Workings of Neural Networks

Video 3: Optimization Techniques - Gradient Descent

Analytics Vidhya

In Air

Minimizing Loss

Minimizing Loss - LR

$$y = \underbrace{w}_{\text{Slope}}x + \underbrace{b}_{\text{Intercept}}$$

1. Closed form mathematical solution
2. Iterating w and b values for minimum loss

1 Closed form mathematical solution

- Direct method to calculate best-fit line
- Approach complexity increases in multi-variable scenarios

2 Iterating w and b values for minimum loss

Infinite combinations of $\boldsymbol{w}$ and $\boldsymbol{b}$ make exhaustive search impractical.

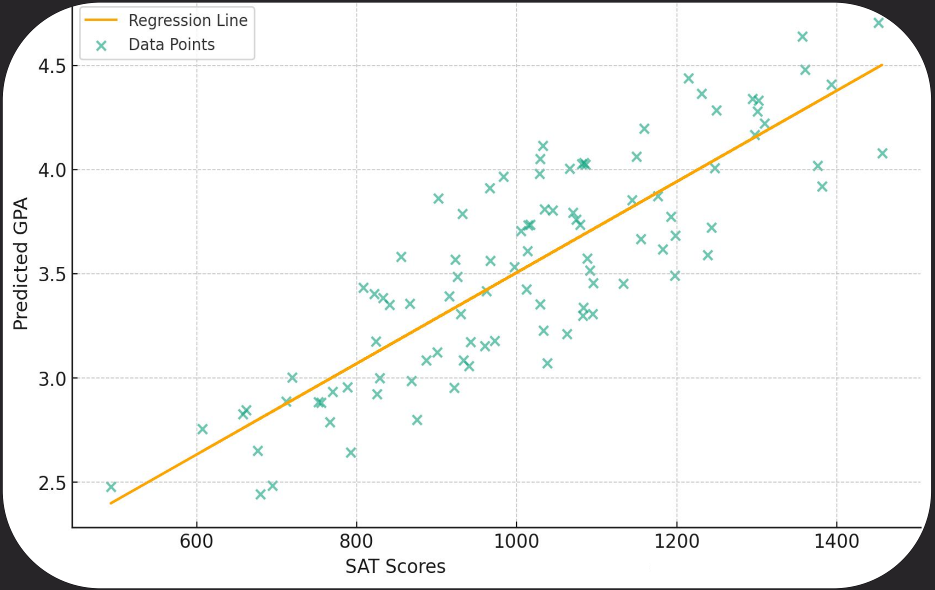
Example: Weight of 3.3 with bias of 10.1, or weight of 3.31 with bias of 10.11, etc

Gradient Descent

This technique is used to find the local minimum or optimize the loss function

- A smarter approach to testing different w and b values
- Backbone of neural networks


Predicting first year GPA of students.



$$y = w_i * x_i + b_i$$

w_i = weight of the slope

x_i = input, sat_score

b_i = bias or intercept

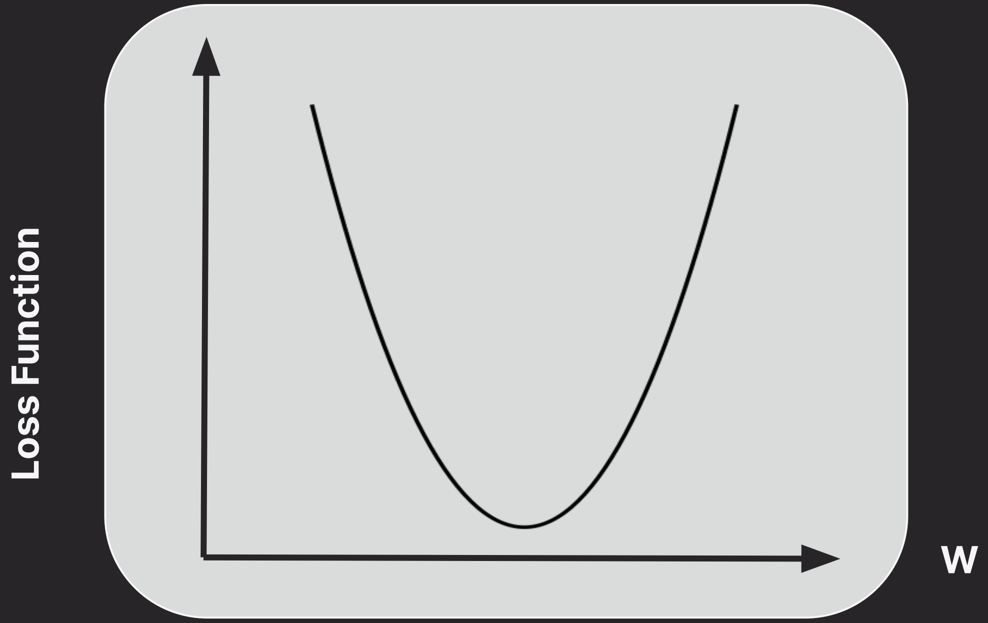
Find the best fit line to minimize the loss

