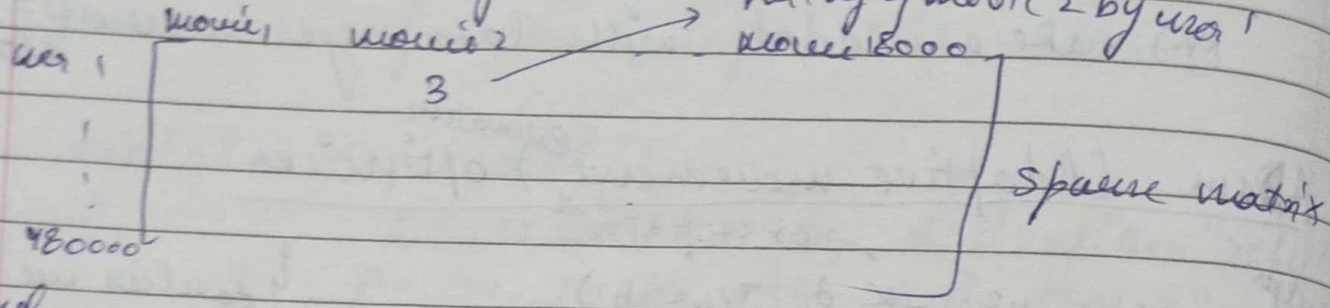
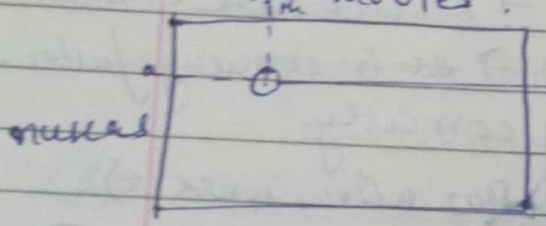
Recommender system



rating of movie 2 by user 1

Generally we need to fill this matrix if user1 watch movie one then we need to fill other ratings. what rating would user would have given

netflix challenge :- give ways to fill matrix, they also have filled matrix → they will check performance by root mean square error.



a^{th} user i^{th} movie ⇒ y_{ai} true label. { y_{ai} is actual rating / rating withheld.

→ This matrix is a sparse matrix.

On an average each one of the movies is only ranked by 5000 users.

Goal :- To fill out the whole matrix i.e to predict what will the opinion of the user for the movie he/she has not seen yet.

Q if similarity b/w two user is measured by ranking given by user

① Ranking given by user to other movie

⇒ Let m be the no. of movies and n are the no. of users.

The ranking y_{ai} of a movie $i = \{1, 2, \ldots m\}$ by a user $a = \{1, \ldots n\}$ may already exist or not. Our goal is to predict y_{ai} in the case y_{ai} doesnot exist.

Let KNN (a) be the set of k users similar to user 'a'. Let sim(a, b) be a similarity measure b/w user a & b belong to KNN(a). The K-Nearest Neighbour method predicts a ranking Y_{ai} to be

~~$\hat{Y}_{ai} = \Sigma \times Y_{bi}$~~

$$\hat{Y}_{ai} = \frac{\sum_{b \in KNN(a)} sim(a,b) Y_{bi}}{\sum_{b \in KNN(a)} sim(a,b)}$$

KNN → make cluster → to find rating of a person in cluster → see all points in cluster to minimise complexity.

b ने i को दिया rating दी है → Y_{bi}

b) in neighbour of a (in cluster of a given by KNN)

sim can be euclidean $||x_a - x_b||$

cosine $\cos\theta = \frac{x_a \cdot x_b}{||x_a|| \, ||x_b||}$

~~Jacard index~~

Drawback of KNN

Success of KNN depends heavily upon the similarity measure

Collaborative filtering Naive Approach

Each user has a set of movies that he or she has already rated. Y be a matrix with n rows m columns whose $(a,i)^{th}$ entry Y_{ai} is the rating by user a to movie i if this is already been given, otherwise blank

Our goal is to build matrix $x = \hat{Y}$ having no blank entries and whose $(a,i)^{th}$ entry is the prediction of the rating of user a to i^{th} movie

Let D be the set of all (a, i)'s for which a user rating Y_{ai} exist, i.e. $(a, i) \in D$ iff rating of user 'a' to movie 'i' exists.

