Till now

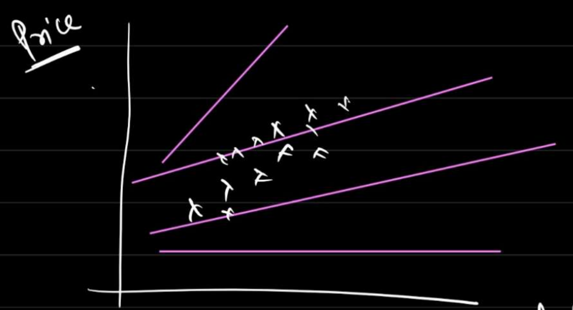
→ intuition of simple linear regression.

→ Error was least.

Agenda

↳ P.S in various forms

↳ implementation.



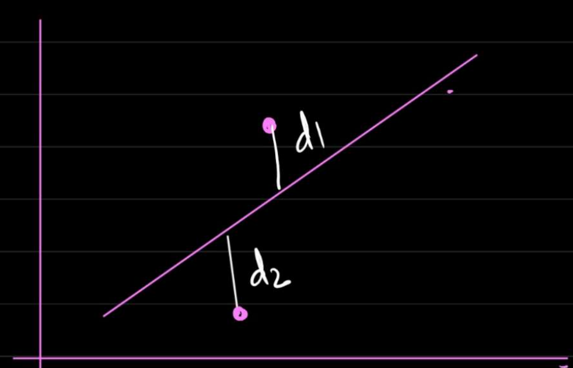
Best fit line

↓

minimise the error.

⇓

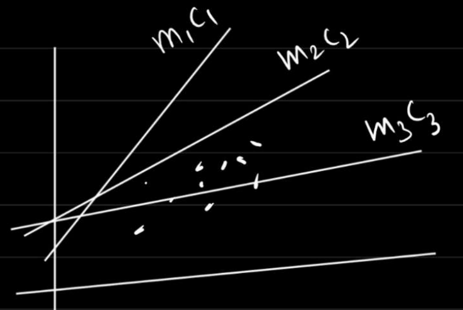
best representative of all the data points.



$$E = d_1 + d_2$$

(should be minimum)

$$T.E = \sum_{i=1}^{n} E_i$$



$$= \sum_{i=1}^{n} \left(y_{act} - y_{pred}\right)^2$$

↓

for least error

$$= \sum_{i=1}^{n} \left(y_i - mx_i - c\right)^2$$

or

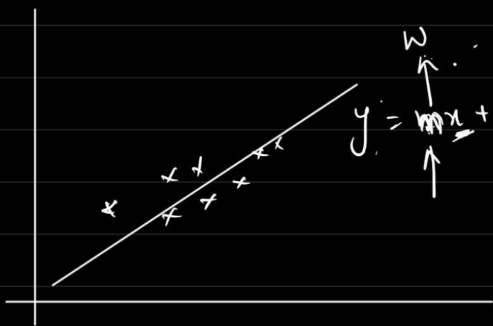
β_0 , β_1

$w \rightarrow$ weights , b - bias.

$\downarrow$

deep learning.

$\Rightarrow \begin{cases} w, & b \\ \theta_0, & \theta_1 \end{cases}$



$$y = mx + c$$

w above m, b above c

for least Error

$$E = \sum_{i=1}^{n} \left(y_i - mx_i - c \right)^2$$

$J(m, c)$

$J(\beta_0, \beta_1)$

$J(\theta_0, \theta_1)$

$J(w, b)$

$$E = \sum_{i=1}^{n} (y_i - mx_i - c)^2$$

Closed form soln.

iterative form solution

$\Downarrow$

Gradient Descent

$\rightarrow$ formulate a problem statement into mathematical equation.

$$ax^2 + bx + c = 0$$

$$\alpha, \beta = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$TE = \sum_{i=1}^{n} (y_i - mx_i - c)^2$$

→ m

→ c

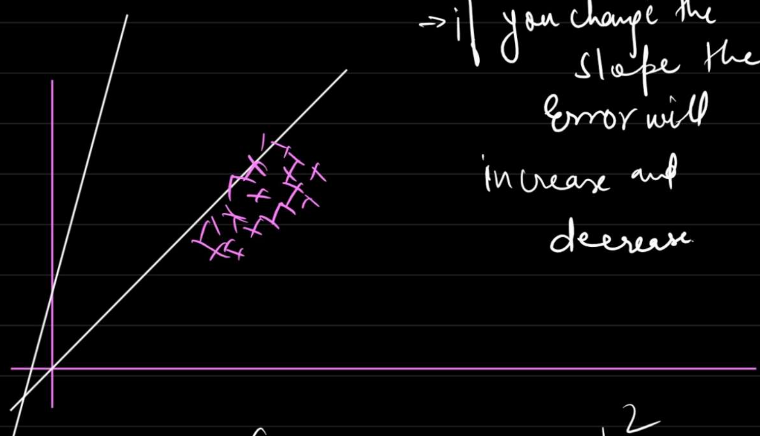
Scen 1

$y = mx + c$

$c = 0$

$y_{pred.} = mx$

→ if you change the slope the Error will increase and decrease



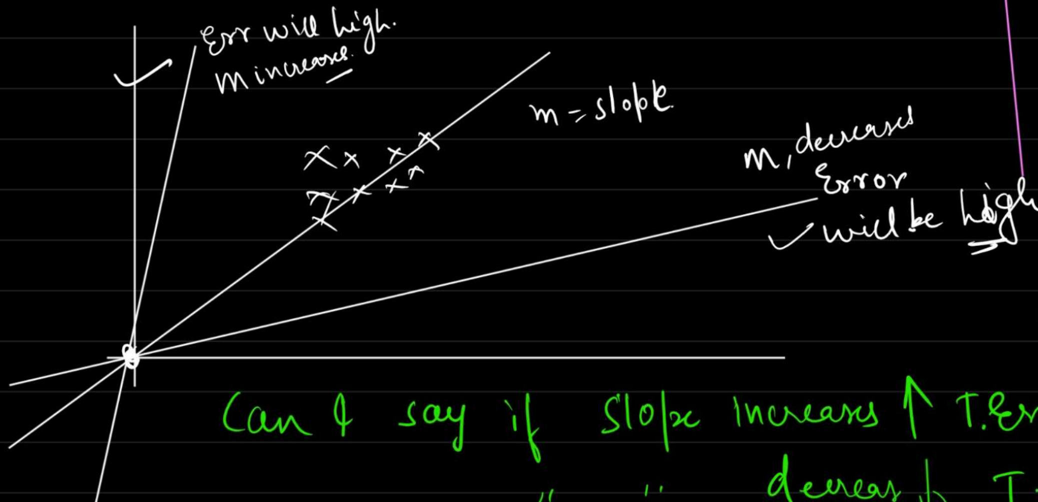
$$E = \sum (y_i - mx_i - c)^2$$

$c \rightarrow 0$

$$E(m) = \sum (y_i - mx_i)^2$$

$y = x^2$ ⟶ Quadratic. ∪

* Observation → if you increase m or decrease m, the error will change.



Can I say if slope increases ↑ T.Error ↑

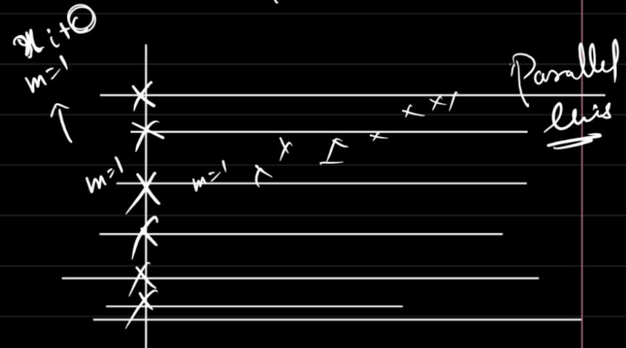
" " decrease ↓ T.Error ↑.