$$MSE = (1/n)\ \Sigma|\hat{y}-y|^2$$

$$MSE = (1/n)\ \Sigma|\hat{y}-y|^2$$

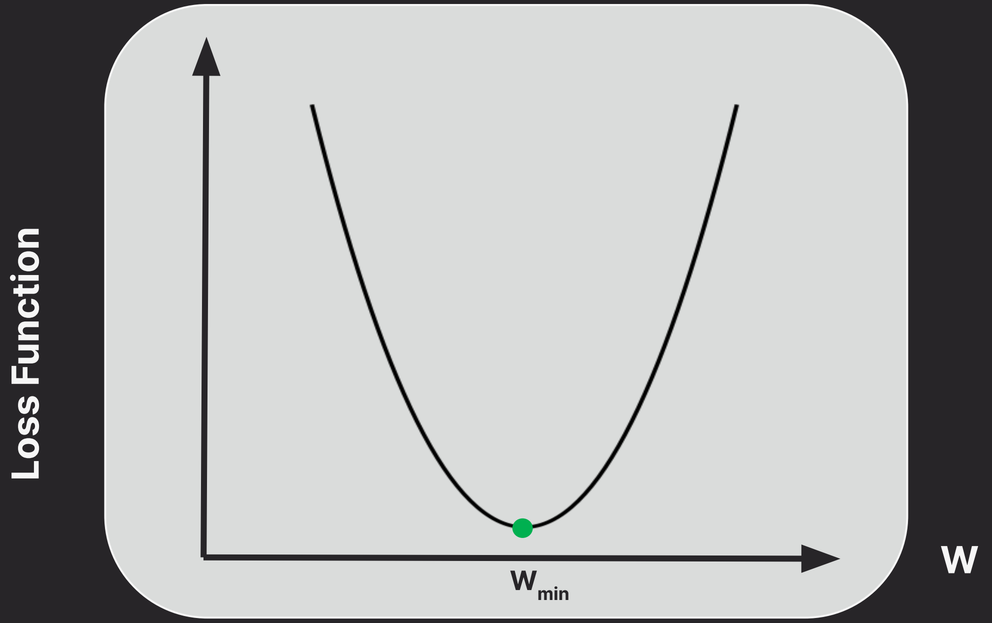
$$= (1/n)\ \Sigma|(wx+b)-y|^2$$

Objective: Find a value for weight where the loss is minimum

