Ayush Chanchal

Sap:500097569

Batch: B6

EXPERIMENT:06

**Backpropagation:**

Backpropagation is a method used to train neural networks. It's like a feedback mechanism that helps the network learn from its mistakes. Here's a simple explanation:

* Forward Pass: During the training process, the neural network makes predictions on the input data. These predictions might not be very accurate at first.
* Calculate Error: We compare these predictions with the actual desired outputs to see how wrong or right the network's predictions are.
* Backward Pass (Backpropagation): This is where the magic happens. We take the error and move it backward through the network, kind of like telling each neuron how much it contributed to the error.
* Adjust Weights: Neurons have weights that determine how much they influence the output. During backpropagation, these weights get adjusted based on how responsible they were for the error. Neurons that contributed more to the error get bigger adjustments.
* Repeat: Steps 1-4 are repeated for many iterations (epochs) until the network's predictions become accurate.

**Implement Backpropagation with Error Correction learning in Python.**

**Source Code:**

import numpy as np

def initialize\_parameter(layers):

parameters={}

for i in range(1,len(layers)):

parameters["W"+str(i)]=np.ones((layers[i-1],layers[i]))\*0.1

return parameters

def forward\_prop(X,parameters):

A=X

for i in range(1,len(parameters)+1):

A\_prev=A

W1=parameters["W"+str(i)]

# print(f"A{i}={A\_prev}")

# print(f"W{i}={W1}")

A=np.dot(A,W1)

# print(f"output={A}")

return A,A\_prev

def update\_parameter(X,layers,learning\_rate,Y):

parameters=initialize\_parameter(layers)

for epochs in range(20):

for i in range(X.shape[0]):

Y\_hat,A=forward\_prop(X[i],parameters)

for row1 in range(parameters["W2"].shape[0]):

for col1 in range(parameters["W2"].shape[1]):

parameters["W2"][row1][col1]=parameters["W2"][row1][col1]+(learning\_rate\*(Y[i]-Y\_hat)\*2\*[row1])

for row in range(parameters["W1"].shape[0]):

for col in range(parameters["W1"].shape[1]):

parameters["W1"][row][col]=parameters["W1"][row][col]+learning\_rate\*(Y[i]-Y\_hat)\*2\*parameters["W2"][col][0]\*X[i][row]

return(parameters)

X\_train=np.array([[1,2,3],[4,5,6]])

Y\_train=np.array([[1],[4]])

layers=[3,2,1]

weight\_best=update\_parameter(X\_train,layers,0.01,Y\_train)

x\_test=np.array([[7,8,9],[10,11,12]])

y\_test=np.array([[7],[10]])

for e in x\_test:

output,output\_prev\_layer=forward\_prop(e,weight\_best)

print("prdicted output is",output)

Screenshot:

