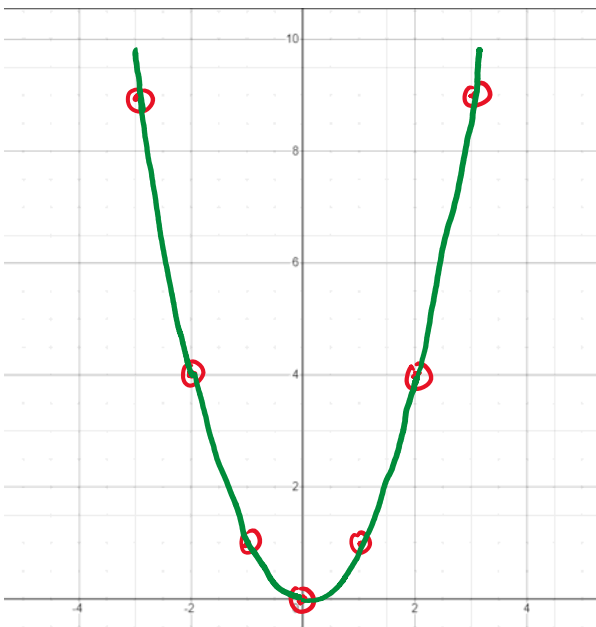


★ It is used to reduce dimensionality (dimensions)

★ 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



$$y = x^2$$

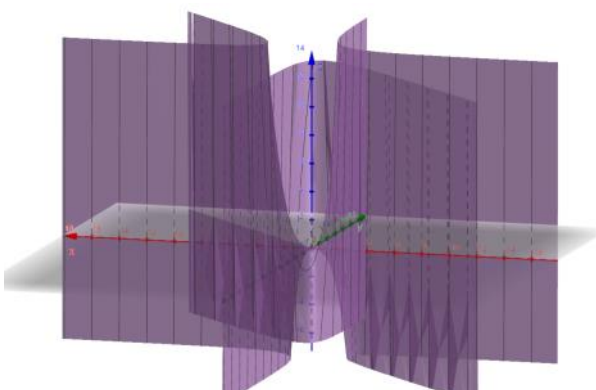
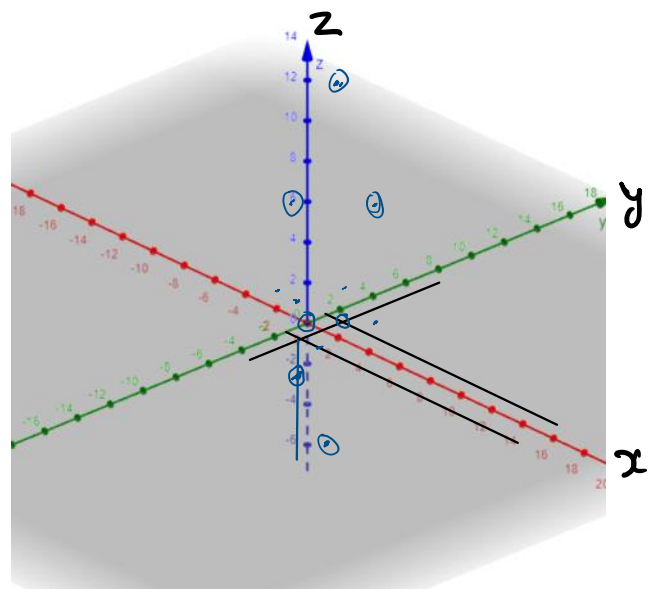
$x$	-3	-2	-1	0	1	2	3
$y$	9	4	1	0	1	4	9

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

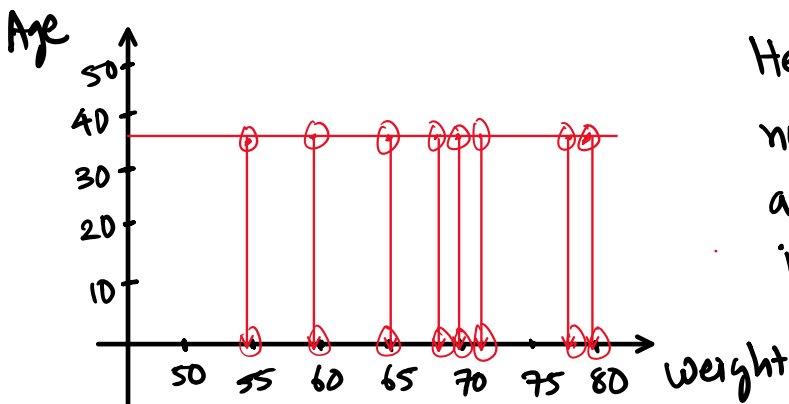
Curse of dimensionality says that as we add a dimension, the distance between the same points increases dramatically.

