Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

AY: 2023-24

Class:	TE	**Semester:**	VI
Course Code:	CSL604	**Course Name:**	Machine Learning Lab

Name of Student:	Ojasi Prabhu
Roll No.:	43
Experiment No.:	8
Title of the Experiment:	Implementation of Error Backpropagation Perceptron Training Algorithm
Date of Performance:	
Date of Submission:	

Evaluation

Performance Indicator	Max. Marks	Marks Obtained
Performance	5	
Understanding	5	
Journal work and timely submission	10	
Total	20	

Performance Indicator	Exceed Expectations (EE)	Meet Expectations (ME)	Below Expectations (BE)
Performance	4-5	2-3	1
Understanding	4-5	2-3	1
Journal work and timely submission	8-10	5-8	1-4

Checked by

Name of Faculty : Mr Raunak Joshi

Signature :

Date :

Aim: Implementation of Error Backpropagation Perceptron Training Algorithm

Objective: Able to design a neural network and use activation function as learning rule that converges using a backpropagation algorithm.

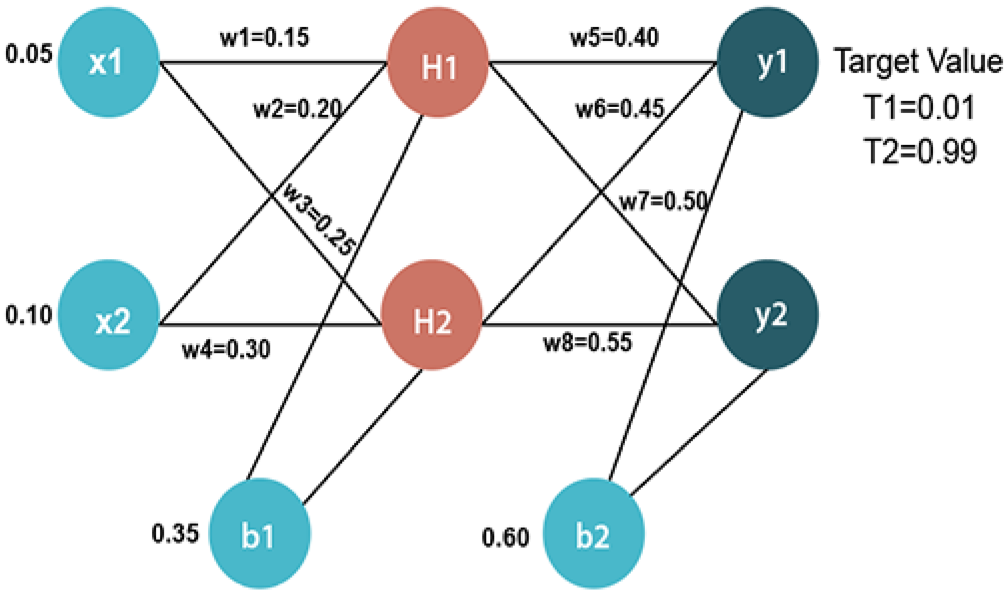
Theory:

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 to Backpropagation.

Now, how error function is used in Backpropagation and how Backpropagation works? Let start with an example and do it mathematically to understand how exactly updates the weight using Backpropagation.



Input values
X1=0.05
X2=0.10

Initial weight

W1=0.15 w5=0.40
W2=0.20 w6=0.45
W3=0.25 w7=0.50
W4=0.30 w8=0.55

Bias Values

b1=0.35 b2=0.60

Target Values

T1=0.01
T2=0.99

Now, we first calculate the values of H1 and H2 by a forward pass.

