Dimensionality Reduction And Visualization

lec1 Dimensionality Reduction

2-D, 3-D: Scatter plot.
4-D, 5-D, 6-D: Pair plot.
10-D, 100-D, 1000D — We reduce dimensions.
PCA & t-SNE

lec2 Row vector & column Vector

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

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

$$x_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \\ \vdots \\ \vdots \\ x_{id} \end{bmatrix}$$ — column vector

$\mathbb{R}$ = real space or real number

If you are not told about ~~ex~~ vector default is Column Vector.

$$x_i = \begin{array}{cccc} SL & PL & SW & PW \\ \downarrow & \downarrow & \downarrow & \downarrow \end{array}$$
$$x_i = \underbrace{[2.1,\ 3.2,\ 4.6,\ 1.2]}_{1\times 4} : \text{row vector}$$

lec 3 Dataset

$$D = \{x_i, y_i\}_{i=1}^{n}$$ — no. of datapoints

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

$y_i \in \{\text{Setosa, Versicolor, Virginica}\}$

For Iris dataset $x_i \in \mathbb{R}^4$

$$x_i = \begin{bmatrix} SL \\ SW \\ PL \\ PW \end{bmatrix} \qquad y_i = \{\text{Setosa, Versicolor, Virginica}\}$$

lec 4 Dataset as a data-matrix

$$X = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \vdots \\ n \end{array} \underbrace{\begin{bmatrix} f_1 & f_2 & f_3 & \cdots & f_d \\ & & & & \\ & & & & \\ \longleftarrow & & x_i^T & & \longrightarrow \\ & & & & \\ & & & & \end{bmatrix}}_{n\times d} \qquad Y = \underbrace{\begin{bmatrix} \\ \\ y_i \\ \\ \\ \end{bmatrix}}_{n\times 1} \begin{array}{c} 1 \\ 2 \\ \vdots \\ i \\ \vdots \\ n \end{array}$$

$x_i \in \mathbb{R}^d$ — features

$y_i \in \{S, Vi, Ve\}$

each datapoint : row; each column = feature

DATE ____________

PAGE ____________

x_i = column vector

x_i^T = row vector

Also,

$$
\begin{array}{c|cccccc|}
 & 1 & 2 & 3 & 4 \cdots & i \cdots & n \\
\hline
X = f_1 & & & & & \uparrow & \\
f_2 & & & & & & \\
f_3 & & & & & & \\
\vdots & & & & & x_i & \\
f_j & & & & & & \\
\vdots & & & & & & \\
f_d & & & & & \downarrow & \\
\hline
\end{array}_{d \times n}
$$

column : data point

row: feature / variable

f_1 = PL

f_2 = PW

f_3 = SL

f_4 = SW

DATE ________
PAGE ________

lec 5 Data preprocessing : Feature normalization

column normalization

$$X = \begin{array}{c} \\ x_1 \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{array} \begin{array}{c} f_1 \quad f_2 \quad f_3 \cdots \quad f_j \quad \cdots \; f_d \\ \left[\begin{array}{c} \\ \\ \\ \longleftarrow x_i^T \longrightarrow \\ \\ \\ \end{array} \right]_{n \times d} \end{array} \quad \begin{array}{c} Y \\ \left[\begin{array}{c} \\ \\ \\ y_i \\ \\ \\ \end{array} \right]_{n \times 1} \end{array}$$

obtain data → Pre processing → data modelling
column normalization (dimensionality reduction)

$$X = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \vdots \\ i \\ \vdots \\ n \end{array} \begin{array}{c} f_1 \quad f_2 \quad f_3 \cdots \overset{x}{f_j} \cdots f_d \\ \left[\begin{array}{c} a_1 \\ a_2 \\ a_3 \\ \vdots \\ \longleftarrow x_i^T \text{———} a_i \longrightarrow \\ \vdots \\ a_n \end{array} \right]_{n \times d} \end{array} \quad \begin{array}{c} Y \\ \left[\begin{array}{c} \\ \\ \\ y_i \\ \\ \\ \\ \end{array} \right]_{n \times 1} \end{array}$$

