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)

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE CODE: DJ22ITL502 **DATE: 29/10/24**

COURSE NAME: Artificial Intelligence Laboratory **CLASS: TY-IT**

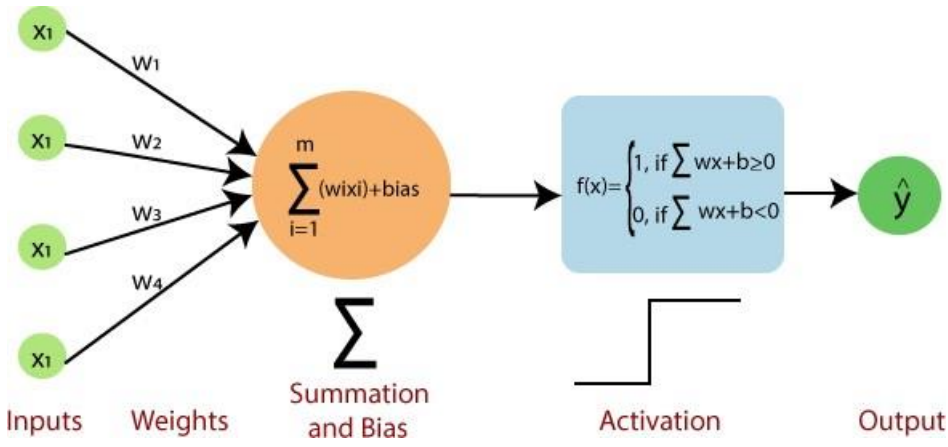
Name: Anish Sharma

EXPERIMENT NO.07

CO/LO: Apply various AI approaches to knowledge intensive problem solving, reasoning, planning and uncertainty.
AIM / OBJECTIVE: Implement learning: Perceptron Learning / Backpropagation Algorithm

DESCRIPTION OF EXPERIMENT:

The **Perceptron** is a type of artificial neuron that performs binary classification. It maps its input x (a real-valued vector) to an output y (a binary value) using a linear transformation.



Mathematical Representation:

The perceptron works by applying a weight w to the input vector x and computing the sum z using a bias b as:

$$z = \mathbf{w} \cdot \mathbf{x} + b$$

The activation function applies a threshold to the computed value z:

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)

$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

Learning Rule**: If the prediction y does not match the actual label y^, the weights are updated using:**

$$\mathbf{w} = \mathbf{w} + \eta(\hat{y} - y)\mathbf{x}$$

$$b = b + \eta(\hat{y} - y)$$

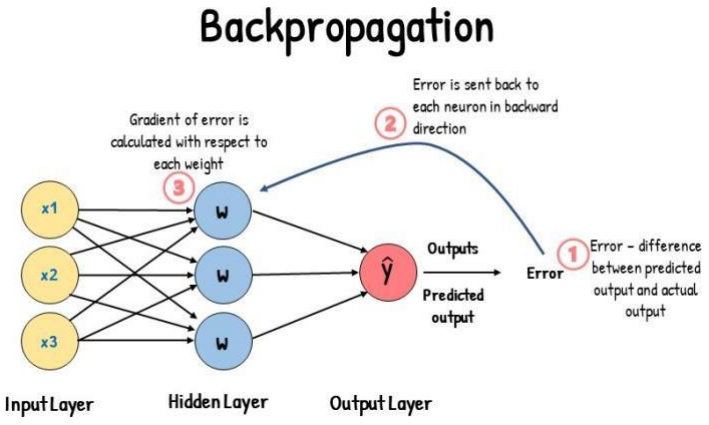
Where:

- η is the learning rate,
- y^ is the actual label, • y is the predicted label.

2. Backpropagation Algorithm:

Backpropagation is a supervised learning algorithm used for training multi-layered neural networks. The algorithm consists of two passes:

1. **Forward Pass**: Propagate the input through the network and compute the output.
2. **Backward Pass**: Compute the error and propagate it backward to adjust the weights.



Key Concepts:

- **Activation Function**: A differentiable function, often sigmoid, used in the neurons. For a neuron output o:

$$o = \frac{1}{1 + e^{-z}}$$

- **Error Calculation**: The difference between the actual and predicted output. For the output layer:

$$\delta_{\text{output}} = (y - \hat{y}) \cdot \text{sigmoid\_derivative}(o)$$

Weight Update: The weights are updated using the gradient of the error with respect to the weights. For the hidden layer weights:

$$\mathbf{w} = \mathbf{w} + \eta \cdot \delta \cdot \mathbf{x}$$

This process is repeated for a number of epochs to minimize the error and improve the model's accuracy.

Procedure:

Perceptron Learning Algorithm Implementation:

1. Initialize the weights and bias to small random values or zero.
2. Input the dataset XXX with corresponding labels yyy for binary classification.
3. For each training sample:
    - Calculate the dot product of the input and weights, then apply the activation function.
    - Update the weights if the prediction is incorrect using the Perceptron learning rule.
4. Repeat this process for a fixed number of epochs until the model converges (i.e., makes accurate predictions for all data points).

Backpropagation Algorithm Implementation:

1. Initialize the weights of the input-hidden and hidden-output layers randomly.
2. Input the dataset XXX with corresponding labels yyy.
3. Forward Pass:
    - Compute the activations of the hidden layer using the sigmoid function.

- Compute the output layer's activation.

4. Backward Pass:
    - Calculate the error between the predicted and actual output.
    - Update the hidden-output and input-hidden layer weights using the gradient of the error.
5. Train the model for a number of epochs until the error is minimized.
6. Test the model using new input data and compare the predicted outputs with actual labels.

