Dimensionality Reduction :-

2D, 3D $\longrightarrow$ Scatter plots can be used to visualized

4D, 5D, 6D, ... $\longrightarrow$ pair plots can be used but not enough.

What if there are more than 10 Dimensions?

10 D, 100 D, 1000 D $\longrightarrow$ can't be visualized easily

n-Dimension $\xrightarrow{\text{reduced}}$ 2D / 3D

techniques used :- ① Principal Component Analysis (old)

② t-Distributed Stochastic Neighbouring Analysis (modern state of the art)

In Iris dataset PL, PW, SL, SW $\longrightarrow$ dependent variables / Features } Data is 4 dimensional.

Species $\longrightarrow$ Target Variable / class

Row vector, Column Vector :-

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

Flower $\rightarrow$ datapoint

$\rightarrow$ i^{th} Datapoint : $x_i \in \mathbb{R}^D$ $\longrightarrow$ x_i is a D dimensional column vector

$\mathbb{R}$ $\rightarrow$ Real Space, Real Numbers

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

$\rightarrow$ Default is a column vector.

$\rightarrow$ $y_i = [y_{i1} \quad y_{i2} \quad \cdots \quad y_{in}]_{1\times n}$ $\longrightarrow$ row vector

$\rightarrow$ Generally assume it's a column vector.

Representing a dataset :-

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

$y_i = \{\text{setosa, virginica, versicolor} \ldots\}$ $\rightarrow$ in case of iris dataset

$x_i \in \mathbb{R}^D$, $x_i \in \mathbb{R}^4$

[$x_i \rightarrow$ data points $\quad y_i \rightarrow$ class labels]

$\rightarrow$ For a single value, X will be column vector & Y will be a single value. For the overall dataset, X will be matrix & Y will be a column vector.

Representing Dataset as a Matrix :-

let $D = \{x_i, y\}$

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

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

$\rightarrow$ d - features. By default it's a column vector

$$X = \begin{array}{c c} & \begin{array}{cccccc} f_1 & f_2 & f_3 & \cdots & f_d \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ \vdots \\ n \end{array} & \begin{bmatrix} \\ \\ \\ \longleftarrow & x_i^T & \longrightarrow \\ \\ \\ \end{bmatrix}_{n\times d} \end{array}$$

$\rightarrow$ column represents a feature

Row represents a datapoint

Transpose because

$$X = \begin{array}{c} \\ f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_j \\ \vdots \\ f_p \end{array} \begin{array}{c} 1 \quad 2 \quad 3 \quad 4 \quad \cdots j \cdots n \\ \left[\begin{array}{ccc} & \uparrow & \\ & x_i & \\ & \downarrow & \end{array} \right]_{d \times n} \end{array}$$

$\longrightarrow$ Alternative way - Both are valid

Data Preprocessing : Feature Normalization :-

$$X = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{array} \begin{array}{c} f_1 \quad f_2 \quad f_3 \cdots f_j \cdots f_d \\ \left[\begin{array}{c} \\ \\ \longleftarrow x_i^T \longrightarrow \\ \\ \end{array} \right]_{n \times d} \end{array} \qquad Y = \begin{array}{c} y \\ \left[\begin{array}{c} \\ y_i \\ \\ \end{array} \right] \end{array}$$

Preprocessing = set of math operations before building ML models.

Done before dimensionality reduction.

obtain data $\longrightarrow$ preprocessing $\longrightarrow$ datamodelling

Column normalisation :-

Let $X = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{array} \begin{array}{c} f_1 \; f_2 \; f_3 \cdots f_j \cdots f_d \\ \left[\begin{array}{ccc} & a_1 & \\ & a_2 & \\ & a_3 & \\ & \vdots & \\ & a_i & \\ & \vdots & \\ & a_n & \end{array} \right]_{n \times d} \end{array} \qquad Y = \left[\quad \right]$

This is a column. Each column is preprocessed in the same way.

The max & min values from $a_1, a_2, \ldots\ldots, a_n$ are taken a_{max} & a_{min}.

Now $a_1, a_2, a_3, \cdots, a_n$ are transformed to $a_1', a_2', a_3', a_4', \cdots, a_n'$ where

