**Name: Abdul Ahad**

**Roll no: CS-22139**

**AI – Lab 05**

Use the above lab example for XNOR gate implementation and retrain the network. Write down the modifications (only) to the above mentioned Python code for XNOR implementation.

ANSWER:

import numpy as np

from matplotlib import pyplot as plt

def forwardPropagation(X, Y, parameters):

m = X.shape[1]

W1 = parameters["W1"]

W2 = parameters["W2"]

b1 = parameters["b1"]

b2 = parameters["b2"]

Z1 = np.dot(W1, X) + b1

A1 = sigmoid(Z1)

Z2 = np.dot(W2, A1) + b2

A2 = sigmoid(Z2)

cache = (Z1, A1, W1, b1, Z2, A2, W2, b2)

logprobs = np.multiply(np.log(A2), Y) + np.multiply(np.log(1 - A2), (1 - Y))

cost = -np.sum(logprobs) / m

return cost, cache, A2

def backwardPropagation(X, Y, cache):

m = X.shape[1]

(Z1, A1, W1, b1, Z2, A2, W2, b2) = cache

dZ2 = A2 - Y

dW2 = np.dot(dZ2, A1.T) / m

db2 = np.sum(dZ2, axis = 1, keepdims = True)

dA1 = np.dot(W2.T, dZ2)

dZ1 = np.multiply(dA1, A1 \* (1- A1))

dW1 = np.dot(dZ1, X.T) / m

db1 = np.sum(dZ1, axis = 1, keepdims = True) / m

gradients = {"dZ2": dZ2, "dW2": dW2, "db2": db2,

"dZ1": dZ1, "dW1": dW1, "db1": db1}

return gradients

# Model to learn the XNOR truth table

X = np.array([[0, 0, 1, 1], [0, 1, 0, 1]]) # XNOR input

Y = np.array([[1, 0, 0, 1]]) # XNOR output

neuronsInHiddenLayers = 2 # number of hidden layer neurons (2)

inputFeatures = X.shape[0] # number of input features (2)

outputFeatures = Y.shape[0] # number of output features (1)

parameters = initializeParameters(inputFeatures, neuronsInHiddenLayers, outputFeatures)

epoch = 100000

learningRate = 0.01

losses = np.zeros((epoch, 1))

for i in range(epoch):

losses[i, 0], cache, A2 = forwardPropagation(X, Y, parameters)

gradients = backwardPropagation(X, Y, cache)

parameters = updateParameters(parameters, gradients, learningRate)

plt.figure()

plt.plot(losses)

plt.xlabel("EPOCHS")

plt.ylabel("Loss value")

plt.show()

X = np.array([[1, 1, 0, 0], [0, 1, 0, 1]]) # XNOR input

cost, \_, A2 = forwardPropagation(X, Y, parameters)

prediction = (A2 > 0.5) \* 1.0

# print(A2)

print(prediction)