Naive Approach will try to minimize the following objective cost function

Date / /

$$J(X) = \sum_{(a,i)\in D} \frac{(Y_{ai} - X_{ai})^2}{2} + \frac{\lambda}{2} \sum_{(a,i)} X_{ai}^2$$

↳ need to fill the missing entries in X

↓

(a,i) for which rating is given

where the first term is the sum of squared errors for entries with obs. rating

second term is regularization term which penalizes those predictions which become extremely large

λ is regularization parameter, it controls balance b/w above 2 terms

compute the derivative, $\frac{\partial J}{\partial X_{ai}}$ of objective function which minimizes

for any fixed $(a,i) \in D$,

$$\frac{\partial J}{\partial X_{ai}} = \frac{\partial}{\partial X_{ai}} \left(\sum_{(a,i)\in D} \frac{(Y_{ai} - X_{ai})^2}{2} + \frac{\lambda}{2} \sum_{(a,i)} X_{ai}^2 \right)$$

(predicted value)

$$= (X_{ai} - Y_{ai}) + \lambda X_{ai} = 0$$

to find what step we should take

$$\hat{X}_{ai} = \frac{Y_{ai}}{(1+\lambda)} \quad \text{for } a,i \in D$$

(predicted value)

for $a,i \notin D$

$$\frac{\partial}{\partial X_{ai}} = \frac{\partial}{\partial X_{ai}} \frac{\lambda}{2} \left(\sum_{(a,i)} X_{ai}^2 \right)$$

(as Y_{ai} is not given ∴ first term is taken 0)

$$= \lambda X_{ai} = 0$$

$$\boxed{X_{ai} = 0}$$

We considered only instance where rating was given but we need to consider all "

→ Cost function — In prev approach → we are changing where Y_{ai} present ⇒ adding noise

case 1 — ↳ where Y_{ai} was not present made it 0

Alternating minimization → Recommender system using matrix factorisation.

*//

minimize the objective function (cost function)

$$J(X) = \sum_{(a,i) \in D} \frac{(Y_{ai} - X_{ai})^2}{2} + \lambda \sum_{(a,i)} X_{ai}^2$$

with constraint on X

$X = UV^T$ (matrix factorisation) → O(n) parameters

for some $n \times d$ matrix U and $d \times m$ matrix V^T ↳ O(m) parameters

$UV^T \Rightarrow n \times m$ } → O(n+m) parameters → earlier there were O(nm) ↳ to guess.

↓

makes matrix "low rank" independent no. of rows & col.

	Comedy	Action
seen liked ← Yes	1	0
not seen not liked ← No	0	1

Rank of matrix X is the smallest no. d. for which ~~there exists~~ ∃ some n×d matrix U and m×d matrix V for which $UV^T = X$

$$X = \begin{bmatrix} 3 & 6 & 3 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

min value of d.

det(X) = 0.

Rank(X) ≠ 3

All minors of order 2 are 0 ∴ Rank(X) ≠ 2

∴ Rank(X) = 1

d = ~~lower~~ Rank (X)

$$\Rightarrow \begin{bmatrix} 1 & 2 & 3 \\ 5 & 10 & 15 \end{bmatrix}_{2\times 3} = \begin{bmatrix} 1 \\ 5 \end{bmatrix}_{2\times 1} \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}_{1\times 3}$$

multiple U, V may exist

$$= \underbrace{\begin{bmatrix} 10 \\ 5 \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} 0.1 & 0.2 & 0.3 \end{bmatrix}}_{V}$$

~~(crossed out)~~

Comedy	Action (d)		movie 1	movie 2	M3	M4	M5
1	0	User A					
0	1	" B					
1	0	" C					
1	0	" D					

4×2

so

	M1	M2	M3	M4	M5
Comedy	3	1	1	3	1
Action	1	2	4	1	3

} 2×5 = V

	Comedy	Action
user A	1	0
user B	0	1
" C	1	0
" D	1	1

U = 4×2 → d

3	1	1	3	1
1	2	4	1	3
3	1	1	3	1
4	3	5	4	4

} on multiply UV

→ with drawing some figures.
→ find these numbers

	F1	F2
user A	0.2	0.5
B	0.3	0.4
C	0.7	0.8
D	0.4	0.5

	M1	M2	M3	M4	M5
F1	1.2	3.1	0.3	2.5	
F2	2.4	1.5	4.4		

1.44

$X = UV^T$

↳ low rank matrix

Alternating minimisation

↳ given u, find v; given v, find u to find $\hat{x}$

$X_{ai} = U_a V_i$

a = 1 . . n (users)

i = 1 . . m (movies)

$$J(X_{ai}) = \sum_{(a,i) \in D} \frac{(Y_{ai} - X_{ai})^2}{2} + \frac{\lambda}{2} \sum_{(a,i)} X_{ai}^2 \quad \text{(earlier)}$$

