Prac6

import numpy as np

from sklearn.datasets import make\_classification

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import OneHotEncoder

def relu(x):

return np.maximum(0,x)

def softmax(x):

exp\_x=np.exp(x-np.max(x,axis=1,keepdims=True))

return exp\_x/np.sum(exp\_x,axis=1,keepdims=True)

def initialize\_parameters(input\_size,hidden\_size,output\_size):

return{

"W1":np.random.randn(input\_size,hidden\_size)\*0.01,

"b1":np.zeros((1,hidden\_size)),

"W2":np.random.randn(hidden\_size,output\_size)\*0.01,

"b2":np.zeros((1,output\_size))

}

def forward(X,params):

Z1= X @ params["W1"] + params["b1"]

A1=relu(Z1)

Z2= A1 @ params["W2"] + params["b2"]

A2=softmax(Z2)

return A1,A2

def backward(X,Y,A1,A2,params,lr):

m=X.shape[0]

dZ2=A2-Y

params["W2"]-=lr \* (A1.T @dZ2) / m

params["b2"]-=lr\*np.mean(dZ2,axis=0,keepdims=True)

dZ1= (dZ2 @ params["W2"].T)\*(A1>0)

params["W1"]-=lr\*(X.T @dZ1)/m

params["b1"]-=lr\*np.mean(dZ1,axis=0,keepdims=True)

def train(X,Y,hidden\_size=100,epochs=1000,lr=0.01):

params=initialize\_parameters(X.shape[1],hidden\_size,Y.shape[1])

for \_ in range(epochs):

A1,A2=forward(X,params)

backward(X,Y,A1,A2,params,lr)

return params

def predict(X,params):

\_,A2=forward(X,params)

return np.argmax(A2,axis=1)

X,y=make\_classification(n\_samples=1000,n\_features=20,n\_classes=3,n\_informative=15)

y=OneHotEncoder(sparse\_output=False).fit\_transform(y.reshape(-1,1))

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, y, test\_size=0.2)

params = train(X\_train, Y\_train)

y\_pred = predict(X\_test, params)

accuracy = np.mean(y\_pred == np.argmax(Y\_test, axis=1))

print(f"Test Accuracy: {accuracy \* 100:.2f}%")