Lab - 6

22.08.25 — IMPLEMENT GRADIENT DESCENT & BACKPROPAGATION IN DEEP NN

AIM: Implementing gradient descent & Backpropagation in deep neural networks.

PSEUDO CODE:-

1. Initialize network parameters (weights & biases) randomly or using heuristics.
2. For each epoch:
   a. For each training sample (x, y):
      (i) Forward Pass:
         - Compute activation layer by layer until output is obtained.

      (ii) Compute loss:
         - Calc. error b/w predicted output & true label (e.g., MSE or cross-entropy).

      (iii) Backward Pass:
         - Compute gradient of loss w.r.t output layer parameters.
         - Propagate errors backward through hidden layer using chain rule.
         - Compute gradients of loss w.r.t each weight & bias.

      (iv) Update Parameters:
         - For each parameters θ:

           $\theta = \theta - \eta * (\partial \text{loss} / \partial \theta)$

           where η = learning rate.
3. Repeat until convergence or stopping condition is met (max epochs or minimal loss).

Justification :-

* Gradient descent : It provides an efficient optimization in high-dimensional neural network
- Ensures iterative improvement of model parameters in the direction of steepest descent.

* Back Progation : Uses chain rule of calculus to compute partial derivatives of the loss function w.r.t all network parameters.
- Allows efficient computation of gradients across multiple layers instead of computing derivatives independently

SMALL NOTE : (Reference purpose)

Epoch = One complete pass for an iteration through the neural network.

loss = Shows how wrong the model prediction is.

16/9/25

Observation:

Epoch 0, Loss : 1.0763

Epoch 1000, Loss : 0.6932

Epoch 2000, Loss : 0.6931

Epoch 3000, Loss : 0.6931

Epoch 4000, Loss : 0.6931

Epoch 5000, Loss : 0.6931

Epoch 6000, Loss : 0.6931

Epoch 7000, Loss : 0.6931

Epoch 8000, Loss : 0.6931

Epoch 9000, Loss : 0.6931

Final Predictions :

[[0]
 [1]
 [0]
 [1]]

Loss Curve

```python
import numpy as np
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
def sigmoid_derivative(x):
    s = sigmoid(x)
    return s * (1 - s)
def relu(x):
    return np.maximum(0, x)
def relu_derivative(x):
    return (x > 0).astype(float)
np.random.seed(42)
W1 = np.random.randn(2, 3)
b1 = np.zeros((1, 3))
W2 = np.random.randn(3, 1)
b2 = np.zeros((1, 1))
def forward(X):
    Z1 = X.dot(W1) + b1
    A1 = relu(Z1)
    Z2 = A1.dot(W2) + b2
    A2 = sigmoid(Z2)
    cache = (X, Z1, A1, Z2, A2)
    return A2, cache
def compute_loss(Y, A2):
    m = Y.shape[0]
    return -np.sum(Y*np.log(A2) + (1-Y)*np.log(1-A2)) / m
def backward(Y, cache):
    X, Z1, A1, Z2, A2 = cache
    m = Y.shape[0]
    dZ2 = A2 - Y
    dW2 = (A1.T).dot(dZ2) / m
```

```
    X, Z1, A1, Z2, A2 = cache
    m = Y.shape[0]
    dZ2 = A2 - Y
    dW2 = (A1.T).dot(dZ2) / m
    db2 = np.sum(dZ2, axis=0, keepdims=True) / m
    dA1 = dZ2.dot(W2.T)
    dZ1 = dA1 * relu_derivative(Z1)
    dW1 = (X.T).dot(dZ1) / m
    db1 = np.sum(dZ1, axis=0, keepdims=True) / m
    return dW1, db1, dW2, db2
def update_params(W1, b1, W2, b2, dW1, db1, dW2, db2, lr=0.1):
    W1 -= lr * dW1
    b1 -= lr * db1
    W2 -= lr * dW2
    b2 -= lr * db2
    return W1, b1, W2, b2
X = np.array([[0,0],[0,1],[1,0],[1,1]])
Y = np.array([[0],[1],[1],[0]])
epochs = 10000
lr = 0.1
for i in range(epochs):
    A2, cache = forward(X)
    loss = compute_loss(Y, A2)
    dW1, db1, dW2, db2 = backward(Y, cache)
    W1, b1, W2, b2 = update_params(W1, b1, W2, b2, dW1, db1, dW2, db2, lr)
    if i % 1000 == 0:
        print(f"Epoch {i}, Loss: {loss:.4f}")
preds = (forward(X)[0] > 0.5).astype(int)
print("Final Predictions:")
print(preds)
```

Week2.py × Week3.py × Untitled.ip × week5.py × jupyter-ra × we

```
jupyter-ra2311047010012@cintel:~/Foundation of AI/SEM 5 DLT LAB$ python week6.py
jupyter-ra2311047010012@cintel:~/Foundation of AI/SEM 5 DLT LAB$ python week6.py
Epoch 0, Loss: 1.0763
Epoch 1000, Loss: 0.6932
Epoch 2000, Loss: 0.6931
Epoch 3000, Loss: 0.6931
Epoch 4000, Loss: 0.6931
Epoch 5000, Loss: 0.6931
Epoch 6000, Loss: 0.6931
Epoch 7000, Loss: 0.6931
Epoch 8000, Loss: 0.6931
Epoch 9000, Loss: 0.6931
Final Predictions:
[[0]
 [1]
 [0]
 [1]]
jupyter-ra2311047010012@cintel:~/Foundation of AI/SEM 5 DLT LAB$
```