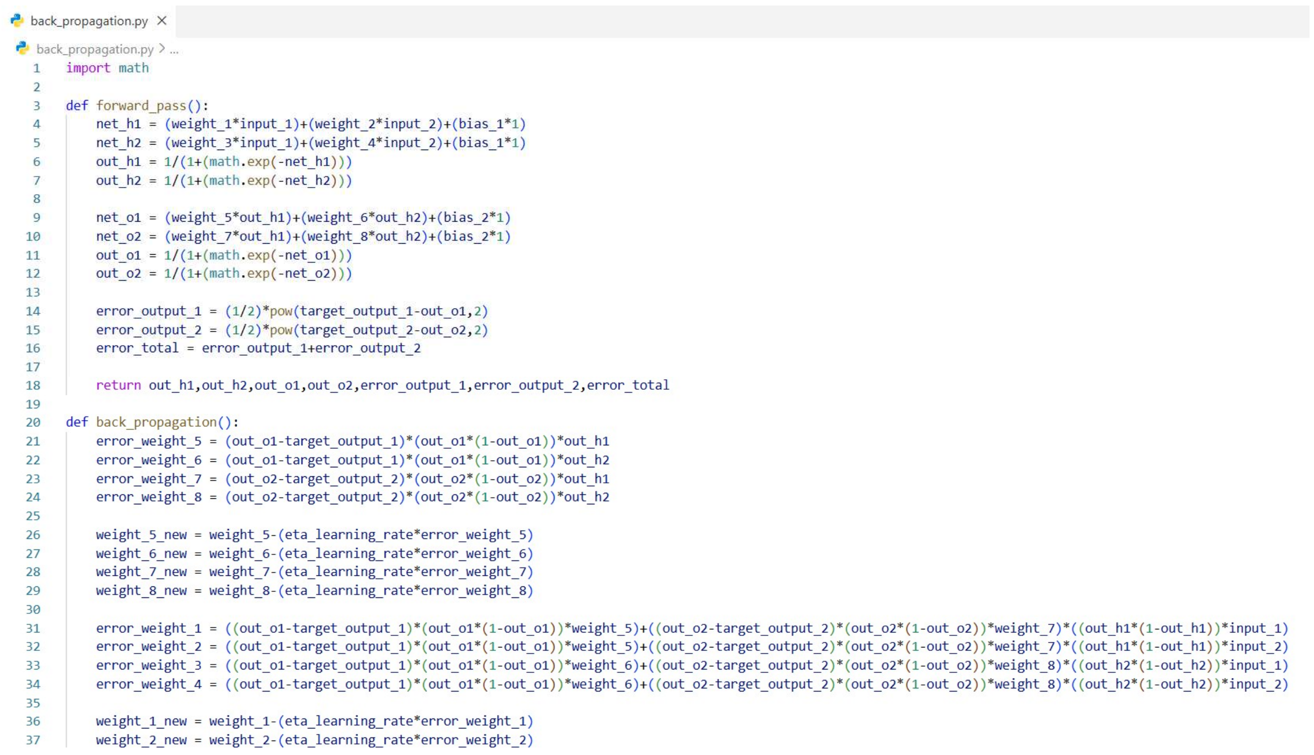
Experiment No: 12

Objective: Forward Pass & Back Propagation program.

Theory: The goal of backpropagation is to optimize the weights so that the neural network can learn how to correctly map arbitrary inputs to outputs.

Source Code:



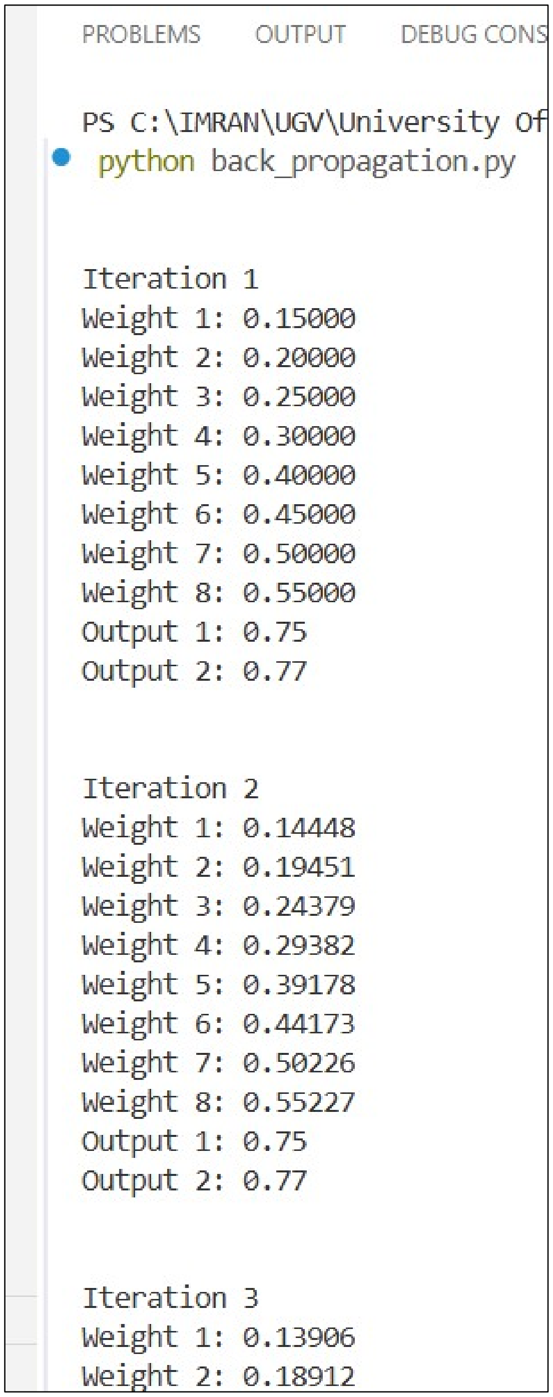
```python
import math

def forward_pass():
    net_h1 = (weight_1*input_1)+(weight_2*input_2)+(bias_1*1)
    net_h2 = (weight_3*input_1)+(weight_4*input_2)+(bias_1*1)
    out_h1 = 1/(1+(math.exp(-net_h1)))
    out_h2 = 1/(1+(math.exp(-net_h2)))

    net_o1 = (weight_5*out_h1)+(weight_6*out_h2)+(bias_2*1)
    net_o2 = (weight_7*out_h1)+(weight_8*out_h2)+(bias_2*1)
    out_o1 = 1/(1+(math.exp(-net_o1)))
    out_o2 = 1/(1+(math.exp(-net_o2)))

    error_output_1 = (1/2)*pow(target_output_1-out_o1,2)
    error_output_2 = (1/2)*pow(target_output_2-out_o2,2)
    error_total = error_output_1+error_output_2

    return out_h1,out_h2,out_o1,out_o2,error_output_1,error_output_2,error_total

def back_propagation():
    error_weight_5 = (out_o1-target_output_1)*(out_o1*(1-out_o1))*out_h1
    error_weight_6 = (out_o1-target_output_1)*(out_o1*(1-out_o1))*out_h2
    error_weight_7 = (out_o2-target_output_2)*(out_o2*(1-out_o2))*out_h1
    error_weight_8 = (out_o2-target_output_2)*(out_o2*(1-out_o2))*out_h2

    weight_5_new = weight_5-(eta_learning_rate*error_weight_5)
    weight_6_new = weight_6-(eta_learning_rate*error_weight_6)
    weight_7_new = weight_7-(eta_learning_rate*error_weight_7)
    weight_8_new = weight_8-(eta_learning_rate*error_weight_8)

    error_weight_1 = ((out_o1-target_output_1)*(out_o1*(1-out_o1))*weight_5)+((out_o2-target_output_2)*(out_o2*(1-out_o2))*weight_7)*((out_h1*(1-out_h1))*input_1)
    error_weight_2 = ((out_o1-target_output_1)*(out_o1*(1-out_o1))*weight_5)+((out_o2-target_output_2)*(out_o2*(1-out_o2))*weight_7)*((out_h1*(1-out_h1))*input_2)
    error_weight_3 = ((out_o1-target_output_1)*(out_o1*(1-out_o1))*weight_6)+((out_o2-target_output_2)*(out_o2*(1-out_o2))*weight_8)*((out_h2*(1-out_h2))*input_1)
    error_weight_4 = ((out_o1-target_output_1)*(out_o1*(1-out_o1))*weight_6)+((out_o2-target_output_2)*(out_o2*(1-out_o2))*weight_8)*((out_h2*(1-out_h2))*input_2)

    weight_1_new = weight_1-(eta_learning_rate*error_weight_1)
    weight_2_new = weight_2-(eta_learning_rate*error_weight_2)
```

Source Code On GitHub:
https://github.com/MdImranKhanSiam/Problem_Solving/blob/main/Practice/All_Program/Python/Back_propagation/back_propagation.py

