

# Machine Learning

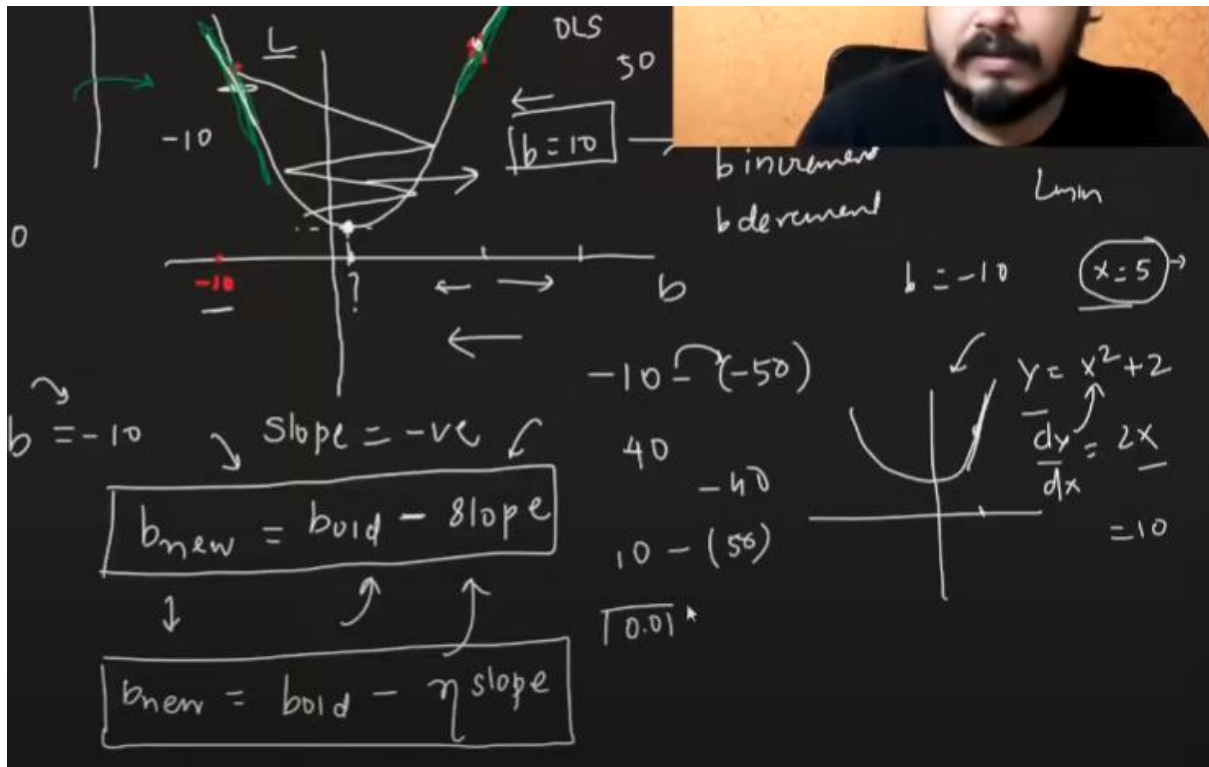
### Video 56:

## Gradient Descent From Scratch | End to End Gradient Descent | Gradient Descent Animation

Gradient descent is an optimization algorithm used in machine learning to minimize the loss function by iteratively adjusting model parameters. It computes the gradient (or slope) of the loss function with respect to each parameter and updates the parameters in the direction of the negative gradient to reduce the error.

Thus, it is a general algorithm which gives minima.

