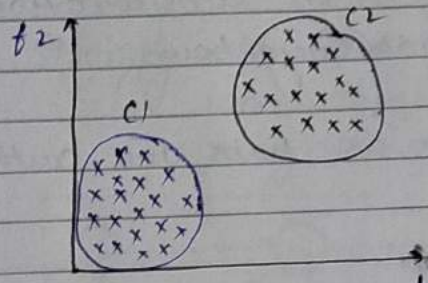
UNSUPERVISED LEARNING

✱ Clustering :-

$D = \{x\}$: no y_i's

Task : Group/Cluster "Similar" data points.



∴ All points are clustered in 2 diff" groups.

∴ The process of grouping 'similar' points into diff groups / clusters is known as clustering.

Clustering :- K-Means, Hierarchical clustering, DBSCAN.

✱ Applications of clustering :-

eg: ① ecommerce :- Amazon, Alibaba, Ebay, Flipkart.

Task :- ↳ Group "Similar" customers based on their purchasing behavior → money
↳ credit
↳ products

② Image Segmentation (Computer Vision and image processing)
↳ grouping / clustering similar pixels

③ Amazon food reviews :- Manually label all the reviews as +ve or -ve to create a dataset. (very time consuming and expensive)

⇒ D = 1M reviews no y_i's
↳ (1) cluster them in 10K groups of similar words (Bow, tf-idf, w2v)
↳ (2) Review & label each cluster → C_i → pick one point randomly
+ve
-ve
if +ve
if -ve

⊛ Metrics for clustering :-

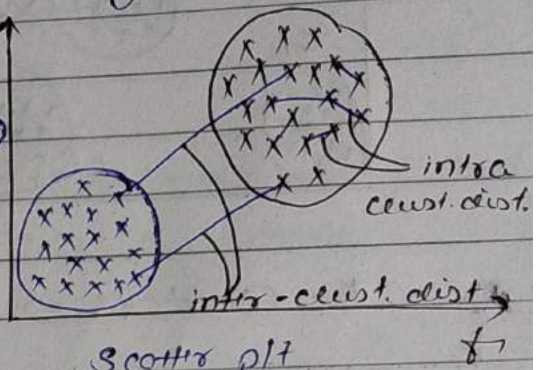
Geometrically :- what is a good clustering result?

Given :- $D = \{x_i\}$ $x_i \in \mathbb{R}^2$

→ Grouped in clustrs such that intra- is (small)

intra-clustr :- with in a clustr

intr-clustr :- across or b/w clustrs → (large)



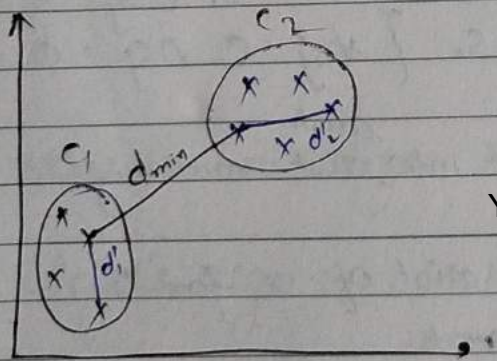
⊛ Ideally :- inter clustr dist. → v. high
intra clustr dist. → v. low.

⊛ Dunn Index :-

$$D = \frac{\min_{i,j} d(i,j)}{\max_{k} d'(k)}$$

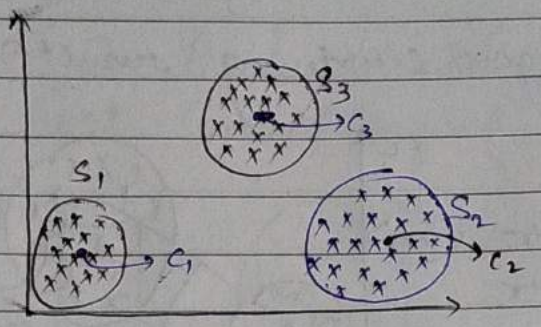
$d(i,j)$ → dist. b/w c_i, c_j

K clustrs $= \{c_1, c_2, c_3 \dots c_k\}$



- for every ptsin c_1 find dist. to all pts in c_2 : $\{d_1, d_2, d_3 \dots d_n\}$ → take min among others
- find all intra clustr dists and take max $\max\{d'_1, d'_2\}$

✱ K-Means: Geometric Intuition, Centroids



at $K = 3$
↳ no. of clusters (Hyperparam)

$S_1 \cap S_2 = \phi$
$S_1 \cap S_3 = \phi$
$S_2 \cap S_3 = \phi$

$C_1, C_2, C_3 \rightarrow$ Centroids
$S_1 \cup S_2 \cup S_3 = D$

K-clusters $\Rightarrow$ K-centroids and K sets.

$$C_i = \frac{1}{n} \sum_{x_j \in S_i} x_j \leftarrow \text{mean-point to } S_i$$

K-means :- Centroid based clustering.

✱ K-Means :- Mathematical formulation of Objective fn.

$D = \{ x_1, x_2 \dots x_n \}$

Task :- K-centroids :- $C_1 C_2 C_3 \dots C_k$ $\{ \forall i \ x_i \in S_j$
sets :- $S_1 S_2 S_3 \dots S_k$ $\{ \forall i,j \ S_i \cap S_j = \phi$

$$\Rightarrow \underset{C_1 C_2 \dots C_k}{\text{argmin}} \sum_{i=1}^{k} \sum_{x \in S_i} \underbrace{\| x - C_i \|^2}_{\text{intra cluster dist.}} \leftarrow \text{minimizn.}$$

↳ SQ. dist. of x for C_i

such that $x \in S_i$
$S_i \cap S_j \neq \phi$ → constraints.

Very hard to solve as it's NP-hard problem
it takes exponential time complexity.

↓

Solved using approximation (Lloyd algo)

✱ K-Means Algorithm :- (Lloyd's algo)

① initialization :-
↳ randomly pick k pts from D and call them $c_1, c_2, c_3 \ldots$

② Assignment :-
for each point in D select the nearest c_j and find dist. (x_i, c_j) $\forall j = 1, 2, \ldots k$ and add x_i to set S_j

③ Recompute Centroid :-
↳ recalculate / update c_j's as follows

$$c_j = \frac{1}{|S_j|} \sum_{x_i \in S_j} x_i$$

$|S_j|$: no. of pts in set S_j
$\sum_{x_i \in S_j} x_i$: mean pt

④ Repeat step ② and ③ until convergence.
↳ when centroids doesn't change much.

✱ How to initialize : K-mean++

Lloyd's algo :- initialization step is done using randomly picking points. BUT there is a problem with it.

