Table of Contents

ECG703 HW5, Gradient Descent and Neural Networks

James Skelly

```
clear all
close all
```

Problem 1: Gradient Descent on a "Simple" Function

```
% Consider the function:

syms x y
f = symfun(x^2 + 2*y^2 + 2*sin(2*pi*x)*sin(2*pi*y),[x y]);

% Gradient of the function:

g = gradient(f, [x,y]);
```

P1. Part A

Implement gradient descent to minimize this function. Let the initial values be x0 = 0.1, y0 = 0.1, let the learning rate be n = 0.01, and let the number of iterations be 50. Give a plot of how the function value drops with the number of iterations performed. Repeat this problem for a learning rate of n = 0.1. What happened?

```
nIterations = 50;

nRate_a_1 = 0.01;
```

```
nRate_a_2 = 0.1;

x0_a = 0.1;
y0_a = 0.1;

values_a_1 = FindMinimum(x0_a, y0_a, nRate_a_1, nIterations, f, g);
values_a_2 = FindMinimum(x0_a, y0_a, nRate_a_2, nIterations, f, g);
```

P1. Part B

Obtain the "minimum" value and the location of the minimum you get for gradient descent using the same learning rate and number of iterations as in part (a), starting from the following initial points:

(x0, y0) = (1, 1) (x0, y0) = (-0.5, -0.5) (x0, y0) = (-1, -1)

```
nRate_b1 = nRate_a_1;
nRate_b2 = nRate_a_2;

x0_1b = 1;
y0_1b = 1;

x0_2b = -0.5;
y0_2b = -0.5;

x0_3b = -1;
y0_3b = -1;

values_1b_1 = FindMinimum(x0_1b, y0_1b, nRate_b1, nIterations, f, g);
values_1b_2 = FindMinimum(x0_1b, y0_1b, nRate_b2, nIterations, f, g);

values_2b_1 = FindMinimum(x0_2b, y0_2b, nRate_b1, nIterations, f, g);
values_2b_2 = FindMinimum(x0_2b, y0_2b, nRate_b2, nIterations, f, g);

values_3b_1 = FindMinimum(x0_3b, y0_3b, nRate_b1, nIterations, f, g);
values_3b_2 = FindMinimum(x0_3b, y0_3b, nRate_b2, nIterations, f, g);
```

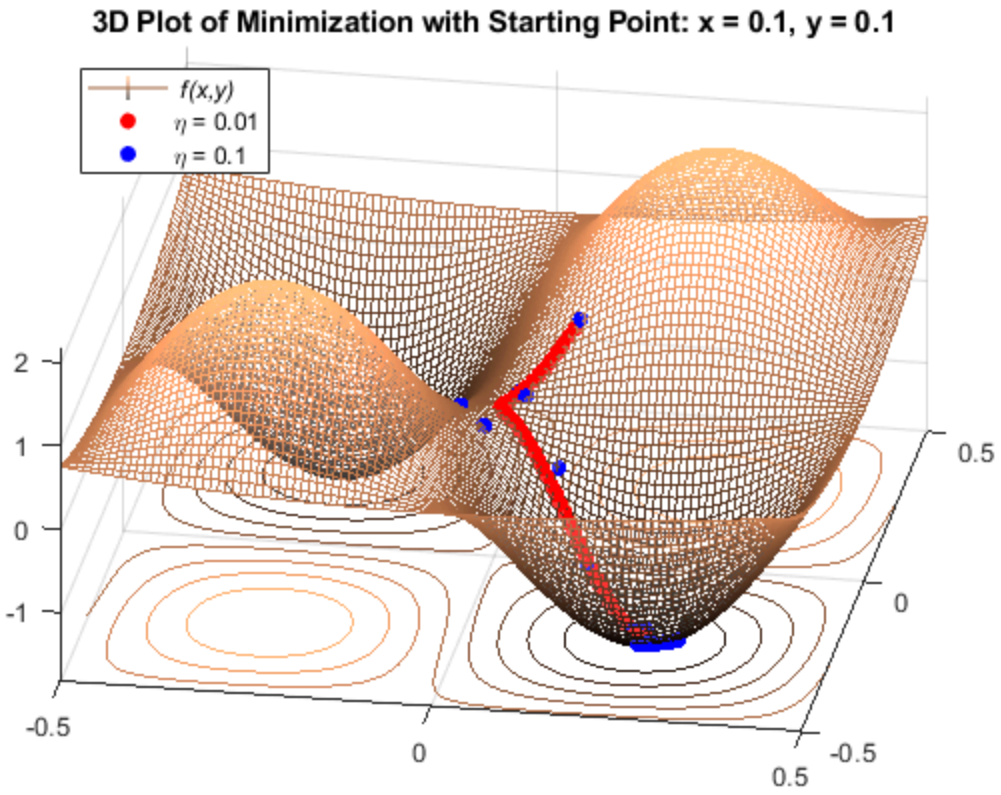
P1. Plotting

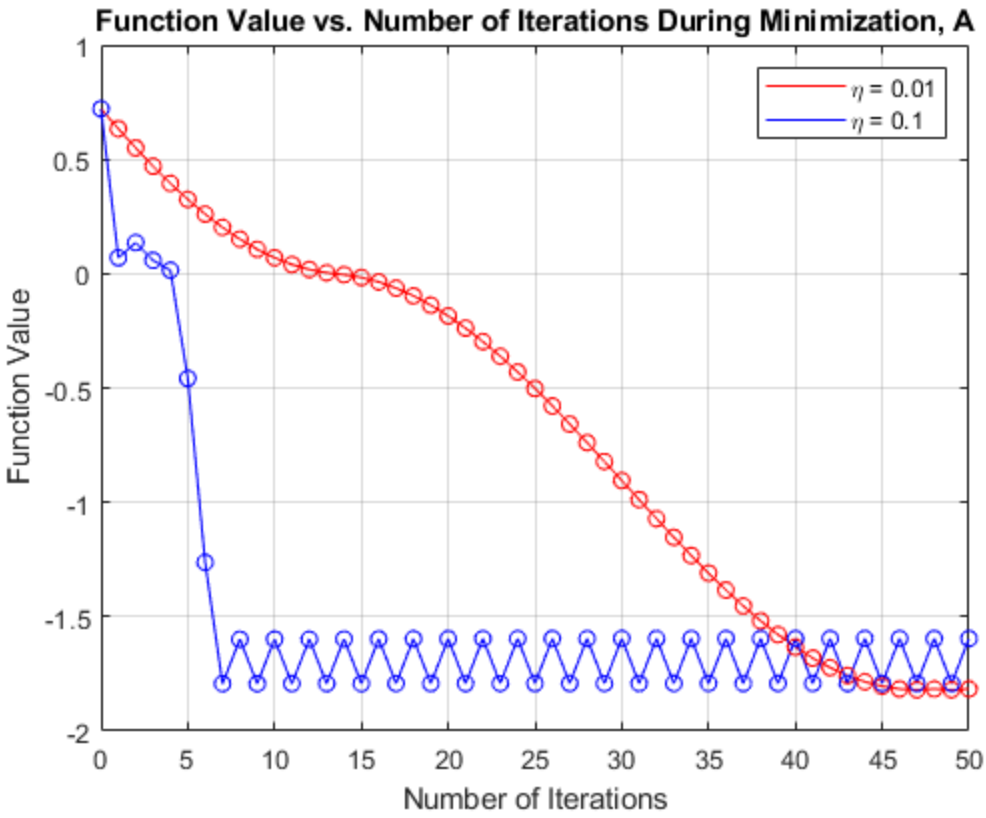
```
% Part A: Plot for Minimizing the Given Function with eta = 0.01, eta
 = 0.1
figure('Name', 'Mesh Plot of Given Function Minimization, Part A');
p1 = fmesh(f, [-0.5 0.5 -0.5 0.5], 'MeshDensity',
 100, 'ShowContours', 'on');
title('3D Plot of Minimization with Starting Point: x = 0.1, y = 0.1')
view([3.696804878048781e+02,42.324671057384734])
colormap(copper)
alpha(p1, 0.2)
grid on
hold on
scatter3(values_a_1(:,1),values_a_1(:,2),values_a_1(:,3), 'red', 'filled');
scatter3(values_a_2(:,1),values_a_2(:,2),values_a_2(:,3), 'blue', 'filled');
legend('\it f(x,y)', '\eta = 0.01', '\eta = 0.1', 'Location', 'nw');
```

