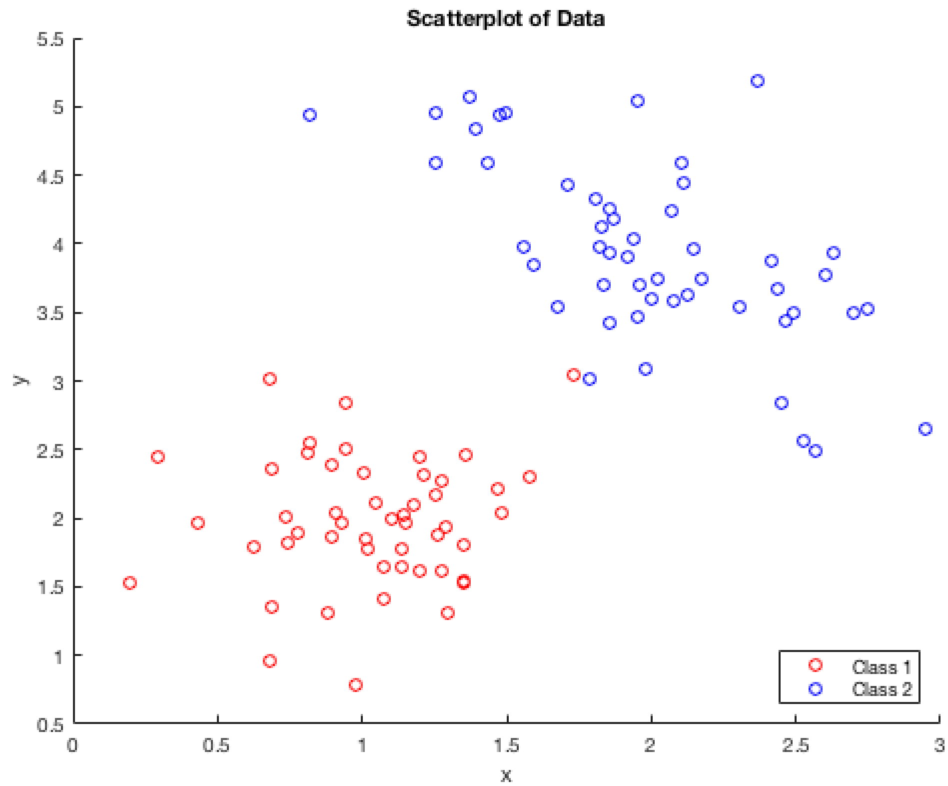
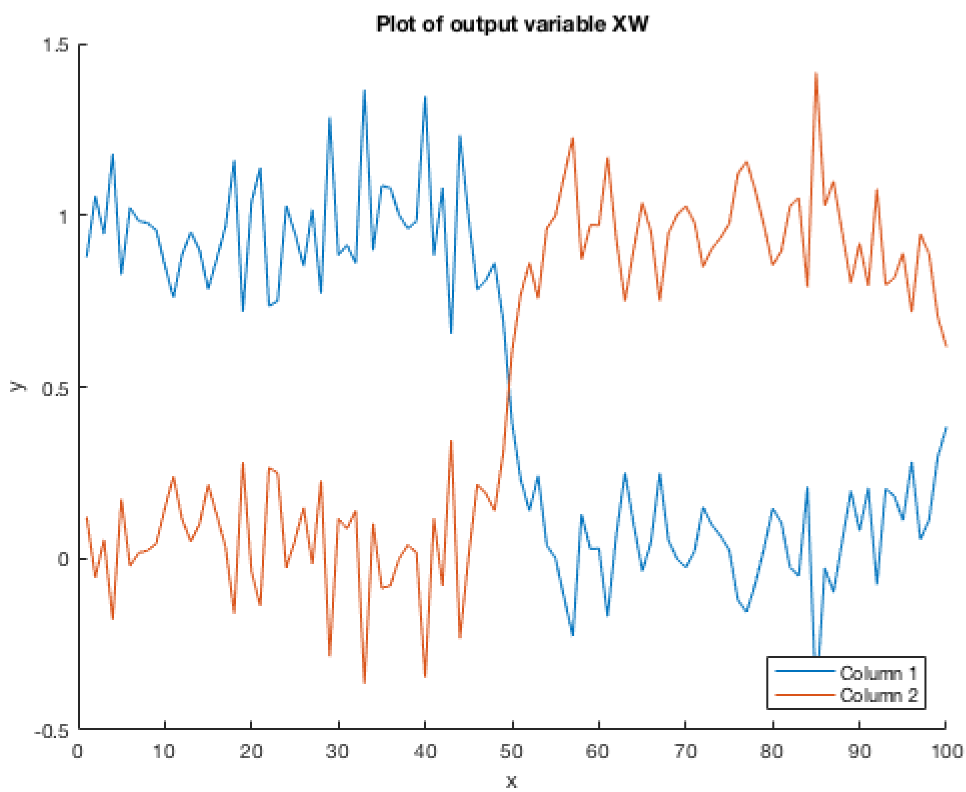
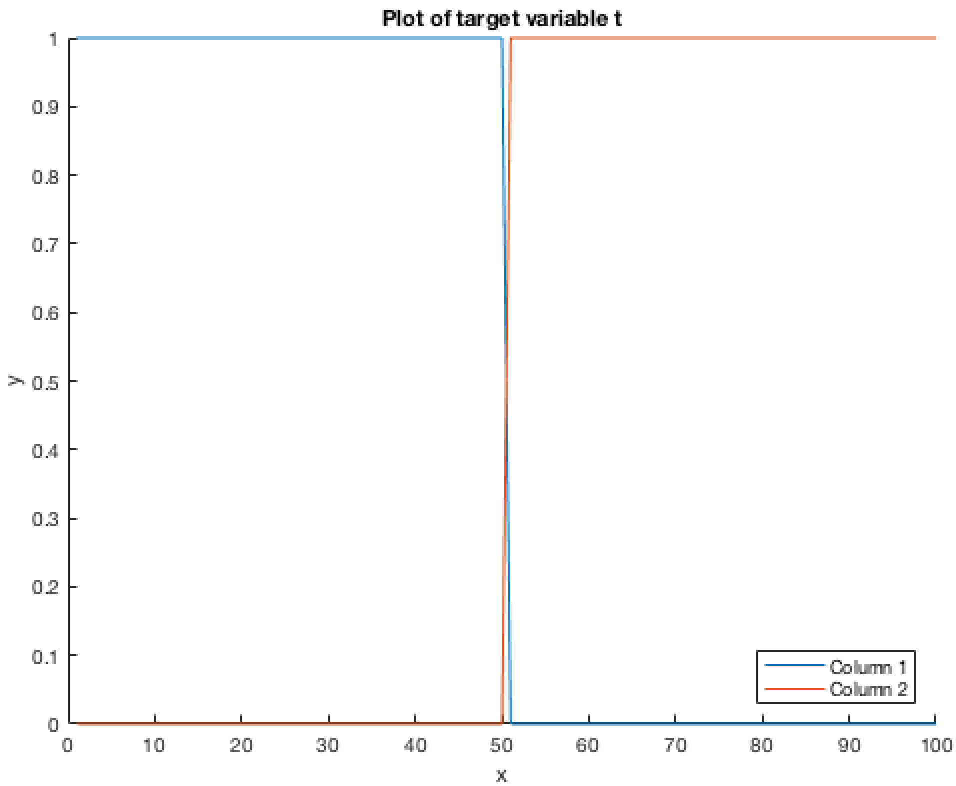
```matlab
%{
1. Generate N = 50 two-dimensional training data points distributed
 according
to the given distributions
%}
m_u = [1, 2];
sigma = [0.1 0.05;0.05 0.2];

m_u2 = [2, 4];
sigma2 = [0.2 -0.1;-0.1 0.3];

rng(40);
r = mvnrnd(m_u, sigma, 50);
r2 = mvnrnd(m_u2, sigma2, 50);

%{
2.