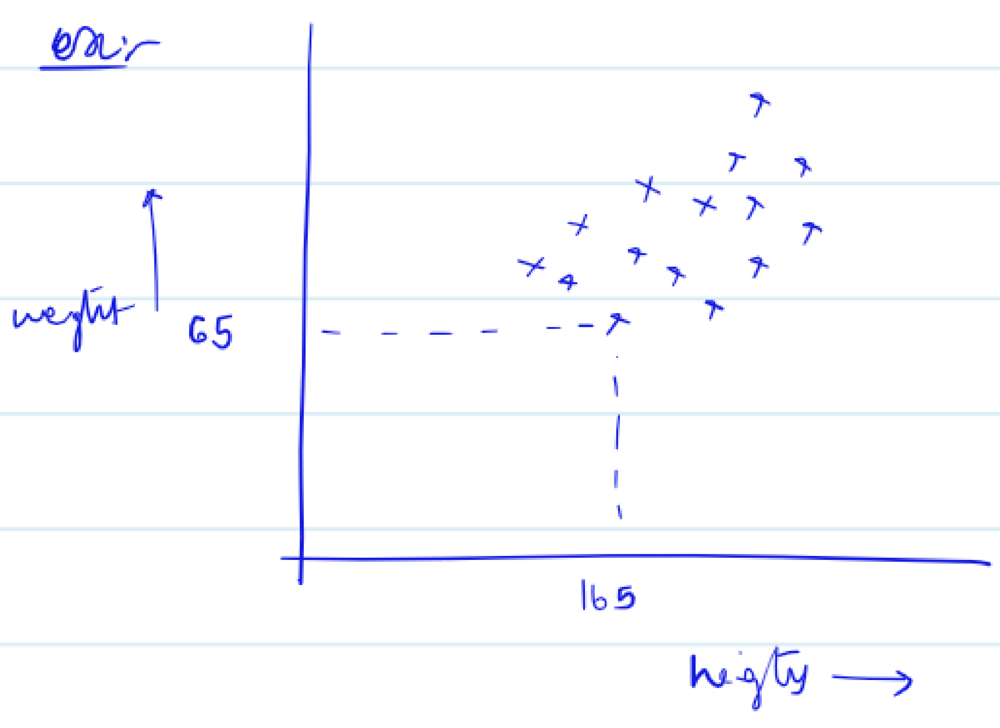
$$a_i' = \frac{a_i - a_{min}}{a_{max} - a_{min}}$$

Column Normalization $\longleftarrow$ All a_i' values lie b/w 0 & 1

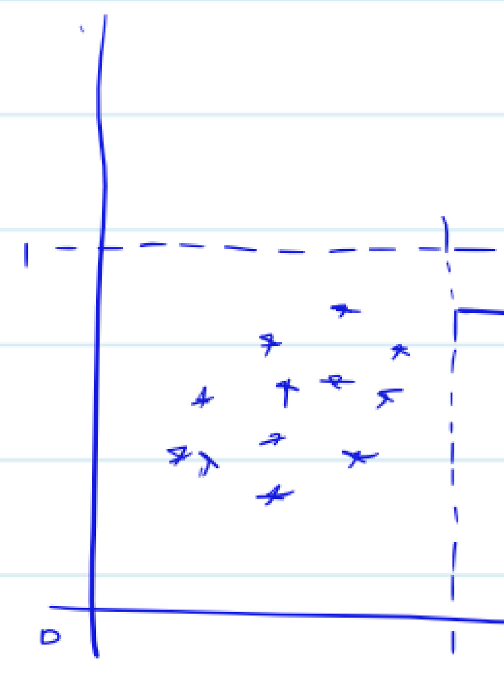
why ? :- Get rid of scales and entire data is in the same scale. ex:- Heights (cm), weights (Kg) dataset when normalised becomes [0 & 1] dataset.

Geometric Intuition :-

ex:-



$\Longrightarrow$



$\longrightarrow$ unit square. All of the data is squished into a 1×1 square without disturbing any of the relationship b/w the data.

If data is 3D, unit cube, - - - - - .

