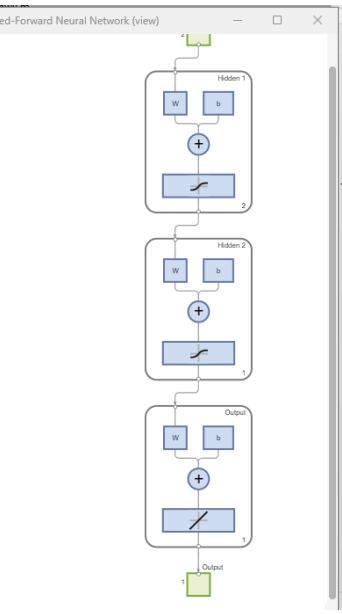
Matlab script(hw8.m)

```
clear;
clc;
close all;

% XOR gate input and output data
P = [0 0; 0 1; 1 0; 1 1];%Input
T = [0; 1; 1; 0]; %Output

% Sigmoid function
sigmoid = @(x) 1 / (1 + exp(-x));

% Part A: (A) Consider a single-layer ANN with two neuron
%Start off with the notation parameters as stated in assignment
w11_A = 1.0; % Weight w11^1
w12_A = -1.0; % Weight w12^2
w21_A = -1.0; % Weight w21^1
w22_A = 1.0; % Weight w22^2
theta1_A = 0.0; % Bias theta1^1
theta2_A = 0.0; % Bias theta2^1
net_A = fitnet(2, 'trainlm'); % Networking code
net_A = train(net_A, P', T');
output_A = net_A(P');

% Part B: (B) Consider a single-layer ANN with three neurons
%Start off with the notation parameters as stated in assignment
w11_B = 1.0; % Weight w11^1
w12_B = -1.0; % Weight w12^2
w21_B = -1.0; % Weight w21^1
w22_B = 1.0; % Weight w22^2
w31_B = 0.0; % Weight w31^1
w32_B = 0.0; % Weight w32^2
theta1_B = 0.0; % Bias theta1^1
theta2_B = 0.0; % Bias theta2^1
theta3_B = 0.0; % Bias theta3^1
net_B = fitnet(3, 'trainlm');% Networking code
net_B = train(net_B, P', T');
output_B = net_B(P');

% Part C: Consider a two-layer ANN with two neurons in
%the first layer and one in the second layer
%Start off with the notation parameters as stated in assignment
w11_C = 1.0; % Weight w11^1
w12_C = -1.0; % Weight w12^2
w21_C = -1.0; % Weight w21^1
```

```
w22_C = 1.0; % Weight w22^2
theta1_C = 0.0; % Bias theta1^1
theta2_C = 0.0; % Bias theta2^1
theta1_2_C = 0.0; % Bias theta1^2
net_C = feedforwardnet([2, 1], 'trainlm');% Networking code
net_C = train(net_C, P', T');
output_C = net_C(P');

% Calculate the outputs of each model
for i = 1:4 %1 and 4 being zeroes
  x1 = P(i, 1);
  x2 = P(i, 2);
  %Now bring everything together(sigmoid, parameters, network code) for
  %the outputs

% Part A output
y_A = w11_A * sigmoid(w11_A * x1 + w12_A * x2 + theta1_A) + w21_A *
sigmoid(w21_A * x1 + w22_A * x2 + theta2_A);
output_A(i) = round(sigmoid(y_A));
% Part B output
y_B = w11_B * sigmoid(w11_B * x1 + w12_B * x2 + theta1_B) + w21_B *
sigmoid(w21_B * x1 + w22_B * x2 + theta2_B) + w31_B * sigmoid(w31_B * x1 +
w32_B * x2 + theta3_B);
output_B(i) = round(sigmoid(y_B));
% Part C output
y_C = w11_C * sigmoid(w11_C * x1 + w12_C * x2 + theta1_C) + w12_C *
sigmoid(w21_C * x1 + w22_C * x2 + theta2_C + theta1_2_C);
output_C(i) = round(sigmoid(y_C));
End

% Display the outputs
disp("A Output:");
disp(output_A);
disp("B Output:");
disp(output_B);
disp("C Output:");
disp(output_C);

%I used the functions and code from the slides after this HW, I probably
%did something incorrect as the output is kinda weird. I added round functions
in the inputs for clearer results My guess
%is, the values that I did something incorrect with the parameters
```

Output on Matlab



Command Window

```
A Output:
     1     0     1     1

B Output:
     1     0     1     1

C Output:
     1     0     1     1

fx >>
```

Zoom: 90% UTF-8

Network Diagram

Training Results

Training finished: Reached minimum gradient

Training Progress

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	5	1000
Elapsed Time	-	00:00:00	-
Performance	0.248	3.68e-23	0
Gradient	0.164	5.85e-12	1e-07
Mu	0.001	1e-06	1e+10
Validation Checks	0	3	6

Training Algorithms

Data Division: Random dividerand
Training: Levenberg-Marquardt trainlm
Performance: Mean Squared Error mse
Calculations: MEX

Training Plots

Performance | Training State
Error Histogram | Regression