Calculating Derivatives to Update the Parameters (B, W_1, W_2)

Understanding the mathematical foundation of parameter optimization in neural networks

Σ

Objective

Calculate the derivatives of the Loss function (L) with respect to the parameters:

B, W_1, W_2

- Update parameters to minimize the loss
- Optimize model performance
- Enable gradient descent algorithm

$\partial L/\partial B = ?$

$\partial L/\partial W_1 = ?$

$\partial L/\partial W_2 = ?$

Calculation Steps Overview

1 **Derivative of Loss w.r.t. predicted value (a)**

Calculate $\partial L/\partial a = -y/a + (1-y)/(1-a)$

2 **Derivative of (a) w.r.t. (z)**

Calculate $\partial a/\partial z = a \cdot (1 - a)$

3 **Derivative of Loss w.r.t. (z)**

Calculate $\partial L/\partial z = (a - y)$ using chain rule

4 **Derivative of (z) w.r.t. parameters**

Calculate $\partial z/\partial w_1$, $\partial z/\partial w_2$, $\partial z/\partial b$

5 **Final derivatives of Loss w.r.t. parameters**

Calculate $\partial L/\partial w_1$, $\partial L/\partial w_2$, $\partial L/\partial b$

Chain Rule Application

$$\partial L/\partial w_1 = \partial L/\partial z \cdot \partial z/\partial w_1$$

$$\partial L/\partial w_2 = \partial L/\partial z \cdot \partial z/\partial w_2$$

$$\partial L/\partial b = \partial L/\partial z \cdot \partial z/\partial b$$

Parameter Update

$$w_1 := w_1 - \alpha \cdot \partial L/\partial w_1$$

$$w_2 := w_2 - \alpha \cdot \partial L/\partial w_2$$

$$b := b - \alpha \cdot \partial L/\partial b$$

Derivative of Loss w.r.t. Predicted Value (a)

First step in calculating parameter derivatives

$$\partial L/\partial a = -y/a + (1-y)/(1-a)$$

Formula Explanation

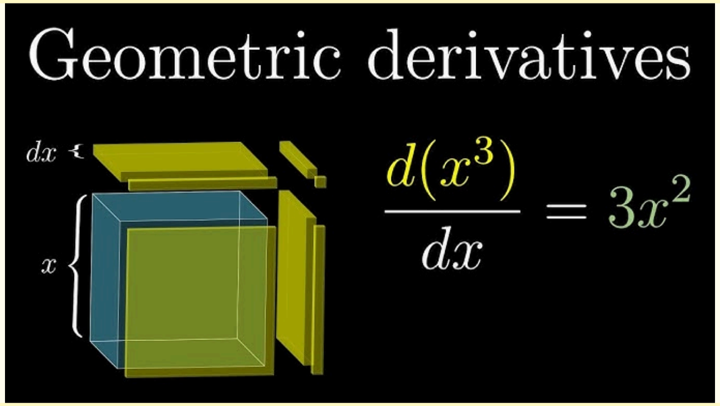
- L = Loss function
- y = True value (ground truth)
- a = Predicted value from model
- Measures how loss changes as prediction changes

Derivative of (a) w.r.t. (z)

Calculating the derivative of the sigmoid activation function

$$a = 1/(1 + e^{-z})$$

$$\partial a/\partial z = a \cdot (1 - a)$$



Key Points

- a = Output of sigmoid function
- z = Input to sigmoid function
- Sigmoid derivative has elegant property: depends only on output
- Maximum derivative at a = 0.5 (z = 0)

Derivative of Loss w.r.t. (z)

Applying the chain rule to connect loss to activation input

Chain Rule:

$$\partial L/\partial z = \partial L/\partial a \cdot \partial a/\partial z$$

$$= [-y/a + (1-y)/(1-a)] \cdot [a \cdot (1-a)]$$

$$\partial L/\partial z = (a - y)$$

Key Insights

- Complex expression simplifies to $(a - y)$
- Result is simply the **difference** between predicted and true values
- When $a > y$, gradient is positive (increase z)
- When $a < y$, gradient is negative (decrease z)

Derivative of (z) w.r.t. Parameters

Calculating how z changes with each parameter

$$z = w_1x_1 + w_2x_2 + b$$

$$\partial z/\partial w_1 = x_1$$

$$\partial z/\partial w_2 = x_2$$

$$\partial z/\partial b = 1$$

w_1	w_2	b
x_1	x_2	1

Key Insights

- z = Linear combination of inputs and weights
- Weight derivatives equal corresponding input values
- Bias derivative is 1 (constant term)
- These derivatives connect parameters to network output

Final Derivatives of Loss w.r.t. Parameters

Using chain rule to complete the derivative calculations

Chain Rule:

$$\partial L/\partial w_1 = \partial L/\partial z \cdot \partial z/\partial w_1$$

$$\partial L/\partial w_2 = \partial L/\partial z \cdot \partial z/\partial w_2$$

$$\partial L/\partial b = \partial L/\partial z \cdot \partial z/\partial b$$

$\partial L/\partial w_1$	$\partial L/\partial w_2$	$\partial L/\partial b$
$(a - y) \cdot x_1$	$(a - y) \cdot x_2$	$(a - y)$

Key Insights

- All derivatives share the **(a - y)** term
- Weight derivatives scaled by **input values**
- Bias derivative remains **unscaled**
- These derivatives directly guide parameter updates

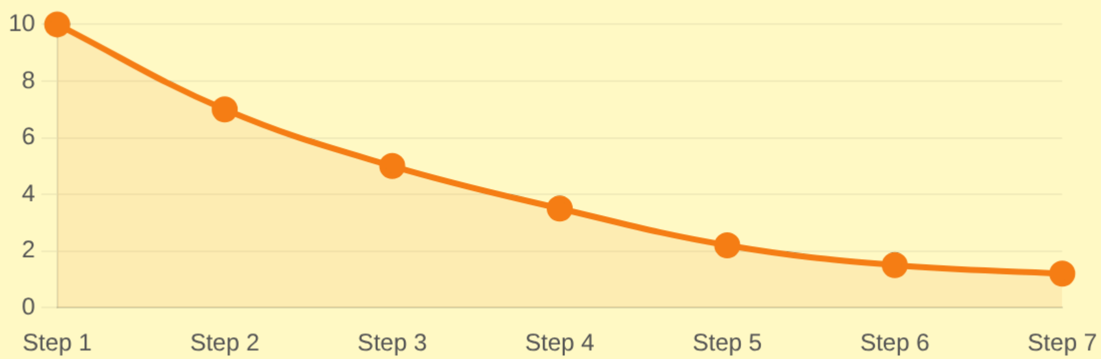
Usage in Gradient Descent Algorithm

Applying derivatives to update model parameters

$$w_1 := w_1 - \alpha \cdot \partial L/\partial w_1$$

$$w_2 := w_2 - \alpha \cdot \partial L/\partial w_2$$

$$b := b - \alpha \cdot \partial L/\partial b$$



α

Learning Rate

:=

Assignment

Gradient Descent Process

- Move parameters in **opposite direction** of gradient
- **Learning rate (α)** controls step size
- Repeat until convergence to **minimum loss**
- Final update formulas:

w_1 $:= w_1 - \alpha \cdot (a-y) \cdot x_1$

w_2 $:= w_2 - \alpha \cdot (a-y) \cdot x_2$

b $:= b - \alpha \cdot (a-y)$