→ "Gradient descent Algorithm" ①

While learning "Maxima & Minima" there are examples where solving $\frac{df}{dx} = 0$ or $\nabla_x f = 0$ → In vector form

is not straight forward, because solving these equations could be tricky. Using these equations we want to get an optimal value of "x" or best value of "x".

When solving these equations is not easy, we have an alternative which is called "Gradient Descent Algorithm" which is an iterative algorithm & is very easy to implement in modern computers.

It works as :⇒

first we make a guess of what our 'x' is, with a random no. as 'x_0'

$$x_0 \longleftarrow \text{first guess of } x^*$$

↳ first guess of best "x"

becoz the problem we are solving here is

$$x^* = \underset{x}{\operatorname{argmin}}\ f(x)$$

So, A first guesses a random value of x as 'x_0'

then using the gradient-descent algorithm we move to a new value called x_1 then x_2 & so on, we keep on computing these values

$$x_0 \longleftarrow \text{first guess of } x^*$$
$$x_1 \longleftarrow \text{iteration 1}$$
$$x_2 \longleftarrow \text{iteration 2}$$

Eventually we will reach our Kth Iteration.
& value of 'x' at 'K' is very close to x^*

$'x_0' \leftarrow$ first guess of x^*

$'x_1' \leftarrow$ iteration 1

$'x_2' \leftarrow$ iteration 2

$\vdots$

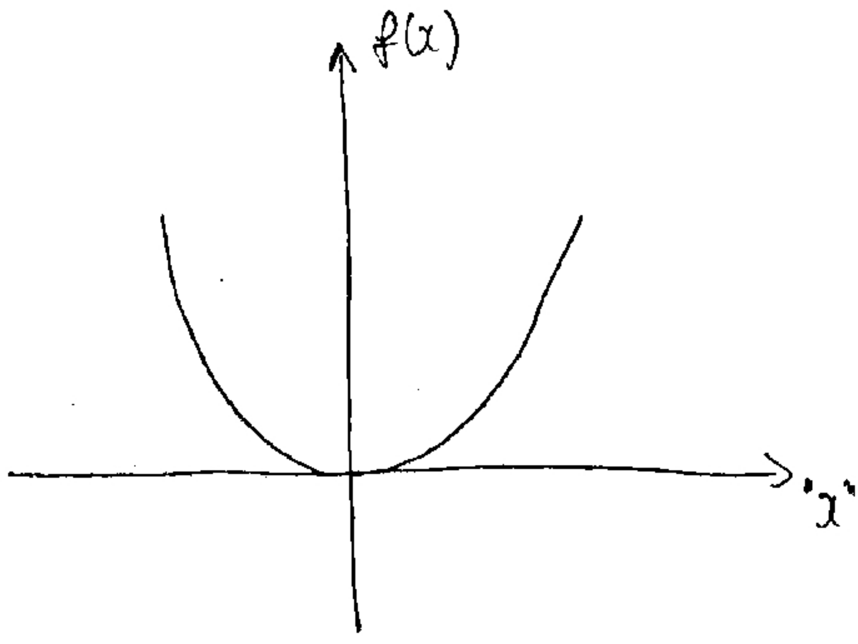
$\leftarrow x_k \leftarrow$ iteration K

This is very close to 'x^*'

So, in each iteration, we have to move closer & closer to $\boxed{x^*}$
This is our objective.

Now let's try to understand "Gradient Descent" from a geometrical perspective

let's have 'x' & f(x)" & a curve as shown below :-



Now for this curve, we want to find 'x^*' which minimizes f(x)

