☆ Dimensionality Reduction (Usually used for unsupervised learning)

2D, 3D : Scatterplots
4D, 5D, 6D : pair plot

but what if 10-D, 100-D, 1000-D plots will not work

so to visualize data we try to reduce the dimensionality.

☆ Row vector and Column vector

$$\text{flower } [\underbrace{SL, PL, SW, PW}_{\text{real values}}]$$

$\mathbb{R} \rightarrow$ Real space | Real numbers

i^{th} point : $x_i \in \mathbb{R}^d$

$$x_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{id} \end{bmatrix}_d \text{ : column vector}$$

$x_i = [2.1, 3.2, 4.1, 1.2] \rightarrow$ data point (row vector)

default representation is column vector

☆ How to represent a dataset?

$\mathcal{D} = \{x_i, y_i\}_{i=1}^{n}$ n = number of datapoint

$x_i \in \mathbb{R}^d$

$y \in \{ ₹1, ₹2, ₹3 \}$

Classifications / labels

Data matrix

$$X = \begin{array}{c|c|} & f_1 \quad f_2 \quad f_3 \quad f_4 \dots \quad f_d \\ \begin{matrix}1\\2\\3\\ \vdots \\ n\end{matrix} & \begin{bmatrix} \\ \longleftarrow \quad x_i^T \quad \longrightarrow \\ \\ \end{bmatrix}_{n \times d} \end{array}$$

each datapoint : row
each column : feature (specific)

◇ Data Preprocessing : Feature Normalisation :

Column $a_1, a_2, \dots \quad a_n \rightarrow$ n-values of f_j

$$\max(a_i) = a_{max} \geq a_i$$
$$\min(a_i) = a_{min} \leq a_i$$

$$a_i' = \frac{a_i' - a'_{min}}{a_{max} - a_{min}} \qquad a_i' \in [0, 1]$$

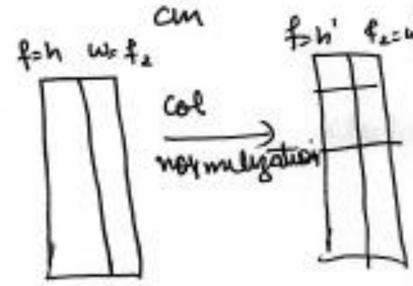
$$a_{min} = \frac{a'_{min} - a'_{min}}{a_{max} - a_{min}} = 0$$

$$a_{max} = \frac{a_{max} - a_{min}}{a_{max} - a_{min}} = 1$$

Transform the given data into a_i'

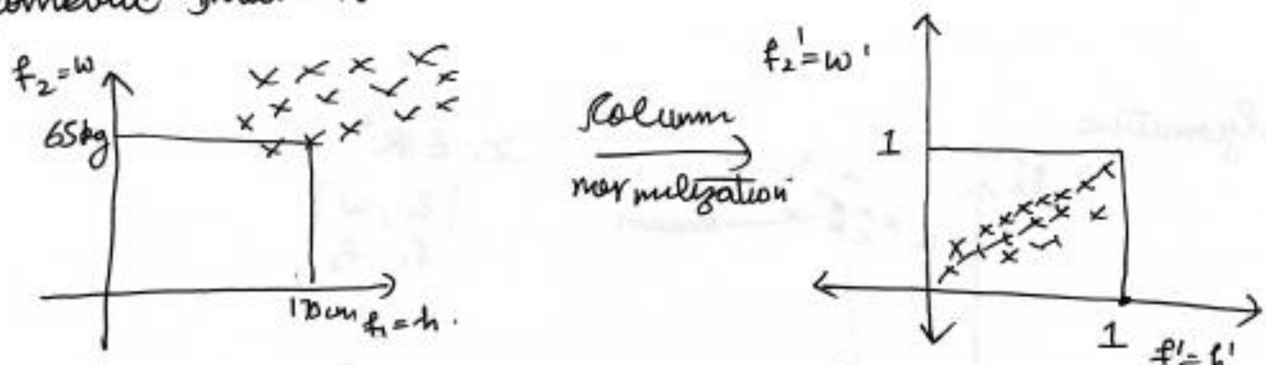
Why? height and weight as 2 features

cm kg

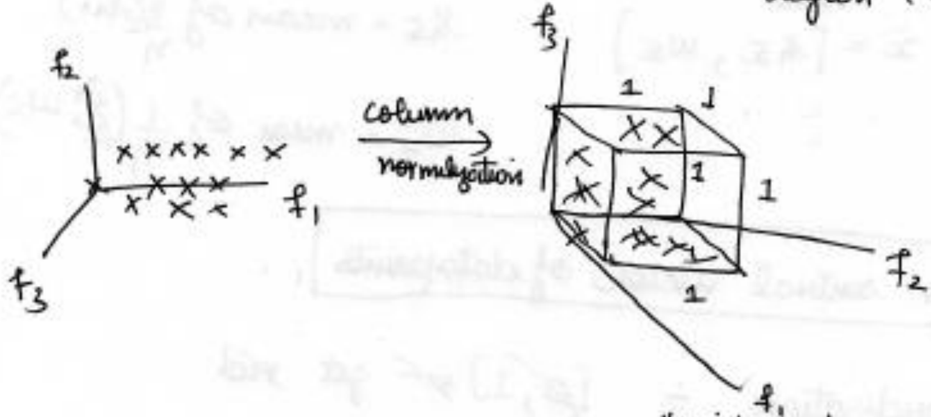


getting rid of scale.

◇ Geometric Intuition



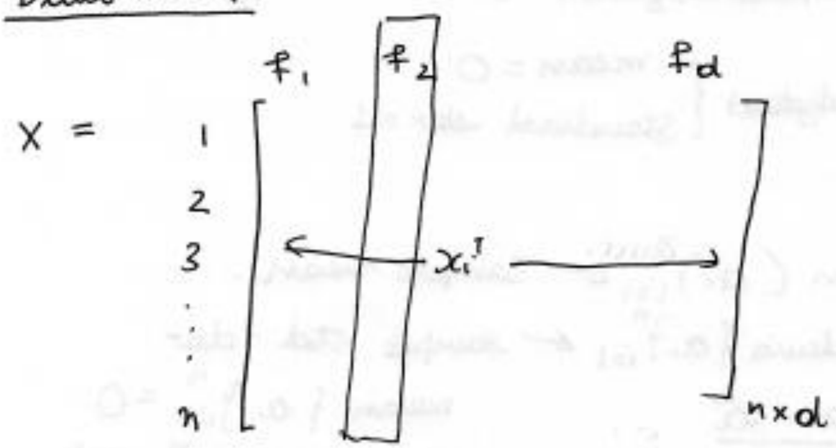
All points brought into region (unit square)



unit cube

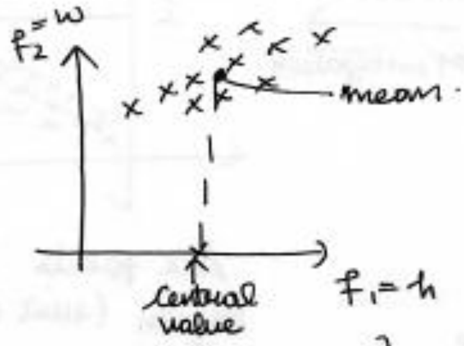
NOTE : Getting rid of scales

◇ Data mean



mean vector $\rightarrow \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i = \frac{1}{n} (x_1 + x_2 + \dots + x_n)$

Geometric



$x_i \in \mathbb{R}^2$

$[h, w]$
$f_1 \quad f_2$

$\bar{x} = [h_{\bar{x}}, w_{\bar{x}}]$

$h_{\bar{x}}$ = mean of $\frac{1}{n}(\sum_{i=1}^{n} h_i)$

$w_{\bar{x}}$ = mean of $\frac{1}{n}(\sum_{i=1}^{n} w_i)$

$\boxed{\text{mean vector central vector of datapoints}}$

◇ Column standardization : $[\cancel{0}, 1] \leftarrow$ get std

Column normalization : $[0, 1] \leftarrow$ get rid of scales of each features.

→ Column standardization : It is more offens used

Column standardization $\begin{cases} \text{mean} = 0 \\ \text{Standard dev} = 1 \end{cases}$

$\bar{a} = \text{mean}(a_i)_{i=1}^{n} \leftarrow$ sample mean.

