Gradient Descent

* Optimizer

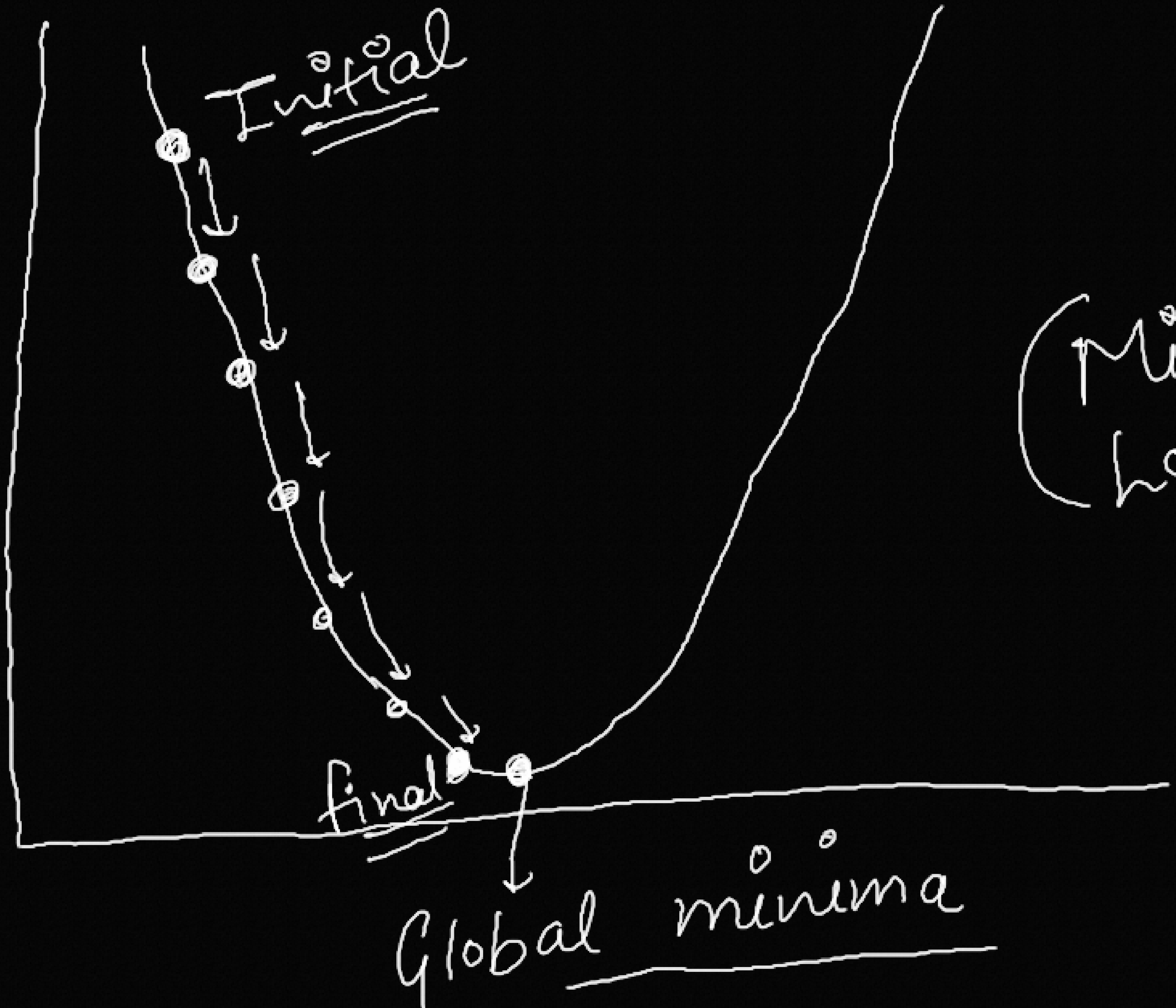
$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 \; - \, \cdot \, \cdot \, -$$

Coefficient | weights
(To find best values)
To minimize loss

$Y_{actual} - Y_{predic}$

Actual = 5
Predicted = 4.8
Loss = 0.2

General



(Minimize Loss)

Mathematical - 1

$f(x) = (x-3)^2$
(y)

GD $\longrightarrow$ minimize our fn

we know, $x = 3$

$$y = (3-3)^2$$
$$= 0$$

Let see how GD works.