%}
figure
hold on
scatter(r(:,1), r(:,2), 'r');
scatter(r2(:,1), r2(:,2),'b');
legend('Class 1', 'Class 2', 'Location', 'Southeast');
xlabel('x');
ylabel('y');
title('Scatterplot of Data');
hold off
```



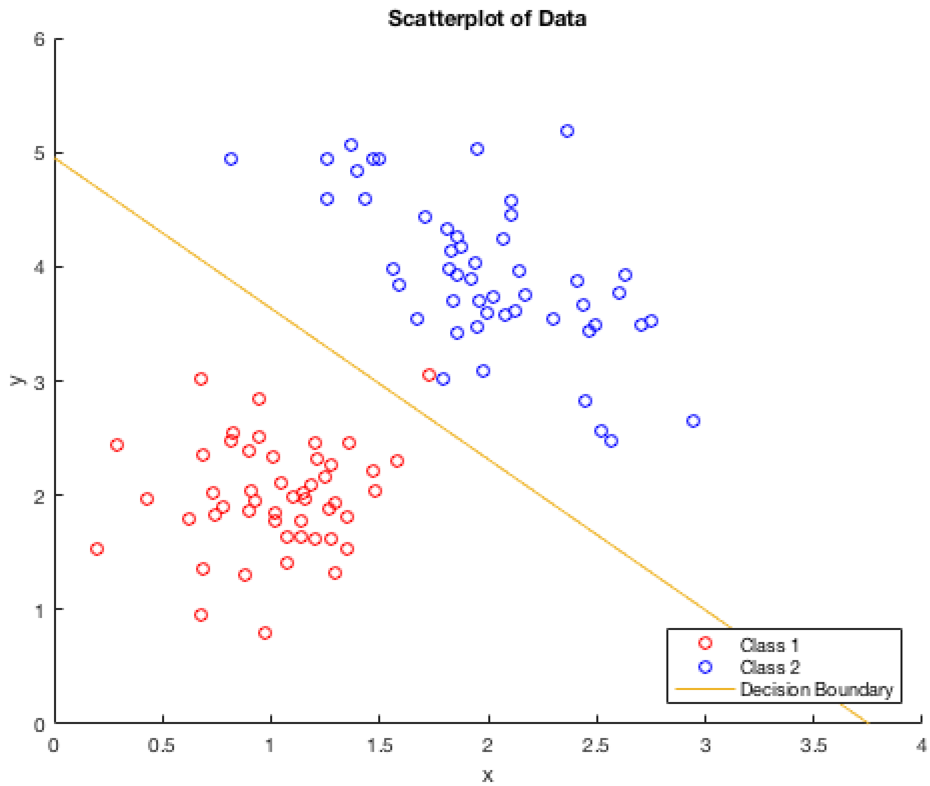

```
%{
3.
%}

x_temp = [r; r2];
data = horzcat(ones(100,1), x_temp);
c1 = ones(50,1);
top_half = horzcat(c1, zeros(50,1));
bot_half = horzcat(zeros(50,1), c1);
t = [top_half; bot_half];

first = inv(data'*data);
second = data'*t;
result = first*second;

x_w = data*result;

figure
hold on
plot(x_w(:,1));
plot(x_w(:,2));
legend('Column 1', 'Column 2', 'Location', 'Southeast');
xlabel('x');
ylabel('y');
title('Plot of output variable XW');
hold off
```

```
figure
hold on
plot(t(:,1));
plot(t(:,2));
legend('Column 1', 'Column 2', 'Location', 'Southeast');
xlabel('x');
ylabel('y');
title('Plot of target variable t');
hold off

%{
Comment: We see that the plot of X_W looks very similar to the plot of
 our
target variable t. In the plot of X_W, we see that from x = 1:50 our y
values oscillate around 1 and then oscillate around 0 for x = 50:100.
Similarly, we see the same thing in the plot of our target variable t,
 but
with no oscillation since we don't have noise.
%}
```






```
%{
4.