column normalization

$a_1, a_2, \ldots a_n \rightarrow$ n values of f_j

$\max(a_i) = \cancel{\text{maximum}}\ a_{max} \geq a_i \ (i: 1 \rightarrow n)$

$\min(a_i) = a_{min} \leq a_i \ (i: 1 \rightarrow n)$

$a'_1, a'_2, a'_3, a'_4 \ldots a'_i \ldots a'_n$

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

why do column normalization

eg →

	h	W	
Student	162	56	1
	172	72	2
	182	84	⋮
	150	58	⋮
	⋮	⋮	n
	cm	kg	

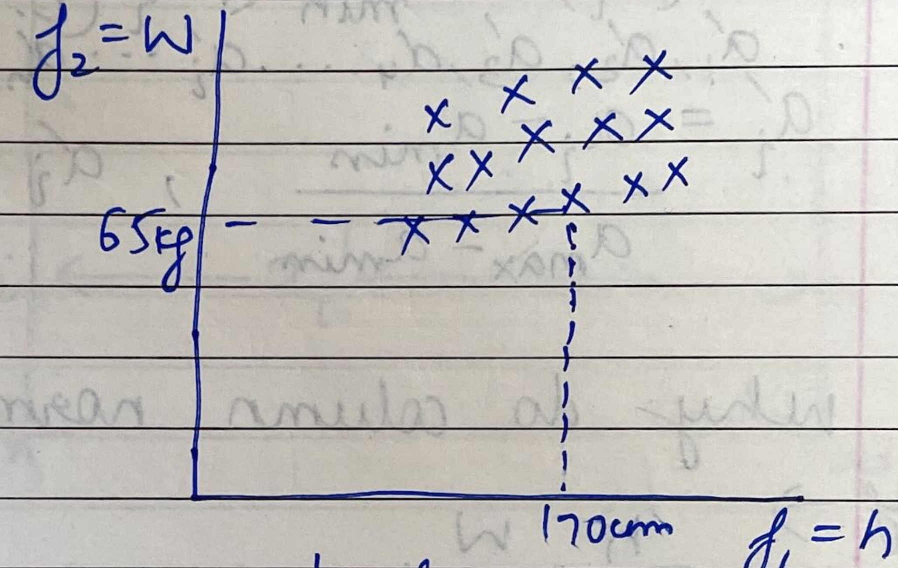
col-normalization →

	$f_1 = h'$	$f_2 = w'$
1		
2		
3		
⋮		
	[0,1]	[0,1]

To make all value lie in [0,1] and not differing in kg/pound.

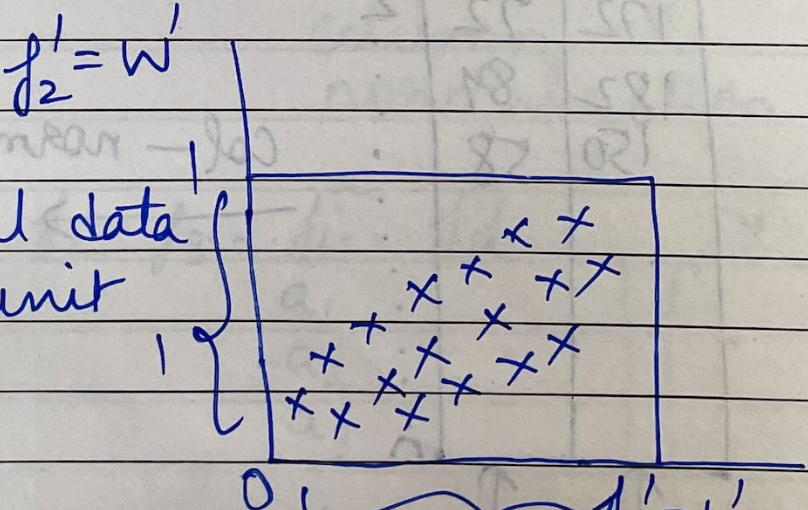
To get rid of scale and put all feature in same scale.

Geometrically,



↓ column normalization

To bring all data points in 1 unit square.



DATE
PAGE

anywhere in n-dim space $\xrightarrow[\text{norm.}]{\text{col.}}$ unit hyper cube

lec6 Mean of a data matrix

Mean Vector

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

$x_1 = [2.2 \ , \ 4.2] \in \mathbb{R}^2$
$x_2 = [1.2 \ , \ 3.2] \in \mathbb{R}^2$

