Homework 5 – Topics in Data Science ECE 592

Name: Alhiet Orbegoso
ID: 200322491

Problem 1

a. Express the pdf of the Gaussian source.

Statement provides the following input:

- Number of Gaussians: k=2, (Classes: Red=1 and Blue=2)
- Mean of Gaussians: u_k, for k =1,2
- Covariance matrices of Gaussians: $\Sigma_k = \sigma^2 . I$, for k =1,2 (All covariance matrices are the same)

From the total probability theorem:

$$P(X) = P(X|Class = Red). P(Red) + P(X|Class = Blue). P(Blue)$$

Assuming red (k=1) and blue (k=2) are i.i.d then the probability of X is:

$$f(X) = \frac{1}{\sqrt{(2\pi)^2|\Sigma_1|}} e^{\left[-\frac{1}{2}(X-u_1)^T\Sigma_1^{-1}(X-u_1)\right]} . \frac{1}{2} + \frac{1}{\sqrt{(2\pi)^2|\Sigma_2|}} e^{\left[-\frac{1}{2}(X-u_2)^T\Sigma_2^{-1}(X-u_2)\right]} . \frac{1}{2}$$

Reducing using the statement information:

$$\boldsymbol{f(X) = \frac{1}{4\pi\sigma}\left(e^{-\frac{1}{2\sigma^2}[(X-u_1)^T(X-u_1)]} + e^{-\frac{1}{2\sigma^2}[(X-u_2)^T(X-u_2)]}\right)}$$

b. Express the posterior probability.

Applying Bayes rule, the posterior probability is given by:

$$P(Class = Red|X) = \frac{P(Class = Red, X)}{P(Class = Red, X) + P(Class = Blue, X)} \dots (1)$$

But:

$$P(Class = Red, X) = P(X|Class = Red). P(Class = Red)$$

$$P(Class = Red, X) = \left(\frac{1}{\sqrt{(2\pi)^2|\Sigma_1|}} e^{\left[-\frac{1}{2}(X-u_1)^T\Sigma_1^{-1}(X-u_1)\right]}\right) . \left(\frac{1}{2}\right)$$

$$P(Class = Red, X) = \frac{1}{4\pi\sigma}\left(e^{-\frac{1}{2\sigma}[(X-u_1)^T(X-u_1)]}\right) \dots (2)$$

And from total probability theorem:

$$P(Class = Red, X) + P(Class = Blue, X) = f(X) \dots (3)$$

Replacing (2) and (3) in (1):

$$P(Class = Red|X) = \frac{\left(\frac{1}{4\pi\sigma}e^{-\frac{1}{2\sigma}[(X-u_1)^T(X-u_1)]}\right)}{\frac{1}{4\pi\sigma}\left(e^{-\frac{1}{2\sigma^2}[(X-u_1)^T(X-u_1)]} + e^{-\frac{1}{2\sigma^2}[(X-u_2)^T(X-u_2)]}\right)}$$

Reducing:

$$\boldsymbol{P(Class = Red|X) = \frac{e^{-\frac{1}{2\sigma}[(X-u_1)^T(X-u_1)]}}{e^{-\frac{1}{2\sigma^2}[(X-u_1)^T(X-u_1)]} + e^{-\frac{1}{2\sigma^2}[(X-u_2)^T(X-u_2)]}}}$$

c. Sketch the LDA.

From section b) the P(Class|X) is equal to the expression:

$$P(Class|X) = \frac{\left(\frac{1}{4\pi\sigma}e^{-\frac{1}{2\sigma^2}[(X-u_1)^T(X-u_1)]}\right)}{\frac{1}{4\pi\sigma}\left(e^{-\frac{1}{2\sigma^2}[(X-u_1)^T(X-u_1)]} + e^{-\frac{1}{2\sigma^2}[(X-u_2)^T(X-u_2)]}\right)}$$

Nevertheless, it was assumed only to classes (red and blue). For the solution of this problem will be assumed N number of classes, where each of this classes belong to a class red or blue.

