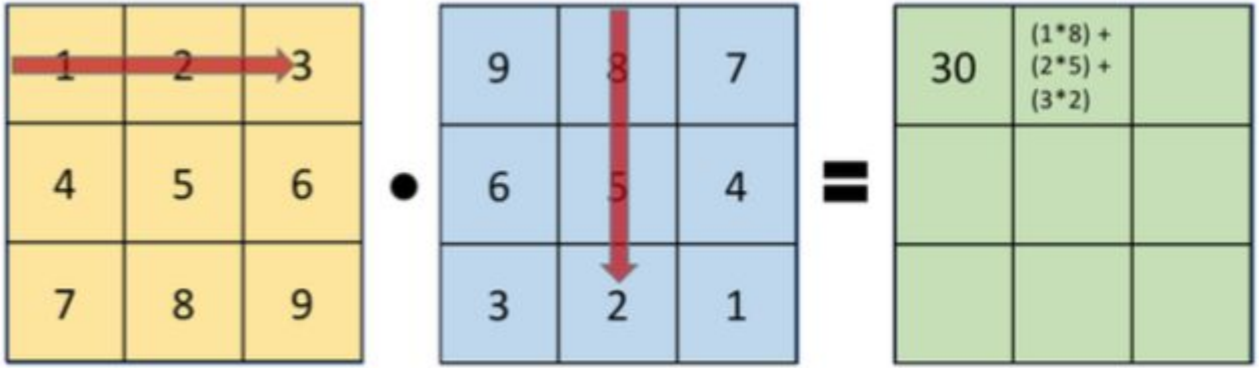
Item similarity matrix

MATRIX FACTORIZATION

Matrix Multiplication



Matrix Multiplication

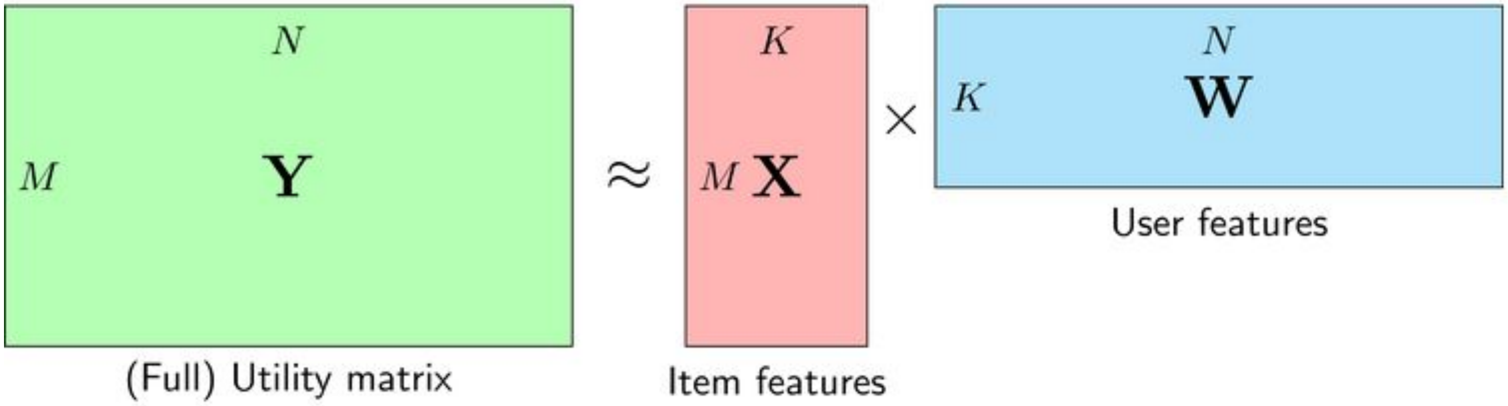


Matrix Multiplication

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \bullet \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 30 & 24 & 18 \\ 84 & 69 & 54 \\ 138 & 114 & 90 \end{bmatrix}$$

MATRIX FACTORIZATION

$$\mathbf{Y} \approx \hat{\mathbf{Y}} = \mathbf{X}\mathbf{W}$$



- M: number of items
- N: number of users
- K: number of feature

Main idea

➔ Each user can be described by *k attributes* or *features*. For example, feature 1 might be a number that says how much each user likes sci-fi movies.