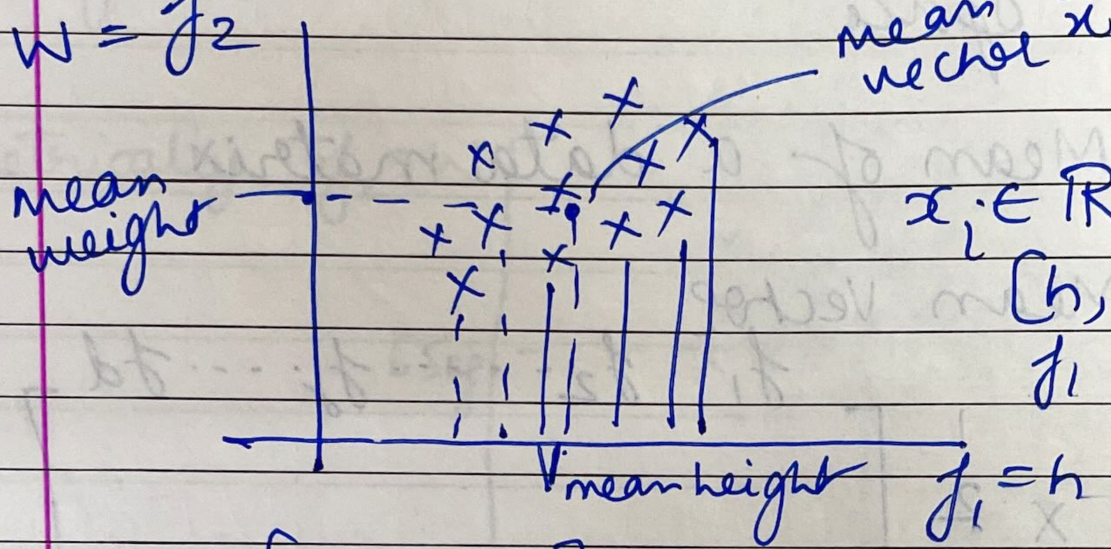
$x_1 + x_2 = [3.4 \ , \ 7.4]$

$\bar{x} \in \mathbb{R}^d$

$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i = \frac{1}{n}(x_1 + x_2 + \cdots x_n)$ $(x_i \in \mathbb{R}^d)$

Mean Vector

Geometrically :-



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

$[h, W]$

$\;f_1 \quad f_2$

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

$$h_{\bar{x}} = \text{mean}(h_i)_{i=1}^{n}$$

$$W_{\bar{x}} = \text{mean}(W_i)_{i=1}^{n}$$

mean vector = central vector

DATE
PAGE

lec7 Data Preprocessing - Column Standard-izations

Column normalization was: $[0, 1]$

↑ get rid of scales of each feature

Coln-std → more often used in practice.

$$X = \begin{array}{c|cccccc} & f_1 & f_2 & \cdots & f_j & \cdots & f_d \\ \hline 1 & & & & a_1 & & \\ 2 & & & & a_2 & & \\ \vdots & & & & \vdots & & \\ i & \longleftarrow & & x_i^T & a_i & & \longrightarrow \\ \vdots & & & & \vdots & & \\ n & & & & a_n & & \end{array}_{n \times d}$$