Forward Pass

To find the value of H1 we first multiply the input value from the weights as

$$H1 = x1 \times w_1 + x2 \times w_2 + b1$$
$$H1 = 0.05 \times 0.15 + 0.10 \times 0.20 + 0.35$$

H1=0.3775

To calculate the final result of H1, we performed the sigmoid function as

$$H1_{final} = \frac{1}{1 + \frac{1}{e^{H1}}}$$

$$H1_{final} = \frac{1}{1 + \frac{1}{e^{0.3775}}}$$

$$\mathbf{H1_{final} = 0.593269992}$$

We will calculate the value of H2 in the same way as H1

$$H2 = x1 \times w_3 + x2 \times w_4 + b1$$
$$H2 = 0.05 \times 0.25 + 0.10 \times 0.30 + 0.35$$

H2=0.3925

To calculate the final result of H1, we performed the sigmoid function as

$$H2_{final} = \frac{1}{1 + \frac{1}{e^{H2}}}$$

$$H2_{final} = \frac{1}{1 + \frac{1}{e^{0.3925}}}$$

$$\mathbf{H2_{final} = 0.596884378}$$

Now, we calculate the values of y1 and y2 in the same way as we calculate the H1 and H2.

To find the value of y1, we first multiply the input value i.e., the outcome of H1 and H2 from the weights as

$$y1 = H1 \times w_5 + H2 \times w_6 + b2$$
$$y1 = 0.593269992 \times 0.40 + 0.596884378 \times 0.45 + 0.60$$

y1=1.10590597

To calculate the final result of y1 we performed the sigmoid function as

$$y1_{final} = \frac{1}{1 + \frac{1}{e^{y1}}}$$

$$y1_{final} = \frac{1}{1 + \frac{1}{e^{1.10590597}}}$$

$$\mathbf{y1_{final} = 0.75136507}$$

We will calculate the value of y2 in the same way as y1

$$y2 = H1 \times w_7 + H2 \times w_8 + b2$$
$$y2 = 0.593269992 \times 0.50 + 0.596884378 \times 0.55 + 0.60$$

y2=1.2249214

To calculate the final result of H1, we performed the sigmoid function as

$$y2_{\text{final}} = \frac{1}{1+\frac{1}{e^{y2}}}$$

$$y2_{\text{final}} = \frac{1}{1+\frac{1}{e^{1.2249214}}}$$

$$\mathbf{y2_{final} = 0.772928465}$$

Our target values are 0.01 and 0.99. Our y1 and y2 value is not matched with our target values T1 and T2.

Now, we will find the **total error**, which is simply the difference between the outputs from the target outputs. The total error is calculated as

$$E_{\text{total}} = \sum \tfrac{1}{2}(\text{target} - \text{output})^2$$

So, the total error is

$$= \frac{1}{2}(\text{t}1 - \text{y}1_{\text{final}})^2 + \frac{1}{2}(\text{T}2 - \text{y}2_{\text{final}})^2$$

$$= \frac{1}{2}(0.01 - 0.75136507)^2 + \frac{1}{2}(0.99 - 0.772928465)^2$$

$$= 0.274811084 + 0.0235600257$$

$$\mathbf{E_{total} = 0.29837111}$$

Now, we will backpropagate this error to update the weights using a backward pass.

Backward pass at the output layer

To update the weight, we calculate the error correspond to each weight with the help of a total error. The error on weight w is calculated by differentiating total error with respect to w.

$$\text{Error}_{\text{w}} = \frac{\partial E_{\text{total}}}{\partial \text{w}}$$

We perform backward process so first consider the last weight w5 as

$$\mathrm{Error}_{w5} = \frac{\partial E_{total}}{\partial w5} \ldots\ldots\ldots..(1)$$

$$E_{total} = \frac{1}{2}(T1 - y1_{final})^2 + \frac{1}{2}(T2 - y2_{final})^2 \ldots\ldots\ldots..(2)$$

From equation two, it is clear that we cannot partially differentiate it with respect to w5 because there is no any w5. We split equation one into multiple terms so that we can easily differentiate it with respect to w5 as

$$\frac{\partial E_{total}}{\partial w5} = \frac{\partial E_{total}}{\partial y1_{final}} \times \frac{\partial y1\_final}{\partial y1} \times \frac{\partial y1}{\partial w5} \ldots\ldots\ldots\ldots (3)$$