$s = \text{std-devia}\{a_i\}_{i=1}^{n} \leftarrow$ sample std-dev

$a_i' = \frac{a_i - \bar{a}}{s}$ $\qquad$ mean $\{a_i\}_{i=1}^{n} = 0$

std-dev $\{a_i\}_{i=1}^{n} = 1$

☆ Standard Normal variate (Z) ⇨ from normal distribution

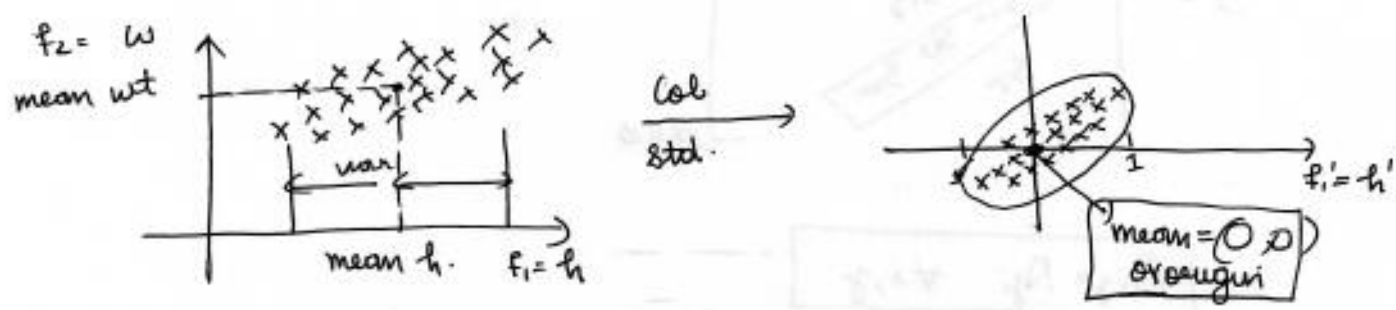
$$Z = \frac{x - \mu}{\sigma}$$

$$x \sim N(\mu, \sigma)$$

$$Z \sim N(0, 1)$$

☆ but for column standardization

→ The data can come from any distributions



std - dev = 1

mean vector to origin.

and $\boxed{std = 1}$ which can be compressed or expand s.t it is 1.

Column standardization : mean centering + scaling

$$\underset{\text{origin}}{\downarrow} \qquad \underset{sd = 1}{\downarrow}$$

☆ Covariance of a data matrix

$X = [\ \ ]$; $S =$ cov of X $[\ \ ]$

↑ Square matrix

$$\boxed{cov(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)}$$

$$S_{ij} = cov(f_i, f_j)$$

$$cov(f_i, f_j) = var(f_i)$$

$$\begin{cases} cov(X,Y) = var(X) & \text{——(1)} \\ cov(f_i, f_j) = cov(f_j, f_i) & \text{——(2)} \end{cases}$$

$$S = \begin{bmatrix} S_{11} & & S_{ij} & \\ & S_{22} & & \\ S_{ji} & & S_{33} & \\ & & & S_{44} \end{bmatrix}_{d \times d}$$

symmetric matrix

$$\boxed{A_{ij} = A_{ji} \quad \forall i,j}$$

avg feature

$$cov(f_1, f_2) = \frac{1}{n}\sum_{i=1}^{n}(x_{i1} - \mu_1)(x_{i2} - \mu_2)$$

μ_1: mean (f_1) $\quad$ μ_2: mean(f_2)

$\mu_1 = 0$
$\mu_2 = 0$ $\quad$ * S has been column standardised

$$cov(f_1, f_2) = \frac{1}{n}\sum_{i=1}^{n}(x_{i1}) * (x_{i2})$$

$$cov(f_1\ f_2) = (f_1^T\ f_2) * \frac{1}{n}$$

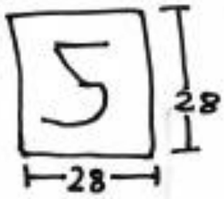
MNIST dataset

data visualization of high dimensions

Basic → database of hand written digits

It is basic simple computer vision dataset.

It consists of 28×28 pixel images of handwritten digits (0-9)

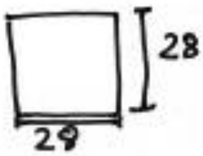


28×28 pixels.

60K training datapoints

10K Test datapoints

$$D = \{x_i, y_i\}_{i=1}^{60K}$$

x_i : 

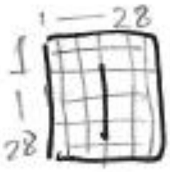
$y_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

objective : Given Image determine the number user has written.

x_i is image of 28×28 so usually x_i are vectors (colum

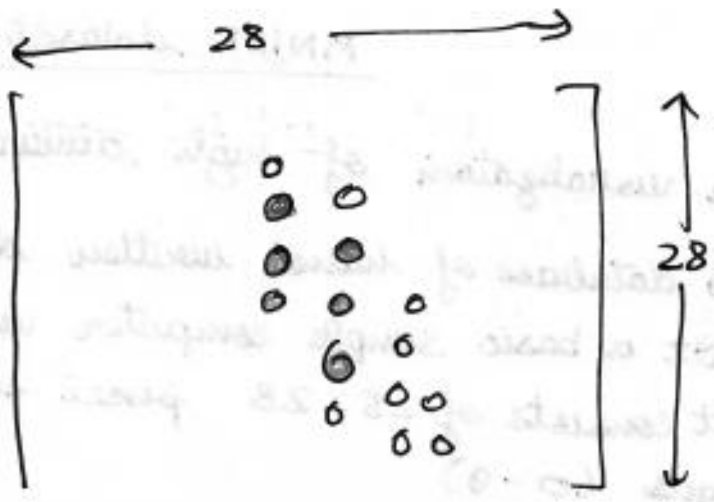
$x_i = \begin{bmatrix} \ \end{bmatrix}$ $\quad x_i \in \mathbb{R}^d$

for eg



where fully dark put

=



O → light grey

fully dark

O denotes 1

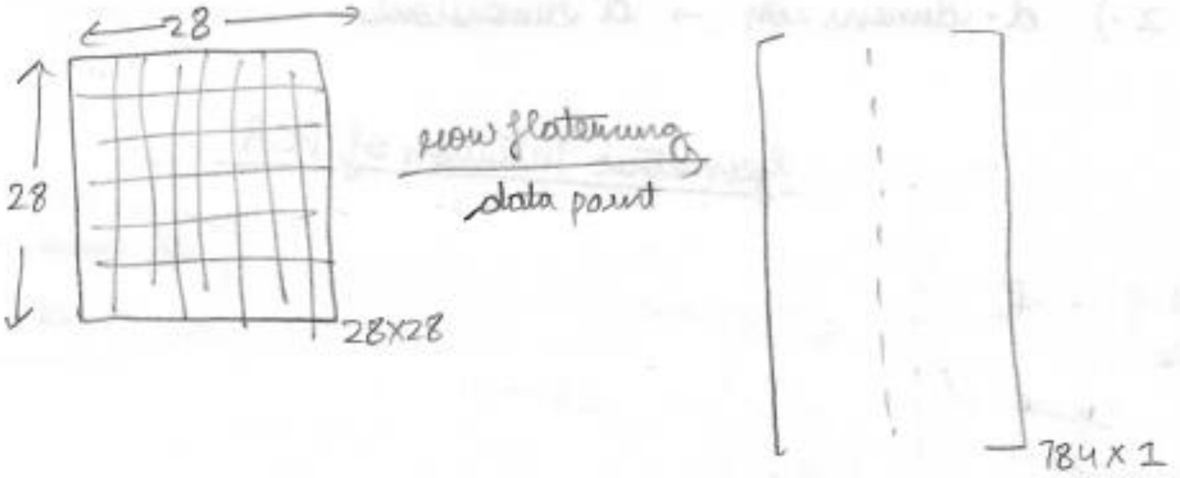
x_i = image $\left[\quad \right]_{28 \times 28}$

numerical / real matrix

NOTE :
matrix representation of image

FLATENING

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 1 & 2 & 4 & 6 & 8 \\ 2 & 3 & 2 & 1 & 8 & 2 \\ 3 & 2 & 1 & 6 & 8 & 4 \\ 4 & 3 & 2 & 1 & 8 & 2 \\ 5 & 4 & 2 & 6 & 8 & 1 \end{array}_{5 \times 5} \xrightarrow{\text{FLATENING}} \begin{bmatrix} 1 \\ 2 \\ 4 \\ 6 \\ 8 \\ 3 \\ 2 \\ 1 \\ 8 \\ 2 \\ \vdots \\ 8 \\ 1 \end{bmatrix}_{25 \times 1}$$



$$X = \begin{array}{c} f_1 \quad f_2 \quad f_3 \quad \cdots\cdots \quad f_{784} \\ \left[\begin{array}{c} \longleftarrow 784 \longrightarrow \\ \\ \\ \end{array} \right]_{n \times d} \end{array} \qquad \left[\; y_i \in \{0, \ldots 9\} \; \right]_{n\times 1}$$