$\boxed{a_1, a_2, a_3 \ldots a_n}$ ← n values of f_j

↓

col std

↓

$\boxed{a_1', a_2' \, a_3' \ldots a_i' \ldots a_n'}$

$\text{mean}\{a_i'\}_{i=1}^{n} = 0$

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

DATE ________
PAGE ________

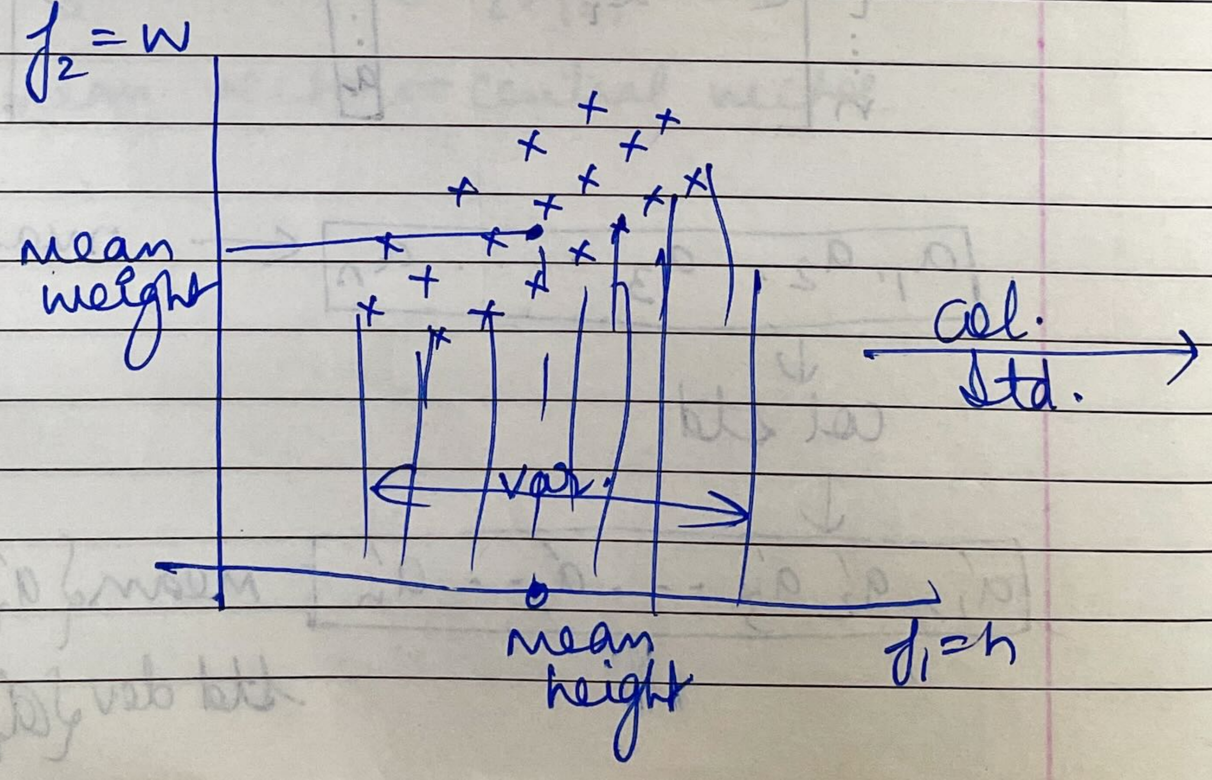
$\bar{a} = \text{mean}\{a_i\}_{i=1}^{n}$ ← Sample mean

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

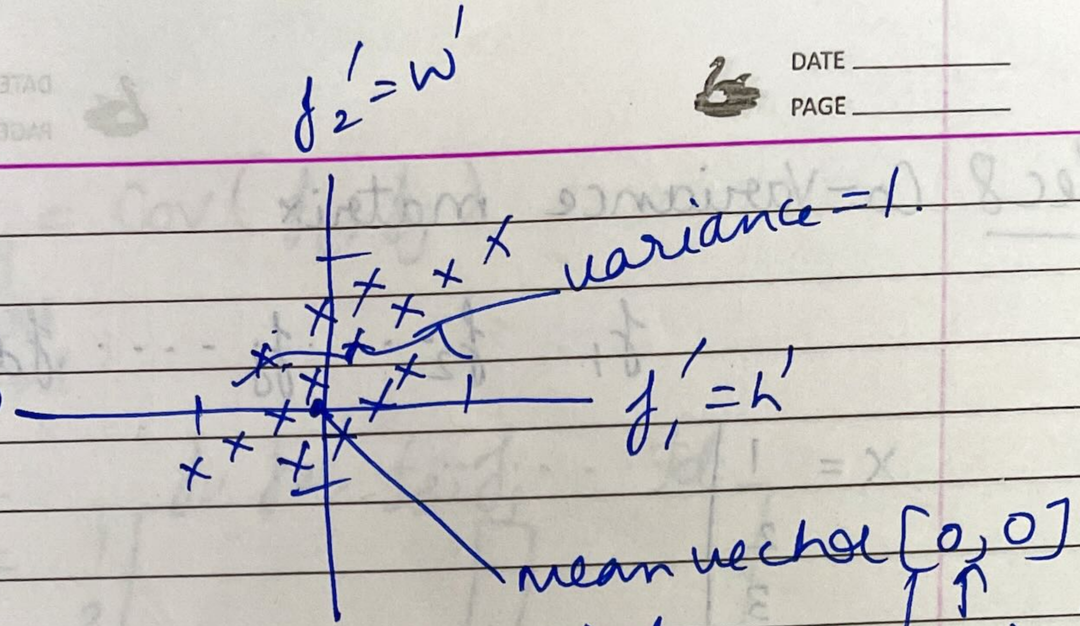
$$a_i' = \frac{a_i - \bar{a}}{s}$$

Similar to standard normal variate in P&S but that had Normal distribution but our ~~a~~ a_i can come from any distribution.

