Unit - III Artificial Neural Networks

3.1 Introduction of Artificial Neural Networks(ANN)

What is an ANN?

Inspired by the biological human brain. A subset of Machine Learning (Deep Learning). To simulate human decision-making and pattern recognition. Adaptive system (learns by example, not just rules).

Biological vs. Artificial

Biological Unit: Neuron.
Artificial Unit: Perceptron (or Node).
Signal Transmission:

- **Dendrites:** Accept input.
- **Soma:** Processes input.
- **Axon:** Sends output.
- **Synapse:** Connection gap (determines signal strength).

Mapping to ANN:

- **Dendrites → Input Layer**.
- **Synapse → Weights**.
- **Axon → Output**.

The Building Blocks (Components)

Inputs (x): The raw data fed into the network.
Weights (w):

- Represents connection strength.
- High weight = High importance.
- Adjustable (this is what the network "learns").

Bias (b):

- Extra input value.
- Allows shifting of the activation curve.
- Prevents zero-output issues.

Summation (Σ):

- Calculates the total input.

- Formula: $\Sigma\,(Input\ *\ Weight)\ +\ Bias$

Activation Function:

- Decides if a neuron "fires".
- Adds non-linearity.
- Examples: Sigmoid, ReLU.

The Architecture (Layers)

Input Layer:

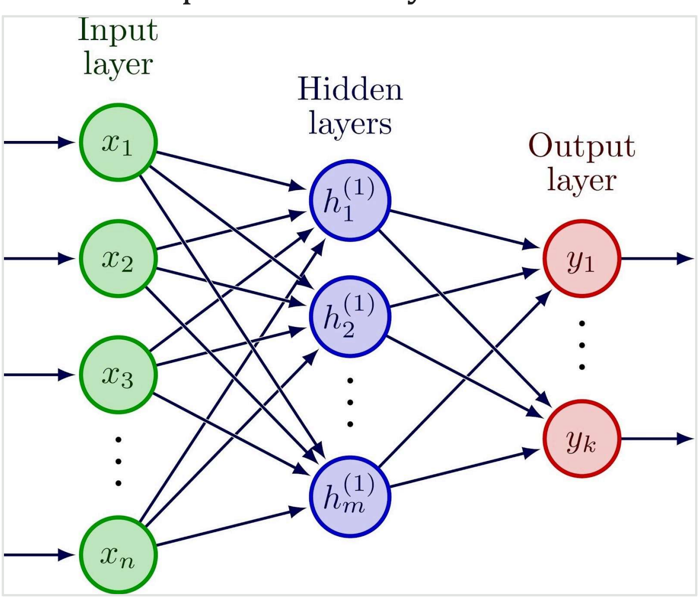
- First layer.
- Passive (no math happens here).
- Passes data to hidden layers.

Hidden Layer:

- Middle layer(s).
- The "Black Box".
- Performs calculations.
- Extracts features (edges, shapes, patterns).

Output Layer:

- Last layer.
- Gives the final result.
- Example: Probability of "0.8" or Class "Cat".



3.2 Perceptron : Basic Components, working, Types ,Training Rule

The Perceptron is the simplest type of artificial neural network, often called a "single-layer" network. It is a linear classifier used for binary classification (0 or 1).

Basic Components

Input Nodes:

Represent the raw data or features. Passed directly to the processing unit.

Weights:

Represent the strength/importance of each connection.

Bias:

An additional parameter (input is always 1). Shifts the decision boundary away from the origin. Ensures the neuron can fire even if all inputs are zero.

Working Mechanism

- Receive: Inputs enter the neuron.
- Weight: Each input is multiplied by its specific weight.
- Sum: The weighted inputs are summed up with the bias.
- Activate: The sum is passed to the Activation Function.
- Decide:
  - If Sum > Threshold → Output = 1.
  - If Sum _< Threshold → Output = 0.

Types of Perceptrons

Single-Layer Perceptron:

Has only Input and Output layers (no hidden layers). Capability: Can only solve linearly separable problems. Examples: Logic gates like AND, OR. Limitation: Fails at XOR problems (non-linear).

Multi-Layer Perceptron (MLP):

Contains one or more Hidden Layers. Uses non-linear activation functions (Sigmoid, ReLU). Capability: Can solve non-linear complex problems. Example: Image classification, XOR gate.

3.3 Gradient Descent Rule, Gradient, Types of Gradient Descent

Gradient Descent Rule

Gradient Descent is an optimization algorithm used to minimize a cost (loss) function by iteratively updating model parameters in the direction of steepest decrease.

Update Rule

$$\theta := \theta - \alpha \nabla J(\theta)$$

Where:

- θ → model parameters
- α → learning rate (step size)
- $J(\theta)$ → cost/loss function
- $\nabla J(\theta)$ → gradient of the cost function

Move parameters **opposite** to the gradient because the gradient points toward the direction of **maximum increase**.

Gradient

The **gradient** is a vector of partial derivatives of the cost function with respect to each parameter.

For parameters $\theta_1, \theta_2, \ldots, \theta_n$:

$$\nabla J(\theta) = \left[\frac{\partial J}{\partial \theta_1}, \frac{\partial J}{\partial \theta_2}, \ldots, \frac{\partial J}{\partial \theta_n}\right]$$

Types of Gradient Descent

Batch Gradient Descent

Uses the entire training dataset to compute the gradient. Update happens once per epoch.

Pros: Stable convergence, Accurate gradient

Cons: Slow for large datasets, High memory usage

Stochastic Gradient Descent (SGD)

Uses one training example at a time. Updates parameters after each sample.

Pros: Faster updates, Works well for large datasets

Cons: Noisy updates, May oscillate around minimum

Mini-Batch Gradient Descent

Uses a small batch of samples (e.g., 32, 64). Compromise between Batch GD and SGD.

Pros: Efficient and stable, Most commonly used in practice

Cons: Requires batch size tuning

3.4 Activation Functions: Sigmoid, ReLU, Hyperbolic tangent, Softmax etc.

Activation Functions

An activation function introduces non-linearity into a neural network, allowing it to learn complex patterns. It decides whether a neuron should be activated or not.

Sigmoid (Logistic) Function

A nonlinear function that maps input values to a range between 0 and 1, often used to represent probabilities.

Hyperbolic Tangent (tanh)

Definition:

A nonlinear activation function that maps inputs to values between **–1 and 1**, making it zero-centered.

ReLU (Rectified Linear Unit)

Definition:

An activation function that outputs **0 for negative inputs** and returns the input value for positive inputs.

Softmax Activation Function

Definition:

An activation function that converts a vector of values into a **probability distribution** whose total sum is 1.