Batch gradient descent

* Types of gradient descent

1. Batch G.D
2. Stochastic G.D
3. Mini batch G.D

In the last lesson, we saw that if we are solving a problem in linear regression and we need to find the equation of a line, — that means finding m and b — then gradient descent starts with random values like m=1 and b=0.

After that, we keep iterating - running a loop — and keep changing the value of m and b step by step until we reach the correct values.

Batch GD

When we use batch gradient, we use the whole data to find the slope.

Let's say our data has 300 rows, then we look at all 300 rows before taking the first update — that means the first iteration happens after using all rows.

So, whatever we studied in the last lesson was batch gradient

1. Batch Gradient Descent (BGD)

Kya hota hai?

Pura training data ek saath leke gradient calculate karta hai.

Har epoch me sirf ek baar weight update hota hai.

Socho:

Tumhare paas 10,000 data points hain → ek epoch = 10,000 data points se gradient calculate karo → ek hi update.

Pros:

Accurate gradient.

Smooth convergence.

⇒ this batch gradient is very slow and also has some computational problems.

To avoid this, we use stochastic gradient.

⇒ In this method, the update happens based on a single row at a time. — that means we look at one row, check how much error we get, and then change m and b according to that

2. Stochastic Gradient Descent (SGD)

Kya hota hai?

Ek hi data point se gradient calculate karta hai.

Har data point ke baad turant weight update karta hai.

Socho:

Tumhare paas 10,000 data points hain → ek epoch me 10,000 updates!

Pros:

Fast updates.

Real-time learning.

Better for online learning.

3. Mini-Batch Gradient Descent

Kya hota hai?

Thoda thoda data (mini-batches) leke gradient calculate karta hai.

Eg: Batch size = 64 → har baar 64 data points pe update karega.

Socho:

Tumhare paas 10,000 data points hain → Batch size = 100 → Ek epoch = 100 batches → 100 updates per epoch.

Pros:

Trade-off between speed and stability.

Loss curve smooth hota hai (not too noisy like SGD).

Highly preferred in deep learning.

Mathematical formulation-batch gd

n-dimention-dataset → 3 col

$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$

(lpa) (cgpa) (iq)

↳ $\{\beta_0, \beta_1, \beta_2\}$

cgpa	iq	lpa
x_1	x_2	y
8.1	93	31.2
7.5	95	35.2

Step

① Random values

$\beta_0 = 0, \beta_1, \beta_2 = 1$

② epochs = 100, $\alpha = 0.1$

$\beta_0 = \beta_0 - \eta \text{ slope}$

$\beta_1 = \beta_1 - \eta \text{ slope}$

$\beta_2 = \beta_2 - \eta \text{ slope}$

$L(\beta_0, \beta_1, \beta_2)$

$\frac{\partial L}{\partial \beta_0} = \frac{\partial L}{\partial \beta_1} = \frac{\partial L}{\partial \beta_2}$

$L = \frac{1}{n} \sum_{i=1}^{n=2} (y_i - \hat{y}_i)^2$ {row = 2, col = 2}

$L = \frac{1}{2}\left[(y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_1)^2\right]$

$L = \frac{1}{2}\left[(y_1 - \beta_0 - \beta_1 x_{11} - \beta_2 x_{12})^2 + (y_2 - \beta_0 - \beta_1 x_{21} - \beta_2 x_{22})^2\right]$

cgpa	iq	lpa
x_1	x_2	y
8.1	93	31.2
7.5	95	35.2

$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2$

$\hat{y}_1 = \beta_0 + \beta_1 x_{11} + \beta_2 x_{12}$

$\hat{y}_2 = \beta_0 + \beta_1 x_{21} + \beta_2 x_{22}$

→ diff w.r.t β_0

$\frac{\partial L}{\partial \beta_0} = \frac{1}{2}\left[2(y_1 - \hat{y}_1)(-1) + 2(y_2 - \hat{y}_2)(-1)\right]$

$\frac{\partial L}{\partial \beta_0} = -\frac{2}{2}\left[(y_1 - \hat{y}_1) + (y_2 - \hat{y}_2)\right]$

$= -\frac{2}{n}\left[(y_1 - \hat{y}_1) + (y_2 - \hat{y}_2) + \cdots (y_n - \hat{y}_n)\right]$

$x_{i1} \Rightarrow$ x_{11}, x_{21}, x_{31}, x_{n1}

$$\frac{\partial L}{\partial \beta_0} = -\frac{2}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)$$

* $L = \frac{1}{2}\sum_{i=1}^{2}(Y_i - \hat{Y}_i)^2$

$$L = \frac{1}{2}\left[(Y_1 - \hat{Y}_1)^2 + (Y_2 - \hat{Y}_2)^2\right]$$

$$L = \frac{1}{2}\left[(Y_1 - \beta_0 - \beta_1 X_{11} - \beta_2 X_{12})^2 + (Y_2 - \beta_0 - \beta_1 X_{21} - \beta_2 X_{22})^2\right]$$

diff w.r.t β_1

$$\frac{\partial L}{\partial \beta_1} = \frac{1}{2}\left[2(Y_1 - \hat{Y}_1)(-X_{11}) + 2(Y_2 - \hat{Y}_2)(-X_{21})\right]$$

$$\frac{\partial L}{\partial \beta_1} = -\frac{2}{n}\left[(Y_1 - \hat{Y}_1)(X_{11}) + (Y_2 - \hat{Y}_2)(X_{21}) + \cdots (Y_n - \hat{Y}_n)(X_{n1})\right]$$

$$\frac{\partial L}{\partial \beta_1} = -\frac{2}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)(X_{i1})$$

$$\frac{\partial L}{\partial \beta_2} = -\frac{2}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)(X_{i2})$$

General formula

$$\frac{\partial L}{\partial \beta_m} = -\frac{2}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)X_{im}$$

$$\hat{Y}_1 = \beta_0 + \beta_1 X_{11} + \beta_2 X_{12} + \beta_3 X_{13}$$
$$\hat{Y}_2 = \beta_0 + \beta_1 X_{21} + \beta_2 X_{22} + \beta_3 X_{23}$$

$$\hat{Y} = \beta_0 + \begin{bmatrix} X_{11} & X_{12} & X_{13} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}$$

$$\hat{Y} = np.dot(coef\_, x\_train) + \beta_0$$