Problem :- initialization sensitivity.

final cluster is dependent on how points are picked.

cs.wmich.edu - clusteringAnalysis.pdf

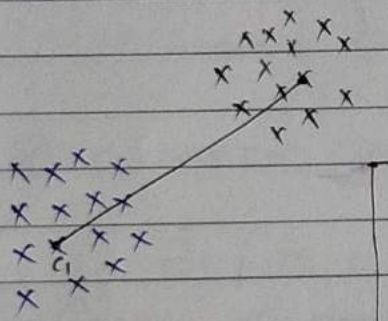
Q) How to deal with initialization sensitivity problem?

A) i) Repeat K-means multiple times with diff initialization
↳ pick the best clustering based on:
smaller intra clust dist and
larger inter clust dist.

ii) K-means++ :- Replace random initialization with smart init. $\{c_1, c_2, c_3, \ldots c_k\}$

probabilistic approach

↳ ① Pick the 1st centroid randomly → c_1 for 'D'
② $\forall x_i \in D$ create a distribution as below.
$x_i \rightarrow dist^2(x_i, \text{nearest centroid})$
③ pick apt from $D - \{c_1\}$ with a prob. proportional to d_i



x_1	d_1
x_2	d_2
x_3	d_3
x_4	d_4
⋮	⋮
x_n	d_n

$\|(x_i - c_1)\|^2$

Q) Why select centroid probabilistically? why not deterministically?

A) Deterministic approach :- Select point with max. distance from already selected nearest centroid.
↳ this approach is heavily affected by the outliers.

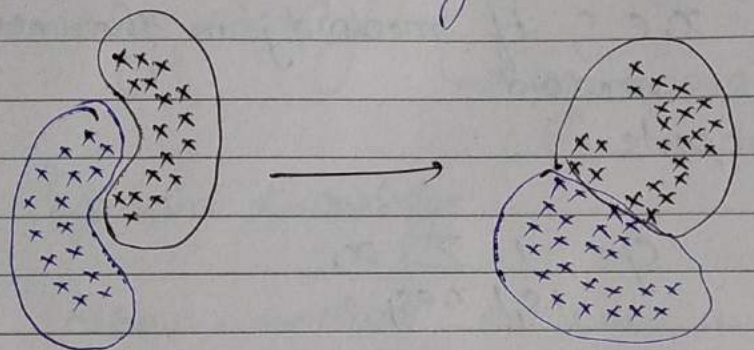
Probabilistic approach :- less affected by outliers.

✱ Failure cases / Limitations :-

① Different Size of clusters :- K-means tries to make clusters of same size, and hence if the original clusters are of diff size K-means may not work well.

② Different density of clusters.

③ Clusters with Non globular shape. (non convex)



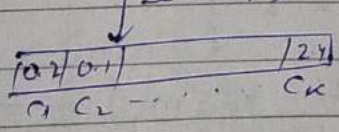
original K-mean

④ Outliers

NOTE :- To solve the above limitations we can ↑ 'k' but it doesn't work very well in all the cases.

✱ K-Medoids :-

problem with K-mean :- it is not interpretable



→ from this we can't understand much.

So instead of giving me c_j's computed using means, if $x_{10} \in \mathcal{N}$ as a first centroid was given it is more usable

Big idea :- what if each centroid is a datapoint in D?

↓

it is called as K-medoids

⊛ Portioning around medoids (PAM) :- Algorithm to implement K-medoids.

step1:- Initialization :- K-Means++ → probabilistic method pick K pts from D

step2:- Assignment :- closest medoid

$x_i \in S_j$ if medoid j is the closest medoid

step3:- Update / Recompute :-

K-means: $c_j = \frac{1}{|S_j|} \sum_{x_i \in S_j} x_i$

K-medoids (PAM): a) Swap each medoid with a non-medoid pt.

b) if loss ↓ keep the swap else undo the swap.

eg:- M_1 (x_1) x_2 x_3 x_4 x_5 M_2 (x_6) x_7 x_8 x_9 x_{10}

non-medoid non-medoid

∴ loss in K-means:- $\sum_{i=1}^{k} \sum_{x \in S_i} \|x - m_j\|^2$ → nothing but distance

m_j → medoid j

eg: m_1 (x_1) x_2 x_3 x_4 x_5 m_2 (x_6) x_7 x_8 x_9 x_{10}

a) loss value $x_1 = m_1$ $x_6 = m_2$ → l_1

b) swap $M_1 = x_2$; x_1 as non medoid pt.

$M_2 = x_6$

↳ loss value $M_1 = x_2$, $M_2 = x_6$ → l_2

if $l_2 < l_1$

$m_1 = x_2$

$m_2 = x_6$

else

$m_1 = x_1$

$m_2 = x_6$

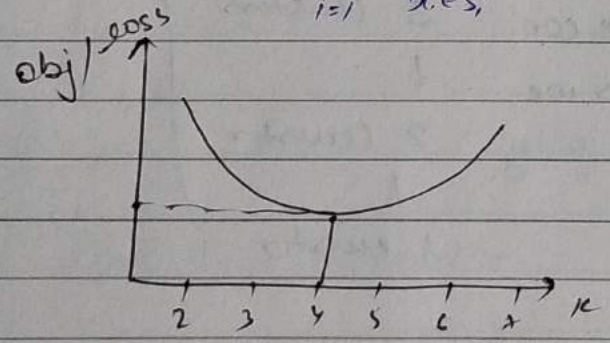
NOTE :- If similarity or distance matrix is given K-medoid can be implemented easily.

* Determining the Right 'k' :- → hyperparameter.

① Domain Knowledge

② elbow-method or scree method.

$$\text{loss} : \sum_{i=1}^{K} \sum_{x \in S_i} \|x - c_i\|^2 \longrightarrow \text{minimize}$$



* Time Complexity :-

K means = $O(nKdi) \approx O(nd)$

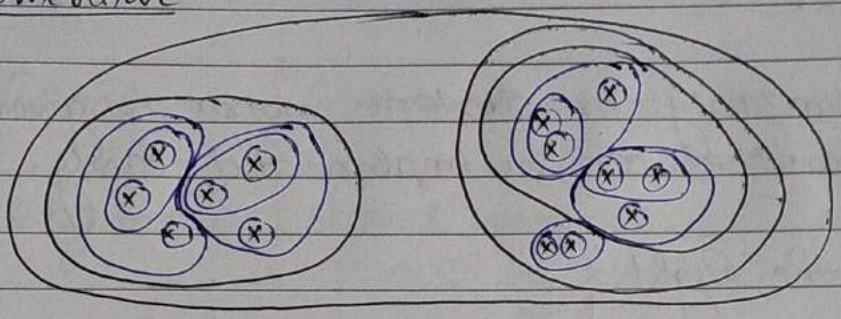
i → iteration, d → dim, K → no. of clusters, n → no. of pts

Space Complexity : $O(nd + Kd) = O(nd)$

⊛ Hierarchical clustering :-

Two types :- Agglomerative and Divisive
(more popular.)