For example, in ***classification.m*** MATLAB script if num_clusters = 3, then there will be 6 classes, but 3 of them belongs to blue class, the other 3 belong to red class:

$$classes = C_1, C_2, C_3, C_4, C_5, C_6, where:$$

$$\{C_1, C_2, C_3\} \in blue\ and\ \{C_4, C_5, C_6\}\ \in red$$

For each class it is calculated the P(class|X) using the equation below:

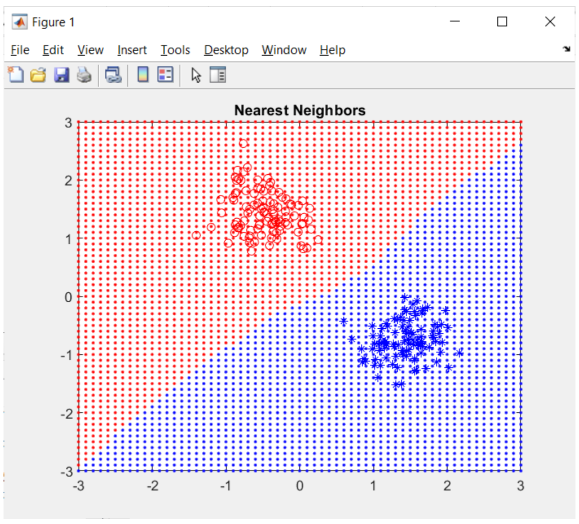
$$P(Class|X) = e^{-\frac{1}{2\sigma^2}[(X-u_1)^T(X-u_1)]}$$

Remark: The denominator is not considered given that it acts as a normalizing factor but in the MATLAB script the probabilities are normalized so that the range of each one is [0,1].

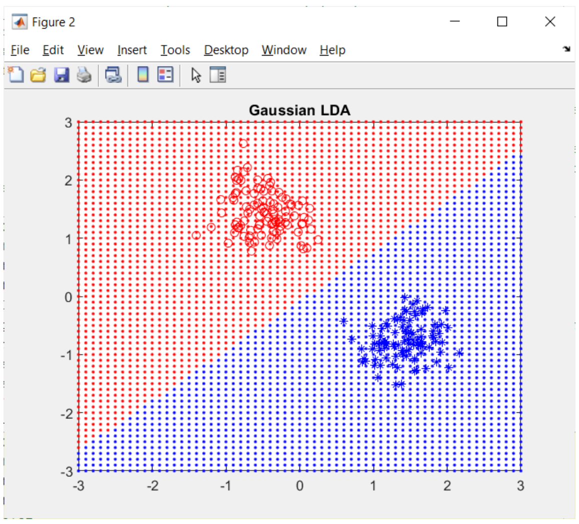
If 6 classes are assumed, there will be 6 conditional probabilities, where each one indicates the probability that X belong to each of the six classes. To assign the class, is taken the max conditional probability among the six values, then the max value indicates to which class belong the data X. For example, if the maximum value belongs to class = 4, then the X is red class for this example.

For class assignment, if the parameter num_clusters is "X" then all classes from 1 to X belong to the blue class and all classes from X+1 to 2X belong to red class.

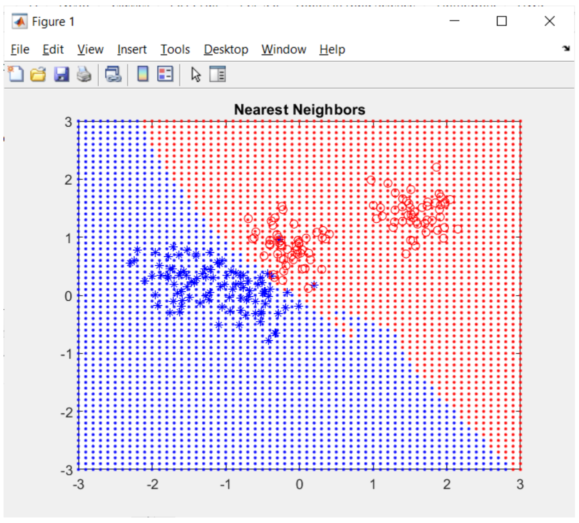
Following is presented the boundary plot for a given num_clusters, all plot are compared with NN:



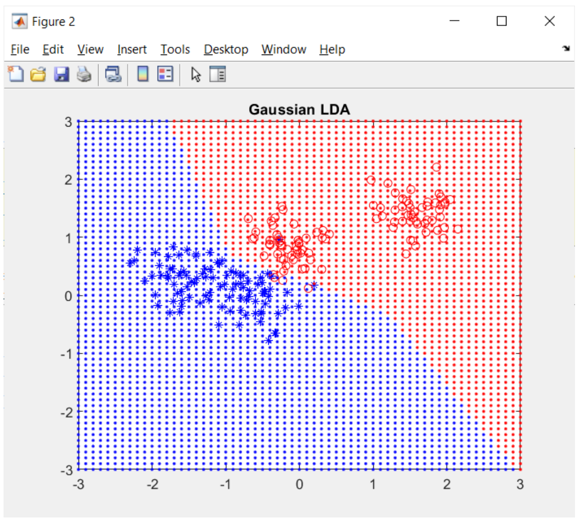
NN, num_clusters = 1



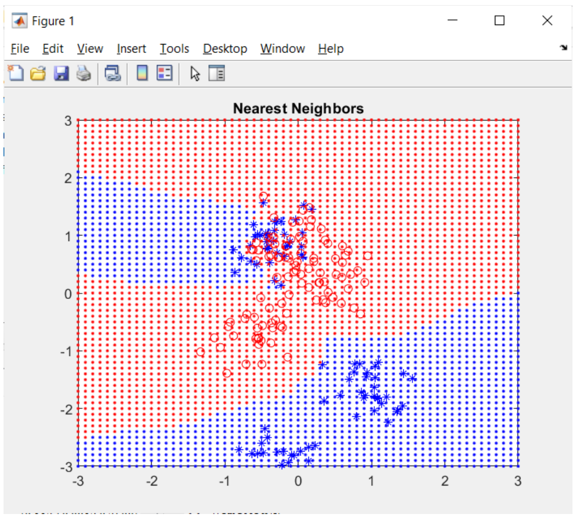
LDA, num_clusters = 1



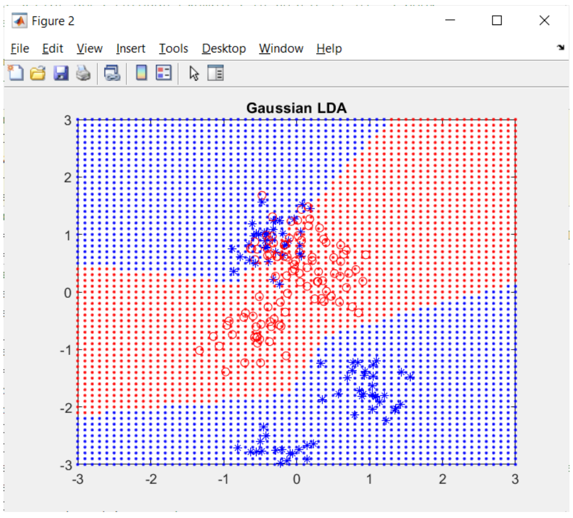
NN, num_clusters = 2



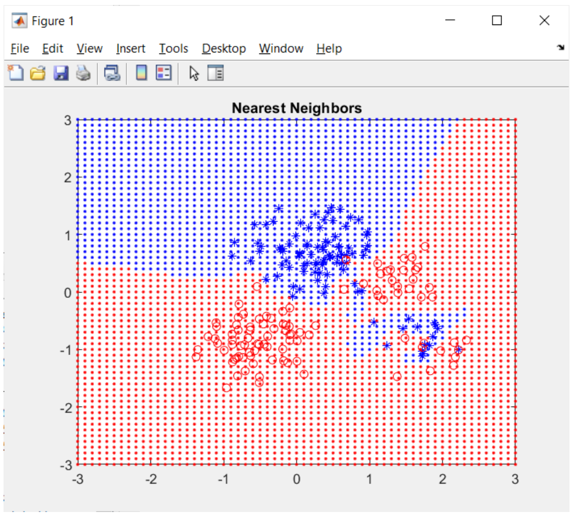
LDA, num_clusters = 2



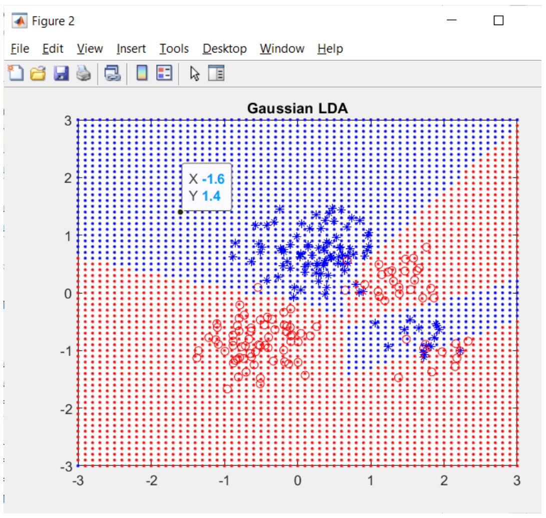
NN, num_clusters = 3



LDA, num_clusters = 3



NN, num_clusters = 5



LDA, num_clusters = 5

Comments: The main difference is that NN seems to sketch non-linear boundaries, while LDA tends to sketch more linear ones. Given that LDA assumes the same variance, the boundaries are a combination of lines. This difference is more appreciated when the number of clusters increases.

d. Sketch the QDA.