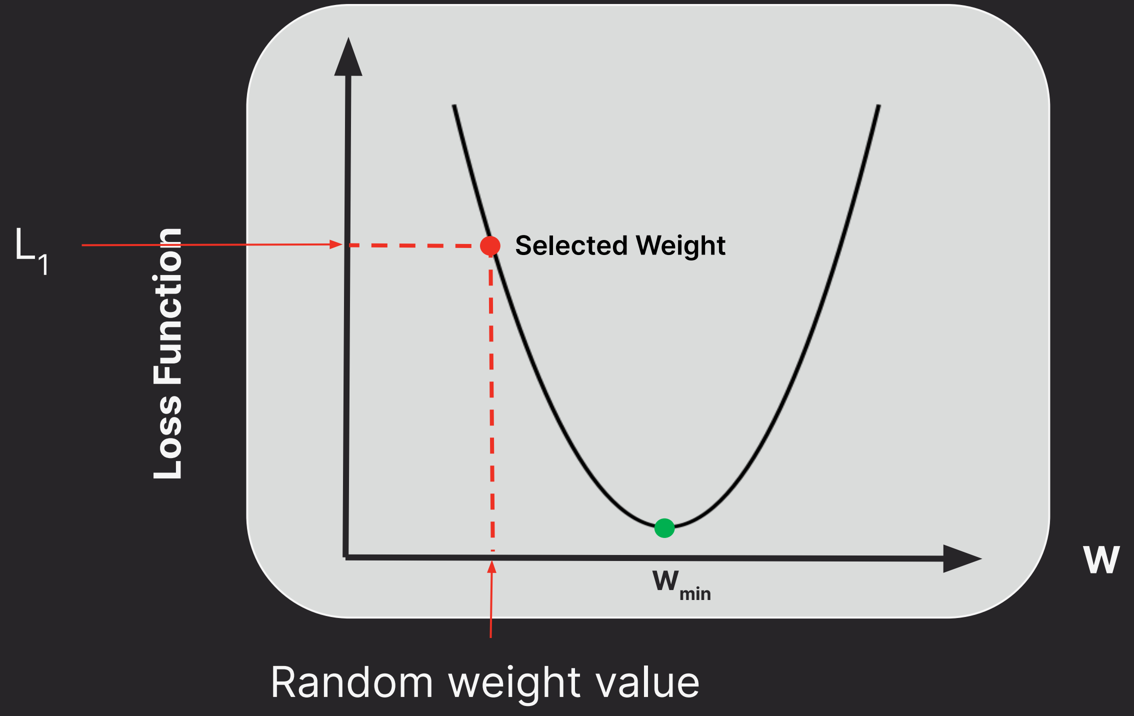


Note: Assume bias as constant to focus only on the impact of weight (w) on loss.

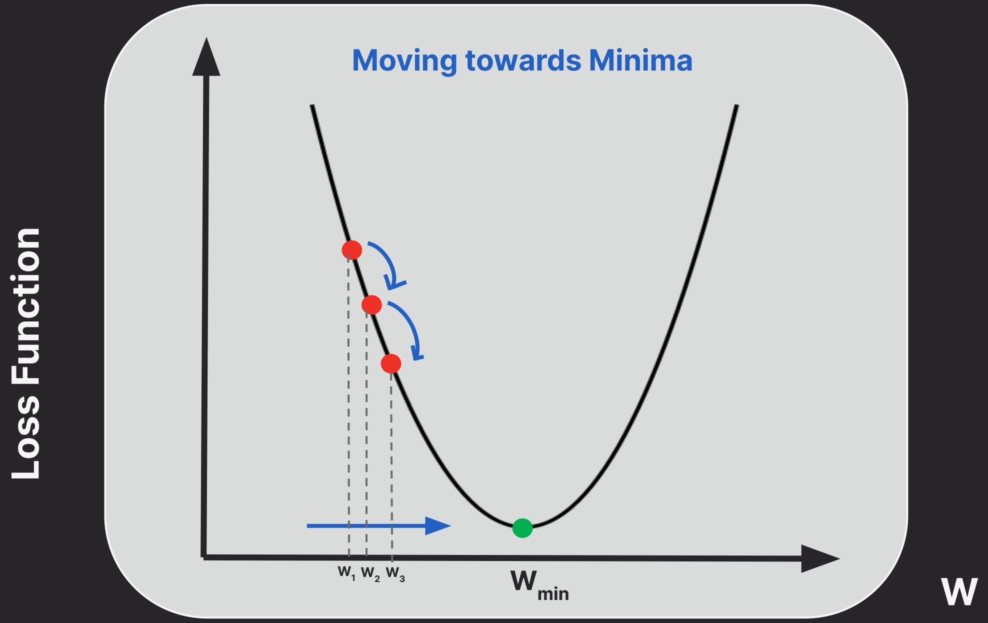
What is W_{min} ?