Scen 2

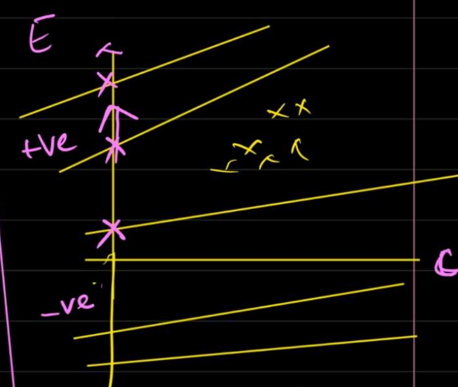
→ $m = 1$

$y = x + c$

$$E(c) = \sum_{i=1}^{n} (y_i - x_i - c)^2$$

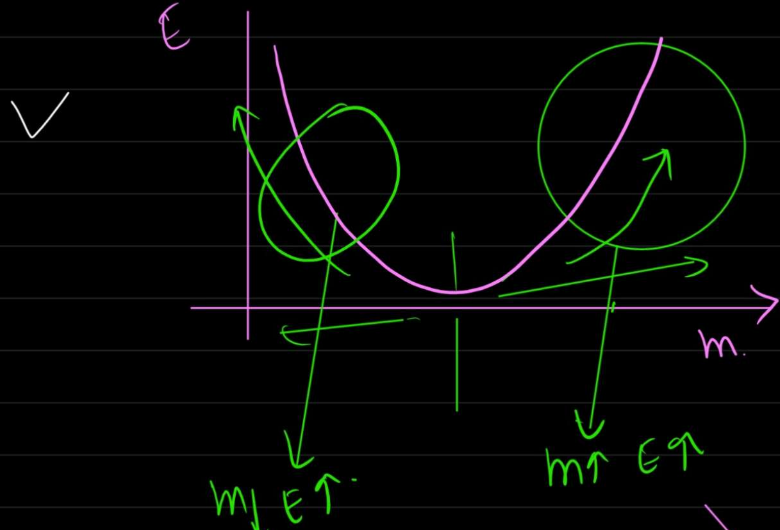


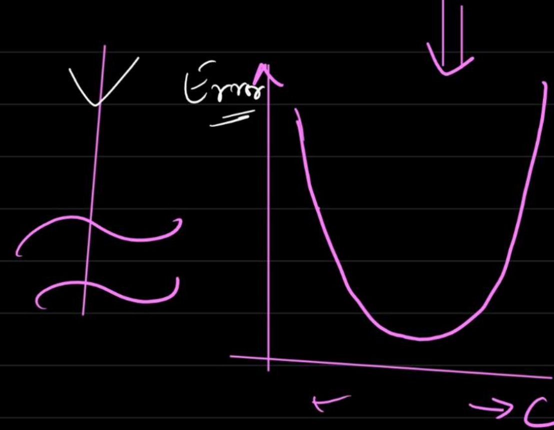
→ if c ↑ Error ↑

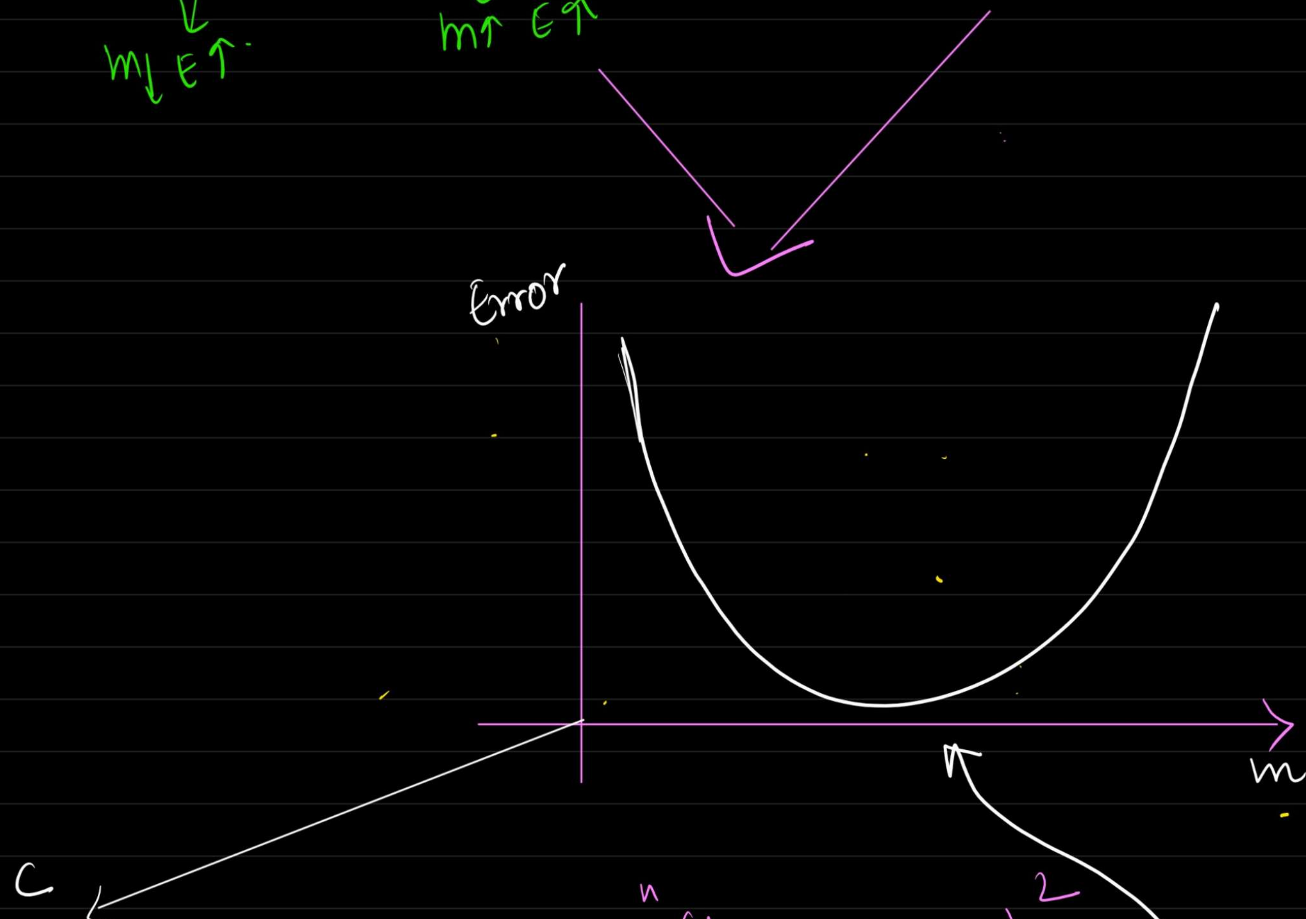


C ↑ E ↑

C ↓ E ↑







$$E = \sum_{i=1}^{n} \left(y_i - \underbrace{mx_i - c}\right)^2$$

$$y = x^2$$

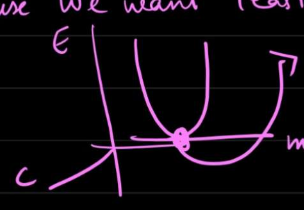
m, c

$$y = x^2 \longrightarrow \text{Parabola}$$

$$E = \sum_{i=1}^{n} \left(y_i - mx_i - c\right)^2$$

differentiation

→ because we want least error → $\frac{\partial E}{\partial m} = 0$



→ slope = 0 wrt m & c $\frac{\partial E}{\partial c} = 0$

$\frac{\partial E}{\partial c} = 0$ Why equating 0 ?