$$x^* = \underset{x}{\operatorname{argmin}}\ f(x)$$

Note :> Minimizing a function $f(x)$ is equivalent to maximizing $-f(x)$

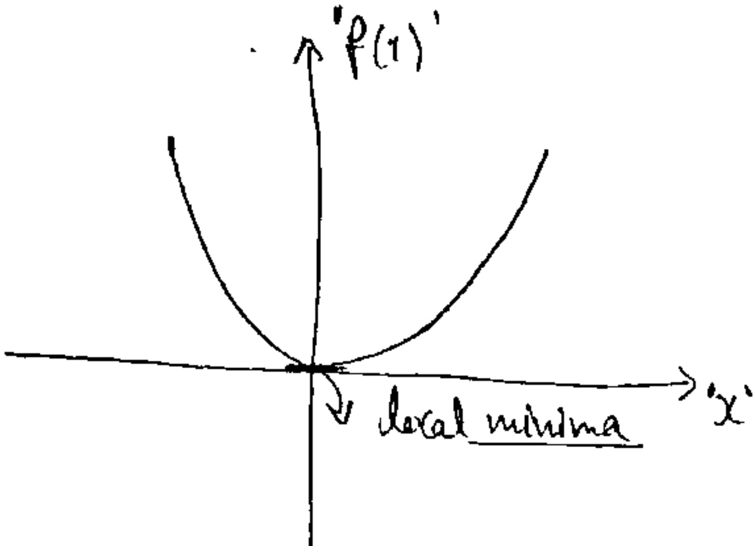
$\{ \ \min f(x) \cong \max -f(x) \ \}$

or

$\{ \ \max f(x) \cong \min -f(x) \ \}$

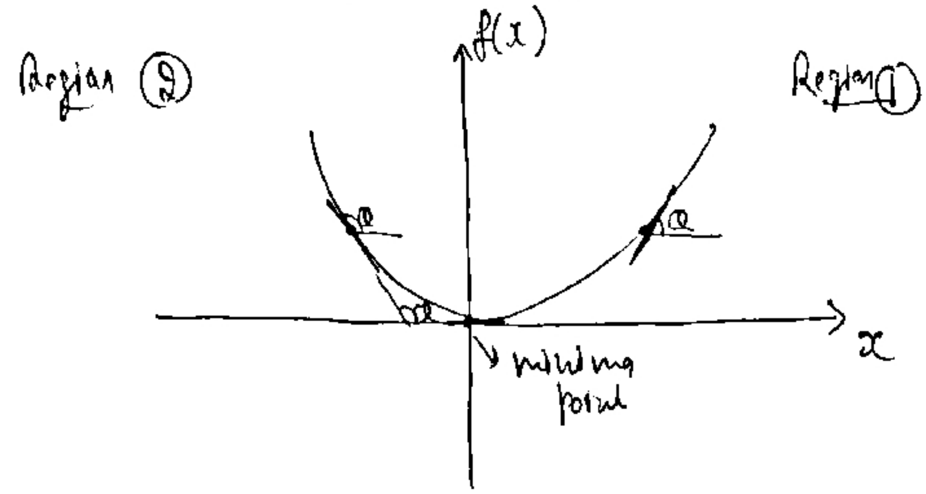
So if we learn for minima, we can use it for maxima by simply changing the sign of the function.

Let's now do it for minima, so geometrically we know that the minima for the curve lies as shown



Now let's understand the core geometric intuition behind gradient descent.

P.T.O



Take a point as shown above in region (1), the gradient is '+ive' or the slope is '+ive', becoz we take 'θ' + the slope of the line is $\tan\theta$', & θ lies b/w 0 & 90° so it is a +ive value.

Slope at minima is '0'. The slope on one side of minima is '+ive'.

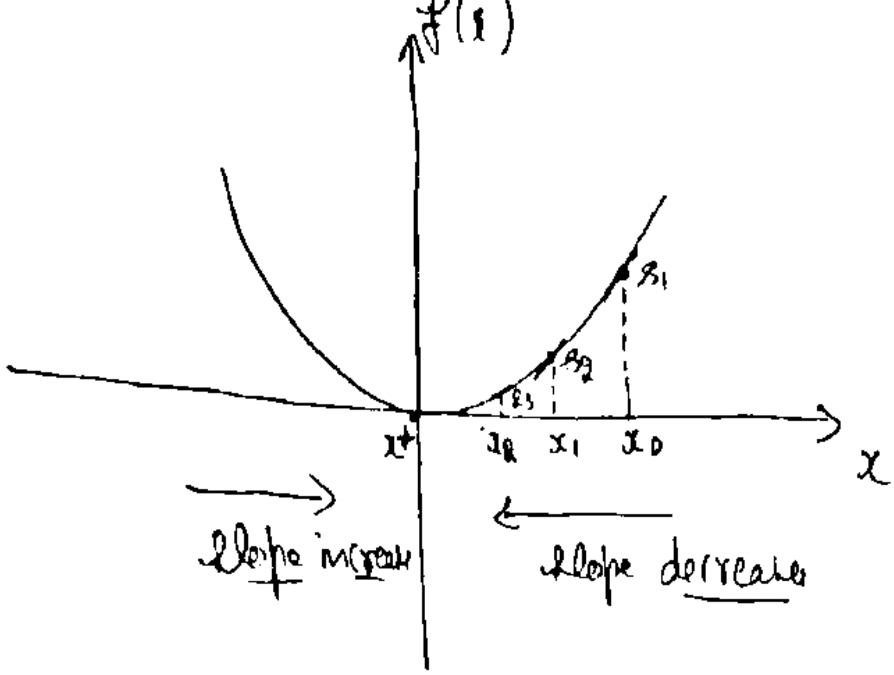
Lets look at the other side of minima. Let's take a point as shown above in region (2), the gradient or slope is '-ive' because the angle with 'x-axis' is 'θ' & which is greater than 90° & less than 180°.