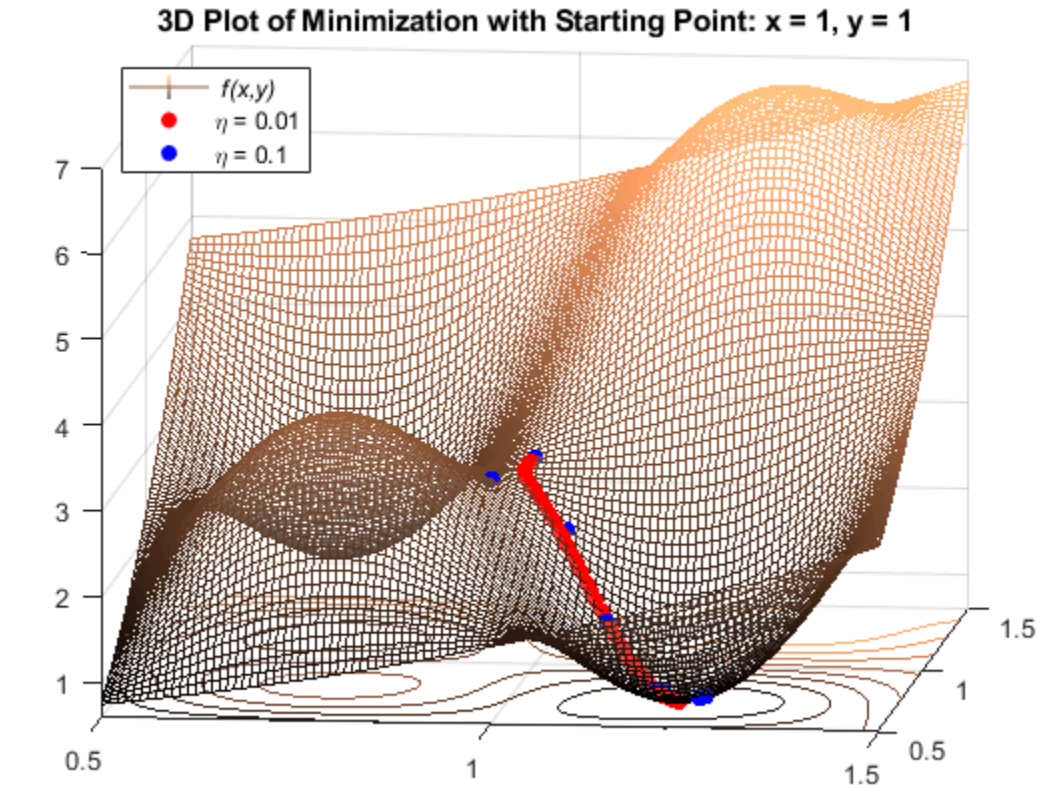
```
hold off

% Part A: Plot of Function Value vs. Number of Iterations
figure('Name', 'Plot of Function Value vs. Number of Iterations, Part
 A');
plot(values_a_1(:,4),values_a_1(:,3),'red');
hold on
plot(values_a_2(:,4),values_a_2(:,3),'blue');
scatter(values_a_1(:,4),values_a_1(:,3),'red');
scatter(values_a_2(:,4),values_a_2(:,3),'blue');
grid on
legend('\eta = 0.01', '\eta = 0.1', 'Location', 'ne');
title('Function Value vs. Number of Iterations During Minimization,
 A')
xlabel('Number of Iterations')
ylabel('Function Value')
hold off


% Part B.1: Plot for Minimizing Given Function with eta = 0.01, eta =
 0.1
figure('Name', 'Mesh Plot of Given Function Minimization, Part B.1');
p2 = fmesh(f, [0.5 1.5 0.5 1.5], 'MeshDensity',
 100, 'ShowContours', 'on');
title('3D Plot of Minimization with Starting Point: x = 1, y = 1')
view([6.615584415584404,12.657534246575342])
colormap(copper)
alpha(p2, 0.2)
grid on
hold on
scatter3(values_1b_1(:,1),values_1b_1(:,2),values_1b_1(:,3), 'red', 'filled');
scatter3(values_1b_2(:,1),values_1b_2(:,2),values_1b_2(:,3), 'blue', 'filled');
legend('\it f(x,y)', '\eta = 0.01', '\eta = 0.1', 'Location', 'nw');
hold off

% Part B.1: Plot of Function Value vs. Number of Iterations
figure('Name', 'Plot of Function Value vs. Number of Iterations, Part
 B.1');
plot(values_1b_1(:,4),values_1b_1(:,3),'red');
hold on
plot(values_1b_2(:,4),values_1b_2(:,3),'blue');
scatter(values_1b_1(:,4),values_1b_1(:,3),'red');
scatter(values_1b_2(:,4),values_1b_2(:,3),'blue');
grid on
legend('\eta = 0.01', '\eta = 0.1', 'Location', 'ne');
title('Function Value vs. Number of Iterations During Minimization,
 B.1')
xlabel('Number of Iterations')
ylabel('Function Value')
hold off


% Part B.2: Plot for Minimizing Given Function with eta = 0.01, eta =
 0.1
```

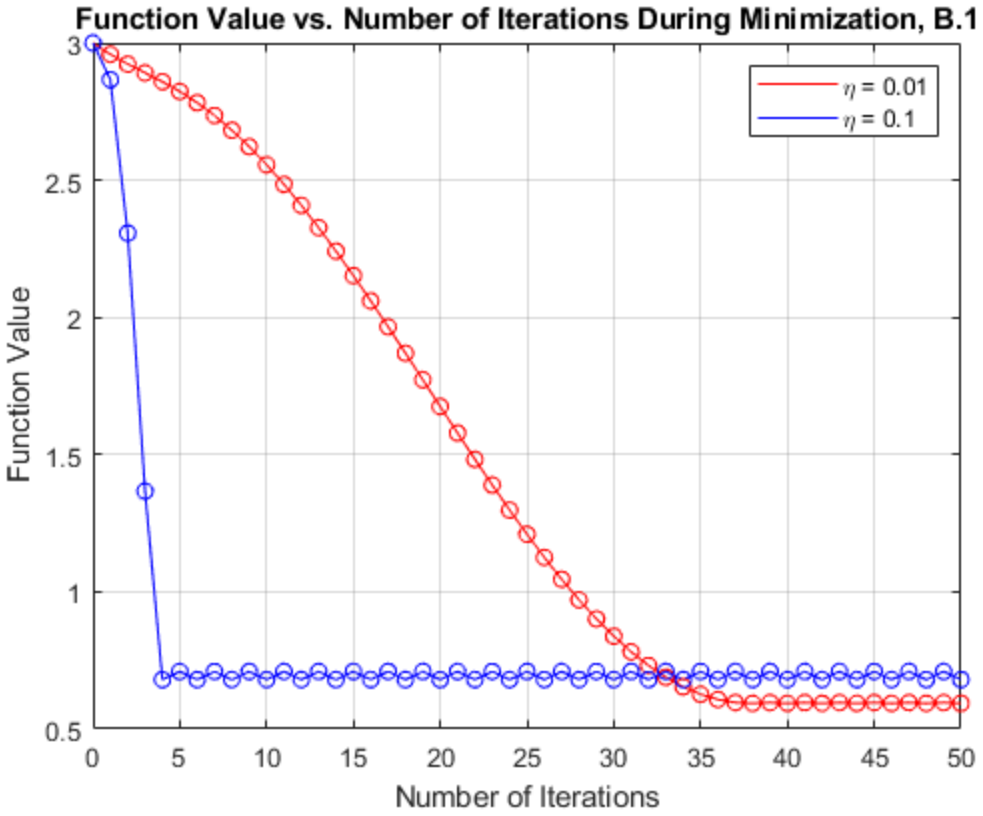
```
figure('Name', 'Mesh Plot of Given Function Minimization, Part B.2');
p3 = fmesh(f, [-1 0 -1 0], 'MeshDensity', 100, 'ShowContours', 'on');
title('3D Plot of Minimization with Starting Point: x = -0.5, y =
 -0.5')
view([-1.659287277701778e+02,29.719256757848694])
colormap(copper)
alpha(p3, 0.2)
grid on
hold on
scatter3(values_2b_1(:,1),values_2b_1(:,2),values_2b_1(:,3), 'red', 'filled');
scatter3(values_2b_2(:,1),values_2b_2(:,2),values_2b_2(:,3), 'blue', 'filled');
legend('\it f(x,y)', '\eta = 0.01', '\eta = 0.1', 'Location', 'nw');
hold off

% Part B.2: Plot of Function Value vs. Number of Iterations
figure('Name', 'Plot of Function Value vs. Number of Iterations, Part
 B.2');
plot(values_2b_1(:,4),values_2b_1(:,3),'red');
hold on
plot(values_2b_2(:,4),values_2b_2(:,3),'blue');
scatter(values_2b_1(:,4),values_2b_1(:,3),'red');
scatter(values_2b_2(:,4),values_2b_2(:,3),'blue');
grid on
legend('\eta = 0.01', '\eta = 0.1', 'Location', 'ne');
title('Function Value vs. Number of Iterations During Minimization,
 B.2')
xlabel('Number of Iterations')
ylabel('Function Value')
hold off


% Part B.3: Plot for Minimizing Given Function with eta = 0.01, eta =
 0.1
figure('Name', 'Mesh Plot of Given Function Minimization, Part B.3');
p4 = fmesh(f, [-1.5 -0.5 -1.5 -0.5], 'MeshDensity',
 100, 'ShowContours', 'on');
title('3D Plot of Minimization with Starting Point: x = -1, y = -1')
view([-5.321161957789208e+02,11.178053849070913])
colormap(copper)
alpha(p4, 0.2)
grid on
hold on
scatter3(values_3b_1(:,1),values_3b_1(:,2),values_3b_1(:,3), 'red', 'filled');
scatter3(values_3b_2(:,1),values_3b_2(:,2),values_3b_2(:,3), 'blue', 'filled');
legend('\it f(x,y)', '\eta = 0.01', '\eta = 0.1', 'Location', 'nw');
hold off

% Part B.3: Plot of Function Value vs. Number of Iterations
figure('Name', 'Plot of Function Value vs. Number of Iterations, Part
 B.3');
plot(values_3b_1(:,4),values_3b_1(:,3),'red');
hold on
plot(values_3b_2(:,4),values_3b_2(:,3),'blue');
scatter(values_3b_1(:,4),values_3b_1(:,3),'red');
```