(split X_{ai} into u & v)

cost function ~~J(UV)~~ ②

$$J(u,v) = \sum_{(a,i) \in D} \frac{(Y_{ai} - u_a v_i)^2}{2} + \frac{\lambda}{2} \sum_{a=1}^{n} u_a^2 + \frac{\lambda}{2} \sum_{i=1}^{m} v_i^2$$

Steps: To estimate X is now indirectly equal to estimating u and estimating v

Steps ① we will randomly pick up $\hat{v}$.

② we will compute the ~~use~~ u based on the random v such that given objective is minimised.

③ Once we get the new u we can then reevaluate v based on that u such that it still minimise the cost function.

This process is repeated until we get correct factorisation (factorisation close to X).

Therefore the above process is an iterative algorithm and it is said to be alternative minimisation because we set one variable and reevaluate another variable based on their values.

Q. Assume 2 users & 3 movies

$$Y = \begin{bmatrix} 5 & ? & 7 \\ 1 & 2 & ? \end{bmatrix}$$

for an initial V given as follows: $V = \begin{bmatrix} 2 \\ 7 \\ 8 \end{bmatrix}$

goal is to find $U = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$ and $V = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$ such that

they minimise the objective function ~~with~~ $J(u, v)$

(2) With constraint $X = UV^T$

$$UV^T = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}_{2\times1} \begin{bmatrix} 2 & 7 & 8 \end{bmatrix}_{1\times3}$$

$$= \begin{bmatrix} 2u_1 & 7u_1 & 8u_1 \\ 2u_2 & 7u_2 & 8u_2 \end{bmatrix}_{2\times3}$$

Find the values of u_1 & u_2 — whose Y is ? leave it

~~$\frac{\partial J}{\partial u_1}$~~ $J = \frac{(5-2u_1)^2}{2} + \frac{(7-8u_1)^2}{2} + \frac{\lambda}{2} u_1^2$

$$\frac{\partial J}{\partial u_1} = 0$$

$$\frac{\partial J}{\partial u_1} = (5-2u_1)\times(-2) + (7-8u_1)(-8) + \lambda u_1 = 0$$

$$= -10 + 4u_1 - 56 + 64u_1 + \lambda u_1 = 0$$

$$= -66 + 68u_1 + \lambda u_1 = 0$$

$$u_1 = \frac{66}{68+\lambda}$$

if $\lambda = 1$ $\quad u_1 = \frac{66}{69}$

for u_2

$$J = \frac{(1-2u_2)^2}{2} + \frac{(2-7u_2)^2}{2} + \frac{\lambda u_2^2}{2}$$

$$\frac{\partial J}{\partial u_2} = (1-2u_2)(-2) + (2-7u_2)(-7) + \lambda u_2$$

$$= -2 + 4u_2 - 14 + 49u_2 + \lambda u_2$$

$$= -16 + 53u_2 + \lambda u_2 = 0$$

$$u_2 = \frac{16}{53+\lambda}$$

if $\lambda = 1 \Rightarrow u_2 = \frac{16}{54}$

Date / /

$$UV^T = \begin{bmatrix} \frac{66}{69} \\ \frac{16}{54} \end{bmatrix}_{2\times1} [v_1 \quad v_2 \quad v_3]_{1\times3}$$

$$= \begin{bmatrix} \frac{66}{69} v_1 & \frac{66}{69} v_2 & \frac{66}{69} v_3 \\ \frac{16}{54} v_1 & \frac{16}{54} v_2 & \frac{16}{54} v_3 \end{bmatrix}$$

Value of v_1 (earlier we were seeing row now we col)

$$J(v_1) = \frac{\left(5 - \frac{66}{69} v_1\right)^2}{2} + \frac{\left(1 - \frac{16}{54} v_1\right)^2}{2} + \frac{\lambda}{2} v_1^2$$

$$\frac{\partial J}{\partial v_1} = 0$$

X

Deep Neural Network

(210)