Principal Component Analysis

[Dimensionality Reduction]

→ Doubt

→ Variants of GD

→ PCA

Doubt in last class:

→ gradient does not change:

$$\frac{\partial L}{\partial w_1} = \underbrace{-\sum y_i x_{1i}}$$

This is not the complete gradient !!

$$\frac{\partial L}{\partial w_1} = -\sum y_i x_{1i} + \frac{\partial \lambda}{\partial w_1}\left(\sqrt{w_1^2 + \cdots}\right)$$

$$\frac{\partial L}{\partial w_1} = -\sum y_i x_{1i} + \frac{1}{2\|w\|} \cdot 2w_1 = \underline{\underline{F(w_1)}}$$

Hence gradient does change.

Variants of Gradient Descent

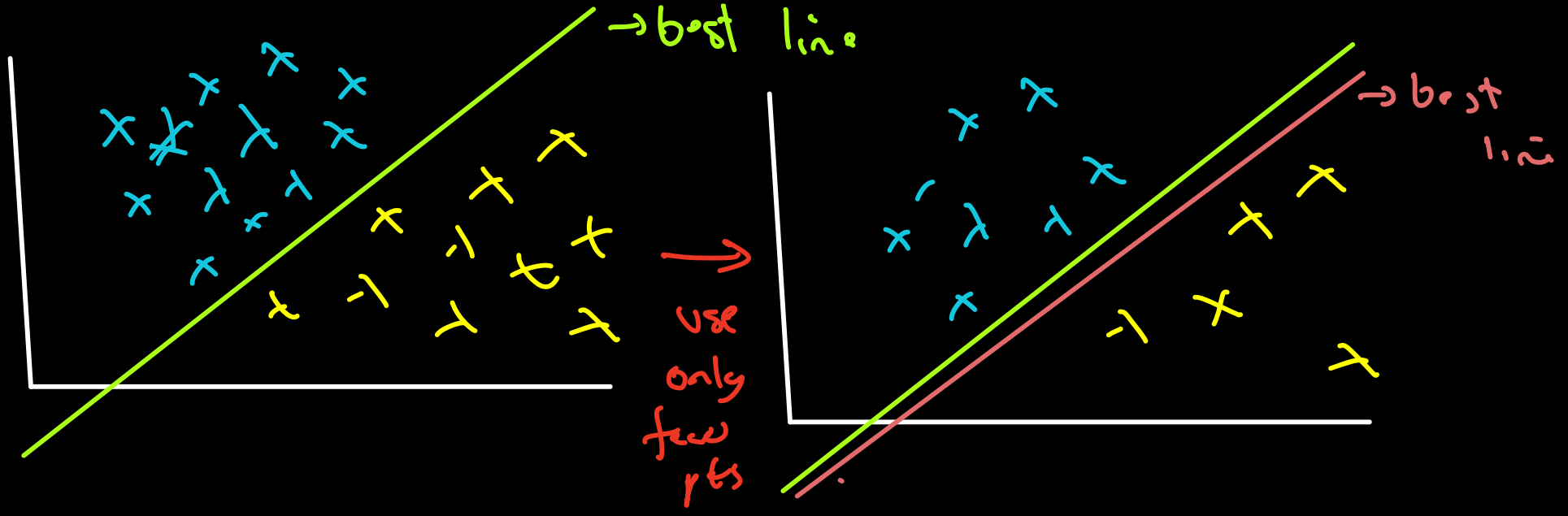
GD: $\theta^{t+1} = \theta^t - \eta \frac{\partial f}{\partial \theta}$

$\rightarrow \theta^{t+1} = \theta^t - \eta \sum_{i=1}^{n} \frac{\partial f(x_i)}{\partial \theta}$

Use all data points for 1 update.

→ high compute time

Q: Let's say I have 1M data points, can I just update θ with 0.5 M points?



→ But both best lines are similar !!

Idea:

For a single update use only a few data points (batch) randomly and do many updates.

↳ This is called **Batch Gradient Decent**

→ Very similar accuracy
→ Huge speed improvement

Batch GD:

$$\theta^{t+1} = \theta^t - \eta \sum_{i \in B} \frac{\partial f(x_i)}{\partial \theta}$$; B is a random sample of N

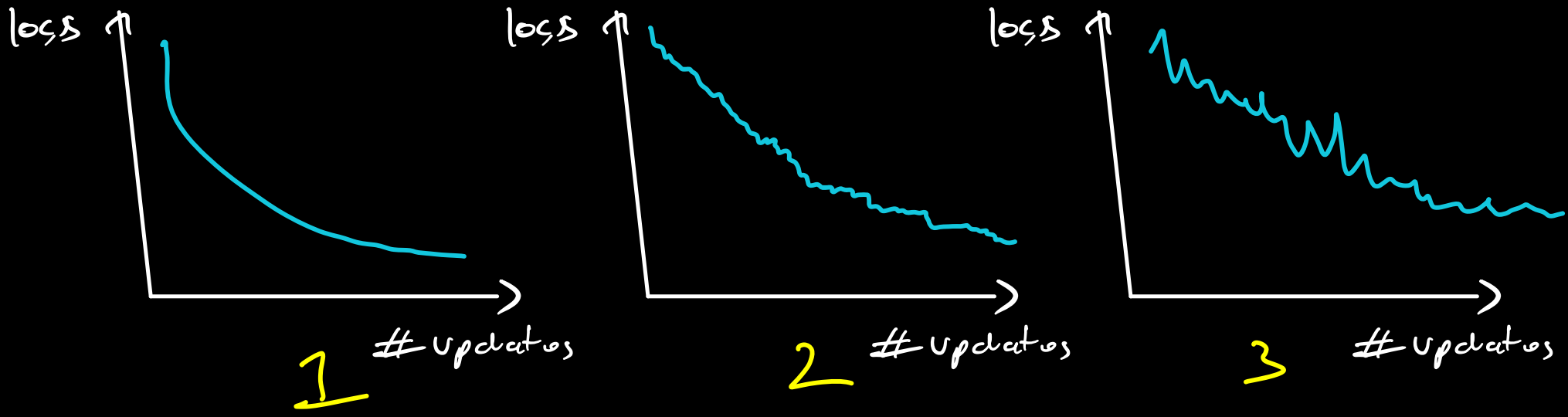
Q: Could I just use 1 point?