Calculating the Slope

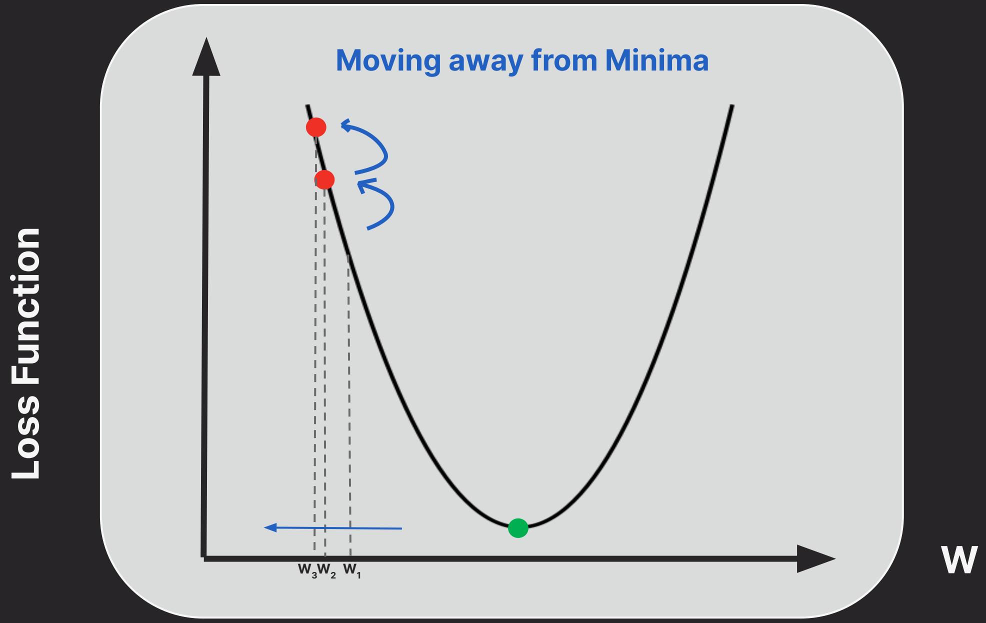


Calculating the Slope



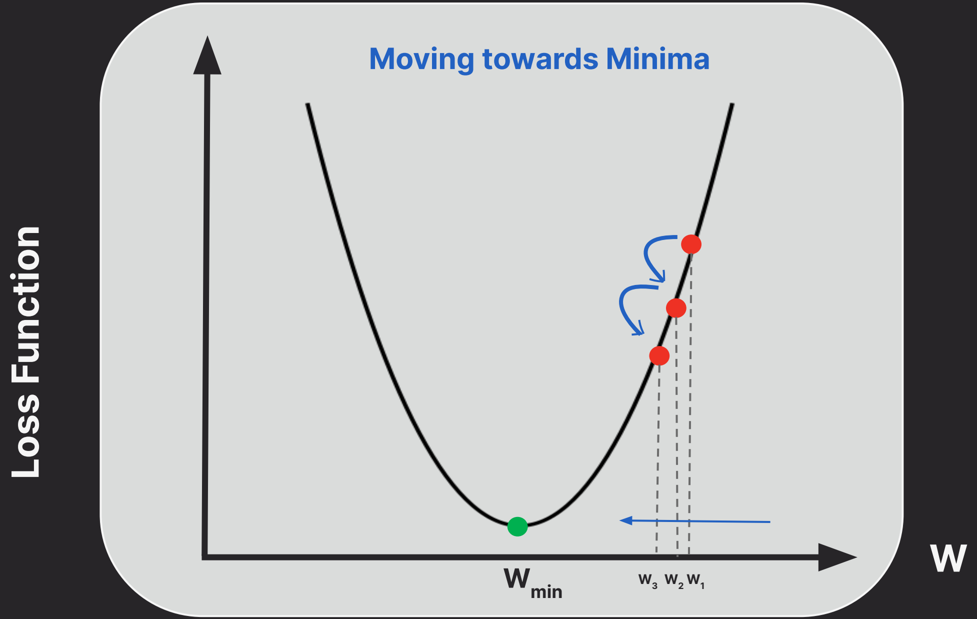
$W_3 > W_2 > W_1$

Calculating the Slope



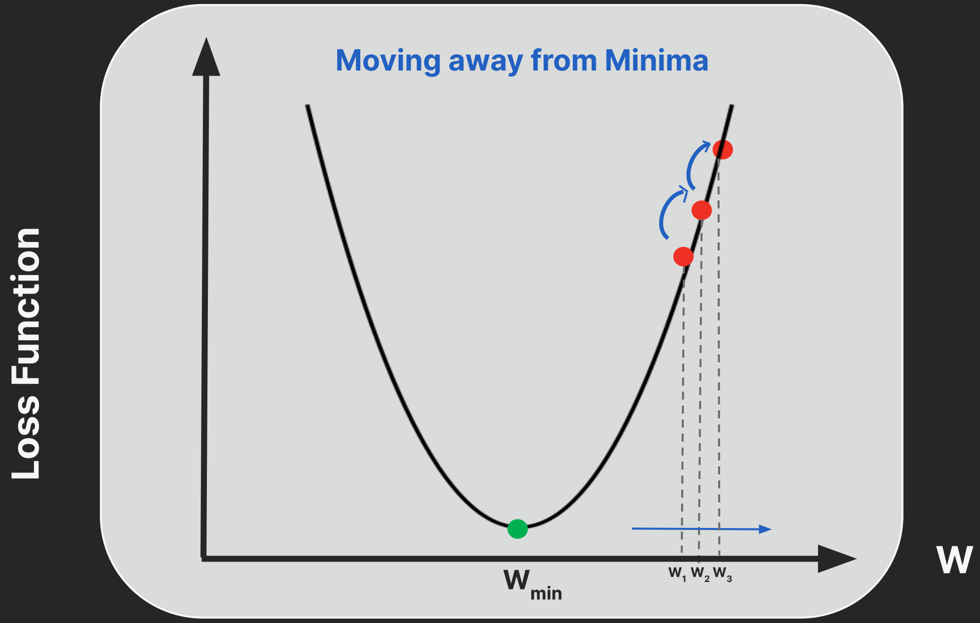
$W_1 > W_2 > W_3$

Calculating the Slope



$W_1 < W_2 < W_3$

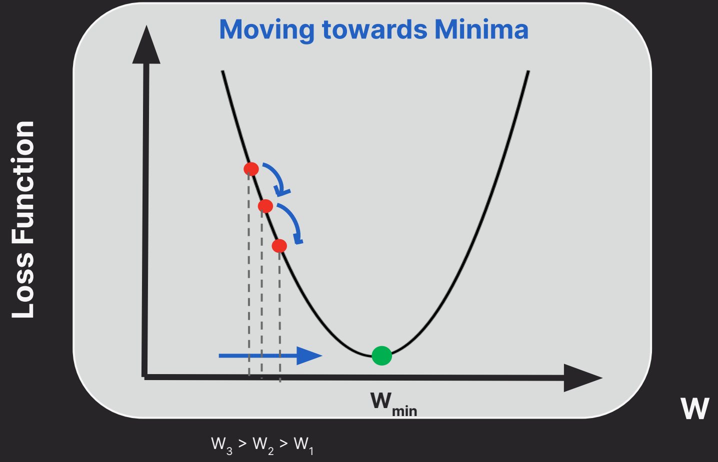