➔ Each item can be described by an analogous set of *k* attributes or features. To correspond to the above example, feature 1 for the movie might be a number that says how close the movie is to pure sci-fi.

➔ If we multiply each feature of the user by the corresponding feature of the movie and add everything together, this will be a good approximation for the rating the user would give that movie.

Method

- ➔ We do not know what these features are. Nor do we know how many *(k)* features are relevant. So we just pick a number for k and learn the relevant values between features and all the users and items.
- ➔ So how do we learn these number? By minimizing the loss function of course!
- ➔ Let's say we have x_m as the vector for m-row in X and w_n as the vector for n-column in W. Then we will have $z_{mn=}x_m w_n$ as the user n's predict rating for item m

Matrix Factorization

$$\begin{bmatrix} 5 & 1 & 4 & 3 & 3 \\ 2 & 2 & 4 & 3 & 2 \\ 1 & 4 & 2 & 4 & 5 \\ 2 & 2 & 3 & 4 & 2 \\ 3 & 4 & 4 & 5 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 2/5 & 1 & 0 & 0 & 0 \\ 1/5 & 19/8 & 1 & 0 & 0 \\ 2/5 & 1 & 2/9 & 1 & 0 \\ 3/5 & 17/8 & 7/9 & 2/43 & 1 \end{bmatrix} \bullet \begin{bmatrix} 5 & 1 & 4 & 3 & 3 \\ 0 & 8/5 & 12/5 & 9/5 & 4/5 \\ 0 & 0 & -9/2 & -7/8 & 5/2 \\ 0 & 0 & 0 & 43/36 & -5/9 \\ 0 & 0 & 0 & 0 & -18/43 \end{bmatrix}$$

Matrix Factorization

$$
\begin{bmatrix} 5 & 1 & 4 & 3 & 3 \\ 2 & 2 & 4 & 3 & 2 \\ 1 & 4 & 2 & 4 & 5 \\ 2 & 2 & 3 & 4 & 2 \\ 3 & 4 & 4 & 5 & 5 \end{bmatrix}
=
\begin{bmatrix} .576 & 2.55 & 0 \\ .639 & .670 & 1.94 \\ 2.25 & .028 & 0 \\ .825 & .494 & 1.80 \\ 1.99 & .982 & .838 \end{bmatrix}
\bullet
\begin{bmatrix} .437 & 1.79 & .850 & 1.77 & 2.19 \\ 1.86 & 0 & 1.38 & .758 & .678 \\ .322 & .382 & 1.12 & .927 & 0 \end{bmatrix}
$$

Matrix Factorization

Number of Latent Features (Rank or k) = 3

$$\begin{bmatrix}5&1&4&3&3\\2&2&4&3&2\\1&4&2&4&5\\2&2&3&4&2\\3&4&4&5&5\end{bmatrix} = \begin{bmatrix}.576&2.55&0\\.639&.670&1.94\\2.25&.028&0\\.825&.494&1.80\\1.99&.982&.838\end{bmatrix} \bullet \begin{bmatrix}.437&1.79&.850&1.77&2.19\\1.86&0&1.38&.758&.678\\.322&.382&1.12&.927&0\end{bmatrix}$$

Matrix Factorization for Sparse Matrix

$$
\begin{bmatrix} 84 & - & 48 & - \\ - & 50 & - & - \\ - & - & 51 & - \\ 91 & - & - & 107 \end{bmatrix}
=
\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 4 & 3 & 2 & 1 \\ 1 & 5 & 2 & 4 & 3 \\ 5 & 1 & 4 & 2 & 3 \end{bmatrix}
\cdot
\begin{bmatrix} 6 & 2 & 3 & 7 \\ 7 & 8 & 4 & 8 \\ 8 & 0 & 5 & 7 \\ 5 & 3 & 3 & 9 \\ 4 & 2 & 2 & 6 \end{bmatrix}
$$

Matrix Factorization for Sparse Matrix

$$\begin{bmatrix} 84 & - & 48 & - \\ - & 50 & - & - \\ - & - & 51 & - \\ 91 & - & - & 107 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 4 & 3 & 2 & 1 \\ 1 & 5 & 2 & 4 & 3 \\ 5 & 1 & 4 & 2 & 3 \end{bmatrix} \bullet \begin{bmatrix} 6 & 2 & 3 & 7 \\ 7 & 8 & 4 & 8 \\ 8 & 0 & 5 & 7 \\ 5 & 3 & 3 & 9 \\ 4 & 2 & 2 & 6 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 84 & 40 & 48 & 110 \\ 96 & 50 & 54 & 112 \\ 89 & 60 & 51 & 115 \\ 91 & 30 & 51 & 107 \end{bmatrix}$$

Loss function

$$\mathcal{L}(\mathbf{X}, \mathbf{W}) = \frac{1}{2s} \sum_{n=1}^{N} \sum_{m:r_{mn}=1} (y_{mn} - \mathbf{x}_m \mathbf{w}_n)^2 + \frac{\lambda}{2}(||\mathbf{X}||_F^2 + ||\mathbf{W}||_F^2)$$

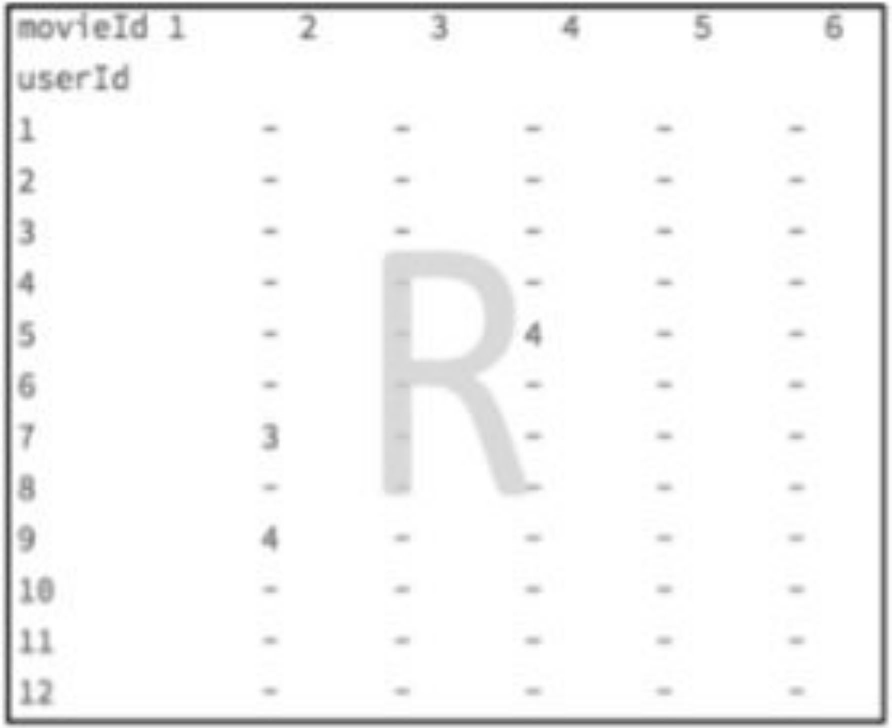
- Note:
    - The L2 regularization terms is to prevent overfitting.
    - s is the total number of ratings
    - r_{mn}=1 when user n has rated for item m

Minimizing loss function

We have 2 different approach:

- ➔ Stochastic Gradient Descent
- ➔ Alternating least squares

Alternating Least Squares



movieId	1	2	3	4	5	6
userId						
1		-	-	-	-	-
2		-	-	-	-	-
3		-	-	-	-	-
4		-	-	-	-	-
5		-	-	4	-	-
6		-	-	-	-	-
7		3	-	-	-	-
8		-	-	-	-	-
9		4	-	-	-	-
10		-	-	-	-	-
11		-	-	-	-	-
12		-	-	-	-	-

