BITS F312

Neural Networks and Fuzzy Logic

Assignment - 1

Name and ID :

Ravi Bharadwaj C

2016AAPS0244H

Question 1 :

```
clear;
clc;
close all;

iter = 1000;
alpha = 0.01;
J = 0;
X = xlsread('data.xlsx', 'Sheet1');
[m, n] = size(X);
%theta = randn(3,1);
theta = zeros(3,1);
temp_x = X(:,[1,2]);
t2_list = zeros(m,1);
t3_list = zeros(m,1);

x = [ones(m,1), temp_x];
x1 = X(:,1);
x2 = X(:,2);
y = X(:,3);

% Feature Scaling for x1 :
u1 = mean(x1);
s1 = std(x1);
x1 = (x1-u1)/s1;

% Feature Scaling for x2 :
u2 = mean(x2);
s2 = std(x2);
x2 = (x2-u2)/s2;

alpha_new = alpha/m;
c = 0.5/m;
J2 = zeros(500,1);
weights_list = zeros(iter,3);

for k = 1 : iter
    J = 0;
    % Computing the weight values :
    for i = 1:m
        t1 = theta(1) - (alpha_new * ((x(i,:)*theta)-y(i)));
        t2 = theta(2) - (alpha_new * ((x(i,:)*theta)-y(i)))*x1(i);
        t3 = theta(3) - (alpha_new * ((x(i,:)*theta)-y(i)))*x2(i);
        theta(1) = t1;
        theta(2) = t2;
        theta(3) = t3;
        t2_list(i) = t2;
        t3_list(i) = t3;
    end
    weights_list(k,:) = theta;
    % Computing the cost function for every weight value using Batch GD:
    for j = 1:m
        J = J + (c*((x(j,:)*theta)-y(j))^2);
    end
```

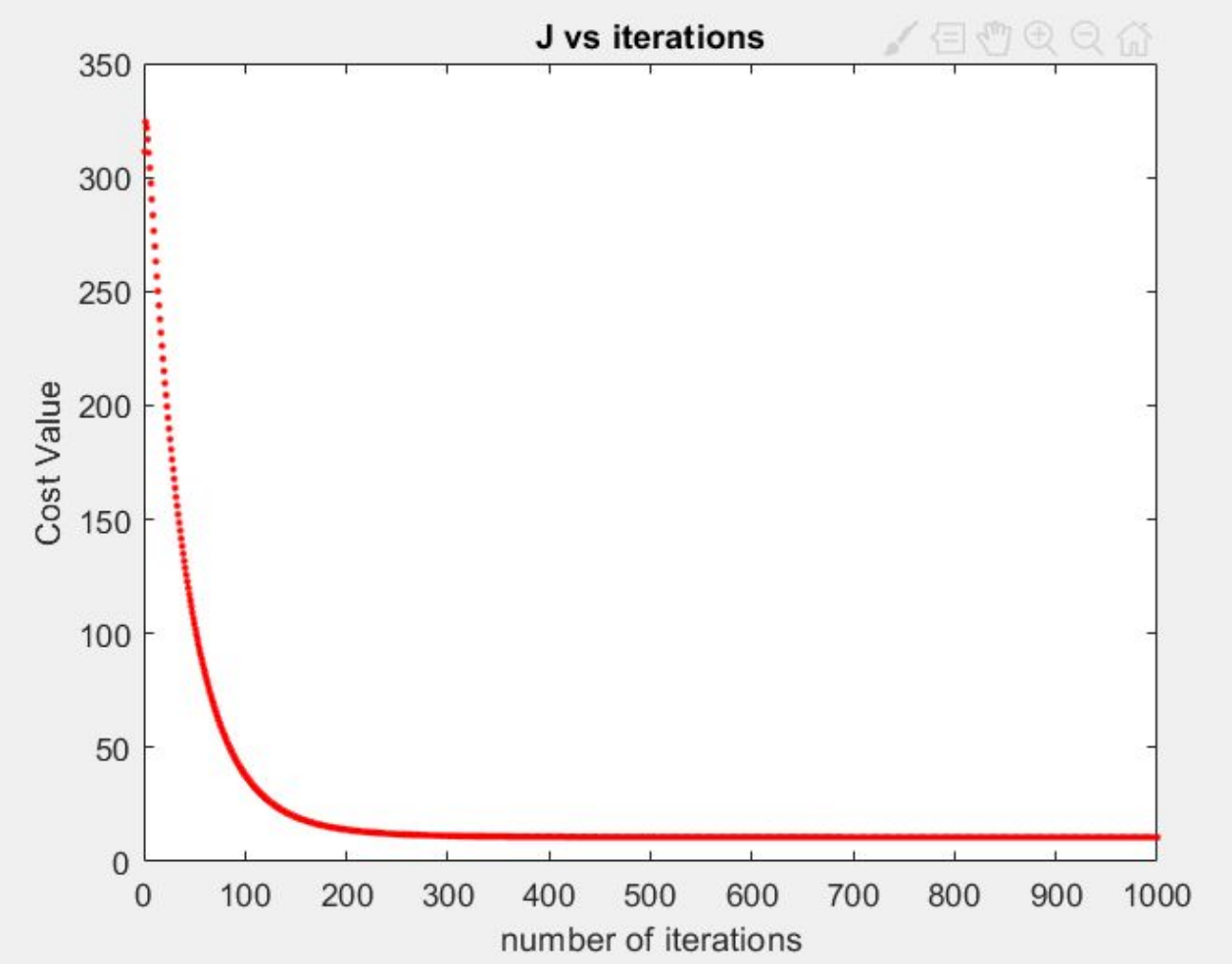
```
54        % Storing every value of Cost for every weight vector for plotting
55
56        J2(k) = J;
57
58    end
59    fprintf("weights = \n")
60    disp(theta);
61    % Plotting J for every iteration :
62    figure(1);
63    plot(J2, 'r.');
64    title('J vs iterations');
65    xlabel('number of iterations');
66    ylabel('Cost Value');
67
68    % Plotting J vs [w1, w2] :
69    figure(2);
70
71    % temp_theta = [ones(m,1), t2_list t3_list];
72    % J3 = temp_theta * x';
73    % s = [t2_list t3_list];
74    % contour3(J3);
75
```