60K

$n = 60K$
$d = 784$

784 - dimensional dataset

Principal Component Analysis (PCA)

Why? (dimensionality reduction)

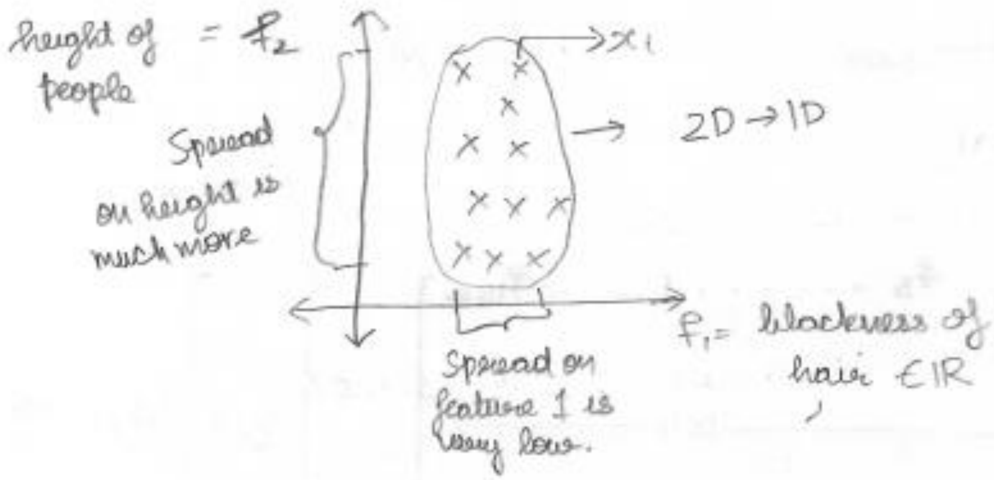
d-dimensional → d'-dimensional
$x_i \in \mathbb{R}^d$ $(d' < d)$

MNIST → 784 dimensions → 2-dimension
(visualize)

2.) d - dimension → d' dimensions

Geometric Intuition of PCA

d - dimen → d' dimension
$d' < d$



Indians ←

$$X = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \vdots \\ n \end{array} \begin{array}{c} f_1 \quad f_2 \\ \left[\begin{array}{cc} & \\ & \\ & \\ & \\ & \end{array} \right] \end{array}$$

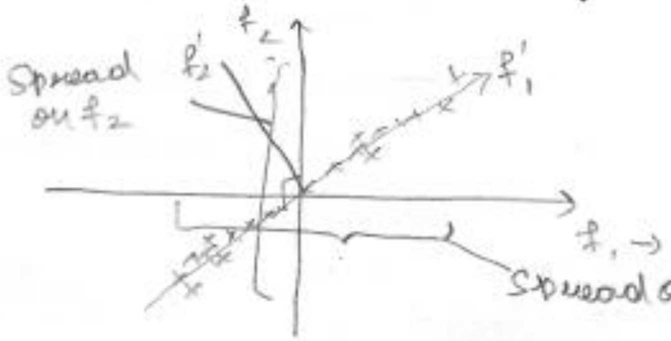
$$X' = \begin{array}{c} f_2 \\ \left[\begin{array}{c} \\ \\ \\ \end{array} \right] \end{array}$$

As spread on f_1 is low (low variance)
on the other hand f_2 variance is large
If forced to convert from 2D → 1D we would consider f_2 as spread is more.

NOTE : preserving the direction with maximal spread (more information)
(spread is measure of information)

Example: 2 X = 2 dimensional dataset

column standardized $\begin{cases} \text{mean}\{f_1\} = \text{mean}\{f_2\} = 0 \\ \text{var}\{f_1\} = \text{var}\{f_2\} = 1 \end{cases}$



NOTE: You can't drop f_1 or f_2 spread are large

2 Dimensional → 1 Dimensional

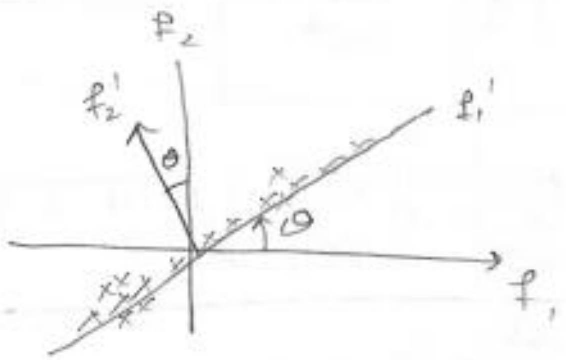
f' has lot of spread S.t f_1' has max spread.

① On the other hand $f_2' \perp f_1$ has less spread

Spread on f_2' << spread on f_1'

② drop f_2'

③ project x_i's on f_1' then 2D → 1D



Objective

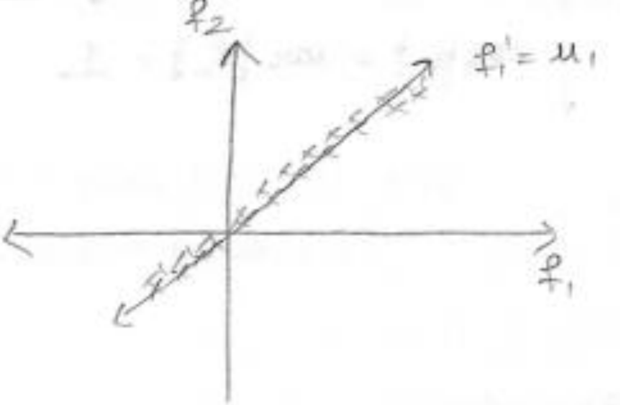
① We want to find a direction f_1' s.t the variance of x_i's projected onto f_1' is maximized.

① Rotating my axis to find f_1' with max-var.

② drop by f_2'

represent direction

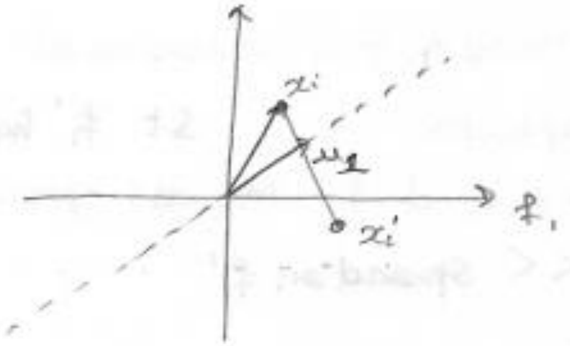
Mathematical objective function of PCA.



u_1 : unit vector

$\|u_1\| = 1$

Objective : To find direction by using u_1 where spread is maximum



$x_i' = proj_{u_1} x_i$

$$\text{Dataset} = \{x_i\}_{i=1}^{n}$$
$$\downarrow \qquad \downarrow$$
$$\text{Dataset}' = \{x_i'\}_{i=1}^{n}$$

$$\boxed{x_i' = \frac{u_1 \cdot x_i}{\|u\|^2}}$$ from linear algebra

$\Downarrow$

$$\boxed{x_i' = u_1^T x_i}$$ As $\boxed{u = 1}$

$$\boxed{\begin{array}{l} \text{If given} \\ \overline{x_1'} = u_1^T \bar{x} \\ \text{mean } \{x_i'\}_{i=1}^{n} \qquad \text{mean } \{x_i\}_{i=1}^{n} \end{array}}$$