∴ On one side of the 'minima', the slope is $\boxed{\text{'+ive'}}$ and on other side of the 'minima' the slope is $\boxed{\text{'-ive'}}$.

And exactly at 'minima' the slope is $\boxed{\text{'zero'}}$. which means the slope changes its sign from +ive to -ive at minima exactly.

The other interesting observation here is, Imagine we have the same plot, we draw earlier.

P.T.O

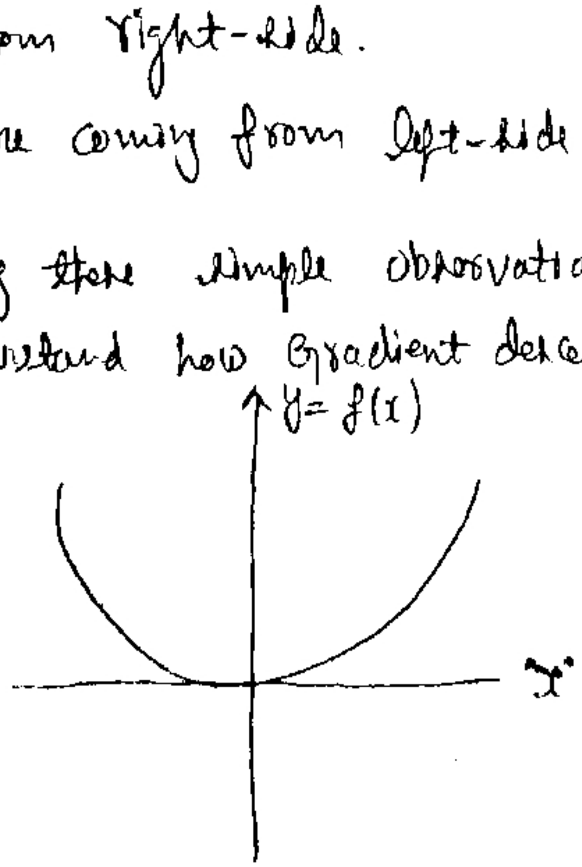


let's take a point as shown above & compute the slope & let's call that slope as "s1", similarly we compute the slope 's2' for a point as shown above, if the point corresponding to slope 's1' is 'x0' & the point corresponding to slope 's2' is 'x1' so on so forth let the optimal point be x^* as shown above.

So as we move closer to 'x^*' the slope reduces, if we are coming from right-side.