weights =

8.2457

0.0096

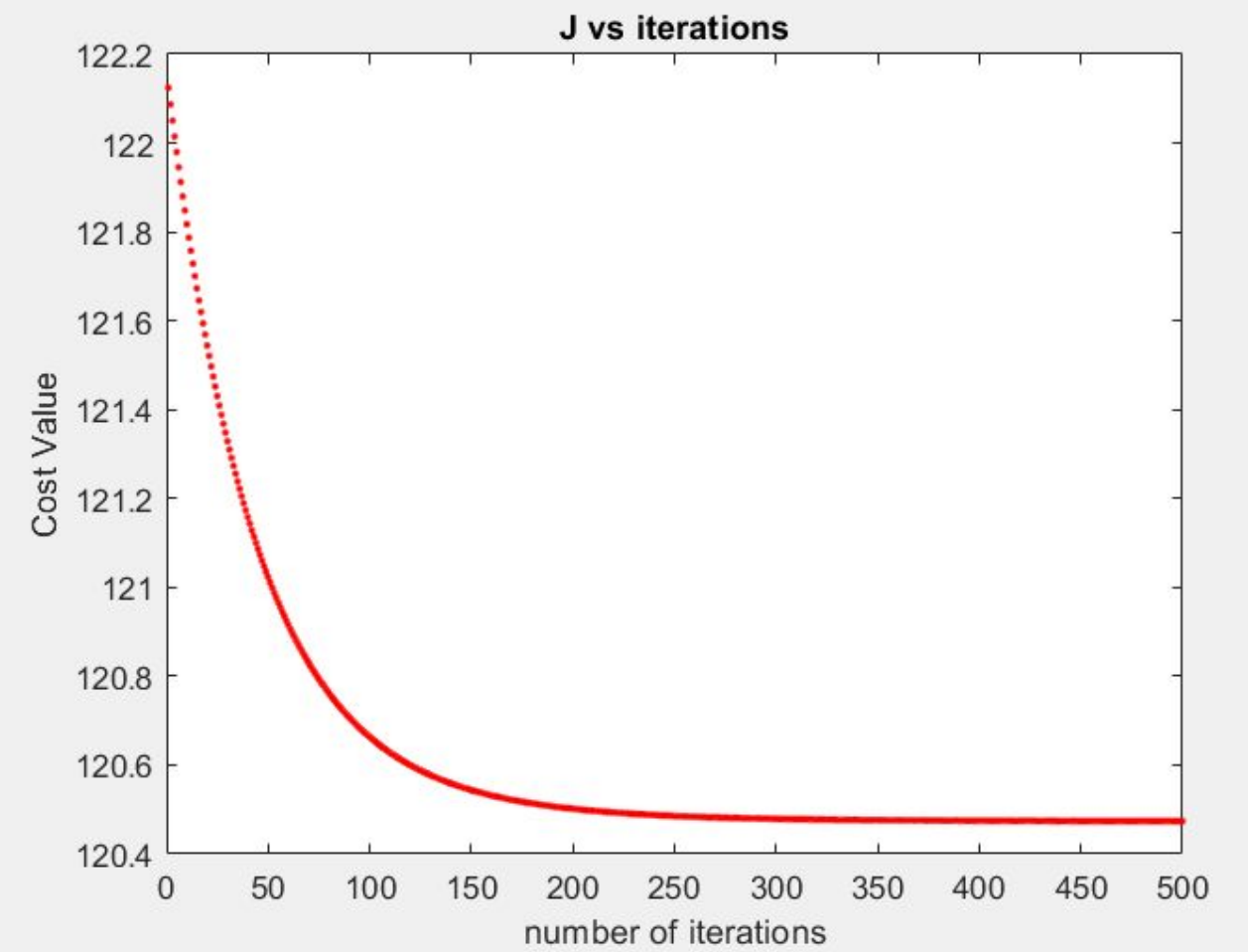
0.0041


J vs iterations
Cost Value
number of iterations
350
300
250
200
150
100
50
0
0
100
200
300
400
500
600
700
800
900
1000

