Principal Component Analysis

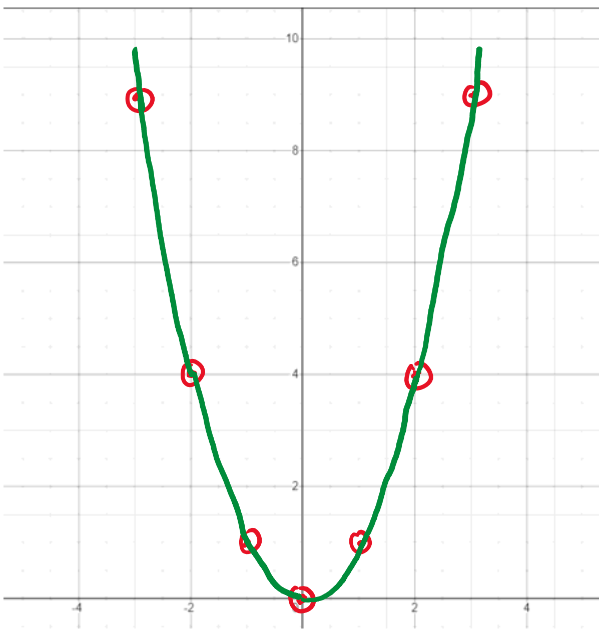
Thursday, September 12, 2024 3:23 PM

☆ It is used to reduce dimensionality (dimensions)

☆ Why to reduce dimensions?

1. Reduces complexity in our formulae
2. Reduces execution time (training time)
3. To reduce "curse of dimensionality"

☆ Curse of dimensionality



$y = x^2$

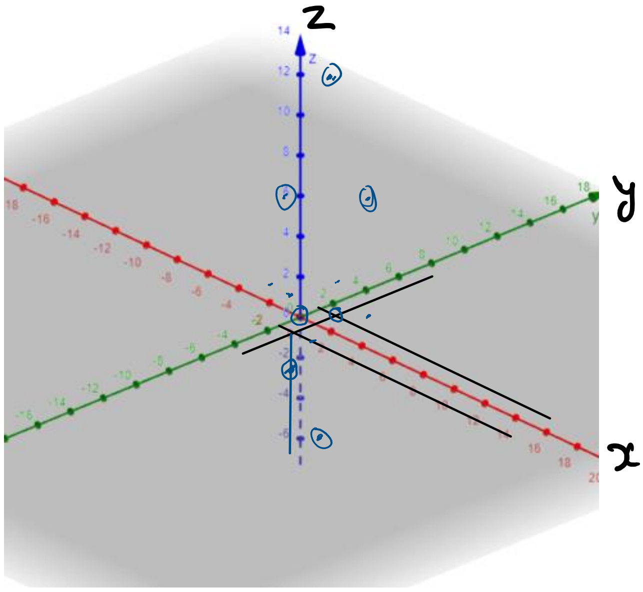
x	-3	-2	-1	0	1	2	3
y	9	4	1	0	1	4	9

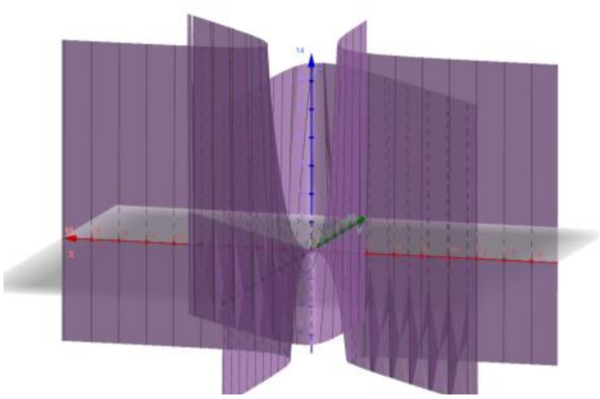
Just from these 7 points, we can very well understand the behaviour of this function.

Now let's try to do a similar thing with 3 dimensions.