Now, we calculate each term one by one to differentiate E_{total} with respect to w5 as

$$\frac{\partial E_{total}}{\partial y1_{final}} = \frac{\partial(\frac{1}{2}(T1 - y1_{final})^2 + \frac{1}{2}(T2 - y2_{final})^2)}{\partial y1_{final}}$$

$$= 2 \times \frac{1}{2} \times (T1 - y1_{final})^{2-1} \times (-1) + 0$$

$$= -(T1 - y1_{final})$$

$$= -(0.01 - 0.75136507)$$

$$\boldsymbol{\frac{\partial E_{total}}{\partial y1_{final}} = 0.74136507} \ldots\ldots\ldots.\mathbf{(4)}$$

$$y1_{final} = \frac{1}{1 + e^{-y1}} \ldots\ldots\ldots\ldots\ldots\ldots (5)$$

$$\frac{\partial y1_{final}}{\partial y1} = \frac{\partial(\frac{1}{1 + e^{-y1}})}{\partial y1}$$

$$= \frac{e^{-y1}}{(1 + e^{-y1})^2}$$

$$= e^{-y1} \times (y1_{final})^2 \ldots\ldots\ldots\ldots\ldots (6)$$

$$y1_{final} = \frac{1}{1 + e^{-y1}}$$

$$e^{-y1} = \frac{1 - y1_{final}}{y1_{final}} \ldots\ldots\ldots\ldots\ldots (7)$$

Putting the value of e^{-y} in equation (5)

$$= \frac{1 - y1_{final}}{y1_{final}} \times (y1_{final})^2$$

$$= y1_{final} \times (1 - y1_{final})$$

$$= 0.75136507 \times (1 - 0.75136507)$$

$$\frac{\partial \mathbf{y1_{final}}}{\partial \mathbf{y1}} = \mathbf{0.186815602} \ldots \ldots \ldots .(\mathbf{8})$$

$$y1 = H1_{final} \times w5 + H2_{final} \times w6 + b2 \ldots \ldots \ldots \ldots \ldots \ldots \ (9)$$

$$\frac{\partial y1}{\partial w5} = \frac{\partial (H1_{final} \times w5 + H2_{final} \times w6 + b2)}{\partial w5}$$

$$= H1_{final}$$

$$\frac{\partial \mathbf{y1}}{\partial \mathbf{w5}} = \mathbf{0.596884378} \ldots \ldots \ldots (\mathbf{10})$$

So, we put the values of $\frac{\partial E_{total}}{\partial y1_{final}}, \frac{\partial y1_{final}}{\partial y1}$, and $\frac{\partial y1}{\partial w5}$ in equation no (3) to find the final result.

$$\frac{\partial E_{total}}{\partial w5} = \frac{\partial E_{total}}{\partial y1_{final}} \times \frac{\partial y1_{final}}{\partial y1} \times \frac{\partial y1}{\partial w5}$$

$$= 0.74136507 \times 0.186815602 \times 0.593269992$$

$$\mathbf{Error_{w5}} = \frac{\partial \mathbf{E_{total}}}{\partial \mathbf{w5}} = \mathbf{0.0821670407} \ldots \ldots \ldots .(\mathbf{11})$$

Now, we will calculate the updated weight $w5_{new}$ with the help of the following formula

$$w5_{new} = w5 - \eta \times \frac{\partial E_{total}}{\partial w5} \text{ Here, } \eta = \text{learning rate} = 0.5$$

$$= 0.4 - 0.5 \times 0.0821670407$$

$$\mathbf{w5_{new}} = \mathbf{0.35891648} \ldots \ldots \ldots (\mathbf{12})$$

In the same way, we calculate $w6_{new}$, $w7_{new}$, and $w8_{new}$ and this will give us the following values

$w5_{new}$=0.35891648
$w6_{new}$=408666186
$w7_{new}$=0.511301270
$w8_{new}$=0.561370121

Backward pass at Hidden layer

Now, we will backpropagate to our hidden layer and update the weight w1, w2, w3, and w4 as we have done with w5, w6, w7, and w8 weights.