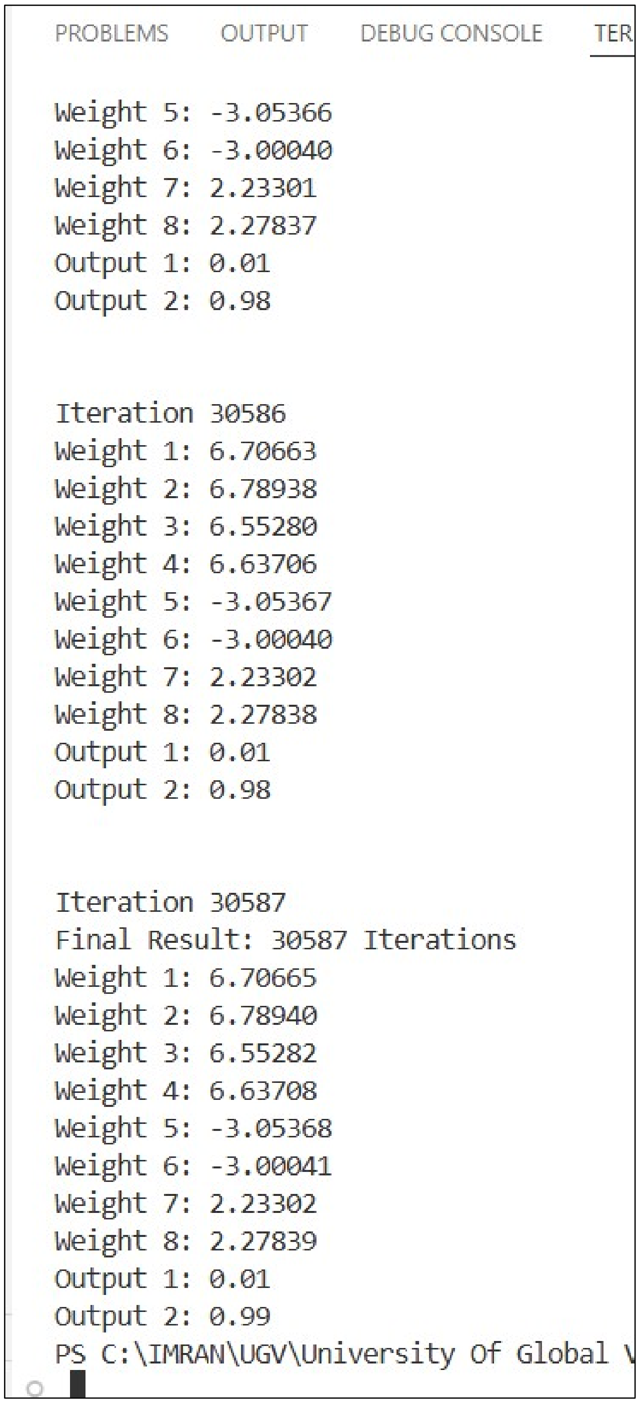
Algorithm:

1. Start
2. Initialize:
    - Learning rate
    - Input 1, 2
    - Target output 1, 2
    - Weight 1, 2, 3, 4, 5, 6, 7, 8
    - Bias 1, 2
3. Repeat for N iterations:
4. Forward Pass
    - Compute hidden layer nets:
        i. net_h1, net_h2
    - Compute hidden activations using sigmoid:
        i. out_h1, out_h2
    - Compute output layer nets:
        i. net_o1, net_o2
    - Compute output activations:
        i. out_o1, out_o2
    - Compute total error:
5. Backpropagation
    - Compute gradients for output layer weights:
    - Update output layer weights.
    - Compute gradients for hidden layer weights:
    - Update hidden layer weights.
6. Check stopping condition:
    - If network output matches target output then stop the iterations.
7. After final iteration, print final weights and outputs.
8. End

Sample Input/Output:



```
PS C:\IMRAN\UGV\University Of
 python back_propagation.py


Iteration 1
Weight 1: 0.15000
Weight 2: 0.20000
Weight 3: 0.25000
Weight 4: 0.30000
Weight 5: 0.40000
Weight 6: 0.45000
Weight 7: 0.50000
Weight 8: 0.55000
Output 1: 0.75
Output 2: 0.77


Iteration 2
Weight 1: 0.14448
Weight 2: 0.19451
Weight 3: 0.24379
Weight 4: 0.29382
Weight 5: 0.39178
Weight 6: 0.44173
Weight 7: 0.50226
Weight 8: 0.55227
Output 1: 0.75
Output 2: 0.77


Iteration 3
Weight 1: 0.13906
Weight 2: 0.18912
```



```
Weight 5: -3.05366
Weight 6: -3.00040
Weight 7: 2.23301
Weight 8: 2.27837
Output 1: 0.01
Output 2: 0.98


Iteration 30586
Weight 1: 6.70663
Weight 2: 6.78938
Weight 3: 6.55280
Weight 4: 6.63706
Weight 5: -3.05367
Weight 6: -3.00040
Weight 7: 2.23302
Weight 8: 2.27838
Output 1: 0.01
Output 2: 0.98


Iteration 30587
Final Result: 30587 Iterations
Weight 1: 6.70665
Weight 2: 6.78940
Weight 3: 6.55282
Weight 4: 6.63708
Weight 5: -3.05368
Weight 6: -3.00041
Weight 7: 2.23302
Weight 8: 2.27839
Output 1: 0.01
Output 2: 0.99
PS C:\IMRAN\UGV\University Of Global V
```