

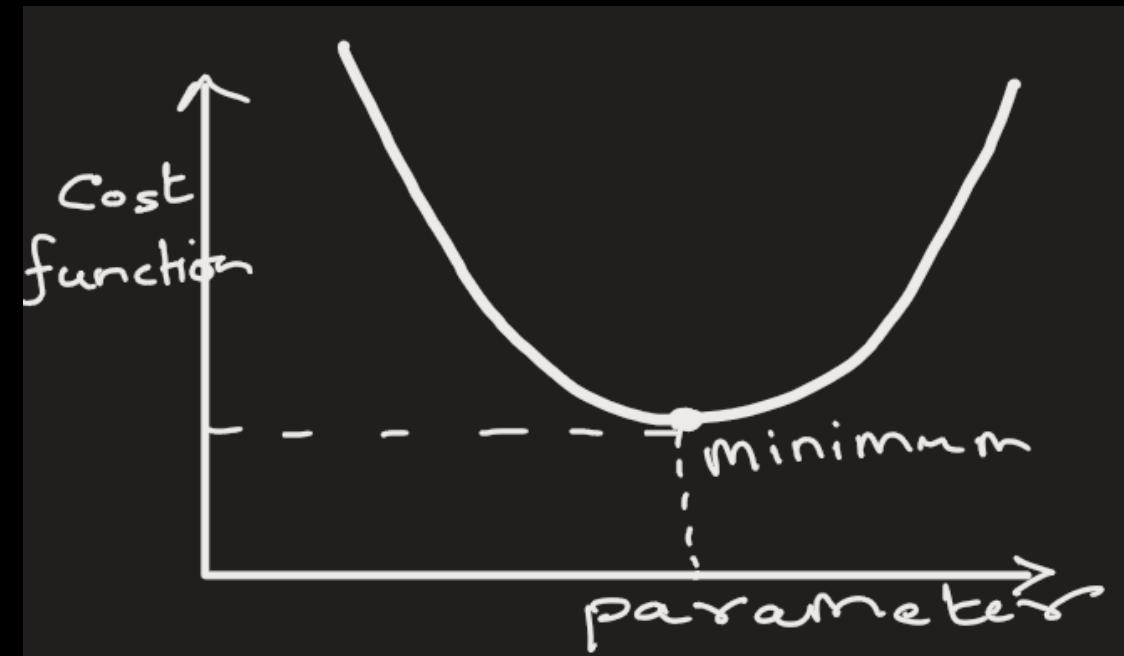
Gradient Descent

Gradient Descent is an iterative optimization algorithm used to find the minimum of a function.

In ML, this function is called **Loss function or Cost function**

This function measures how wrong the model's predictions are.
It is the difference between actual value and predicted value.