We will calculate the error at w1 as

$$\text{Error}_{w1} = \frac{\partial E_{total}}{\partial w1}$$

$$E_{total} = \frac{1}{2}(T1 - y1_{final})^2 + \frac{1}{2}(T2 - y2_{final})^2$$

From equation (2), it is clear that we cannot partially differentiate it with respect to w1 because there is no any w1. We split equation (1) into multiple terms so that we can easily differentiate it with respect to w1 as

$$\frac{\partial E_{total}}{\partial w1} = \frac{\partial E_{total}}{\partial H1_{final}} \times \frac{\partial H1\_final}{\partial H1} \times \frac{\partial H1}{\partial w1} \dots\dots\dots\dots\ (13)$$

Now, we calculate each term one by one to differentiate E_{total} with respect to w1 as

$$\frac{\partial E_{total}}{\partial H1_{final}} = \frac{\partial(\frac{1}{2}(T1 - y1_{final})^2 + \frac{1}{2}(T2 - y2_{final})^2)}{\partial H1} \dots\dots\dots\dots (14)$$

We again split this because there is no any $H1^{final}$ term in E^{toatal} as

$$\frac{\partial E_{total}}{\partial H1_{final}} = \frac{\partial E_1}{\partial H1_{final}} + \frac{\partial E_2}{\partial H1_{final}} \dots\dots\dots (15)$$

$\frac{\partial E_1}{\partial H1_{final}}$ and $\frac{\partial E_2}{\partial H1_{final}}$ will again split because in E1 and E2 there is no H1 term. Splitting is done as

$$\frac{\partial E_1}{\partial H1_{final}} = \frac{\partial E_1}{\partial y1} \times \frac{\partial y1}{\partial H1_{final}} \dots\dots\dots (16)$$

$$\frac{\partial E_2}{\partial H1_{final}} = \frac{\partial E_2}{\partial y2} \times \frac{\partial y2}{\partial H1\_final} \dots\dots\dots (17)$$

We again Split both $\frac{\partial E_1}{\partial y1}$ and $\frac{\partial E_2}{\partial y2}$ because there is no any y1 and y2 term in E1 and E2. We split it as

$$\frac{\partial E_1}{\partial y1} = \frac{\partial E_1}{\partial y1_{fianl}} \times \frac{\partial y1_{final}}{\partial y1} \ldots\ldots\ldots (18)$$

$$\frac{\partial E_2}{\partial y2} = \frac{\partial E_2}{\partial y2_{fianl}} \times \frac{\partial y2_{final}}{\partial y2} \ldots\ldots\ldots (19)$$

Now, we find the value of $\frac{\partial E_1}{\partial y1}$ and $\frac{\partial E_2}{\partial y2}$ by putting values in equation (18) and (19) as

From equation (18)

$$\frac{\partial E_1}{\partial y1} = \frac{\partial E_1}{\partial y1_{fianl}} \times \frac{\partial y1_{final}}{\partial y1}$$

$$= \frac{\partial(\frac{1}{2}(T1 - y1_{final})^2)}{\partial y1_{final}} \times \frac{\partial y1_{final}}{\partial y1}$$

$$= 2 \times \frac{1}{2}(T1 - y1_{final}) \times (-1) \times \frac{\partial y1_{final}}{\partial y1}$$

From equation (8)

$$= 2 \times \frac{1}{2}(0.01 - 0.75136507) \times (-1) \times \ 0.186815602$$

$$\boldsymbol{\frac{\partial E_1}{\partial y1} = 0.138498562} \ldots\ldots\ldots (20)$$