Suppose $z = x^2y - y^2x$

x	1	1	2	0	-2	3	-1
y	1	-1	-1	0	1	2	3
z	0	-2	-6	0	6	6	12





← Actual curve of the function

← Actual curve of the function

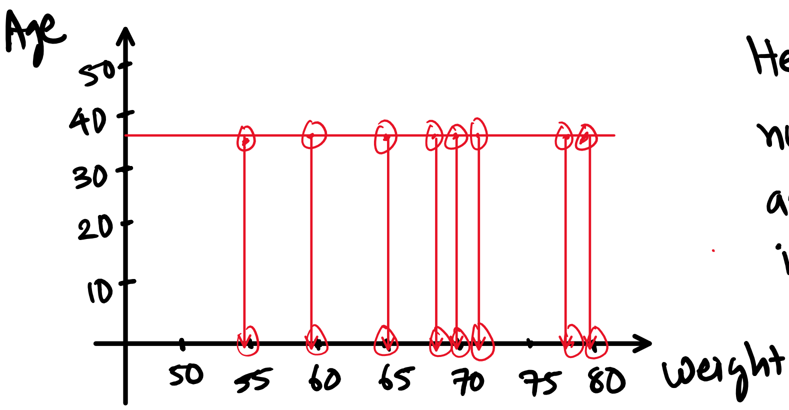
Curse of dimensionality says tha as we add a dimension, the distance between the same points increases drametically.

☆ How to reduce dimensions?

By removing unnecessary feature – $\boxed{\text{Case-1}}$

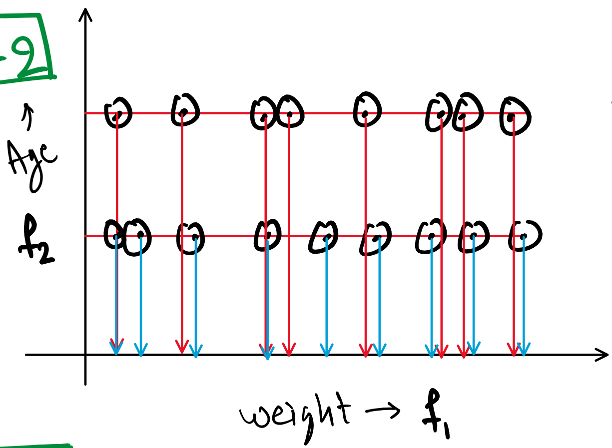
Consider the following table:

Weight:	80	60	55	65	70	68	72	98	79	85
Age:	35	35	35	35	35	35	35	35	35	35
Diabetic:	1	0	1	0	0	1	0	1	1	1



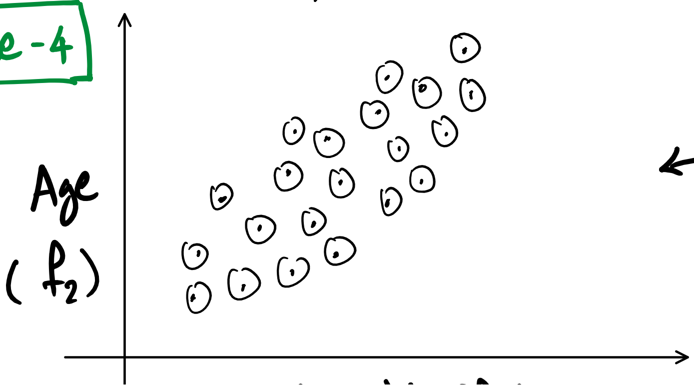
Here 'Age' dimension has no effect on our analysis and hence, if we remove it, we will not loose any information.