- Agglomerative



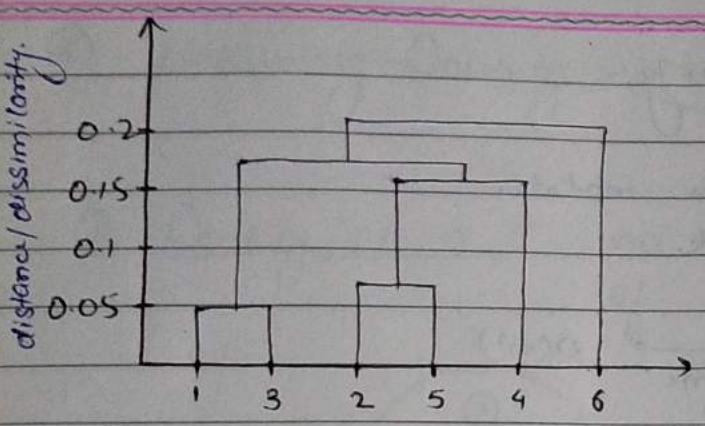
Initially each point is a cluster : 14 clusters
↓
Now it groups close-by clusters : 9 clusters
↓
5 clusters
↓
3 clusters
↓
2 cluster
↓
1 cluster.

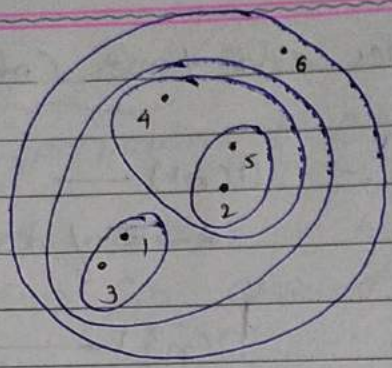
Hierarchy.

NOTE :- No need to specify no. of clusters (K). Once the hierarchy is build, we can select how many clusters we want.

- Divisive :- Opposite of Agglomerative
  - Initially whole dataset / all points are part of 1 big cluster. Then in each step points will be divided into various clusters.

1 cluster → 2 clusters → 3 clusters → 5 clusters → 9 clusters
↓
14 clusters.





→ Dendrogram :- A tree like diagram that records the sequences of merges or splits.

• How to define inter-cluster similarity?

Can be Knn clustering

Single link

=> Min :- $Sim(C_1, C_2) = \min Sim(P_i, P_j)$ such that $P_i \in C_1, P_j \in C_2$

Complete link

=> MAX :- $Sim(C_1, C_2) = \max Sim(P_i, P_j)$ such that $P_i \in C_1, P_j \in C_2$

=> Group avg :- $Sim(C_1, C_2) = \frac{\sum_{P_i \in C_1, P_j \in C_2} Sim(P_i, P_j)}{|C_1| * |C_2|}$

$|C_1|$ ↓ Size of C_1 ; $|C_2|$ ↓ Size of C_2

Youtube: Programming Cradle

=> Dist. B/w Centroid :-

• Limitations :-

Min :- Sensitive to outliers / noise.

Max :- • Biased towards globular clusters
• Tends to break large clusters into diff parts.

Group avg :- Biased towards globular clusters.

✱ Space and time Complexity :-

Space :- $O(n^2)$ ⟶ Sim. matrix.
n → # of datapts.

Time :- $O(n^3)$ ⟶ (with some optimizn) $O(n^2)$

$O(n^3)$ { almost n iterations
↳ grouping 2 clusters (const.)
↳ updating Sim mat ($O(n^2)$)

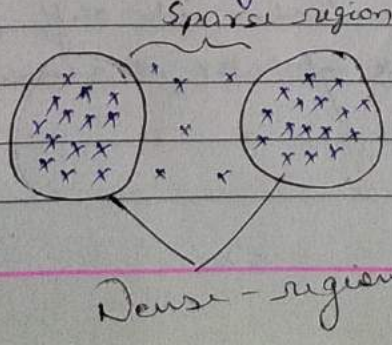
✱ Limitations of Hierarchical clustering.

1. No mathematical objective fn that we are directly solving. Hence we can not link it with other concepts.
2. Difficult to select one from Min, Max, avg, clust.
3. Space and time complexity. (can't be used for large dataset)

✱ DBSCAN :- Density based clustering

↳ Density-based spatial clustering of applications with noise (DBSCAN)

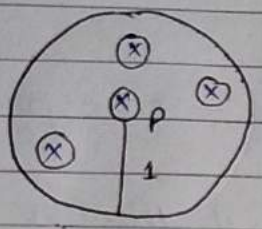
Dense-region ⟶ clusters (valid data points)
Sparse-region ⟶ noise (outliers)



✱ Measuring Density :- MinPts, Eps (ϵ)
hyperparam of DBSCAN

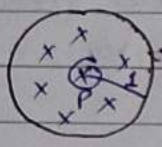
① Density at a P : no of pts within a hypersphere of radius Eps around P.
↓ wt (1)

eg:-

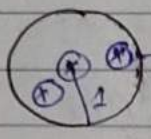


⇒ density = 4

② Dense region :- a hypersphere / circle of radius Eps that contains atleast MinPts points
wt ④



→ dense region bcoz 7 points in region which is more than 4



→ sparse region bcoz no. of pts = 3 < 4

✱ Core point; border pt and noise pt.

① Core point (P) :- if p has ≥ MinPts points in an Eps radius around it then pt is core pt.

② Border pt (P) :- i) p is not a core pt. ⇒ p has < MinPts pts in Eps radius
ii) p ∈ Neighbourhood (Q) where Q is core pt
↓
dist(p, Q) ≤ Eps

③ Noise pts :- point which is neither core pt. nor Border pt.

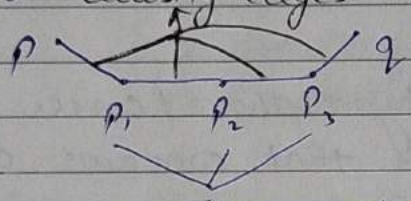
✱ Density edge & density connected pts.

① **Density edge** :- $p, q \rightarrow$ core point and $d(p,q) \leq eps$ then edge ↷ Density edge. connecting these pts is density edge.

p q

② **Density connected pts**:- $p, q \rightarrow$ core point.

density edges



if there is a path b/w two core points and intermediate points are also core points then 2 points are density connected pts.

✱ DBSCAN Algorithm :-

Steps

① $\forall x_i \in D \rightarrow$ Label them as core pt. border pt, noise pt.