Question 2 :

```
1
2   alpha = 0.01;
3   J = 0;
4   X = xlsread('data.xlsx', 'Sheet1');
5   [m, n] = size(X);
6   %theta = randn(3,1);
7   theta = zeros(3,1);
8   temp_x = X(:,[1,2]);
9   t2_list = zeros(m,1);
10  t3_list = zeros(m,1);
11
12  x1 = X(:,1);
13  x2 = X(:,2);
14  y = X(:,3);
15
16  % Feature Scaling for x1 :
17  u1 = mean(x1);
18  s1 = std(x1);
19  x1 = (x1-u1)/s1;
20
21  % Feature Scaling for x2 :
22  u2 = mean(x2);
23  s2 = std(x2);
24  x2 = (x2-u2)/s2;
25
26  x = [zeros(m,1) x1 x2];
27  k1 = 500;
28  alpha_new = alpha/m;
29  c = 0.5/m;
30  J2 = zeros(k1, 1);
31  J1 = zeros(m,1);
32
33  for k = 1 : k1,
34      J = 0;
35
36      % Computing the weight values :
37      for i = 1:m,
38          t1 = theta(1) - (alpha_new * ((x(i,:)*theta)-y(i)));
39          t2 = theta(2) - (alpha_new * ((x(i,:)*theta)-y(i)))*x1(i);
40          t3 = theta(3) - (alpha_new * ((x(i,:)*theta)-y(i)))*x2(i);
41          theta(1) = t1;
42          theta(2) = t2;
43          theta(3) = t3;
44          t2_list(i) = t2;
45          t3_list(i) = t3;
46
47          J = J + (c*((x(i,:)*theta)-y(i))^2);
48  %         J2(i,k) = J;
49      end;
```

```
50
51  %     % Computing the cost function for every weight value using Stochastic GD:
52  %     for j = 1:m,
53  %         J = J + (c*((x(j,:)*theta)-y(j))^2);
54  %     end;
55
56      % Storing every value of Cost for every weight vector for plotting
57      J2(k) = J;
58
59  end;
60  weights = theta;
61  % Plotting J for every iteration :
62  figure(1);
63  % J2 = J2(:);
64  plot(J2, 'r.');
65  title('J vs iterations');
66  xlabel('number of iterations');
67  ylabel('Cost Value');
68
69  % Plotting J vs [w1, w2] :
70  figure(2);
71  temp_theta = [ones(m,1), t2_list t3_list];
72  J3 = temp_theta * x';
73  s = [t2_list t3_list];
74  contour3(J3);
75
76
```



weights =

74.5265

0.3810

1.6703

Question 4 :

```
X = xlsread('data.xlsx', 'Sheet1');
[m, n] = size(X);
t2_list = zeros(m,1);
t3_list = zeros(m,1);

x2 = X(:,1);
x3 = X(:,2);
y = X(:,3);
x = [ones(m,1) x2 x3];
k = 500;
j = zeros(k,1);

for i = 1 : k
    theta = pinv(x' * x)* x' * y;
    % cost = 0.5 * (y'*y - (y' * theta * x)- (x' * theta' * y) + (theta' * x' * x * theta));
    temp_theta = theta';

    f = y - (temp_theta*x')';

    cost = 0.5 * (f' *f);
end
sprintf('the weight values in [w0, w1, w1] order are :')
sprintf('%.4f  ', theta)
sprintf('the cost value is : %0.2f', cost)