For this case is assumed all the considerations in section c) about the classes. But the P(Class|X) is recalculated with the condition that covariance matrix of gaussian probability has different variance values in the diagonal.

The P(Class|X) is given by the following equation:

$$P(Class|X) = e^{-\frac{1}{2}[(X-u_1)^T \Sigma_1^{-1}(X-u_1)]}$$

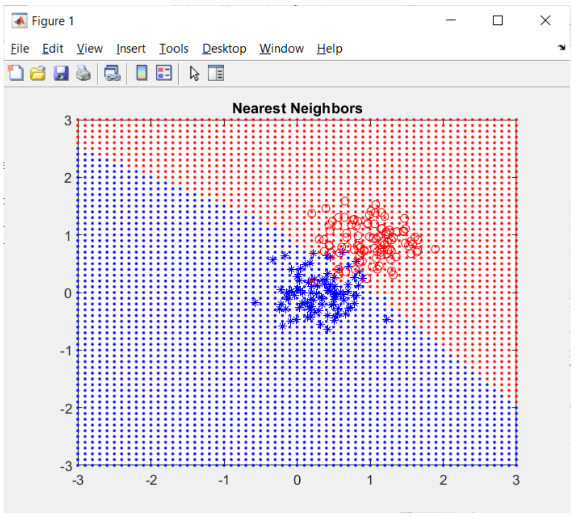
Solving the covariance matrix, and reducing:

$$P(Class|X) = e^{-\frac{1}{2}[P]}, where: P = \frac{(X_x - u_{1x})^2}{\sigma_x^2} + \frac{(X_y - u_{1y})^2}{\sigma_y^2}$$

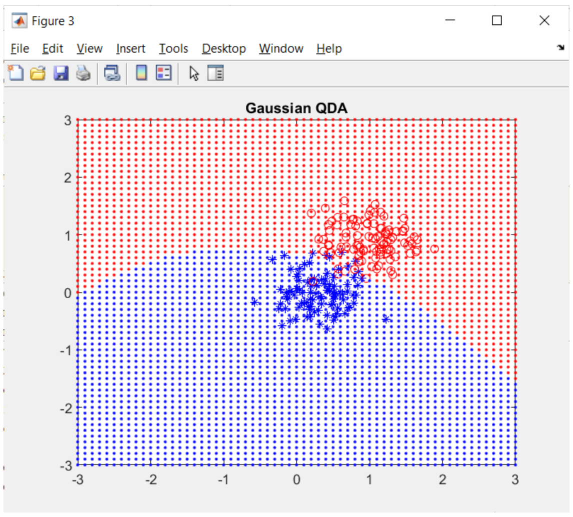
Also:

$$X = [X_x, X_y]$$
$$u_1 = [u_{1x}, u_{1y}]$$

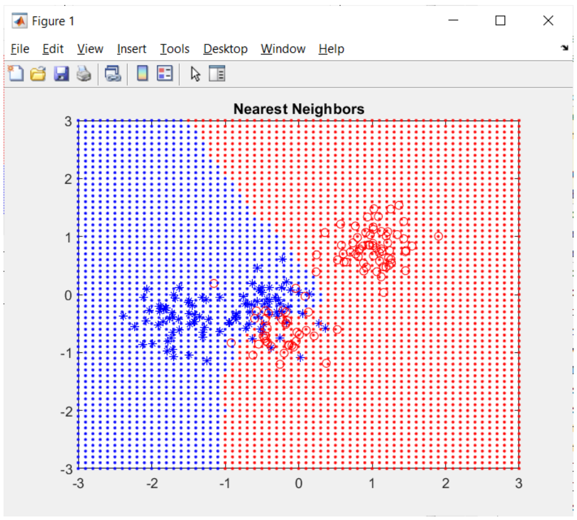
Applying the condition described, the following plot are obtained:



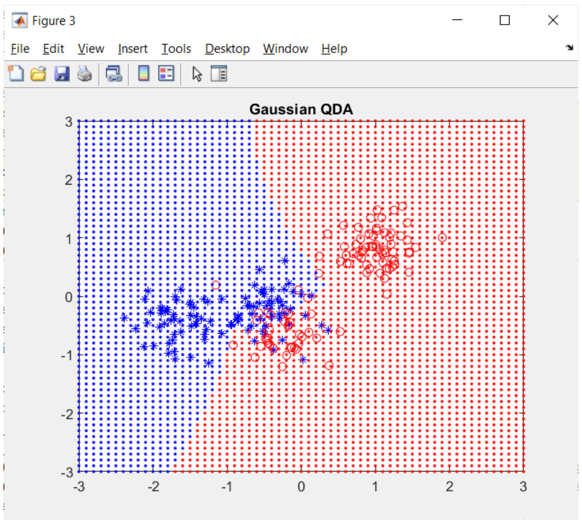
NN, num_clusters = 1



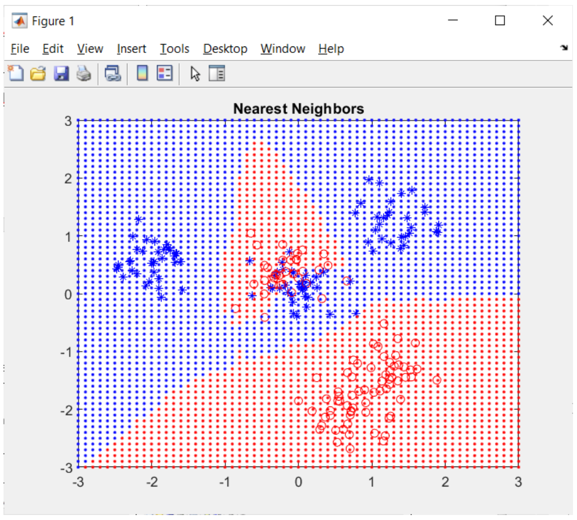
QDA, num_clusters = 1



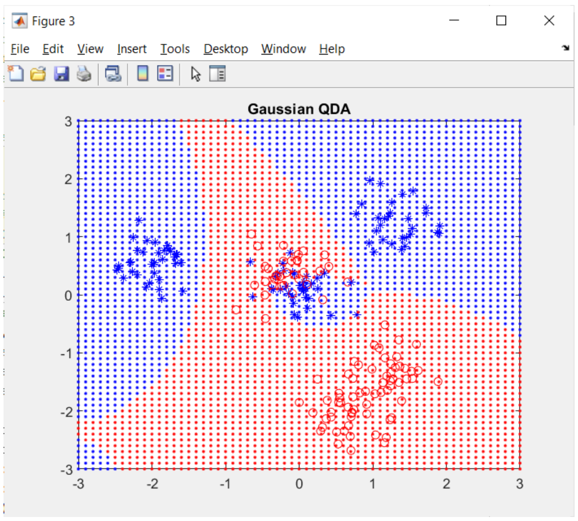
NN, num_clusters = 2



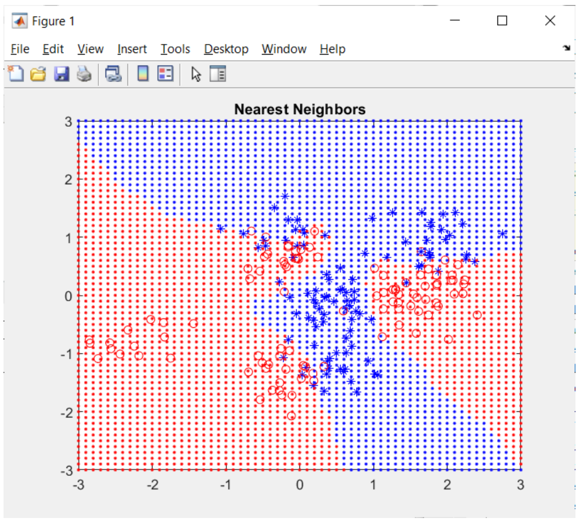
QDA, num_clusters = 2



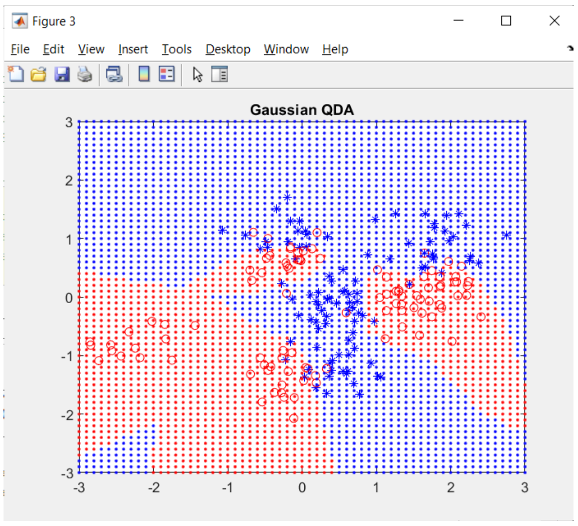
NN, num_clusters = 3



QDA, num_clusters = 3



NN, num_clusters = 5



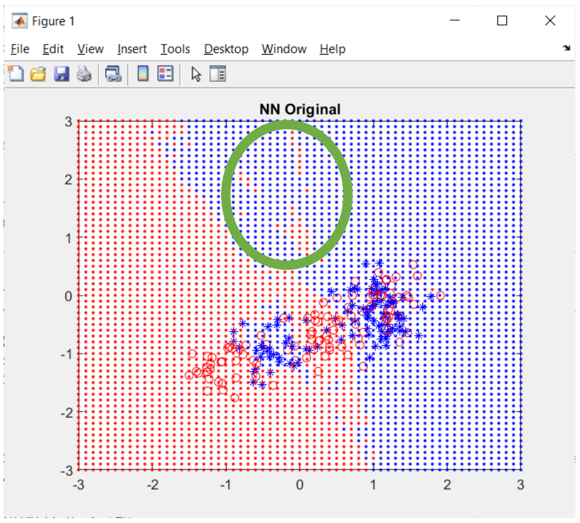
QDA, num_clusters = 5

Comments: With the QDA, the boundaries are softer, and they look more as a combination of