Geometrically,



after col std →



moved the points such that mean lies at origin

col std
1. Moving mean vector to origin
2. Expanding/squishing such that standard deviation for any feature is 1.

<u>Col. std</u>

mean-centering → origin

Scaling → std dev = 1 for all features

DATE ______
PAGE ______

lec8 Co-Variance matrix

$$X = \begin{array}{cc} & \begin{array}{ccccccc} f_1 & f_2 & \cdots & f_j & \cdots & f_d \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ \vdots \\ i \\ \vdots \\ n \end{array} & \left[\begin{array}{c} \\ \\ \\ \\ \longleftarrow x_i^T \longrightarrow \\ \\ \\ \end{array} \right]_{n\times d} \end{array}$$

$$\text{Co-Variance of } X \text{ is } S = \begin{array}{cc} & j \\ i & \left[\begin{array}{ccc} & & \\ & & \\ & S_{ij} & \\ & & \end{array} \right]_{d\times d} \end{array}$$

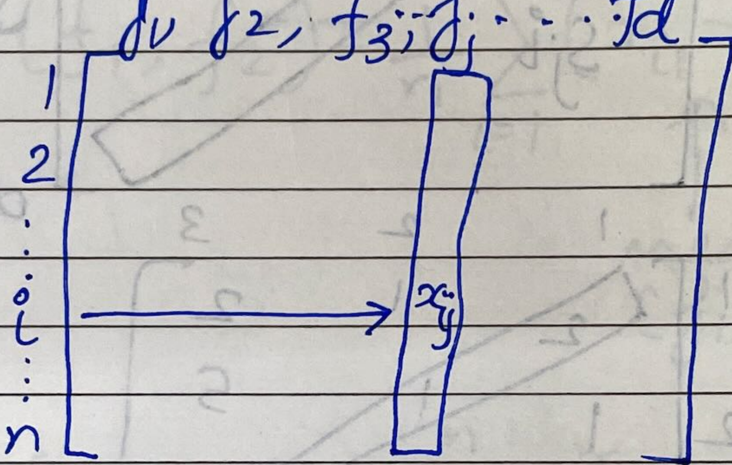
↑ Square matrix

S_{ij} = i^{th} row & j^{th} column element in S

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

$i : 1 \to d$
$j : 1 \to d$

$X =$



x_{ij} = j^{th} feature for the i^{th} datapoint

$$Cov(x, y) = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu_x)(y_i - \mu_y)$$

$Cov(f_i, f_i) = Var(f_i)$

$Cov(X, X) = Var(X)$ ——— ①

$Cov(f_i, f_j) = Cov(f_j, f_i)$ ——— ②

Square Symmetric matrix

variance of features

$$S = \begin{bmatrix} \boxed{\ddots} & S_{ij} \\ S_{ji} & \end{bmatrix}_{d \times d}$$

Symmetric matrix

Square matrix

$d \times d$

eg →

$$\begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & & & \\ 2 & & & \\ 3 & & & \end{array}
\quad
\begin{array}{cccc} & 1 & 2 & 3 \\ 1 & \multicolumn{3}{c}{\multirow{3}{*}{$\begin{bmatrix} 2 & 1 & 2 \\ 1 & 1 & 5 \\ 2 & 5 & 3 \end{bmatrix}$}} \\ 2 & & & \\ 3 & & & \end{array}$$

$A_{21} = A_{12}$

$A_{ij} = A_{ji} \ \forall \ i, j$ ∴ Symmetric matri

$$\begin{array}{c} \\ 1 \\ 2 \\ \vdots \\ n \end{array}\begin{array}{c} f_1 \quad f_2 \cdots f_i \quad f_j \cdots f_d \\ \left[\begin{array}{cccc} & x_{1i} & x_{1j} & \\ & x_{2i} & x_{2j} & \\ & \vdots & \vdots & \\ & & & \end{array}\right] \\ \quad\quad X \quad\quad Y \end{array}$$

DATE ______

PAGE ______

Let X col. standardized $\Rightarrow$ mean$\{f_i\} = 0$

std. dev$\{f_i\} = 1$

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

mean (f_1) mean (f_2)

