

# Back Propagation

---

**Backpropagation** (short for *backward propagation of errors*) is an algorithm used to train artificial neural networks. It helps the network learn by **minimizing the difference between predicted outputs and actual outputs** (i.e., the error) using **gradient descent**.

In simple words:

- ❑ The network makes a prediction.
- ❑ The error is calculated.
- ❑ This error is propagated backward through the network to update the weights.
- ❑ The weights are adjusted so that the error reduces in future predictions.

Backpropagation is essential for **supervised learning** in deep networks.

**Gradient Descent** — the optimization method used in backpropagation to adjust the weights of a neural network.

**In simple terms:**

- Gradient descent calculates the **slope (gradient)** of the loss function with respect to each weight.
- It then moves the weights in the **opposite direction of the gradient** (downhill) to reduce the loss.
- The step size is controlled by the **learning rate ( $\eta$ )**.

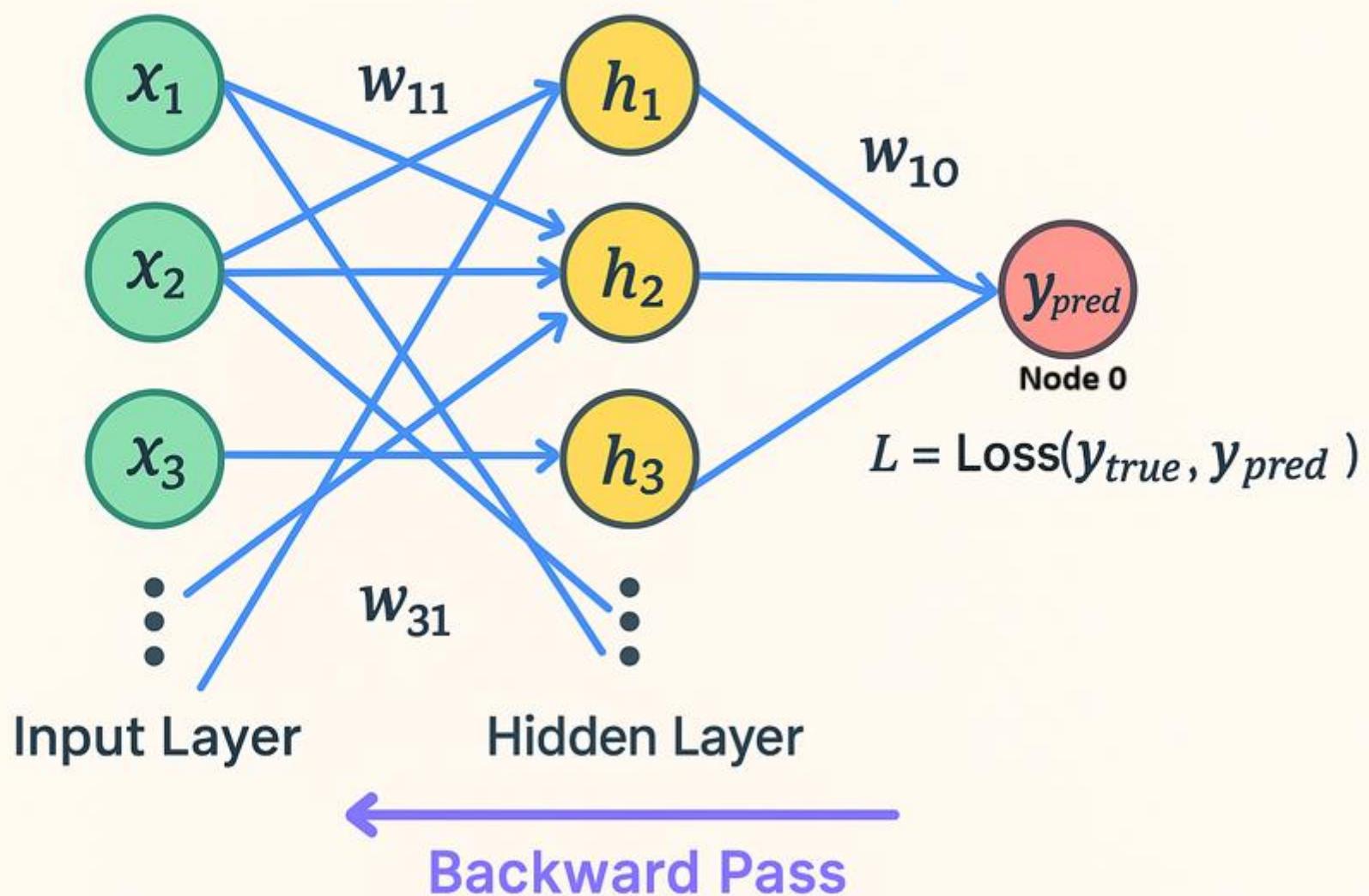
**In backpropagation, the gradients are computed using the chain rule, and gradient descent updates the weights to make the network learn.**

## 2. How Backpropagation Works:

A neural network consists of:

- Input layer** – receives input data
- Hidden layers** – process data with weights and activations
- Output layer** – produces the final prediction

## Forward Pass



**Step-by-Step Process of back propagation is as follows:**

## Step 1: Forward Pass

- Input features are fed into the network.
- Each neuron calculates its **weighted sum**:

$$z = \sum(w_i \cdot x_i) + b$$

- Then applies an **activation function**  $a = f(z)$ .
- Output  $y_{pred}$  is generated.

## Step 2: Compute Loss

- The loss function measures how far the prediction is from the actual value:

$$L = \text{Loss}(y_{true}, y_{pred})$$

Common loss functions:

- Mean Squared Error (MSE) for regression
- Cross-Entropy for classification

## **Step 3: Backward Pass (Backpropagation)**

The goal is to **update the weights** to minimize the loss.

**Key idea:** Use **gradient descent** to adjust weights in the direction that reduces error.

## Steps in Backpropagation:

1. Compute the derivative of the loss with respect to the output:

$$\frac{\partial L}{\partial y_{pred}}$$

2. Compute the derivative with respect to each weight using the **chain rule**:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial y_{pred}} \cdot \frac{\partial y_{pred}}{\partial z} \cdot \frac{\partial z}{\partial w}$$

3. Update weights using gradient descent:

$$w_{new} = w_{old} - \eta \frac{\partial L}{\partial w}$$

Where  $\eta$  is the **learning rate**.

This process continues **layer by layer**, moving backward from output to input.

## **Step 4: Repeat**

- Forward pass → compute loss → backward pass → update weights
- Repeat until the network converges (loss becomes minimal).

# Summary

- ❑ **Backpropagation** is the backbone of training neural networks.
- ❑ Uses **forward pass** to compute outputs.
- ❑ Uses **loss function** to compute error.
- ❑ Uses **backward pass with chain rule** to update weights.
- ❑ Iteratively reduces the loss until the network learns.