Stochastic GD:

$$\theta^{t+1} = \theta^t - \eta \frac{\partial f(x_k)}{\partial \theta}$$; k is random number from 1 – n

→ Faster but too many updates are needed
→ less preffered.

Quiz: Match loss trajectory of respective variant



a) 1 - sgd
2 → bgd
3 → gd

b) 1 → gd
2 → sgd
3 → bgd

c) 1 → gd
2 → bgd
3 → sgd

Principal Component Analysis

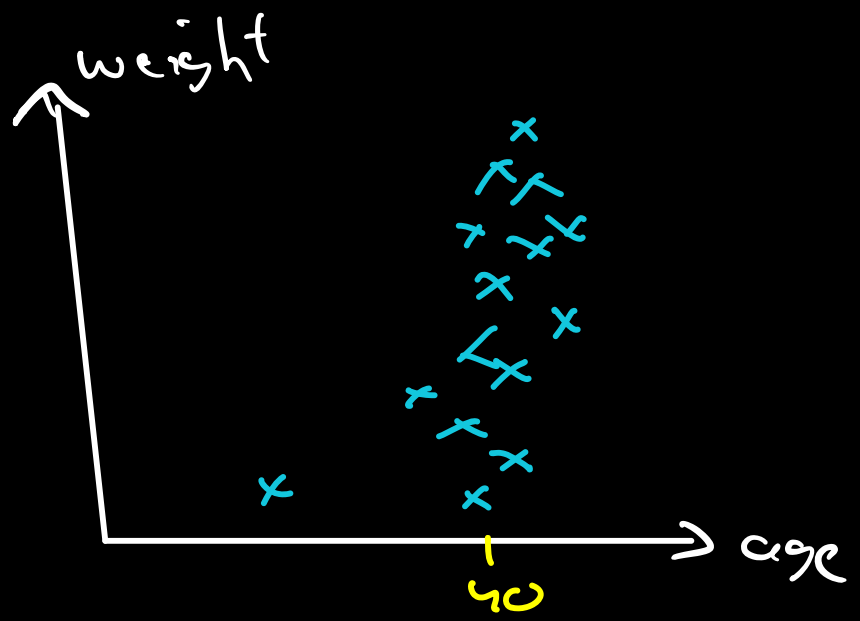
Let's take a few examples:

Diabetes prediction:

W	A	D
:	:	Y
:	:	N
:	:	Y
!	!	!

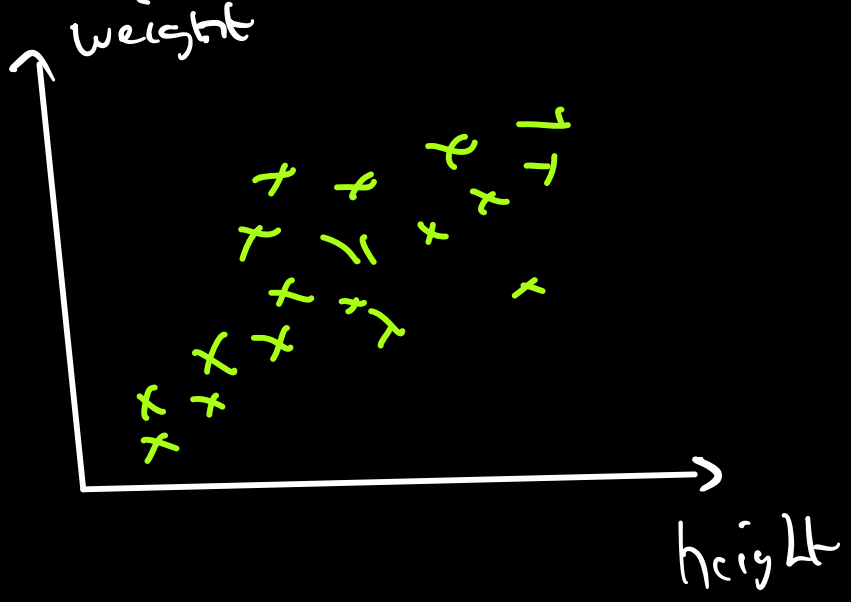
f_1 = weight

f_2 = age



Q: If you could only use 1 feature which would you use?

