MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING
(A Unit of Rajalaxmi Education Trust®, Mangalore)
Autonomous Institute affiliated to VTU, Belagavi, Approved by AICTE, New Delhi
Accredited by NAAC with A+ Grade & ISO 9001:2015 Certified Institution

Subject:Mathematics for Artificial Intelligence

Subject Code:23AIPC304

Module 3: Neural network Mathematics

The Brain Cortex and Artificial Neural Networks, Training Function: Fully Connected or Dense, Loss Functions, Optimization, Regularization Techniques, Hyperparameters in Machine Learning, Chain Rule and Backpropagation, Assessing the Significance of the Input Data Features.

Text book 3 : Ch 4

Introduction to Neural Network Mathematics

Neural network mathematics:

Neural Network Mathematics refers to the mathematical foundation that governs how artificial neural networks (ANNs) learn, represent, and make decisions. It combines several key areas of mathematics: linear algebra, calculus, probability, statistics, and optimization to model the behavior of interconnected neurons that process data.

How does the neural network exactly work in ML.

A neural network in ML works by repeatedly predicting, measuring how wrong it is, and then adjusting its internal parameters (weights) until it becomes right.

1. Structure of a Neural Network

A neural network consists of:

- **Input layer** : receives data (features)
- **Hidden layers** : perform transformations
- **Output layer** : produces prediction/output

Mathematically, each neuron performs a simple computation:

$$z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

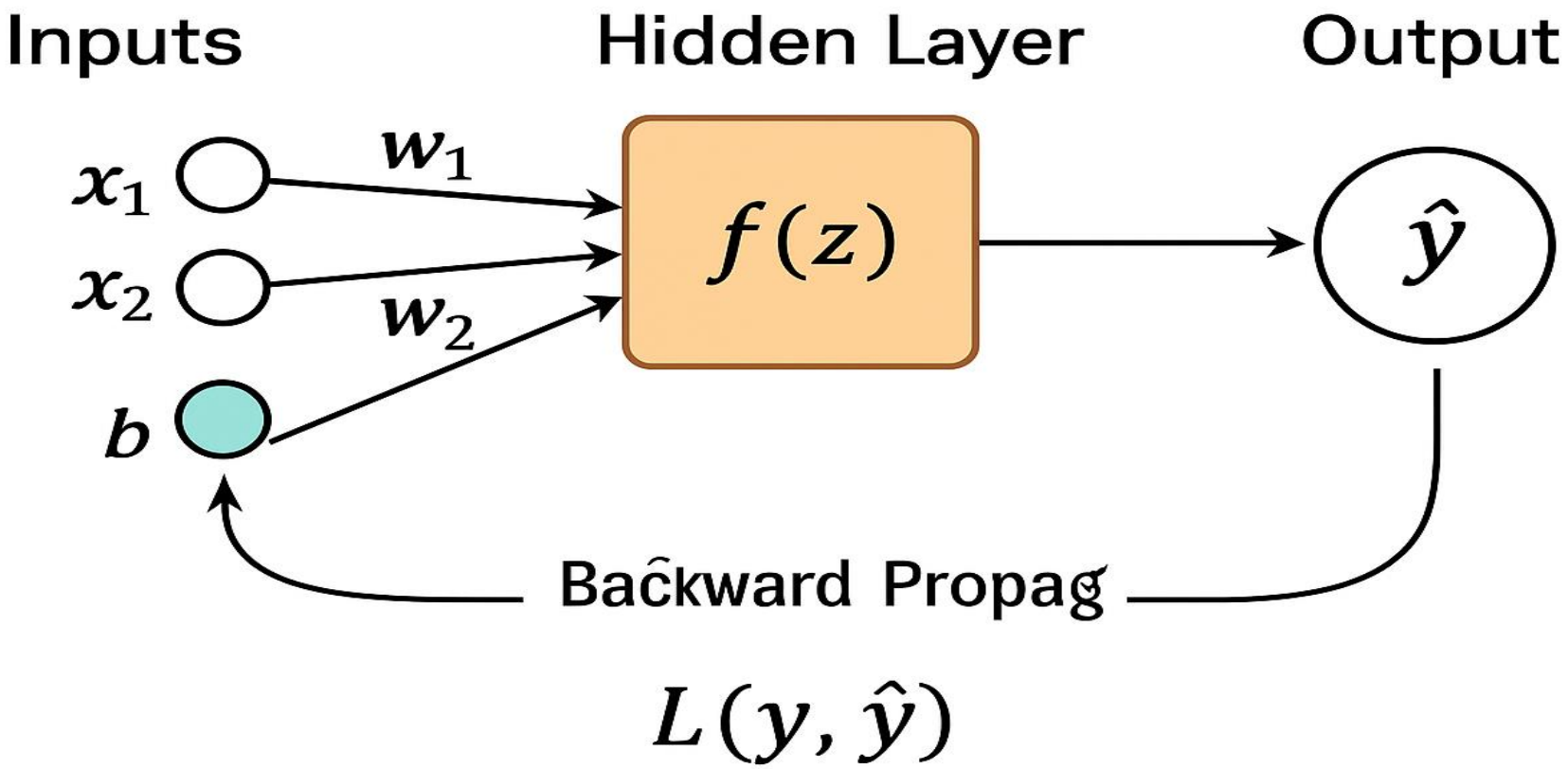
were

- x_i =inputs
- w_i =weights
- b= bias term
- z= weighted sum (pre-activation)

Then an **activation function** $f(z)$is applied to produce the neuron's output:

$$a = f(z)$$

$$z = wx_1 + w_2x_2 + b$$



$$L(y, \hat{y})$$

2. Linear Algebra (Matrix Formulation)

Neural networks are efficiently represented using matrices and vectors.

For one layer:

$$z = Wx + b$$

$$a = f(z)$$

where

- $x \in \mathbb{R}^n$:input vector
- $W \in \mathbb{R}^{m \times n}$:weight matrix
- $b \in \mathbb{R}^m$:bias vector
- $a \in \mathbb{R}^m$:output vector

This matrix form allows multiple neurons to be computed at once.

3. Composition of Functions

A neural network with Llayers is a composition of functions:

$$\mathrm{a}^{(L)} = f^{(L)}\left(\mathrm{W}^{(L)} f^{(L-1)}\left(\ldots f^{(1)}\left(\mathrm{W}^{(1)}\mathrm{x} + \mathrm{b}^{(1)}\right)\ldots\right) + \mathrm{b}^{(L)}\right)$$

This nested structure gives neural networks their nonlinear representational power.

4. Loss Function (Error Measurement)

To train a neural network, we define a loss function $\mathcal{L}$ that measures how far predictions are from true values.

Examples:

Mean Squared Error (Regression) $\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$

Cross-Entropy Loss (Classification) $\mathcal{L} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i)$

5. Calculus (Learning through Gradients)

Learning = Adjusting weights to minimize the loss.

Using calculus, specifically partial derivatives, we compute how changes in weights affect the loss:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w_{ij}}}$$

This is done efficiently using the Backpropagation Algorithm, which applies the Chain Rule of differentiation through the layers.

6. Optimization

Weights are updated using gradient-based optimization algorithms, such as:

$$w_{ij}^{(new)} = w_{ij}^{(old)} - \eta \frac{\partial \mathcal{L}}{\partial w_{ij}}$$

where η = learning rate

Common optimizers:

- Gradient Descent
- Stochastic Gradient Descent (SGD)
- Adam, RMSProp, etc.