## Intuition



Mathematical formulation:

Mathematical Formulation  
Thursday, May 20, 2021 1:47 PM

$m = 78.35$

Step → start with a random  $b = b$

for i in epochs;  $1000, 100$

$\eta = 0.01$

$b = 0$

$b_{new} = b_{old} - \eta \times \text{slope}(b = 0)$

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

$\frac{dL}{db} = \frac{d}{db} \left( \sum_{i=1}^n (y_i - \hat{y}_i)^2 \right) = 2 \sum_{i=1}^n (y_i - m x_i - b)$

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

$\frac{d}{db} \sum_{i=1}^n (y_i - m x_i - b)^2 \text{ slope} = -2 \sum_{i=1}^n (y_i - m x_i - b)$

$= -2 \sum_{i=1}^n (y_i - 78.35 x_i - 0)$

$\text{slope}(b=0)$

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

Example:

Code link: (Part1)

<https://colab.research.google.com/drive/16rNbHidjCza2Tk0ytPJEhpNfMCGdauPk?usp=sharing>

Code link: (Part2)

<https://colab.research.google.com/drive/19VxfGZNxRgJoJYy7Jl9zkEybluvN945w?usp=sharing>

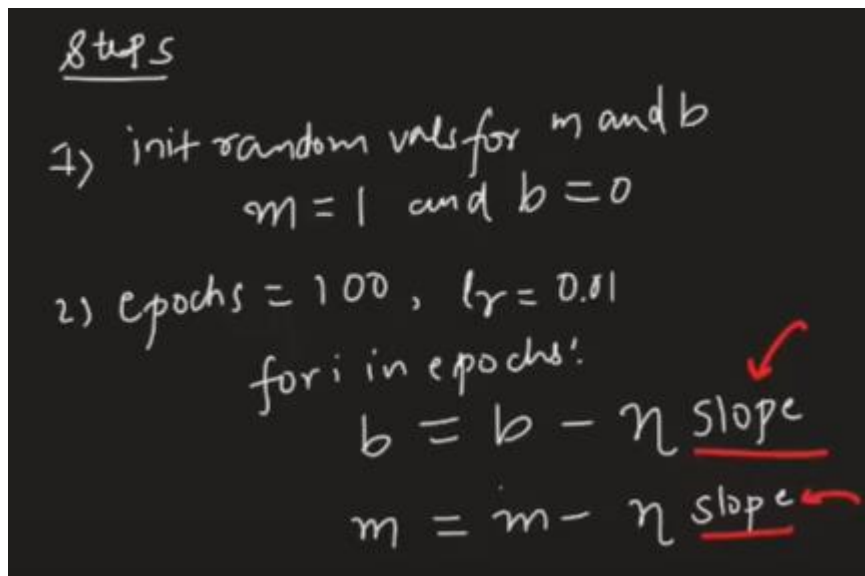
What is the effect of learning rate on gradient descent?

The learning rate in gradient descent controls how much the model's weights are updated during each iteration. A high learning rate can cause the model to overshoot the optimal solution, while a low learning rate may lead to slow convergence or getting stuck in local minima.

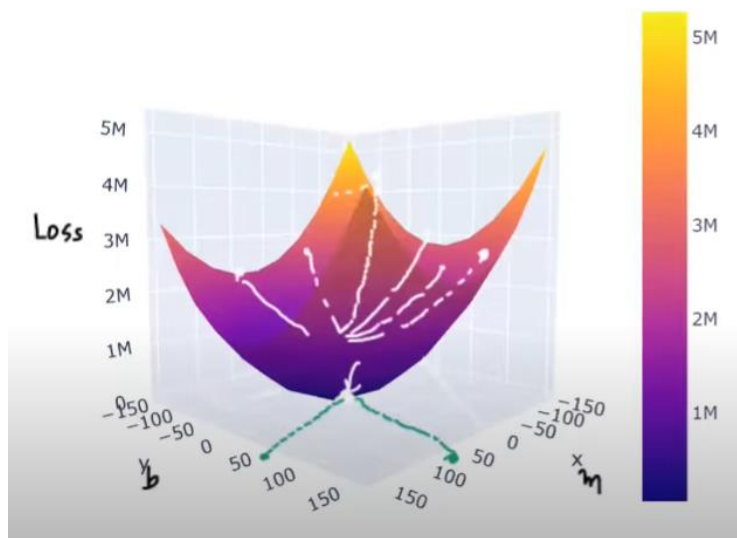
The universality of gradient descent refers to its ability to be applied to a wide range of machine learning problems, from simple linear regression to complex deep learning models. It is a general optimization algorithm that can minimize a variety of loss functions, making it versatile across different domains.