$\boxed{\text{Case-2}}$



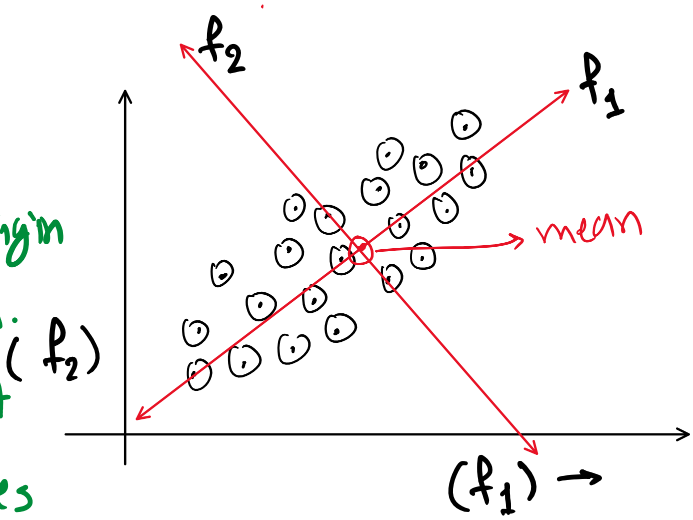
$\boxed{\text{case-3}}$

$\boxed{\text{Case-4}}$



← Here PCA will help us to reduce dimension! Let's see that...

Let's see that...



☆ We will perform two steps:

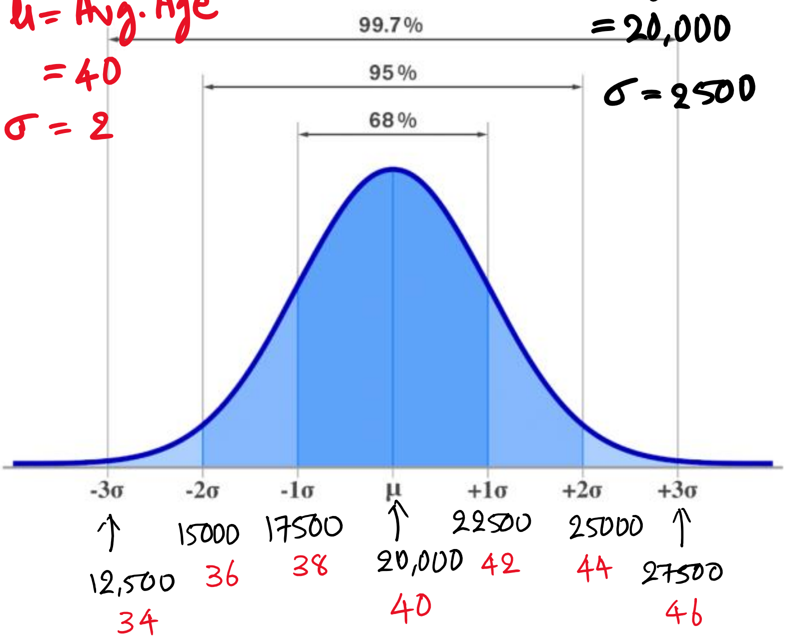
① Translate (move) our origin to the mean of the data.

② Rotate our axis such that our one of the features aligns in the direction of maximum variance.

☆ Step-1: Translate the origin to the mean of the data

μ = Avg. Age
= 40
σ = 2

μ = Avg. Income
= 20,000
σ = 2500



We have solved this kind of problem in the past by computing z-score.

Let's apply the same technique here:

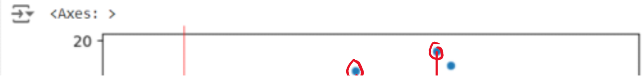
Before:

```
[32] import numpy as np
     import seaborn as sns
     import matplotlib.pyplot as plt
```

```
[50] f1 = np.random.normal(3, 2, size=200)
     m = 2
     c = -3
     y = m * f1 + c
     f2 = np.random.normal(y, 5)
```

```
sns.scatterplot(x=f1, y=f2)
```