7. Probability and Statistics

Used for:

- Modeling uncertainty in predictions
- Initializing weights (from distributions)
- Interpreting outputs (e.g., softmax for probabilities)

Example (Softmax):

$$P(y_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

8. Example

Suppose:

$$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, W = \begin{bmatrix} 0.5 & 0.2 \\ 0.1 & 0.7 \end{bmatrix}, b = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}$$

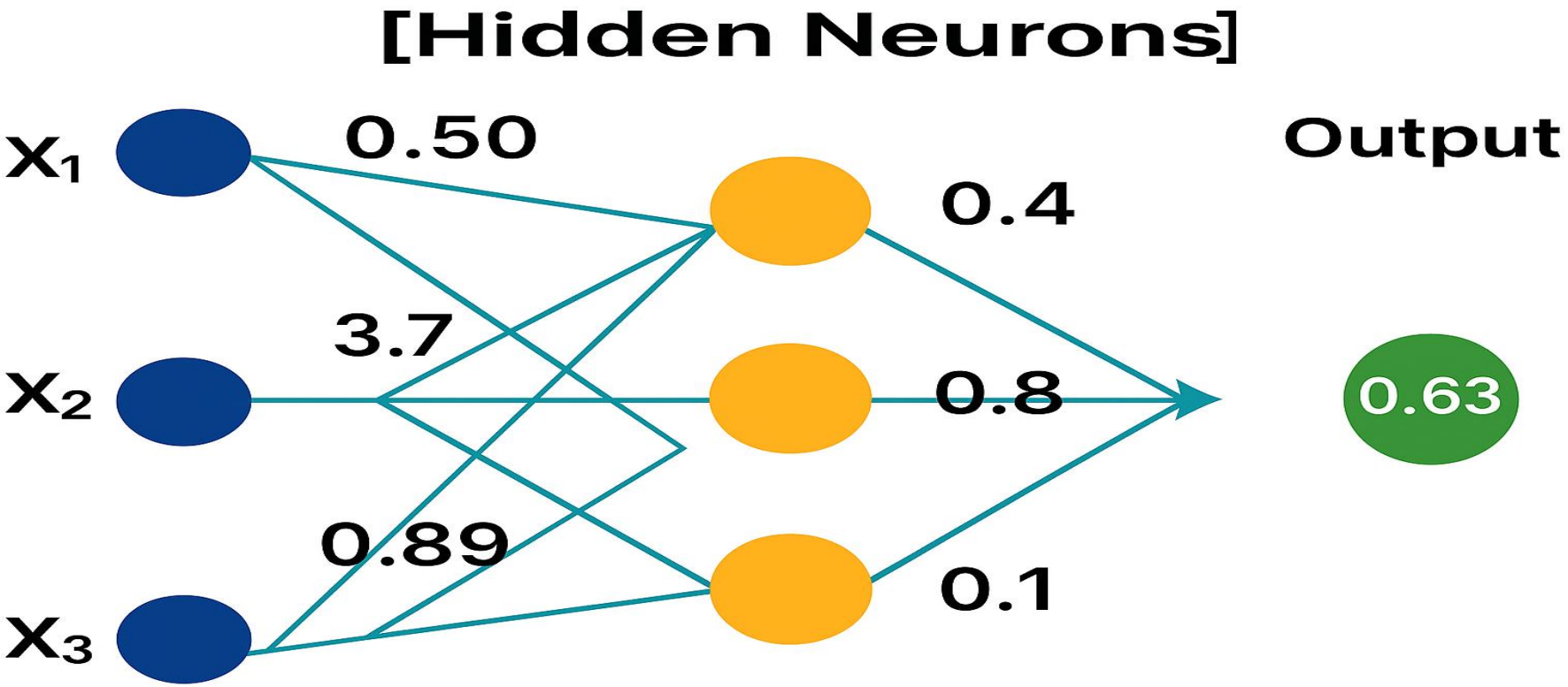
Then:

$$z = Wx + b = \begin{bmatrix} 0.5(1) + 0.2(2) + 0.1 \\ 0.1(1) + 0.7(2) + 0.2 \end{bmatrix} = \begin{bmatrix} 1.0 \\ 1.7 \end{bmatrix}$$

If :

$$a = \begin{bmatrix} 1.0 \\ 1.7 \end{bmatrix}$$

Visual Structure:



The Brain Cortex and Artificial Neural Networks

Brain Cortex

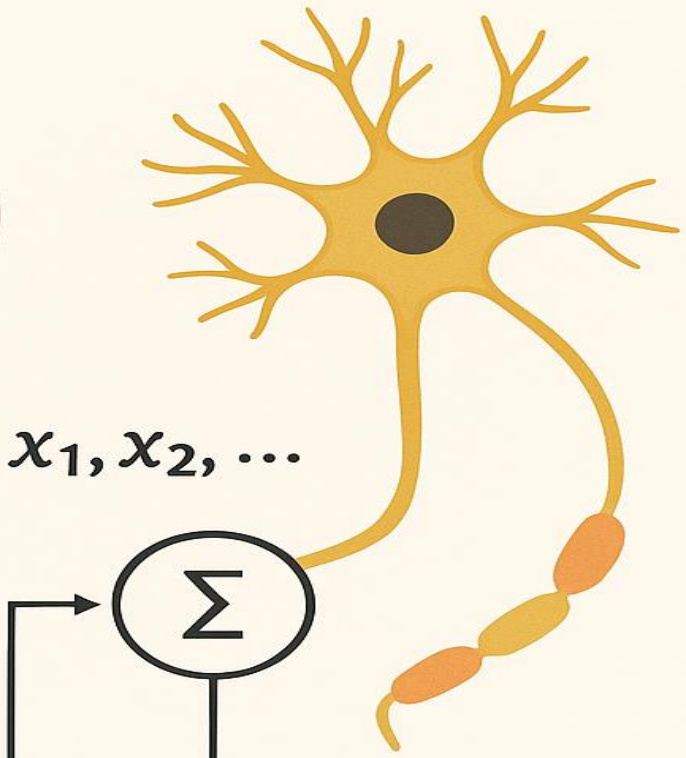
- **Introduction**: The cerebral cortex is the outermost layer of the brain, responsible for higher-level functions such as perception, memory, thought, language, and decision-making. It is made up of billions of interconnected neurons that process and transmit information.
- **Definition**: The cerebral cortex is the folded, grey matter layer of the brain that controls complex cognitive functions by processing sensory inputs, integrating information, and coordinating responses.

Artificial Neural Networks (ANNs)

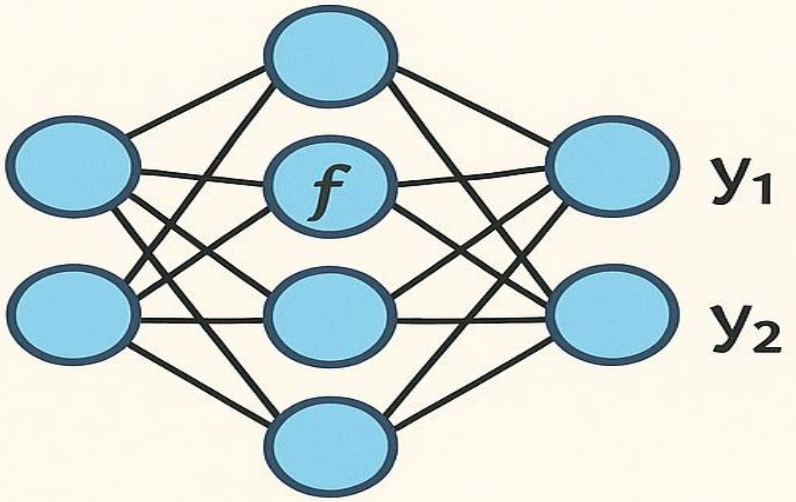
•**Introduction**: Artificial Neural Networks are computational models inspired by the structure and functioning of the human brain, particularly the way neurons in the cerebral cortex interact. They are used in machine learning to recognize patterns, classify data, and make predictions.

•**Definition**: An Artificial Neural Network is a system of interconnected nodes (artificial neurons) arranged in layers that process input data, learn from examples, and produce outputs, mimicking the learning and decision-making of the brain.

BRAIN
CORTEX



NEURON



A

ARTIFICIAL
NEURAL
NETWORK

Question 1: Neuron Activation Calculation

An artificial neuron has 3 inputs:

$x_1 = 2,\ x_2 = -1$ with weights $w_1 = 0.6,\ w_2 = -0.4,\ w_3 = 0.2$ and a basis $b = 1$.

The neuron uses a sigmoid activation function:

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Compute the output of this neuron.