From equation (19)

$$\frac{\partial E_2}{\partial y2} = \frac{\partial E_2}{\partial y2_{fianl}} \times \frac{\partial y2_{final}}{\partial y2}$$

$$= \frac{\partial(\frac{1}{2}(T2 - y2_{final})^2)}{\partial y2_{final}} \times \frac{\partial y2_{final}}{\partial y2}$$

$$= 2 \times \frac{1}{2}(T2 - y2_{final}) \times (-1) \times \frac{\partial y2_{final}}{\partial y2} \dots\dots\dots (21)$$

$$y2_{final} = \frac{1}{1 + e^{-y2}} \dots\dots\dots\dots\dots\dots \ (22)$$

$$\frac{\partial y2_{fianl}}{\partial y2} = \frac{\partial(\frac{1}{1 + e^{-y2}})}{\partial y2}$$

$$= \frac{e^{-y2}}{(1 + e^{-y2})^2}$$

$$= e^{-y2} \times (y2_{final})^2 \dots\dots\dots\dots\dots (23)$$

$$y2_{final} = \frac{1}{1 + e^{-y2}}$$

$$e^{-y2} = \frac{1 - y2_{final}}{y2_{final}} \dots\dots\dots\dots\dots (24)$$

Putting the value of e^{-y2} in equation (23)

$$= \frac{1 - y2_{final}}{y2_{final}} \times (y2_{final})^2$$

$$= y2_{final} \times (1 - y2_{final})$$

$$= 0.772928465 \times (1 - 0.772928465)$$

$$\mathbf{\frac{\partial y2_{fianl}}{\partial y2} = 0.175510053 \dots\dots\dots . (25)}$$

From equation (21)

$$= 2 \times \frac{1}{2}(0.99 - 0.772928465) \times (-1) \times \ 0.175510053$$

$$\mathbf{\frac{\partial E_1}{\partial y1} = -0.0380982366126414 \dots\dots\dots (26)}$$

Now from equation (16) and (17)

$$\frac{\partial E_1}{\partial H1_{final}} = \frac{\partial E_1}{\partial y1} \times \frac{\partial y1}{\partial H1_{final}}$$

$$= 0.138498562 \times \frac{\partial(H1_{final} \times w_5 + H2_{final} \times w_6 + b2)}{\partial H1_{final}}$$

$$= 0.138498562 \times \frac{\partial(H1_{final} \times w_5 + H2_{final} \times w_6 + b2)}{\partial H1_{final}}$$

$$= 0.138498562 \times w5$$

$$= 0.138498562 \times 0.40$$

$$\mathbf{\frac{\partial E_1}{\partial H1_{final}} = 0.0553994248 \ldots\ldots\ldots (27)}$$

$$\frac{\partial E_2}{\partial H1_{final}} = \frac{\partial E_2}{\partial y2} \times \frac{\partial y2}{\partial H1\_final}$$

$$= -0.0380982366126414 \times \frac{\partial(H1_{final} \times w_7 + H2_{final} \times w_8 + b2)}{\partial H1_{final}}$$

$$= -0.0380982366126414 \times w7$$

$$= -0.0380982366126414 \times 0.50$$

$$\mathbf{\frac{\partial E_2}{\partial H1_{final}} = -0.0190491183063207 \ldots\ldots\ldots (28)}$$

Put the value of $\frac{\partial E_1}{\partial H1_{final}}$ and $\frac{\partial E_2}{\partial H1_{final}}$ in equation (15) as

$$\frac{\partial E_{total}}{\partial H1_{final}} = \frac{\partial E_1}{\partial H1_{final}} + \frac{\partial E_2}{\partial H1_{final}}$$

$$= 0.0553994248 + (-0.0190491183063207)$$

$$\mathbf{\frac{\partial E_{total}}{\partial H1_{final}} = 0.0364908241736793 \ldots\ldots\ldots (29)}$$