Set of pts. labelled as core, ↖ border pts.

eg. $x_1 \rightarrow$ core pt $x_4 \rightarrow$ bord. pt.
$x_2 \rightarrow$ noise pt $x_5 \rightarrow$ core pt.
$x_3 \rightarrow$ core pt $x_6 \rightarrow$ bord pt.

S_i = rangeQuery (x_i, D, eps)

SK learn uses kdtree to implement range query in DBSCAN

② Remove noise pts.

③ For each core pt. p' not assigned to a cluster
- a) create a new cluster in the p
- b) add all pts that are density connected to p into this new cluster.

④ / each border pt. → assign it to the nearest core pt's cluster.

✱ Hyperparam :- MinPts, Eps

① MinPts :-

a) Rule of thumb :- MinPts $\geq d+1$ ← dimensionality

Typically MinPts $\approx 2*d$

b) If dataset is more noisy :- choose larger value of MinPts to remove noise.

c) MinPts often chosen by domain expert.

② Eps : cut MinPts : (4)

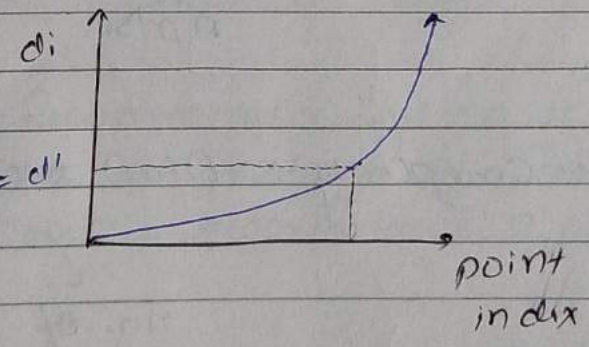
a) $\forall x_i \xrightarrow{\text{compute}} d_i$: d_i → dist from x_i to the 4th NN of x_i

b)

x_1	d_1
x_2	d_2
x_3	d_3
⋮	d_4
x_n	d_n

↓ sort d_i's in increasing order.

c) plot point index vs d_i and use elbow method.



if d_i → high: chances that x_i is noisy is also high

✱ Advantages :-

i) Resistant to Noise
ii) Can handle clusters of diff shapes and sizes
iii) No need to specify no. of clusters
iv) DBSCAN works really well with standard DB's such as MySQL.

Disadvantages :- i) Very Sensitive to hyperparameter
ii) High - dimensional data
(may not work very well with text data)
using Euclidian distance here can get in problem of curse of dimensionality.
iii) DBSCAN cannot cluster datasets well with large diff" in density. Since the minpts-e combination cannot then be chosen appropriately for all clusters.
iv) If domain expert could not understand the data and its scale suitable hyperparameter is difficult.

⊛ Time and Space Complexity :-

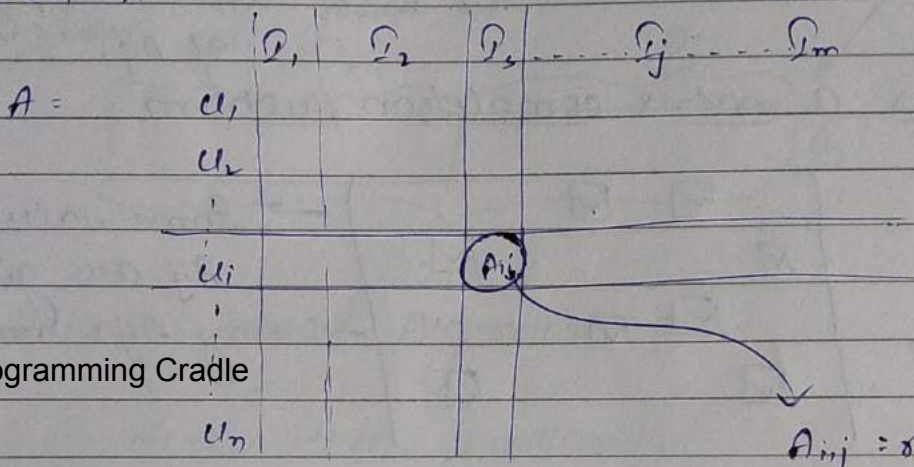
Time complexity ≈ O(n log n)
- n pts.
- log → range Query / region Query

Space Complexity : O(n) ≈ O(nd)
- n → no. of data pts
- d → often d is small as DBSCAN doesn't work very well with high dim.

✱ Problem formulation : Recommendir Systems

Recommender System :- Recommends relevant items to a user based on historical data.

Dataset : A



n : users
m : items

A_{ij} : rating given by u_i on I_j (1 to 5)
(or)
u_i watched I_j or not (0 or 1)

A : matrix A of ratings is very sparse.

Sparsity of A : $\frac{\text{number of non-empty cells}}{\text{total number of cells}} = \frac{5 \times 10^6}{10^{10}}$

$= 5 \times 10^{-4}$

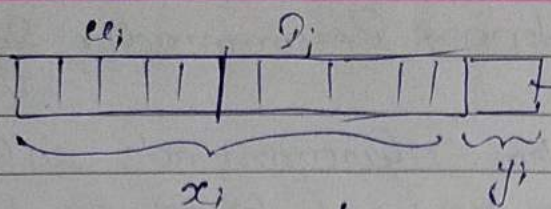
-> Big task of building Recommendir System is :-
Given a u_i and his/her ratings of previously watched movies / shows, recommend a new item I_j

(1) RS can be considered as regression problem if rating is given. (1 to 5)

(2) RS can be considered as classification problem if watched or not is given (0 or 1)

Given D : A

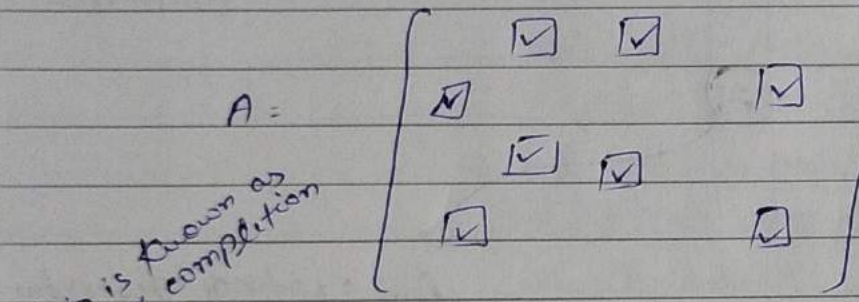
transform



D_{train} : $A_{ij} \neq$ Null

D_{test} : all the empty cells of A_{ij}

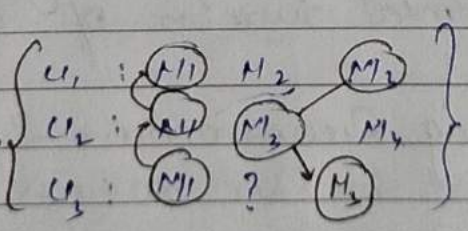
③ RS as a matrix completion problem



- Fill up the empty cells with reasonable values based on values in non empty cells.