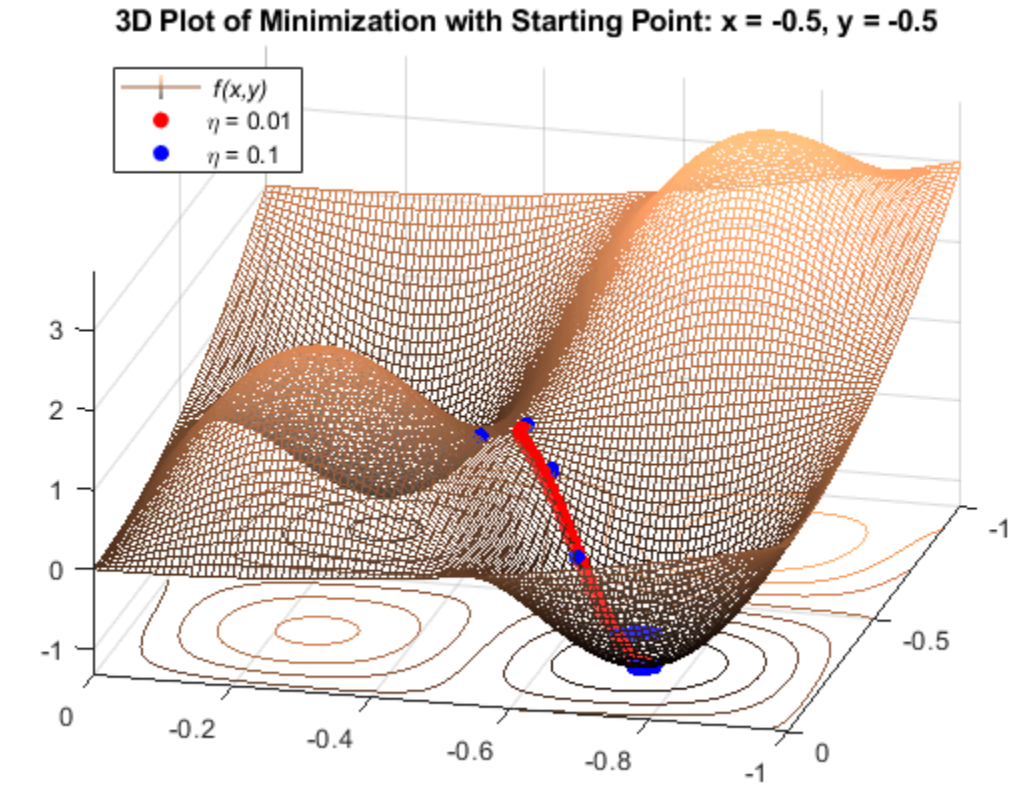
```
scatter(values_3b_2(:,4),values_3b_2(:,3),'blue');
grid on
legend('\eta = 0.01', '\eta = 0.1', 'Location', 'ne');
title('Function Value vs. Number of Iterations During Minimization,
 B.3')
xlabel('Number of Iterations')
ylabel('Function Value')
hold off
```