a) Age b) Weight c) A + W

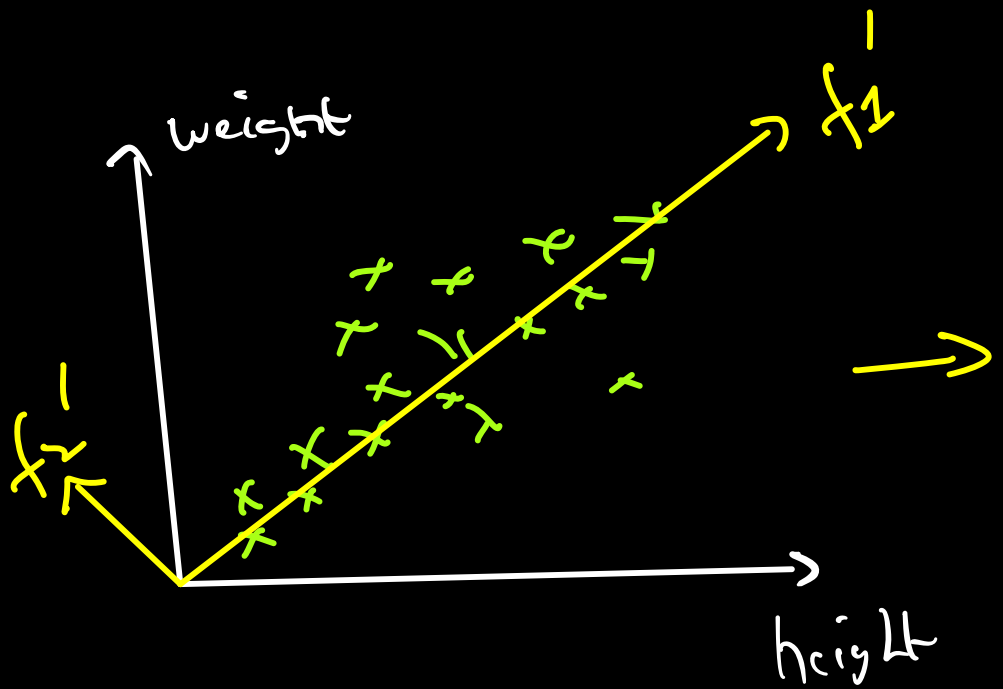


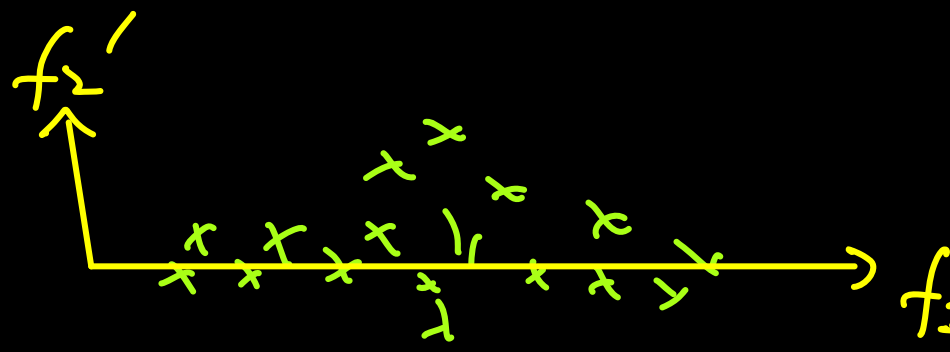
Q: which feature?

a) weight

b) height

c) w + h





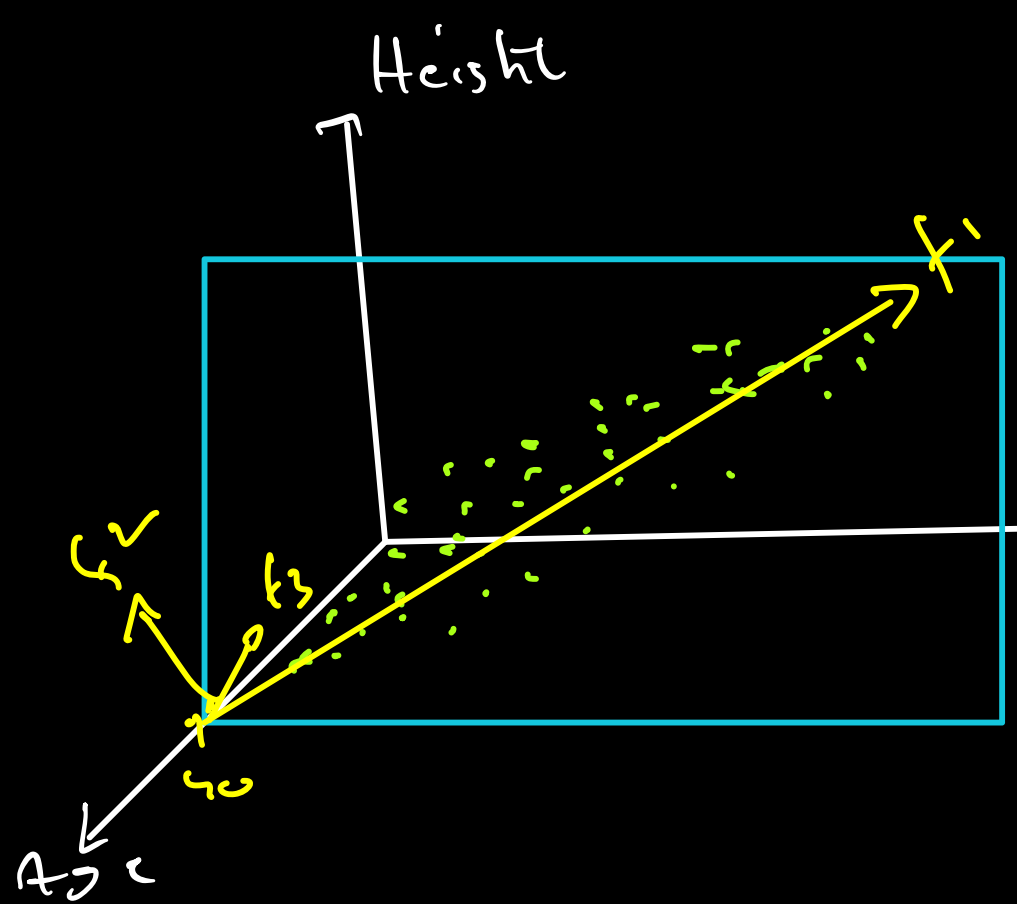
Q: which feature?

a) f1

b) f2

c) $f_1 + f_2$

d) $f_1 - f_2$



Q: What is the order of feature importance?

a) $f_1 > f_2 > f_3$

b) $f_3 > f_2 > f_1$

c) $f_1 > f_3 > f_2$

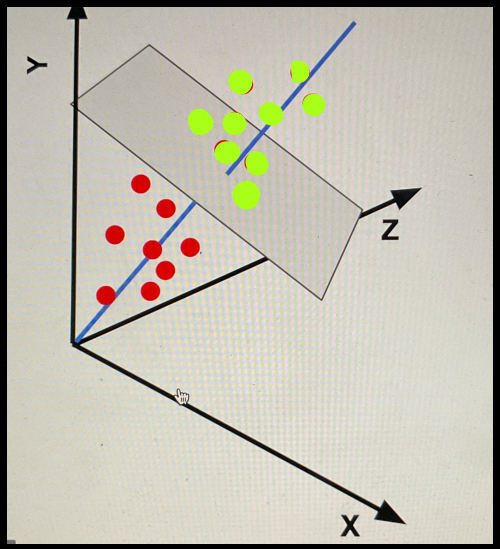
Goal:

→ We see that all the features don't carry equal amounts of info.