Alternating Least Squares

R

movieId	1	2	3	4	5	6
userId						
1		-	-	-	-	-
2		-	-	-	-	-
3		-	-	-	-	-
4		-	-	-	-	-
5		-	-	4	-	-
6		-	-	-	-	-
7		3	-	-	-	-
8		-	-	-	-	-
9		4	-	-	-	-
10		-	-	-	-	-
11		-	-	-	-	-
12		-	-	-	-	-

=

U

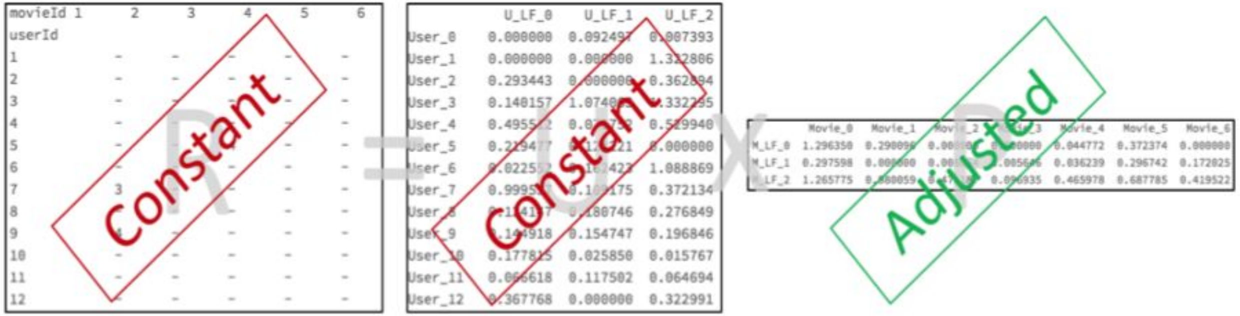
	U_LF_0	U_LF_1	U_LF_2
User_0	0.000000	0.092497	0.007393
User_1	0.000000	0.000000	1.322806
User_2	0.293443	0.000000	0.362894
User_3	0.140157	1.074063	1.332295
User_4	0.495512	0.075752	0.529940
User_5	0.219477	0.124221	0.000000
User_6	0.022552	0.162423	1.088869
User_7	0.999537	0.1[illegible]9175	0.372134
User_8	0.124147	0.180746	0.276849
User_9	0.144918	0.154747	0.196846
User_10	0.177815	0.025850	0.015767
User_11	0.066618	0.117502	0.064694
User_12	0.367768	0.000000	0.322991

x

P

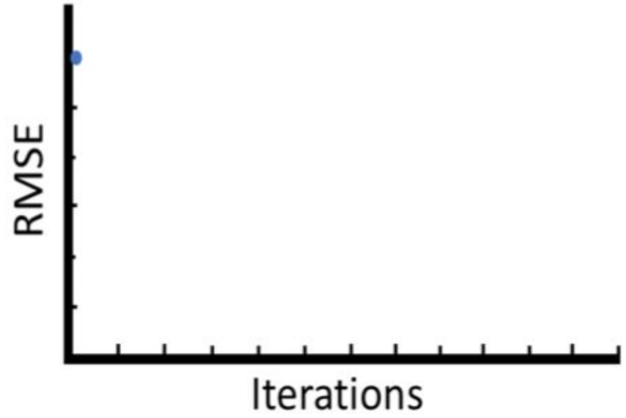
	Movie_0	Movie_1	Movie_2	Movie_3	Movie_4	Movie_5	Movie_6
M_LF_0	1.296350	0.290096	[illegible]	[illegible]	0.044772	0.372374	0.000000
M_LF_1	0.297598	0.000000	[illegible]	[illegible]	0.036239	0.296742	0.172025
M_LF_2	1.265775	0.980059	[illegible]	0.096935	0.465978	0.687785	0.419522