quadratics functions, this approach is better handling non-linear boundaries compared to LDA approach.

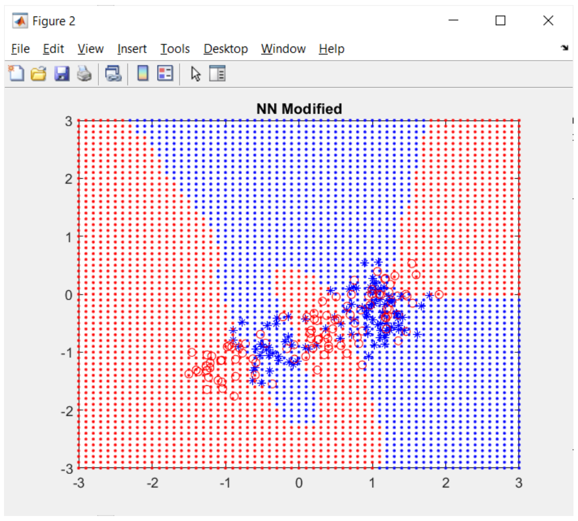
Problem 2

For the solution it has been modified the script ***"classification.m"***. For this new classification system is taken into consideration the distance of the neighbor and its weight. The formula used is proposed in problem statement. For classification, the sum of all weights for each the class is calculated, the class with greater sum is assigned to the region point.