```

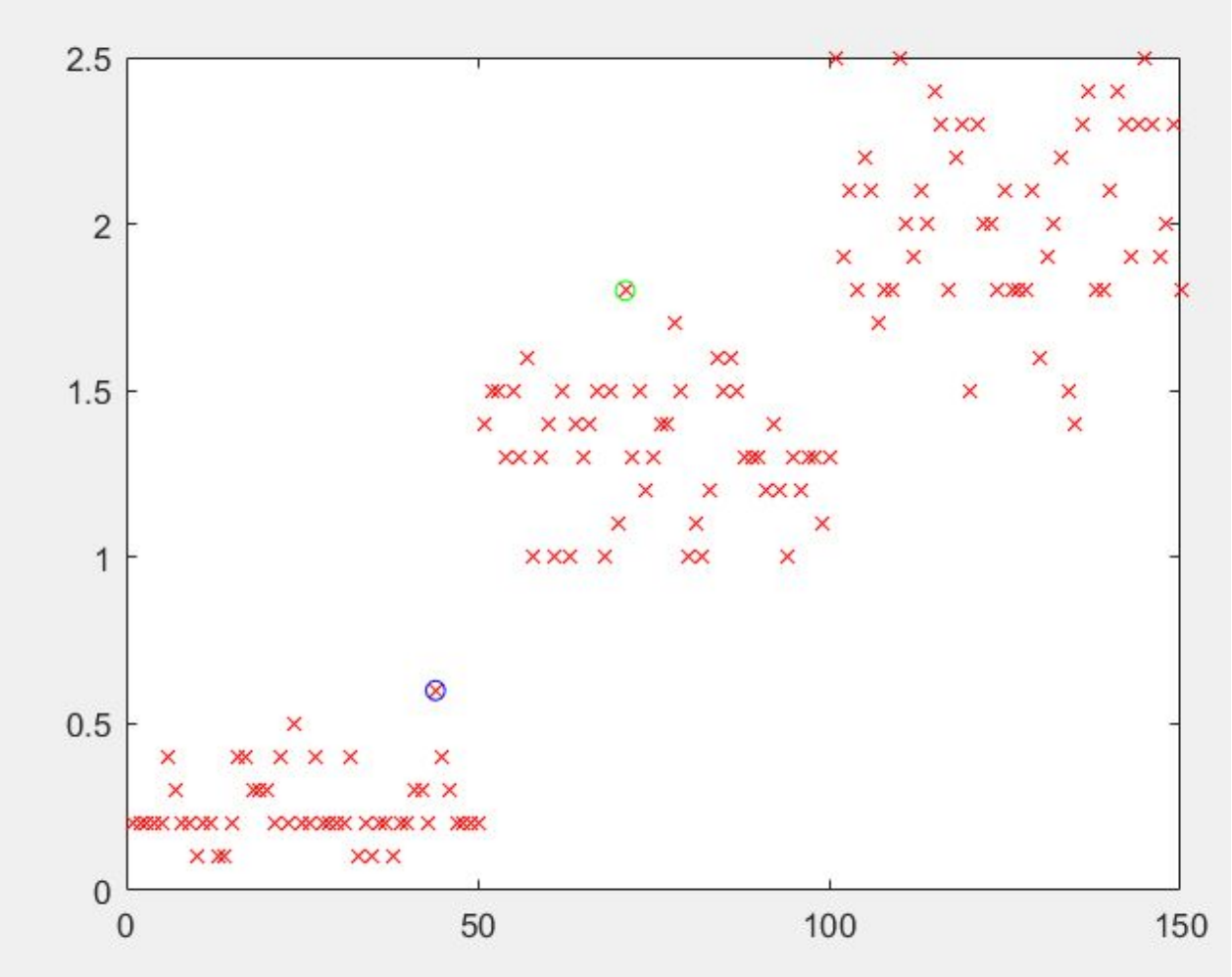
Weights = 6.2786 0.0087 0.0068

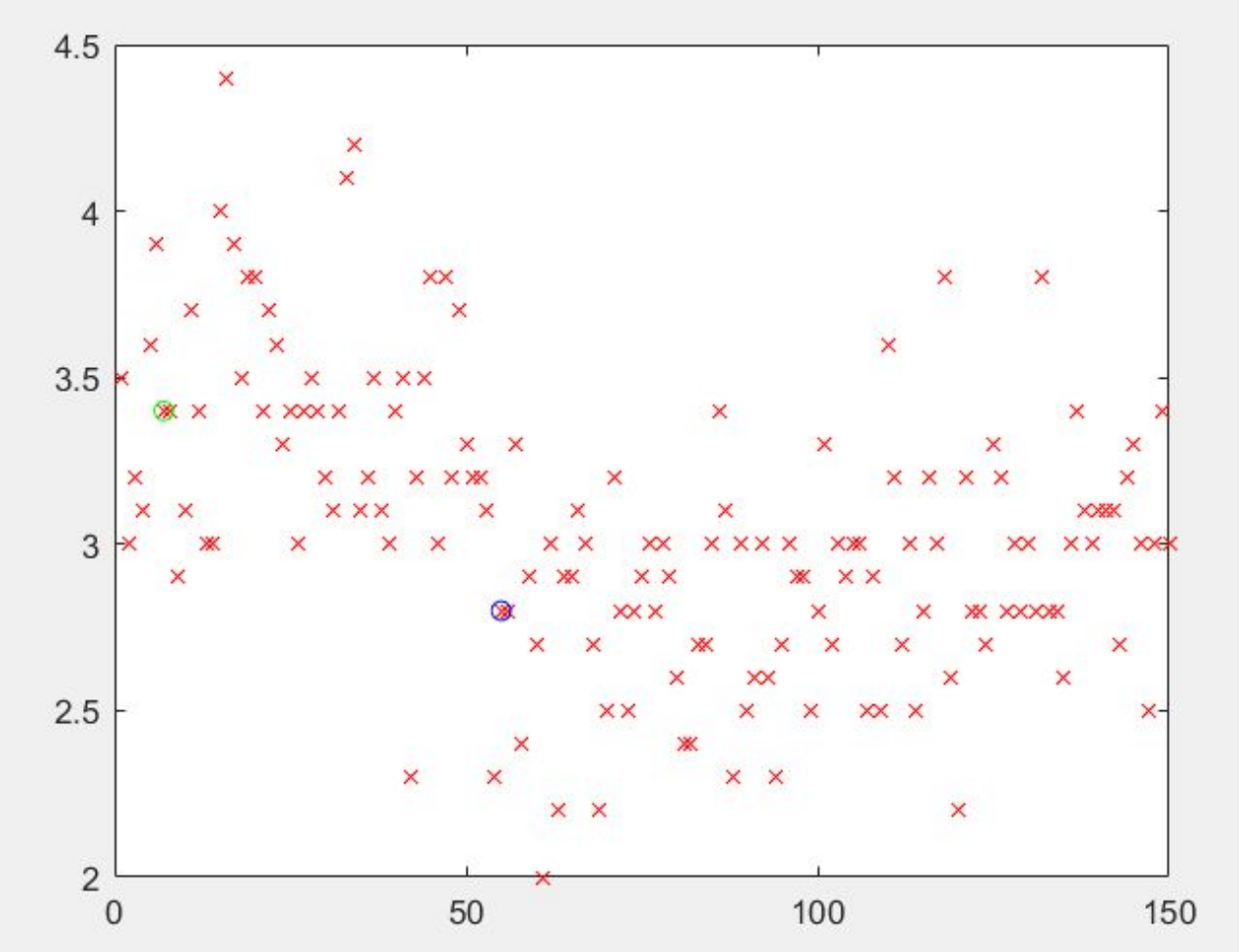
Question 6 :