Important

<u>Objective</u>: u_1 s.t. $\underset{(\text{Spread})}{\text{var.}} \{ \text{proj}_{u_1} x_i \}_{i=1}^{n}$ is maximum

$$\text{var}\{u_1^T x_i\}_{i=1}^{n} = \frac{1}{n} \sum (\underbrace{u_1^T x_i}_{x_i'} - \underbrace{u_1^T \bar{x}}_{\text{mean}\{x_i\}_{i=1}^{n}})^2$$

Remember
$$= \underbrace{(u_1^T)_{1\times n} \; x_{i\,n\times 1}}_{1\text{ scaler value}}$$

X: Column standardized {mean = 0}.

$$\bar{x} = [0\ 0\ 0\ 0 \ldots 0]$$

$$\text{var}\{x_i'\}_{i=1}^{n} = \frac{1}{n} \sum_{i=1}^{n} (u_1^T x_i)^2$$

variance $\{x_i'\}$

NOTE

Objective of optimization problem

$$\boxed{\max_{u_1} \frac{1}{n} \sum_{i=1}^{n} (u_1^T x_i)^2}$$

→ data matrix (given)

Find u_1 to be maximized

s.t $u_1^T u = 1 = \|u\|^2$

→ constraint u_1 is unit vector as we want to represent direction

→ Eigenvalues & Eigen vectors

Objective → As we have very large dataset (for eg Image of something, ~~wh~~ weather forcast)

↓

we extract lot of data from it in numerical values. (lot of numerical matrices are 0 or sparse)

To reduce the dimensionality we predict Eigen values and Eigen vector

$$\boxed{A\vec{x} = \lambda \vec{x}}$$

eigenvalue

Eigen vector

Singular Value Decomposition

diagnol

$$A = U \Sigma V^T$$

orthogonal

$$AA^T = U \Sigma V^T V \Sigma'^T U^T$$

Solution to our optimization problems: λ, V_1

$$X = \begin{array}{c|c} & 1\ 2\ 3 \ldots d \\ \begin{array}{c}1\\2\\\vdots\\n\end{array} & \begin{bmatrix} & & \\ & & \\ & & \\ & & \end{bmatrix}_{n\times d} \end{array}$$

cov. std

$d \rightarrow$ features
$n \rightarrow$ data points

Covariance matrix of $X = S$

$$S_{d\times d} = X^T_{d\times n} X_{n\times d}$$

$\hookrightarrow$ symmetric matrix

NOTE:
$\boxed{\lambda_n \rightarrow \text{here represents features}}$

Eigen values & Eigen vectors

$\downarrow$ $(\lambda_1, \lambda_2, \lambda_3, .. \lambda_d)$ $\quad$ $\downarrow$ $(v_1, v_2, \ldots V_d)$

$S_{d\times d} \rightarrow$

eigen values of $S = \lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_d$

eigen vector of $S = V_1, V_2, V_3, \ldots, V_d$

$\lambda_1 \geq \lambda_2 \geq \lambda_3 \ldots \geq \lambda_d$

maximal eigen value

def : $\underbrace{\lambda_1}_{\text{scalar}} \underbrace{V_1}_{d\times 1 \text{ vector}} = S_{d\times d} \overbrace{V_1}^{d\times 1 \text{ vector}}$

λ_1 : eigen value of S
V_1 : eigen vec. to S corr. to λ_1

Property $\Rightarrow \boxed{V_i \perp V_j}$

$V_i^T V_j = 0$

$V_i \cdot V_j = 0$

$u_1 = V_1$ = eigen-vector of $S (= X^T X)$ corresponding to largest eigen-value $(= \lambda_1)$

max-variance direction

Eigen values & vectors

Steps   streching the value of X   not changing

Given $X = \begin{bmatrix} \quad \end{bmatrix}$

2.) Column Standardization done $\mu = 0, \sigma = 1$

3.) $S_{d\times d} = X^T X$.

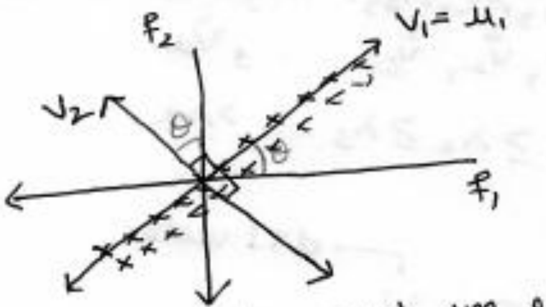
4.) Compute ~~values eigent~~ eigenvalues & vectors

$\lambda_1 \geq \lambda_2 \geq - - - - , \lambda_d$

$v_1 , v_2 , \cdots , v_d$

5.) $u_1 = v_1$ (why)

→ Geometric Interpretation of λ_i & V_i



2-dimensional
$d = 2$
$\lambda_1 \geq \lambda_2$
$v_1 \perp v_2$

We just take the data and we rotated the axis with θ such that where variance is maximum.

Example: If we have 10 dimensional data

$d = 10$

$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \cdots \cdots \geq \lambda_{10}$

$V_1 , v_2 , \cdots \cdots , V_{10}$

V_1 ↓ direction with max variance

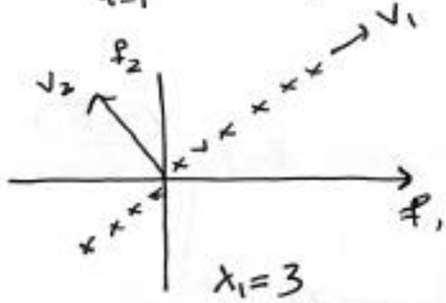
v_2 ↓ 2nd most variance

V_{10} ↓ least variance

λ_i: significance $\quad \lambda_1 \geq \lambda_2 \geq \lambda_3 \ldots \ldots \geq \lambda_d$

$\sum_{i=1}^{d} \lambda_i$

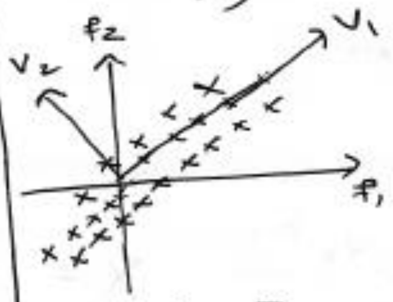
1)



$\lambda_1 = 3$
$\lambda_2 = 0$

$\frac{\lambda_1}{\lambda_1+\lambda_2} = \frac{3}{3} = 1$

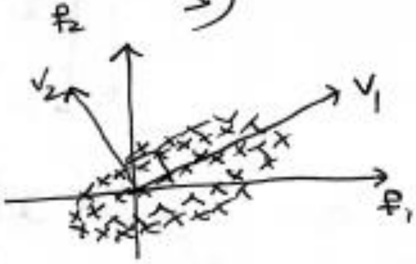
2)



$\lambda_1 = 3$
$\lambda_2 = 1$

$\frac{\lambda_1}{\lambda_1+\lambda_2} = \frac{3}{4}$

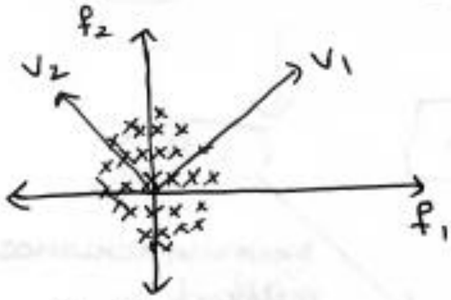
3)



$\lambda_1 = 3$
$\lambda_2 = 2$

$\frac{\lambda_1}{\lambda_1+\lambda_2} = \frac{3}{5}$

4)



$\lambda_1 = 3$
$\lambda_2 = 3$

$\frac{\lambda_1}{\lambda_1+\lambda_2} = \frac{3}{6} = \frac{1}{2}$

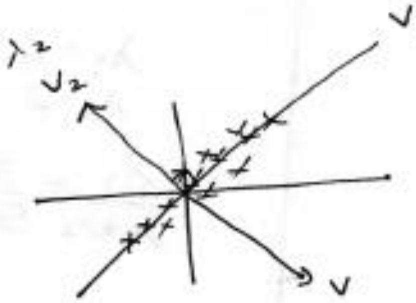
NOTE

$\frac{\lambda_1}{\lambda_1+\lambda_2}$: %age of variance or information preserved
project all data to v_1 → we are going to preserve 100% of information for 1 75% of information for 2 60% ———— 3 50% ———— 4
$\frac{\lambda_i}{\sum \lambda_i}$ what % age of variance
v_1 tells direction where variance is maximum