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

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

If f_1 and f_2 have been standardized

$$Cov(f_1, f_2) = \frac{f_1^T f_2}{n}.$$

$x.$

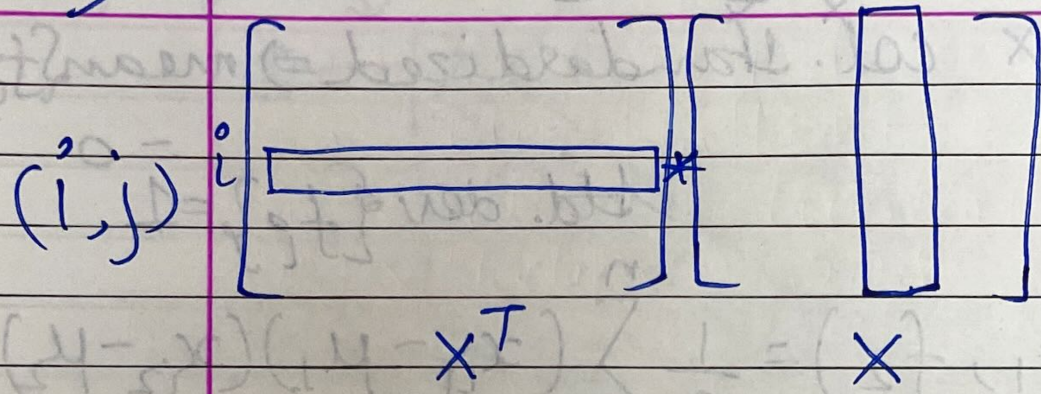
* $S_{d\times d} = \frac{1}{n} (X^T)_{d\times n} (X)_{n\times d} = (\ )_{d\times d}$

$\hookrightarrow$ data matrix

<u>L.H.S</u> Assuming X has been col. standard-ized

$$S_{ij} = Cov(f_i, f_j) = \frac{f_i^T f_j}{n}$$

RMS



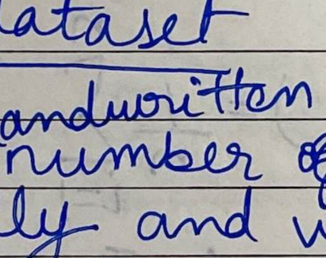
$S_{d\times d} = \frac{1}{n} X^T_{d\times n} \times X_{n\times d}$ if X has been column standardized.

lec 9 MNIST dataset

Images of handwritten number images. 28 pixel horizontally and vertically.
60 K – Training datapoints.
10 K – Test datapoints.

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

x_i:



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

DATE ____________

PAGE ____________

Obj – classify the written characters into one of the numeric characters.

We learnt $x_i \rightarrow$ column vector.
How to convert the 28x28 image to column vector.

We have converted 28x28 image to 28x28 numerical / real matrix.
wherever it is black there is 1
gray has some value <1 but >0
and white has 0. But this is not data matrix.

Let us assume we have a matrix 5x5.

	1	2	3	4	5
1	1	2	4	6	8
2	.5	2	1	8	2
3	2	1	6	8	1
4	3	2	1	8	2
5	4	2	6	8	12

5x5

Flattening $\rightarrow$

row flattening.

25X

$28 \times 28 = 784$

So after flattening of rows we get a column vector of 784×1 dimensions

So, whole MNIST dataset can be represented as :-

$$
X = \begin{array}{c|cccccc|}
 & f_1 & f_2 & f_3 & \cdots & \cdots & f_{784} \\
\hline
1 & & & & & & \\
2 & & & & & & \\
3 & & & & & & \\
\vdots & & & x_i^T \in \mathbb{R}^{784} & & & \\
\vdots & \longleftarrow & & 784\ dim & & \longrightarrow & \\
\vdots & & & & & & \\
60K & & & & & & \\
\end{array}_{n \times d}
\quad
\begin{bmatrix} y_i \in \\ 0,1,2, \\ 3,4,5, \\ 6,7,8, \\ 9 \end{bmatrix}_{n \times 1}
$$

$n = 60K$

$d = 784.$

lec 10 Code to Load MNIST Data Set

MNIST dataset downloaded from Kaggle.

Functions to read and show images.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

d0= pd.read_csv('train.csv')
print(d0.head(5))
# save labels into a variable l.
l = d0['label']

# Drop the label feature and store the pixel data in d.

d = d0.drop("label", axis=1)

print(d.shape)
print(l.shape)
```

DATE
PAGE

```
# display or plot a no.

plt.figure(figsize=(7,7))
idx = 180
# reshape from 1d to 2d pixel array
grid_data = d.iloc[idx].as_matrix()
            .reshape(28,28)

plt.imshow(grid_data, interpolati
                             on="no
           cmap='gray')

plt.show()
print(l[idx])
```