All simulations are performed with 3 number of clusters. The results are presented below:



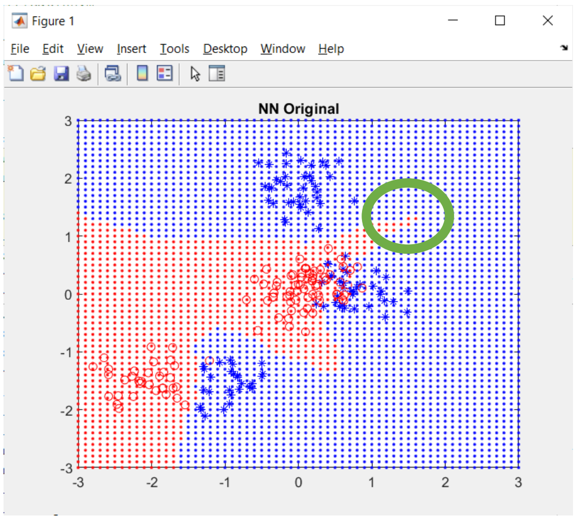
Original NN, neighbors = 50



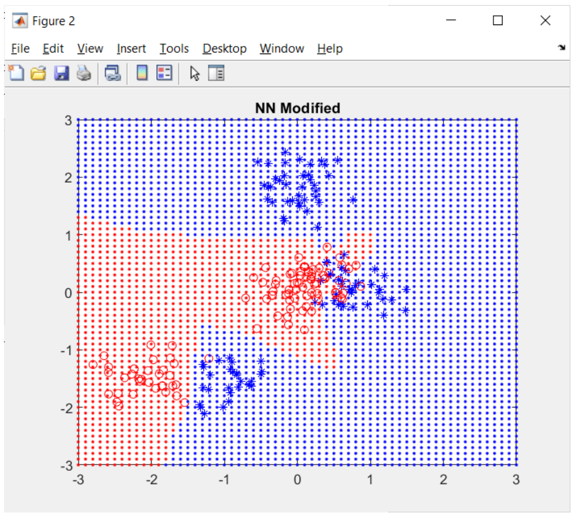
Modified NN, neighbors = 50

Comments: Given that the weight is inversely proportional to the distance, the nearest classes will have more importance than the farther ones. The original NN plot shows isolated red regions (green circle) because the number of red neighbors in its neighborhood is greater than blue ones. But the modified NN plot does not show those regions. Despite the number of red neighbors, the blue neighbors are near than red ones causing that weights in blue neighbors increase the probability of being blue instead of red for these regions.

It can be observed also in the simulations below:

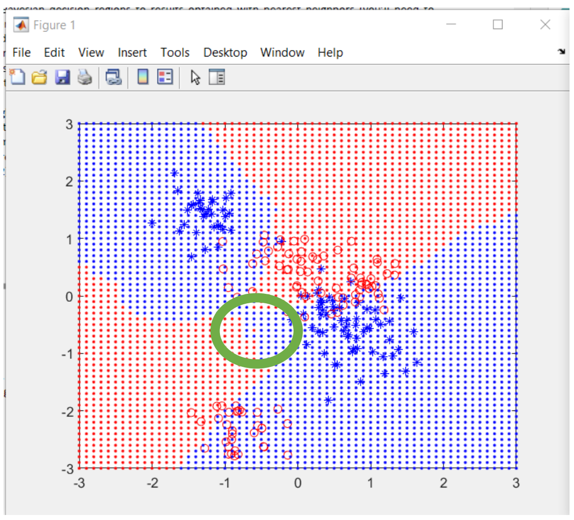


Original NN, neighbors = 10

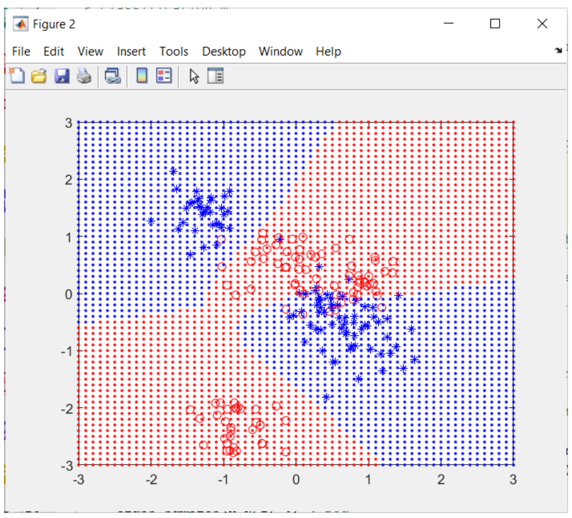


Modified NN, neighbors = 10

Comments: The region in green circle in original NN is shortened in modified NN.



