collision of folg is Output :-Epoch 2000, Loss: 0.2454 1900 (cueix of state gains) Epoch 0, Loss: 0.2558 Epoch 4000, Loss: 0.1532 apach \$000 , 2013: 0.1336 Epoch 8000, Loss: 0.1267 -: Dues Final poudiations: 2 milestina between in alle passance. [[0.05300868] and hadred by how down of the [0.49554213] [0.95091319] [050019888]] Loss Curve - Gradient Degrant and Back propagation - Training Loss. 0.24+ 0.22+ 0.20+ 0.18 + 0.15+ 0.14 0 8000 2000 6000 4000

Ex: 6 Implement gradient decent and backpropagation in

depp neural network.

Aim: To implement arradient descent and Backpropogation in depp tournal motoorie.

Objectives:

\* To understand gradient descent in an optimized method

\* To implement Backproppogation in deep neural network to

update weights. \* To implement a simple neural network for classification

+ To Obsone how loss decrease with iteration.

## Observation:

\* Loss docreases as needear of itomations increase

+ weights and bias object to minimize over.

\* Back propagation encures corers are afficiently

distributed layer of Layer. \* Leavining reate (n) greatly influences con Speed

## P. seudocode:

Begin

Initialize weight and bies Standomly

For epoch in stange (max-epochs)

for each input Sample

\* Forward press

Compute Z= Wx x+6

apply activation to get A

Compute output posediction

of Compute loss

loss = costly (y.tree, y. poud)

of Backward pase

Compute gradient du , db using Chain

embaparanthed has track trackens trackens to State update parameters: w=w-a+dw b=b-a+dbEND for Porint after epoch
ENDFOR to complement Each proprogram in dup nevolation of 4 Results:-Therefore implemention of gradient descent and backpropagation in neural metions. to loss decreases as neuclear of stendions increase of weights and bies object to minimize occur the Earlies encourse Covers and afficiently distributed layer of layer.

\* Learning outs (1) greatly influences Con Specification and the coming outs (1) greatly influences Con Specification and con Specification and con Specification and con Specification and control of the ter epoch in sange (max epochs) seed facoust to Compute to with 6 apply activation to get A compute suspect psecriction as described has of publicants loss due, the wing the

```
import numpy as np
import matplotlib.pyplot as plt
              # Sigmoid activation and derivative
              def sigmoid(x):
    return 1 / (1 + np.exp(-x))
              def sigmoid_derivative(x):
    return x * (1 - x) # derivative assumes input = sigmoid(x)
              # Training data (XOR problem)
X = np.array([[0,0],[0,1],[1,0],[1,1]])
y = np.array([[0],[1],[1],[0]])
              # Set random seed
np.random.seed(42)
               # Network architecture
              input_size = 2
              hidden_size = 2
output_size = 1
              # Initialize weights and biases
              W1 = np.random.randn(input_size, hidden_size)
b1 = np.zeros((1, hidden_size))
              W2 = np.random.randn(hidden_size, output_size)
              b2 = np.zeros((1, output_size))
              # Hyperparameters
lr = 0.1
epochs = 10000
losses = []
                  Training loop
              for epoch in range(epochs):
                    # ---- Forward pass ----
z1 = np.dot(X, W1) + b1
a1 = sigmoid(z1)
                     z^2 = no.dot(a1, W2) + b2
                     a2 = sigmoid(z2)
                     # ---- Loss (Mean Squared Error) ----
                    loss = np.mean((y - a2) ** 2)
losses.append(loss)
                    # ---- Backpropagation ----
d_a2 = (a2 - y)
d_z2 = d_a2 * sigmoid_derivative(a2)
dW2 = np.dot(a1.T, d_z2)
db2 = np.sum(d_z2, axis=0, keepdims=True)
                    d_a1 = np.dot(d_z2, W2.T)
d_z1 = d_a1 * sigmoid_derivative(a1)
dW1 = np.dot(X.T, d_z1)
db1 = np.sum(d_z1, axis=0, keepdims=True)
                     # ---- Update weights --
                    W2 -= lr * dW2
b2 -= lr * db2
W1 -= lr * dW1
b1 -= lr * db1
                    # Print loss occasionally
                    if epoch % 2000 == 0:
    print(f"Epoch {epoch}, Loss: {loss:.4f}")
                       - Final predictions
              print("\nFinal Predictions:")
             print(a2)
             # ---- Plot the loss curve ----
plt.figure(figsize=(8,5))
plt.plot(losses, label="Training Loss", color="blue")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.title("Loss Curve - Gradient Descent & Backpropagation")
                                                                                                                                                                                                                              6
             plt.legend()
plt.grid(True)
plt.show()
                                                                                                                                                                                                                                  1/2
colab.research.google.com/drive/1rUsDZe-II4OeDqlfkkRaTsqlQJ-Vo62v#printMode=true
```

## 5, 7:29 PM

```
DLT_Lab_6.ipynb - Colab
```