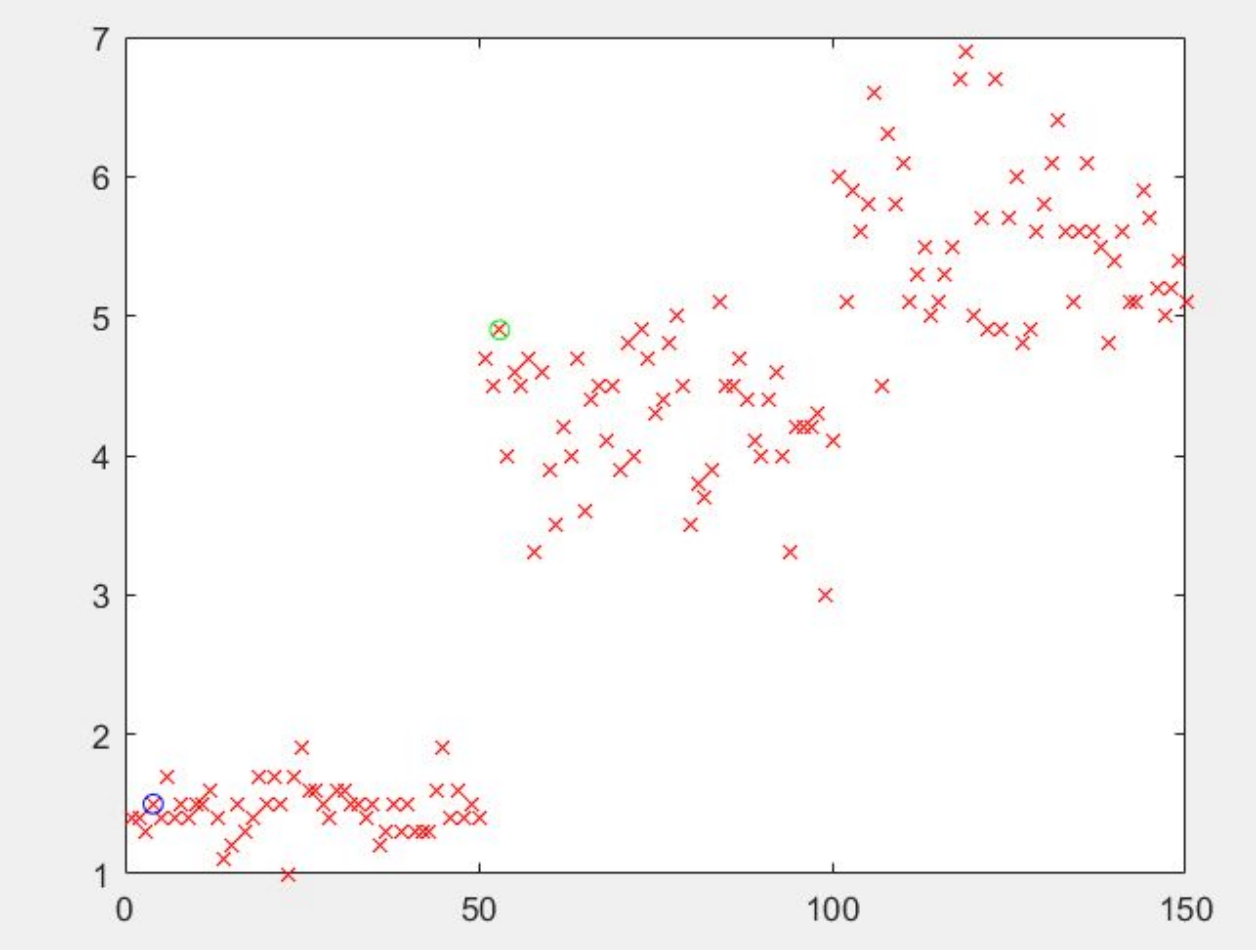
```
clear;
clc;
close all;

X = xlsread('data2.xlsx', 'Sheet1');
[m, n] = size(X);

iter = 1000;
c1 = zeros(n,1);
c2 = zeros(n,1);

for feature = 1 : n
    classifier = zeros(m,n);
    D = zeros(m,2);
    min_d = [5 5];
    k1 = 0;
    k2 = 0;

    % Initialization of centers
    rand1 = randi([1,150],1,1);
    rand2 = randi([1,150],1,1);
    c1(feature) = X(rand1,feature);
    c2(feature) = X(rand2,feature);

    for k = 1 : iter
        % Assigning the data points to a center
        for i = 1 : m
            d1 = sqrt((X(i,feature)-c1(feature))^2);
            d2 = sqrt((X(i,feature)-c2(feature))^2);
            D(i,1) = d1;
            D(i,2) = d2;

            if d1 <= d2
                classifier(i,feature) = 1;
            elseif d1 > d2
                classifier(i,feature) = 0;
            end

            if (d1 < min_d(1)) && d1 ~= 0
                k1 = i;
                min_d(1) = d1;
            end

            if (d2 < min_d(2)) && d2 ~= 0
                k2 = i;
                min_d(2) = d2;
            end
        end

        % Move the centers :

        c1_new = 0;
        c2_new = 0;
```