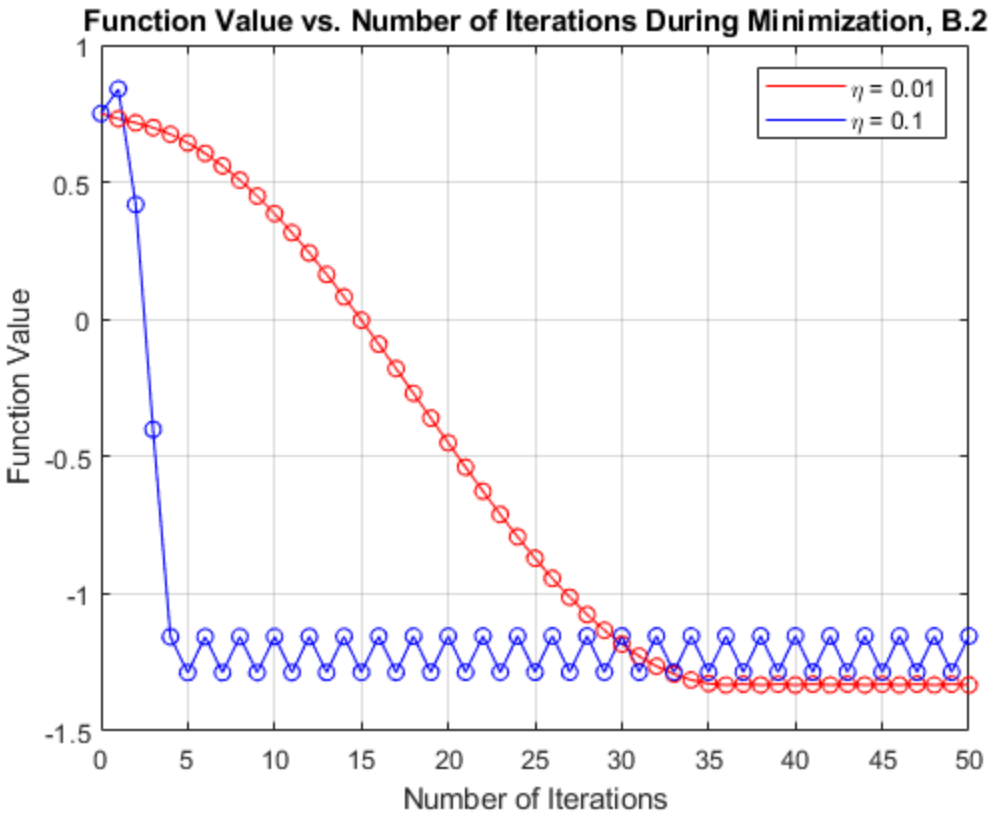


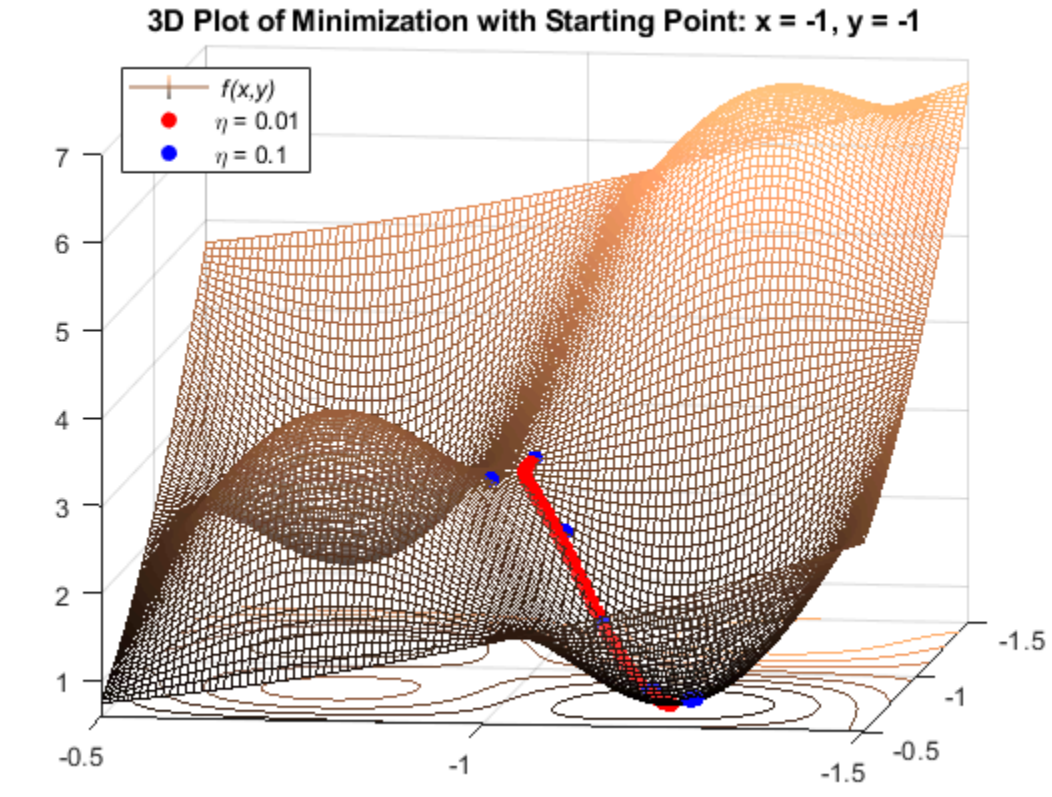

Function Value vs. Number of Iterations During Minimization, A
Function Value
Number of Iterations
η = 0.01
η = 0.1
1
0.5
0
-0.5
-1
-1.5
-2
0
5
10
15
20
25
30
35
40
45
50

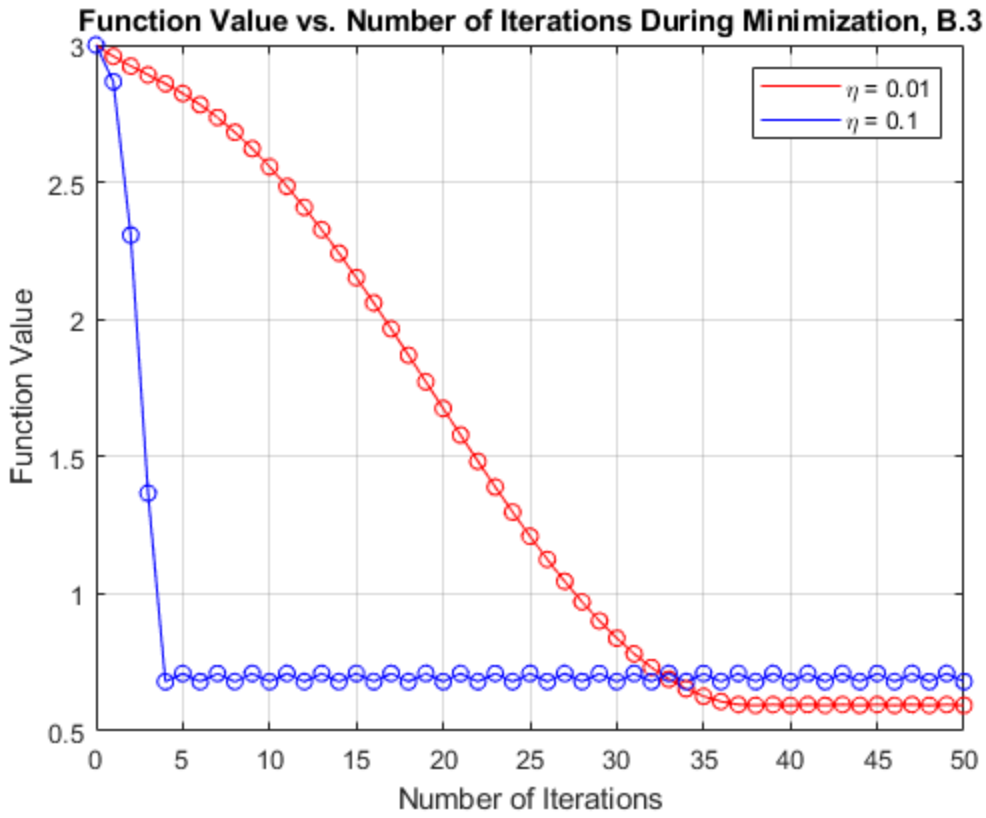

3D Plot of Minimization with Starting Point: x = 1, y = 1
f(x,y)
η = 0.01
η = 0.1
7
6
5
4
3
2
1
0.5
1
1.5
1
0.5
1.5


Function Value vs. Number of Iterations During Minimization, B.1
Function Value
Number of Iterations
η = 0.01
η = 0.1
3
2.5
2
1.5
1
0.5
0
5
10
15
20
25
30
35
40
45
50


3D Plot of Minimization with Starting Point: x = -0.5, y = -0.5
f(x,y)
η = 0.01
η = 0.1
3
2
1
0
-1
0
-0.2
-0.4
-0.6
-0.8
-1
0
-0.5
-1


Function Value vs. Number of Iterations During Minimization, B.2
Function Value
Number of Iterations
η = 0.01
η = 0.1
1
0.5
0
-0.5
-1
-1.5
0
5
10
15
20
25
30
35
40
45
50


3D Plot of Minimization with Starting Point: x = -1, y = -1
f(x,y)
η = 0.01
η = 0.1
7
6
5
4
3
2
1
-0.5
-1
-1.5
-1.5
-1
-0.5



P1. Table of Results

```
A = 'A';
B = 'B';

Part = [A;A;B;B;B;B;B;B];

Eta = [nRate_a_1;
       nRate_a_2;
       nRate_b1;
       nRate_b2;
       nRate_b1;
       nRate_b2;
       nRate_b1;
       nRate_b2];

xo = [x0_a;x0_a;x0_1b;x0_1b;x0_2b;x0_2b;x0_3b;x0_3b];

yo = [y0_a;y0_a;y0_1b;y0_1b;y0_2b;y0_2b;y0_3b;y0_3b];

Min_X = round([values_a_1(51,1);
               values_a_2(51,1);
               values_1b_1(51,1);
               values_1b_2(51,1);
               values_2b_1(51,1);
```

```
                values_2b_2(51,1);
                values_3b_1(51,1);
                values_3b_2(51,1)], 2);

Min_Y = round([values_a_1(51,2);
                values_a_2(51,2);
                values_1b_1(51,2);
                values_1b_2(51,2);
                values_2b_1(51,2);
                values_2b_2(51,2);
                values_3b_1(51,2);
                values_3b_2(51,2)], 2);

Min_Z = round([values_a_1(51,3);
                values_a_2(51,3);
                values_1b_1(51,3);
                values_1b_2(51,3);
                values_2b_1(51,3);
                values_2b_2(51,3);
                values_3b_1(51,3);
                values_3b_2(51,3)], 2);

T = table(Part, Eta, xo, yo, Min_X, Min_Y, Min_Z);

fprintf('\n')
fprintf('Final Values Table:\n')
fprintf('\n')
disp(T)
fprintf('\n')
```