Calculating the Slope

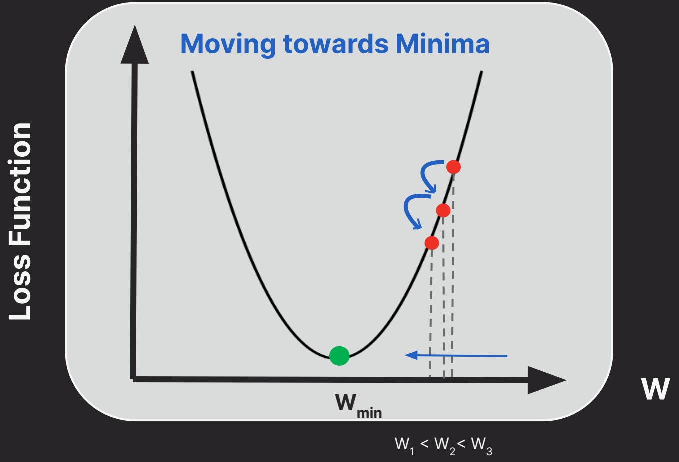


$W_3 > W_2 > W_1$

Analytics Vidhya

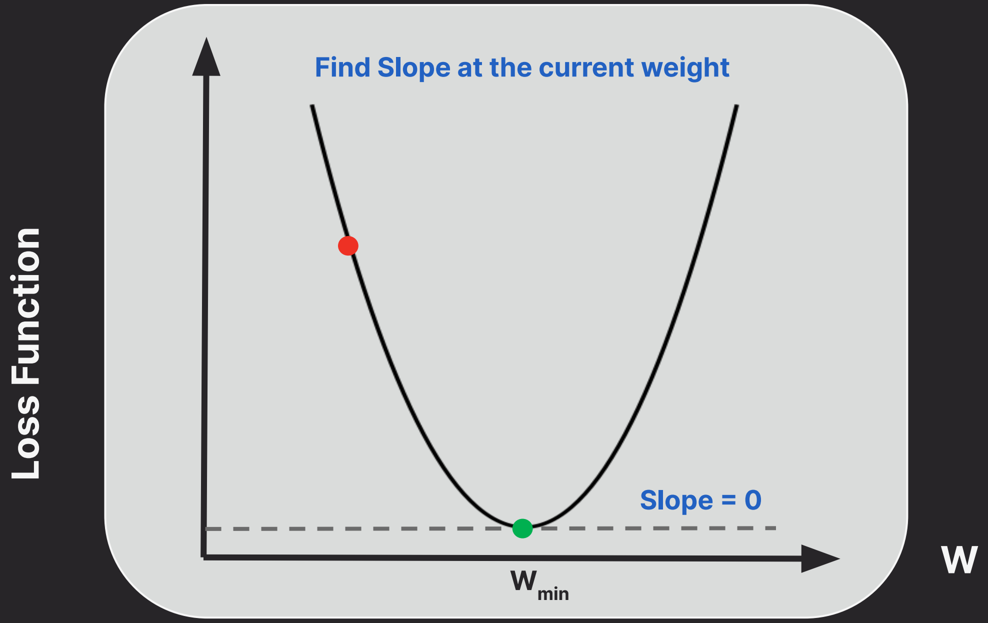
Calculating the Slope



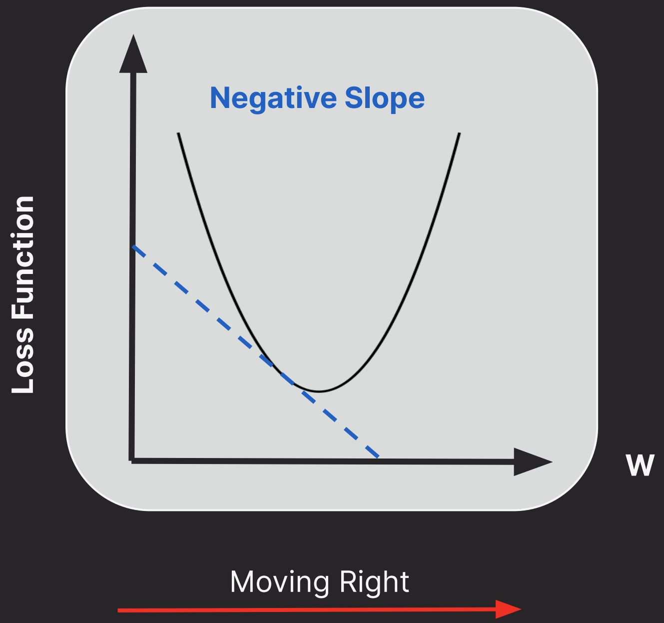


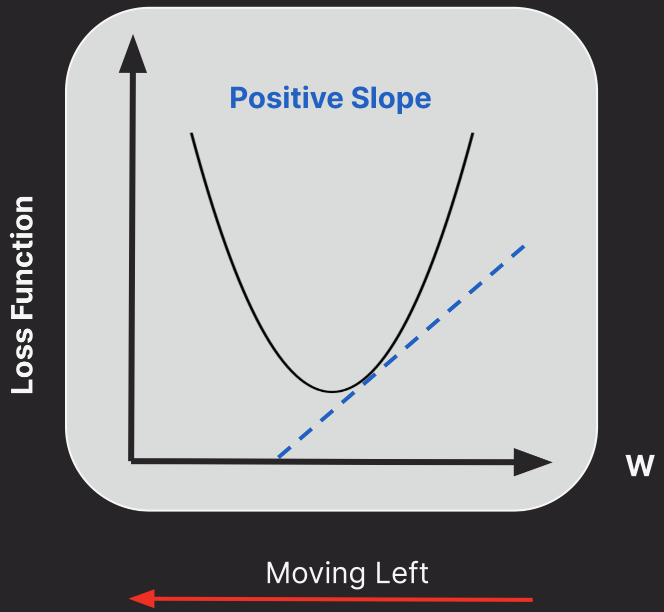
How would the algorithm know which side to move towards?