Question 2: Weight Update Using Gradient Descent

A neuron outputs $y = 0.7$ for a given input, and the true target is t = 1. Assume the loss function is Mean Squared Error:

$$L = \frac{1}{2}(t - y)^2$$

The neuron's input was x = 0.5. If the weight associated with this input is w = 0.4 and the learning rate is η = 0.1, compute the updated weight using gradient descent:

Question 3: Simple Feedforward Output

Consider a 2-layer ANN with 2 input neurons, 2 hidden neurons, and 1 output neuron.

Input: x = [1, 2]

Hidden layer weights:

$$W_1 = \begin{bmatrix} 0.5 & -0.5 \\ 0.3 & 0.8 \end{bmatrix}, \qquad b_1 = [0.1, \quad 0.2]$$

Output layer weights: W_2 =[0.7, -0.6], b_2 = 0.05

Assume ReLU(Rectified Linear unit) activation for hidden and linear output.

Compute the final output of the network.

Question 4 : Simple Feedforward Output

Consider a 2-layer Artificial Neural Network (ANN) with:

- 2 input neurons
- 2 hidden neurons
- 1 output neuron

Given: $x = [2, 1]$

Hidden layer weights and biases:

$$W_1 = \begin{bmatrix} 0.4 & -0.7 \\ 0.6 & 0.9 \end{bmatrix}, b_1 = [0.2, 0.1]$$

Output layer weights and bias:

$$W_2 = [0.5, -0.3], b_2 = 0.2$$

Activation functions:

- Hidden layer → ReLU
- Output layer → Linear (no activation)

Compute the final output of the network using the feedforward equations:

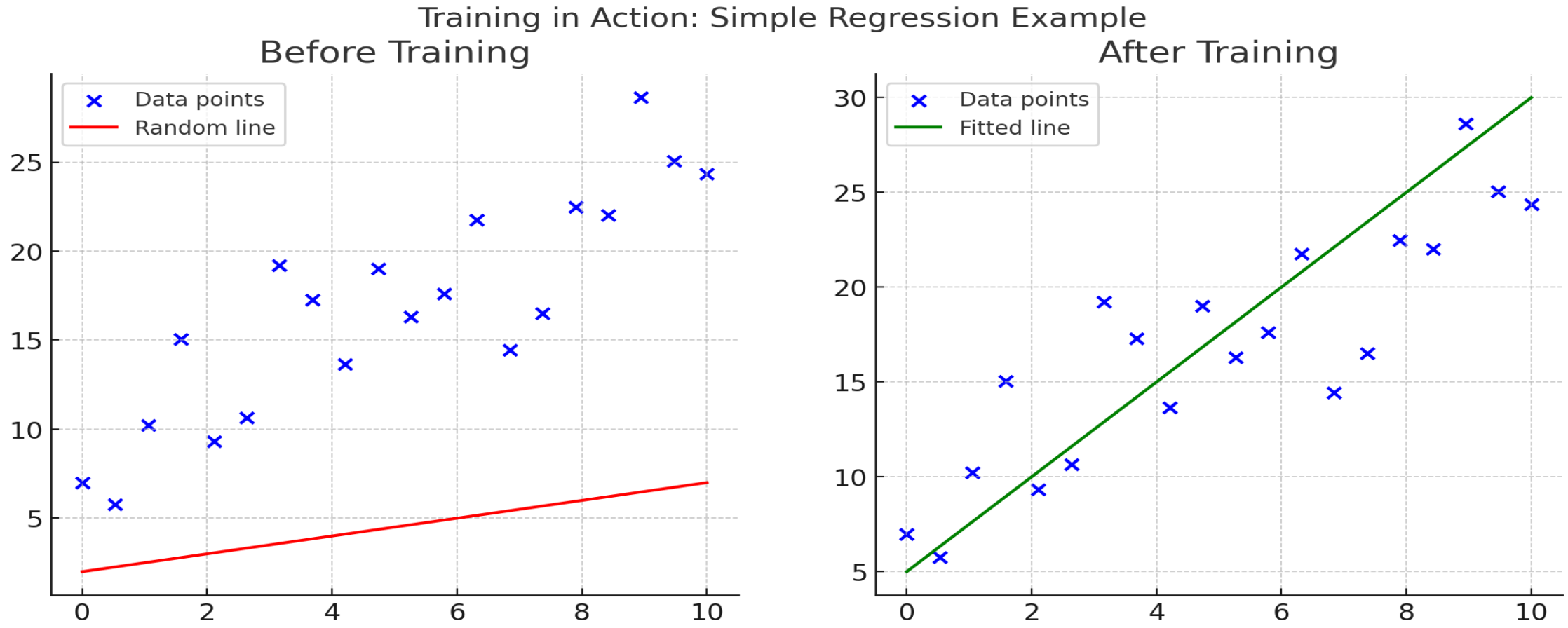
Training Function in Machine Learning

What is a Training Function?

A training function is the process or rule that updates the parameters (weights and biases) of a machine learning model based on data. Its goal is to minimize the error (or loss function) so that the model learns to make accurate predictions.
Think of it as the "learning engine" of an algorithm.

Steps in Training a Model

1.Input Data is fed into the model.
2.Model Prediction is generated.
3.Loss Function calculates the error between prediction and actual value.
4.Training Function (like Gradient Descent) updates the weights to reduce error.
5.Repeat until the error is minimized.



- **Left (Before Training):** The red line is random and doesn't fit the data.
- **Right (After Training):** The green line is well-fitted to the blue data points.

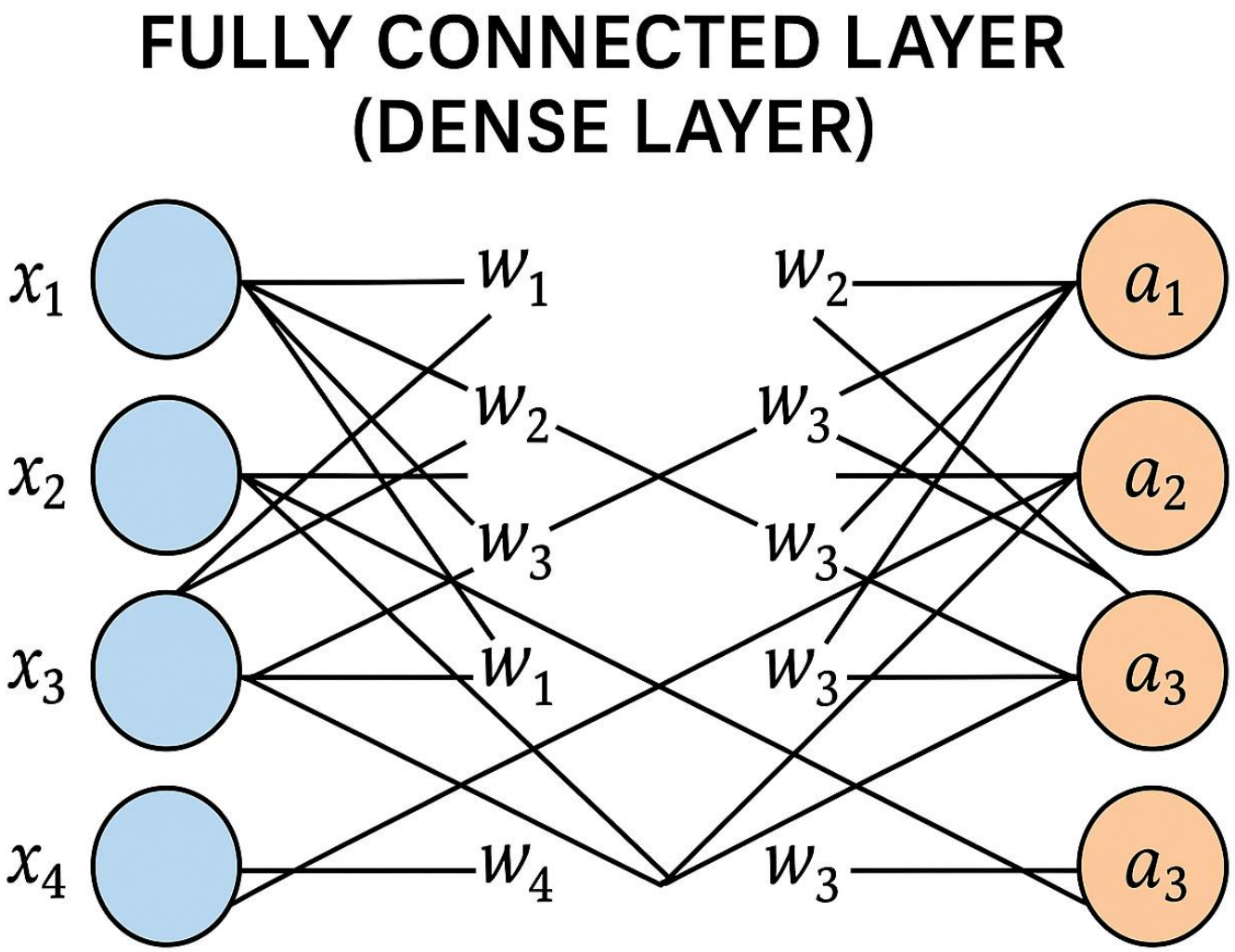
This shows how the **training function** adjusts weights (w) and bias (b) to minimize error.