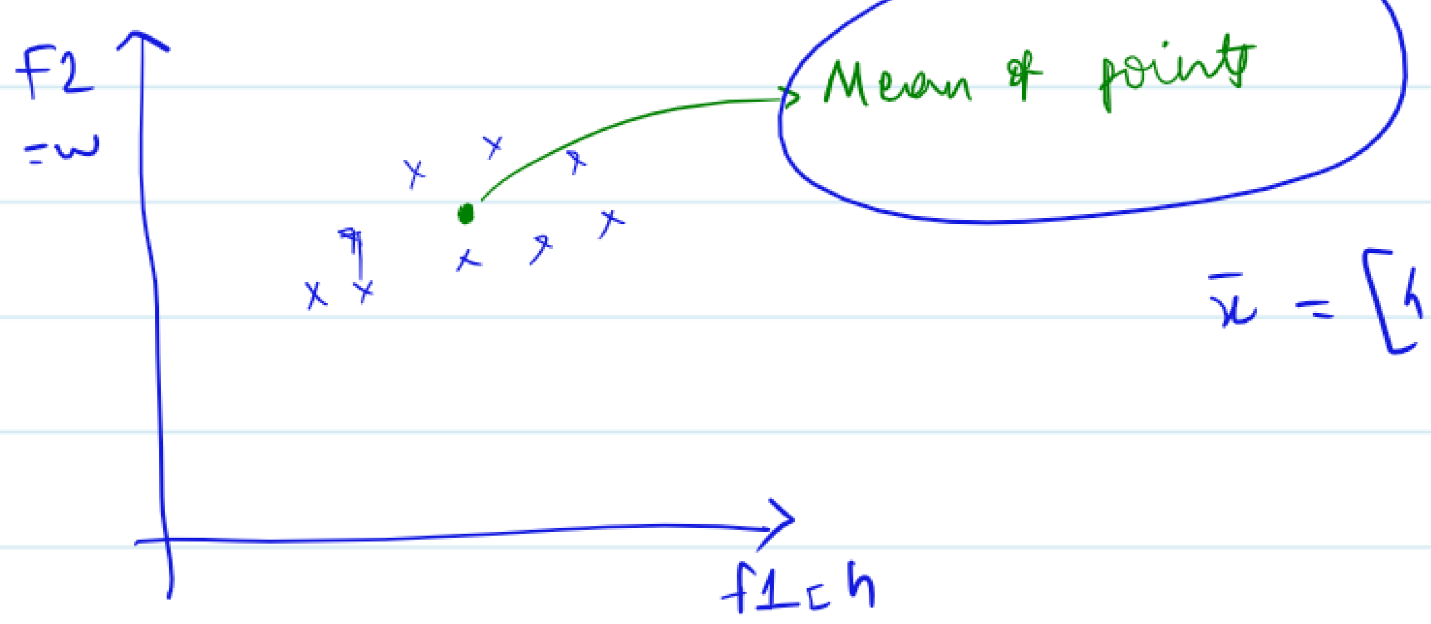
Mean of a matrix :-

$$X = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ \vdots \\ i \\ \vdots \\ n \end{array} \begin{array}{c} f_1 \; f_2 \; f_3 \cdots f_j \cdots f_d \\ \left[\begin{array}{c} \\ \\ \\ \longleftarrow x_i \longrightarrow \\ \\ \end{array} \right]_{n \times d} \end{array}$$

mean vector $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$

$\longrightarrow$ mean vector = central vector.

Geometric Intuition :-



$\bar{x} = [h_{\bar{x}}, w_{\bar{x}}]$, $h_{\bar{x}} = mean(h_i)_{i=1}^{n}$

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

Column Standardization :-

column normalization → b/w 0 & 1, gets rid of scale, converts to a unit hyper cube.

column standardization → more commonly used in practice.

$$X = \begin{array}{c|c} & f_1 \; f_2 \; f_3, \cdots \cdots f_j \cdots \cdots f_d \\ \hline \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ \vdots \\ i \\ \vdots \\ n \end{array} & \left[\begin{array}{ccc} & \boxed{\begin{array}{c} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_i \\ \vdots \\ a_n \end{array}} & \end{array} \right]_{n \times d} \end{array}$$

$a_1, a_2, a_3, a_4, \cdots, a_i, \cdots a_n$ → any distribution

↓

$a_1', a_2', a_3', \cdots \cdots, a_i', \cdots a_n'$ → $\text{mean}\{a_i'\}_{i=1}^{n} = 0$

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

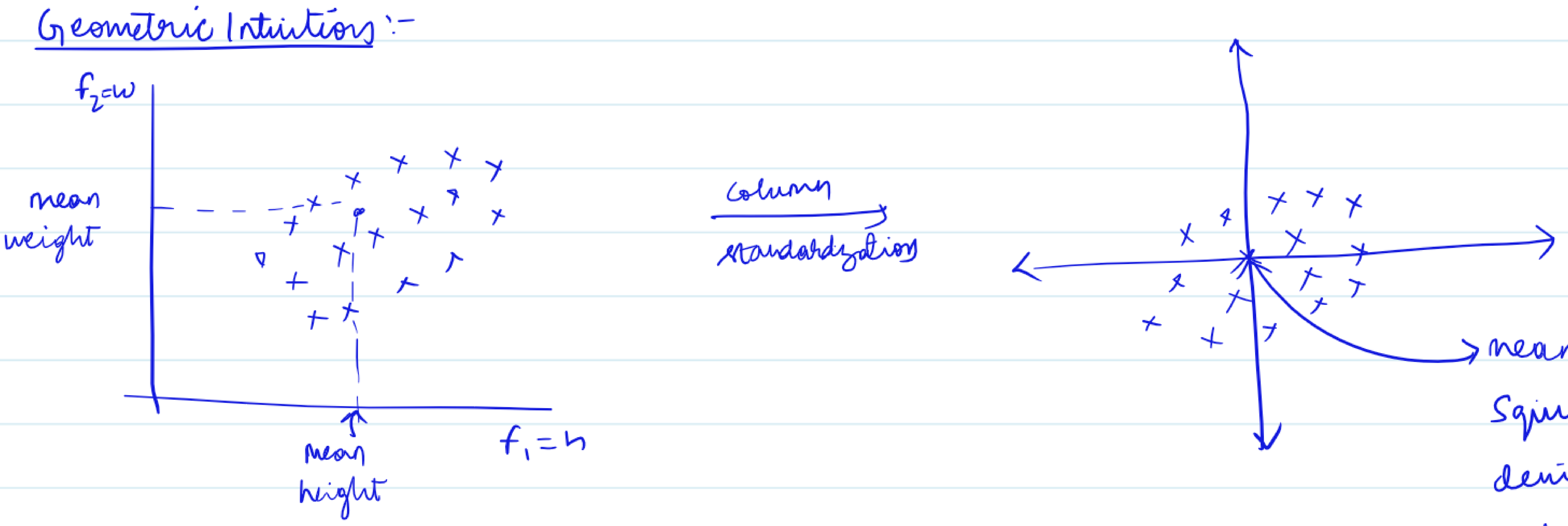
$\bar{a} = \text{mean}\{a_i\}_{i=1}^{n}$ $\qquad s = \text{std-dev}\{a_i\}_{i=1}^{n}$

