Exp 6 - (Implement Gradient Descent and Back propagation in DNN)

AIM :- Implementing gradient descent & Back propagation in deep NN.

Pseudocode :- 1) Initialize network parameters (weight & bias) randomly.

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

→ Calculate error b/w predicted output & true label.

iii) Backward pass

→ Compute gradient of loss w.r.t output layer parameters.

→ propagate error backward through hidden layer using chain rule.

→ compute gradient of loss w.r.t each weight & bias

iv) Update parameters.

→ $\theta = \theta - \eta \cdot (\partial loss / \partial \theta)$

η value of η - learning rate.

v) Repeat until convergence or stopping condition is that (max epochs or minimal loss).

Justification:

→ Gradient Descent :- It provides an efficient optimization in high dimensional NN.

→ Back propagation :- Uses chain rule of calculus to compute partial derivative of the loss function w.r.t all network parameters.

→ Allows efficient computation of gradients across multiple layers instead of computing derivatives independently.

Result: By succeeding epoch, loss is being reduced

Observations

Epoch 0 , Loss = 1.076

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

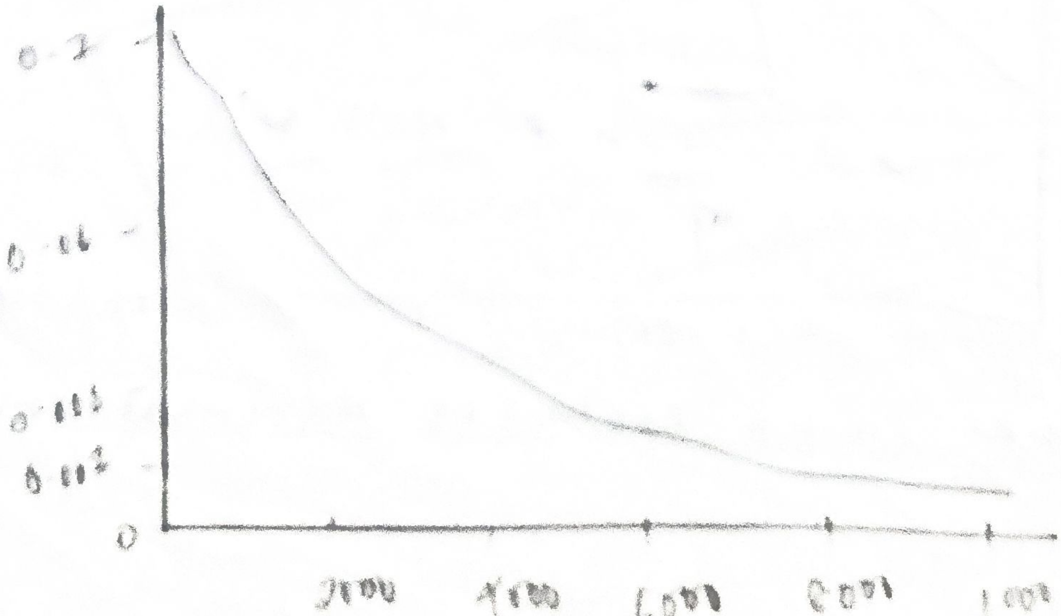
Epoch 8000, loss = 0.6931

Epoch 9000, loss = 0.693

Final Prediction

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

Final prediction



epoch vs loss