Final Values Table:

Part	Eta	xo	yo	Min_X	Min_Y	Min_Z
A	0.01	0.1	0.1	0.24	-0.23	-1.82
A	0.1	0.1	0.1	0.25	-0.31	-1.6
B	0.01	1	1	1.22	0.71	0.59
B	0.1	1	1	1.19	0.75	0.68
B	0.01	-0.5	-0.5	-0.73	-0.24	-1.33
B	0.1	-0.5	-0.5	-0.73	-0.3	-1.15
B	0.01	-1	-1	-1.22	-0.71	0.59
B	0.1	-1	-1	-1.19	-0.75	0.68

P1. Concluding Remarks

We can see from the final values table printed above that the minimum value is always smaller when the learning rate is smaller. This is because a smaller learning rate results in smaller adjustments to the weights on each iteration, so we can accurately reach the local minima with a small learning rate. However, the number of iterations required to get near the minimum is much larger. For large learning rates, the number of iterations requried to get near the minimum is very small, but we wind up oscillating back and forth at values near the minimum and never reach the actual minimum. In general, it is important to weigh

the tradeoffs between number of iterations and minimum value obtained when using gradient descent to minimize the error function.

Problem 2: Neural Network Training with GD, BP

You are given to train the network to match the output. You must show the intermediate steps, i.e. the intermediate weight matrices as you go through different iterations. Notice that you should solve the problem for 3 different learning rates and make your observations. As before, you should write the code from scratch and not use the library functions available in ML tools.

```
syms x
f = symfun(1/(1+(exp(-1*x))), x);

df = gradient(f, x);
```

P2. Initializing Eta Values and Iterations

```
% Learning rates

eta1 = 0.05;
eta2 = 0.50;
eta3 = 1.00;

eta = [eta1;eta2;eta3];
[nRowsEta,nColsEta] = size(eta);

% Set necessary number of iterations for each learning rate

nIterations_eta1 = 101;
nIterations_eta2 = 16;
nIterations_eta3 = 16;

iterations = [nIterations_eta1;
              nIterations_eta2;
              nIterations_eta3];

% Initialize matrices to hold values of outputs, iterations, errors

values_eta1 = zeros(nIterations_eta1,5);
values_eta2 = zeros(nIterations_eta2,5);
values_eta3 = zeros(nIterations_eta3,5);
```

P2. Neural Network Training

```
for j = 1:nRowsEta

    % given weights

    w01_0 = 0.35;
```

```
    w02_0 = 0.35;
    w11_0 = 0.15;
    w12_0 = 0.25;
    w21_0 = 0.20;
    w22_0 = 0.30;

    w01_1 = 0.60;
    w02_1 = 0.60;
    w11_1 = 0.40;
    w12_1 = 0.50;
    w21_1 = 0.45;
    w22_1 = 0.55;

    winit = [w01_0 w02_0 w01_1 w02_1;
             w11_0 w12_0 w11_1 w12_1;
             w21_0 w22_0 w21_1 w22_1];

    w = winit;


    %*** Forward Propagation ***

    % ******Layer 1 Calculations******

    x1_0 = 0.05;
    x2_0 = 0.10;

    y1 = 0.01;
    y2 = 0.99;

    Y = [y1; y2];

    % Store initial input data as a 3x1 matrix
    X0 = [1; x1_0; x2_0];


    for i = 1:iterations(j)

        w1 = w(:,[1,2]);
        w2 = w(:,[3,4]);

        w1_2 = [w(2,3);w(3,3)];
        w2_2 = [w(2,4);w(3,4)];

        % Compute sum signal for neurons in layer L = 1
        s1_1 = w(1,1)*X0(1) + w(2,1)*X0(2) + w(3,1)*X0(3);
        s2_1 = w(1,2)*X0(1) + w(2,2)*X0(2) + w(3,2)*X0(3);

        % Pass sum signals for neurons in layer L = 1 through
activation function
        x1_1 = double(f(s1_1));
        x2_1 = double(f(s2_1));

        % Store L = 1 outputs (L = 2 inputs) as a 3x1 matrix
```

```
        X1 = [1; x1_1; x2_1];

        % ******Layer 2 Calculations******

        s1_2 = w(1,3)*X1(1) + w(2,3)*X1(2) + w(3,3)*X1(3);
        s2_2 = w(1,4)*X1(1) + w(2,4)*X1(2) + w(3,4)*X1(3);

        x1_2 = double(f(s1_2));
        x2_2 = double(f(s2_2));

        X2 = [x1_2; x2_2];


        %*** Back Propagation ***

        % Compute the sensitivities in the output layer
        d_2 = double([(2*(X2(1)-Y(1))*df(s1_2));(2*(X2(2)-
Y(2))*df(s2_2))]);
        d1_2 = d_2(1);
        d2_2 = d_2(2);

        % Compute the sensitivities in the hidden layer
        d_1 = double((d1_2*df(s1_1)*w1_2)+(d2_2*df(s2_1)*w2_2));
        d1_1 = d_1(1);
        d2_1 = d_1(2);

        % Compute the gradient for the weights from layer 1 to layer 2
        del_e_2 = X1*transpose(d_2);

        % Compute the gradient for the weights from layer 0 to layer 1
        del_e_1 = X0*transpose(d_1);

        % Update the weights
        w2 = w2 - (eta(j)*del_e_2/(norm(del_e_2)));
        w1 = w1 - (eta(j)*del_e_1/(norm(del_e_1)));

        w = [w1,w2];


        if j == 1
            values_eta1(i,1) = i-1;
            values_eta1(i,2) = X2(1);
            values_eta1(i,3) = X2(2);
            values_eta1(i,4) = (Y(1) - X2(1))^2;
            values_eta1(i,5) = (Y(2) - X2(2))^2;
            if i == (iterations(j)+1)/2
                wintermediate1 = w;
            elseif i == iterations(j)
                wfinal1 = w;
            end

        elseif j == 2
            values_eta2(i,1) = i-1;
            values_eta2(i,2) = X2(1);
```

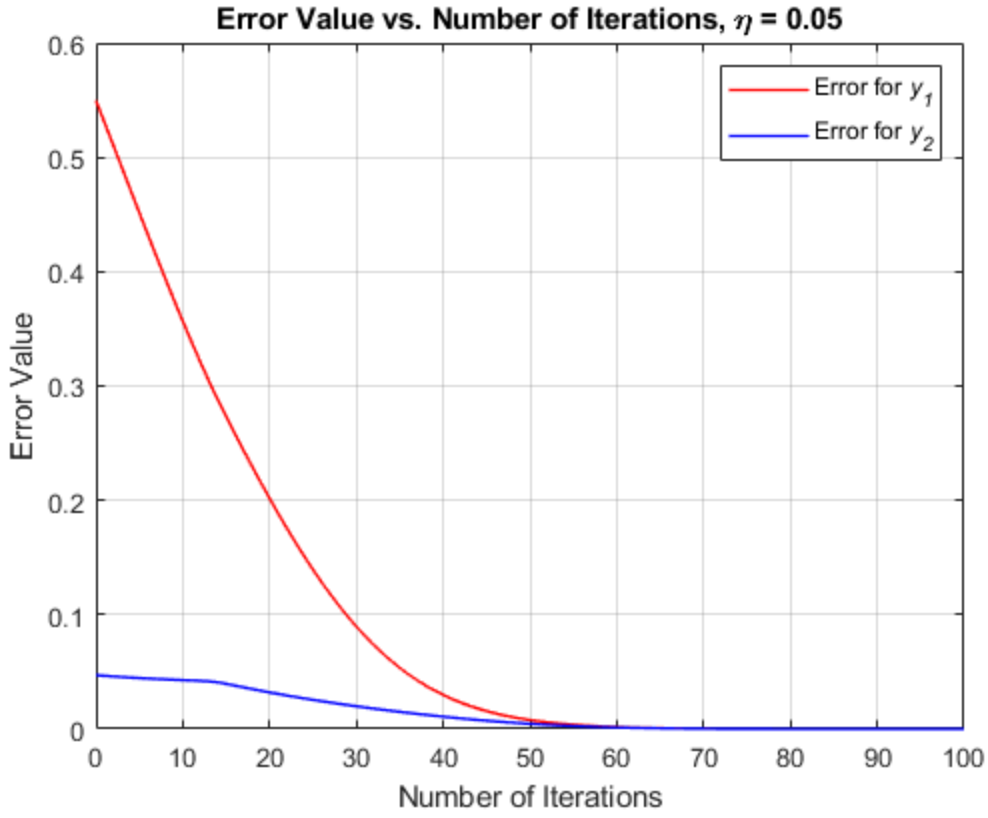
```
            values_eta2(i,3) = X2(2);
            values_eta2(i,4) = (Y(1) - X2(1))^2;
            values_eta2(i,5) = (Y(2) - X2(2))^2;
            if i == iterations(j)/2
                wintermediate2 = w;
            elseif i == iterations(j)
                wfinal2 = w;
            end

        elseif j == 3
            values_eta3(i,1) = i-1;
            values_eta3(i,2) = X2(1);
            values_eta3(i,3) = X2(2);
            values_eta3(i,4) = (Y(1) - X2(1))^2;
            values_eta3(i,5) = (Y(2) - X2(2))^2;
            if i == iterations(j)/2
                wintermediate3 = w;
            elseif i == iterations(j)
                wfinal3 = w;
            end
        end
    end

end
```