④ Content based RS vs Collaborative filtering RS

⇒ Collab. fill. :-



not A

we see collab^n btw users/data pts. here collab fill.

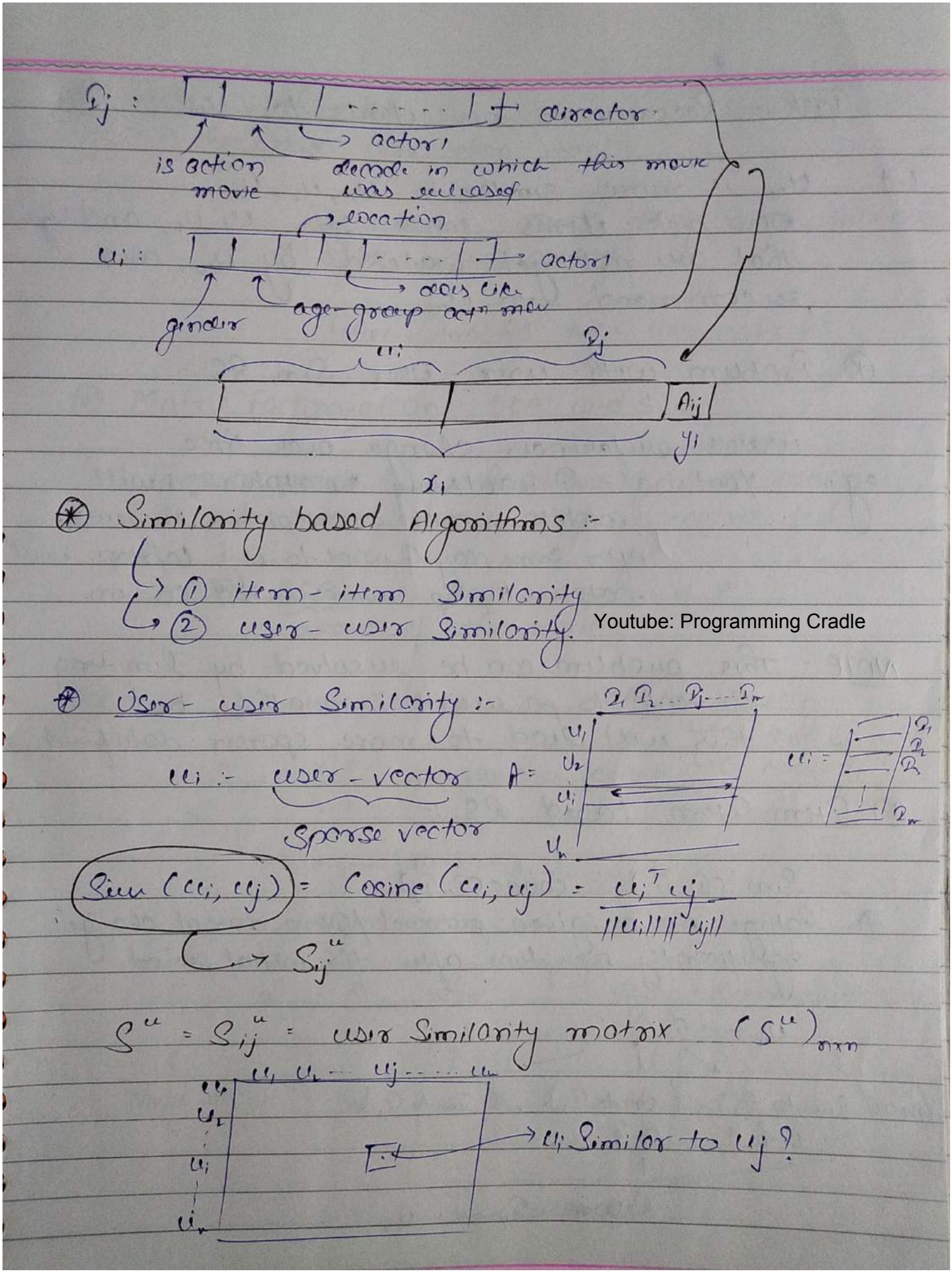
core-idea (assumption) :- users who agreed in the past tend to also agree in the future.

⇒ Content based :- ⇒ ~~dont~~ not use the A_{ij} info as label

⇒ represent Q_j & u_i into various features.

features ← m_j / Q_j : action, drama, 90's, beach, actor/ director,

features u_i : likes action movie, likes rom-com, male, ...



Similarity based Algorithms :-

1. item - item Similarity
2. user - user Similarity.

User - user Similarity :-

u_i :- user - vector

Sparse vector

$$Sim(u_i, u_j) = Cosine(u_i, u_j) = \frac{u_i^T u_j}{\|u_i\| \|u_j\|}$$

$\rightarrow S_{ij}^u$

$S^u = S_{ij}^u$ = user Similarity matrix $(S^u)_{n \times n}$

Task :- Recommend new items to U_{10}

Let U_{10} is more similar to U_1, U_2, U_3.
Now pick items liked by U_1, U_2, and U_3 that are not yet watched by U_{10}, and recommend to U_{10}

* Problem with user - user Sim RS :-

users's preferences change over time
eg :- Youtube :- I want to buy smartphone, i will watch more smartphone reviews.
After some day I want to buy laptop. i will watch laptop reviews and so on...

NOTE :- This problem can be resolved by limiting the data to recent 3 months, but this will lead to more sparser data set.

* Item - Item based RS :-

Sim (I_i, I_j) = cosine (I_i, I_j)

* Ratings on a given product / Item do not change significantly over time after the initial period

$U_{10} \rightarrow I_1, I_2, I_3$

{prod Sim to I_1}, {sim to I_2}, {Sim to I_3}

I_4 I_4

Recom. I_4 to U_{10}

⊛ Rule of thumb:-

When there more usr than items

↓

Item ratings don't change much over time after initial period.

↓

Item-Item RS over usr-usr RS

⊛ Matrix factorization: PCA and SVD

multiplicative

⇒ Given a matrix A if, the process of decomposition of A into several matrix is called as matrix fact.

$A = \underbrace{BCD}_{\text{product of other matrix.}}$ ∴ B, C, D are factors of A

PCA:- $X_{n \times d}$ → Data Matrix

PCA functionality:-
- ① dim ↓
- ② Mat. fact.