Fully Connected Layer (Dense Layer)

Introduction:

In Artificial Neural Networks (ANNs), a Fully Connected Layer (also called a Dense Layer) is one where every neuron in one layer is connected to every neuron in the next layer.

It is the most common and fundamental type of layer in neural networks. Fully connected layers allow the model to combine information from all features and learn complex patterns.



Key Points

- Every input neuron connects to every output neuron.
- Bias shifts the activation function to fit the data better.
- Dense layers are mostly used at the **end of CNNs and DNNs** for classification or regression.
- In a neural network, most of the parameters (weights + biases) come from fully connected layers.

Definition:

A Fully Connected Layer is a neural network layer where each neuron receives input from all neurons of the previous layer and produces an output that is passed to all neurons of the next layer.

Mathematically, if the input vector is:

$$x = \{x_1, x_2, \dots\dots, x_n\}$$

Then the output is:

$$y = f(Wx + b)$$

- W → Weight matrix (connections between neurons)
- b → Bias vector
- f → Activation function (e.g., ReLU, Sigmoid, Tanh)

Key Points

All neurons are interconnected.

Useful for pattern recognition and classification tasks.

Found in the final layers of Convolutional Neural Networks (CNNs) and many other models.

Question 1

An input batch $X \in \mathbb{R}^{(20\times 8)}$ (20examples, each with 8 features) is passed through:

A fully connected layer with **10 neurons**, using **ReLU activation**, and

Another fully connected layer with **5 neurons**, using **Sigmoid activation**.

a) Compute the **output dimensions** after each layer.

b) If the total number of trainable parameters (weights + biases) in the first layer is W_1 ,compute W_1.

c) Compute the total number of trainable parameters in the entire network.

Question 2

A dataset has **64 samples**, each with **100 features**. This is passed through a sequence of fully connected layers as follows:

Layer 1: 128 neurons, ReLU activation

Layer 2: 64 neurons, Tanh activation

Layer 3: 16 neurons, Sigmoid activation

a) Calculate the **output dimension** after each layer.
b) Determine the **number of parameters** in each layer (include biases).
c) Compute the **total number of parameters** in the full network.

Question 3

An image dataset is flattened into vectors of size **1024** (i.e., input dimension =1024).
Build a neural network with the following dense layers:
Layer 1: 256 neurons, ReLU activation
Layer 2: 64 neurons, ReLU activation
Layer 3: 10 neurons, Softmax activation

a) Compute the **output dimensions** after each layer for a batch size of 50.
b) Compute the **total number of trainable parameters** (weights + biases) in the network.

Question 4

A neural network is used for **handwritten digit classification** (0 – 9) using the **MNIST** dataset.

Each input image (after flattening) has **784 features** (28 × 28 pixels).

The architecture is:

Layer 1: 128 neurons, ReLU activation

Layer 2: 64 neurons, ReLU activation

Layer 3: 32 neurons, ReLU activation

Layer 4: 10 neurons, Softmax activation

The **batch size** during training is **100**.

a) Compute the **output dimensions** after each layer.

b) Compute the **total number of trainable parameters** (weights + biases) for the entire network.

Loss Functions

Introduction:

In machine learning and deep learning, the ultimate goal of a model is to make accurate predictions. However, models rarely predict perfectly. To quantify how well a model is performing, we need a measure of the difference between the model's predictions and the actual target values. This measure is called a **loss function** (also known as a cost function or error function).

The loss function plays a central role in training machine learning models. During the training process, the model adjusts its parameters (like weights in neural networks) to **minimize the loss**. This minimization guides the model towards better predictions and improved performance.

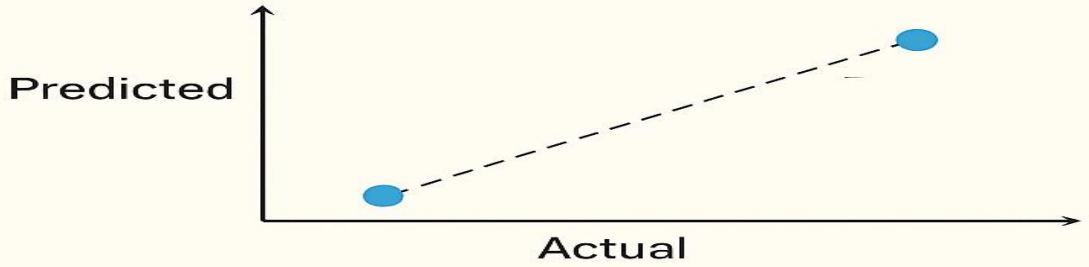
Different types of tasks (regression, classification, etc.) require different loss functions. For instance:

- Regression tasks often use **Mean Squared Error (MSE)** or **Mean Absolute Error (MAE)**.
- Classification tasks often use **Cross-Entropy Loss** or **Hinge Loss**.

In essence, the loss function provides a **numerical feedback** to the model about its prediction accuracy and drives the optimization process.

LOSS FUNCTIONS

In machine learning and deep learning, the ultimate goal of a model is to make accurate predictions. However, models rarely predict perfectly. To quantify how well a model is performing, we need a measure of the difference between the model's predictions and the actual target values. This measure is c called a loss function (also known as a cost function or error function).



Loss function

↓

Loss = Prediction – Actual

The loss function plays a central role in training machine learning models. During the training process, the model adjusts its parameters (like weights in neural networks) to minimize the loss. This minimization guides the model towards better predictions and improved performance.

Regression	Classification
Mean Squared Error (MSE)	Cross-Entropy Loss
Mean Absolute Error (MAE)	Hinge Loss

Definition:

A **loss function** is a mathematical function that measures the discrepancy between the predicted output of a model and the actual target values.

Formally, for a dataset with inputs x_i and corresponding true outputs y_i ,and a model with predictions $\hat{y}_i = f(x_i)$,the loss function L can be defined as:

$$L(y_i\ ,\ \hat{y}_i) = \textit{measure of error between } y_i \textit{ and } \hat{y}_i$$

For the entire dataset, the **total loss** (or cost) is often calculated as the average over all data points:

$$\text{Total } Loss = \frac{1}{n}\sum_{i=1}^{n} L\,(y_i\ ,\ \hat{y}_i)$$

Minimizing this total loss during training is the key principle behind learning in most machine learning algorithms.

Question 1: Mean Squared Error (MSE) Calculation

A regression model predicts the following values for 5 data points:

Data Point	Actual y_i	Predicted $\hat{y}_i$
1	3	2.5
2	7	6.8
3	5	5.2
4	9	8.5
5	4	4.5

a) Calculate the Mean Squared Error (MSE) for the model.

b) Explain in one line whether the model is underestimating or overestimating overall.

Question 2: Mean Absolute Error (MAE) Calculation

A model outputs the following predictions for 6 data points:

Data Point	Actual y_i	Predicted $\hat{y}_i$
1	10	8
2	15	14
3	12	13
4	20	18
5	25	28
6	30	27

a) Compute the Mean Absolute Error (MAE).

b) Compare MAE and MSE: Which is more sensitive to large errors? Justify briefly.