%}
%{
Need to find two points that satisfy: (w_1 - w_2)^t*(x) = (w_20 -
 w_10)
We can find these points by finding the intercepts (i.e setting
x and y to 0). The resulting points are : (3.755, 0) and (0, 4.953)

%}
figure
hold on
scatter(r(:,1), r(:,2), 'r');
scatter(r2(:,1), r2(:,2),'b');
plot([3.755 0], [0 4.953]);
legend('Class 1', 'Class 2', 'Decision Boundary'
 , 'Location', 'Southeast');
xlabel('x');
ylabel('y');
title('Scatterplot of Data');
hold off
```

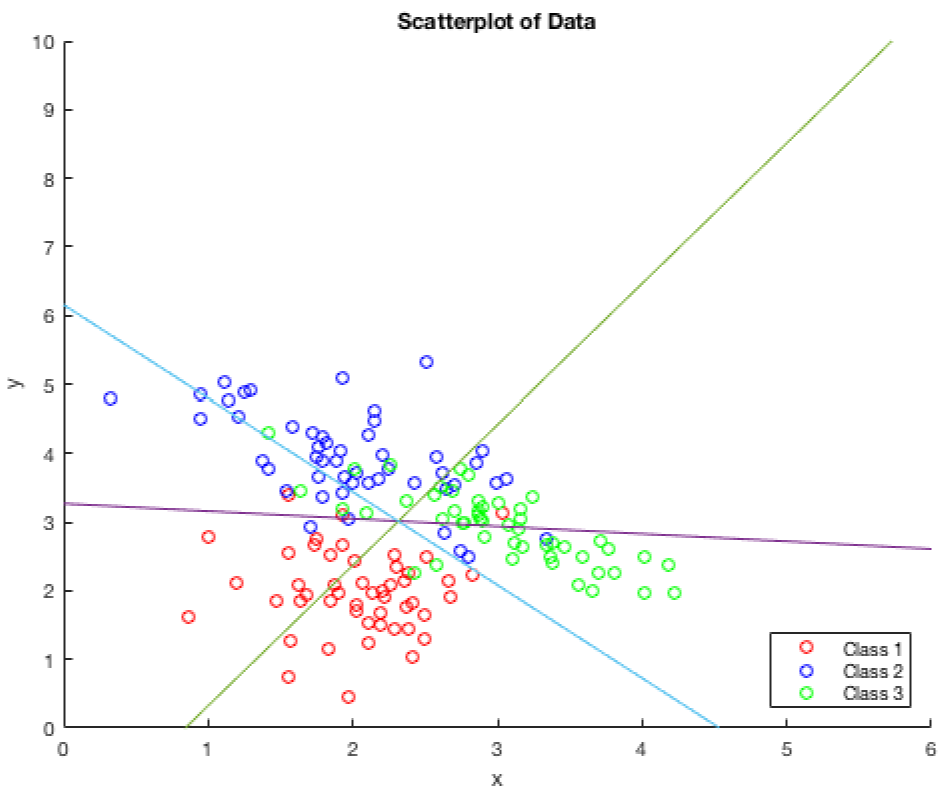



```
%{
5.
Generate N = 50 two-dimensional training data points distributed
 according
to the distributions:
%}
mean1 = [2, 2];
variance1 = [0.2 0.05;0.05 0.3];

mean2 = [2, 4];
variance2 = [0.4 -0.1;-0.1 0.3];

mean3 = [3, 3];
variance3 = [0.5 -0.3;-0.3 0.4];

rng(40);
class1 = mvnrnd(mean1, variance1, 50);
class2 = mvnrnd(mean2, variance2, 50);
class3 = mvnrnd(mean3, variance3, 50);

%{
6.
Compute the LS classificiation coefficients, and plot the decision
 boundaries,
using the result of question 3 in the theory part. Be sure to
 represent the
```

```
actual 3-class decision boundaries, and not just the pairwise ones.
You can just highlight them with pen using the pairwise boundaries to
 avoid
a few tedious lines of code.

