Gradient Descent From Scratch

"Gradient Descent is a first-order iterative optimization Algorithm for finding a local minimum of a Differentiable function"

→ The idea is to take repeated steps in the opposite Direction of the Gradient (or approximate gradient) of the function at the current plot, because this is the direction of steepest descent. Conversely, stepping in direction of the gradient will lead to a local maximum of the function, the procedure is known as Gradient Ascent.

Higher Dimension → Closed form Solution (because Costly)

↳ New Approach → Gradient Descent

↳ Minima nikal ke de deta Hai for any Differentiable funcn.

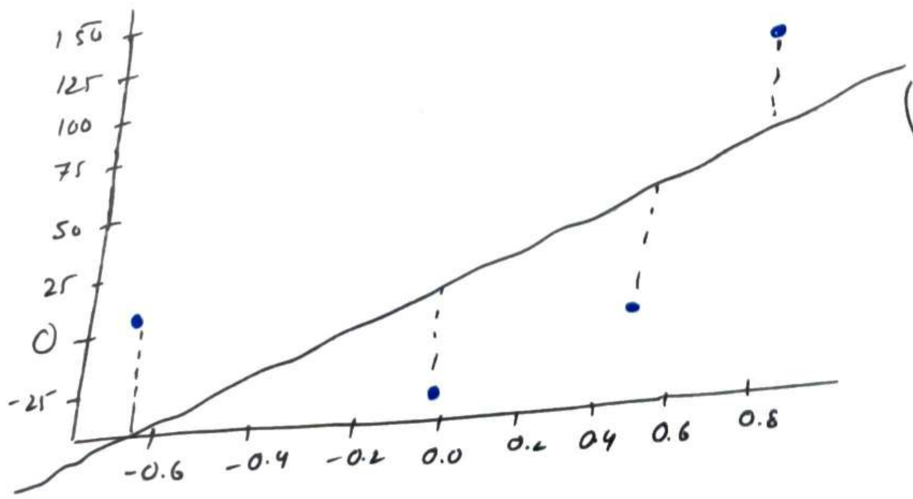
nutshell mai → just an Optimisation Algorithm.

Linear Regn
Logistic Regn Deep learning
t-sne **

Types
- → Batch GD
- → SGD → SGD Regressor
- → MBGD

Intuition (Here, understanding through lenses of LR but applicable on all)

→ 2column 4 rows.



cgpa / lpa of four student

→ want best fit line but not using OLS

$$\text{loss func } L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$\hat{y}_i = mx_i + b$$

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

$L(m,b)$

Consider m pata hai $m = 78.35$

$$L = \sum_{i=1}^{4} (y_i - 78.35x_i - b)^2$$

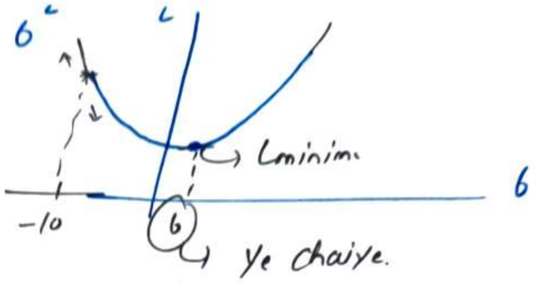
$L(b)$ → find b ka value taaki L minimum ho

Now $L \to b^2$ relation

GD

Step1 → Select a random bmin

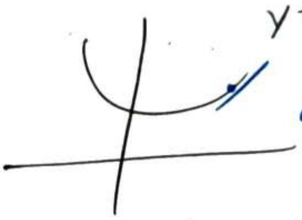
$b = (-10)$



* kisi bhi point mai kidhar jana hai - yahi trick hai GD ki.

* Slope nikalenge aur Slope ka Direction ye point karega ki hame kis Direction mai jana hai.

yaha pe slope hai



$y = x^2 + 2$

$x = 5$

$dy/dx = 2x = 10$

if ** Slope = (-ve) Aage jana hai b increment

Slope = (+ve) peeche jana b decrement

in short * $\boxed{b_{new} = b_{old} - \text{slope}}$ → idea is to take repeated steps in opp. direction of gradient.

inme bahut Drastic change ayenge toh

Ab kaha pe rukna Hai.

$$\boxed{b_{new} = b_{old} - \eta\, \text{slope}}$$

η → learning rate $\eta = 0.01$

→ to grow and gradually, slowly towards ans.