↓

Error should be least

↓

Slope = 0

$nx^{n-1} \leftarrow x^n$

$$\Rightarrow \frac{\partial}{\partial c} \sum_{i=1}^{n} (y_i - mx_i - c)^2$$

$$\frac{\partial y_i}{\partial c} - \frac{\partial mx_i}{\partial c} - \frac{\partial c}{\partial c} \rightarrow -1$$

$$\Rightarrow \sum_{i=1}^{n} 2(y_i - mx_i - c)^{2-1}(-1)$$

$$\Rightarrow \sum_{i=1}^{n} -2(y_i - mx_i - c)$$

$$\frac{\partial E}{\partial c} = 0$$

$$\Rightarrow \sum -2(y_i - mx_i - c) = 0$$

dividing by -2

$$\sum_{i=1}^{n} (y_i - mx_i - c) = 0$$

$$\sum_{i=1}^{n} y_i - \sum_{i=1}^{n} mx_i - \sum_{i=1}^{n} c = 0$$

divide by n (no. of DP's)

$$\frac{\sum_{i=1}^{n} y_i}{n} - \sum_{i=1}^{n} \frac{mx_i}{n} - \sum_{i=1}^{n} \frac{c}{n}$$

$c + c + c + c + c \dots n$

$$\bar{y} - m\bar{x} - \frac{\cancel{n}c}{\cancel{n}} = 0$$

$$\bar{y} - m\bar{x} - c = 0$$

$$\boxed{c = \bar{y} - m\bar{x}}$$

what is m ?

for m

$$\frac{\partial E}{\partial m} = 0$$

$$E = \sum_{i=1}^{n} (y_i - mx_i - c)^2$$

$$c \rightarrow \bar{y} - m\bar{x}$$

$$E = \sum (y_i - mx_i - \bar{y} + m\bar{x})^2$$

$$\frac{\partial E}{\partial m} = \sum \frac{\partial}{\partial m} (y_i - mx_i - \bar{y} + m\bar{x})^2$$

$$= \sum 2 (y_i - mx_i - \bar{y} + m\bar{x})(-x_i + \bar{x}) = 0$$

$$\Rightarrow$$

$$\Rightarrow -2 (y_i - mx_i - \bar{y} + m\bar{x})(x_i - \bar{x}) = 0$$

divide by -2 on both sides

$$\sum_{i=1}^{n} (y_i - mx_i - \bar{y} + m\bar{x})(x_i - \bar{x}) = 0$$

$$\sum_{i=1}^{n} \left((y_i - \bar{y}) - m(x_i - \bar{x})\right)(x_i - \bar{x}) = 0$$

$$\sum_{i=1}^{n} \left((y_i - \bar{y})(x_i - \bar{x}) - m(x_i - \bar{x})^2\right) = 0$$

$$\Rightarrow \sum (y_i - \bar{y})(x_i - \bar{x}) = m \sum (x_i - \bar{x})^2$$

$$m = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

if let's say multiple variable

x_1, x_2, x_3, x_4

$\Rightarrow \dfrac{\partial E}{\partial x_1}, \dfrac{\partial E}{\partial x_2}, \dfrac{\partial E}{\partial x_3}, \dfrac{\partial E}{\partial x_4}, \dfrac{\partial E}{\partial x_5} \Rightarrow$ Soln would be more complex