$S_{d \times d} = \frac{X^T X}{n-1}$: co-variance matrix

→ eigen-values :- $\lambda_1, \lambda_2 \ldots \lambda_d$

eigen-vectors :- $w_1, w_2 \ldots w_d$

$S_{d \times d} = W_{d \times d} \Lambda_{d \times d} W^T_{d \times d}$ } → eigen-decomposition of S

$S_{d \times d}$ → cover mat

$W = \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ w_1 & w_2 & \cdots & w_d \\ \downarrow & \downarrow & & \downarrow \end{bmatrix}_{d \times d}$; $\Lambda_{d \times d} = \begin{bmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ 0 & & \lambda_d \end{bmatrix}$

eigen vectors of S

diagnol mat.

Singular value Decomposition (SVD)

↳ Mat. fact. technique related to PCA

→ PCA is applied on co-var(s) mat. which is sqr. mat.

→ SVD can be applied on any rectangular mat.

•> PCA ⟵ $S_{d\times d}$ ⟵ $(D_{n\times d})$

•> SVD ⟵ $(D_{n\times d})$

$$X_{n\times d} = \underset{n\times n}{U}\ \underset{n\times d}{\Sigma}\ \underset{d\times d}{V^T}$$

cols of U, are called left singular vectors of X

cols of V, are called right singular vectors of X

Singular values (s_i): $\frac{S_i^2}{n-1} = d_i$

$$\Sigma = \begin{bmatrix} s_1 & 0 & 0 & 0 \\ 0 & s_2 & 0 & \\ & & s_3 & 0 \\ 0 & 0 & 0 & s_d \end{bmatrix}_{n\times d}$$ → diagonal matrix.

u_i = ith col of u = i^{th} eig. vect of $(X_{n\times d} X^T_{d\times n})$

v_i = ith col of v = ith eig. vect of $(\underbrace{X^T_{d\times n} X_{n\times d}}_{S\ (\text{cov. mat})})$

$cov(x) = S = W \Lambda W^T$ → PCA

W: eig-val of $S(w_i)$

Λ: eig val of $S(d_i)$

V: eig-vec of $(X^T X)$

$X = U \Sigma V^T$

U: eig-vec of (XX^T)

Singular val S_i : $\frac{S_i^2}{n-1} = d_i$

⊛ Non-negative Matrix Factorization :- (NMF or NNMF)

$$A_{n \times m} = B_{n \times d} (C^T)_{d \times m}$$

$B : n \times d$

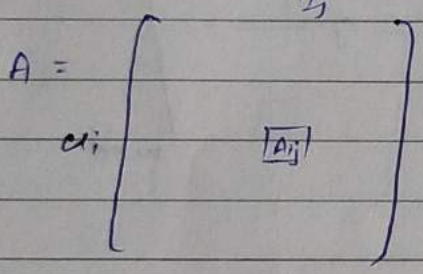
$C : m \times d$

Such that $B_{ij} \geq 0 \ \forall i,j$

$C_{ij} \geq 0 \ \forall i,j$

∴ All the values of mat B and C (factors of A) are non-negative

⊛ Matrix Factorization for collaborative filtering :-



-) A_{ij} : rating on I_j by u_i

-) sparse matrix : many empty cells

Let $A = B * C^T$

$A : m \times m$ ↙ user, ↓ item

$B : n \times d$

$C^T : d \times m$

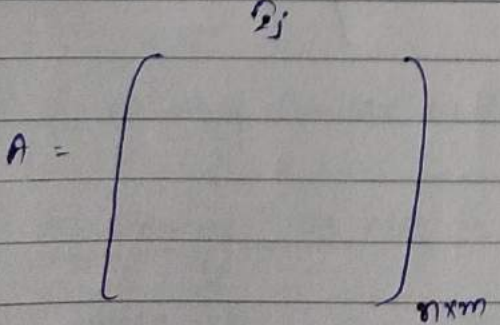
$B : n \times d$

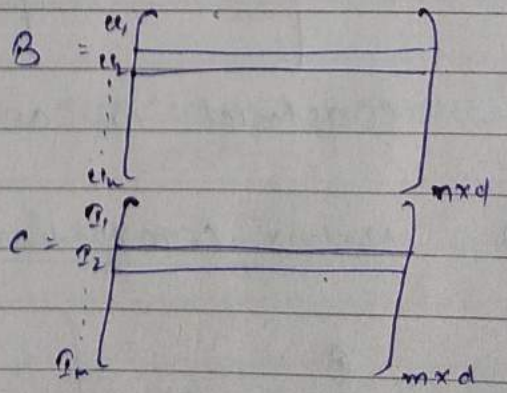
$C : m \times d$

$d > 0$

$d \leq \min(m, n)$

$$A_{ij} = B_i * C_j = B_i^T * C_j$$





=> Task : Find 'B' and 'C' :- (MF)

Optimization eqn (squared loss)

$$\underset{B,C}{\operatorname{argmin}} \sum_{ij} (A_{ij} - B_i^T C_j)^2 \longrightarrow (1)$$

such that A_{ij} is not empty

- After solving eqn (1) we will get

$B = \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_i \\ \vdots \\ B_n \end{bmatrix}$ ← B_i → $\Big]_{n \times d}$ $\quad C = \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_j \\ \vdots \\ C_m \end{bmatrix}$ ← C_j → $\Big]_{m \times d}$

① Solve optimization problem using SGD

$$\underset{B,C}{\operatorname{argmin}} \sum_{ij} (A_{ij} - B_i^T C_j)^2$$

valid non empty A_{ij}

② $B = \begin{bmatrix} \longleftarrow B_i \longrightarrow \end{bmatrix}$, $C = \begin{bmatrix} \longleftarrow C_j \longrightarrow \end{bmatrix}$

construct B and C

③ Matrix completion :-

$A = \begin{matrix} & 5 \\ 10 & \begin{bmatrix} \square \end{bmatrix} \end{matrix}$ ← empty

$A_{10,5} = \underbrace{B_{10}^T * C_5}$

we have computed this hence we can fill empty cell with this value

$A = [\quad] \rightarrow$ mostly empty

(actual rating: A)

(predicted rating: $\hat{A}$)

$\hat{A} = B * C^T$

$B * C^T \hookrightarrow$ computed using A_{ij} that are not empty by solving optimization problem

$\hat{A} \rightarrow$ without empty cell

⊛ Matrix factorization for feature engineering.

want to make u_i and I_j by using the data in A_{ij}

$A = $ (rows u_i, columns I_j)

- u_i [| | | |] $\rightarrow$ d dim
- I_j [| | | |] $\rightarrow$ d-dim

$\hookrightarrow$ using the data in A_{ij}

$A = B \, C^T$

$n \times m \quad n \times d \quad d \times m$

($n \rightarrow$ users, $m \rightarrow$ items)

$B = $ ($n \times d$ matrix, columns $1, 2, 3, \dots, d$; rows $1, 2, \dots, i, \dots, n$; row $i \rightarrow u_i$)