PCA for dimensionality Reduction and Normalization

$$X' = \begin{array}{c} \\ 1\\ 2\\ \vdots\\ n\end{array}\begin{array}{c} V_1 \\ \begin{bmatrix} \\ x_i \\ \\ \\ \end{bmatrix}_{1\times n} \end{array}$$

$$x_i' = x_i^T V_1$$



$$X = \begin{array}{c} \\ 1\\ 2\\ \vdots\\ n\end{array}\begin{array}{c} f_1 \quad f_2 \\ \begin{bmatrix} \\ \longleftarrow x_i^T \longrightarrow \\ \\ \\ \end{bmatrix} \end{array} \; \text{2D}$$

(2D)

↓

(1-D)

maximum variance method

$$\boxed{S = X^T X}$$

$$X' = \begin{array}{c} \\ 1\\ 2\\ \vdots\\ n\end{array}\begin{array}{c} V_1 \\ \begin{bmatrix} \vdots \\ x_i' \\ \vdots \\ \end{bmatrix} \end{array} \; \text{(1D)}$$

$$\boxed{x_i' = x_i^T V_1}$$

→ V_1 choose as it has maximum variance.

$$X = \begin{array}{c} \\ 1\\ 2\\ \vdots\\ n\end{array}\begin{array}{c} f_1 \quad f_2 \quad \dots \quad , f_{10} \\ \begin{bmatrix} \\ \longleftarrow x_i^T \longrightarrow \\ \\ \\ \end{bmatrix}_{n\times 10} \end{array} \xrightarrow[\text{reduct (PCA)}]{\text{dim}} X' = \begin{array}{c} \\ 1\\ 2\\ \vdots\\ n\end{array}\begin{array}{c} V_1 \; f_1 \; V_2 \\ \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix}_{n\times 2} \end{array}$$

→ which can be visualized

$$S = X^T X$$

eigen value & eigen vector

$$\begin{array}{ccccccc} \lambda_1 & \geq & \lambda_2 & \geq & \lambda_3 & \dots & \geq \lambda_{10} \\ \downarrow & & \downarrow & & \downarrow & & \downarrow \\ v_1 & & v_2 & & v_3 & & v_{10} \end{array}$$

$$x_i' = [x^T v_1, \; x^T v_2]$$

V_1 V_2

Top 2 features

Suppose want to do 50 dimensions instead of 2

$$X = \begin{array}{c} f_1 \quad f_2 \quad \cdots \quad f_{100} \\ \left[\begin{array}{ccc} & & \\ & & \\ & & \end{array} \right]_{n\times 100} \end{array} \xrightarrow[\text{convert 50 values}]{PCA} X' = \begin{array}{c} V_1 \, V_2 \cdots V_{50} \\ \left[\begin{array}{ccc} & & \\ & \boxed{} & \\ & & \end{array} \right]_{n\times} \end{array}$$

x_{ij}

$\lambda_1 \geq \lambda_2 \geq \lambda_3 \cdots \geq \lambda_{50}$

$V_1 \quad V_2 \quad V_3 \cdots V_{50}$

$x_i \in \mathbb{R}^{100}$; $x_i' \in \mathbb{R}^{d'}$

$d' < 100$

preserve 99% of the variance

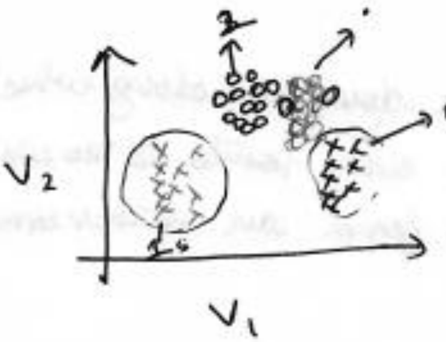
let $\dfrac{\lambda_1 + \lambda_2 + \cdots + \lambda_{51}}{\sum_{i=1}^{100} \lambda_i} = 0.99$

99% variance preserved

$\boxed{d' = 51}$

☆ <u>Visualize MNIST dataset</u>

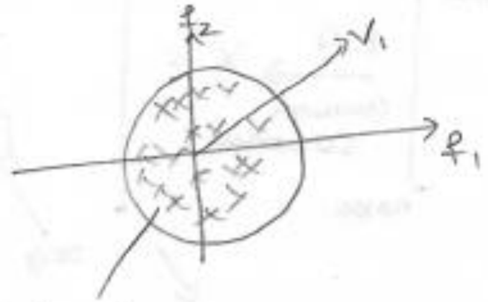
$\text{MNIST} \longrightarrow 784 \xrightarrow[\text{dimension reduction}]{PCA} 2$ [top 2 eigen vectors V_1 & V_2]



All 0 and 1's grouped together
2's are not well seperated

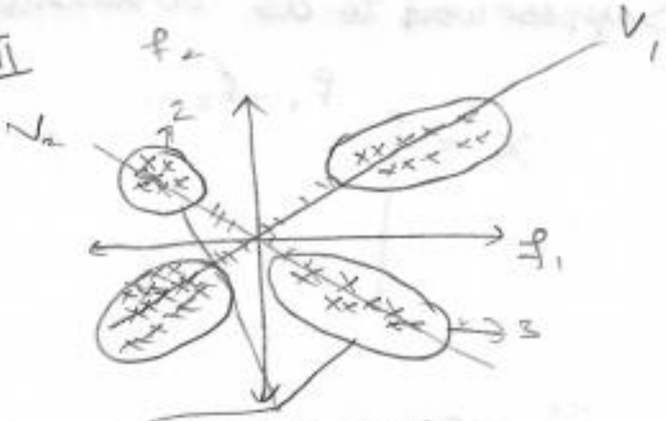
☆ Limitations of PCA

Case I



$\lambda_1 \approx \lambda_2$ (info-lost is very high)

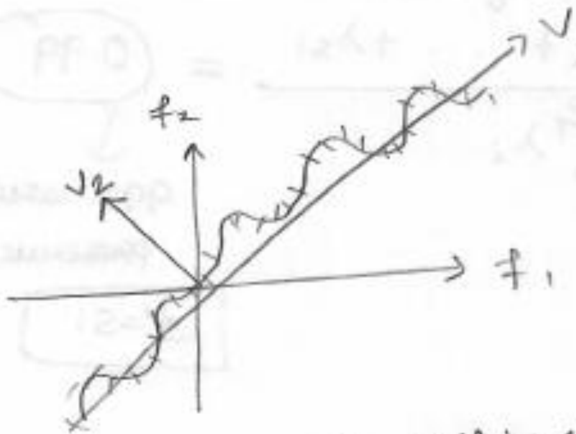
Case II



different clusters

These points being separate will project to same point

Case III



→ visualize in 1D all will be same

→ but in 2D dimensional sin wave can be easily distinguished

Goal: PCA is to find projection directions along which the variance of the projected data points is maximum, and projection vectors must form an orthonormal basis

PCA: constrained optimization:

$$u^* \leftarrow \left\{ \begin{array}{l} \max_u \; u^T S u \\ \text{s.t. } u^T u = 1 \end{array} \right\} \rightarrow \mathcal{L}(u,\lambda) = u^T S u - \lambda(u^T u - 1)$$

$= u_1 = \lambda_1$

largest
maximum eigenval

largest eigval

$$\frac{\partial \mathcal{L}}{\partial u} = 0 \Rightarrow Su - \lambda u = 0$$

u is eig vec of S

λ is eig val of S

$Su = \lambda u$

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \lambda_4 \geq \dots$$
$$u_1 \quad u_2 \quad u_3 \quad u_4, \dots \quad u_i\text{'s}$$

eig vec.