Now, till now we were keeping one of the  $b$  or  $m$  as constant, but now we will change both.

Steps:



Visualisation of loss:



Example:

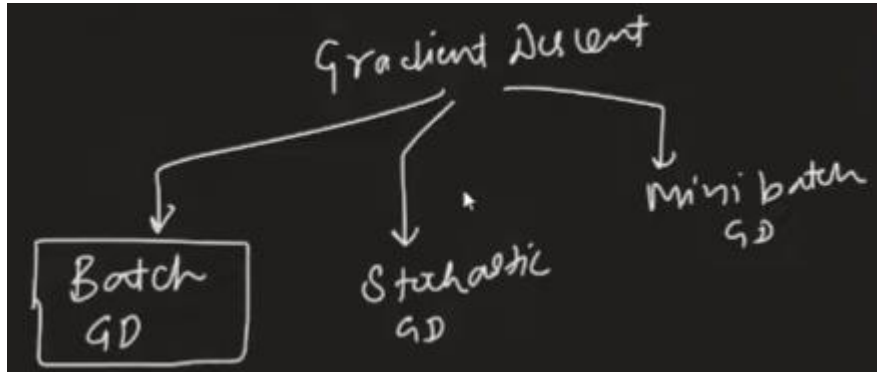
Code link: (Part3)

[https://colab.research.google.com/drive/1MrDI0bnuv7dTcxpdR99UrKiC40j5\\_IN-?usp=sharing](https://colab.research.google.com/drive/1MrDI0bnuv7dTcxpdR99UrKiC40j5_IN-?usp=sharing)

## Video 57:

### Batch Gradient Descent with Code Demo:

We know the following types of gradient descent exists: on basis of



Batch gradient descent is an optimization algorithm used in machine learning to minimize a loss function. It computes the gradient of the entire training dataset to update model parameters in each iteration. This approach ensures steady convergence but can be computationally expensive with large datasets.

Mathematical formulation:

Mathematical Formulation

Saturday, May 22, 2021 3:38 PM

$n$ -dim-dataset 3-cols

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

(lpa) (cgpa) (iq)

$\{ \beta_0, \beta_1, \beta_2 \}$

cgpa	iq	lpa
$x_1$	$x_2$	$y$
8.1	93	3.2
7.5	95	3.5

(2,3)

$\{m, b\}$

# Mathematical Formulation

Saturday, May 22, 2021 3:38 PM

$\eta$ -dim - dataset 3-cols

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

(lpr) (cgp) (iq)

cgp | iq | lpr

$x_1$	$x_2$	$y$
8.1	93	3.2
7.5	95	3.5

(2,3)

{m, b}

{ $\beta_0, \beta_1, \beta_2$ }

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

1) Random values

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

2) epoch = 100,  $\eta = 0.1$

$$\beta_0 = \beta_0 - \eta \frac{\partial L}{\partial \beta_0}$$

$$\beta_1 = \beta_1 - \eta \frac{\partial L}{\partial \beta_1}$$

$$\beta_2 = \beta_2 - \eta \frac{\partial L}{\partial \beta_2}$$

1) Random values

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

2) epoch = 100,  $\eta = 0.1$

$$\begin{cases} \beta_0 = \beta_0 - \eta \frac{\partial L}{\partial \beta_0} \\ \beta_1 = \beta_1 - \eta \frac{\partial L}{\partial \beta_1} \\ \beta_2 = \beta_2 - \eta \frac{\partial L}{\partial \beta_2} \end{cases}$$