Question 3: Cross-Entropy Loss for Classification

A binary classification model gives predicted probabilities for 4 samples as follows:

Sample	Actual Label y	Predicted Probability $\hat{y}$
1	1	0.9
2	0	0.2
3	1	0.6
4	0	0.7

a). Calculate the binary cross-entropy loss for each sample using:

$$L(y,\ \hat{y}) = -[\text{ylog}(\hat{y}) + (1-y)\log(1-\hat{y})]$$

b). Compute the average loss for all 4 samples.

Question 4: Comparing Two Models Using Loss

Two regression models, Model A and Model B, predict house prices (in lakhs) for 5 houses:

House	Actual Price	Model A Prediction	Model B Prediction
1	50	52	49
2	60	63	58
3	55	54	56
4	70	68	72
5	65	66	64

a) Compute the Mean Squared Error (MSE) for both models.

b) Identify which model is better and explain why using the loss values.

Optimization in Neural Networks

Introduction:

In the context of neural networks, **optimization** refers to the process of adjusting the network's parameters (weights and biases) to minimize a **loss function** (also called a cost function). The loss function quantifies the difference between the network's predicted output and the actual target values.

Mathematically, optimization is the process of finding the set of parameters $\theta = \{W, b\}$ that **minimizes the loss function** $L(\theta)$ .This ensures that the neural network learns the underlying patterns in the data and generalizes well to unseen data.

Optimization is central to **training neural networks**. Techniques like **Gradient Descent**, **Stochastic Gradient Descent**, and more advanced methods like **Adam, RMSProp, and AdaGrad** are widely used to iteratively update network parameters in the direction that reduces the loss.

Definition :

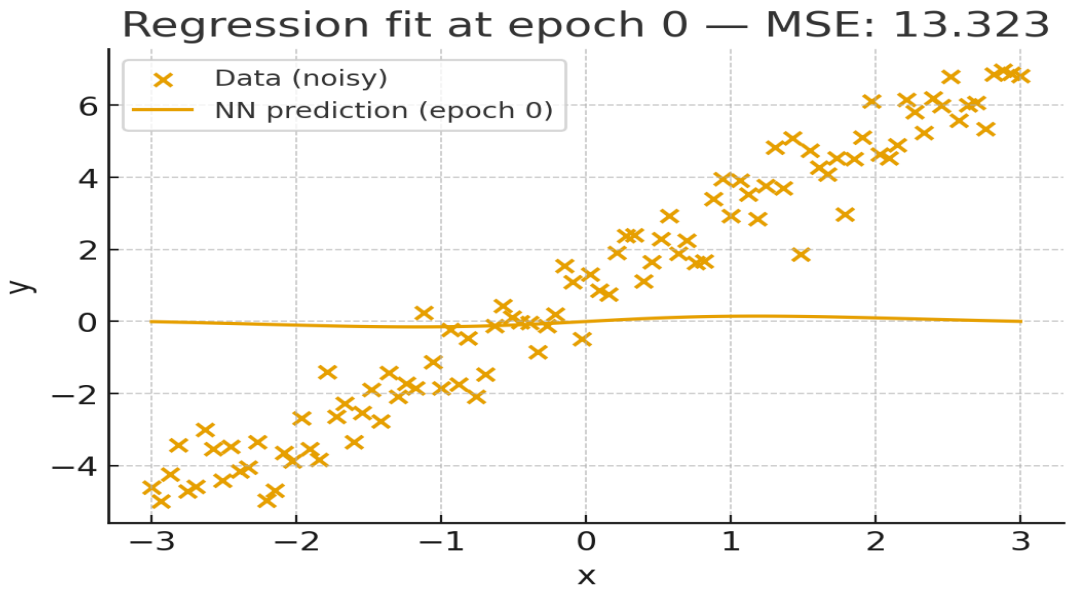
Optimization in neural networks is the mathematical procedure of finding the optimal set of parameters θ^* that minimizes a given loss function $L(\theta)$:

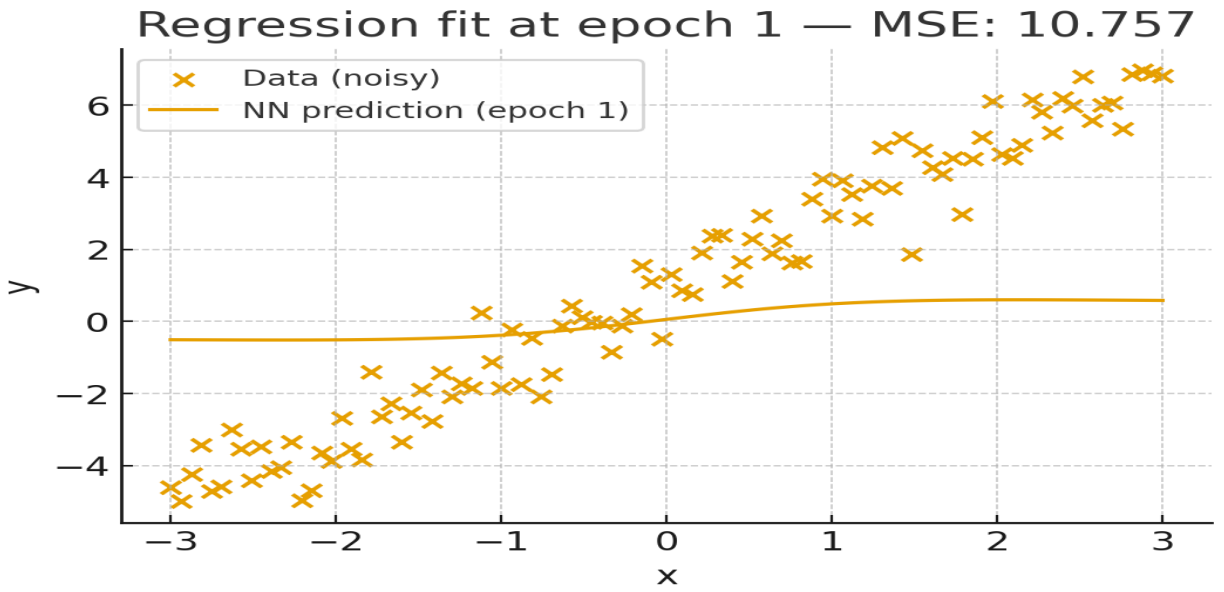
$$\theta^* = \arg\min_{\theta} L(\theta)$$

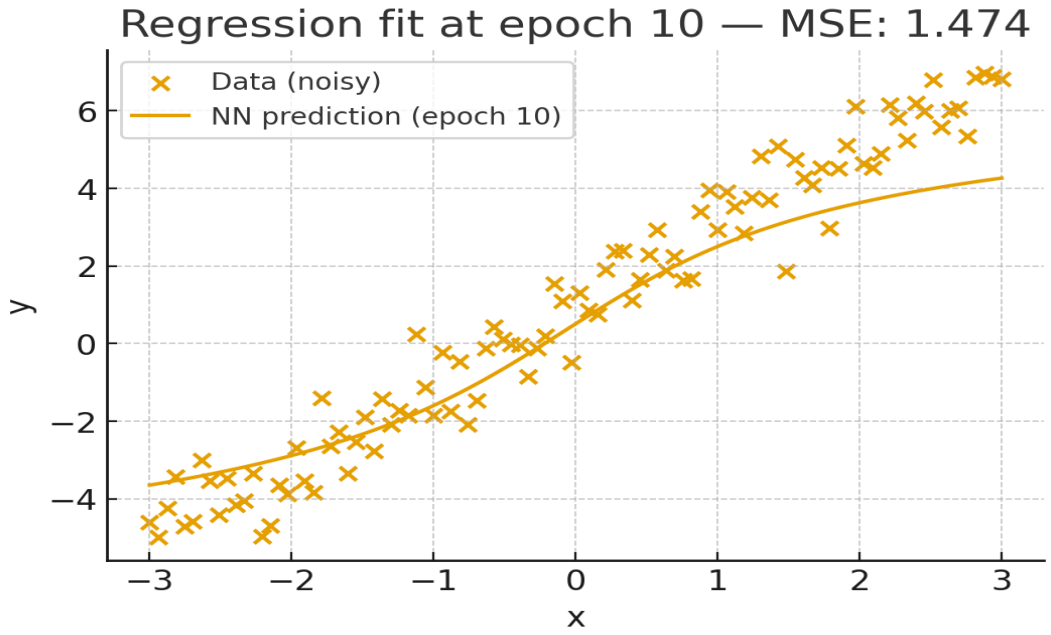
where:

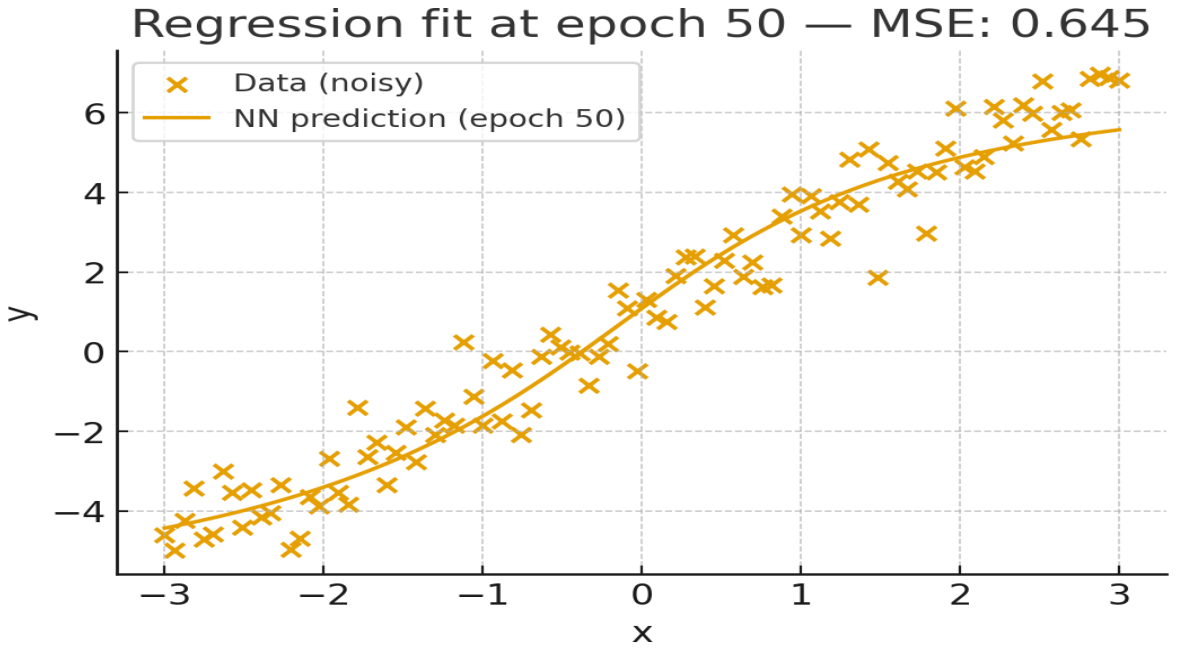
- $\theta = \{W, b\}$ represents the weights and biases of the network.
- $L(\theta)$ measures the error between predicted outputs and actual outputs.
- θ^* represents the optimal parameters that give the best performance of the neural network.

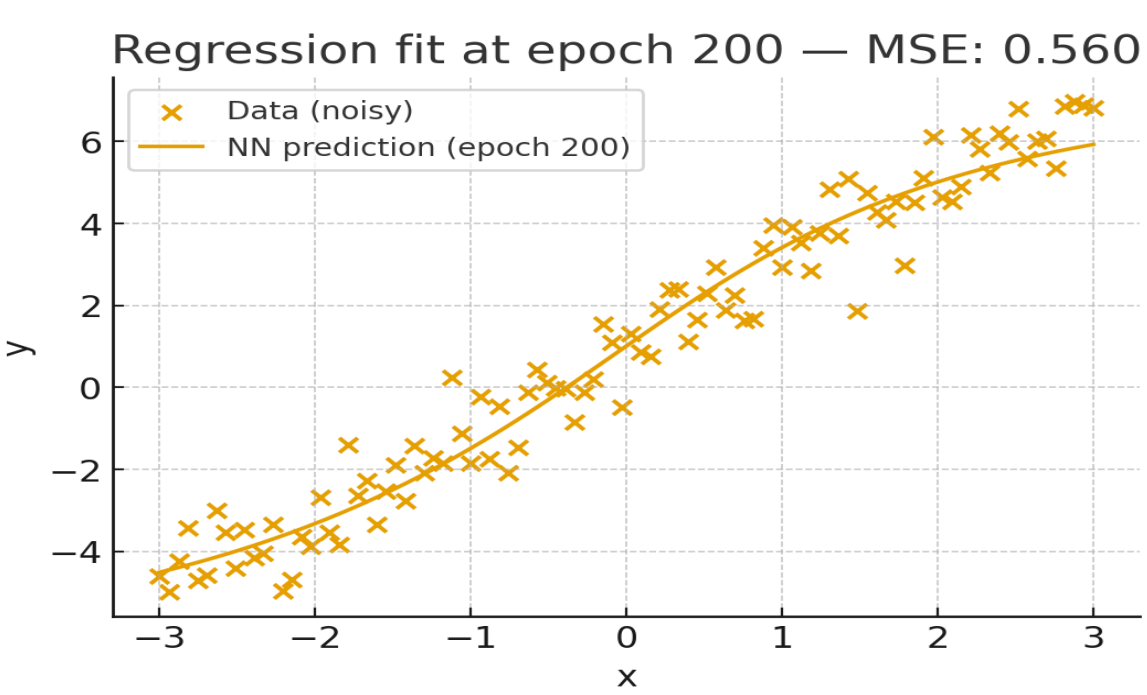
Key Idea: Optimization transforms learning into a **mathematical minimization problem**, where the network iteratively adjusts its parameters to reduce the prediction error.