We have $\frac{\partial E_{total}}{\partial H1_{final}}$, we need to figure out $\frac{\partial H1\_final}{\partial H1}$, $\frac{\partial H1}{\partial w1}$ as

$$\frac{\partial \mathrm{H1_{final}}}{\partial \mathrm{H1}} = \frac{\partial(\frac{1}{1+e^{-H1}})}{\partial \mathrm{H1}}$$

$$= \frac{e^{-H1}}{(1+e^{-H1})^2}$$

$$e^{-H1} \times (\mathrm{H1_{final}})^2 \ldots\ldots\ldots (30)$$

$$\mathrm{H1_{final}} = \frac{1}{1+e^{-H1}}$$

$$e^{-H1} = \frac{1-\mathrm{H1_{final}}}{\mathrm{H1_{final}}} \ldots\ldots\ldots\ldots\ldots (31)$$

Putting the value of e^{-H1} in equation (30)

$$= \frac{1-\mathrm{H1_{final}}}{\mathrm{H1_{final}}} \times (\mathrm{H1_{final}})^2$$

$$= \mathrm{H1_{final}} \times (1-\mathrm{H1_{final}})$$

$$= 0.593269992 \times (1-0.593269992)$$

$$\frac{\partial \mathbf{H1_{final}}}{\partial \mathbf{H1}} = \mathbf{0.2413007085923199}$$

We calculate the partial derivative of the total net input to H1 with respect to w1 the same as we did for the output neuron:

$$\mathrm{H1} = \mathrm{H1_{final}} \times \mathrm{w5} + \mathrm{H2_{final}} \times \mathrm{w6} + \mathrm{b2} \ldots\ldots\ldots\ldots\ldots\ldots\ (32)$$

$$\frac{\partial \mathrm{y1}}{\partial \mathrm{w1}} = \frac{\partial(\mathrm{x1} \times \mathrm{w1} + \mathrm{x2} \times \mathrm{w3} + \mathrm{b1} \times 1)}{\partial \mathrm{w1}}$$

$$= \mathrm{x1}$$

$$\frac{\partial \mathbf{H1}}{\partial \mathbf{w1}} = \mathbf{0.05} \ldots\ldots\ldots (\mathbf{33})$$

So, we put the values of $\frac{\partial \mathrm{E_{total}}}{\partial \mathrm{H1_{final}}}, \frac{\partial \mathrm{H1_{final}}}{\partial \mathrm{H1}}$, and $\frac{\partial \mathrm{H1}}{\partial \mathrm{w1}}$ in equation (13) to find the final result.

$$\frac{\partial \mathrm{E_{total}}}{\partial \mathrm{w1}} = \frac{\partial \mathrm{E_{total}}}{\partial \mathrm{H1_{final}}} \times \frac{\partial \mathrm{H1_{final}}}{\partial \mathrm{H1}} \times \frac{\partial \mathrm{H1}}{\partial \mathrm{w1}}$$

$$= 0.0364908241736793 \times 0.2413007085923199 \times 0.05$$

$$\mathbf{Error_{w1}} = \frac{\partial \mathbf{E_{total}}}{\partial \mathbf{w1}} = \mathbf{0.000438568} \ldots\ldots\ldots. (\mathbf{34})$$

Now, we will calculate the updated weight $w1_{new}$ with the help of the following formula

$$w1_{new} = w1 - \eta \times \frac{\partial E_{total}}{\partial w1} \text{ Here } \eta = \text{learning rate} = 0.5$$

$$= 0.15 - 0.5 \times 0.000438568$$

$$\mathbf{w1_{new} = 0.149780716 \ldots\ldots\ldots(35)}$$

In the same way, we calculate $w2_{new}$,$w3_{new}$, and w4 and this will give us the following values

$w1_{new}$=0.149780716
$w2_{new}$=0.19956143
$w3_{new}$=0.24975114
$w4_{new}$=0.29950229

We have updated all the weights. We found the error 0.298371109 on the network when we fed forward the 0.05 and 0.1 inputs. In the first round of Backpropagation, the total error is down to 0.291027924. After repeating this process 10,000, the total error is down to 0.0000351085. At this point, the outputs neurons generate 0.159121960 and 0.984065734 i.e., nearby our target value when we feed forward the 0.05 and 0.1.