$a_i' = \dfrac{a_i - \bar{a}}{s}$ → looks similar to standard Normal Variate $z = \dfrac{x-\mu}{\sigma}$

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

$z \sim N(0,1)$

Geometric Intuition :-



→ mean-vector is at origin. Squished such that standard deviation is 1 while keeping relationship b/w data intact

Column Standardization = mean centering + scaling.

Covariance of data Matrix :-

$$X = \begin{array}{c|c} & f_1 \; f_2 \cdots j \cdots f_d \\ \hline \begin{array}{c} 1 \\ 2 \\ 3 \\ \vdots \\ i \\ \vdots \\ n \end{array} & \left[\begin{array}{c} \\ \\ \\ \\ \end{array} \right] \end{array}$$

covariance matrix of X, $S = \left[\begin{array}{c} S_{ij} \end{array} \right]_{d \times d}$ → Square Matrix

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

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

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

$cov(x, x) = var(x)$

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

} Properties of covariance.

$S = \begin{bmatrix} \ddots & S_{ij} \\ S_{ji} & \ddots \end{bmatrix}_{d \times d}$ → Square Symmetric matrix

→ Variance of features (diagonal elements) = 1. Since variance of element in it is 1.

Let X : column standardized → mean $(f_i) = 0$

std-dev $(f_i) = 1$

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

μ_1, μ_2 → 0. Since column standardized.

$\Rightarrow cov(f_1, f_2) = 1/n \sum_{i=1}^{n} x_{i1} * x_{i2}$ component wise multiplication = dot product

$\Rightarrow cov(f_1, f_2) = (f_1^T f_2) * 1/n$

$$\begin{array}{c|ccccc} & f_1 & f_2 & \cdots & f_j & \cdots f_d \\ \hline 1 & f_{11} * f_{12} & & & & \\ 2 & & & & & \\ 3 & & & & & \\ \vdots & & & & & \\ i & f_{i1} * f_{i2} & & & & \\ \vdots & & & & & \\ n & & & & & \end{array}$$

$S = \frac{1}{(n-1)} \underbrace{(X^T)}_{d \times n} \underbrace{(X)}_{n \times d}$ = d×d matrix. (assuming X has been stdized)

Assuming Sample data is given

If population is given,

$S = \frac{1}{n} (X^T)(X)$

→ For any parameter θ, our estimate $\hat{\theta}$ is unbiased if, $E\{\hat{\theta} - \theta\} = 0$

→ Covariance is not a dimensionality reduction technique. Covariance matrix is used for PCA

→ We can use np.cov to get covariance.

Exploration of MNIST Dataset :-

→ 28×28 size image for each number.

→ 60K train + 10K test

→ $D = \{x_i, y_i\}_{i=1}^{60K}$ $y_i \in \{0, 1, 2, \ldots 8, 9\}$ x_i : 28 × 28

image → numerical matrix (NOT DATA MATRIX) → This is Flattening

↳ All row elements are placed one after the other resulting in long matrix with single column

↳ (row flattening)

[784 × 1] Size matrix

$$\Rightarrow X = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \vdots \\ 60000 \end{array} \begin{array}{c} f_1 \; f_2 \cdots f_{784} \\ \begin{bmatrix} \; & \; & \; \\ & & \\ & & \\ & & \\ & & \end{bmatrix} \end{array}$$

→ t SNE is used to convert 784 dimension dataset to 2 dimensional