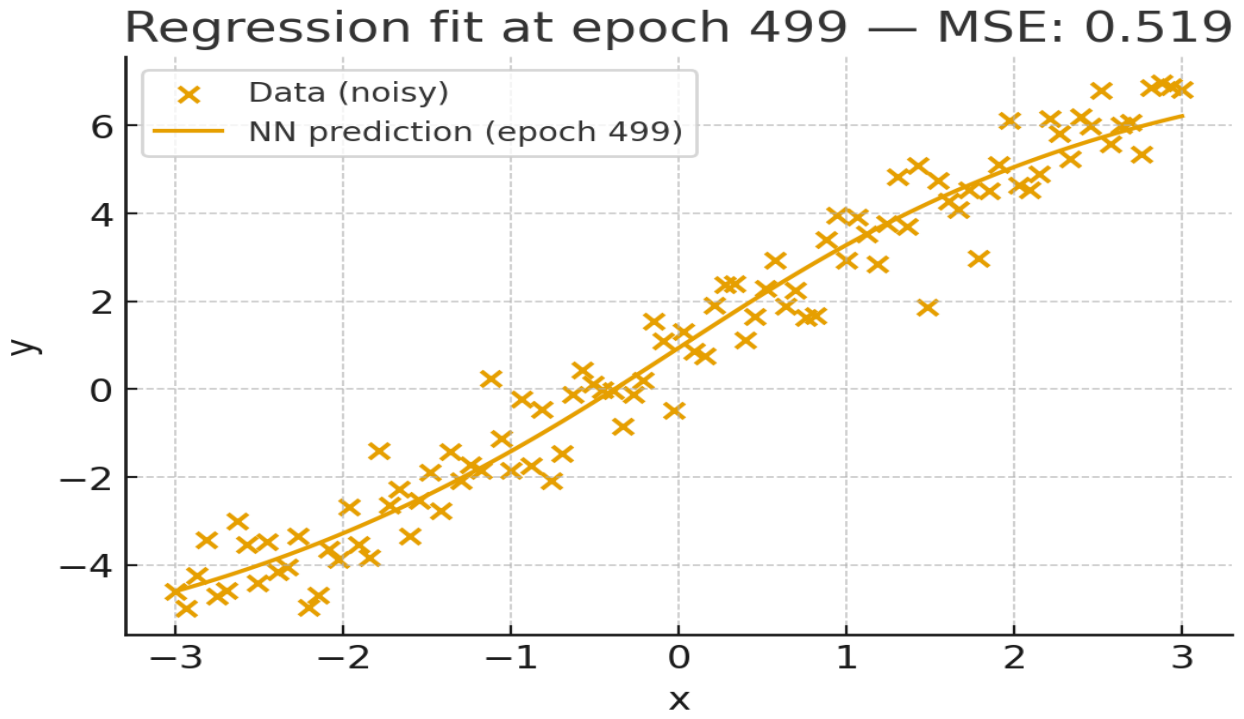

Regression fit at epoch 0 — MSE: 13.323
Data (noisy)
NN prediction (epoch 0)
x
y


Regression fit at epoch 1 — MSE: 10.757
Data (noisy)
NN prediction (epoch 1)
x
y


Regression fit at epoch 10 — MSE: 1.474
Data (noisy)
NN prediction (epoch 10)
x
y


Regression fit at epoch 50 — MSE: 0.645
Data (noisy)
NN prediction (epoch 50)
x
y


Regression fit at epoch 200 — MSE: 0.560
Data (noisy)
NN prediction (epoch 200)
y
x
6
4
2
0
−2
−4
−3
−2
−1
0
1
2
3


Regression fit at epoch 499 — MSE: 0.519
Data (noisy)
NN prediction (epoch 499)
y
x
6
4
2
0
−2
−4
−3
−2
−1
0
1
2
3

Problem 1:

A neural network has a single weight w. The loss function is given as:

$$L(w) = \left(w - 5\right)^2$$

1.Compute the gradient at $w = 2$.

2.If the learning rate $\eta = 0.1$, perform one step of gradient descent to update w.

3.What is the new value of the loss function after the update?

Problem 2:

Consider a neural network with weights w_1 and w_2 .The loss function is:

$$L(w_1, w_2) = \left(w_1 - 3\right)^2 + \left(w_2 + 1\right)^2$$

1. Find the partial derivatives $\frac{\partial L}{\partial w_1}$ and $\frac{\partial L}{\partial w_2}$ at $w_1 = 0,\ w_2 = 0.$
2. Using a learning rate $\eta = 0.2$, compute the updated weights after one gradient descent step.
3. Compute the loss at the new weights.

Problem 3:

A neural network's bias b has a loss function:

$$L(b) = b^3 - 6b^2 + 9b + 1$$

1. Determine the critical point(s) of $L(b)$.
2. Verify whether the critical point(s) is a minimum or maximum.
3. Compute the minimum value of the loss function.
4. Interpret what this result implies for tuning the bias in the context of minimizing loss in a neural network.

Problem 4: Multi-step Gradient Descent

A neuron has a weight w and the loss function is:

$$L(w) = \left(w - 4\right)^3$$

1. Starting from $w = 0$ and learning rate $\eta = 0.5$, perform 2 steps of gradient descent.
2. Compute the loss after each step.
3. Explain if the weight is moving closer to the optimal value.

Regularization Techniques

Introduction:

In machine learning and neural networks, models are trained to learn patterns from data. However, a common problem is **overfitting**, where the model performs extremely well on the training data but poorly on unseen data. Overfitting occurs when the model becomes too complex, capturing noise along with the underlying patterns.

Regularization techniques are mathematical methods used to **prevent overfitting** by adding constraints or penalties to the model during training. These techniques help the model generalize better to new, unseen data, leading to improved performance and robustness.

Regularization is a key concept in optimization and loss function design, where it modifies the original objective function to control model complexity while maintaining accuracy.

Definition :

Regularization is a process in machine learning and statistics where additional information or constraints are added to a model to **reduce overfitting** and improve generalization.

Mathematically, if a model minimizes a loss function $L(\theta)$,regularization modifies it as:

$$L_{reg}(\theta) = L(\theta) + \lambda R(\theta)$$

Where:

- $L(\theta)$ = original loss function
- $R(\theta)$ = regularization term (penalty on model complexity)
- λ = regularization parameter controlling the strength of the penalty

Problem 1:

A dataset contains 5 data points with one feature each:

x	y
1	2
2	3
3	5
4	4
5	6

Fit a liner regression model $y = wx + b$ with L_2 regularization(Ridge) using the cost function

$$J(w,b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - (wx_i + b))^2 + \lambda w^2$$

Given $\lambda = 0.5$.

1. Calculate the regularized cost $J(w,b)$ for $w = 0.9,\ b = 1.1$
2. Compute on how L_2 regularization affects the weight w compared to unregularized regression.

Problem 2:

A dataset contains 4 data points with one feature each:

x	y
1	3
2	5
3	7
4	10

We fit the model $y = wx + b$ using Ridge regression with

$$J(w,\ b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - (wx_i + b))^2 + \lambda w^2.$$

Given $\lambda = 2,\ w = 1.8,\ \ b = 1.2$:

1. Compute the regularized cost $J(w,\ b)$
2. Compute the effect of $\lambda = 2$ on the weight w with what you would expect from the unregularized (OLS) regression

Problem 3:

A small dataset is given as:

x	y
2	3
4	7
6	8
8	12

A model $y = wx + b$
using the cost function

$$J(w,b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - (wx_i + b))^2 + \lambda w^2$$

Given w $= 1.4$, $b = 0.8$, $\lambda = 1.5$.

1. Calculate the regularized cost $J(w,b)$
2. Compute the same cost for $\lambda = 0$ (no regularization)

Hyperparameters in Machine Learning

Introduction

Hyperparameters are parameters set before the training process that guide how a machine learning model learns. Unlike model parameters (weights, biases), they are not learned from the data but are chosen by the practitioner.

They control important aspects such as learning rate in Gradient Descent, number of hidden layers in Neural Networks, maximum depth in Decision Trees, and regularization strength in regression models.

The right choice of hyperparameters ensures good model performance, balanced complexity, and efficient training. Poor selection can cause underfitting (too simple) or overfitting (too complex). To find the best values, methods like grid search, random search, and Bayesian optimization are commonly used.

Definition

Hyperparameters are parameters in a machine learning model that are set before the learning process begins and govern the overall behavior of the training algorithm. They are external to the model and cannot be directly estimated from the training data. Instead, they must be specified or tuned by the practitioner.

Example:

Learning Rate in Gradient Descent → Controls the step size of weight updates.
Number of Hidden Layers in Neural Networks → Determines model depth and capacity.
Maximum Iterations → Defines how long training runs.
Regularization Strength (λ) → Balances bias and variance.

Question1.

Consider the scalar cost function

$$J(\theta) = 25\left(\theta - 2\right)^2$$

Starting from $\theta_0 = 0$, perform gradient descent updates

$$\theta_{t+1} = \theta_t - \alpha \frac{dJ}{d\theta} \mid_{\theta=\theta_t}$$

for **5 iterations** for each learning rate below:

Case A: $\alpha = 0.01$

Case B: $\alpha = 0.5$

1. For each case show the value of θ_t and $J(\theta_t)$ at iterations $t = 0, 1, \ldots, 5$,
2. State which learning rate is more effective and explain any risk for large α.

Question2.

Consider the quadratic cost function

$$J(\theta) = 10\left(\theta - 3\right)^2$$

Starting from $\theta_0 = 6$,perform **gradient descent** updates for **5 iterations** using the rule

$$\theta_{t+1} = \theta_t - \alpha \frac{dJ}{d\theta} |_{\theta=\theta_t}$$

for the following two learning rates:

- **Case A:** $\alpha = 0.1$
- **Case B:** $\alpha = 0.6$

For each case:

1.Compute $\theta_t$and $J(\theta_t)$for $t = 0, 1, 2, 3, 4, 5$.

2.State which learning rate is more effective and what risk occurs if α is too high.

Question3.