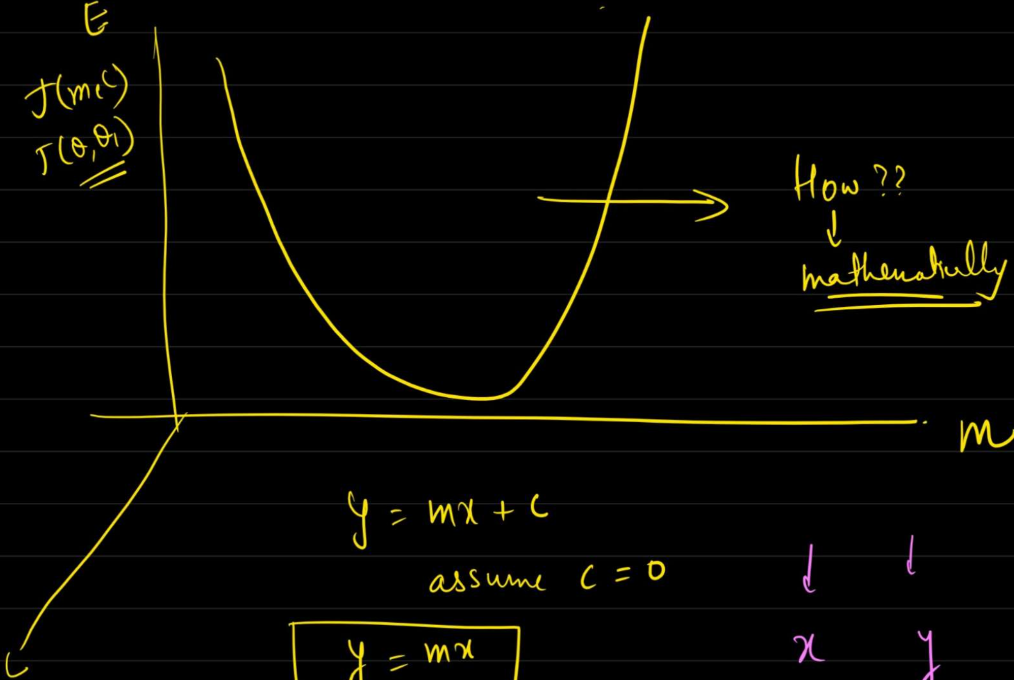
$\Downarrow$

iterative soln

$\Downarrow$

Gradient descent

$$y = \sum_{i=1}^{n} (y_i - mx_i - c)^2$$



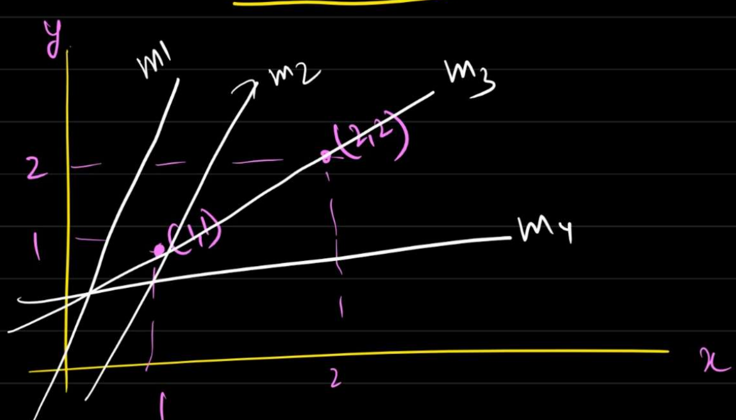
How ??
↓
mathematically

$y = mx + c$

assume $c = 0$

$\boxed{y = mx}$

$\downarrow$	$\downarrow$
x	y
1	1
2	2
3	3



$\boxed{y = mx}$

x	y
1	1
2	2
3	3

Scen-1

$m = 1$

$x = 1,\ y_{pred} = 1\times 1 + 0$
$y_{pred} = 1$

$x = 2,\ y_{pred} = 1\times 2 + 0$
$= 2$

$x = 3,\ y_{pred} = 1\times 3 + 0$
$= 3$

x	y	y_{pred}
1	1	1
2	2	2
3	3	3

$T.E = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_{pred})^2$

$T.E = \frac{1}{3}\left((1-1)^2 + (2-2)^2 + (3-3)^2\right)$

$\Downarrow$

$\boxed{J(m) = 0}$

Scen2

$m = 0.5$

$x = 1 \Rightarrow y_{pred} = 0.5\times 1$
$= 0.5$

$x = 2 \Rightarrow y_{pred} = 0.5\times 2$
$y_{pre} = 1$

$x = 3,\ y_{pred} = 0.5\times 3$
$= 1.5$

x	y	y_{pred}
1	1	0.5
2	2	1
3	3	1.5

$m = 0.5$
$c = 0$

$J(m) = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_{pred})^2$

$= \frac{1}{3}\left((1-0.5)^2 + (2-1)^2 + (3-1.5)^2\right)$

$= \frac{1}{3}(0.25 + 1 + 2.25)$

$= 1.16$

Scen-3

$m = 0$

$x_1 = 1,\ y_{pred} \Rightarrow 0\times 1$
$= 0$

$x_1 = 2\ \ y_{pre} = 0$