Step ① :- Initial guess,

Lets say our guess, $x = 0$.

Step ② : Calculate gradient (partial derivative w.r.t to coefficient (x))

$$f(x) = (x-3)^2$$

$$\frac{df}{dx} = 2(x-3) \longrightarrow \text{gradient}$$

Step ③ :- Update Rule

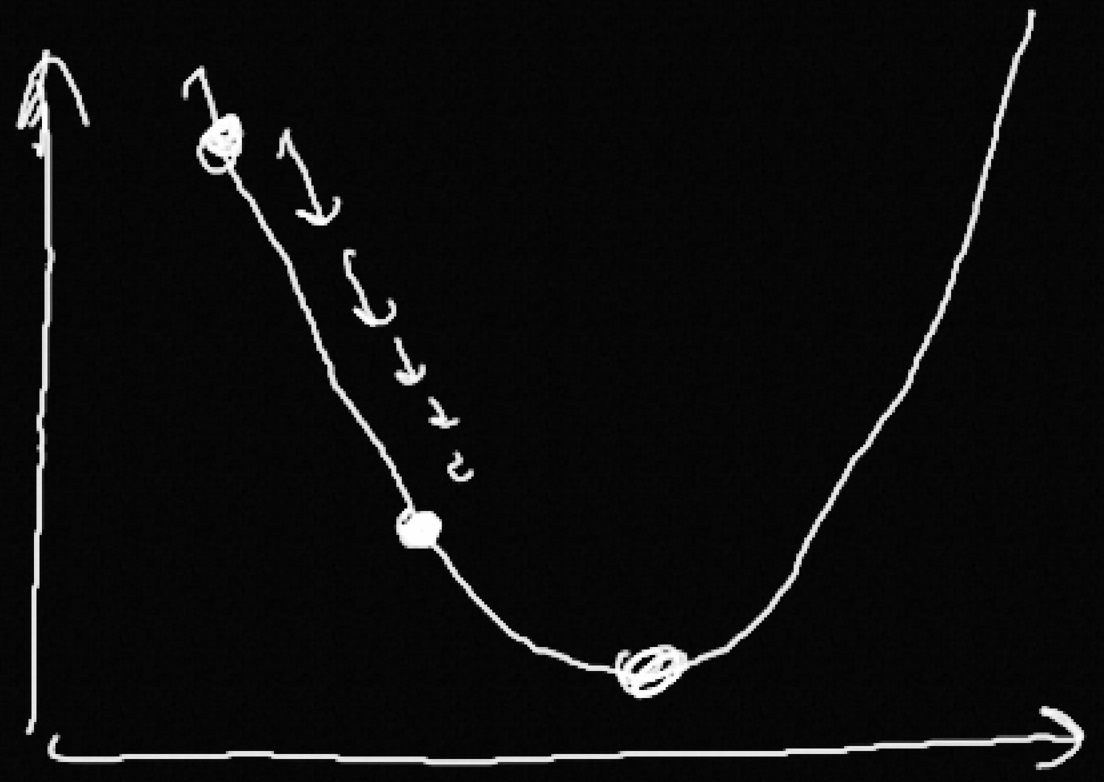
$$\boxed{x_{new} = x_{old} - \alpha \cdot \text{gradient}}$$

What is alpha (α)?