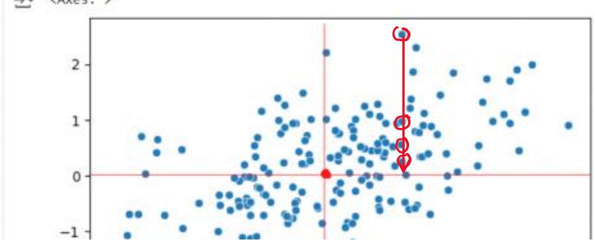
<Axes: >

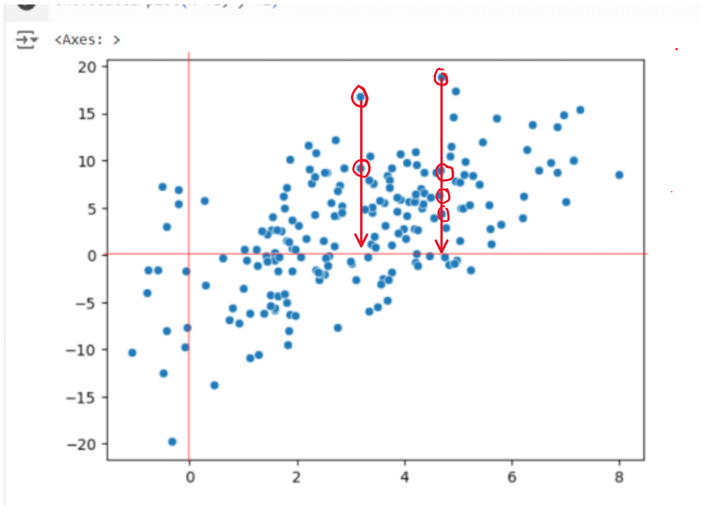


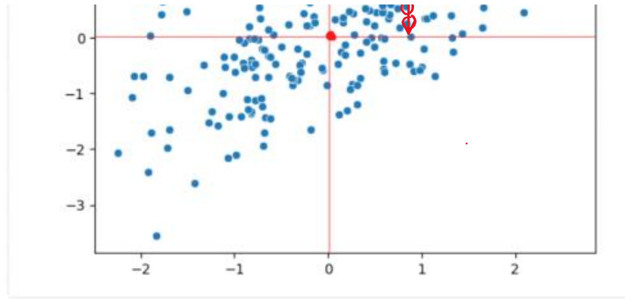
After step-1:

```
new_f1 = (f1 - f1.mean())/f1.std()
new_f2 = (f2 - f2.mean())/f2.std()
sns.scatterplot(x=new_f1, y=new_f2)
```

<Axes: >

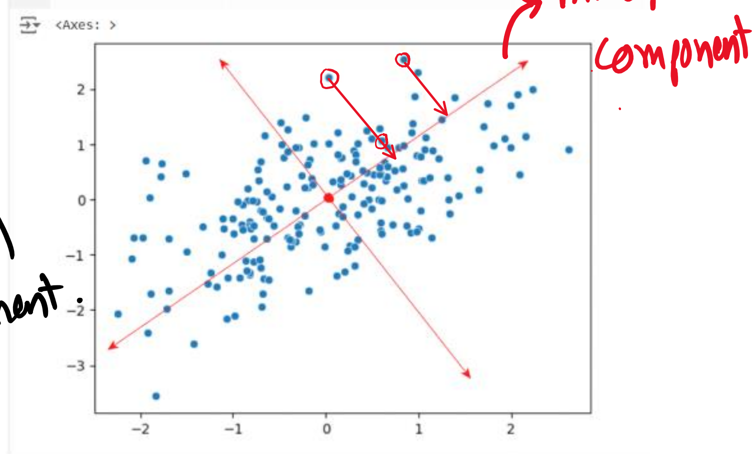




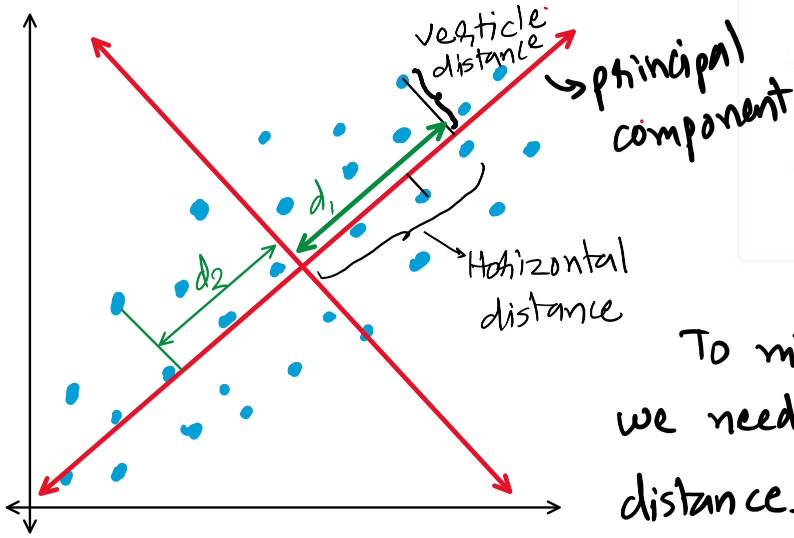


After Step-2:

```
new_f1 = (f1 - f1.mean())/f1.std()
new_f2 = (f2 - f2.mean())/f2.std()
sns.scatterplot(x=new_f1, y=new_f2)
```