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

$\eta$ -dim

( $\eta+1$ )  $\beta_0 - \beta_\eta$

$$MSE \quad \beta_2 = \beta_2 - \eta \frac{\partial L}{\partial \beta_2}$$

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

{row=2, cols=2+1}

$$\hat{y}_i = \beta_0 +$$

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

$x_1$	$x_2$	$y$
8.1	93	3.2
7.5	95	3.5

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

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

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

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

$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2$   
 $\hat{y}_1 = \beta_0 + \beta_1 x_{11} + \beta_2 x_{12}$   
 $\hat{y}_2 = \beta_0 + \beta_1 x_{21} + \beta_2 x_{22}$

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

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

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

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

$$\frac{\partial L}{\partial \beta_1} = \frac{1}{2} [2(y_1 - \hat{y}_1)(-x_{11}) + 2(y_2 - \hat{y}_2)(-x_{21})]$$

$\frac{\partial}{\partial \beta_1} - \beta_1 x_{11} = -x_{11}$

	$x_1$	$x_2$	$y$
1	8.1	9.3	3.2
2	7.5	9.5	3.5

$$\frac{\partial L}{\partial \beta_1} = \frac{1}{2} \left[ 2(y_1 - \hat{y}_1)(-x_{11}) + 2(y_2 - \hat{y}_2)(-x_{21}) + \dots + 2(y_n - \hat{y}_n)(-x_{n1}) \right]$$

$$\frac{\partial L}{\partial \beta_1} = \frac{-2}{n} \left[ (y_1 - \hat{y}_1)x_{11} + (y_2 - \hat{y}_2)x_{21} + (y_3 - \hat{y}_3)x_{31} + \dots + (y_n - \hat{y}_n)x_{n1} \right]$$

$$\frac{\partial L}{\partial \beta_1} = -\frac{2}{n} \sum_{i=1}^n (y_i - \hat{y}_i) x_{i1} \quad (x_{i1})$$


04:48 Mathematical Formulation of BCD

$$\frac{\partial L}{\partial \beta_1} = -\frac{2}{n} \sum_{i=1}^n (y_i - \hat{y}_i) x_{i1}$$

$x_{i1} \rightarrow 1^{\text{col data}}$   
 $\beta_1 \rightarrow \text{values of 1 col.}$

$m \text{ cols}$   
 $\beta_0 - \beta_m$

$\left\{ \frac{\partial L}{\partial \beta_m} \right\} = -\frac{2}{n} \sum_{i=1}^n (y_i - \hat{y}_i) x_{im}$  code



Example:

Code link:

<https://colab.research.google.com/drive/1v4-iCnj9S6JPuUoY8pW-1znm2bMhfmcP?usp=sharing>



## **Video 58:**

### **Stochastic Gradient Descent**

Batch Gradient Descent (BGD) faced difficulties with large datasets due to high computation time and memory requirements, as it computed gradients over the entire dataset before updating model parameters. This led to slow convergence and made it impractical for big data. Additionally, BGD could get stuck in local minima due to its deterministic nature. Stochastic Gradient Descent (SGD) addressed these issues by updating parameters after each data point or small batch, speeding up training and reducing memory consumption. Although SGD introduces noisy updates, this randomness helps escape local minima, leading to faster convergence, especially for complex models.

Example:

Code link:

<https://colab.research.google.com/drive/1v4-iCnj9S6JPuUoY8pW-1znm2bMhfmcP?usp=sharing>

Stochastic Gradient Descent (SGD) is particularly useful in the following situations:

1. **Large Datasets:**
  - a. When dealing with massive datasets that don't fit into memory, SGD processes one data point (or a small mini-batch) at a time, reducing memory requirements and making it more scalable.
2. **Online Learning:**
  - a. When new data arrives continuously, such as in real-time applications, SGD can be used to update the model incrementally without needing to retrain on the entire dataset.
3. **Faster Convergence (for Complex Models):**
  - a. For large or complex models (e.g., deep learning), SGD can speed up convergence by providing frequent parameter updates, helping the model learn faster, especially in the early stages.
4. **Avoiding Local Minima:**
  - a. In non-convex optimization problems (like deep neural networks), the noisy updates of SGD can help the algorithm escape local minima and explore a larger part of the solution space, potentially finding better global minima.
5. **Real-Time or Dynamic Environments:**
  - a. In scenarios where the model needs to be updated frequently (like in reinforcement learning or dynamic systems), SGD is a good choice due to its ability to handle continuous updates efficiently.

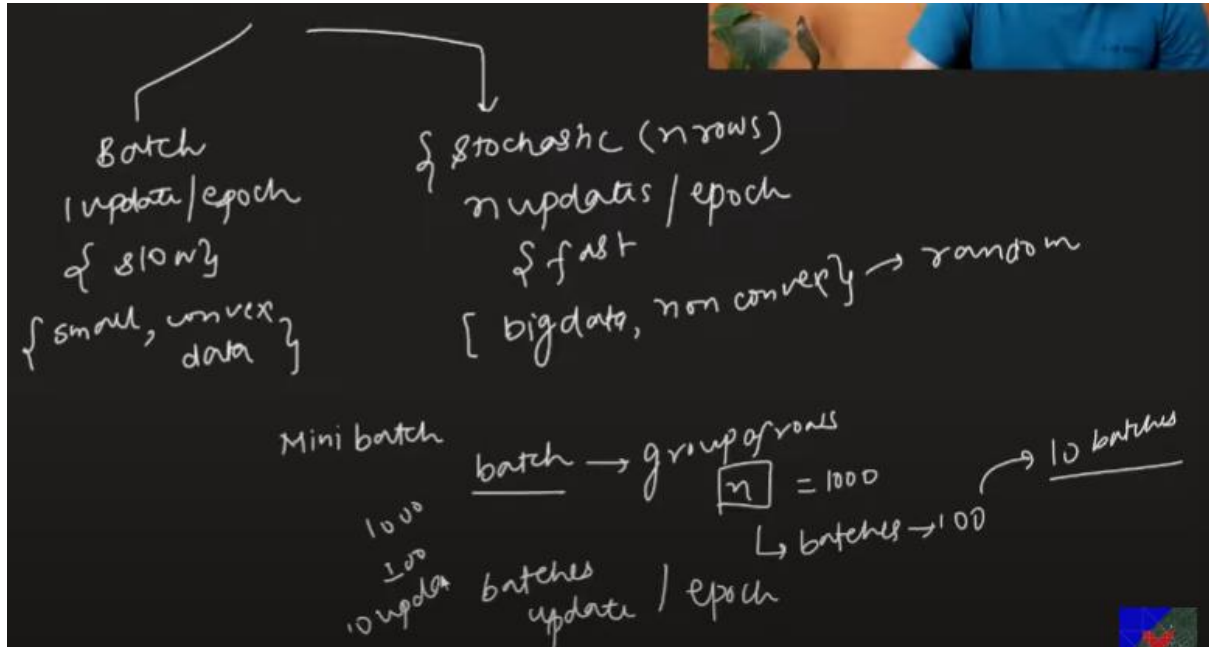
Overall, SGD is preferred when speed, scalability, and the ability to handle large, dynamic datasets are crucial.



## Video 59:

### Mini-Batch Gradient Descent

We know,



Example:

Code link:

<https://colab.research.google.com/drive/1momw0kV2w51x0L847d4lMYx3x-aWUuiQ?usp=sharing>

## **Video 60:**

### **Polynomial Regression**

When linear regression fails?



Example:

Code link:

<https://colab.research.google.com/drive/1momw0kV2w51x0L847d4lMYx3x-aWUuiQ?usp=sharing>