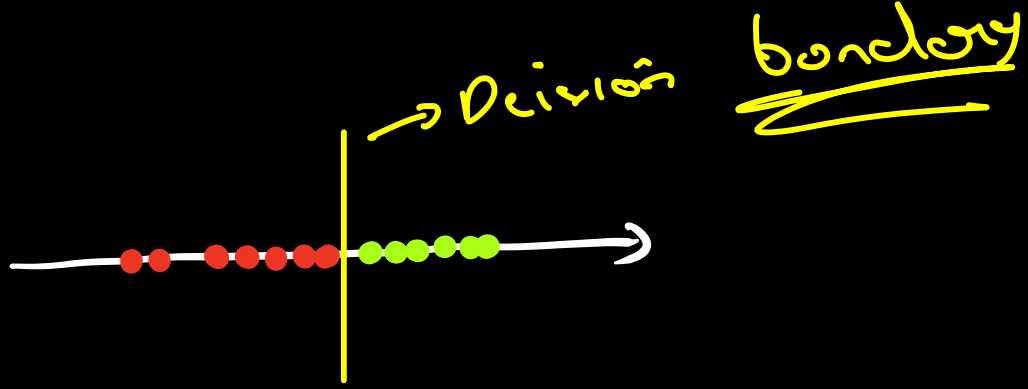
→ Sometimes in ML, we need to reduce the number of features.