%}
x_temp = [class1; class2; class3];
data2 = horzcat(ones(150,1), x_temp);
c1 = ones(50,1);
top_half = horzcat(c1, zeros(50,2));
mid = horzcat(zeros(50,1), c1, zeros(50,1));
bot_half = horzcat(zeros(50,2), c1);
t = [top_half; mid; bot_half];
tic
first = inv(data2'*data2);
second = data2'*t;
result2 = first*second;

x_w2 = data2*result2;
toc


%{
To plot decision boundaries, we plot three different lines satisfying:
(w_1 - w_3)^t * x = w_30 - w_10
(w_1 - w_2)^t * x = w_20 - w_10
(w_2 - w_3)^t * x = w_30 - w_20

Result:
x = (29.699, 0) and x = (0, 3.271)
x = (.84168, 0) and x = (0, -1.723)
x = (4.535, 0) and x = (0, 6.1558)
%}

figure
hold on
scatter(class1(:,1), class1(:,2), 'r');
scatter(class2(:,1), class2(:,2),'b');
scatter(class3(:,1), class3(:,2),'g');

%boundary between class 1 and class 2
plot([29.699 0], [0 3.271]);

decision = 0:1:10;
%Need to explicity plot decisino boundary for class 2 and class 3
y_val = zeros(1, 11);
for i = 1:11
    y_val(i) = decision(i)*2.047 - 1.723;
end
%decision boundary between class 2 and class 3
plot(decision, y_val);

%decision boundary between class 1 and class 3
plot([4.535 0], [0 6.1558]);
```

```
legend('Class 1', 'Class 2', 'Class 3', 'Location', 'Southeast');
xlabel('x');
ylabel('y');
title('Scatterplot of Data');
xlim([0 6]);
ylim([0 10]);
hold off
```

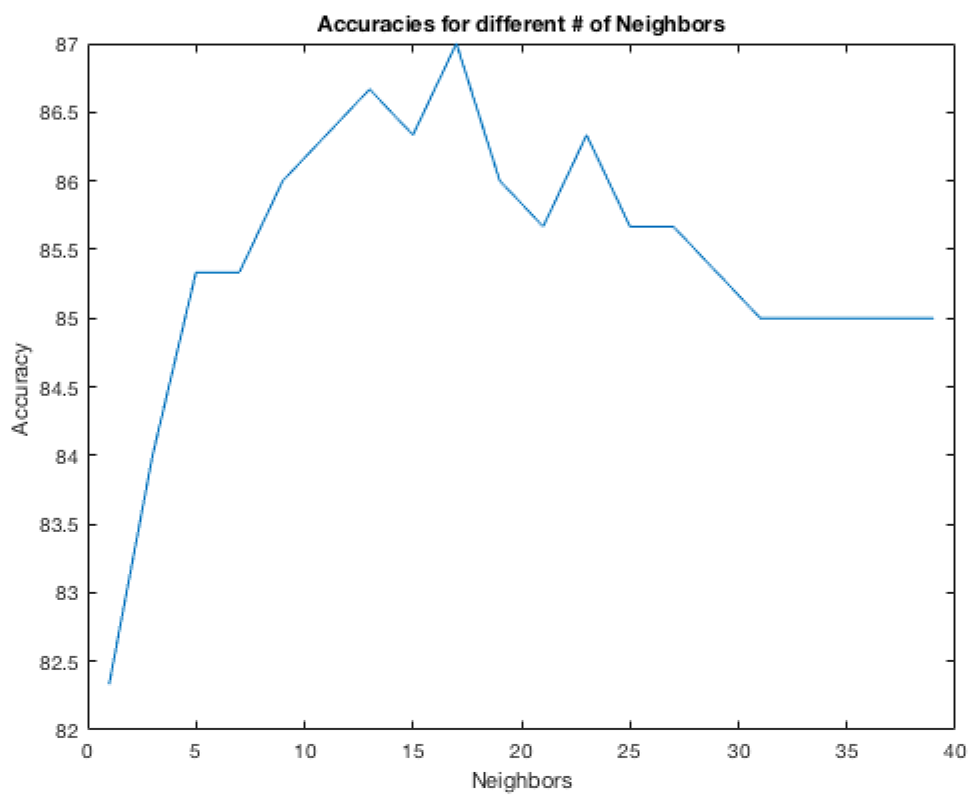
Elapsed time is 0.043653 seconds.