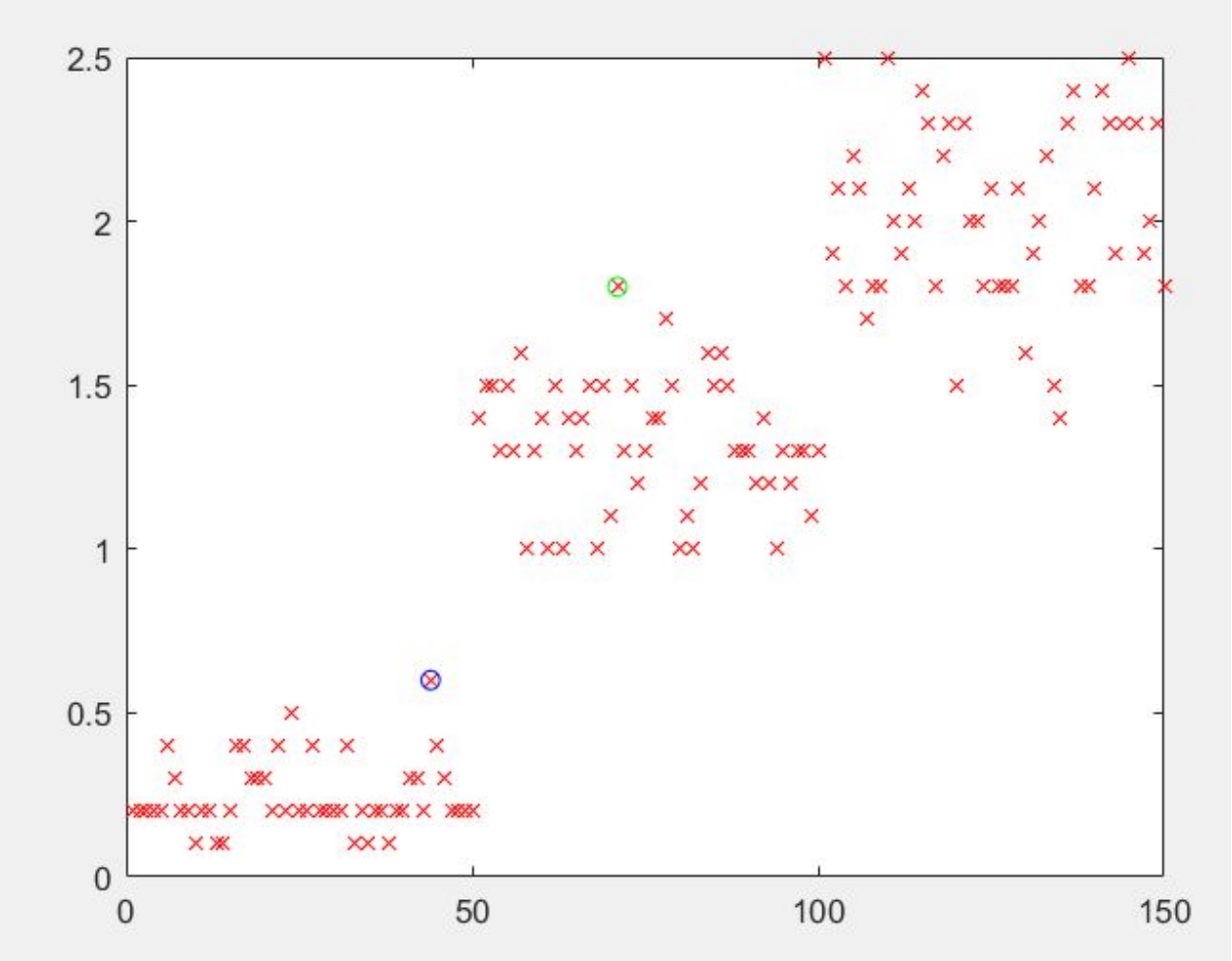
```
        c2_new = 0;
        num_c1_new = 0;
        num_c2_new = 0;
        for j = 1:m
            if classifier(j,feature) == 1
                c1_new = c1_new + X(j,feature);
                num_c1_new = num_c1_new + 1;
            else
                c2_new = c2_new + X(j,feature);
                num_c2_new = num_c2_new + 1;
            end
        end
        c1(feature) = c1_new/num_c1_new;
        c2(feature) = c2_new/num_c2_new;

    end

    min_D = min(D);

    figure(feature);
    plot(X(:,feature), 'rx');
    hold on;
    plot(k1, X(k1,feature), 'bo');
    hold on;
    plot(k2, X(k2, feature), 'go');
end
```

Center locations for different data columns :


2.5
2
1.5
1
0.5
0
0
50
100
150


4.5
4
3.5
3
2.5
2
0
50
100
150


7
6
5
4
3
2
1
0
50
100
150



Question 7 :

```
clc;
clear;
close all;

data = xlsread('data3.xlsx');
data = data(randperm(size(data,1)),:);
X = data(:,(1:4));
X = normalize(X);
Y = data(:,5);

train_x = X(1:60,:);
train_y = Y(1:60,:);
test_x = X(61:100,:);
test_y = Y(61:100,:);

[m,n] = size(train_x);
N = 2; % Number of classes
iter  = 1000;
alpha = 0.5;

rmin = -0.01;
rmax = 0.01;
w = rmin + (rmax-rmin)*rand(1,n);
h = zeros(m, 1);
h_output = zeros(size(test_x,1),1);
predicted_classes = zeros(size(test_x,1),1);

% Training :
for k = 1:iter
    for i = 1:m
        h(i) = 1/(1 - exp(-(train_x(i,:)*w')));
    end

    for i = 1:m
        cost = (train_y(i)*log(h(i)) + (1-train_y(i))*log(1-h(i)));
    end

    for j = 1:n
        gradient = 0;
        for i = 1:m
            gradient = gradient + (train_y(i)*(1-h(i))+(1-train_y(i))'*h(i))*train_x((i),j);
        end
        w(j) = w(j) - alpha*gradient;
    end
end

% Testing :
for i = 1:size(test_x,1)
    h_output(i) = 1/(1 - exp(-(test_x(i,:)*w')));
end

for i = 1:size(test_x,1)
    predicted_classes(i) = 1 + h_output(i);
```

```
53      predicted_classes(i) = 1 + h_output(i);
54  end
55
56  % Accuracy and Confusion matrix
57  cm = confusionmat(test_y, predicted_classes);
58  diagonal = 0;
59  for i = 1:2
60      diagonal = diagonal + cm(i,i);
61  end
62  overall_accuracy = diagonal/sum(sum(cm));
63
```