η (↑) can skip solution

η (↓) very slow towards solⁿ

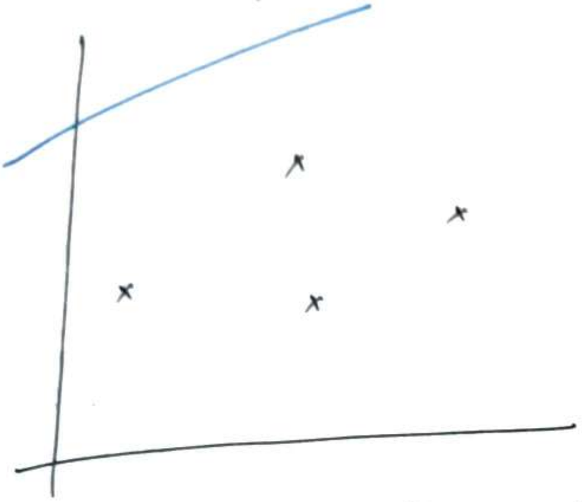
When to Stop ??

bnew - bold = 0.0001 ↳ koi fayda nahi hai increment ∴ Converged to sol^n

↳ 1) diff bold bnew => 0.0001

2) Iteration = 1000 ↳ expect sol^n mil jayega

↑ epochs



m = 78.35 (pata hai)
Sirf b mai Gradient Descent lagayenge.

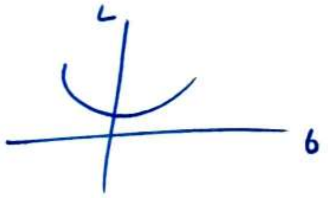
Step 1 → Start with random value of b (b=0)

$\boxed{\eta = 0.01}$

for i in epochs:

$$\boxed{b_{new} = b_{old} - \eta * slope}$$

Loss function $L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$



$$\frac{dL}{db} = \frac{d}{db}\left(\sum_{i=1}^{n} (y_i - mx_i - b)^2 \right)$$

$$\sum 2(y_i - mx_i - b)(-1)$$

$$slope = -2 \sum (y_i - mx_i - b)$$

$$\underline{b=0} \quad = -2 \sum (y_i - mx_i)$$

$$= -2 \sum_{i=1}^{n} (y_i - 78.35x_i)$$

slope at b=0 = ____ mil jayega

This term is called step size

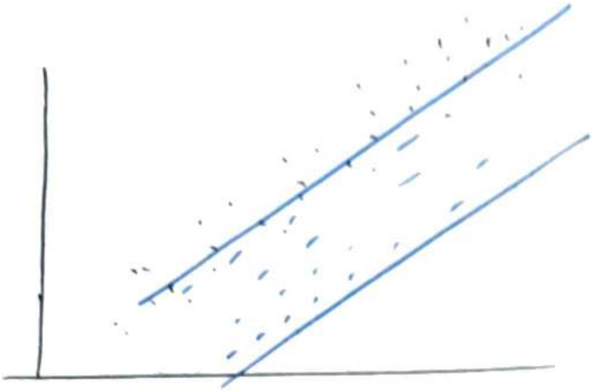
$$\boxed{b_{new} = b_{old} - \eta * slope}$$