```
%{
7.
%}
rng(40);
test1 = mvnrnd(mean1, variance1, 100);
test2 = mvnrnd(mean2, variance2, 100);
test3 = mvnrnd(mean3, variance3, 100);
xtemp2 = [test1; test2; test3];
data3 =  horzcat(ones(300,1), xtemp2);

predicted = zeros(300, 3);
for i = 1:300
    max_val = [data3(i,:)*result2(:,1) data3(i,:)*result2(:,2)
 data3(i,:)*result2(:,3)];
    [label, index] = max(max_val);
    predicted(i, index) = 1;
end
```

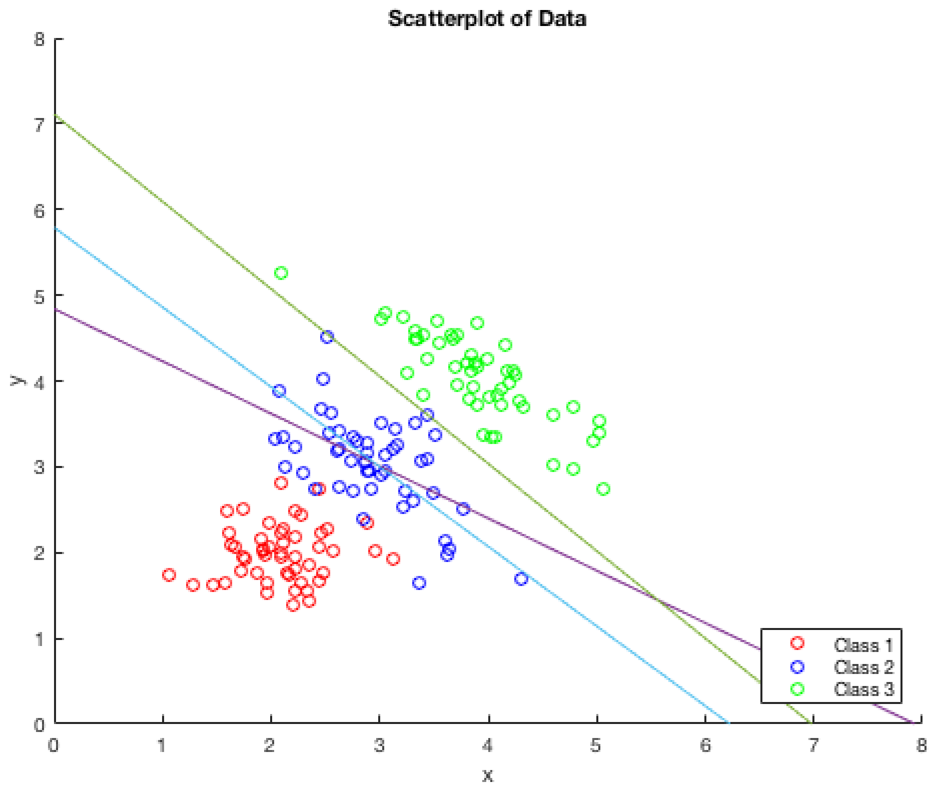
```
sum = 0;
for i = 1:100
    if predicted(i,1) == 1
        sum = sum + 1;
    end
end
for i = 101:200
    if predicted(i,2) == 1
        sum = sum + 1;
    end
end
for i = 201:300
    if predicted(i,3) == 1
        sum = sum + 1;
    end
end
accuracy = (100*sum)/300;

%{
We get that our accuracy is equal to 86. If c = 100, then we would
 correctly
classify all our points. We cannot have c = 100 because
the gaussian distributions that generated our points overlap in our
 input
space, so we cannot form hyperplanes that perfectly separate the
 points.