$\alpha \rightarrow$ small positive number generally (0.01, 0.001, 0.1

It should be optimall acc to data

If $\alpha \rightarrow$ too small $\rightarrow$ <u>Vanishing Gradient</u> problem
(0.000001)



It takes forever to reach global minima

$\alpha \rightarrow$ too large $\rightarrow$ <u>Exploding Gradient</u>
(10,100)

It will jump here & there but will not reach global minima.

We select $\alpha = \underline{\underline{0.1}}$

④ Repeat process untill we minimize cost $\underline{\underline{fn}}$.

Loss $= \underline{\underline{3}}$

$x = 0$

$\underline{\text{Iteration}}$ $\underline{\underline{1}}$. :- $\dfrac{df}{dx} = 2(x-3) = 2(0-3) = \underline{\underline{-6}}$.

$x_{new} = 0 - 0.1 \times (-6)$

$= \underline{\underline{0.6}}$.

Loss $= \underline{\underline{2.4}}$

Iteration 2 $\Rightarrow$ $x = 0.6$

$$\frac{df}{dx} = 2(x-3) = 2(0.6-3) = -4.8$$

$$x_{new} = 0.6 - 0.1 \times (-4.8)$$

$$= 1.08$$

Loss = 2

Iteration 3 $\Rightarrow$ $x = 1.08$

$$\frac{df}{dx} = -3.84$$

Loss = 1.5

$$x_{new} = 1.08 - 0.1(-3.84) = 1.464$$

Repeat thesse process until $x = 3$. or close to 3.

OR.

Your loss will be closer to zero.

ML Mathematics

Data

X_1	X_2	Y
1	2	5

We set $\alpha = 0.01$

Step 1 $\Rightarrow$ Linear model.

$$Y = \beta_1 X_1 + \beta_2 X_2$$

Our

GD start

Step ② Initial guess

$\beta_1 = 0$, $\beta_2 = 0$

Do prediction

$\hat{y}_1 = 0 \times 1 + 0 \times 2$

$= \underline{\underline{0}}$.

Actual $= 5$.

Loss $= 5 - 0$

$= \underline{\underline{5}}$.

Not good

Weight are not the best

Loss should be close to zero

Step ③ :- Calculate gradient

. β_1 , β_2 → two gradient

w.r.t. β_1 ⇒ $\dfrac{dL}{d\beta_1} = -\dfrac{2}{n} \sum (y - \hat{y}) \, x_1$

→ normalization factor

$= -\dfrac{2}{1} \times (5 - 0) \times 1$

$= -10$

w.r.t $\underline{\underline{B_2}}$ $\Rightarrow \dfrac{dL}{dB_2} = \dfrac{-2}{n} \sum (y-\hat{y}) x_2$

$= -2(5-0) \times 2$

$= \underline{\underline{-20}}$

$\underline{\underline{\text{Step 4}}}$ Update Rule

$B_1 \text{ new} = 0 - 0.01 \times (-10) = \underline{0.1}$

$B_2 \text{ new} = 0 - 0.01 \times (-20) = \underline{0.2}$

$B_1 = 0.1 \quad , \quad B_2 = 0.2 \longrightarrow$ <u>Repeat the process</u>

$B_1 = 0.1$, $B_2 = 0.2$

Do prediction

$$\hat{y} = 0.1 \times 1 + 0.2 \times 2$$
$$= \underline{0.5}$$

Actual = 5

$$\text{Loss} = 5 - 0.5$$
$$= \underline{\underline{4.5}}$$

{Now weight are improving}

Repeat till you get best values for $\underline{\underline{B_1}}$ & $\underline{\underline{B_2}}$

- Advantages

* Handle large datasets or complex models.
* Can help us find best values for our model

  $\downarrow$

  (coefficient/weights)