$x_1 = 3\ \ y_{pred} = 0$

$m = c$
$c = 0$

x	y	y_{pred}
1	1	0
2	2	0
3	3	0

$= \frac{1}{3}\left((1-0)^2 + (2-0)^2 + (3-0)^2\right)$

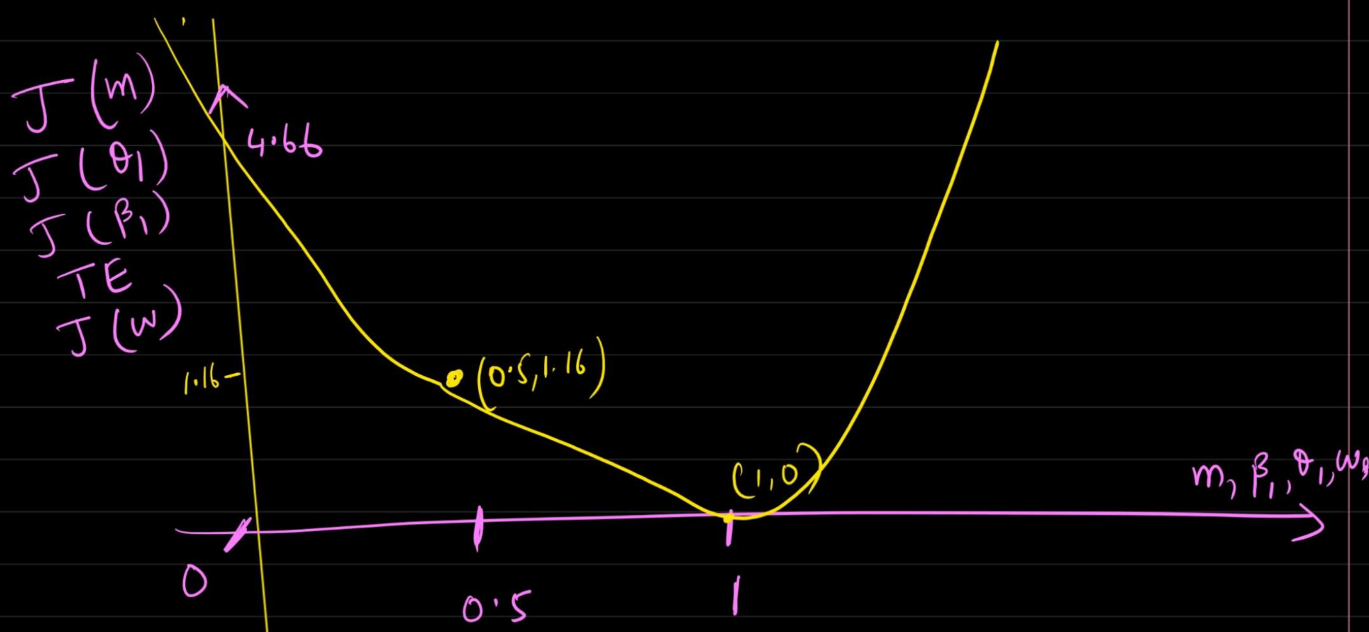
$= \frac{1}{3}(1+4+9) = 4.66$

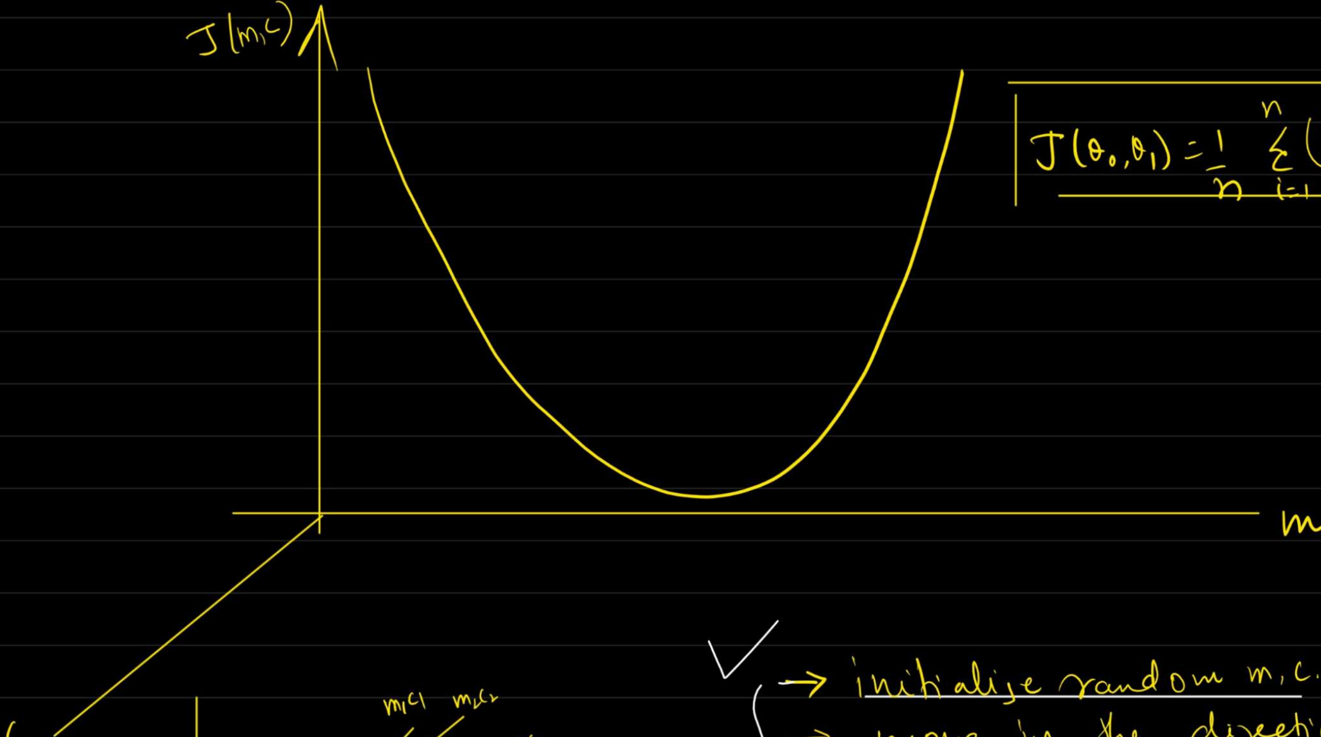
$J(m) = 4.66$

Total Error $= \text{MSE} = J(m, c)$

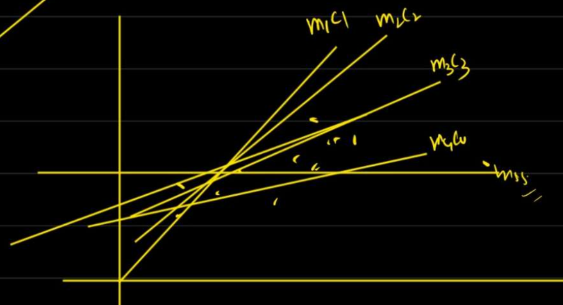
$\Downarrow$

$J(m)$

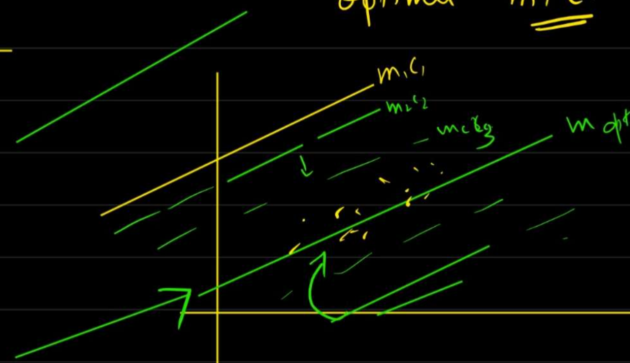




$$J(\theta_0,\theta_1) = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_{pred})^2$$



→ initialise random m, c.
→ move in the direction where error decreases
→ least error will be optimal m, c.



* Convergence Algorithm

→ keep making new best fit line in the direction of dp until Error (CF) is reduced as compared to previous line.

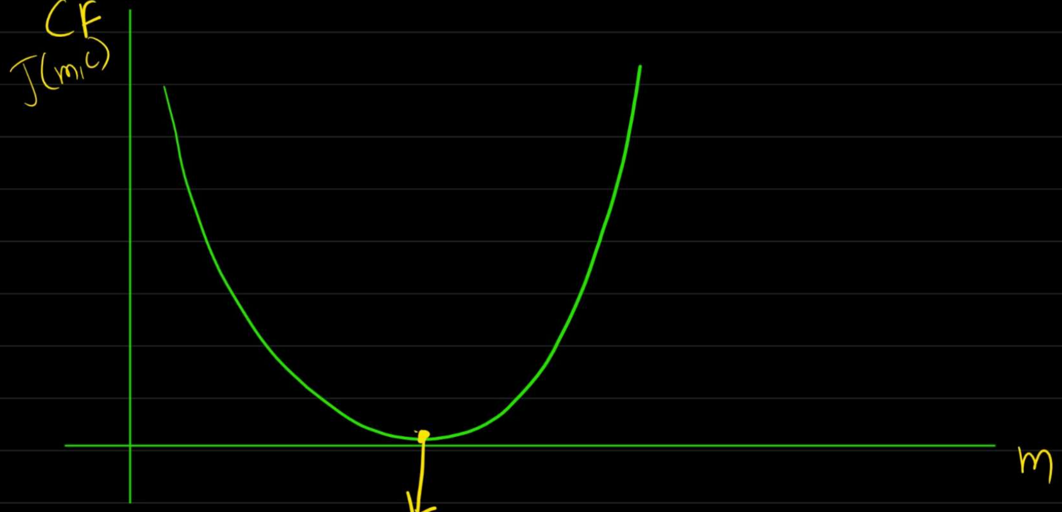
⇓

Till when ??

⇓

(where m, c is optimal

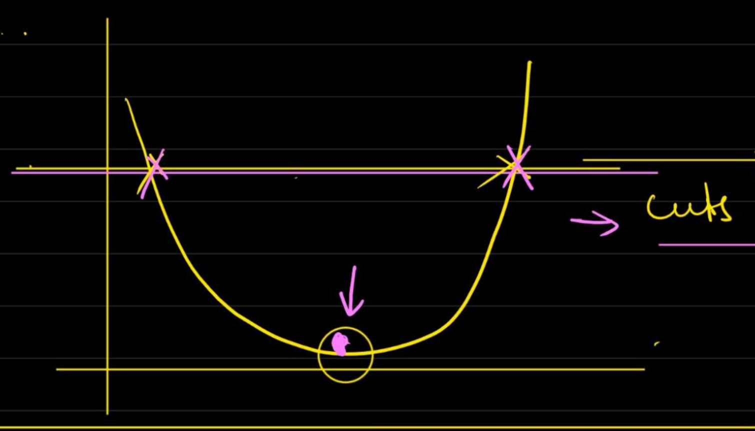
Error or is least
or
Cf is least



Cf is least

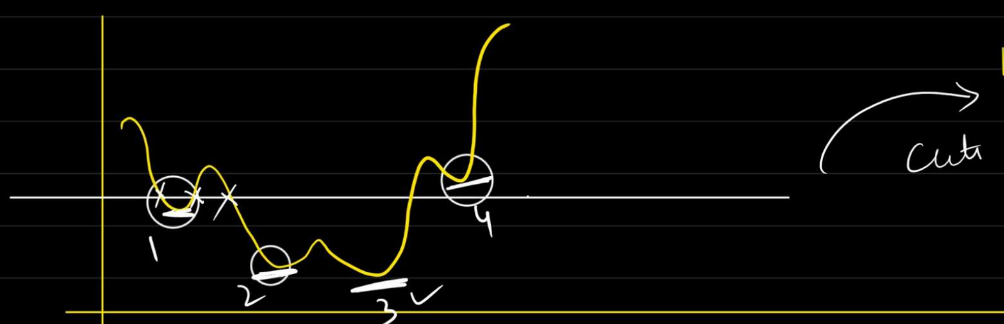
Point of minima → global minima

function.



Convex function.

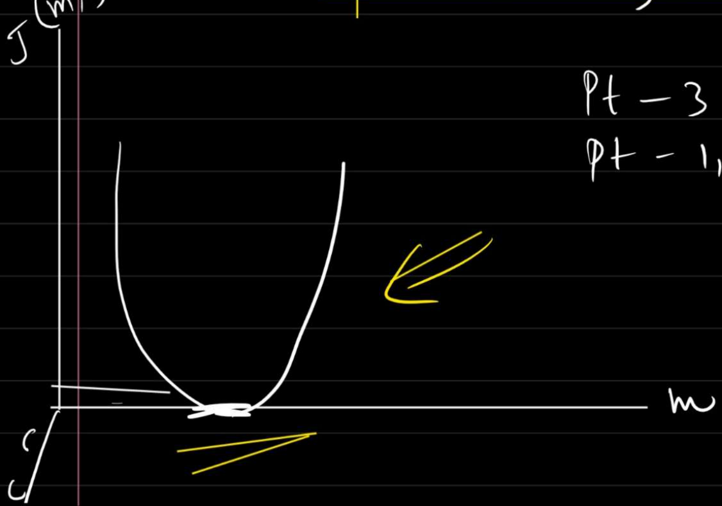
→ cuts at only two points
↳ convex
↓
one minima.



non-convex fn

cuts more than two points
⇓
Local, global minima.