$B_i = i^{th}$ row of $B = u_i \in \mathbb{R}^d$

$C = $ ($m \times d$ matrix, columns $1, 2, \dots, d$; rows $1, 2, \dots, i, \dots, m$; row $i \rightarrow I_i$)

$C_i = i^{th}$ row of $C = I_i \in \mathbb{R}^d$

*) let $u_i, u_j \rightarrow$ very similar in their ratings of products/items

$\Rightarrow dist(u_i, u_j) \rightarrow$ small, $\in \mathbb{R}^d$

*) $I_i, I_j \rightarrow$ very similar in their ratings :- $dist(u_i, u_j) \rightarrow$ small, $\in \mathbb{R}^d$

⊛ clustering as MF :-

K-Mean :- $D = \{x_i\}$

$$\min_{c_i} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - c_i\|^2 \qquad x \in \mathbb{R}^d$$

lets define Z s.t $Z_{ij} = \begin{cases} 1 & \text{if } x_j \in S_i \\ 0 & \text{o/w} \end{cases}$

$Z =$ (K × n matrix; columns 1, 2, 3, ..., j, ..., n; rows 1, 2, ..., K; column j = 0, 0, 1, 0, 0)

$z_{ij} = 1 \Rightarrow$ this means point j belongs to cluster i

K → no. of clusters; n → no. of datapts/rows in original dataset.

$$\min_{c_i} \sum_{i=1}^{K} \sum_{x \in s_i} \|x - c_i\|^2 \quad \text{— (1)}$$

$$z_{ij} = \begin{cases} 1 & \text{if } x_j \in s_i \\ 0 & \end{cases}$$

$$\min_{c_i, z_{ij}} \sum_{i=1}^{K} \sum_{j=1}^{n} z_{ij} \|x_j - c_i\|^2 \quad \text{— (2)}$$

Let's define some matrices

$X =$ (columns $x_1, x_2, \dots, x_n$) — $d \times n$

each col is a data point

$C =$ (columns $c_1, c_2, \dots, c_K$) — $d \times K$

each col is centroid

$Z \to z_{ij} = \begin{cases} 1 & \text{if } x_j \in s_i \\ 0 & \text{otherwise} \end{cases}$

using above matrices we can write eq (2) as follows.

$\Rightarrow \min_{c,z} \|X - CZ\|^2$ ∴ find c, z such that $X - CZ$ is minimized ($X \approx CZ$ ideally)

such that $z_{ij} = 0$ or 1 — (1)

$\therefore z_j$: col of Z — (2)

↳ Sum of elements in a particular col is 1

NOTE :- • If we don't have above 2 constraints K-means ≈ MF

• If we have constraints K-means = MF + col. constraints + 0/1 constraints.

MF :- C and Z

NMF :- $Z_{ij} \geq 0$ and $C_{ij} \geq 0$

K-means :- $\Rightarrow Z_{ij} = 0$ or 1

$\Rightarrow$ col-sum of $Z_j = 1$

⊛ Hyperparamter tuning :d

$A_{n \times m}$ = rows 1, 2, 3, ..., n; columns 1, 2, 3, ..., m; $n \times m$

$A_{n \times m} = B_{n \times d} C^T_{m \times d}$

$d > 0$

d: hyperparamter $d \leq \min(m, n)$

$\Downarrow$

$d < m$ and $d < n$

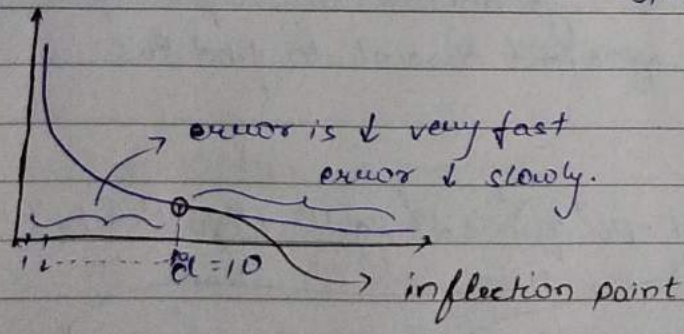
Methods

① problem specific :-

eg :- Given ratings data where (U_i rates $I_j \rightarrow A_{ij}$) then for each U_i and I_j we want 100-dim we don't want dim to be too small or too large

② Systmatic way :

optimization : $\min_{B,C} \underbrace{\sum_{ij} (A_{ij} - B_i^T C_j)^2}_{\text{loss / error}}$



✱ Netflix Prize :- (wikipedia)

performance-matrix → RMSE

$$\sqrt{\frac{1}{n}\sum_{i,j}(r_{ij}-\hat{r}_{ij})^2}$$

r_{ij} → rating, $\hat{r}_{ij}$ → pred. rat.

given u, m, r, d (date of rating)
u → user, m → movies, r → rating

Read paper [datajobs.com → recommender-Systems-[Netflix].pdf]

✱ Optimization :-

$$\min_{p,q} \left\{ \sum_{u,i} (r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2) \right\} \quad L$$

Solve:

① SGD :- $\frac{\partial L}{\partial p_u}$ and $\frac{\partial L}{\partial q_i}$ → takes more time.

② Alternating Least Squares (ALS)
alternating btw these → a) fix p_u ; gradient descent to find q_i's
b) fix q_i ; gradient descent to find p_u's

① repeat this

$$\min_{q,p,b} \sum_{u,i} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda\{\|q_i\|^2 + \|p_u\|^2 + b_u^2 + b_i^2\}$$

μ → const.

- μ : avg rating across all users and items; $\mu = \frac{1}{n}\sum_{u,i} r_{ui}$
- b_u : user bias — +ve if user gives higher rating than other users; -ve if user usually gives lower rating than other users
- b_i : item/movie bias
- $p_u^T q_i$: interaction effect

.> ut Joe rates Titanic as 3.9

3.9 can be decomposed as below:-

μ

$(3.0 + 0.5 + (-0.3) + 0.7)$

- 3.0 → avg rating.
- 0.5 → item/movie bias
- −0.3 → user bias
- 0.7 → $p_u^T q_i$ (interaction effect)

→ In Research paper:-

ratings by users and items are time dependent

- $b_u(t)$ → p_u
- $b_i(t)$ → $r_{ui}(t)$

✱ Cold Start Problem :-

→ new user (what item to recommend to this user?)

→ new Item (to whom this item should be recommended?)

✱ New user :- Top items based on users meta-data such as

content based RS
- → geo-location
- → browser (chrome, mozila etc)
- → device (android, ios)
- ;

✱ New Item :- meta data

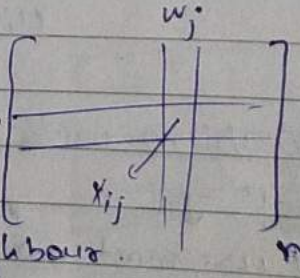
↳ category, price, description

✱ Word-Vectors using MF

→ W2V using SVD

① find co-occurance matrix (X)

x_{ij} = no. of times w_j occurs in context of w_i



w_j in context with w_i if :- they are in neighbour.