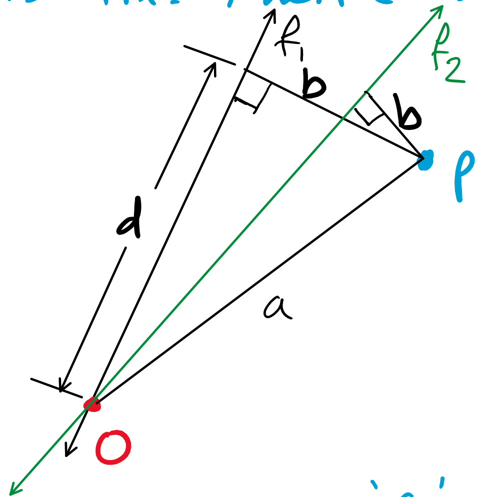


☆ How to rotate the axis?



To minimize the verticle distances, we need to maximize the horizontal distances.

Mathematical Proof: suppose we have a point 'P' (which is fixed, hence we cannot move it.) and the origin



'O' (which is also fixed at mean of the data, so we can not move it as well.)

Let perpendicular distance of 'P' from f_1 is b.

'a' which is distance from 'P' to 'O' is also constant because both 'P' & 'O' are fixed. And, 'd' is the horizontal distance of 'P' from 'O'

... And, a is the horizontal distance of p from 'O'. Now, if we rotate our axis such that we get a new axis 'f_2' which has lower vertical distance from 'p'.

Now, in both the right-angled triangles

$$d^2 + b^2 = a^2$$ where a^2 is constant.

$\therefore$ If we want to reduce b, we have to increase d.

$$\underbrace{(5)^2}_{?\ \text{increase!}} + \underbrace{(12)^2}_{\text{Reduce}} = \underbrace{(13)^2}_{\text{constant}}$$

Back to Axis Rotation: To rotate the axis, we will focus on increasing the total horizontal distance.

Total Horizontal distance is sum of horizontal distance of every point. But some distances will come -ve & if we straight away add them, they may cancel-out each other. So, we need to square them before adding.

$\therefore$ Total Horizontal Distance is also known as

Sum of Squares Distance (SS Distance)

$$\boxed{\text{SS Distance} = d_1^2 + d_2^2 + d_3^2 + \ldots + d_n^2}$$

(This SS Distance itself is also the 'Eigen Value')

(This SS Distance itself is also the Eigen value)

Here, d_1^2 = distance of x_1 from mean $= (x_1 - \bar{x})^2$

d_2^2 = " " x_2 " " $= (x_2 - \bar{x})^2$

$\vdots$

d_n^2 = " " x_n " " $= (x_n - \bar{x})^2$

$\therefore$ SS Distance $= \sum (x_i - \bar{x})^2$

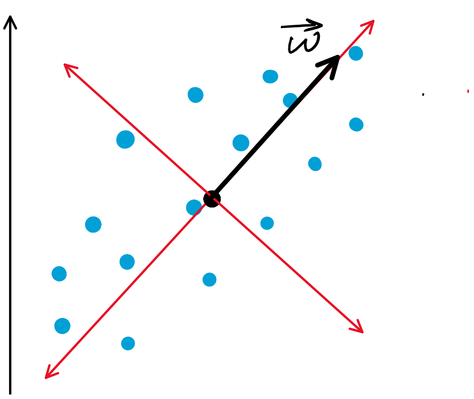
If we just divide this by d.o.f. or no. of observations we will get $\sum \frac{(x_i - \bar{x})^2}{n}$. This is the formula for variance!