Pt – 3 Global minima
Pt – 1,2,4 → local minima



* Convergence Algorithm

Repeat until convergence

$$m_{new} = m_{old} - \eta \frac{\partial(CF)}{\partial m_{old}}$$

$$C_{new} = C_{old} - \eta \frac{\partial(CF)}{\partial(C_{old})}$$

gradient descent

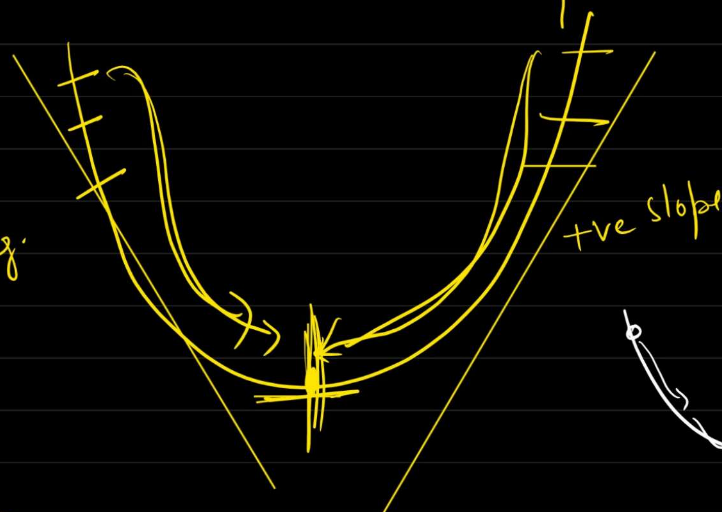
↓ slope

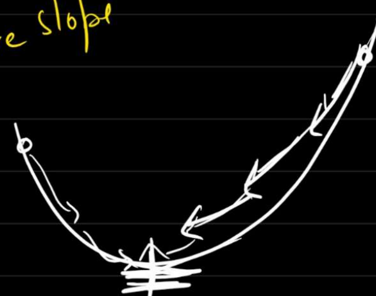
↓ Coming down.

⇒ slope guides for convergence

trekking

descend the slope

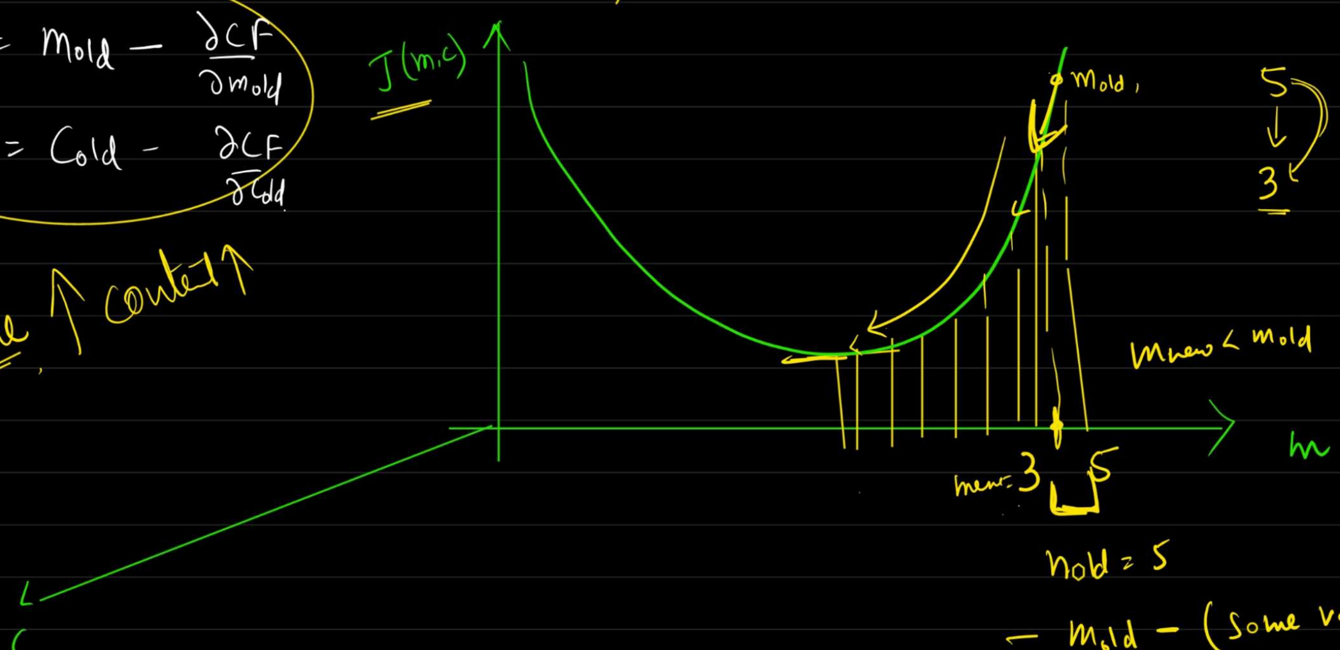




$$m_{new} = m_{old} - \frac{\partial CF}{\partial m_{old}}$$

$$C_{new} = C_{old} - \frac{\partial CF}{\partial C_{old}}$$

⇒ Time ↑ content ↑



5 ↓ 3

$n_{old} = 5$

$= m_{old} - (\text{some val})$

$5 - 2$

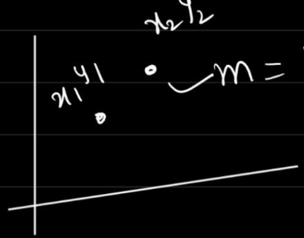
$m_{new} = 3$

$\frac{dy}{dx}$ = slope

$m_{new} = m_{old} -$ (some value)

⇒ $\frac{\partial CF}{\partial m_{old}}$ ⇒ slope

→ +ve value



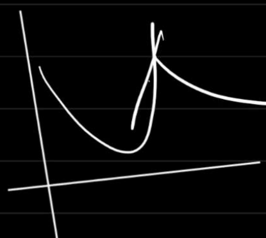
$m = \frac{y_2 - y_1}{x_2 - x_1}$

↓

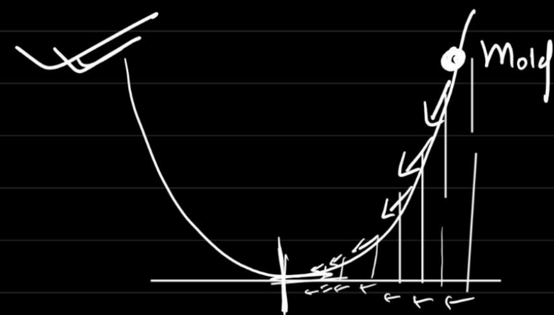
$\frac{\text{Change in } y}{\text{Change in } x}$

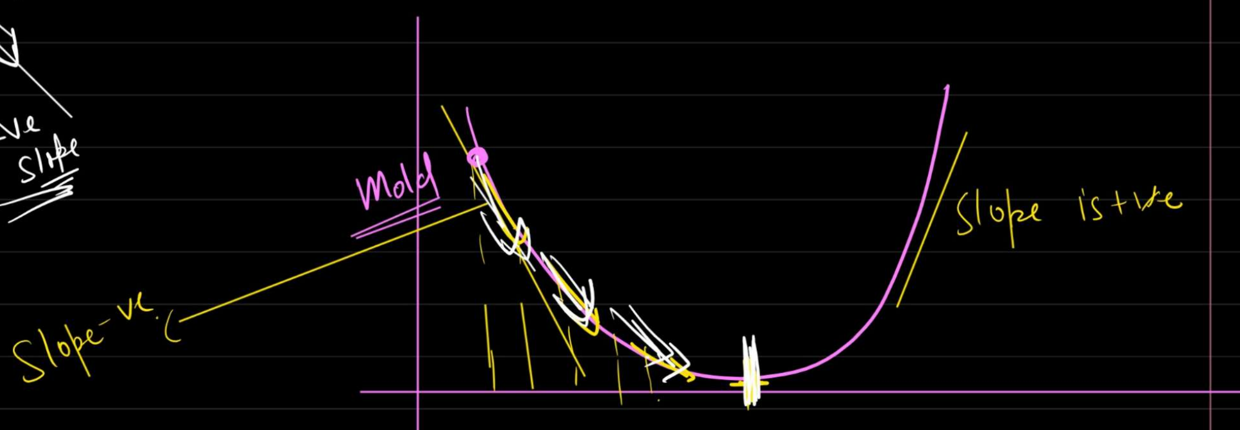
$m_{new} = m_{old} -$ (+ve val)