cm

2x2 double

	1	2	3
1	24	0	
2	0	16	
3			

Overall accuracy = 1

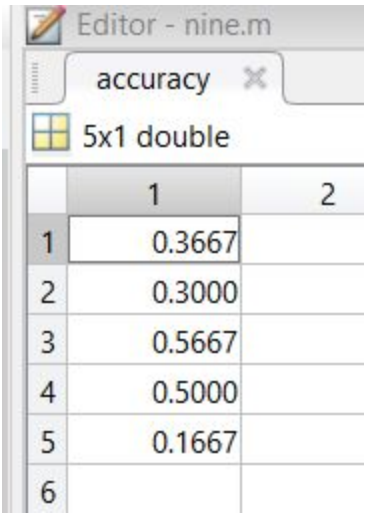
Question 9 :

```
clc;
clear;
close all;

data = xlsread('data4.xlsx');
data = data(randperm(size(data,1)),:);
X = data(:,(1:4));
X = normalize(X);
Y = data(:,5);

iter  = 1000;
alpha = 0.5;
accuracy = zeros(5,1);

% 5 fold cross validation
for f = 1:5
    if f == 1
        x_train = X(31:150, :);
        y_train = Y(31:150, :);
        x_test = X(1:30, :);
        y_test = Y(1:30, :);
    elseif f == 2
        x_train = X([1:30 61:150], :);
        y_train = Y([1:30 61:150], :);
        x_test = X(31:60, :);
        y_test = Y(1:30, :);
    elseif f == 3
        x_train = X([1:60 91:150], :);
        y_train = Y([1:60 91:150], :);
        x_test = X(61:90, :);
        y_test = Y(61:90, :);
    elseif f == 4
        x_train = X([1:90 121:150], :);
        y_train = Y([1:90 121:150], :);
        x_test = X(91:120, :);
        y_test = Y(91:120, :);
    elseif f == 5
        x_train = X(1:120, :);
        y_train = Y(1:120, :);
        x_test = X(121:150, :);
        y_test = Y(121:150, :);
    end

    w = logistic_regression(x_train,y_train);

    for i = 1:size(x_test,1)
        y_output(i) = 1 + 1/(1 - exp(-(x_test(i,:)*w')));
    end

    diagonal = 0;
    cm = confusionmat(y_test,y_output);
    for g = 1:3
        diagonal = diagonal + cm(g,g);
```

```
        diagonal = 0;
        cm = confusionmat(y_test,y_output);
        for g = 1:3
            diagonal = diagonal + cm(g,g);
        end
        accuracy(f) = diagonal/sum(sum(cm));
    end
```

```
function w = logistic_regression(train_x, train_y)

[m,n] = size(train_x);
iter  = 1000;
alpha = 0.5;

rmin = -0.01;
rmax = 0.01;
w = rmin + (rmax-rmin)*rand(1,n);
h = zeros(m, 1);

% Training :
for k = 1:iter
    for i = 1:m
        h(i) = 1/(1 - exp(-(train_x(i,:)*w')));
    end

    for i = 1:m
        cost = (train_y(i)*log(h(i)) + (1-train_y(i))*log(1-h(i)));
    end

    for j = 1:n
        gradient = 0;
        for i = 1:m
            gradient = gradient + (train_y(i)*(1-h(i))+(1-train_y(i))'*h(i))*train_x((i),j);
        end
        w(j) = w(j) - alpha*gradient;
    end
end
end
```

Accuracy values for the different k-fold cross validation runs :

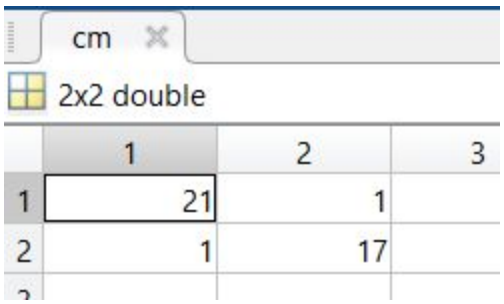

Editor - nine.m
accuracy
5x1 double
1
2
1
0.3667
2
0.3000
3
0.5667
4
0.5000
5
0.1667
6

Question 10 :

```
clc;
clear;
close all;

data = xlsread('data3.xlsx');
data = data(randperm(size(data,1)),:);
X = data(:,(1:4));
X = normalize(X);
Y = data(:,5);

train_x = X(1:60,:);
train_y = Y(1:60,:);
test_x = X(61:100,:);
test_y = Y(61:100,:);

[m,n] = size(train_x);
N = 2; % Number of classes

py = zeros(1,N);
pxy = zeros(40,N);
pyx = zeros(40,N);

% Calculate Priors :
for i = 1:m
    if train_y(i) == 1
        py(1) = py(1) + 1;
    elseif train_y(i) == 2
        py(2) = py(2) + 1;
    end
end
py(1) = 1/py(1);
py(2) = 1/py(2);

% Calculate Likelihood :
mu = [mean(train_x(train_y==1)) mean(train_x(train_y==2))];
sigma = [std(train_x(train_y==1)) std(train_x(train_y==2))];
for i = 1:40
    pxy(i, 1) = (1/(sqrt(2*pi)*sqrt(abs(sigma(1)))))*exp(-0.5*((test_x(i)-mu(1))^2)/(sigma(1)^2));
    pxy(i, 2) = (1/(sqrt(2*pi)*sqrt(abs(sigma(2)))))*exp(-0.5*((test_x(i)-mu(2))^2)/(sigma(2)^2));
end

% Prediction :
y_pred = zeros(40, 1);
for i = 1:40
    if pxy(i, 1)*py(1) > pxy(i, 2)*py(2)
        y_pred(i) = 1;
    elseif pxy(i, 1)*py(1) < pxy(i, 2)*py(2)
        y_pred(i) = 2;
    end
end

% Accuracy and Confusion matrix
cm = confusionmat(test_y,y_pred);
```

```
52   % Accuracy and Confusion matrix
53   cm = confusionmat(test_y,y_pred);
54   diagonal = 0;
55   for i = 1:2
56       diagonal = diagonal + cm(i,i);
57   end
58   overall_accuracy = diagonal/sum(sum(cm));
59
```