$u_i^T u_i = 1$ | As unit vector

$$\boxed{u^* = \max_u \; u^T S u} \quad \text{s.t. } u^T u = 1$$

$$u_1 = \arg\max_u \; \lambda \xrightarrow{\text{from}}$$

$u^T S u$

$\Rightarrow$ We know $Su = \lambda u$

$u^T(\lambda u)$

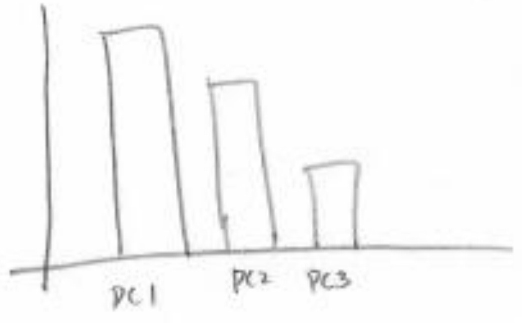
$\downarrow$

scaler

$\rightarrow \lambda (u^T u)$

$\rightarrow \lambda * 1 = \lambda$.

☆ Plot for PCA visualization is called scree plot



Practical Tips for PCA ÷

1.) Make sure that data is ~~serialized~~ normalized.

2.) Make sure data is centered that is mean = 0

3.) How many principal component you expect to find?

4.) In summary, technically there is a PC for each variable in the data set. However, if there are fewer sample than variables, then the number of samples put an upper bound on the number. PCs with eigenvalues greater than 0.

(t-SNE) T-distributed Stochastic Neighbourhood Embedding

→ State of the best dimensional reduction → visualization

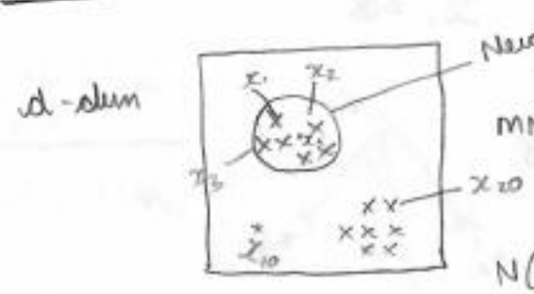
→ PCA → basic, old } → 2 dim visualization not very good

PCA: tries to preserve global structure of data.

t-SNE: preserves local-structure of data

Neighbourhood; Embedding

1) Neighbourhood



MNIST: 784-dim

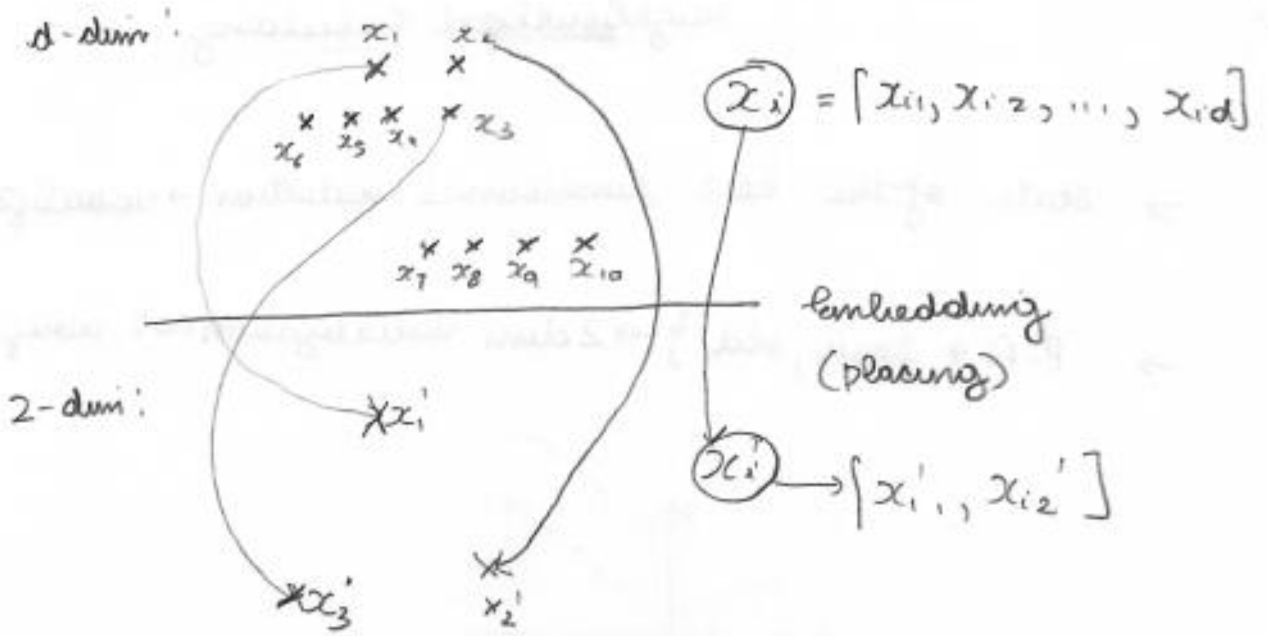
$N(x_i) = \{ x_j \mid x_i \,\&\, x_j$ are geometrically close $\}$

$\|x_i - x_j\|^2 = \text{dist}$

$N(x_1) = \{ x_2, x_3, x_4 \}$

does Not contain x_{10} & x_{20}

Embedding

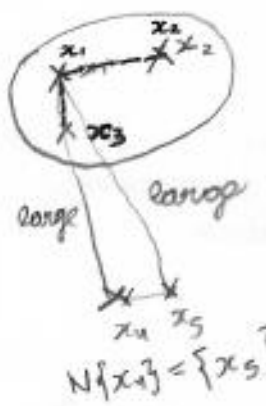


☆ Every point in high dimensional space have corresponding in low dimension space such a thing is called embedding.

Geometric Intuition of t-SNE

d-dimension

$N(x_1) = \{x_2, x_3\}$

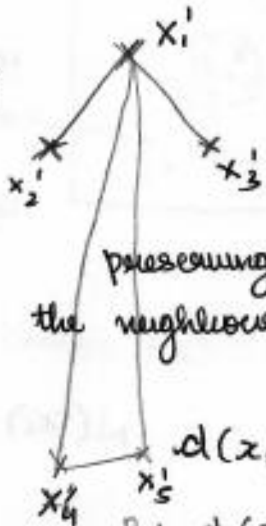


$N\{x_4\} = \{x_5\}$

⇨ Embedding

2-dimension

$x_i' \in \mathbb{R}^2$



$d(x_1, x_2) \approx d(x_1', x_2')$

$d(x_1, x_3) \approx d(x_1', x_3')$

preserving distances of point in the neighbourhood

$d(x_1, x_4) \not\approx d(x_1', x_4')$

But $d(x_4, x_5) \approx d(x_4', x_5')$

distance between the neighbourhood are preserved i.e

$d(x_1, x_2) \approx d(x_1', x_2')$
$d(x_1, x_3) \simeq d(x_1', x_3')$
$d(x_4, x_5) \approx d(x_4', x_5')$

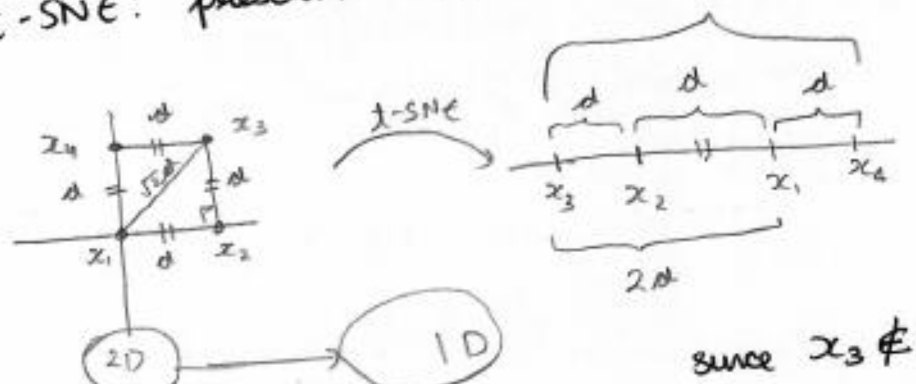
On the other hand

$d(x_1, x_4) \not\approx d(x_1', x_4')$
$d(x_1, x_5) \not\approx d(x_1', x_5')$

Crowding Problem

t-SNE: preserve dist in a N $\quad N(x_3) = \{x_4, x_2\}$

2D Square
3D Cube
nD hyper cube



since $x_3 \notin N_d(x_1)$

some times it is impossible to preserve distance in all the N.

such problem is called crowding problem

It can be resolved using

t-distribution → Stochastic neighbourhood embedding

Students t-distribution

distill.pub → Go to the web link.