$m_{new} <<< m_{old}$



→ $\frac{dy}{dx}$ = slope = $\frac{\partial y}{\partial m_{old}}$



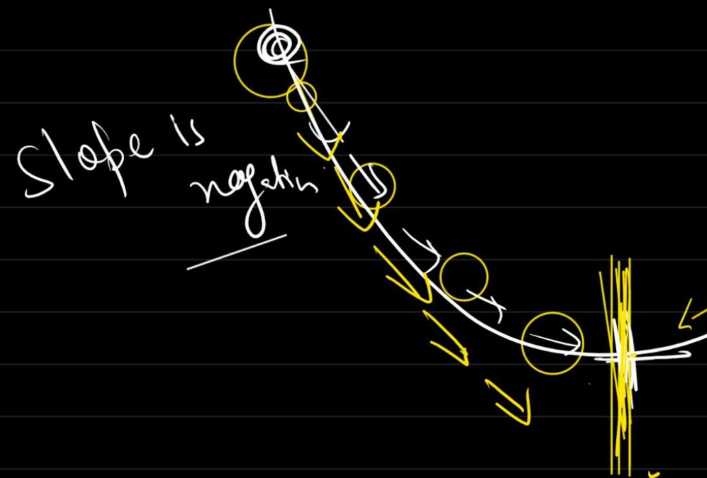


⇒ $m_{new} = m_{old} - \frac{\partial CF}{\partial m_{old}}$

→ $= m_{old} -$ (−ve slope)

$m_{new} >> m_{old}$

→ $m_{new} = m_{old}$ + ve value



When slope is positive

$$\begin{cases} m_{new} = m_{old} - \eta \dfrac{\partial CF}{\partial m_{old}} \\ C_{new} = C_{old} - \eta \dfrac{\partial CF}{\partial C_{old}} \end{cases}$$

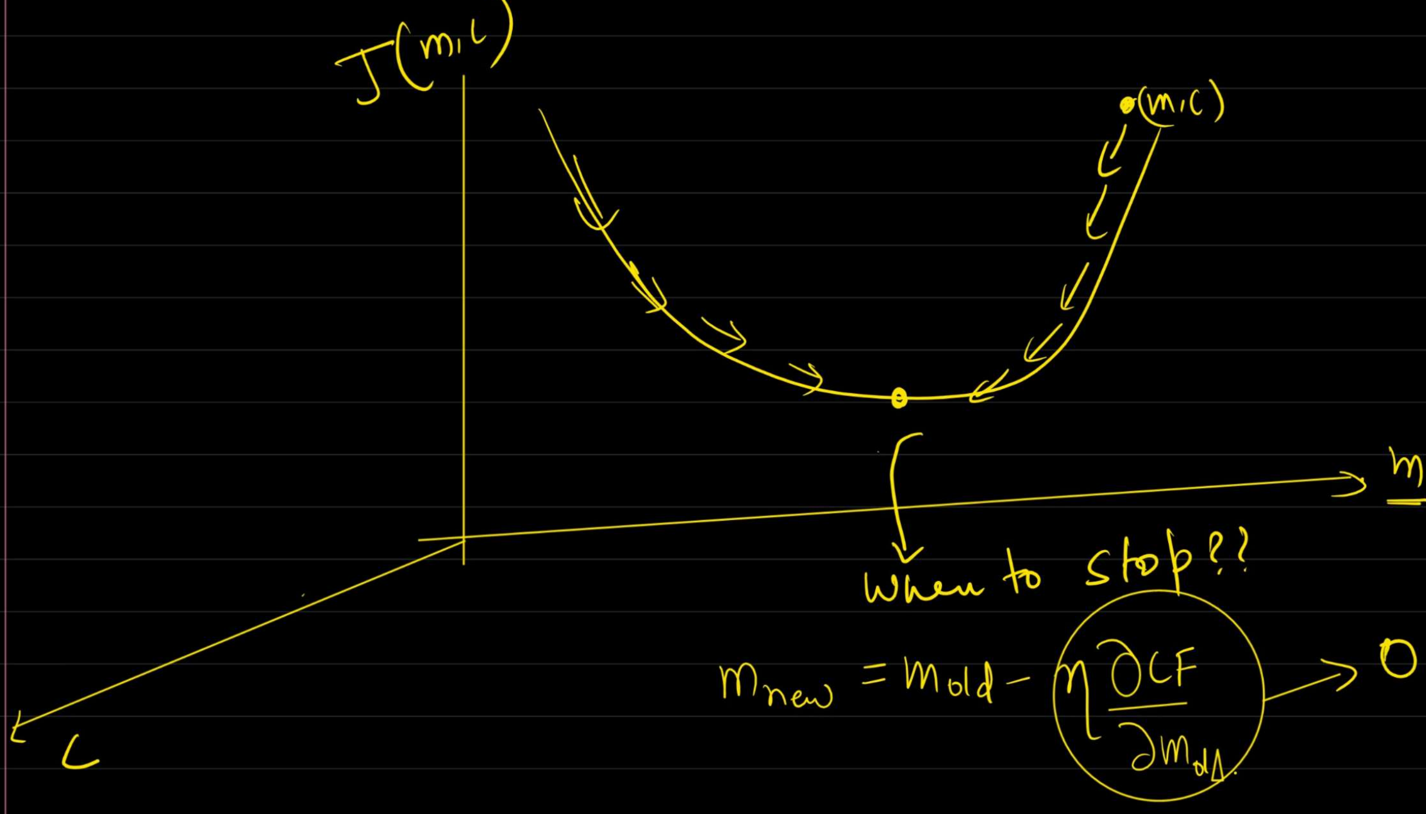
multithreading | multiprocessing
True multith | True multi
 | ||
 | Parallel Executi

Gradient desce

Gradien descent → PyH

equ —— ||
↓
m, c ⟷ ||
↓
Optimis ⟷ ||
Opt. ⟶ ||
⟵ eq ||

→ multiprocessing



When to stop??