Calculating the Slope



Calculating the Slope





(wxi

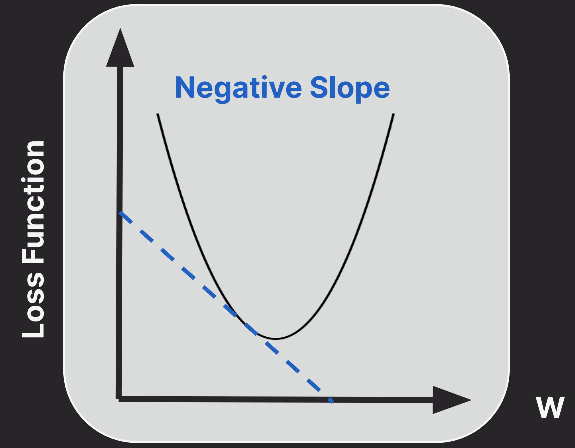
Calculating the Slope

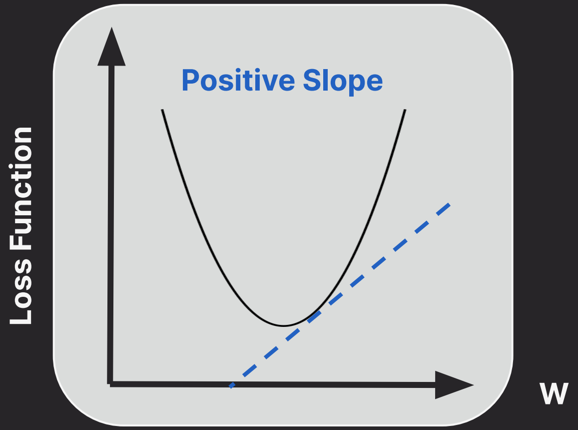
$$\frac{\mathrm{d}}{\mathrm{d}w}(MSE)$$

$$MSE = \frac{1}{N}\Sigma_{\mathrm{i}=1}^{\mathrm{n}}\left((wx_i + b) - y_i\right)^2$$

