Training Report – Day 12

Topic Covered Today:

- Introduction to **Artificial Neural Networks (ANN)**
- Concept of **Forward Propagation**
- Concept of **Backward Propagation (Backpropagation)**
- Activation functions and error calculation

Key Learning:

Artificial Neural Network (ANN):

Today I learned about **Neural Networks**, which are the foundation of **Deep Learning**. Neural networks are inspired by the structure of the human brain and consist of layers of **neurons** (nodes) that process input data and learn patterns.

Structure of ANN:

1. **Input Layer** – Takes input features from the dataset.
2. **Hidden Layers** – Perform computations and extract features.
3. **Output Layer** – Produces the final prediction or result.

Each connection between neurons has a **weight**, which determines the importance of the input.

Forward Propagation:

Forward propagation is the process by which input data moves through the network to generate an output.

Steps:

1. Inputs are multiplied by weights and added with bias.
2. The result is passed through an **activation function** (like ReLU, sigmoid, or tanh).
3. This process continues layer by layer until the final output is obtained.

Example Formula:
[
$y = f(Wx + b)$

]
where

- (W) = weights
- (x) = input
- (b) = bias
- (f) = activation function

Python Example (Simplified):

```
import numpy as np

# Input and weights
x = np.array([1, 2])
w = np.array([0.4, 0.6])
b = 0.5

# Forward propagation
z = np.dot(x, w) + b
output = 1 / (1 + np.exp(-z))  # Sigmoid activation
print("Output:", output)
```

This produces the neural network's output after forward propagation.

Backward Propagation:

Backward propagation (or **backpropagation**) is the learning phase of the network. It works by minimizing the **error** between predicted output and actual output through **gradient descent**.

Steps:

1. Calculate error = (predicted – actual).
2. Compute the **gradient** (partial derivative of error with respect to each weight).
3. Update weights and biases to reduce error.

Weight Update Formula:
[
W_{new} = W_{old} - \eta \times \frac{\partial E}{\partial W}
]
where

- (\eta) = learning rate
- (E) = error

This process continues until the model's output becomes accurate.

Key Terms:

- **Loss Function:** Measures how wrong the predictions are.
- **Learning Rate:** Controls how much the weights change in each iteration.
- **Epoch:** One complete pass through the entire dataset during training.

Activation Functions:

- **Sigmoid:** Converts values between 0 and 1.
- **ReLU (Rectified Linear Unit):** Speeds up learning by allowing only positive outputs.
- **Tanh:** Outputs values between -1 and 1, used for normalized data.

These functions help the model learn non-linear relationships.

Activities / Assignments:

- Studied the architecture of **neural networks** with input, hidden, and output layers.
- Performed a **forward propagation** example in Python.
- Understood **error calculation** and **weight updates** in backpropagation.
- Visualized how learning rate affects convergence.
- Prepared a flowchart showing steps of **forward and backward propagation**.

Personal Reflection for Day 12:

Today's session was one of the most important in understanding how **machine learning models actually learn**. Forward propagation showed me how data moves through layers to generate predictions, while backward propagation explained how models correct themselves by adjusting weights.

Initially, the mathematics behind derivatives and gradients seemed challenging, but the instructor's step-by-step explanation made it clear. I realized that **backpropagation** is what truly makes neural networks "learn" from data.

This topic helped me appreciate the complexity and power of deep learning and how neural networks form the base for advanced AI systems like image recognition and natural language processing.