- Visualisation
- Compression (Space saving)
- Faster ML training

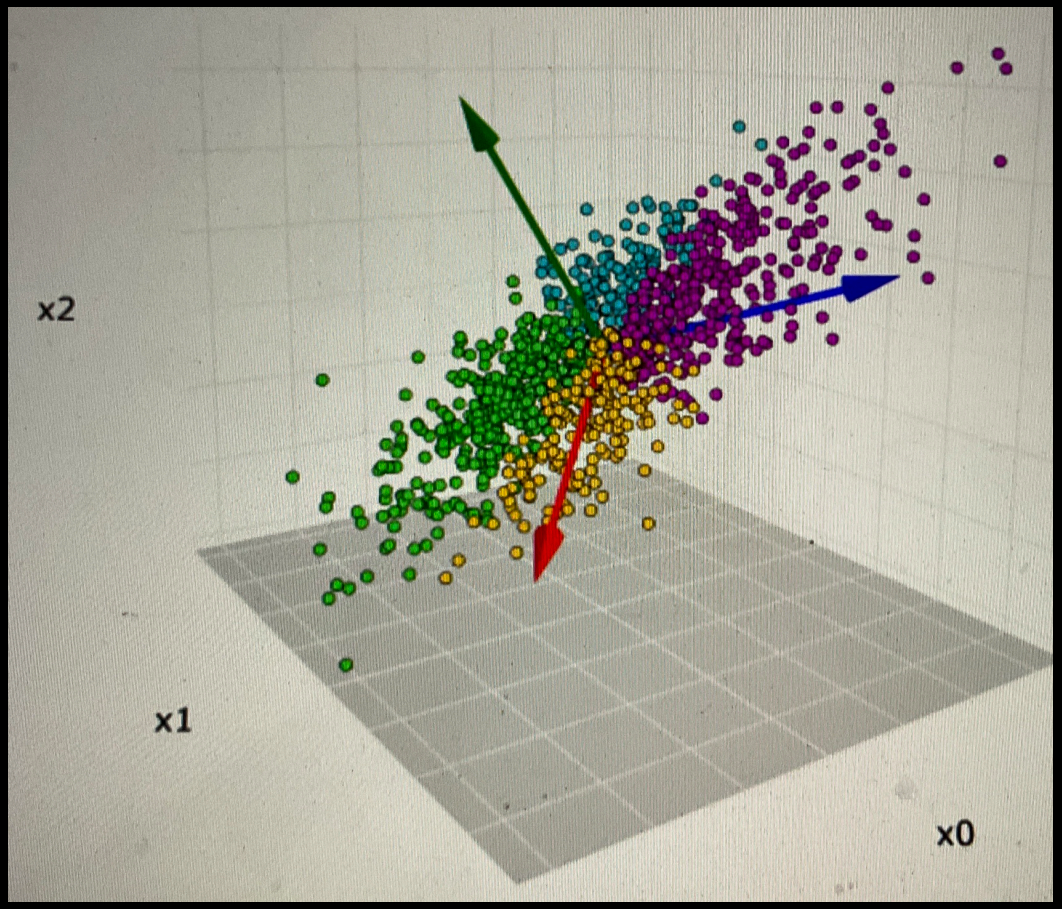
3D → 1D

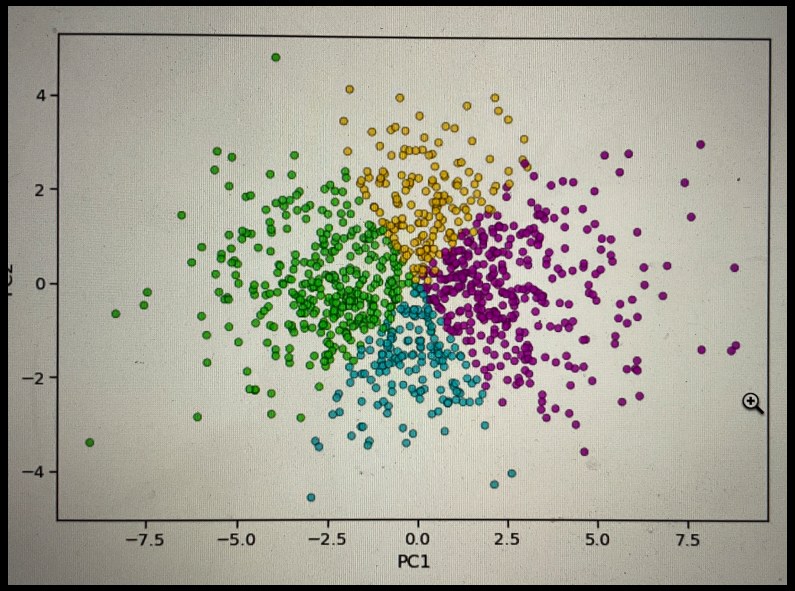


→



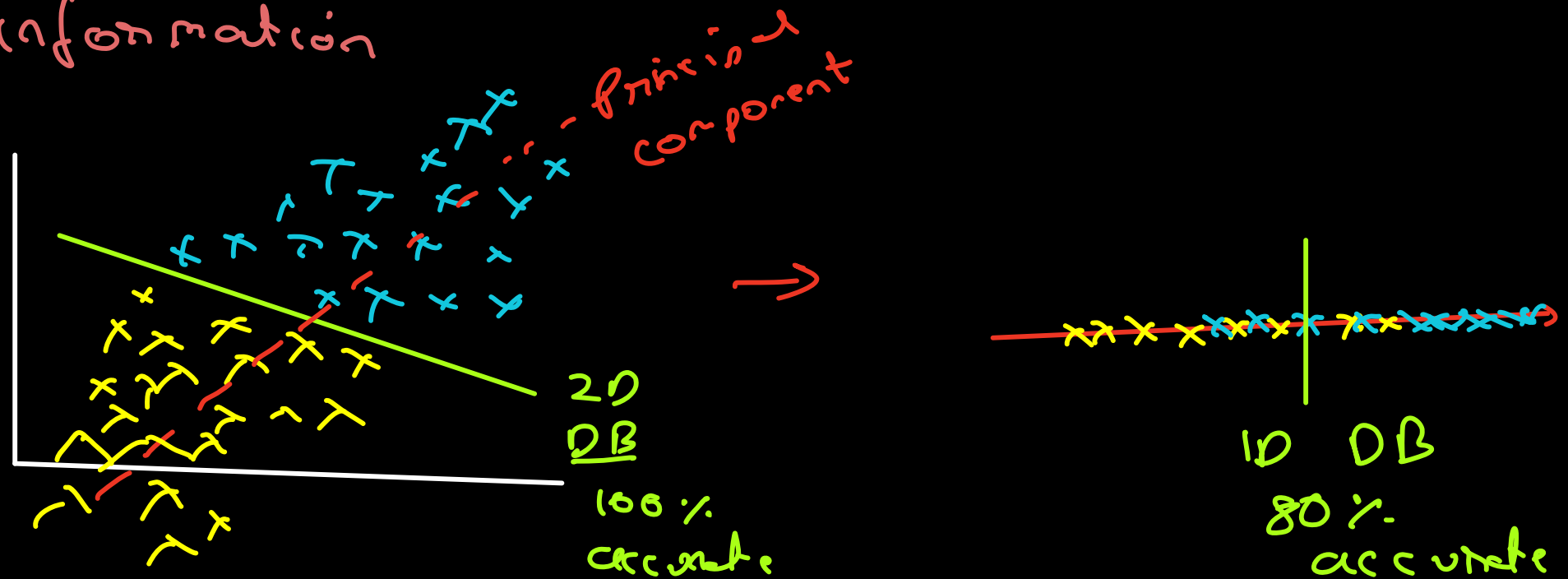
3D - 2D





Principal component analysis refers to the act of finding new axis to represent the data so that a few principal components may contain most information.

However, this may lead to some loss of information

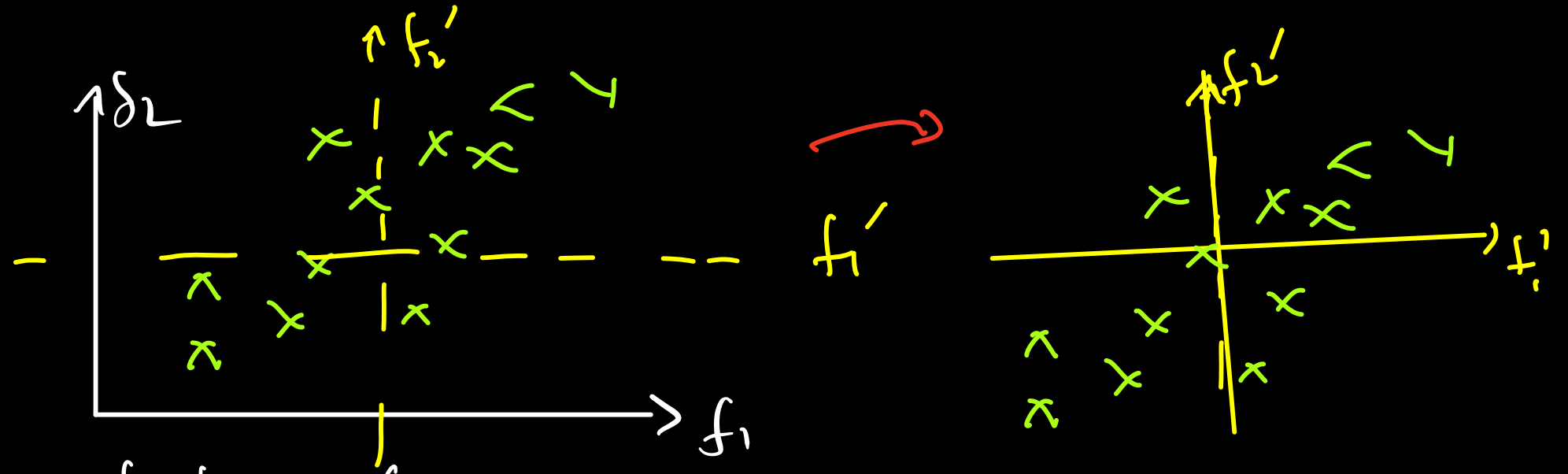


PCA Mathematics

→ Change of axis

→ Retain most information in few axis.