Thus, we will have some errors in classifying points that are close to
 our
decision boundary.
%}

%9.
accuracies = zeros(1, 20);
for i = 1:20
    pred_k = knn(x_temp, xtemp2, t, 2*i);
    sum = 0;
    for j = 1:100
        if pred_k(j,1) == 1
            sum = sum + 1;
        end
    end
    for j = 101:200
        if pred_k(j,2) == 1
            sum = sum + 1;
        end
    end
    for j = 201:300
        if pred_k(j,3) == 1
            sum = sum + 1;
        end
    end
    accuracies(1, i) = (100*sum)/300;
end
```

```
figure
plot(1:2:40, accuracies(1,:));
title('Accuracies for different # of Neighbors');
ylabel('Accuracy');
xlabel('Neighbors');


%{
We see that our KNN accuracy increases from k = 1 to 5. Then, the
 accuracy
plot oscillates around 83% and then increases again from k = 11 to 23.
The accuracy hovers around 84% after k = 23 neighbors. While the
 accuracies
are basically equivalent, the running time for least squares is much
 faster
than that of KNN. Thus, I would suggest using least squares
 classification
for our problem.
%}
```



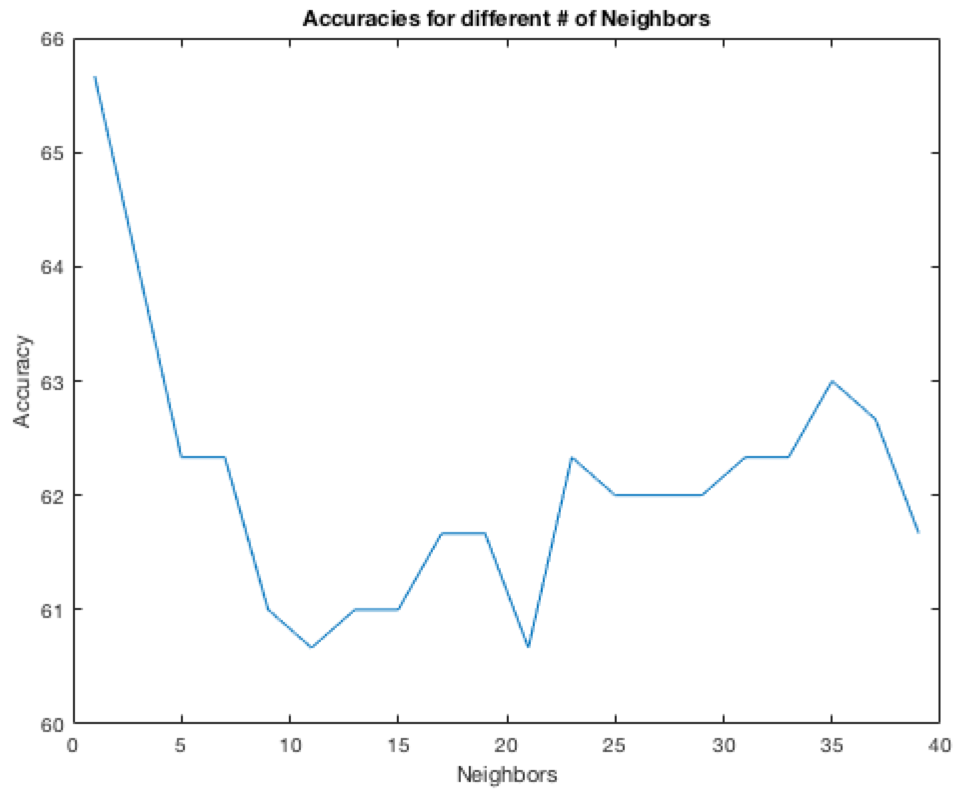

```
%10.