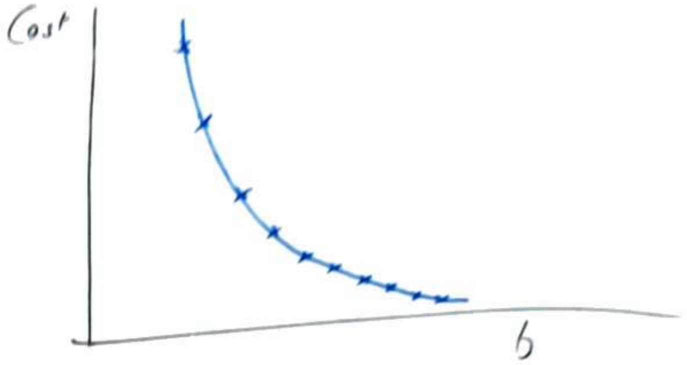
then for i=1 bnew becomes bold

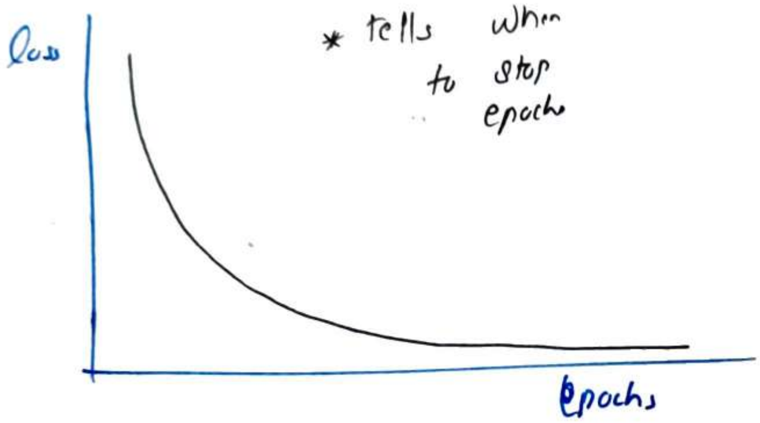
repeating epoch times

Learning rate → initially take big jump and as soon as get closer it takes Smaller Steps

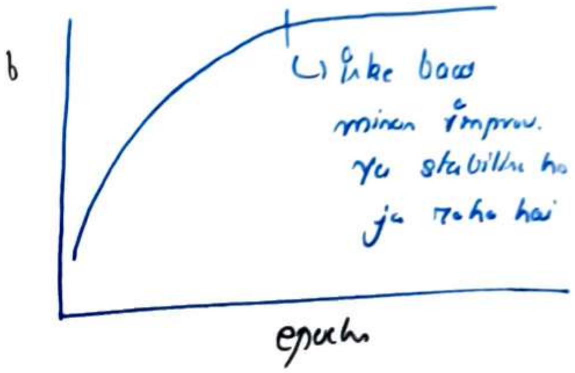
$$\boxed{b = b - \eta \, slope}$$







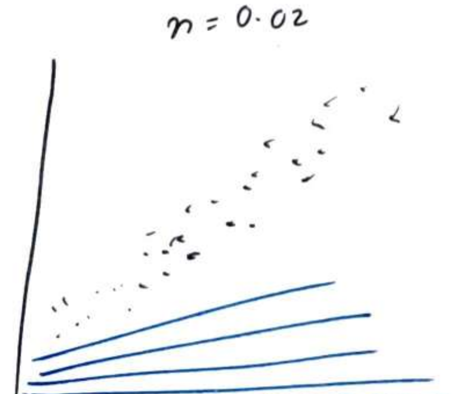
* tells when to stop epochs



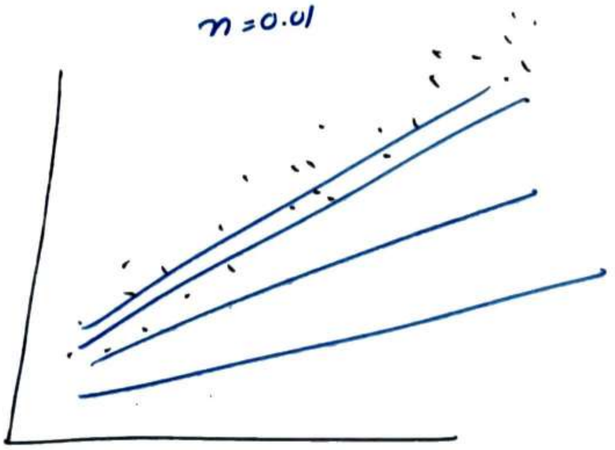
↳ like bas minor improv. ya stabilize ho ja raha hai

Shows ki aur epochs Chalane ki jarurat nahi hai

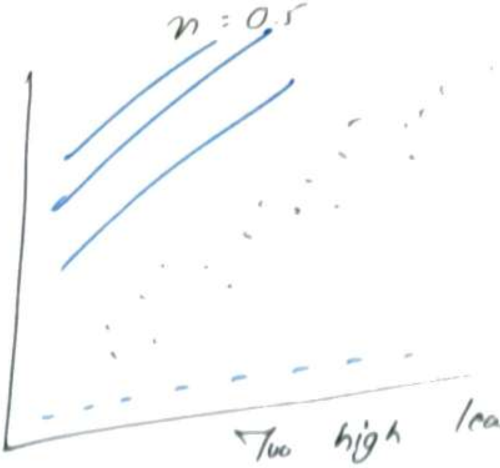
1. Effect of Learning Rate (epoch = 10)



too low



Appropriate learning rate.



jump hi bahut
dagy hai gaye
worst

Too high learning rate

* Hence, keep learning rate accurate as per the Data

② Universality of Gradient Descent.