GD is well suited for **large datasets**

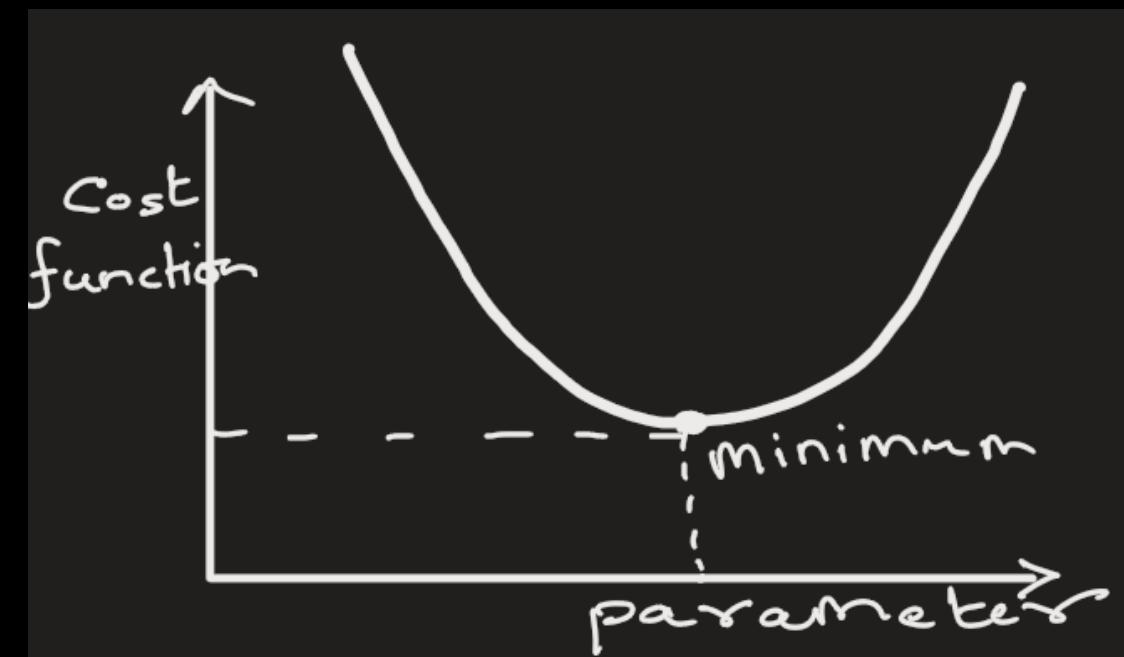


Gradient Descent



In ML, there is difference between actual numerical value and predicted numerical value.
Example:

age	wt.	actual	Pred.
x_1	x_2	y_i	\hat{y}_i
1	8	3	5
2	4	1	2
2	5	7	6
:	:	:	:



Gradient Descent

The goal of gradient descent is to **adjust model parameters so that the loss becomes as small as possible.**

In other words, find the parameters that makes the cost function minimum.

In linear regression, the cost function is given by,

$$\begin{aligned} C = \text{MSE} &= \frac{1}{n} \sum (y_i - \hat{y}_i)^2 \\ &= \frac{1}{n} \sum (y_i - (b + mx_i))^2 \end{aligned}$$

Here we have to find parameter m and b ,
such that C is minimum



Gradient Descent

So, we start with some parameter and then keep changing it constant amount until we find the minima.