★ How to reduce dimensions?

By removing unnecessary feature - **Case-1**

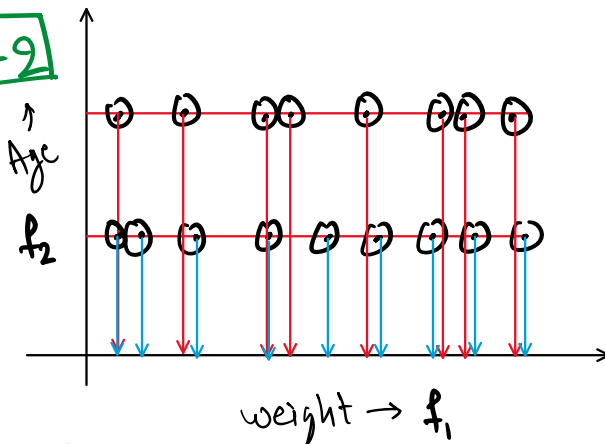
Consider the following table:

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 lose any information.

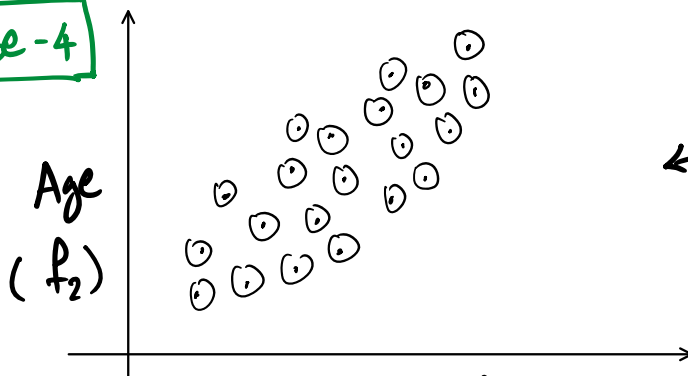
**Case-2**



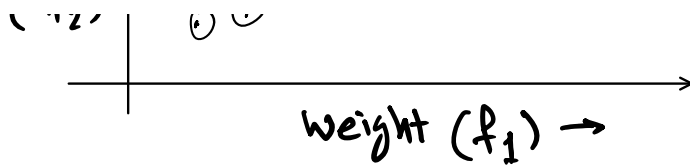
**Case-3**



**Case-4**



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

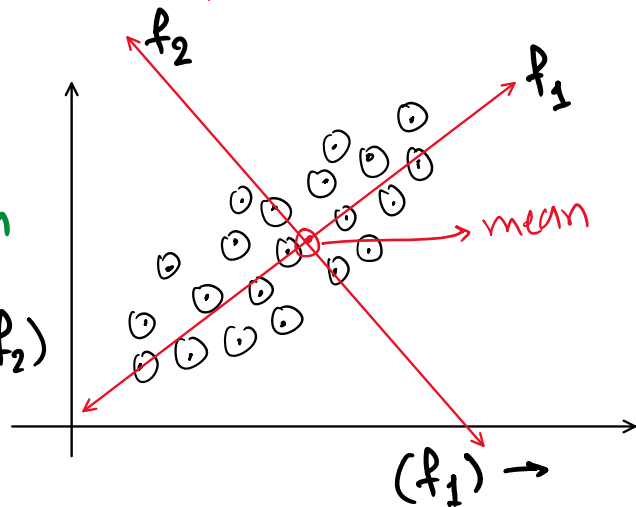


Let's see that...

★ 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.

