Experiment :9

Problem statement:

Implement the classification system using Back-propagation algorithm

Aim: to Implement the classification system using Back-propagation algorithm

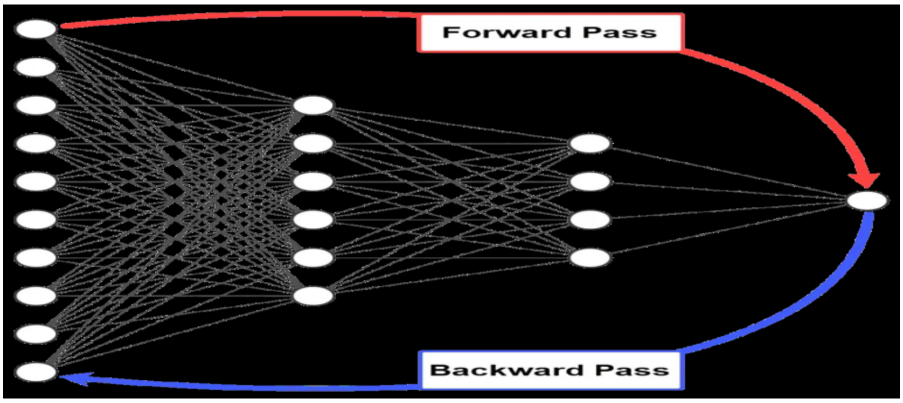
ALGORITHM:

Step 1: Start

Step 2: The architecture of a neural network consists of some sequential layers, where thelayer numbered *i* is connected to the layer numbered *i+1*.

Step 3: The layers can be classified into 3 classes:

- Input
- Hidden
- Output



Step 4: each neuron in the hidden layer uses an activation function like sigmoid or rectified linear unit (ReLU).

Step 5: The neurons in the output layer also use activation functions like sigmoid (for regression) or SoftMax (for classification).

Step 6: To train a neural network, there are 2 passes (phases):

- Forward
- Backward

Step 7: The forward and backward phases are repeated from some epochs. In each epoch, the following occurs:

7.1. The inputs are propagated from the input to the output layer.

7.2. The network error is calculated.

7.3. The error is propagated from the output layer to the input layer.

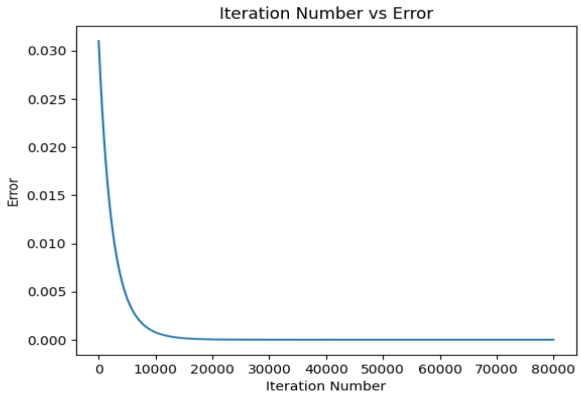
Step 8: To minimize network error, wemust change something in the network

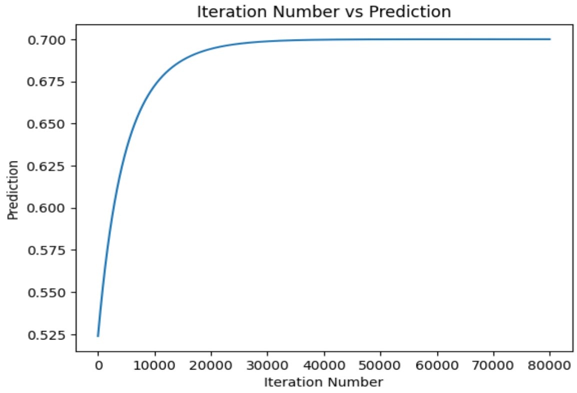
Step 9: End

PROGRAM:

```
import numpy
import matplotlib.pyplot as plt
def sigmoid(sop):
 return 1.0/(1+numpy.exp(-1*sop))
def error(predicted, target):
 return numpy.power(predicted-target, 2)
def error_predicted_deriv(predicted, target):
 return 2*(predicted-target)
def sigmoid_sop_deriv(sop):
 return sigmoid(sop)*(1.0-sigmoid(sop))
def sop_w_deriv(x):
 return x
def update_w(w, grad, learning_rate):
 return w - learning_rate*grad
x1=0.1
x2=0.4
target = 0.7
learning_rate = 0.01
w1=numpy.random.rand()
w2=numpy.random.rand()
print("Initial W : ", w1, w2)
predicted_output = []
network_error = []
old_err = 0
for k in range(80000):
# Forward Pass
 y = w1*x1 + w2*x2
 predicted = sigmoid(y)
 err = error(predicted, target)
 predicted_output.append(predicted)
 network_error.append(err)
# Backward Pass
 g1 = error_predicted_deriv(predicted, target)
 g2 = sigmoid_sop_deriv(y)
 g3w1 = sop_w_deriv(x1)
 g3w2 = sop_w_deriv(x2)
 gradw1 = g3w1*g2*g1
 gradw2 = g3w2*g2*g1
 w1 = update_w(w1, gradw1, learning_rate)
 w2 = update_w(w2, gradw2, learning_rate)
#print(predicted)
plt.figure()
plt.plot(network_error)
plt.title("Iteration Number vs Error")
plt.xlabel("Iteration Number")
plt.ylabel("Error")
plt.show()
plt.figure()
plt.plot(predicted_output)
plt.title("Iteration Number vs Prediction")
plt.xlabel("Iteration Number")
plt.ylabel("Prediction")
plt.show()
```

OUTPUT:





Result: The program has been executed successfully and the back propagation algorithm is implemented.