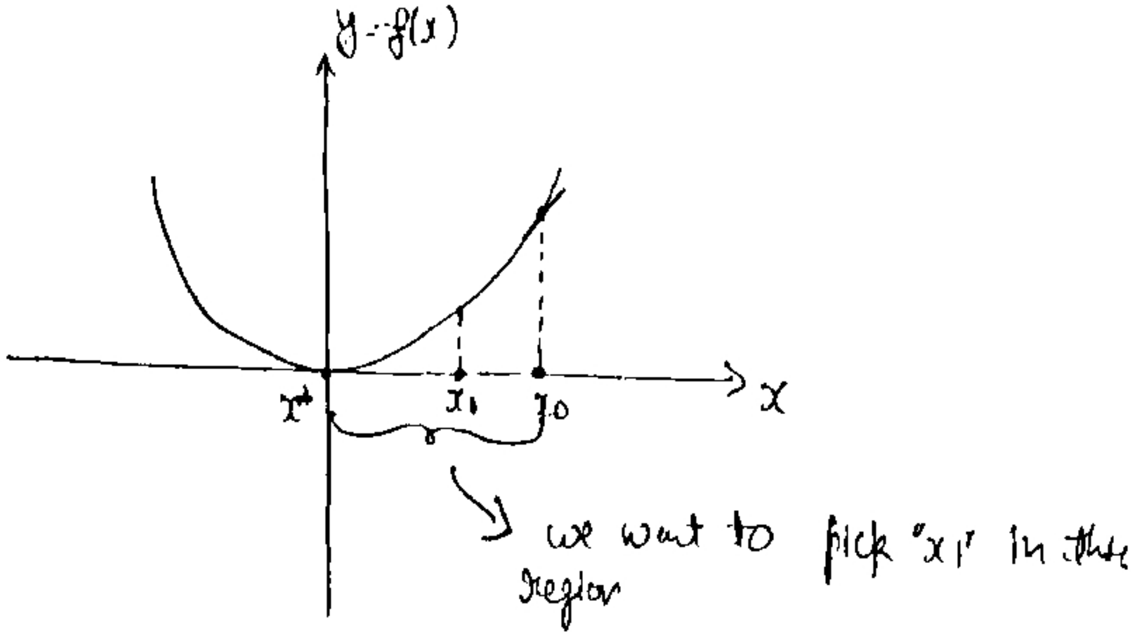
& if we are coming from left-side, the slope increases.

Now using these simple observations from geometry. let's now understand how Gradient descent actually works.



① Pick an initial point called 'x_0' at random
(we can pick it on any side of the graph)

let's start with an example where we pick a point (randomly.)
on the right side of minima.



② we want to find x_1, such that, "x_1" is closer to "x^*" than "x_0".

It is done as:

$$x_1 = x_0 - r\left[\frac{df}{dx}\right]_{x_0}$$

"r" is a constant & it is often called as a "step-size"
we will see later, as what happens when step-size changes.
for simplicity, till this point, let's say "$r = 1$".

$\left[\frac{df}{dx}\right]_{x_0}$ is basically the slope. & this slope is +ive

so what happens is

if we do $x_1 = x_0 - \gamma * \left[\frac{df}{dx}\right]_{x_0}$

Since slope is +ve

$$x_1 = x_0 - \gamma * (+ve)$$

Here $\gamma = 1$

$$x_1 = x_0 - 1 * (+ve\ value)$$

Which means we are subtracting something from "x_0"

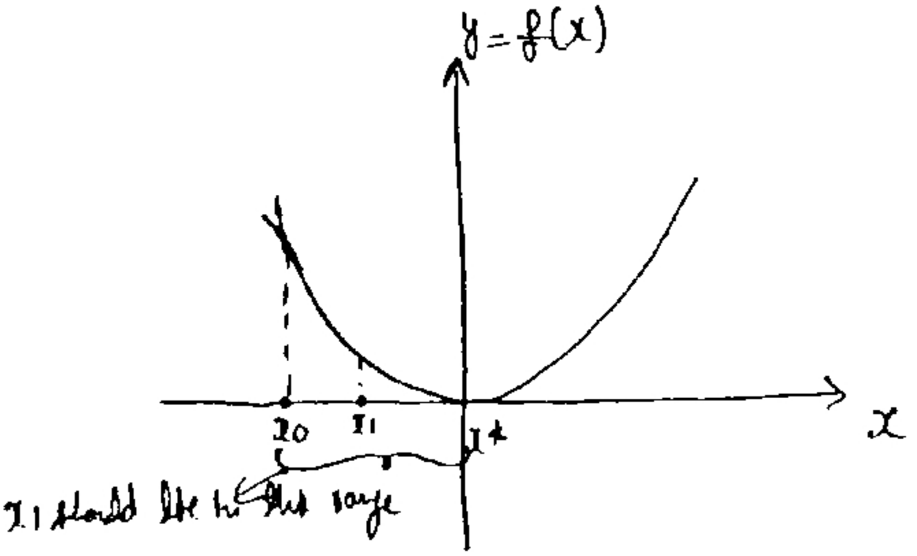
When we do that we are actually moving towards x^*

This implies that "$x_1 < x_0$"

& we are moving closer to x^*

x_1 is closer to x^* than x_0.