Alternating Least Squares

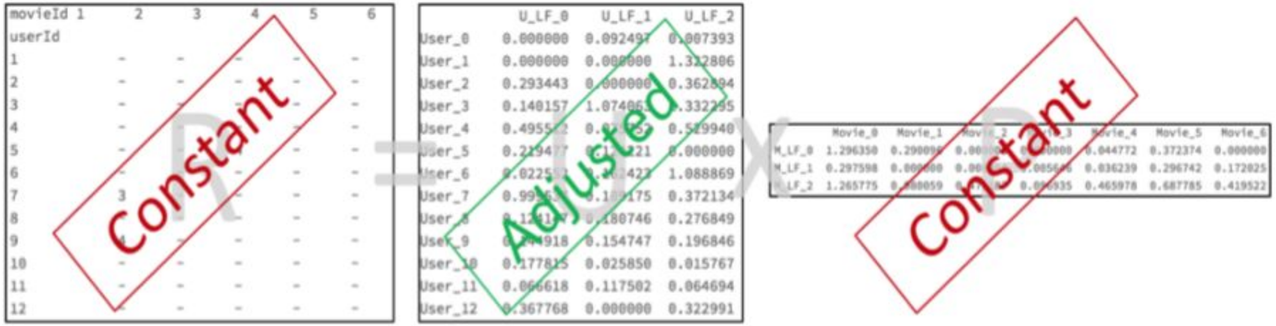


Iteration: 1

RMSE = 12,000

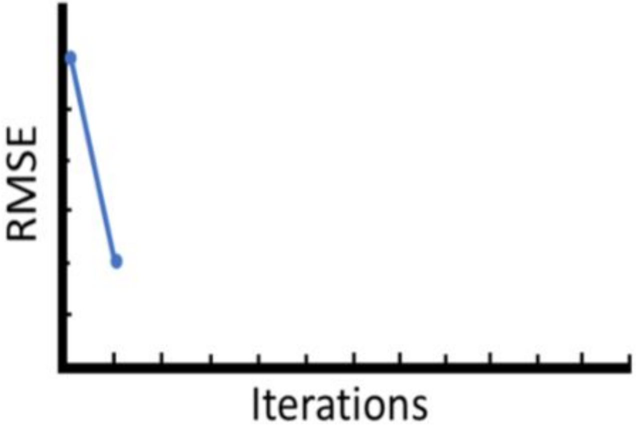


Alternating Least Squares



Iteration: 2

RMSE = 4,000



Alternating Least Squares

movieId	1	2	3	4	5	6
userId						
1		-	-	-	-	-
2		-	-	-	-	-
3		-	-	-	-	-
4		-	-	-	-	-
5		-	-	-	-	-
6		-	-	-	-	-
7		3	-	-	-	-
8		-	-	-	-	-
9		4	-	-	-	-
10		-	-	-	-	-
11		-	-	-	-	-
12		-	-	-	-	-

Constant

	U_LF_0	U_LF_1	U_LF_2
User_0	0.000000	0.092497	0.007393
User_1	0.000000	0.000000	1.322806
User_2	0.293443	0.000000	0.362894
User_3	0.140157	[illegible]	0.332295
User_4	[illegible]	[illegible]	0.529940
User_5	0.219477	[illegible]	0.000000
User_6	[illegible]	[illegible]	1.088869
User_7	[illegible]	[illegible]	0.372134
User_8	[illegible]	0.180746	0.276849
User_9	0.144918	0.154747	0.196846
User_10	0.177815	0.025858	0.015767
User_11	0.066618	0.117502	0.064694
User_12	0.367768	0.000000	0.322991

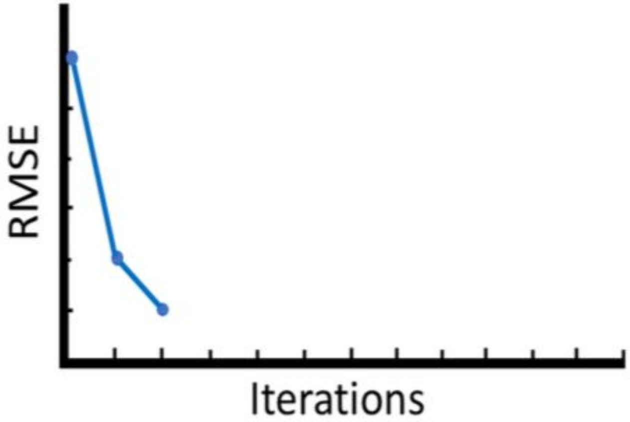
Constant

	Movie_0	Movie_1	Movie_2	Movie_3	Movie_4	Movie_5	Movie_6
M_LF_0	1.296350	0.290096	[illegible]	[illegible]	0.044772	0.372374	0.000000
M_LF_1	0.297598	0.000000	[illegible]	[illegible]	0.036239	0.296742	0.172025
M_LF_2	1.265775	[illegible]	[illegible]	0.096935	0.465978	0.687785	0.419522