How to Use t-SNE Effectively

Matrix Factorization : PCA & SVD

* Matrix Factorization : $\underset{\text{matrix}}{A} = \underbrace{BCD}_{\text{product of other matrices}}$

$\boxed{A = BC}$ → multiplicative decomposition

$6 \to 2 \times 3 \to (2, 3)$ factors of 6

Similarly → we say B, C, D are factors of A.

MF :→ PCA (Principal component Analysis)
↳ Dimensionality reduction (high dim → low dim.)

PCA :→ $X_{n \times d}$ - Data Matrix (Centered)

$$S_{d \times d} = \frac{X^T X}{n-1} \Rightarrow \text{covariance matrix}$$

↳ eigen values :- $\lambda_1, \lambda_2, \ldots, \lambda_d$
eigen vectors :- $w_1, w_2, \ldots, w_d$

$d \to d' \to$ basically take top $\underset{\text{eigen vals}}{\lambda\text{'s}} \to \underset{\text{eigen vectors}}{w_i\text{'s}}$

$$\underset{\text{covariance matrix}}{S_{d \times d}} = W_{d \times d} \; \lambda_{d \times d} \; W^T_{d \times d}$$

W^T = Transpose of W.

$$W = \begin{bmatrix} \uparrow & \uparrow & \overset{\text{eigen vector of } S}{\uparrow} & & \uparrow \\ w_1 & w_2 & w_3 & \cdots & w_d \\ \downarrow & \downarrow & \downarrow & & \downarrow \end{bmatrix}_{d \times d}$$

$$\lambda_{d \times d} = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & & \lambda_3 \cdots & \lambda_d \\ 0 & & & \end{bmatrix}_{d \times d}$$

These are eigenvalues of S.

This whole decomposition is called eigen decomposition of S which is purely matrix factorization

---

☆ Singular value decomposition (~~Singular~~ VD)

It is related to PCA

PCA :- performed on co-variance matrix (S) → square, symm.

SVD :- performed on rectangular matrix.

$X_{n \times d}$

$$X_{n \times d} = U_{n \times n} \Sigma_{n \times d} V^T_{d \times d} \longrightarrow \text{SVD}$$

for PCA $\quad Cov(\cancel{X}) = S = W \lambda W^T \longrightarrow \text{PCA}$

$$\Sigma = \begin{bmatrix} s_1 & & & 0 \\ & s_2 & & \\ & & s_3 \cdots & \\ 0 & & & s_d \end{bmatrix}_{n \times d} \rightarrow \text{diagnol matrix}$$

Singular values of x

$$\boxed{\frac{s_i^2}{n-1} = \lambda_i}$$

Relationship between eig-val of ~~s~~ and singular value.

$$X_{n \times d} = \underbrace{U_{n \times n}}_{\text{left singular vector of } X} \Sigma_{n \times d} \underbrace{V^T_{d \times d}}_{\text{Right singular vectors of } X}$$

$\boxed{U_i :- \; i^{th} \text{ column of } U = i^{th} \text{ eig. vector of } (X_{n \times d} X^T_{d \times n})}$ not S

$\boxed{V_i :- \; i^{th} \text{ column of } V = \text{eig}^{i^{th}}\text{-vect of } (X^T X) = S}$

$$\text{cov}(x) \quad S = W \lambda W^T \longrightarrow PCA$$

W: eig vector of S (w_i)

λ: eig values of S (w_i)

$$X = U \Sigma V^T \longrightarrow SVD$$

U: eig vec of (XX^T)

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

Σ: singular values

s_i ; $\frac{s_i^2}{n-1} = \lambda_i$

dt //

SVD → ~~At~~ Stanford University Video

$$A_{[m \times n]} = U_{[m \times r]} \, \Sigma_{[r \times r]} \left(V_{[n \times r]}\right)^T$$

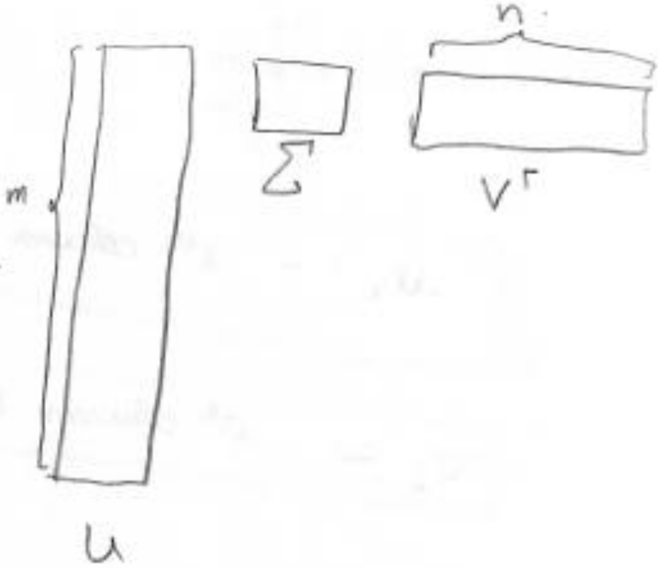
- A: Input data matrix

eg. Users to Movies

$$\begin{array}{c|ccccc}
 & \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \\
\hline
u_1 & 1 & 1 & 1 & 0 & 0 \\
u_2 & 3 & 3 & 3 & 0 & 0 \\
u_3 & 4 & 4 & 4 & 0 & 0 \\
u_4 & 5 & 5 & 5 & 0 & 0 \\
u_5 & 0 & 2 & 0 & 4 & 4 \\
u_6 & 0 & 0 & 0 & 5 & 5 \\
u_7 & 0 & 1 & 0 & 2 & 2
\end{array}$$

people

=



movies and User both break into 2 groups.

If we apply SVD to the matrix

Sci-fi concept, Romance concept

$$\begin{array}{c} u_1\\u_2\\u_3\\u_4\\u_5\\u_6\\u_7\end{array} = \underbrace{\begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix}}_{U} \times \underbrace{\begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix}}_{\Sigma}$$

Strength of Sci-fi

Strength of Romance concept

low strength ignore it

$$\times \; V^T \quad \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

Sci fi concept

Romance concept

movies to concept matrix

Note.

From above 'movies', 'users' & 'concepts'

U : user to concept similarity matrix

V : movies to concept similarity matrix

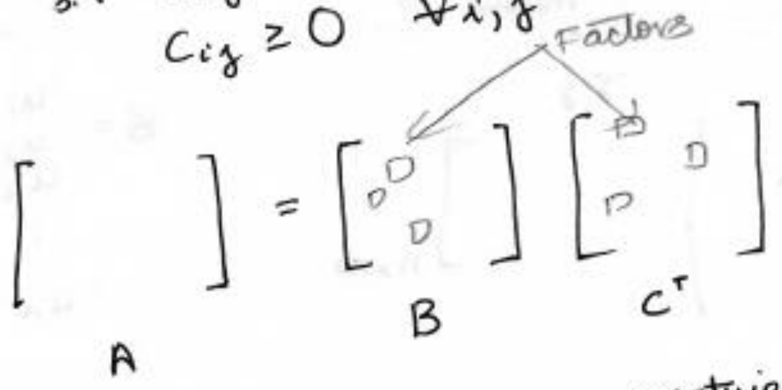
Σ : its diagonal element : 'strength of each concept'

→ Non-negative Matrix Factorization (NMF)

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

$B: n\times d$
$C: m\times d$

$$\text{s.t. } B_{ij} \geq 0 \quad \forall i,j$$
$$C_{ij} \geq 0 \quad \forall i,j$$



Factors here are non negative

MF for collaborative Filtering

$$A = \begin{array}{cc} & I_j \\ u_i & \begin{bmatrix} & A_{ij} & \end{bmatrix} \end{array}$$

A_{ij} - rating on I_j by U_i :
Sparse matrix : many empty cells.

$$A_{n \times m} = B * C^T$$

users — items

B ; $n \times d$

$d \geq 0$

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

$d \leq m$, $d \leq n$

C^T ; $d \times m$

A can be decomposed as a product of 2 matrix B & C.