EXPLANATION / SOLUTIONS (DESIGN):

Code:

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)

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

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

def sigmoid_derivative(x):
    return x * (1 - x)
 input_size = 3
hidden_size = 4
output_size = 1
learning_rate =
0.1
epochs = 10000

np.random.seed(42)            weights_input_hidden            =
np.random.rand(input_size, hidden_size) weights_hidden_output =
np.random.rand(hidden_size, output_size)

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

for epoch in range(epochs):
    hidden_layer_input = np.dot(X, weights_input_hidden)
    hidden_layer_output = sigmoid(hidden_layer_input)
```

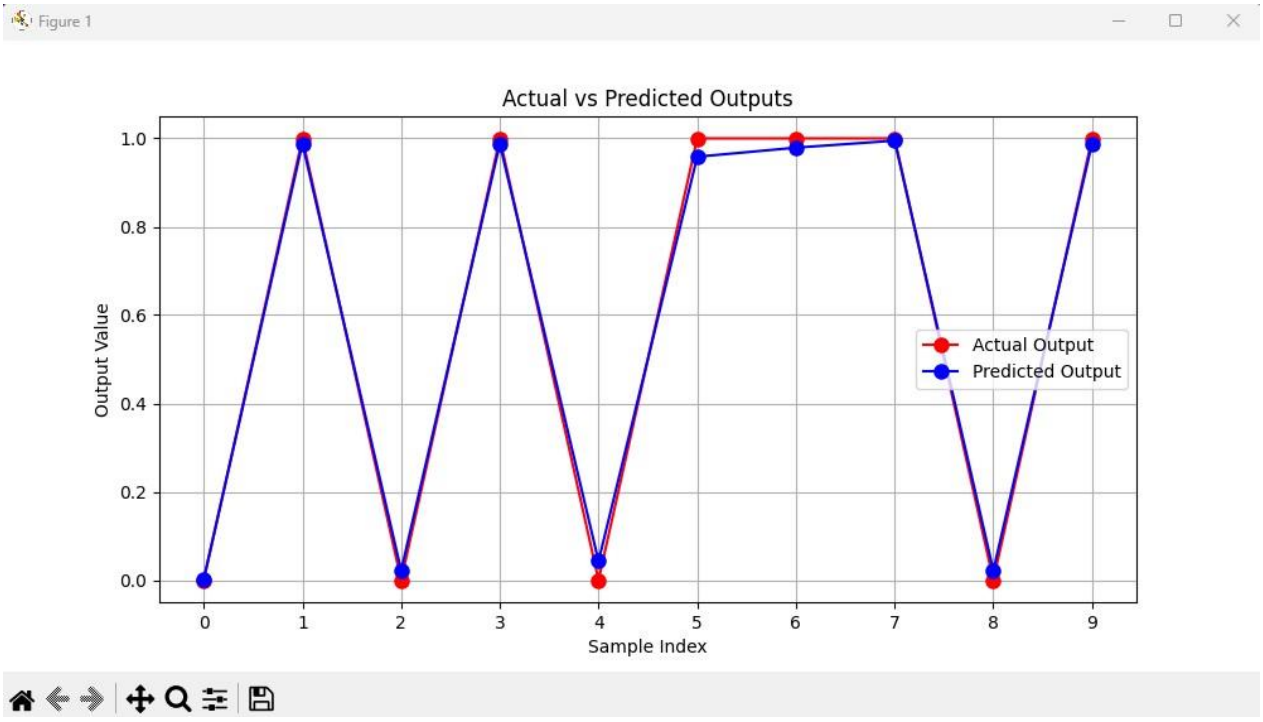
Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

```
    output_layer_input   =   np.dot(hidden_layer_output,   weights_hidden_output)
    predicted_output = sigmoid(output_layer_input)
 error = y - predicted_output
    d_predicted_output = error * sigmoid_derivative(predicted_output)

    weights_hidden_output += hidden_layer_output.T.dot(d_predicted_output) *
learning_rate
 error_hidden_layer = d_predicted_output.dot(weights_hidden_output.T)
    d_hidden_layer_output = error_hidden_layer *
sigmoid_derivative(hidden_layer_output) weights_input_hidden +=
    X.T.dot(d_hidden_layer_output) * learning_rate

hidden_layer_input        =        np.dot(X,        weights_input_hidden)
hidden_layer_output = sigmoid(hidden_layer_input) output_layer_input =
np.dot(hidden_layer_output,  weights_hidden_output)  predicted_output  =
sigmoid(output_layer_input)
 print("Predicted Output after
training:") print(predicted_output)
 plt.figure(figsize=(10,   5))   plt.plot(range(len(y)),   y,   'ro-',
label='Actual Output', markersize=8)
plt.plot(range(len(predicted_output)), predicted_output, 'bo-',
label='Predicted Output', markersize=8) plt.title('Actual vs Predicted
Outputs') plt.xlabel('Sample Index') plt.ylabel('Output Value')
plt.xticks(range(len(y))) plt.legend() plt.grid() plt.show()
```

Output:

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)

```
Predicted Output after training:
[[0.00211111]
 [0.98855991]
 [0.02151933]
 [0.98754493]
 [0.04286148]
 [0.95848878]
 [0.97900408]
 [0.99493926]
 [0.02151933]
 [0.98855991]]
```



CONCLUSION: Implementing the Perceptron Learning Algorithm or Backpropagation allows a neural network to learn from its errors, iteratively adjusting weights to minimize the difference between predicted and actual outputs, ultimately improving accuracy in classification tasks.

REFERENCES:

[1] Stuart Russell and Peter Norvig, "Artificial Intelligence: A Modern Approach", 2nd Edition,Pearson Education, 2010