Shri Vile Parle Kelavani Mandal's

DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING

(Autonomous College Affiliated to the University of Mumbai)

NAAC Accredited with "A" Grade (CGPA : 3.18)

Continuous Assessment for Laboratory / Assignment sessions

Academic Year 2022-23

Name: Ayush Jain SAP ID: 60004200132

Course: Machine Learning Laboratory Course Code: **DJ19CEEL6021**

Year: **T.Y. B.Tech.** Sem: **VII** Batch: B 3

Department: Computer Engineering

Performance Indicators (Any no. of Indicators) (Maximum 5 marks per indicator)	1	2	3	4	5	6	7	8	9	10	11	Σ	Avg	A1	A2	Σ	Avg
Course Outcome	2, 4	2, 4	2, 4	2, 4	2, 4	3	2, 4	2, 4	5								
1. Knowledge (Factual/Conceptual/Procedural/ Metacognitive)	4	5	5	5	5	5	5	5									
2. Describe (Factual/Conceptual/Procedural/ Metacognitive)	4	4	4	5	5	4	4	4									
3. Demonstration (Factual/Conceptual/Procedural/ Metacognitive)	5	5	5	5	5	5	5	4									
4. Strategy (Analyse & / or Evaluate) (Factual/Conceptual/ Procedural/Metacognitive)	4	4	4	4	4	5	5	5									
5. Interpret/ Develop (Factual/Conceptual/ Procedural/Metacognitive)	–	–	–	–	–	–	–	–									
6. Attitude towards learning (receiving, attending, responding, valuing, organizing, characterization by value)	4	4	4	4	5	4	4	5									
7. Non-verbal communication skills/ Behaviour or Behavioural skills (motor skills, hand-eye coordination, gross body movements, finely coordinated body movements speech behaviours)	–	–	–	–	–	–	–	–									
Total	21	22	22	23	24	23	23	23									
Signature of the faculty member																	

Outstanding (5), Excellent (4), Good (3), Fair (2), Needs Improvement (1)

Laboratory marks Σ Avg. =	Assignment marks Σ Avg. =	Total Term-work (25) =
Laboratory Scaled to (15) =	Assignment Scaled to (10) =	Sign of the Student:

Signature of the Faculty member:
Name of the Faculty member:

Signature of Head of the Department
Date:

Machine Learning

Experiment 6

Ayush Jain 60004200132 B3

Aim : To implement Backpropagation in python without using any inbuilt function.

Theory:

Backpropagation is a widely used algorithm for training artificial neural networks (ANNs). It is a supervised learning algorithm that involves propagating errors backwards through the layers of the network, in order to adjust the weights and biases of the neurons. The goal of backpropagation is to minimize the difference between the predicted output of the network and the actual output, which is known as the error or loss.

The backpropagation algorithm works by first making a forward pass through the network, in which the inputs are propagated through the layers of neurons until they reach the output layer. The output of the network is then compared to the desired output, and the error or loss is calculated. This error is then propagated backwards through the layers of neurons, starting at the output layer and working backwards towards the input layer.

During the backward pass, the algorithm calculates the gradient of the error with respect to each weight and bias in the network. The gradient is a measure of how much the error changes when the weight or bias is changed, and it is used to update the weights and biases in a way that reduces the error.

The backpropagation algorithm can be broken down into the following steps:

1. Initialize the weights and biases: The weights and biases of the neurons in the network are randomly initialized before training begins.

2. Make a forward pass: The inputs are propagated through the layers of neurons until they reach the output layer, and the output of the network is calculated.

3. Calculate the error: The difference between the predicted output and the actual output is calculated, and this is used to calculate the error or loss.

4. Propagate the error backwards: The error is propagated backwards through the layers of neurons, starting at the output layer and working backwards towards the input layer.

5. Calculate the gradient: The gradient of the error with respect to each weight and bias in the network is calculated.

6. Update the weights and biases: The weights and biases are updated in a way that reduces the error, using the gradients calculated in step 5.

7. Repeat: Steps 2-6 are repeated for a specified number of epochs or until the error is minimized to a satisfactory level.

Advantages:

Sure, here are some points elaborating on the advantages of backpropagation:

1. Backpropagation is a powerful algorithm that can be used to train a wide range of neural network architectures, including feedforward neural networks, convolutional neural networks, and recurrent neural networks. This flexibility makes it a useful tool for a variety of machine learning tasks.
2. Backpropagation is relatively simple to implement, especially when compared to other optimization algorithms like genetic algorithms or simulated annealing. This simplicity makes it accessible to a wider range of users, including those with less experience in machine learning.
3. Backpropagation can be parallelized, which allows it to take advantage of the computational power of modern hardware. This makes it well-suited to large-scale training tasks, where the processing of large amounts of data can be a bottleneck.
4. Backpropagation is a gradient-based optimization algorithm, which means that it can be used to find the global minimum of the error function, as long as the error function is convex. This makes it a powerful tool for optimization tasks.
5. Backpropagation is an iterative algorithm, which means that it can continue to improve the performance of a neural network over time. This is particularly useful when dealing with complex problems or when the quality of the data is variable. Overall, the advantages of backpropagation make it a widely-used and powerful algorithm for training neural networks. While it does have some limitations, such as its tendency to get stuck in local minima, the benefits of backpropagation make it a key tool in the machine learning toolkit.