Q lot of empty values in matrix

$$A_{ij} = B_i \times C_j$$

j^{th} row of matrix C.

i^{th} row of matrix B

$$A = \begin{array}{cc} & I_j \\ u_i & \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}_{n \times m} \end{array}$$

user

$$B = \begin{array}{c} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_n \end{array} \begin{bmatrix} \\ \\ \\ \\ \\ \end{bmatrix}_{n \times d}$$

$$C = \begin{array}{c} I_1 \\ I_2 \\ \vdots \\ I_m \end{array} \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix}_{m \times d}$$

item

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

(many empty cells)

$$\boxed{A = BC^T}$$

optimization problem

SGD

Find B & C $\Rightarrow$ MF

$$\underset{B,C}{\operatorname{argmin}} \sum_{\substack{\{i,j \\ \text{s.t.} \\ A_{ij} \text{ is not} \\ \text{empty}\}}} \underbrace{\left(A_{ij} - B_i^T C_j\right)^2}_{\text{minimizing squared loss.}}$$

$\to$ can't use empty $A_{i,j}$

Full matrices

$$B^T = \begin{bmatrix} \uparrow & \uparrow & \uparrow & & \uparrow \\ B_1 & B_2 & B_3 & \cdots & B_n \\ \downarrow & \downarrow & \downarrow & & \downarrow \end{bmatrix}_{d \times n}.$$

$$B = \begin{array}{c} B_1 \\ B_2 \\ \vdots \\ B_n \end{array} \begin{bmatrix} \leftarrow B_i \rightarrow \\ \\ \\ \\ \end{bmatrix}_{n \times d} \qquad C = \begin{array}{c} C_1 \\ C_2 \\ \vdots \\ C_m \end{array} \begin{bmatrix} \leftarrow C_j \rightarrow \\ \\ \\ \\ \end{bmatrix}_{m \times d}$$

n = number of users

m = number of items

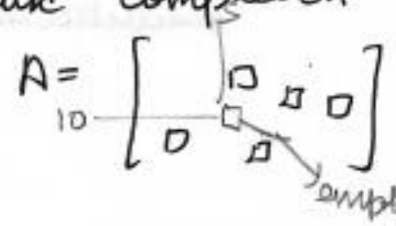
① solve the optimization problem using SGD

$$\underset{\substack{B,C \\ \text{st } A_{ij} \neq 0}}{\operatorname{argmin}} \sum_{\substack{i,j \\ \text{valid}}} \left(A_{ij} - B_i^T C_j\right)^2$$

nonempty A_{ij}'s to find B, C

② $B = \begin{bmatrix} \longleftarrow B_i \longrightarrow \end{bmatrix}$; $C = \begin{bmatrix} \leftarrow C_j \rightarrow \end{bmatrix}$

③ matrix completion



$(A_{10,5})$ = empty

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

Fill my empty cells in A using $B_{10}^T * C_5$.

↓

it is ~~to~~ not using $A_{10,5}$
as it was zero it is prediction
based on some other user
whose output $\neq 0$

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

no empty cells — $\hat{A} = B * C^T$

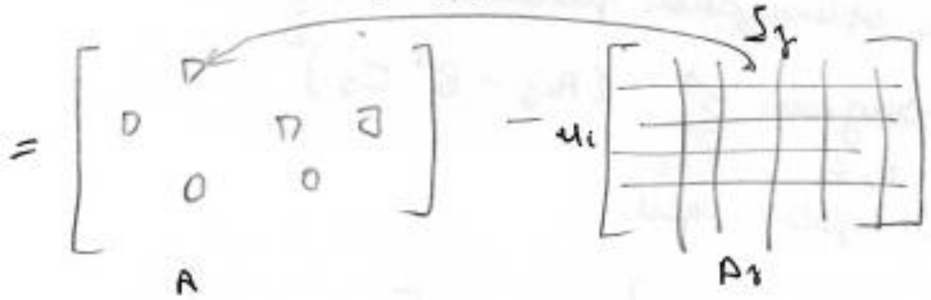
$\longrightarrow$ computed using A_{ij} that are not empty by solving optimization problem

$[\quad]$

$$A - \hat{A} \longrightarrow \text{v. small}$$

↓

it is fully filled / completed form of A using matrix factorization



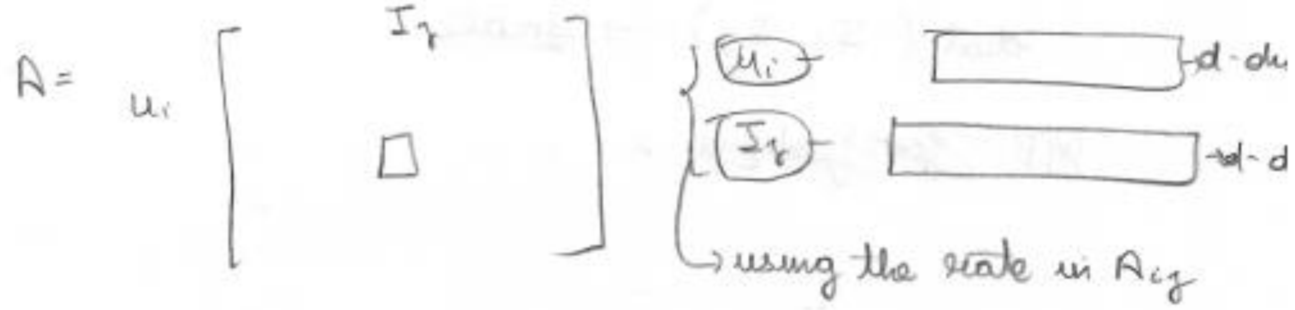
↓

u_i & $I_j \rightarrow$ get a rating

↓

predicted rating

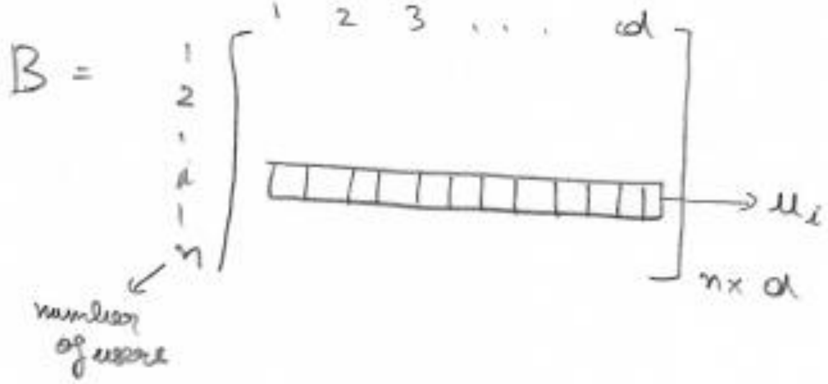
Matrix Factorization for feature engineering

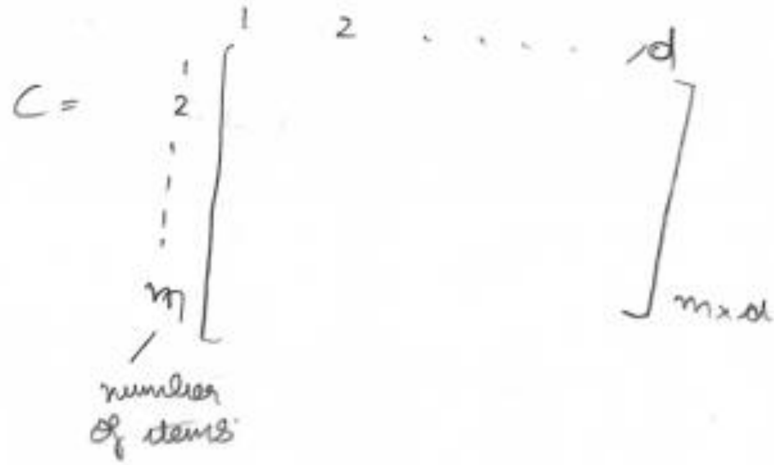


$$A_{n\times m} = B\,C^T$$

$A_{n\times m}$: users × items; B: $n\times d$; C^T: $d\times m$

$d \geq 0$; $d \leq \min(m,n)$





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

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

$I_i, I_j \rightarrow$ V similar based on rating data

dist $(I_i, I_j) \rightarrow$ small

MF for features