☆ **The entire process will be:**

Step-1: Standardize the data (Move the origin)
(By doing $\frac{x - \bar{x}}{\sigma}$)

Step-2: Rotate the axis: By finding a line (or just a vector in that direction) $\vec{w}$ such that the variance along that direction is maximum

Converting step-2 into mathematical form:



variance $= \frac{1}{n} \sum (x_i - \bar{x})^2$

But $x_i = x_i \cdot \hat{w}$

$\therefore x_i = x_i \cdot \vec{w}$

$$\therefore x_i = \frac{x_i \cdot \vec{\omega}}{\|\vec{\omega}\|}$$

Replacing x_i in eq^n of variance:

$$\text{variance} = \frac{1}{n} \sum \left(\frac{x_i \cdot \bar{\omega}}{\|\bar{\omega}\|} - \bar{x} \right)^2$$

Now we want such a $\vec{\omega}$ so that this variance becomes maximum.

$$\underset{\vec{\omega}}{\operatorname{argmax}} \ \frac{1}{n} \sum \left(\frac{x_i \cdot \bar{\omega}}{\|\bar{\omega}\|} - \bar{x} \right)^2$$

But $\bar{x}$ (= mean) will be 0 as we have shifted the origin to the mean.

$$\underset{\vec{\omega}}{\operatorname{argmax}} \ \frac{1}{n} \sum \left(\frac{x_i \cdot \bar{\omega}}{\|\bar{\omega}\|} \right)^2$$

Simplifying the above formula (as we were doing in Lagrange's multiplier method):

→ Converting it into a constrained problem

$$\underset{\vec{\omega}}{\operatorname{argmax}} \ \frac{1}{n} \sum \left(x_i \cdot \bar{\omega} \right)^2 \quad s.t. \ \|\bar{\omega}\| = 1$$

→ Converting to unconstrained problem using Lagrange's multiplier:

$$\boxed{\vec{w} = \underset{\vec{w}}{\operatorname{argmax}} \ \frac{1}{n}\left(\sum x_i \bar{w}\right)^2 + \lambda\left(\|\bar{w}\| - 1\right)} \rightarrow \text{Objective of P.C.A.}$$

☆ Understanding the above formula:

Here, x_i are the dapoints & each datapoint has several features. ∴ One of our arrays will be somewhat like:

$$\begin{array}{c} x_1 \\ x_2 \\ \vdots \\ x_n \end{array} \begin{bmatrix} x_1^1, & x_1^2, & \cdots, & x_1^d \\ x_2^1, & x_2^2, & \cdots, & x_2^d \\ & \vdots & & \\ x_n^1, & x_n^2, & \cdots, & x_n^d \end{bmatrix}_{n \times d} \cdot \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}_{d \times 1} = \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix}_{n \times 1}$$

☆ Using Gradient Descent to find the best $\bar{w}$:

From the pdf of G.D., we know that $x^T \cdot x = x^2$

$$\therefore (x_i \cdot \bar{w})^2 = (x_i \cdot \bar{w})^T \cdot (x_i \cdot \bar{w})$$

$$\bar{w} = \underset{\bar{w}}{\operatorname{argmax}} \ \frac{1}{n} \sum \left((x_i \cdot \bar{w})^T \cdot (x_i \cdot \bar{w})\right) + \lambda\left(\|\bar{w}\| - 1\right)$$

Differentiating w.r.t. $\bar{w}$ to find gradient:

$$\frac{1}{n} \sum \frac{\partial}{\partial \bar{w}}\left(\bar{w}^T \cdot x_i^T \cdot x_i \cdot \bar{w}\right) + \frac{\partial}{\partial \bar{w}} \lambda\left(\|\bar{w}\| - 1\right)$$

$$\left[\because (a \cdot b)^T = b^T \cdot a^T\right]$$

A fact: $f(w) = w^T \cdot S \cdot w \Rightarrow \frac{\partial}{\partial w} f(w) = (S + S^T) \cdot w$

(Try to prove this yourself)

Using this fact, we will get:

$$\underbrace{x^T \cdot x \cdot \bar{\omega} = \lambda'}_{\text{Eigen Value}} \cdot \underbrace{\omega}_{\text{Eigen Vector}} \quad ; \lambda' = \text{costant} = -n\lambda$$