$b = 0$

$\boxed{b = b_{old} - \eta \text{ slope}}$

Milta hai loss function ko Diffe. karke

$\frac{dL}{db} = [\ \ ]$ $\sum(y_i - \hat{y})^2$

Koi aur loss function ho

→ eqn is independent of Machine learning Algorithm toh kahi bhi lag Sakta hai

Adding m into the mix

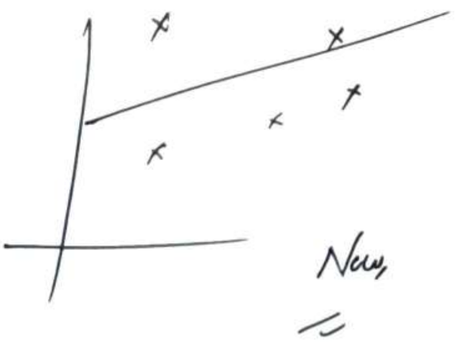
step ① Initial random Value for $m = 1$ and $b = 0$

② epoch = 100, $lr = 0.01$

for i in epochs

$b = b - \eta$ slope

$m = m - \eta$ slope

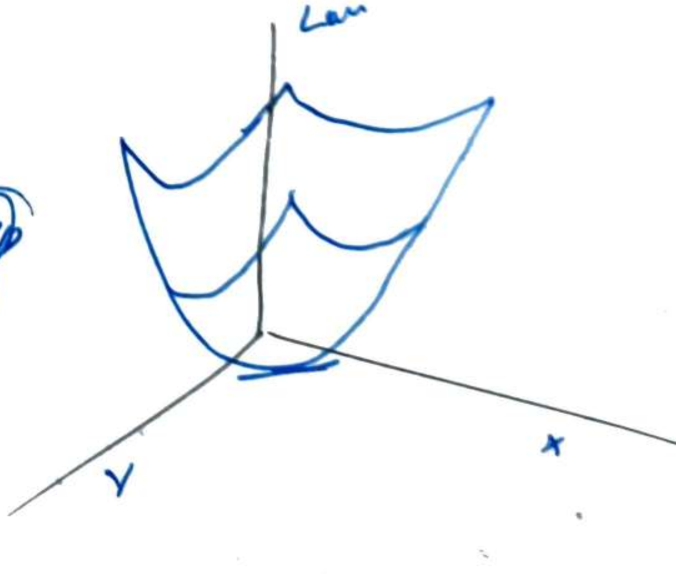


Now

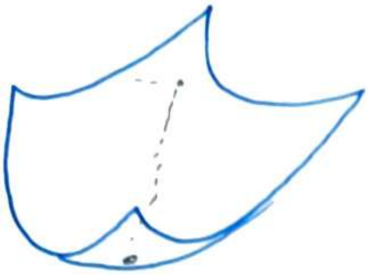
$L = \sum(y_i - \hat{y}_i)^2 = \sum(y_i - mx_i - b)^2$

$L(m, b)$

3D mai parabola hai



min^m lou function → pe m,b nikalna hai



pahad me wali khai mai girna hai to get lowest point

New, L(m,b) ~ gradient

b-slope = $\frac{\partial L}{\partial b} = \frac{\partial}{\partial b}\left(\sum (y_i - mx_i - b)^2 \right)$