%Generate Points according to distribution
mean_10_1 = [2, 2];
variance_10_1 = [0.1 0;0 0.1];
```

```
mean_10_2 = [3, 3];
variance_10_2 = [0.2 -0.1;-0.1 0.3];

mean_10_3 = [4, 4];
variance_10_3 = [0.4 -0.3;-0.3 0.3];

rng default;
class_10_1 = mvnrnd(mean_10_1, variance_10_1, 50);
class_10_2 = mvnrnd(mean_10_2, variance_10_2, 50);
class_10_3 = mvnrnd(mean_10_3, variance_10_3, 50);


%Combine matrices to get linear model and solve Least Squares
x_temp_10 = [class_10_1; class_10_2; class_10_3];
data_10 = horzcat(ones(150,1), x_temp_10);
c1 = ones(50,1);
top_half_10 = horzcat(c1, zeros(50,2));
mid_10 = horzcat(zeros(50,1), c1, zeros(50,1));
bot_half_10 = horzcat(zeros(50,2), c1);
t_10 = [top_half_10; mid_10; bot_half_10];
first_10 = inv(data_10'*data_10);
second_10 = data_10'*t_10;
result_10 = first_10*second_10;
x_w_10 = data_10*result_10;

%{
To plot decision boundaries, we plot three different lines satisfying:
(w_1 - w_3)^t * x = w_30 - w_10
(w_1 - w_2)^t * x = w_20 - w_10
(w_2 - w_3)^t * x = w_30 - w_20

Result:
x = (7.93, 0) and x = (0, 4.84)
x = (5.179, 0) and x = (0, 7.11)
x = (6.225, 0) and x = (0, 9.998)
%}

figure
hold on
scatter(class_10_1(:,1), class_10_1(:,2), 'r');
scatter(class_10_2(:,1), class_10_2(:,2),'b');
scatter(class_10_3(:,1), class_10_3(:,2),'g');
plot([7.93 0], [0 4.84]);
plot([6.98 0], [0 7.11]);
plot([6.225 0], [0 5.79]);
legend('Class 1', 'Class 2', 'Class 3', 'Location', 'Southeast');
xlabel('x');
ylabel('y');
title('Scatterplot of Data');
hold off
```




```
rng default;
test_10_1 = mvnrnd(mean_10_1, variance_10_1, 100);
test_10_2 = mvnrnd(mean_10_2, variance_10_2, 100);
test_10_3 = mvnrnd(mean_10_3, variance_10_3, 100);
xtemp_10_2 = [test1; test2; test3];
data_10_3 =  horzcat(ones(300,1), xtemp_10_2);

predicted_10 = zeros(300, 3);
for i = 1:300
    max_val = [data_10_3(i,:)*result_10(:,1)
 data_10_3(i,:)*result_10(:,2) data_10_3(i,:)*result_10(:,3)];
    [label, index] = max(max_val);
    predicted_10(i, index) = 1;
end

%Class by class accuracies
sum = 0;
for i = 1:100
    if predicted_10(i,1) == 1
        sum = sum + 1;
    end
end
for i = 101:200
    if predicted_10(i,2) == 1
        sum = sum + 1;
    end
```

```
end
for i = 201:300
    if predicted_10(i,3) == 1
        sum = sum + 1;
    end
end
accuracy_10 = (100*sum)/300;

%{
Accuracy is
%}

%KNN Part
accuracies_10 = zeros(1, 20);
for i = 1:20

    pred_k = knn(x_temp_10, xtemp_10_2, t_10, 2*i);

    sum = 0;
    for j = 1:100
        if pred_k(j,1) == 1
            sum = sum + 1;
        end
    end
    for j = 101:200
        if pred_k(j,2) == 1
            sum = sum + 1;
        end
    end
    for j = 201:300
        if pred_k(j,3) == 1
            sum = sum + 1;
        end
    end
    accuracies_10(1, i) = (100*sum)/300;
end

figure
plot(1:2:40, accuracies_10(1,:));
title('Accuracies for different # of Neighbors');
ylabel('Accuracy');
xlabel('Neighbors');
```



Published with MATLAB® R2017a