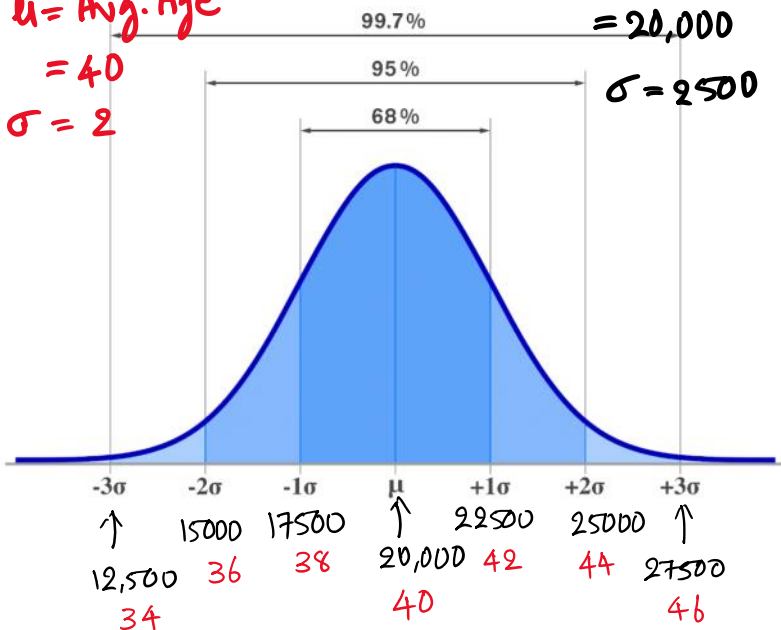


★ Step-1: Translate the origin to the mean of the data

$\mu = \text{Avg. Age}$   
 $= 40$   
 $\sigma = 2$

$\mu = \text{Avg Income}$   
 $= 20,000$   
 $\sigma = 2500$

We have solved this kind of problem in the past by computing z-score. Let's apply the same technique here:



Before:

After step-1:

```
[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)
```

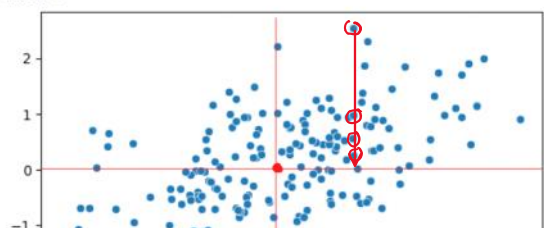
```
sns.scatterplot(x=f1, y=f2)
```

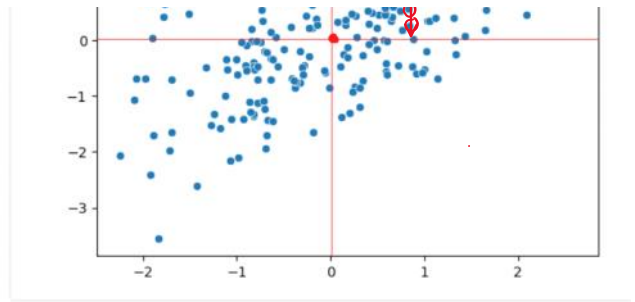
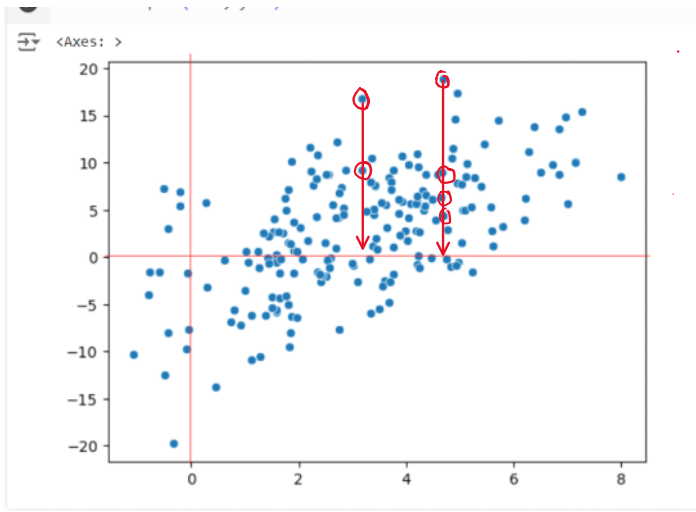
<Axes: >