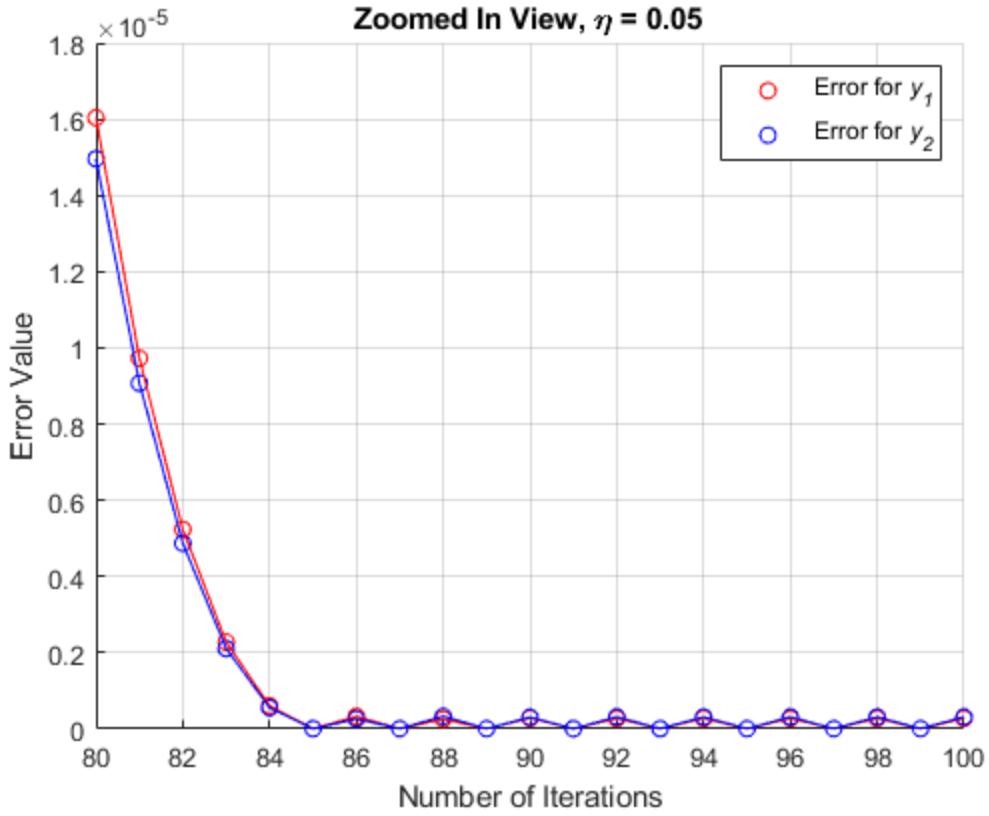
P2. Plotting

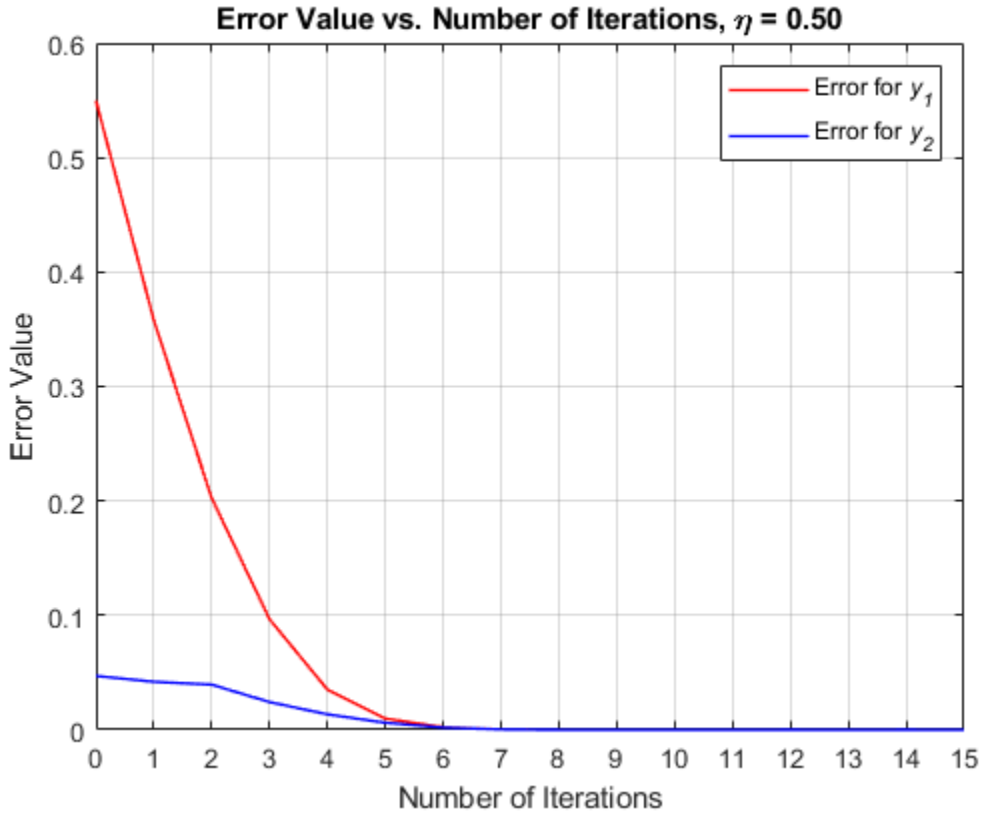
```
% Plotting error value vs. number of iterations for eta = 0.05
figure('Name', 'Error Value Decreasing with Number of Iterations,
 p.1')
plot(values_eta1(:,1),values_eta1(:,4),'red', 'LineWidth', 1)
title('Error Value vs. Number of Iterations, \eta = 0.05')
hold on
plot(values_eta1(:,1),values_eta1(:,5),'blue', 'LineWidth', 1)
xlabel('Number of Iterations')
ylabel('Error Value')
legend('Error for \ity_{1}', 'Error for \ity_{2}')
grid on
hold off

% Plotting instability with eta = 0.05
figure('Name', 'Error Value Instability, p.1')
scatter(values_eta1(81:101,1),values_eta1(81:101,4),'red')
title('Zoomed In View, \eta = 0.05')
hold on
scatter(values_eta1(81:101,1),values_eta1(81:101,5),'blue')
plot(values_eta1(81:101,1),values_eta1(81:101,4),'red')
plot(values_eta1(81:101,1),values_eta1(81:101,5),'blue')
xlabel('Number of Iterations')
ylabel('Error Value')
legend('Error for \ity_{1}', 'Error for \ity_{2}')
grid on
hold off
```