Now let's look at the other way around, what happens if x_0 is chosen on the other side of minima at random.



We know our update is

$$x_1 = x_0 - \gamma * \left[\frac{df}{dx}\right]_{x_0}$$

Now if we take derivative at x_0, we are going to get a -ve value

$$x_1 = x_0 - \gamma \left[\frac{df}{dx}\right]_{x_0}$$

it's a -ve value

$$x_1 = x_0 + 1 * (\text{some value})$$

$\therefore$ '$x_1 > x_0$'

Which means 'x_1' lies closer to x^* than 'x_0'.

So whether you pick your random point on left or right of the minima, it does not matter..

$\therefore$ In second step of gradient descent, we get 'x_1'

(3) let's now compute x_2

$$x_2 = x_1 - \gamma * \left[\frac{df}{dx}\right]_{x_1}$$

of 'x_2' lies still closer to 'x^*'

$\therefore$ at any iteration we do the following..

$$\left(x_{i+1} = x_i - \gamma \left[\frac{df}{dx}\right]_{x_i} \right)$$

This is the update function to reach minima.

So, this is the simple iterative algorithm.

$\therefore$ using this we get,

we start with 'x_0' randomly, we go to x_1, x_2, x_3 - - - - - x_p.

let's assume at some iteration 'K', we reach "x_k"

& $x_k = x_{k-1} - r\left(\frac{df}{dx}\right)_{x_{k-1}}$

Now once we reach 'x_k', we want to compute "x_{k+1}"

if $x_{k+1} - x_k$ is very very small

which means our 'x_k' has reached very close to 'x^*' & we are not going any further.

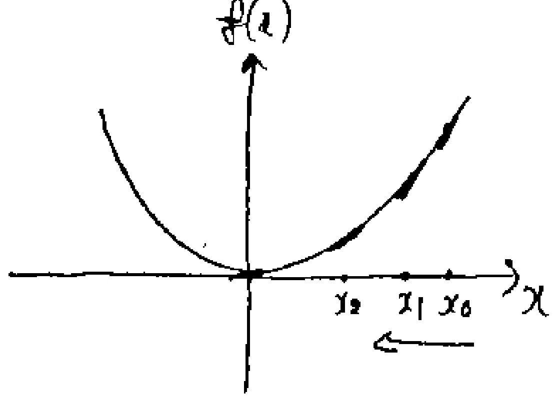
then we say terminate the loop & declare

$$\boxed{x^* = x_k}$$

Becoz we are making some progress at each iteration & we will stop at some point.

This is how Gradient Descent Algorithm works.

So, it is an iterative algorithm with simple update function



At every iteration slope reduces if we go from right side of the minima & eventually will become zero.. this means

$$\left(\frac{df}{dx}\right)_{x_0} \geq \left(\frac{df}{dx}\right)_{x_1} \geq \left(\frac{df}{dx}\right)_{x_2} - - - -$$

So, what is happening is, that gradient or slope is slowly reducing as we approach minima from right side.

Initially in gradient descent, we make a larger jump and as we come closer and closer to our solution our jump size also reduces.