$= 2\sum (y_i - mx_i - b)(-1)$

$= -2 \sum (y_i - mx_i - b)$

Slope_b at b = 0

m_slope : $\frac{\partial L}{\partial m} = 2 \sum (y_i - mx_i - b)(-x_i)$

$= -2 \sum (y_i - mx_i - b)(x_i)$

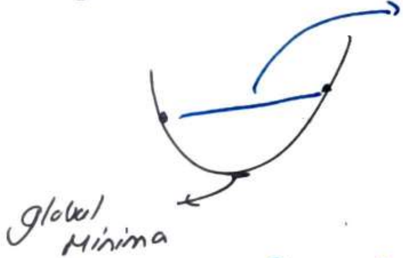
Slope_m at m = 1

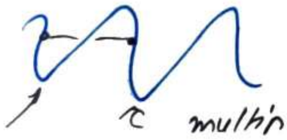
Effect of loss function

$L = \sum (y_i - \hat{y}_i)^2$ → Convex function

⇓

Two points ke bich ki line kabhi function ko cross nahi karta





→ Non-Convex

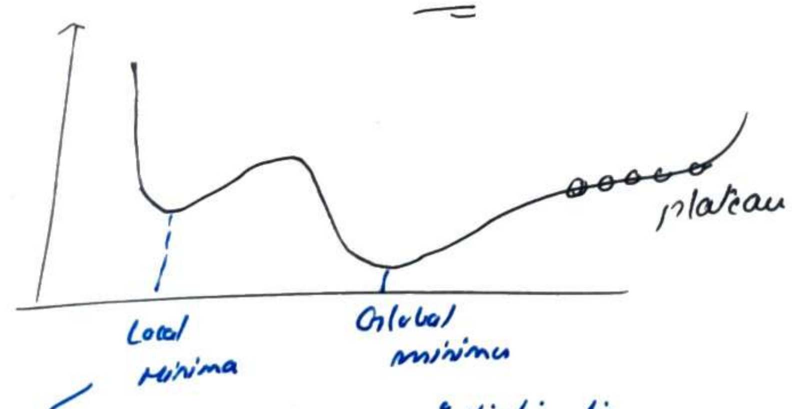
plateau → flat Surface

↓

Slope bahut slow change kar raha hai

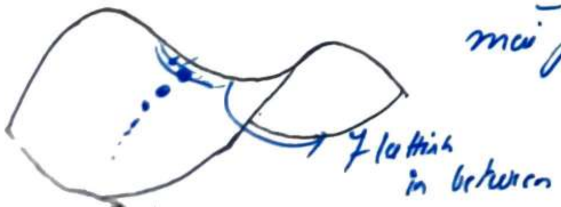
⇓

bahut time lagega



mai fas sakti hai } so initialisation is imp.

(30) Saddle point

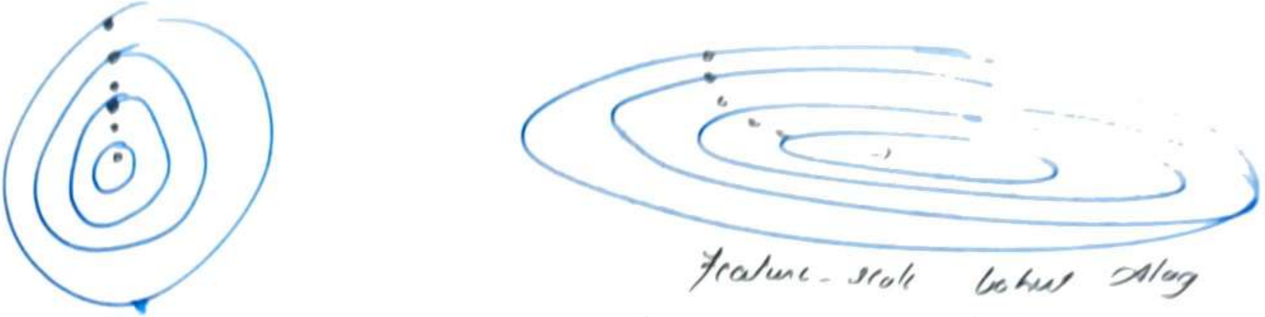


Effect of Data

if Data features - columns barabar scale pe hote hai

⇓

apko Contour plot bahut Circular Hota hai aur bahut jaldi Coverge karte ho



feature-scale bahut Alag hai

Converge bahut dhire se karenge.

islie data ko pehle same scale pe lana Chaiye.

Gradient Descent

Batch (✓)

Stochastic

Mini Batch

$m_n = m_0 - \eta$ Slope $\hookrightarrow \frac{\partial L}{\partial m}$

$b_n = b_0 - \eta$ Slope $\hookrightarrow \frac{\partial L}{\partial b}$

Now,
Slope Calculation mai full data use karte ho (300 Rows hongi
tok ~~300 [illegible] bagi paai Hall~~
~~line sahi kaam ke liye)~~
300 ghumke fir ek update.
(Generally Convex function)

jo update sirf ek row ke basis pe hota hai (faster)

300 row mai har row pe ek update.

∴ 300 update

(Mostly used)

Beech ki Cheez
(isme Batch Size define hota hai ... batch_size = 30)

jo m,b ko update hai vo 30 rows dekh ke baad Hoga

30 → 1 update