20

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

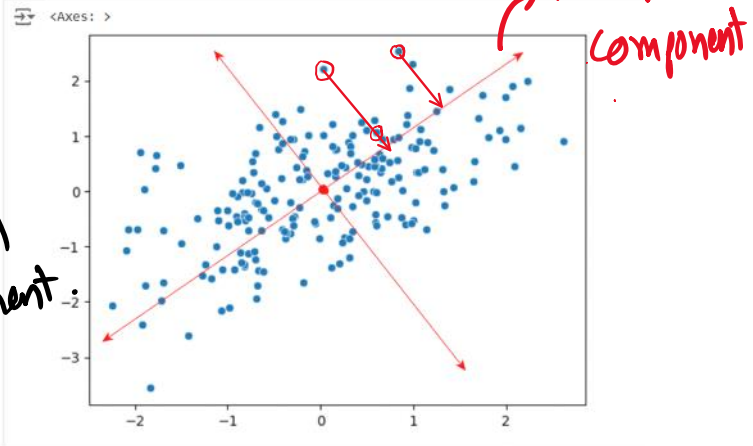
<Axes: >



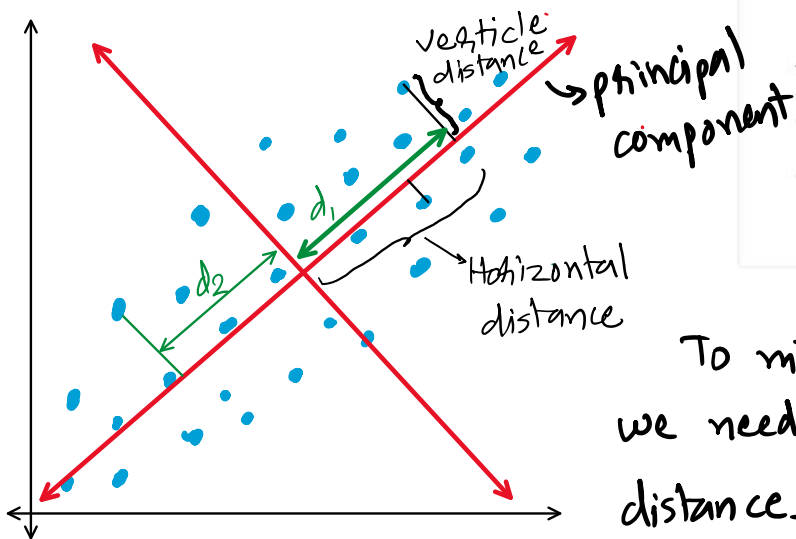


After Step-2:

```
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 vesticle 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'.

then,  $a$  is the horizontal distance of  $P$  from '0'. 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

$$d^2 + b^2 = a^2 \quad \text{where } a^2 \text{ is constant.}$$

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

$$(5)^2 + (12)^2 = (13)^2$$

$\downarrow$        $\downarrow$        $\downarrow$   
 ?    Reduce    constant  
 increase!

**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)**

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

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

(This  $\Rightarrow$  distance from  $x_i$  to  $\bar{x} = 0$ )

$$\begin{aligned}\text{Here, } d_1^2 &= \text{distance of } x_1 \text{ from mean} = (x_1 - \bar{x})^2 \\ d_2^2 &= \text{ " " } x_2 \text{ " " } = (x_2 - \bar{x})^2 \\ &\vdots \\ d_n^2 &= \text{ " " } x_n \text{ " " } = (x_n - \bar{x})^2\end{aligned}$$

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

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  $\frac{x - \bar{x}}{\sigma}$ )

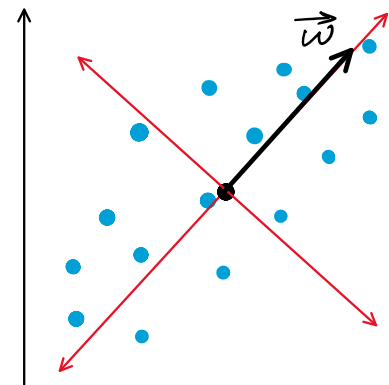
Step-2: Rotate the axis: By finding a line (or just a vector in that direction)  $\vec{w}$  such that the variance along that direction is maximum

Converting step-2 into mathematical form:

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

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

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



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



Replacing  $x_i$  in eq<sup>n</sup> of variance:

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

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

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

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

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

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

→ converting it into a constrained problem

$$\underset{\vec{w}}{\text{argmax}} \frac{1}{n} \sum (x_i \cdot \vec{w})^2 \text{ s.t. } \|\vec{w}\| = 1$$

→ converting to unconstrained problem using Lagrange's multiplier:

$$\vec{w} = \underset{\vec{w}}{\operatorname{argmax}} \frac{1}{n} \left( \sum x_i \vec{w} \right)^2 + \lambda (\|\vec{w}\| - 1)$$

→ Objective of P.C.A.

## ★ Understanding the above formula:

Here,  $x_i$  are the datapoints & each datapoint has several features.  $\therefore$  One of our arrays will be somewhat like:

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

## ★ Using Gradient Descent to find the best $\vec{w}$ :

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

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

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

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

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

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

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:

$$x^T \cdot x \cdot \bar{\omega} = \lambda' \cdot \omega \quad ; \quad \lambda' = \text{constant} = -n\lambda$$

