Vivekanand Education Society's
Institute of Technology
Approved by AICTE & Affiliated to University of Mumbai

Artificial Intelligence and Data Science Department

Subject/Odd Sem 2023-23/Experiment 2

Name : Prasad Jawale	**Class/Roll No. :** D16AD 20	**Grade :**

Title of Experiment : Implement a backpropagation algorithm to train a DNN with at least 2 hidden layers.

Objective of Experiment : The objective is to implement a backpropagation algorithm to train a Deep Neural Network (DNN) with at least 2 hidden layers. Through this implementation, we aim to demonstrate how backpropagation computes gradients and updates weights and biases to minimize the network's error. The goal is to show the fundamental process of training a DNN using the backpropagation algorithm.

Outcome of Experiment : The outcome of this implementation will be a trained DNN with at least 2 hidden layers. We will observe how the network's performance improves over epochs as the backpropagation algorithm adjusts the model's parameters to minimize the error between predicted and actual outputs.

Problem Statement : To implement a backpropagation algorithm to train a DNN with at least 2 hidden layers.

Subject/Odd Sem 2023-23/Experiment 2

Description / Theory :

Backpropagation:

Backpropagation is the heart of DNN training. It's a process where errors are propagated backward through the network, allowing the model to learn and adjust its parameters for improved performance. As data flows forward through the network, predictions are compared to actual outputs using a loss function. Backpropagation then computes the gradients of the loss with respect to each parameter, indicating how much each parameter contributes to the error. These gradients guide parameter updates to minimize the loss over time.

Gradient Descent:

Gradient Descent is the optimization technique used alongside backpropagation to iteratively update model parameters. The goal is to find the optimal parameter values that minimize the loss function. During each iteration, the gradients computed through backpropagation indicate the direction of steepest ascent in the loss landscape. Gradient Descent reverses this direction to find the path of steepest descent, effectively updating parameters to reduce the loss.

Vivekanand Education Society's
Institute of Technology
Approved by AICTE & Affiliated to University of Mumbai

Artificial Intelligence and Data Science Department

Subject/Odd Sem 2023-23/Experiment 2

Program :

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
df = pd.DataFrame([[8,8,4],[7,9,5],[6,10,6],[5,12,7]],columns=['cgpa','profile_score','lpa'])
```

In [3]:
```
df
```

Out[3]:

	cgpa	profile_score	lpa
0	8	8	4
1	7	9	5
2	6	10	6
3	5	12	7

In [4]:
```
import tensorflow
from tensorflow import keras
from keras import Sequential
from keras.layers import Dense
```

In [5]:
```
model = Sequential()

model.add(Dense(2,activation='linear',input_dim=2))
model.add(Dense(1,activation='linear'))
```

In [6]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 2)                 6

 dense_1 (Dense)             (None, 1)                 3

=================================================================
Total params: 9
Trainable params: 9
Non-trainable params: 0
_________________________________________________________________
```

In [7]:
```
model.get_weights()
```

Out[7]:
```
[array([[-0.58026016,  0.95435727],
        [-0.29844964,  0.09312904]], dtype=float32),
 array([0., 0.], dtype=float32),
 array([[ 0.49506593],
        [-0.578511  ]], dtype=float32),
 array([0.], dtype=float32)]
```

```
In [8]: new_weights = [np.array([[0.1,0.1],[0.1,0.1]],dtype=np.float32),
                      np.array([0.,0.],dtype=np.float32),
                      np.array([[0.1],[0.1]],dtype=np.float32),
                      np.array([0.],dtype=np.float32)]

In [9]: model.set_weights(new_weights)

In [10]: model.get_weights()

Out[10]: [array([[0.1, 0.1],
                 [0.1, 0.1]], dtype=float32),
          array([0., 0.], dtype=float32),
          array([[0.1],
                 [0.1]], dtype=float32),
          array([0.], dtype=float32)]

In [11]: optimizer =  keras.optimizers.Adam(learning_rate=0.001)
         model.compile(loss='mean_squared_error',optimizer=optimizer)

In [11]: optimizer =  keras.optimizers.Adam(learning_rate=0.001)
         model.compile(loss='mean_squared_error',optimizer=optimizer)

In [12]: model.fit(df.iloc[:,0:-1].values,df['lpa'].values,epochs=75,verbose=1,batch_size=1)
         Epoch 67/75
         4/4 [==============================] - 0s 0s/step - loss: 2.1196
         Epoch 68/75
         4/4 [==============================] - 0s 876us/step - loss: 2.0313
         Epoch 69/75
         4/4 [==============================] - 0s 2ms/step - loss: 1.9359
         Epoch 70/75
         4/4 [==============================] - 0s 5ms/step - loss: 1.8588
         Epoch 71/75
         4/4 [==============================] - 0s 0s/step - loss: 1.7441
         Epoch 72/75
         4/4 [==============================] - 0s 1ms/step - loss: 1.6645
         Epoch 73/75
         4/4 [==============================] - 0s 5ms/step - loss: 1.6072
         Epoch 74/75
         4/4 [==============================] - 0s 0s/step - loss: 1.5729
         Epoch 75/75
         4/4 [==============================] - 0s 662us/step - loss: 1.5081

Out[12]: <keras.callbacks.History at 0x11b0a8f7b20>

In [18]: model.get_weights()

Out[18]: [array([[0.37387073, 0.37387073],
                 [0.36561698, 0.36561698]], dtype=float32),
          array([0.2724311, 0.2724311], dtype=float32),
          array([[0.37302735],
                 [0.37302735]], dtype=float32),
          array([0.20472449], dtype=float32)]
```

Results and Discussions : Implementing the backpropagation algorithm for training a DNN with multiple hidden layers demonstrates the power of this technique in optimizing neural network parameters. Through this implementation, we gain insights into how gradients are calculated and used to update weights and biases, leading to improved model performance over time. Backpropagation is a foundational concept in deep learning that enables networks to learn complex patterns and relationships from data.