Experiment No. 4
Implement a backpropagation algorithm to train a DNN with at least 2 hidden layers.
Date of Performance: 15/2/24
Date of Submission: 19/2/24

Title: Implement a backpropagation algorithm to train a DNN with hidden layers.

Aim: To implement a backpropagation algorithm to train a DNN with hidden layers.

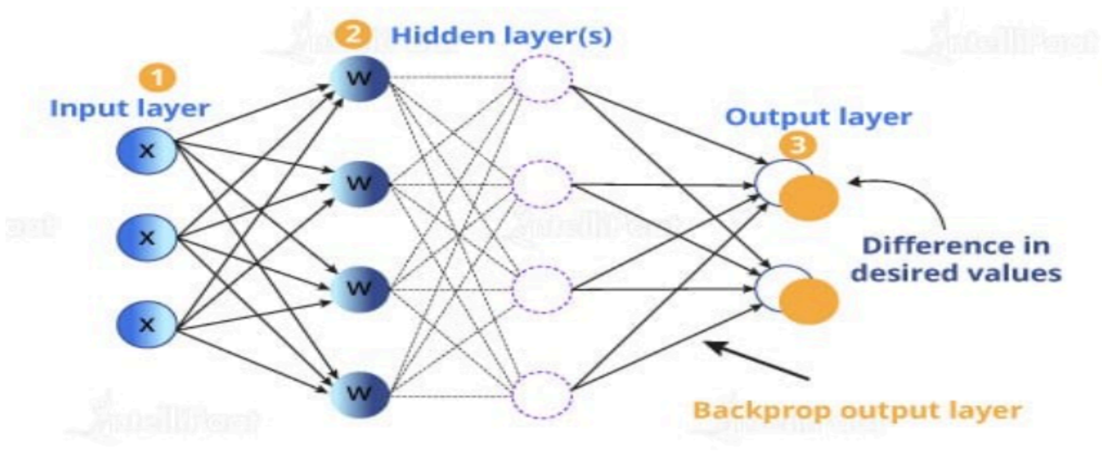
Objective: To implement various training algorithms for feedforward neural networks.

Theory:

Backpropagation is one of the important concepts of a neural network. Our task is to classify our data best. For this, we have to update the weights of parameter and bias, but how can we do that in a deep neural network? In the linear regression model, we use gradient descent to optimize the parameter. Similarly here we also use gradient descent algorithm using Backpropagation.

For a single training example, Backpropagation algorithm calculates the gradient of the error function. Backpropagation can be written as a function of the neural network. Backpropagation algorithms are a set of methods used to efficiently train artificial neural networks following a gradient descent approach which exploits the chain rule.

The main features of Backpropagation are the iterative, recursive and efficient method through which it calculates the updated weight to improve the network until it is not able to perform the task for which it is being trained. Derivatives of the activation function to be known at network design time is required for Backpropagation.



Backpropagation Algorithm:

Step 1: Inputs X, arrive through the preconnected path.

Step 2: The input is modeled using true weights W. Weights are usually chosen randomly.

Step 3: Calculate the output of each neuron from the input layer to the hidden layer to the output layer.

Step 4: Calculate the error in the outputs

Backpropagation Error= Actual Output – Desired Output

Step 5: From the output layer, go back to the hidden layer to adjust the weights to reduce the error.

Step 6: Repeat the process until the desired output is achieved.

Code & Output :

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
#sigmoid function
def nlinear(x,deriv=False):
  if(deriv==True):
    return x*(1-x)
  return 1/(1+np.exp(-x))
```

```
#input
X=np.array([[1,1,0],[0,1,1],[0,0,1],[1,1,0]])
#output
y=np.array([[1,0,0,1]]).T
#seed for random number distribution
#np.random.seed(1)
#wts initialization
synapse0=2*np.random.random((3,1))-1
```

```
for i in range(1000):
  #forward propagation
  layer0=X
  layer1=nlinear(np.dot(layer0,synapse0))
  #error
  layer1_error=y-layer1
  #multiply with error backpropagated
  layer1_delta=layer1_error*nlinear(layer1,True)
  #update wts
  synapse0+=np.dot(layer0.T,layer1_delta)
```

```
print("op:")
print(layer1)
print("ao:")
print(y)
```

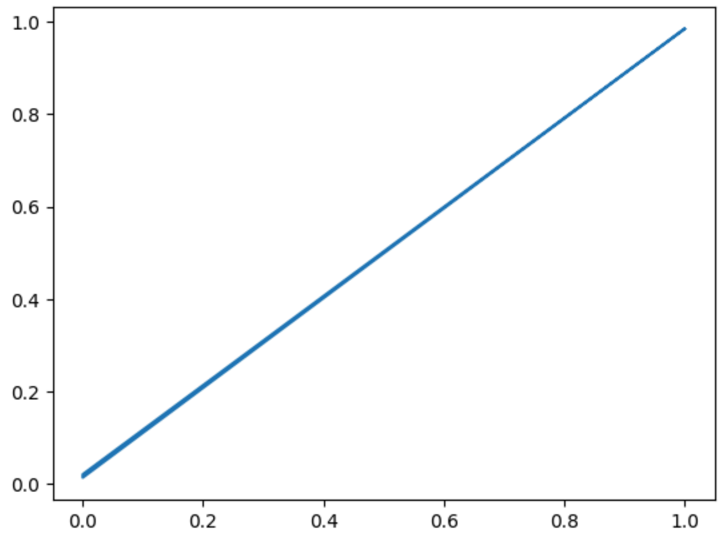
```
op:
[[0.98482533]
 [0.02005896]
 [0.01325443]
 [0.98482533]]
ao:
[[1]
 [0]
 [0]
 [1]]
```

```
df=[y,layer1]
df
```

```
[array([[1],
        [0],
        [0],
        [1]]),
 array([[0.98482533],
        [0.02005896],
        [0.01325443],
        [0.98482533]])]
```

```
plt.plot(y,layer1)
```

```
[<matplotlib.lines.Line2D at 0x7d5fc2169690>]
```



1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

Conclusion: Implementing the backpropagation algorithm to train a Deep Neural Network (DNN) with at least two hidden layers involves a sequential process aimed at optimizing the network's parameters. Beginning with the initialization of weights and biases, the algorithm proceeds through forward passes to compute predicted outputs, followed by the calculation of loss between predictions and actual targets. The crucial step of the backward pass then facilitates the computation of gradients, providing insights into how each parameter affects the overall loss. By iteratively adjusting weights and biases using optimization techniques like gradient descent, the network gradually learns to minimize the loss function, refining its predictive capabilities. This iterative process allows the DNN to capture intricate patterns in the data, enabling it to make accurate predictions on unseen samples.