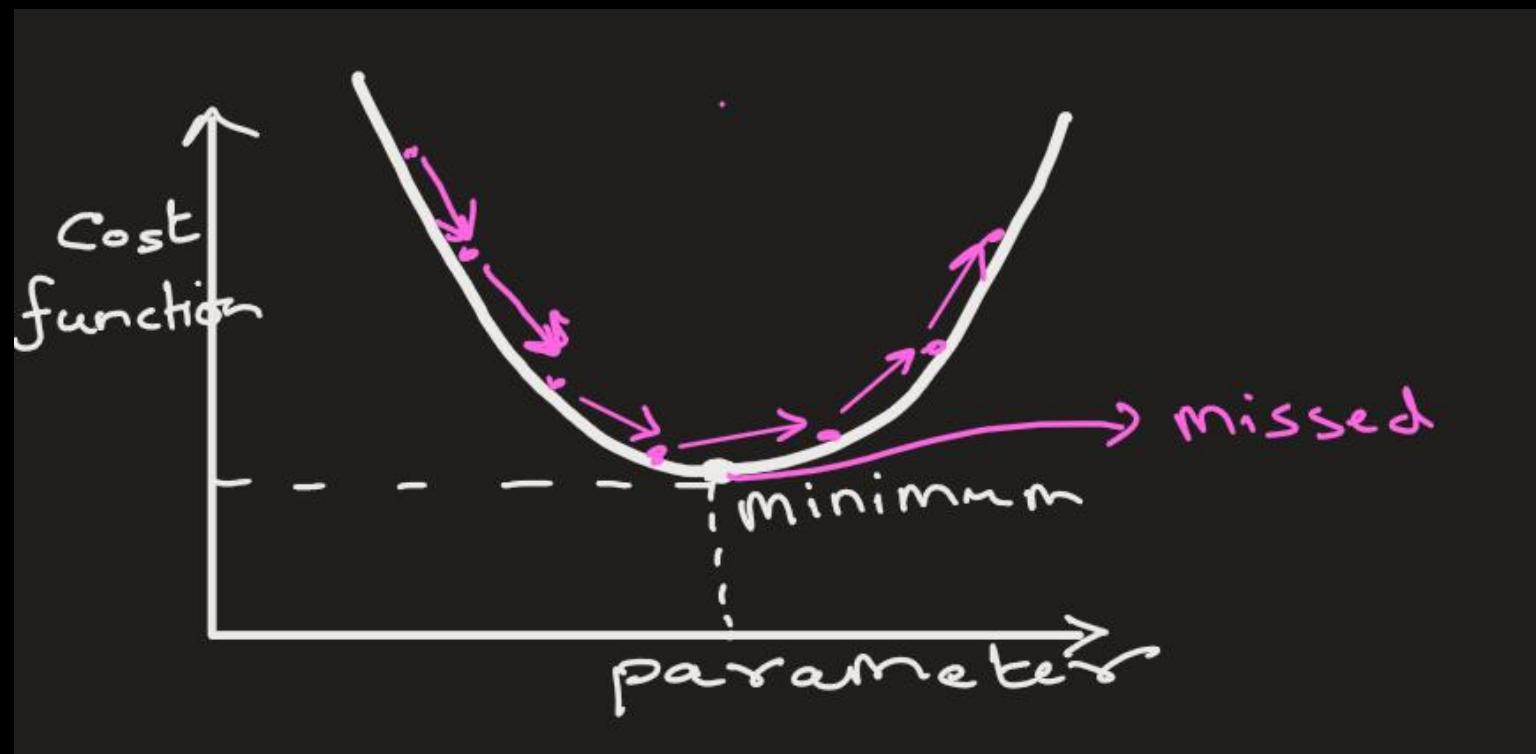
For example: start with $b = 5$

Next $b = 5 - 1 = 4$

Next $b = 4 - 1 = 3$

Next $b = 3 - 1 = 2$

and so on



There is a problem with this approach:

If we reduce parameter(say b), by **fixed size** then we may overshoot our minimum point and miss it completely.

Gradient Descent

The solution is that we change the parameter not by a fixed value but by a variable rate that depends on the slope.

This rate is also known as **learning rate**.

$$\begin{aligned} C = \text{MSE} &= \frac{1}{n} \sum (y_i - \hat{y}_i)^2 \\ &= \frac{1}{n} \sum (y_i - (b + mx_i))^2 \end{aligned}$$