Confusion matrix :



	1	2	3
1	21	1	
2	1	17	

Accuracy = 0.95

Question 11 :

```
clc;
clear;
close all;

data = xlsread('data4.xlsx');
data = data(randperm(size(data,1)),:);
X = data(:,(1:4));
X = normalize(X);
Y = data(:,5);

train_x = X(1:105,:);
train_y = Y(1:105,:);
test_x = X(106:150,:);
test_y = Y(106:150,:);

[m,n] = size(train_x);

py = zeros(1,3);
pxy = zeros(45,3);
pyx = zeros(45,3);

% Calculate Priors :
for i = 1:m
    if train_y(i) == 1
        py(1) = py(1) + 1;
    elseif train_y(i) == 2
        py(2) = py(2) + 1;
    elseif train_y(i) == 3
        py(3) = py(3) + 1;
    end
end
py(1) = 1/py(1);
py(2) = 1/py(2);
py(3) = 1/py(3);

% Calculate Likelihood :
mu = [mean(train_x(train_y==1)) mean(train_x(train_y==2)) mean(train_x(train_y==3))];
sigma = [std(train_x(train_y==1)) std(train_x(train_y==2)) mean(train_x(train_y==3))];
for i = 1:45
    pxy(i, 1) = (1/(sqrt(2*pi)*sqrt(abs(sigma(1)))))*exp(-0.5*((test_x(i)-mu(1))^2)/(sigma(1)^2));
    pxy(i, 2) = (1/(sqrt(2*pi)*sqrt(abs(sigma(2)))))*exp(-0.5*((test_x(i)-mu(2))^2)/(sigma(2)^2));
    pxy(i, 3) = (1/(sqrt(2*pi)*sqrt(abs(sigma(3)))))*exp(-0.5*((test_x(i)-mu(3))^2)/(sigma(3)^2));
end

% Prediction :
y_pred = zeros(45, 1);
for i = 1:45
    pyx(i, 1) = pxy(i, 1)*py(1);
    pyx(i, 2) = pxy(i, 2)*py(2);
    pyx(i, 3) = pxy(i, 3)*py(3);
    [val, idx] = max([pyx(i, 1) pyx(i, 2) pyx(i, 3)]);
    y_pred(i) = idx;
end
```

```
% Accuracy and Confusion matrix
cm = confusionmat(test_y,y_pred);
diagonal = 0;
for i = 1:3
    diagonal = diagonal + cm(i,i);
end
overall_accuracy = diagonal/sum(sum(cm));
```

Confusion matrix :

cm

3x3 double

	1	2	3
1	17	2	0
2	0	10	3
3	0	2	11
4			

Accuracy = 0.844

Question 12 :

```
clc;
clear;
close all;

data = xlsread('data4.xlsx');
data = data(randperm(size(data,1)),:);
X = data(:,(1:4));
X = normalize(X);
Y = data(:,5);

train_x = X(1:105,:);
train_y = Y(1:105,:);
test_x = X(106:150,:);
test_y = Y(106:150,:);

[m,n] = size(train_x);

py = zeros(1,3);
pxy = zeros(45,3);

% Calculate Priors :
for i = 1:m
    if train_y(i) == 1
        py(1) = py(1) + 1;
    elseif train_y(i) == 2
        py(2) = py(2) + 1;
    elseif train_y(i) == 3
        py(3) = py(3) + 1;
    end
end
py(1) = 1/py(1);
py(2) = 1/py(2);
py(3) = 1/py(3);

% Calculate Likelihood :
mu = [mean(train_x(train_y==1)) mean(train_x(train_y==2)) mean(train_x(train_y==3))];
sigma = [std(train_x(train_y==1)) std(train_x(train_y==2)) mean(train_x(train_y==3))];
for i = 1:45
    pxy(i, 1) = (1/(sqrt(2*pi)*sqrt(abs(sigma(1)))))*exp(-0.5*((test_x(i)-mu(1))^2)/(sigma(1)^2));
    pxy(i, 2) = (1/(sqrt(2*pi)*sqrt(abs(sigma(2)))))*exp(-0.5*((test_x(i)-mu(2))^2)/(sigma(2)^2));
    pxy(i, 3) = (1/(sqrt(2*pi)*sqrt(abs(sigma(3)))))*exp(-0.5*((test_x(i)-mu(3))^2)/(sigma(3)^2));
end

% Prediction :
y_pred = zeros(45, 1);
for i = 1:45
    [val, idx] = max([pxy(i, 1) pxy(i, 2) pxy(i, 3)]);
    y_pred(i) = idx;
end
```

```
% Accuracy and Confusion matrix
cm = confusionmat(test_y,y_pred);
diagonal = 0;
for i = 1:3
    diagonal = diagonal + cm(i,i);
end
overall_accuracy = diagonal/sum(sum(cm));
```

Confusion matrix :

cm

3x3 double

	1	2	3
1	17	0	0
2	1	15	3
3	0	2	7
4			

Accuracy = 0.8667