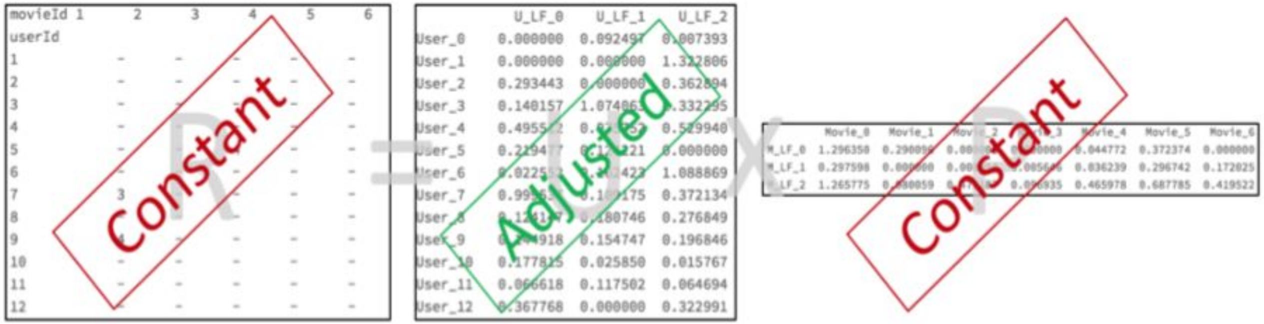
Adjusted

Iteration: 3

RMSE = 2,000

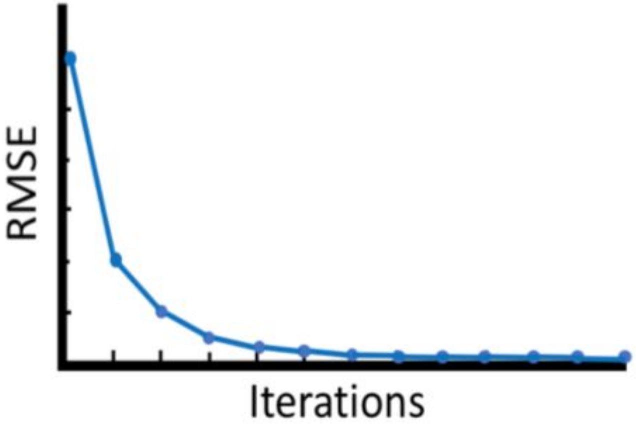


Alternating Least Squares



Iteration: n

RMSE = 0.6



Alternating Least Squares

R

	movie1	movie2	movie3	movie4	movie5
user1	-	-	3	-	5
user2	3	-	-	-	1
user3	-	-	-	3	2
user4	-	-	1	-	-
user5	-	2	-	-	-

->

U

	u_lf_1	u_lf_2	u_lf_3	u_lf_4
user1	1.0	0.0	2.0	1.0
user2	1.0	3.0	0.0	0.0
user3	0.0	3.0	0.0	0.0
user4	1.0	1.0	3.0	0.0
user5	2.0	1.0	0.0	0.0

X

P

	movie1	movie2	movie3	movie4	movie5
m_lf_1	1.0	0.0	1.0	2.0	1.0
m_lf_2	1.0	1.0	1.0	0.0	0.0
m_lf_3	0.0	0.0	1.0	0.0	1.0
m_lf_4	0.0	2.0	0.0	2.0	3.0

=

$\hat{R}$

	movie1	movie2	movie3	movie4	movie5
user1	1.0	2.0	3.0	4.0	6.0
user2	4.0	3.0	4.0	2.0	1.0
user3	3.0	3.0	3.0	1.0	1.0
user4	2.0	1.0	5.0	1.0	4.0
user5	3.0	1.0	3.0	4.0	2.0

DEMO