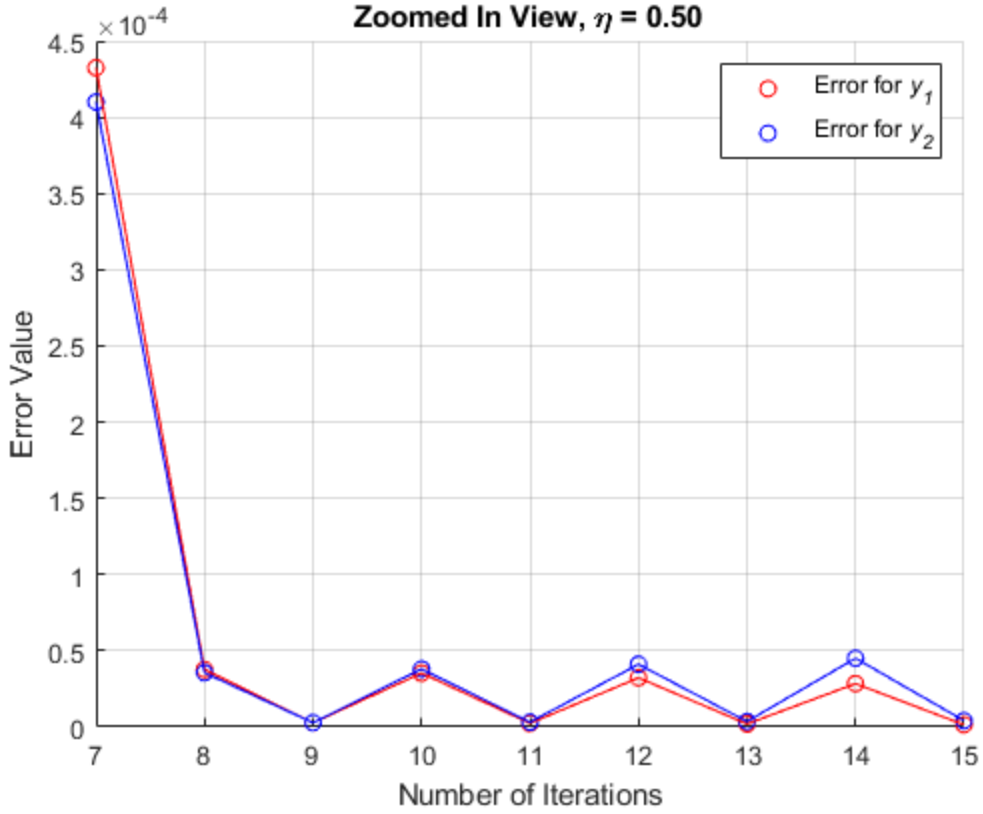
```
% Plotting error value vs. number of iterations for eta = 0.50
figure('Name', 'Error Value Decreasing with Number of Iterations,
 p.2')
plot(values_eta2(:,1),values_eta2(:,4),'red', 'LineWidth', 1)
title('Error Value vs. Number of Iterations, \eta = 0.50')
hold on
plot(values_eta2(:,1),values_eta2(:,5),'blue', 'LineWidth', 1)
xlabel('Number of Iterations')
ylabel('Error Value')
xticks(0:1:15)
legend('Error for \ity_{1}', 'Error for \ity_{2}')
grid on
hold off

% Plotting instability with eta = 0.50
figure('Name', 'Error Value Instability, p.2')
scatter(values_eta2(8:16,1),values_eta2(8:16,4),'red')
title('Zoomed In View, \eta = 0.50')
hold on
scatter(values_eta2(8:16,1),values_eta2(8:16,5),'blue')
plot(values_eta2(8:16,1),values_eta2(8:16,4),'red')
plot(values_eta2(8:16,1),values_eta2(8:16,5),'blue')
xlabel('Number of Iterations')
ylabel('Error Value')
legend('Error for \ity_{1}', 'Error for \ity_{2}')
grid on
hold off

% Plotting error value vs. number of iterations for eta = 1.00
figure('Name', 'Error Value Decreasing with Number of Iterations,
 p.3')
plot(values_eta3(:,1),values_eta3(:,4),'red', 'LineWidth', 1)
title('Error Value vs. Number of Iterations, \eta = 1.00')
hold on
plot(values_eta3(:,1),values_eta3(:,5),'blue', 'LineWidth', 1)
xlabel('Number of Iterations')
ylabel('Error Value')
xticks(0:1:15)
legend('Error for \ity_{1}', 'Error for \ity_{2}')
grid on
hold off

% Plotting instability with eta = 1.00
figure('Name', 'Error Value Instability, p.3')
scatter(values_eta3(4:16,1),values_eta3(4:16,4),'red')
title('Zoomed In View, \eta = 1.00')
hold on
scatter(values_eta3(4:16,1),values_eta3(4:16,5),'blue')
plot(values_eta3(4:16,1),values_eta3(4:16,4),'red')
plot(values_eta3(4:16,1),values_eta3(4:16,5),'blue')
xticks(3:1:15)
xlim([3 15])
xlabel('Number of Iterations')
```

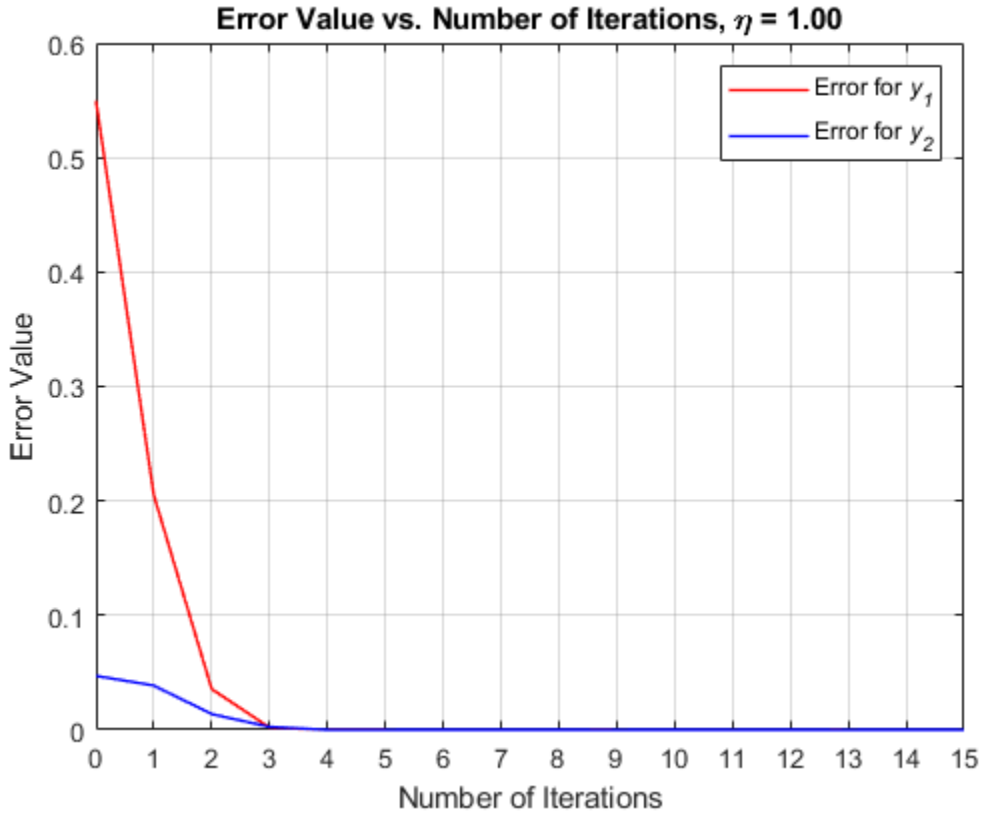
```
ylabel('Error Value')
legend('Error for \ity_{1}', 'Error for \ity_{2}')
grid on
hold off
```