Step-1 : Centering



f_1 f_2 .. f_n

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad X$$

centers $\longrightarrow$ $[\bar{f_1}\ \ \bar{f_2}\ ..\ \bar{f_n}]$

$X - X.mean() = X_c$

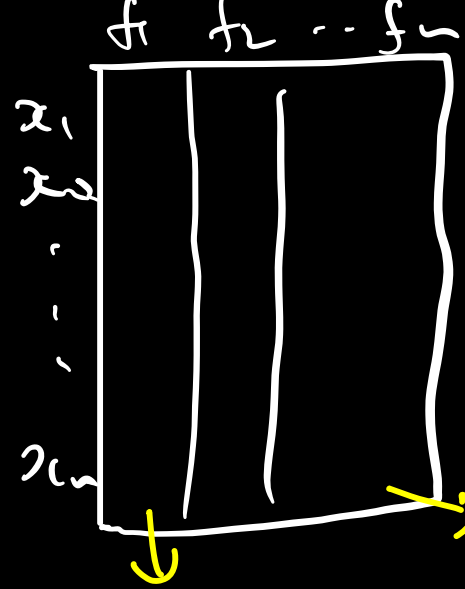
Step -2: Standardize / Scaling

We need to make different features such as height (4 - 6) ft and weight (40 - 150) kg comparable.

Best is to use Standard Scaling because it is less affected by outliers

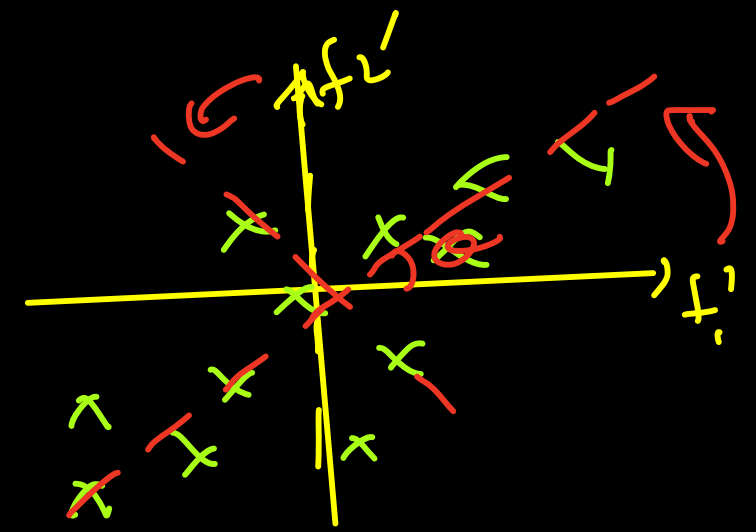
Combining step 1 and 2

$$x' = \frac{x - x.mean()}{x.std()}$$



→ use sklearn

Step - 3: Rotate

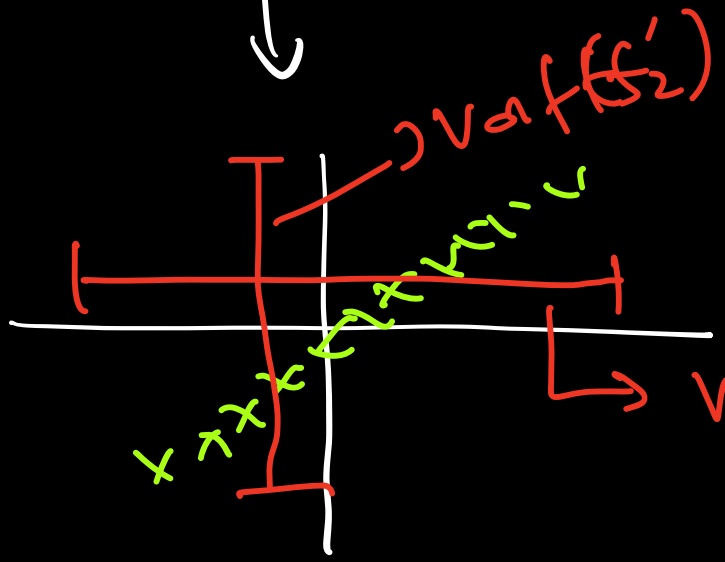


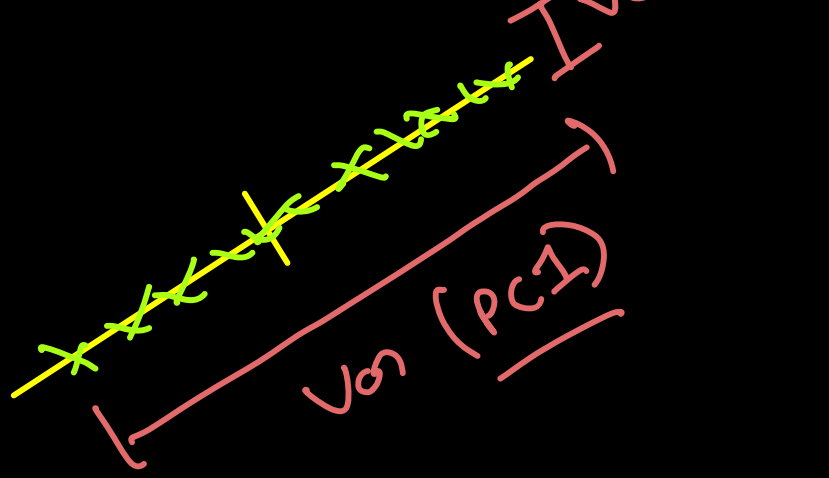
How to rotate?

Q: Any suggestions on how to determine the best axis?