The slope of C is given by

$$\frac{\partial C}{\partial m} = -\frac{2}{n} \sum x_i (y_i - (b + mx_i))$$

$$\frac{\partial C}{\partial b} = \frac{2}{n} \sum (y_i - (b + mx_i))$$

Gradient Descent

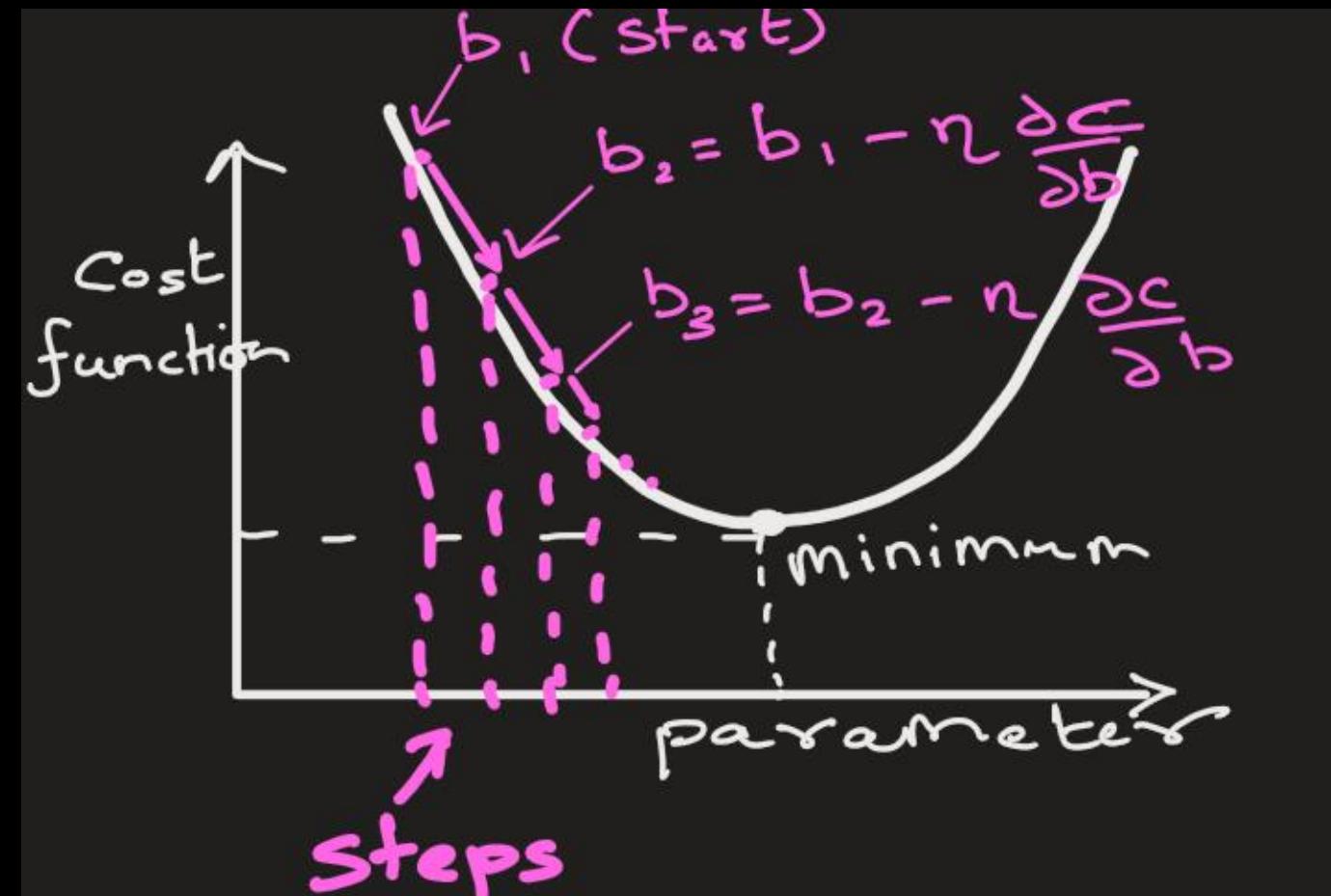
We find m and b by repetitively updating its value that depends on **learning rate and slope**, , and then calculating the cost function.

We keep doing this until we find m and b where **cost is minimum**

$$m = m - n \frac{\partial C}{\partial m}$$

$$b = b - n \frac{\partial C}{\partial b}$$

learning rate
(Step size)



Gradient Descent

How Gradient Descent Works (Step-by-Step)

1. Initialize parameters (weights) randomly
2. Compute loss using current parameters
3. Compute gradient (how loss changes w.r.t parameters)
4. Update parameters
5. Repeat until convergence

The image shows handwritten mathematical notes on a dark background, likely a chalkboard. At the top, there is a diagram with two arrows pointing from the text 'Parameter' and 'gradient w.r.t' to the variables m and b respectively. Below this, the update rule for parameter m is written as:

$$m = m - n \underbrace{\frac{\partial C}{\partial m}}_{\text{gradient w.r.t } m}$$

Below this, the update rule for parameter b is shown:

$$b = b - n \frac{\partial C}{\partial b}$$

An arrow points from the text 'learning rate (Step size)' to the term n in the equation above.

Heading Goes Here



Fhdsklf
Fjdsklf
Fjskldf