Updating Weights

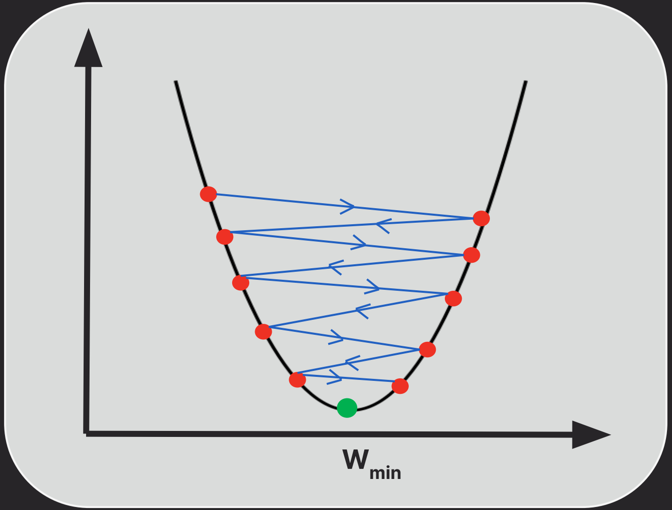
New weight (w_{new}) = Old Weight (w_{old}) - Slope





Challenges in Weight Update

New weight (w_{new}) = Old Weight (w_{old}) - Slope



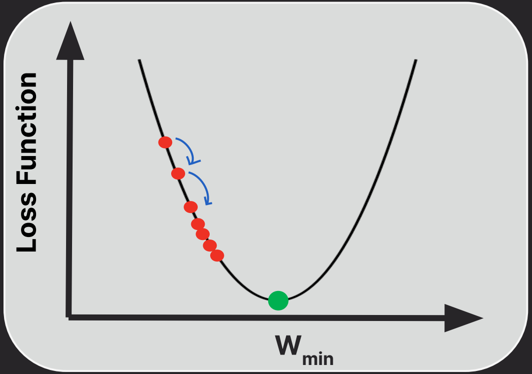
Subtracting large slope can jump over the minimum.

Using a Learning Rate

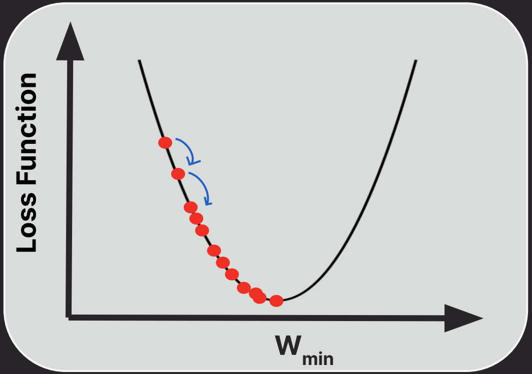
Learning Rate : A small positive number that scales the slope to refine weight updates.

$$w_{new} = w_{old} - \text{Learning Rate} * \text{slope}$$

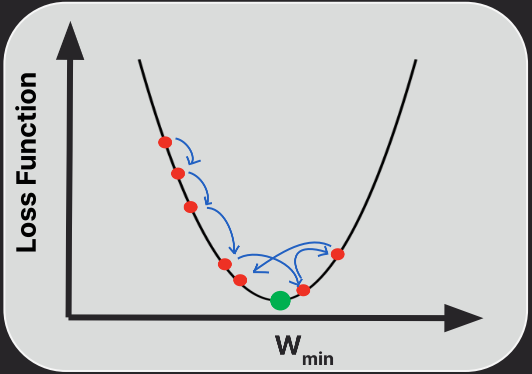
Choosing a Learning Rate



Small Learning Rate



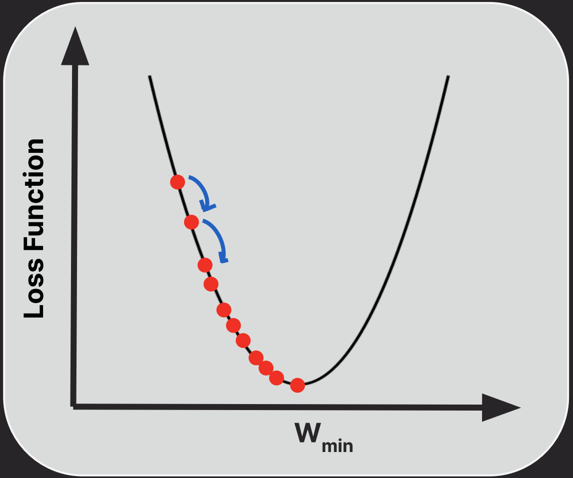
Ideal Learning Rate



Large Learning Rate

Using a Learning Rate

$$w_{new} = w_{old} - \text{Learning Rate} * \text{slope}$$



- **Learning rate = 0.001** (Good starting point to train models)
- This value can be increased or decreased based on observations.

