GRADIENT DESCENT

An algorithm → set of instructions that work on a given data in a particular manner.

eg: for gradient descent

↳ Class Test — 50 marks

your friend scores — 'x'

→ now asks you to guess the marks

→ you say 40

he : No, not that intelligent

you : 30

he : No, more, coming close to the answer

you : 35

he : around

* In G.D., we start with a random guess, and slowly move to the correct / right answer

* By stepwise adjustment in values is the way to optimize parameters in GD.

+ Parameter optimization →

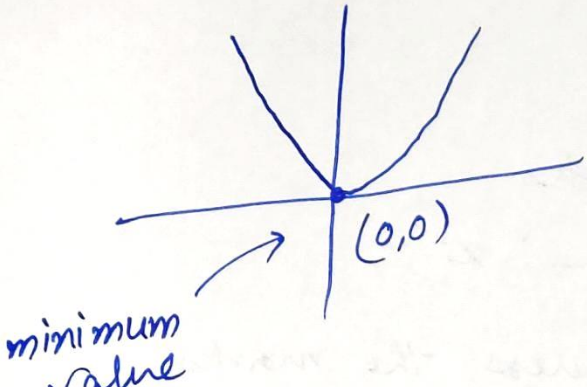
New Value = Old value − stepsize

↓

(known as learning rate × slope)

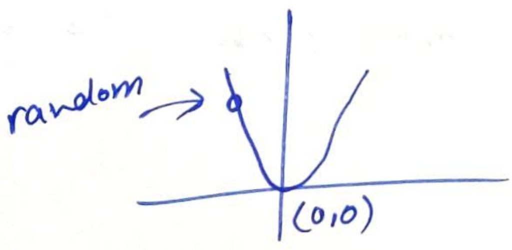
eg:→ $f(x) = x^2$ [consider this function]

↳ G.D. has to minimize this function and find minimum value of this function.



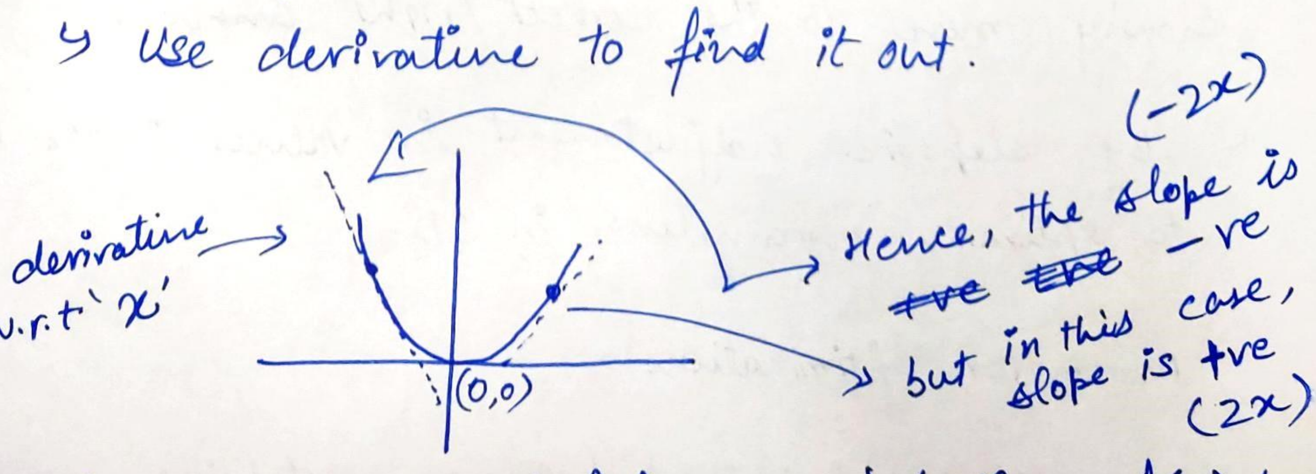
↳ How to reach this minimum value by using G.D.

↳ by making random guess



↳ So, now to find minimum value where we must move? upside or downside

↳ Use derivative to find it out.



If from this point I increase 'x' then f(x) decreases

So, correct movement would be to go downwards new value = old value − (learning rate × slope)

Case 1 → If slope is −ve
the 'new value' will be 'more' ~~added~~ to the 'old value'

Case 2 → If slope is +ve
the 'new value' will be 'less' than the 'old value'

Hence, your 'slope' decides the 'direction' to move for the parameter optimization to reach the 'global minima'

So, far we have discussed about $f(x) = x^2$, i.e. only one parameter, but if we have more parameters, how can we use G.D?

$$f(x, y) = x^2 + y^2$$

or, consider a regression model

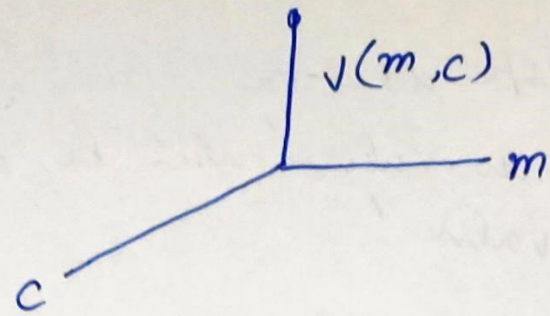
↳ i.e. $y = mx + c$

↳ cost function for this will be

$$J(m, c) = \Sigma(y_i - (mx_i + c))^2$$

Here, use of G.D. is to minimize this cost function, so your model fits the best way possible.

Here, we have 2 parameters m and c



Suppose, you have training data

(x)	(y)
1	2
3	4

Initial assumptions, $c=0$, $m=1$

$$J(m,c) = [2-(c+m.1)]^2 + [4-(c+m.3)]^2$$

Now, this needs to be minimized to find the best m and c combination

now, take derivative w.r.t 'c' using 'chain rule of the differentiation.

⇒ partial derivative of 'c' not 'm'

$$\frac{dJ}{dc} = -2[2-(c+m)] + [-2(4-(c+3m))]$$

$$= -2[2-(1)] + [-2(4-3)]$$

$$= -2[1] + [-2]$$

$$= -4$$

$$C_{new} = C_{old} - LR \times \underbrace{(-4)}_{\text{slope}}$$

Here, Learning rate (LR) is how rigorously you want to change your step. Means if you want to directly go to '30' or want to move from '40' '39' from '40' '38' ... so on...

$$C_{new} = C_{old} - \underbrace{LR}_{\text{aggressiveness to change value}} \times (-4)$$

$$= 0 - (0.001 \times (-4))$$

$$= 0.004$$

Similarly, do it for 'm' then change 'c & m' based on the 'new value' you get

So, there would be a point when the cost value will stop changing and will be close to 'zero' (0).

These will be Important for your neural network and deep learning models.

machine Learning + Deep learning models

classification & regression.

whenever, parameter optimization is required.