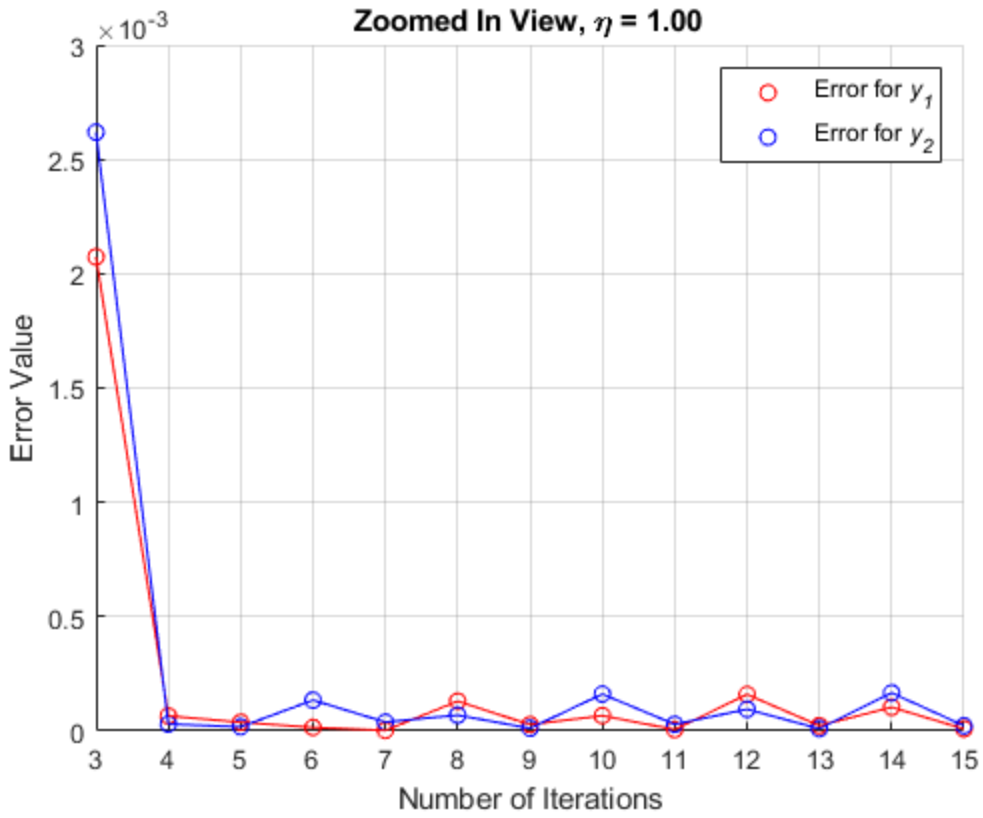



Zoomed In View, η = 0.05
$\times 10^{-5}$
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
Error Value
80
82
84
86
88
90
92
94
96
98
100
Number of Iterations
Error for y_1
Error for y_2


Error Value vs. Number of Iterations, η = 0.50
0.6
0.5
0.4
0.3
0.2
0.1
0
Error Value
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Number of Iterations
Error for y_1
Error for y_2


Zoomed In View, η = 0.50
$\times 10^{-4}$
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
Error Value
7
8
9
10
11
12
13
14
15
Number of Iterations
Error for y_1
Error for y_2


Error Value vs. Number of Iterations, η = 1.00
0.6
0.5
0.4
0.3
0.2
0.1
0
Error Value
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Number of Iterations
Error for y_1
Error for y_2



P2. Table of Results, Outputting Weights, Final Values

```
Part = ['Part 1';
        'Part 2';
        'Part 3'];

Eta = eta;

Iterations = iterations;


y1_Final = round([values_eta1(nIterations_eta1,2);
                  values_eta2(nIterations_eta2,2);
                  values_eta3(nIterations_eta3,2)], 2);

y2_Final = round([values_eta1(nIterations_eta1,3);
                  values_eta2(nIterations_eta2,3);
                  values_eta3(nIterations_eta3,3)], 2);


T = table(Part, Eta, Iterations, y1_Final, y2_Final);

fprintf('\n')
```

```
fprintf('Final Values Table:\n')
fprintf('\n')
disp(T)
fprintf('\n')

fprintf('\n')
fprintf('Initial Weights for Part 1,2,3:\n')
disp(winit)
fprintf('_____________________________________________\n');

fprintf('\n')
fprintf('Intermediate Weights for Part 1, Iteration 51:\n')
disp(wintermediate1)

fprintf('\n')
fprintf('Final Weights for Part 1:\n')
disp(wfinal1)
fprintf('_____________________________________________\n');

fprintf('\n')
fprintf('Intermediate Weights for Part 2, Iteration 8:\n')
disp(wintermediate2)

fprintf('\n')
fprintf('Final Weights for Part 2:\n')
disp(wfinal2)
fprintf('_____________________________________________\n');

fprintf('\n')
fprintf('Intermediate Weights for Part 3, Iteration 8:\n')
disp(wintermediate3)

fprintf('\n')
fprintf('Final Weights for Part 3:\n')
disp(wfinal3)
fprintf('\n')
```

Final Values Table:

Part	Eta	Iterations	y1_Final	y2_Final
Part 1	0.05	101	0.01	0.99
Part 2	0.5	16	0.01	0.99
Part 3	1	16	0.01	0.99

Initial Weights for Part 1,2,3:

0.3500	0.3500	0.6000	0.6000
0.1500	0.2500	0.4000	0.5000
0.2000	0.3000	0.4500	0.5500

```
Intermediate Weights for Part 1, Iteration 51:
    1.5001    0.8278   -1.2457    1.1792
    0.2075    0.2739   -0.7732    0.8840
    0.3150    0.3478   -0.5698    0.8812

Final Weights for Part 1:
    2.7607    1.9526   -2.0753    1.8829
    0.2705    0.3301   -1.5105    1.5124
    0.4411    0.4603   -1.2306    1.4460
```

```
Intermediate Weights for Part 2, Iteration 8:
    2.0179    1.5122   -2.1072    1.8449
    0.2334    0.3081   -1.2632    1.3424
    0.3668    0.4162   -1.0650    1.3108

Final Weights for Part 2:
    2.0212    1.5087   -2.1621    1.8419
    0.2336    0.3079   -1.2868    1.3093
    0.3671    0.4159   -1.0762    1.2669
```

```
Intermediate Weights for Part 3, Iteration 8:
    1.8871    1.6210   -2.0635    1.8529
    0.2269    0.3135   -1.0847    1.2414
    0.3537    0.4271   -0.9624    1.2385

Final Weights for Part 3:
    1.8970    1.6104   -2.1190    1.8772
    0.2274    0.3130   -1.0313    1.1557
    0.3547    0.4260   -0.8893    1.1325
```

P2. Concluding Remarks

The neural network of this problem consists of an input layer (L=0), 1 hidden layer (L=1), and the output layer (L=2). The input layer and the hidden layer each have two dimensions plus one bias neuron. The training data is passed into the network and the network outputs some values for y1 and y2 which are not correct at first because the initial weights are random. Back propagation and gradient descent are used to minimize the error function and finalize the weights for three different learning rates. Since each learning rate is different, training the network requires a different number of iterations for each learning rate. In general, if the learning rate is small, more iterations are required and the error function is reasonably stable after a large number of iterations. If the learning rate is large, less iterations are required, but the error function is unstable and a true minimum value for the error function is never reached. In the case of this problem, the best results are obtained when the learning rate is 0.05, but that learning rate requires 100 iterations to reach the minimum. The fast results are obtained with a learning rate of 1.0, but the error function is unstable, and we do not reach the true minimum.

Functions

```
function values = FindMinimum(x0, y0, nRate, nIterations, f, g)

    w = [x0;y0];
```

```
    values = zeros(nIterations+1,4);

    values(1,1) = x0;
    values(1,2) = y0;
    values(1,3) = f(x0,y0);
    values(1,4) = 0;

    for i = 1:nIterations

    w = w - double((nRate*g(x0,y0)/norm(g(x0,y0))));
    x0 = w(1);
    y0 = w(2);
    values(i+1,1) = x0;
    values(i+1,2) = y0;
    values(i+1,3) = f(x0,y0);
    values(i+1,4) = i;

    end

end
```

Published with MATLAB® R2018b