Lecture 7
Backpropagation

Contents

1 What is Backpropagation?

Backpropagation (short for "backward propagation of errors") is the core learning algorithm used in training neural networks. It's how the network figures out which of its weights were responsible for the mistakes it made—and how to adjust them to improve next time.

In simple terms: Backpropagation = Learning from mistakes.

Every time your network makes a prediction, it compares its output with the true (target) value, calculates an error, and then propagates that error backward through the network to update its internal parameters.

2 Why Do We Need to Train a Neural Network?

When we first build a neural network, we assign **random weights** to the connections between neurons. These weights control how strongly one neuron influences another.

Think of weights as the *importance level* the network gives to each input feature.

- If the weight from "glucose level" to the next neuron is high, the model believes glucose has a strong effect on the output.
- If it's low, the model treats glucose as less important.

Initially, these weights are random, so the network's predictions are basically guesses. **Training** means tuning these weights so that the model's predictions get closer and closer to the correct answers.

3 The Training Objective

After every prediction, we check how far the model's output is from the ground truth (the real answer). This difference is quantified using a **loss function** (such as Mean Squared Error or Cross-Entropy Loss).

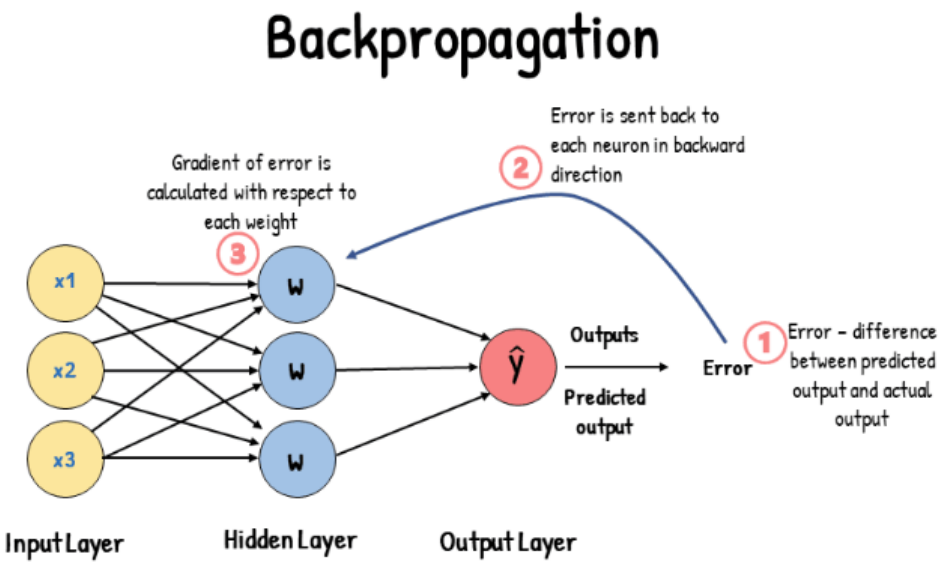
Goal: Minimize this loss by adjusting the weights.

But here's the catch—there are often millions of weights, and each affects the final output differently. We need a systematic way to know which weights to change and by how much.

That's where **backpropagation** comes in.

4 The Intuition Behind Backpropagation

1. **Forward Pass:** Inputs pass through the network layer by layer to generate an output.
2. **Loss Calculation:** The output is compared with the target to compute the error.



3. **Backward Pass:** Backpropagation uses calculus (specifically, the chain rule) to figure out how the error changes with respect to each weight:

$$\frac{\partial \text{Loss}}{\partial \text{Weight}}$$

This tells us the direction and magnitude of change needed.

4. **Weight Update (Gradient Descent):** Once we know the gradients, we adjust the weights slightly in the opposite direction of the gradient (since we want to reduce error):

$$w_{\text{new}} = w_{\text{old}} - \eta \times \frac{\partial \text{Loss}}{\partial w}$$

where η (eta) is the **learning rate**—a small step size that controls how fast we learn.