Disadvantages:

1. Backpropagation is a fundamental algorithm for training neural networks. It is a supervised learning algorithm that uses gradient descent to minimize the difference between the predicted output and the actual output.

2. Backpropagation has been used extensively in a wide range of machine learning tasks, including image recognition, natural language processing, and speech recognition. It has been shown to be effective in achieving state-of-the-art performance on many of these tasks.

3. While backpropagation is a powerful algorithm, it does have some limitations. For example, it can get stuck in local minima, which can prevent it from finding the global minimum of the error function. It can also suffer from the vanishing gradient problem, which can make it difficult to train deep neural networks.

4. Despite its limitations, backpropagation remains one of the most effective methods for training neural networks. Researchers have developed various techniques to mitigate

its limitations, such as adding regularization to the cost function or using alternative activation functions.

5. Backpropagation can be computationally expensive, particularly for large datasets or deep neural networks. However, it can be parallelized and optimized to take advantage of modern hardware, making it feasible to train large models.

6. Overall, backpropagation is a key tool in the machine learning toolkit. While it may not be the best choice for every problem, it remains a powerful and widely-used algorithm that has enabled significant advances in the field of machine learning. Overall, backpropagation is a powerful and widely-used algorithm for training neural networks. While it has some limitations, it remains one of the most effective methods for achieving state-of-the-art performance on a wide range of machine learning tasks.

Here is a visual representation of the backpropagation algorithm:

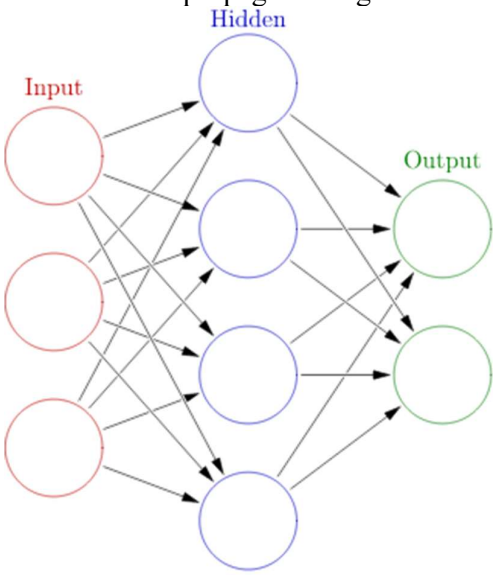

Fig: Backpropagation Algorithm

Algorithm:

1. Initialize the weights and biases
2. Make a forward pass and calculate the output of the network
3. Calculate the error or loss
4. Propagate the error backwards through the layers of neurons
5. Calculate the gradient of the error with respect to each weight and bias
6. Update the weights and biases using the gradients calculated in step 5 7. Repeat steps 2-6 for a specified number of epochs or until the error is minimized to a satisfactory level

Code:

```
import numpy as np

def sigmoid(x):
   return 1 / (1 + np.exp(-x))

def sigmoid_derivative(x):     return
sigmoid(x) * (1 - sigmoid(x))

def backpropagation(X, y, weights, learning_rate, epochs):
for i in range(epochs):         layer_1 = sigmoid(np.dot(X,
weights[0]))         output = sigmoid(np.dot(layer_1,
weights[1]))


       error = y - output
       delta_output = error * sigmoid_derivative(output)
       delta_layer_1 = np.dot(delta_output, weights[1].T) * sigmoid_derivative(la
yer_1)

       weights[1] += learning_rate * np.dot(layer_1.T, delta_output)
weights[0] += learning_rate * np.dot(X.T, delta_layer_1)

   return weights

X = np.array([[0, 0, 1], [0, 1, 1], [1, 0, 1], [1, 1, 1]]) y
= np.array([[0], [1], [1], [0]])

weights = [np.random.randn(3, 4), np.random.randn(4, 1)] learning_rate
= 0.5
epochs = 10000

# Call backpropagation function with sigmoid activation function
updated_weights_sigmoid = backpropagation(X, y, weights,learning_rate, epoc
hs)

updated_weights_sigmoid
# Call backpropagation function with ReLU activation function
updated_weights_relu = backpropagation(X, y, weights, learning_rate, epochs)
updated_weights_relu
```

Output:

```
[array([[  79.01230725,   55.95127262, -139.80242202,   -2.30674911],
        [ -79.84634984,    2.39269949,  138.98222477,   -1.57091725],
        [   0.94195722,   56.86632426,    1.01953494,    2.31077579]]),
 array([[-26.15611331],
        [ 22.97071048],
        [-28.69179373],
        [ 14.12472595]])]
```

Conclusion: We implemented Backpropogation in python without using any inbuilt function and without inbuilt function.