Day 20: Understanding Weight Updation in Neural Networks

🔍 Topic Overview:

Today's session focused on the **core mechanism that drives learning in neural networks — weight updation**. We studied how neural networks adjust their internal parameters (weights and biases) to minimize the error between predicted and actual outputs using **backpropagation** and **gradient descent**.

□ What Are Weights in Neural Networks?

- **Weights** are numerical parameters associated with the input features of neurons in each layer.
- They control how much influence each input has on the output.
- During training, these weights are **updated to reduce prediction error**.

⚙□ The Learning Process:

1. **Forward Propagation**: The input data is passed through the network to produce a prediction.
2. **Loss Calculation**: The difference between predicted and actual values is calculated using a **loss function** (e.g., MSE, Cross-Entropy).
3. **Backpropagation**:
    - Derivatives of the loss are computed w.r.t each weight (i.e., how sensitive the loss is to each weight).
4. **Gradient Descent**:
    - Weights are updated using the gradient and a learning rate.
    - Formula:

      w=w−η·∂L∂ww = w - \eta \cdot \frac{\partial L}{\partial w}w=w−η·∂w∂L

      where:

        - www = weight
        - η\etaη = learning rate
        - ∂L∂w\frac{\partial L}{\partial w}∂w∂L = gradient of loss w.r.t. weight

---

📌 Example Code Snippet: Weight Update Using Gradient Descent

```
# Simple weight update demo using gradient descent
import numpy as np

# Inputs (x) and actual output (y)
x = np.array([1, 2, 3])
y = np.array([2, 4, 6])
```

```
# Initialize weight
w = 0.0
lr = 0.01  # learning rate

# Mean Squared Error Loss
def loss_fn(y, y_pred):
    return ((y - y_pred) ** 2).mean()

# Training loop
for epoch in range(50):
    y_pred = w * x
    loss = loss_fn(y, y_pred)
    grad = -2 * (y - y_pred).dot(x) / len(x)  # Derivative of MSE w.r.t
w
    w -= lr * grad

    if epoch % 10 == 0:
        print(f"Epoch {epoch} | Loss: {loss:.4f} | Weight: {w:.4f}")
```

```
Epoch 0 | Loss: 18.6667 | Weight: 0.1867
Epoch 10 | Loss: 2.6304 | Weight: 1.3193
Epoch 20 | Loss: 0.3707 | Weight: 1.7445
Epoch 30 | Loss: 0.0522 | Weight: 1.9041
Epoch 40 | Loss: 0.0074 | Weight: 1.9640
```

✅ Key Takeaways:

- **Weights are crucial parameters** that are adjusted during training to improve model performance.
- **Gradient Descent** is the most common method for weight updates.
- Smaller learning rates lead to **slower but stable** convergence, while larger rates may cause **oscillations** or **divergence**.
- Modern optimizers (like Adam, RMSProp) enhance basic gradient descent with **adaptive learning rates** and **momentum**.