Analytics Vidhya

When to Stop Updating?

1. Stop when $w_{new} = w_{old}$

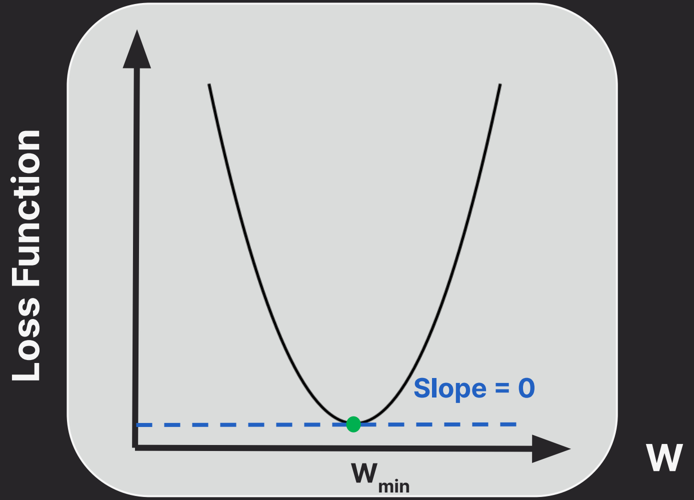
$$w_{new} = w_{old} - \text{learning rate} * \text{slope}$$

↓

$$w_{new} = w_{old} - \text{learning rate} * 0$$

↓

$$w_{new} = w_{old}$$



When to Stop Updating?

Analytics Vidhya

2. Limit number of iterations

Ex: Limiting iterations to 1000

The Math Behind Gradient Descent

Analytics Vidhya

Find the derivative of the slope with respect to the weight w

$$L = \frac{1}{n}\Sigma_{i=1}^{n}(\hat{y}i - yi)^2$$

$$\hat{y} = \omega_i^{*} x_i + b$$

x_i = features from dataset

Analytics Vidhya

Chain Rule of Differentiation

Chain Rule:

Derivative of a composite function is the product of the individual derivatives of the composite function.

$$L = \frac{1}{n}\Sigma_{i=1}^{n}(\hat{y}i - yi)^2$$

$$\hat{y} - y = u$$

$$L = \frac{1}{N}\sum u^2$$

Chain Rule of Differentiation

$$\frac{\mathrm{d}L}{\mathrm{d}u} = \frac{1}{N}\frac{\mathrm{d}\left(u^2\right)}{\mathrm{d}u}$$

$$\frac{dL}{du} = \frac{2u}{N}$$

Chain Rule of Differentiation

$$\frac{dL}{d\hat{y}} = \frac{du}{d\hat{y}} * \frac{dL}{du}$$

How u impacts the **loss value** and then how $\hat{y}$ impacts u.

Chain Rule of Differentiation

Analytics Vidhya

$$\frac{du}{d\hat{y}} = \frac{d}{d\hat{y}}(\hat{y} - y) = 1$$ 1

$$\frac{dL}{du} = \frac{2u}{N}$$ 2

$$\frac{dL}{d\hat{y}} = \frac{du}{d\hat{y}} * \frac{dL}{du} \rightarrow \frac{dL}{d\hat{y}} = \frac{2u}{N}$$ Combining 1 & 2

Chain Rule of Differentiation

$$y_i = w . x_i + b$$

$$\frac{dL}{dw} = \frac{dL}{d\hat{y}} * \frac{d\hat{y}}{dw} \quad (1)$$

$$\frac{dL}{db} = \frac{dL}{d\hat{y}} * \frac{d\hat{y}}{db} \quad (2)$$

Chain Rule of Differentiation

$$\frac{dL}{dw} = 2u.x_i \quad or \quad 2(\hat{y} - y) \cdot x_i$$

$$2((w^*x + b) - y \;\; \cdot x_i)$$

Chain Rule of Differentiation

$w_{new} = w_{old}$ - learning rate * slope

$b_{new} = b_{old}$ - learning rate * slope

$$\frac{dL}{dB} = 2u \quad = 2\,(w_{old} * x + b) - y$$

Backpropagation

Find how weights in earlier layers affect the loss function in a multi neural network.