A neural network is trained on a binary classification task with 30 input features, using fully connected layers and a single sigmoid output neuron. The dataset has 2000 training examples and 500 test examples. Three architectures are tried:

Model	Hidden Layer	Neurons Per Layer	Train Accuracy	Test Accuracy
A	1	10	78%	75%
B	2	10	95%	90%
C	4	20	99%	60%

(a) For each model compute the total number of learnable parameters (weights + biases).

(b) Compute the generalization gap (train − test accuracy) for each model and comment numerically which model overfits most.

Question-4:

A neural network is trained for image classification on a dataset of 10,000 training images and 2,000 test images, each represented by 64 input features (after preprocessing). The architectures and results are summarized below:

Model	Hidden Layer	Neurons Per Layer	Train Accuracy	Test Accuracy
A	1	32	82%	78%
B	2	64	96%	91%
C	5	128	99%	65%

Each model uses fully connected layers with a single softmax output layer of 10 neurons

1. Calculate the total number of trainable parameters in each model (weights + biases).
2. Compute the generalization gap (Train − Test accuracy) for each model and identify which one overfits most.

Chain Rule and Backpropagation

Introduction

In Machine Learning, especially in Neural Networks, we often need to update model parameters (weights and biases) to minimize the loss function. This requires calculating derivatives of the loss with respect to each parameter. Since neural networks are built from many nested functions (layers), we use the Chain Rule of Calculus to compute these derivatives efficiently.

The Backpropagation algorithm applies the chain rule systematically across the network, layer by layer, to compute gradients. These gradients are then used in optimization methods (like Gradient Descent) to update weights and improve the model.

Definition

•Chain Rule (Mathematics):

If a function y depends on u, and u depends on x, then the derivative of y with respect to x is:

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

•Backpropagation (Machine Learning):

Backpropagation is an algorithm that uses the **chain rule** to compute the gradient of the loss function with respect to each weight in a neural network by propagating errors backward from the output layer to the input layer.

Example:

If $y = (3x + 2)^2,$

then using chain rule:

$$\frac{dy}{dx} = 2(3x + 2) \cdot 3 = 6(3x + 2).$$

This same principle is extended in backpropagation for multi-layer neural networks.

Question1.

A single neuron with input $x \in \mathbb{R}$ computes $\hat{y} = wx + b$ .The loss for a single training example (x, y) is the mean squared error $L = \frac{1}{2}(\hat{y} - y)^2$.

Given one training example: $x = 2.0$, $y = 5.0$. Current parameters: $w = 1.5$, $b = 0.5$. Learning rate $\eta = 0.1$.

(a) Compute the neuron output $\hat{y}$ and the loss L.

(b) Compute the gradients $\frac{\partial L}{\partial w}$ and $\frac{\partial L}{\partial b}$ using the chain rule.

(c) Perform one gradient descent update for w and b and give the updated parameter values.

Question2:

A single neuron with input $x \in \mathbb{R}$ computes

$$\hat{y} = wx + b$$

The loss for one training example $(x, y)$is given by the Mean Squared Error (MSE):

$$L = \frac{1}{2}(\hat{y} - y)^2$$

Given:

$$x = 3.0, y = 7.0, w = 2.0, b = 1.0, \eta = 0.05$$

(a) Compute the neuron output $\hat{y}$ and the loss L.

(b) Compute the gradients $\frac{\partial L}{\partial w}$ and $\frac{\partial L}{\partial b}$ using the Chain Rule.

(c) Perform one gradient descent update for w and b, and give the updated parameter values.

Question 3:

A neuron has input $x = 1$, weight $w = 2$, bias $b = 0$.

The activation (sigmoid) function is given by:

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \qquad z = w \cdot x + b$$

The output is compared with target $t = 0$, and the loss is:

$$L = \frac{1}{2}(y - t)^2$$

Compute

1.The forward pass (y) and the Loss.

2.The derivative $\frac{\partial L}{\partial w}$ using backpropagation.

3. Compute how this effects over all in the Machine learning training

Question 4:

A neuron has input $x = 2$, weight $w = 1.0$, and bias $b = -1.0$.

The activation function is the sigmoid:

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \text{where } z = w \cdot x + b$$

The target output is $t = 1$.

The loss function is the mean squared error:

$$L = \frac{1}{2}(y - t)^2$$

Tasks:

(a) Compute the forward pass z, y, and L.

(b) Using backpropagation, compute the gradient $\frac{\partial L}{\partial w}$.

(c) If the learning rate $\eta = 0.2$, perform one gradient descent update for w and b.

Assessing the Significance of the Input Data Features

Introduction

In Machine Learning, the quality and relevance of input features directly determine the accuracy and efficiency of a model. Not all input features contribute equally to prediction; some may be highly informative, while others may add noise or redundancy. Assessing the significance of input data features helps in identifying which features are most important for the learning task. This process improves model performance, reduces overfitting, decreases computational cost, and provides interpretability.

Common methods include feature importance scores (from Decision Trees or Random Forests), correlation analysis, mutual information, PCA (dimensionality reduction), and regularization techniques (like LASSO).

Definition

Assessing the significance of input data features is the process of evaluating and ranking the contribution of each feature in determining the output of a machine learning model. It helps in identifying the most relevant features, eliminating irrelevant or redundant ones, and enhancing the model's predictive power and interpretability.

Example:

In predicting house prices, features like square footage and location may be highly significant, while features like wall color may have negligible impact.

Question 1.

A dataset is used to predict student exam scores. The correlation of different features with the target (exam score) is given:

Feature	Correlation with score
Hours of study	0.85
Attendance %	0.70
Mobile usage (Hrs)	-0.60
Favorite colour	0.05

(i) Rank the features in terms of importance.

(ii) Which feature can be dropped and why?

(iii) Explain what the negative correlation of *Mobile usage (Hrs)* indicates about student performance.

(iv) If the model uses all features without dropping any, what potential problem might occur during training?

Question 2.

A Decision Tree Regressor is trained to predict car prices (in ₹ lakhs) using the following features. The model produces the following feature importance scores:

Feature	Importance Score
Engine Capacity(cc)	0.40
Mileage(km/l)	0.25
Age of care(years)	0.20
Brand Rating	0.10
Color	0.05

(i) Rank the features in terms of their importance in predicting car price.
(ii) Which feature can be dropped to simplify the model? Why?
(iii) If the model's training accuracy is 98% but test accuracy is 80%, identify and explain the problem.
(iv) Suggest one method to reduce the problem identified in (iii).
(v) If "Engine Capacity" and "Mileage" are highly correlated, what issue can arise and how to handle it?

Question 3. A Decision Tree Classifier predicts whether an employee will leave the company (Attrition = Yes/No). The model reports the following Gini-based feature importances:

Feature	Important Score
Job Satisfaction	0.38
Monthly Income	0.25
Years at Company	0.22
Distance from Home	0.10
Age	0.05

(i) Which feature most strongly affects attrition prediction?
(ii) Which feature has the least impact and can be considered for removal?
(iii) If pruning is applied and "Distance from Home" is removed, explain how this affects model bias and variance.
(iv) Suppose a new feature "Work-life Balance" with correlation 0.82 to "Job Satisfaction" is added. Explain its possible impact.
(v) The model misclassifies 10 out of 50 employees in the test set. Compute the accuracy and error rate.

Question 4. A Decision Tree model is trained to predict crop yield (in tons/hectare) using multiple environmental factors. The calculated feature importance scores are:

Feature	Importance Score
Rainfall (mm)	0.35
Soil Quality Index	0.30
Fertilizer Usage (kg/acre)	0.20
Temperature (°C)	0.10
Wind Speed (km/h)	0.05

(i) Identify the top two factors that most influence crop yield.
(ii) If the feature "Wind Speed" is removed, explain the effect on model complexity and generalization.
(iii) If after pruning, model accuracy on training data drops from 95% to 88% and test accuracy rises from 82% to 90%, explain what this indicates.
(iv) Suggest a reason why "Rainfall" might have higher importance than "Temperature."
(v) If rainfall data is missing for 5% of records, suggest one data preprocessing technique to handle it.


Thank You