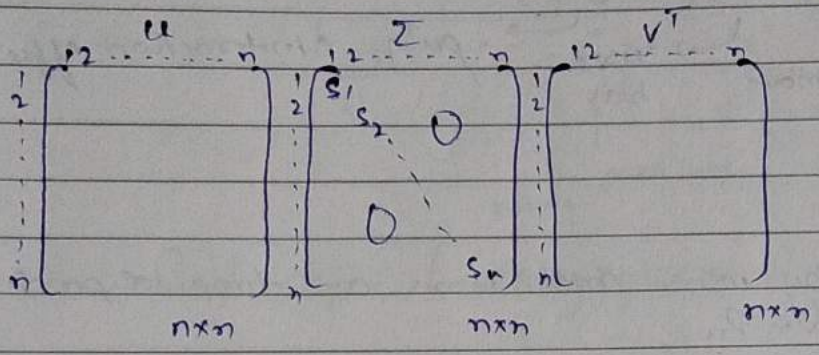
eg. review :- $w_1 w_2 w_3 w_4 w_5 w_6 w_7 w_8 w_9 w_{10}$ ∴ neighbour = (5)

w_j is in context with w_i if w_j is ~~not~~ at a distance less than 5

② Decompose $X_{n \times n}$ using SVD

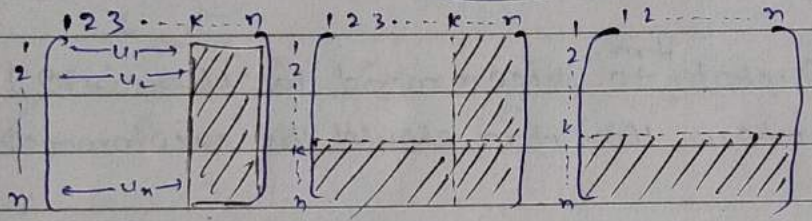
$$X_{n \times n} = U_{n \times n} \Sigma_{n \times n} V^T_{n \times n}$$

$\frac{s^2}{n-1} = \lambda$ eig. value of S (cov. mat)



$s_1 \geq s_2 \geq s_3 \ldots \geq s_n$

Truncated SVD :- $1 < K < n$ → hyperparameter



▨ :- Truncated / deleted part.

$n \times K$ → $\hat{U}$; $K \times K$ → $\hat{\Sigma}$; $K \times n$ → $\hat{V}^T$

① K - can be large for logistic Reg. but not for Decision tree, hence depends on model.

② Reconstruction error (diff b/w X and $\hat{X}$) should be less (use elbow method)

$$X \approx \hat{X} = \hat{U} \hat{\Sigma} \hat{V}^T$$

u_i : word-vector corresponding to word$_i$

③ $\hat{U}$: each row of $\hat{U}$ as $u_1, u_2 \ldots u_n$ (↑ word$_1$, ↑ word$_2$, ↑ word$_n$)

$u_i \in \mathbb{R}^K$

problem with :- co-occurrence matrix + Truncated SVD

Soln → use most imp. top m (top tf-idf) words instead to all the words.

↓ V. large n = no. of words

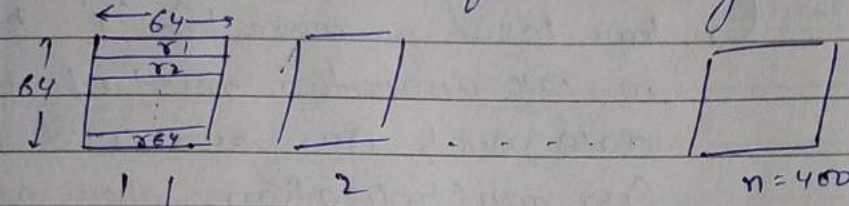
→ because of this computation of Truncated SVD is very expensive

uk.research.att.com/facedatabase.html

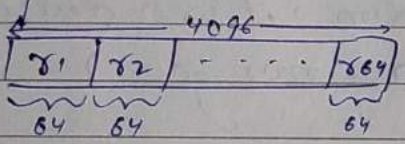
⊛ Eigen Faces : PCA on images

↳ early technique to recognize faces in images.

dataset consist of 400 imgs. 64×64 px.



64×64 = 4096



Step ① ⇒ $A = \begin{bmatrix} 1 & 2 & 3 & \cdots & 4096 \\ & & & & \\ 2 & & & & \\ 3 & & & & \\ \vdots & & & & \\ 400 & & & & \end{bmatrix}_{n \times d}$

n → no. of images; d → flat dim. of each img.

• Now from $A_{n \times d}$ we can compute $cov(A) = S_{d \times d}$ and :-

Step ② $S_{d \times d} = \omega_{d \times d} \wedge_{d \times d} \omega^T_{d \times d}$

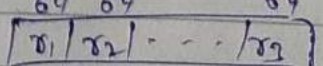
ω = $[v_1\ v_2 \cdots v_d]$ (eigen vectors)

$\wedge = \begin{bmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ 0 & & \ddots \end{bmatrix}$ (eigen values)

$\lambda_1 > \lambda_2 > \cdots > \lambda_d$

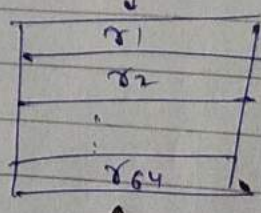
from d-dim to K-dim such that $K < d$ using dimensionality reduction (Drop k eigen-val → eigen vect.)

Step ③ $\omega_k = \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ v_1 & v_2 & \cdots & v_k \\ \downarrow & \downarrow & & \downarrow \end{bmatrix}_{d \times k}$ $\quad v_i \in \mathbb{R}^d$, $d = 4096$ →



Step ④ $A_{n \times d}\ \omega^k_{d \times k} = \tilde{A}_{n \times k}$ →

$\tilde{A} = \begin{bmatrix} 1 & 2 & \cdots & k \\ & & & \\ \tilde{2}_1 & & & \\ \tilde{2}_2 & & & \\ \vdots & & & \\ \tilde{2}_{400} & & & \end{bmatrix}$



eigen faces.

NOTE :- Eigen faces is nothing but taking face dataset and applying PCA on it.

⊛ Important point:-

time taken is 'T'

$x_i \in R^{17k}$ (is a sparse data) ⟶ Similarity mat

↓ PCA / SVD

$x_i \in R^{500}$ ⟶ Similarity mat

time taken is more than 'T' because: in 17k dimention most of the data points are zero and hence less multiplication, where as in 500 dim. it is a dense data (most pts are non zero)

Youtube: Programming Cradle