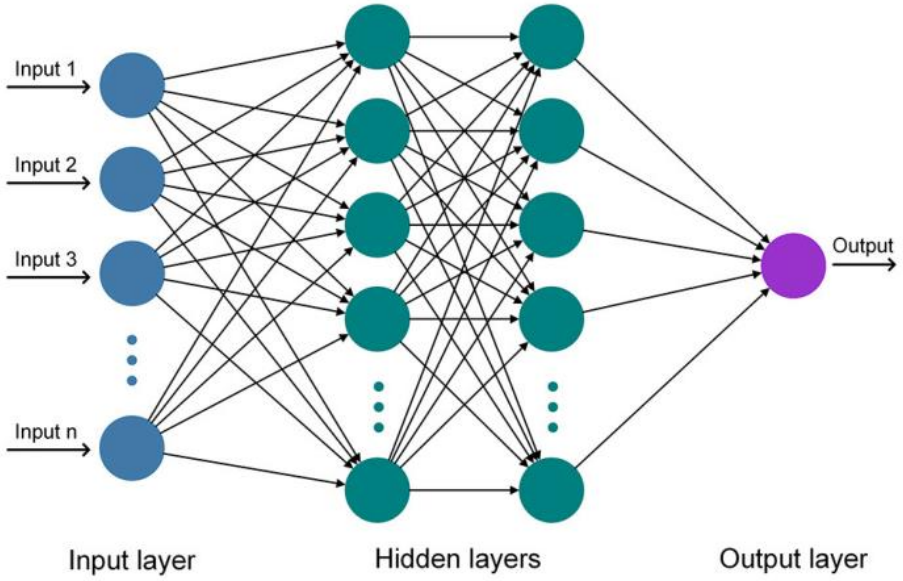
Artificial Neural Networks (ANN) – Summary

Introduction to ANN

Artificial Neural Networks (ANN), also known **as Fully Connected Neural Networks** or **Multi-Layer Perceptrons** (MLPs), consist of interconnected neurons organized in layers: input, hidden, and output. Each neuron in one layer is connected to every neuron in the next layer, forming a dense network. ANN is commonly used for classification and regression tasks.

The input layer receives data from the external world (dataset), which the network analyzes. This data flows through one or more hidden layers, and finally, the output layer produces the result.



Weights and Biases

Each input is associated with a weight:

When you feed input features into an ANN, each input x_1, x_2, ..., x_n is multiplied by a corresponding weight w_1, w_2, ..., w_n. These weights determine the importance of each input feature in making predictions. Higher weight means more influence; lower weight means less.

Bias in ANN:

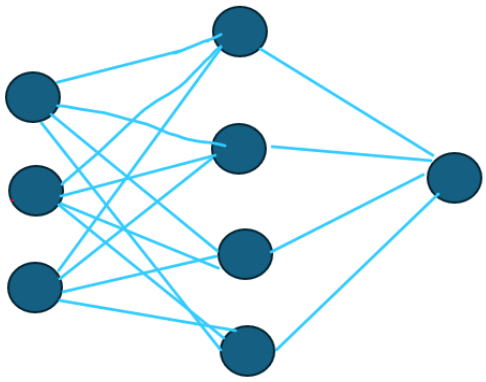
Each neuron (excluding input neurons) has one bias term. Hence, the number of biases equals the number of neurons in a layer.

Example :

- Input layer: 3 features

- Hidden layer: 4 neurons

- Output layer: 1 neuron



→ Input-to-hidden weights: 3 × 4 = 12 weights

→ Hidden layer biases: 4

→ Hidden-to-output weights: 4 × 1 = 4 weights

→ Output layer bias: 1

→ Total trainable parameters = 12 + 4 + 4 + 1 = 21

Forward Propagation (How Prediction Happens)

Input features go into the network (e.g., age, weight). Each input is multiplied by a weight, a bias is added, and the result is passed through an activation function like ReLU or Sigmoid. The output of layer became the input for next layer.This process continues layer by layer until the final output is produced. This flow of computation is called forward propagation.

Backpropagation (How the Model Learns)

After prediction, the model calculates the error using a loss function (e.g., MSE, MAE). It then moves backward through the network to update weights and biases, adjusting them to reduce the error. This process repeats over many iterations (epochs) to improve accuracy.

Gradient Descent (How Weights Are Updated)

Gradient Descent is an algorithm used to find the optimal values of weights and biases by minimizing the loss function.

Steps:

1. Start with initial values (often w = b = 0)

2. Iteratively update w and b to reduce the loss until the optimal minima is reached

Update Rules:

$$w = w_o - \alpha \times (\partial L/\partial w)$$

$$b = b_o - \alpha \times (\partial L/\partial b)$$

Where:

- L = Loss function

- α = Learning rate

- ∂L/∂w and ∂L/∂b = Gradients (partial derivatives)

Gradient of Loss

The gradient of the loss indicates the direction and magnitude by which each weight and bias should be adjusted to minimize the error.

Learning Rate (α)

The learning rate controls the size of steps taken during training:

- Too small → training is very slow

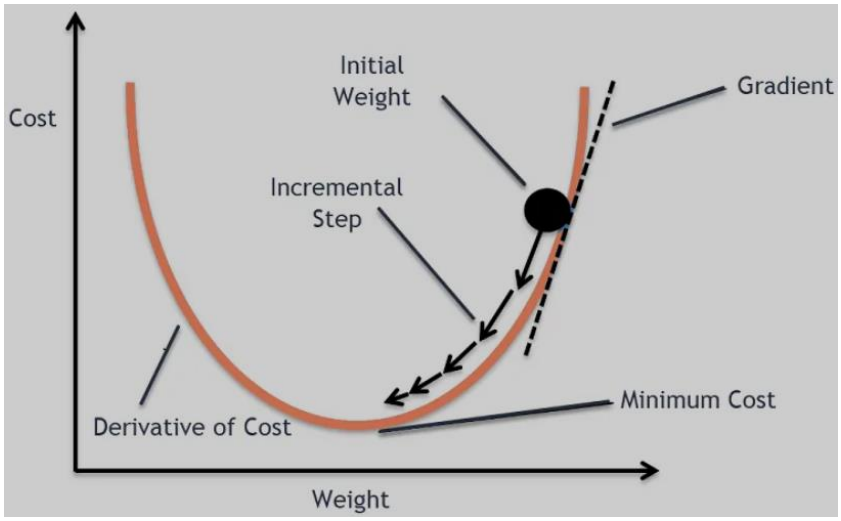
- Too large → might overshoot the minima and diverge

So start with larger step and gradually decrease the size of steps (towards minima).

Minima and Optimal Minima

- Minima: A low point on the loss curve (low error)

- Optimal Minima: The point where the model performs best on training data (may be global or local)


Cost
Initial
Weight
Gradient
Incremental
Step
Minimum Cost
Derivative of Cost
Weight