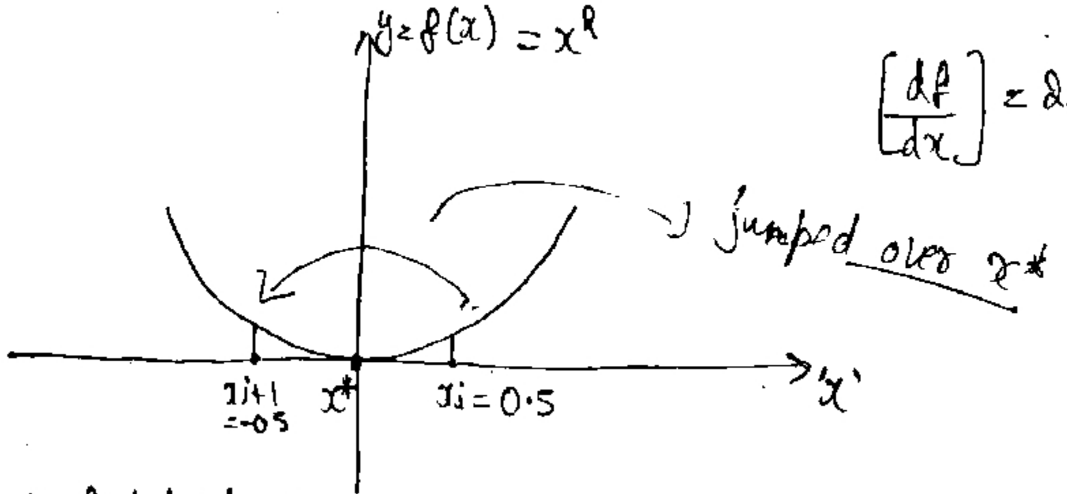
⇒ Learning rate or step size :-

In gradient descent, we have seen that

$$\left[x_i = x_{i-1} - r \left[\frac{df}{dx}\right]_{x_{i-1}} \right]$$ → This is often called as update equation.

Earlier we have seen that 'r' is kept constant, there is basically a problem with it.

Let's understand what problem it might create, if 'r' is kept constant.
let $r = 1$, let's take the equation of a parabola



let the first point $x_i = 0.5$ Now according to update equation

$$x_{i+1} = x_i - r\left[\frac{df}{dx}\right]_{x_i}$$

$$x_{i+1} = 0.5 - 1 * (2 * 0.5) = -0.5$$

∴ our x_{i+1} will cross into other region

We have moved from $x_i = 0.5$ to $x_{i+1} = -0.5$ & this is on the other side, but remember we should move closer to the x^*, & here we have jumped to the other side.

Here we simply jumped over x^*

Let's find out what is $\boxed{"x_{i+2}"}$ using the same update equation.

$$x_{i+2} = -0.5 - 1*(2*-0.5) = -0.5-1(-1) = 0.5$$

$$x_{i+2} = 0.5$$

Now our $x_{i+2} = 0.5$ again

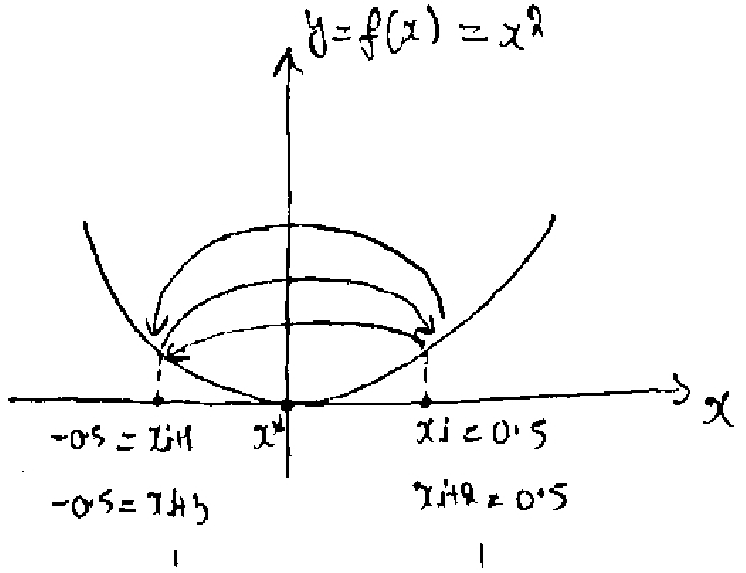
If we go like that then

$x_{i+3} = -0.5$

$x_{i+4} = 0.5$

& so-on

So, we are basically oscillating between $\boxed{"+0.5"}$ & $\boxed{"-0.5"}$



We are oscillating b/w +0.5 & −0.5

- This is happening, because 'γ' is kept constant at 'i'
Here we will never converge to x^* which is a problem
This problem is called an oscillation problem
& remedy to oscillation is, change 'γ' with each iteration

One technique to achieve this, is to reduce ['γ'] with each iteration.

'γ' becomes a function of iteration number

$$\gamma = h(i)$$

where 'i' is the iteration.

we can reduce 'γ' using some function.

such that as 'i' increase 'γ' should reduce

In Deep learning you will learn about, how to modify 'γ' more effectively.