Experiment-7

Aim: Build an Artificial Neural Network by implementing the Back propagation algorithm and test the same using appropriate data sets.

Program:

```
import numpy as np

def sigmoid(x):

   return 1 / (1 + np.exp(-x))

def derivatives_sigmoid(x):

   return x * (1 - x)

X_train = np.array([[2, 9], [1, 5], [3, 6]], dtype=float)

y_train = np.array([[92], [86], [89]], dtype=float)

X_test = np.array([[4, 8], [5, 3]], dtype=float)

y_test = np.array([[95], [82]], dtype=float)


X_train_normalized = X_train / np.amax(X_train, axis=0)

X_test_normalized = X_test / np.amax(X_test, axis=0)


epoch = 7000

lr = 0.1

inputlayer_neurons = 2

hiddenlayer_neurons = 3

output_neurons = 1
```

```
wh = np.random.uniform(size=(inputlayer_neurons, hiddenlayer_neurons))
bh = np.random.uniform(size=(1, hiddenlayer_neurons))
wout = np.random.uniform(size=(hiddenlayer_neurons, output_neurons))
bout = np.random.uniform(size=(1, output_neurons))

for i in range(epoch):
    hinp1 = np.dot(X_train_normalized, wh)
    hinp = hinp1 + bh
    hlayer_act = sigmoid(hinp)
    outinp1 = np.dot(hlayer_act, wout)
    outinp = outinp1 + bout
    output = sigmoid(outinp)

    EO = y_train - output
    outgrad = derivatives_sigmoid(output)
    d_output = EO * outgrad

    EH = d_output.dot(wout.T)
    hiddengrad = derivatives_sigmoid(hlayer_act)
    d_hiddenlayer = EH * hiddengrad

    wout += hlayer_act.T.dot(d_output) * lr
    bout += np.sum(d_output, axis=0, keepdims=True) * lr
    wh += X_train_normalized.T.dot(d_hiddenlayer) * lr
    bh += np.sum(d_hiddenlayer, axis=0, keepdims=True) * lr

hinp1_test = np.dot(X_test_normalized, wh)
hinp_test = hinp1_test + bh
hlayer_act_test = sigmoid(hinp_test)
outinp1_test = np.dot(hlayer_act_test, wout)
outinp_test = outinp1_test + bout
output_test = sigmoid(outinp_test)

print("Input Test Data:\n", X_test_normalized)
print("Actual Output Test:\n", y_test)
print("Predicted Output Test:\n", output_test)

mse = np.mean((output_test - y_test) ** 2)
print("Mean Squared Error:", mse)
```

Output:

```
Input Test Data:
 [[0.8   1.   ]
 [1.    0.375]]
Actual Output Test:
 [[95.]
 [82.]]
Predicted Output Test:
 [[0.99999965]
 [0.99999946]]
Mean Squared Error: 7698.500076612787
```