→ maximise variance

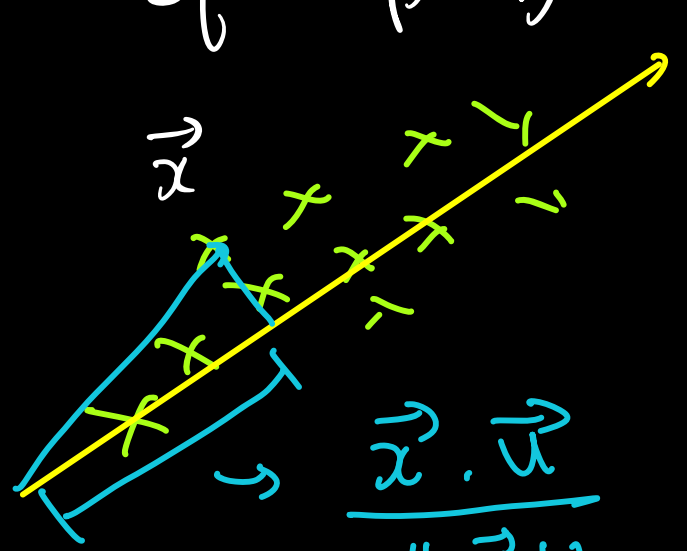
→ max length of projection





$$var(PC1) > var(JL1') > var(f_2') > var(PC2)$$

We can also think in terms of length of projection:



$$\frac{\vec{x} \cdot \vec{u}}{\|\vec{u}\|}$$

Best $\vec{v}$ will be where

$$\max_{\vec{v}} \sum_{i=1}^{n} \frac{\vec{v} \cdot \vec{x_i}}{\|\vec{v}\|} \cdot \frac{1}{n}$$

However, projections can also be -ve

$$\max_{\vec{v}} \sum_{i=1}^{n} \left| \frac{\vec{v} \cdot \vec{x_i}}{\|\vec{v}\|} \right| \frac{1}{n}$$ → But this is not differentiable

$$\max_{\vec{v}} \frac{1}{n} \sum_{i=1}^{n} \frac{(\vec{v} \cdot \vec{x_i})^2}{\|\vec{v}\|} \rightarrow \text{still hard to diff}$$

$$\downarrow$$

Constraint: $\|u\| = 1$

$$\downarrow$$

$$\max_{\vec{v}, \lambda} \frac{1}{n} \sum_{i=1}^{n} (\vec{u} \cdot x_i)^2 + \lambda(\|\vec{v}\| - 1)$$

Lagrangian loss function for P.C.A

This can be solved using gradient descent.

However there is also a direct

Linear algebra way.

Math Ahead!! → Not required to remember/reproduce

→ We will use some identities w/o proof

→ We will learn a new concept (digression)

Variance ↓

$$\Rightarrow \max_{\vec{v},\lambda} \frac{1}{n}\sum_{i=1}^{n}(\vec{u}\cdot\vec{x}_i)^2 + \lambda(\|\vec{v}\|-1)$$

called squared (homework)

$$\Rightarrow \max_{\vec{u},\lambda} \frac{1}{n}\sum_{i=1}^{n}(\vec{x}_i\cdot\vec{u})^2 + \lambda(\|u\|^2-1)$$

$\square \Rightarrow [\cdot] = \square$ → scalar

identity : $A^2 = A^T A = \|A\|^2$

if A is invertible square matrix

$$\rightarrow \max_{\vec{v}, \lambda} \frac{1}{n} \sum_{i=1}^{n} (x_i \cdot u)^T (x_i \cdot u) + \lambda (u^T u - 1)$$

identity $(AB)^T = B^T A^T$

$$\rightarrow \max_{\vec{v}, \lambda} \frac{1}{n} \sum_{i=1}^{n} u^T x^T x u + \lambda (u^T u - 1)$$

$$\rightarrow \max_{\vec{u}, \lambda} \frac{1}{n} u^T X^T X u + \lambda (u^T u - 1)$$

$$\rightarrow \max_{\vec{u},\lambda} \; u^T \frac{X^T X}{n} u + \lambda(u^T u - 1)$$

$$\rightarrow \max_{\vec{u},\lambda} \; u^T V u + \lambda(u^T u - 1)$$

V matrix :

$$\frac{X^T X}{n} = \frac{1}{n} \begin{bmatrix} \vdots & & \vdots \\ f_1 & \cdots & f_d \\ \vdots & & \vdots \end{bmatrix}_{d \times n} \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^T \quad \begin{matrix} X \\ \begin{bmatrix} f_1 & f_2 & \cdots & f_d \\ x_1 & & & \\ x_2 & & & \\ \vdots & & & \\ x_n & & & \end{bmatrix}_{n \times d} \end{matrix}$$

V = Covariance matrix

$$= \begin{bmatrix} \mathrm{var}(f_1) & \mathrm{cov}(f_1, f_2) & \cdots & \mathrm{cov}(f_1, f_d) \\ & \mathrm{var}(f_2) & \cdots & \vdots \\ & & & \mathrm{var}(f_d) \end{bmatrix}$$

Is This matrix has pairwise covering of all features [similar to pd.corr()]

So now we have

$$L = u^T V u + \lambda(u^T u) - \lambda$$

$$\frac{\partial L}{\partial u} = 2Vu + 2\lambda u$$

$$\frac{\partial L}{\partial \lambda} = u^T u - 1$$

Now we set both to $= 0$

$\therefore u^T u = 1$

identity

$\frac{\partial A^T B A}{\partial A} = 2BA$

$\frac{\partial A^T A}{\partial A} = 2A$

this is like

$\frac{\partial x \cdot x}{\partial x} = 2x$

but in matrix

$$Vu = -\lambda u$$

$$Vu = \lambda' u$$

matrix · vector · scalar → vector

Interesting

→ we have V, which is just pp.cov(X)

→ $\vec{v}$ must be such that when I multiply V to it, it is equivalent to multiplying some scalar to it.