Implementation:

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

# Loading dataset
data = load_iris()
X = data.data
y = data.target

# Split dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=20, random_state=4)

# Hyperparameters
learning_rate = 0.1
iterations = 5000
N = y_train.size
input_size = 4
hidden_size = 2
output_size = 3

np.random.seed(10)
W1 = np.random.normal(scale=0.5, size=(input_size, hidden_size))
W2 = np.random.normal(scale=0.5, size=(hidden_size, output_size))

# Helper functions

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

```

```python
def mean_squared_error(y_pred, y_true):
    # One-hot encode y_true (i.e., convert [0, 1, 2] into [[1, 0, 0], [0, 1, 0], [0, 0, 1]])
    y_true_one_hot = np.eye(output_size)[y_true]

    # Reshape y_true_one_hot to match y_pred shape
    y_true_reshaped = y_true_one_hot.reshape(y_pred.shape)

    # Compute the mean squared error between y_pred and y_true_reshaped
    error = ((y_pred - y_true_reshaped)**2).sum() / (2*y_pred.size)

    return error

def accuracy(y_pred, y_true):
    acc = y_pred.argmax(axis=1) ==  y_true.argmax(axis=1)
    return acc.mean()

results = pd.DataFrame(columns=["mse", "accuracy"])

# Training loop

for itr in range(iterations):
    # Feedforward propagation
    Z1 = np.dot(X_train, W1)
    A1 = sigmoid(Z1)
    Z2 = np.dot(A1, W2)
    A2 = sigmoid(Z2)

    # Calculate error
    mse = mean_squared_error(A2, y_train)
    acc = accuracy(np.eye(output_size)[y_train], A2)
    new_row = pd.DataFrame({"mse": [mse], "accuracy": [acc]})
```


```python
    new_row = pd.DataFrame({"mse": [mse], "accuracy": [acc]})
```


```python
    results = pd.concat([results, new_row], ignore_index=True)

    # Backpropagation
    E1 = A2 - np.eye(output_size)[y_train]
    dW1 = E1 * A2 * (1 - A2)
    E2 = np.dot(dW1, W2.T)
    dW2 = E2 * A1 * (1 - A1)

    # Update weights
    W2_update = np.dot(A1.T, dW1) / N
    W1_update = np.dot(X_train.T, dW2) / N
    W2 = W2 - learning_rate * W2_update
    W1 = W1 - learning_rate * W1_update

# Visualizing the results

results.mse.plot(title="Mean Squared Error")
plt.show()

results.accuracy.plot(title="Accuracy")
plt.show()

# Test the model

Z1 = np.dot(X_test, W1)
A1 = sigmoid(Z1)
Z2 = np.dot(A1, W2)
A2 = sigmoid(Z2)
test_acc = accuracy(np.eye(output_size)[y_test], A2)
print("Test accuracy: {}".format(test_acc))
```

Conclusion:

The Error Backpropagation Perceptron Training Algorithm implements a powerful technique for training neural networks with several layers. By propagating errors backward through the network, it adjusts weights to reduce the discrepancy between expected and actual output. This recurrent approach allows the network to learn complicated relationships in the data and generalize to previously unseen examples. Despite its effectiveness, backpropagation incurs processing costs and may encounter problems such as vanishing or expanding gradients. However, due to its adaptability and extensive application, it has become a cornerstone of current deep learning, revolutionizing domains such as computer vision, natural language processing, and reinforcement learning. .