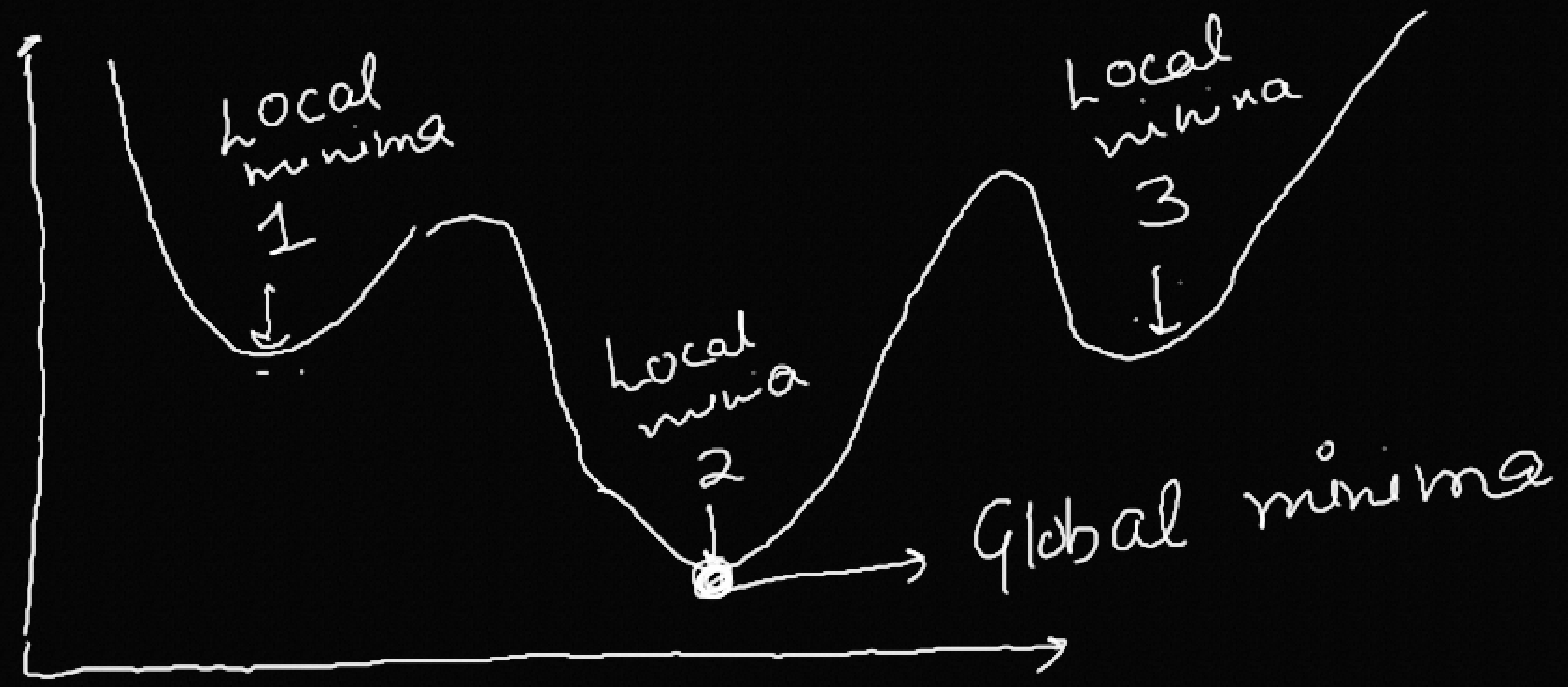
* Works well with high-dimensional data
* Best for convex functions.

  (Linear reg, Lasso reg, Ridge reg, etc
  Global minima is guaranteed)

- Disadvantages

* Not well with non- convex function
(Neural Networks (Deep Learning))

It converges to local minima instead of global minima.



To handle this problem $\longrightarrow$ SGD (Stochastic GD) , Adam , RMSProp

* find best α value $\longrightarrow$ It can be hectic.

<u>Soln</u> $\rightarrow$ <u>Adam Optimizer</u>

$\alpha \rightarrow$ learning rate $\Rightarrow$ Automatically adjut α value acc to data