$$m_{new} = m_{old} - \eta \frac{\partial CF}{\partial m_{old}} \rightarrow 0$$

$m_{new} \approx m_{old}$

$C_{new} \approx C_{old}$

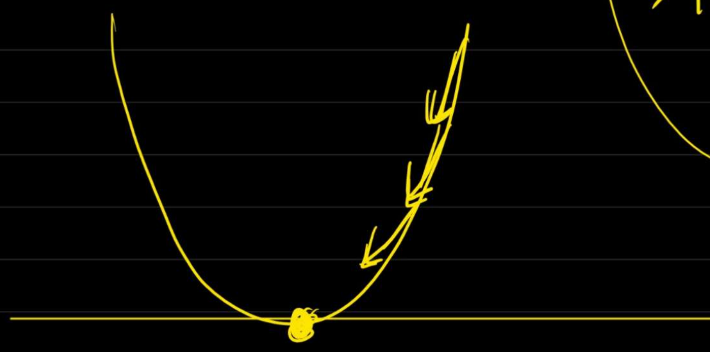
Gradient descent visualisation :-

blog.skz.dev gradient-descent

$$m_{new} = m_{old} - \eta \frac{\partial CF}{\partial m_{old}}$$

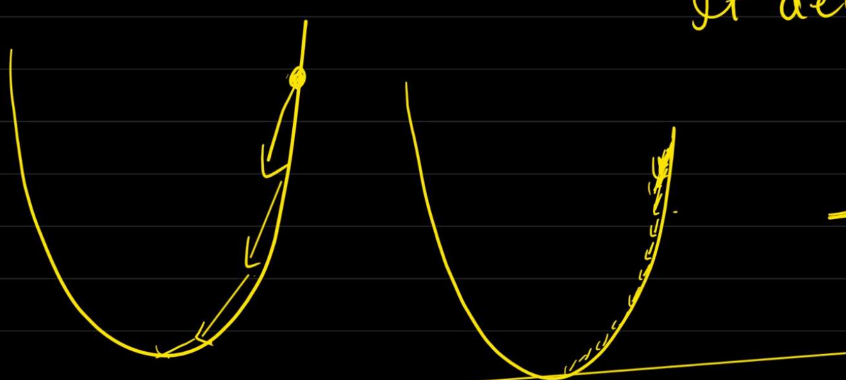
→ learning rate, α.

→ Eta (0.1 — 0.001)



η - learning rate

It decides the convergence Speed.



$$- \boxed{\eta \frac{\partial (CF)}{\partial m_{old}}}$$

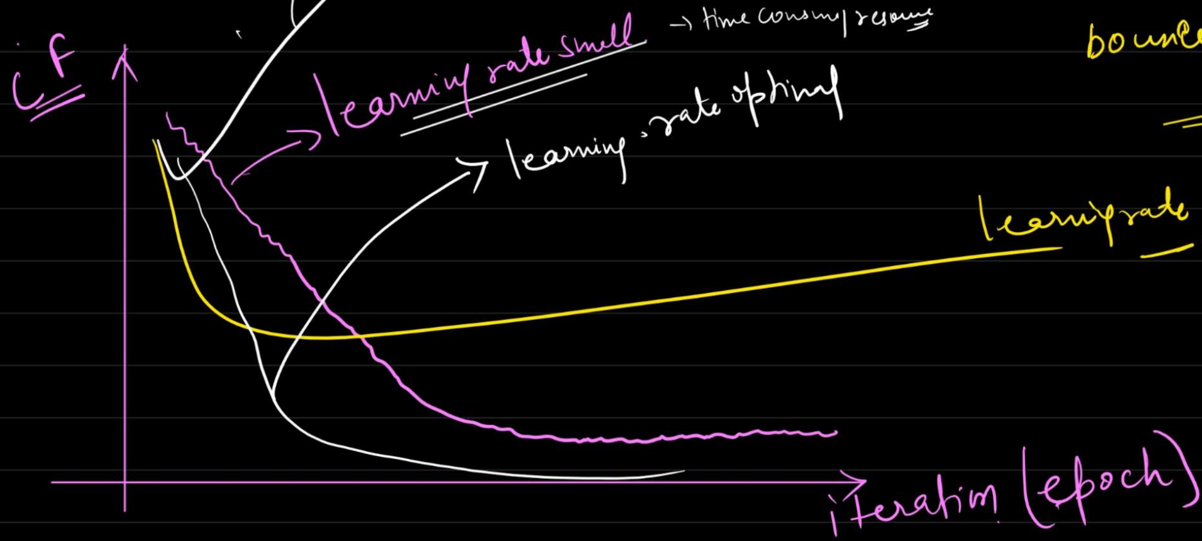
η → too small
very slow
for convergence.

η - too high.
→ Exploding gradient problem.
⇓
Will never converge,

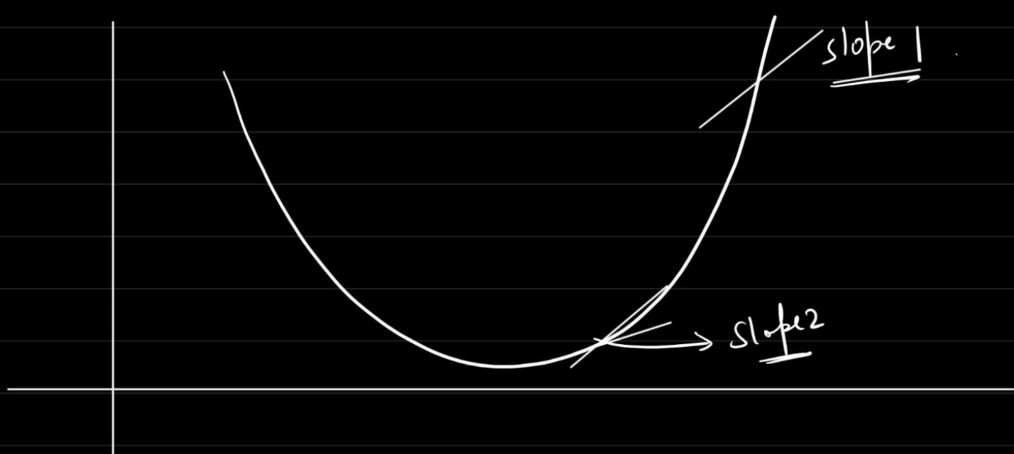
high learning

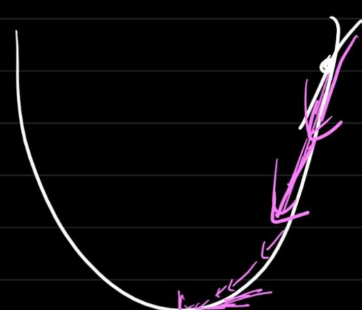
0.1 to 0.01

→ very high learning rate.



bounce to & fro.

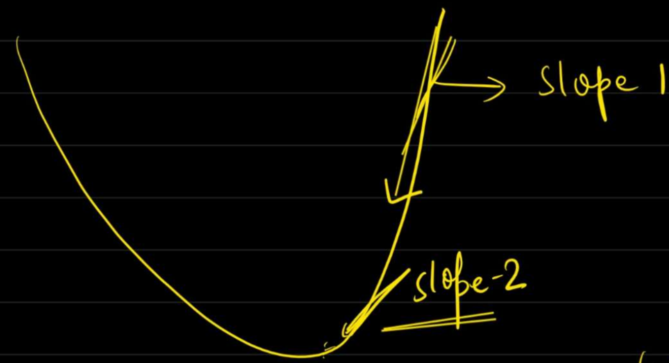




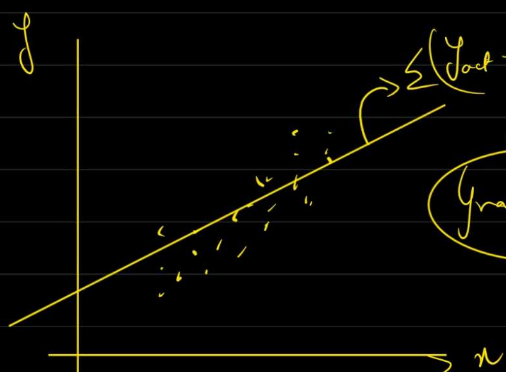
$$m_{new} = m_{old} - \eta \frac{\partial cf}{\partial m_{old}}$$

initially → slope will be higher, it will take big jump and while close to minima, slope decreases and steps become smaller.

developers.google.com/machine-learning/crash-course/fitter/graph



$-\eta$ (slope)



$\sum (y_{act} - y_{pr})^2$

Gradient Desc

.fit (X_{train}, Y_{train})

$\Rightarrow$ β_0, β_1

To predict on test

r square