Eigen Values and Vectors

For any matrix A, there exists* one or more vectors, such that

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

$\vec{x}$ → eigen vector

λ → eigen value

Geometrically

Lets say: $\vec{x} = \begin{bmatrix} 2 \\ 5 \end{bmatrix}$, $A = \begin{bmatrix} -6 & 3 \\ 4 & 5 \end{bmatrix}$

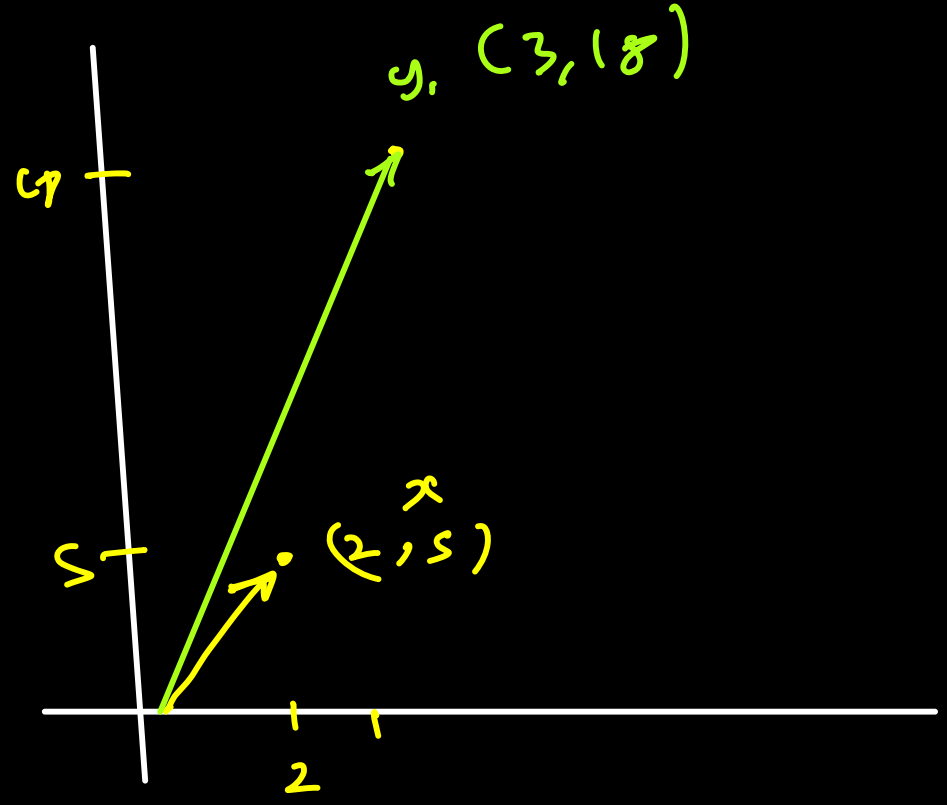
when we multiply $A\vec{x} = \vec{y}$, then

$\vec{y}$ has a magnitude and direction, but if the direction of $\vec{x}$ and $\vec{y}$ are same, then $\vec{x}$ is said to be an eigen vector of A with eigen value

$$\lambda = \frac{\|y\|}{\|x\|}$$

There can be multiple eigen vectors, which are always orthogonal (⊥) to each other.

Eg: $\begin{bmatrix} -6 & 3 \\ 4 & 5 \end{bmatrix} \begin{bmatrix} 2 \\ 5 \end{bmatrix} = \begin{bmatrix} -12 + 15 \\ 8 + 10 \end{bmatrix} = \begin{bmatrix} 3 \\ 18 \end{bmatrix}$

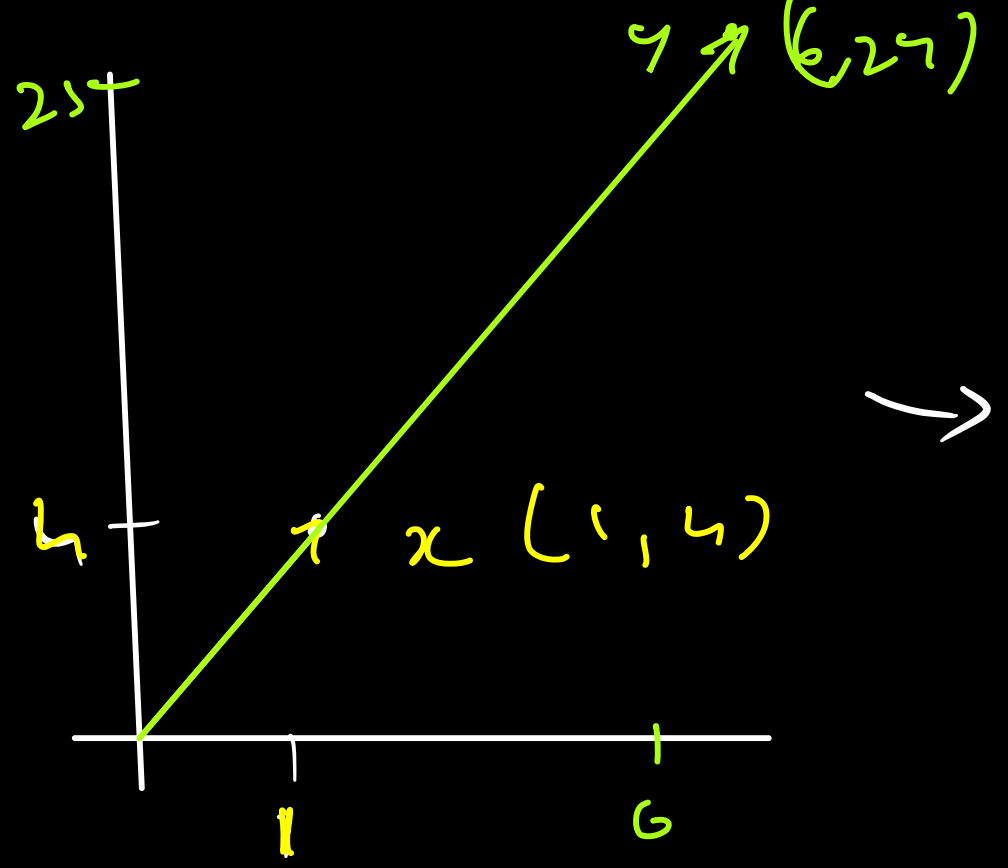


But, if $x = \begin{bmatrix} 1 \\ 4 \end{bmatrix}$

$$\begin{bmatrix} -6 & 3 \\ 4 & 5 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \end{bmatrix} = \begin{bmatrix} 6 \\ 24 \end{bmatrix} = 6 \begin{bmatrix} 1 \\ 4 \end{bmatrix}$$

eigen value

eigen vector



$y = \underline{6 \cdot x}$

→ x is such a vector that will not change direction when multiplied by 6

We can find eigen values and vectors very easily in numpy.

→ code