Original NN, neighbors = 60



Modified NN, neighbors = 60

Comments: In original NN the red region is disconnected. In modified is connected. It is important to mention that this behavior is more observable when neighbors are more than 30.

Problem 1 Script

```
%-------
% classification.m
% This script goes through examples in Hastie et al.'s book.
% Binary classification is performed in two ways: linear regression
% and nearest neighbors
% Dror Baron, 8.30.2016
%-------
clear % often useful to clean up the work space from old variables
%% Parameters
num_clusters=5; % number of components (clusters) in mixture model
N=200; % total number of samples of training data
grid=-3:0.1:3; % test data grid for each dimension
num_neighbors=20; % number of neighbors used in nearest neighbors
%% Mixture models
Gmean=randn(2,num_clusters); % locations of centers of clusters for green class
Rmean=randn(2,num_clusters); % -"- red class
%% Training data
samples=zeros(2,N); % locations of samples in 2 dimensions
cluster_id=zeros(2,N); % Created to identify the cluster id
class_samples=zeros(N,1); % class of each one (green or red)
cluster_variance=0.1; % variance of each cluster around its mean
for n=1:N/2
    Gcluster=ceil(rand(1)*num_clusters); % select green cluster
    Rcluster=ceil(rand(1)*num_clusters); % -"- red
    cluster_id(:,n) = Gcluster; % Green ids from [1,num_clusters]
    cluster_id(:,n+N/2) = Rcluster; % Red ids from [1,num_clusters]
    samples(:,n)=Gmean(:,Gcluster)+sqrt(cluster_variance)*randn(2,1); % generate
green sample
    samples(:,n+N/2)=Rmean(:,Rcluster)+sqrt(cluster_variance)*randn(2,1); % -"- red
    class_samples(n)=1; % green
    class_samples(n+N/2)=0; % red
end
%% Test data - basically a 2-D grid
test_samples=zeros(2,length(grid)^2); % locations of test samples
for n1=1:length(grid)
    for n2=1:length(grid)
        test_samples(1,n1+length(grid)*(n2-1))=grid(n1); % first coordinate
        test_samples(2,n1+length(grid)*(n2-1))=grid(n2); % second
    end
end
%% Nearest Neighbors Classifier
test_NN=zeros(length(grid)^2,1); % classification results on test data
for n1=1:length(grid)
    for n2=1:length(grid)
        distances=(grid(n1)-samples(1,:)).^2+(grid(n2)-samples(2,:)).^2; %
distances to training samples
        [distances_sort,distances_index]=sort(distances);
        neighbors=distances_index(1:num_neighbors);
        class_predicted=(sum(class_samples(neighbors))/num_neighbors>0.5); % NN
classifier
        test_NN(n1+length(grid)*(n2-1))=class_predicted; % store classification
    end
end
%% PLOT NN
% identify location indices (in test grid) that are red and green
r_locations=find(test_NN==0);
g_locations=find(test_NN==1);
% NN plot
    figure(1),plot(samples(1,1:N/2),samples(2,1:N/2),'b*',... % green training
samples
    samples(1,N/2+1:N),samples(2,N/2+1:N),'ro',... % red training
    test_samples(1,g_locations),test_samples(2,g_locations),'b.',... % green test
    test_samples(1,r_locations),test_samples(2,r_locations),'r.'),... % red
title('Nearest Neighbors')
axis([-3 3 -3 3]); % boundaries for figure aligned with grid
%% LDA (Section to solve 1c)
```

```
G_samples = samples(:,1:100);
R_samples = samples(:,101:200);
G_id = cluster_id(:,1:100);
R_id = cluster_id(:,101:200);

% Calculating Mean of the Classes
mean_G = zeros(2,num_clusters);
mean_R = zeros(2,num_clusters);
std_G = zeros(1,num_clusters);
std_R = zeros(1,num_clusters);

% Get means and variances from data
for i = 1:num_clusters
    meanG = G_samples(G_id==i);
    mean_G(:,i) = mean(reshape(meanG,[2,length(meanG)/2]),2);
    meanR = R_samples(R_id==i);
    mean_R(:,i) = mean(reshape(meanR,[2,length(meanR)/2]),2);
    stdG = G_samples(G_id==i);
    std_G_aux = var(reshape(stdG,[2,length(stdG)/2]),0,2);
    std_G(:,i) = mean(std_G_aux); % It is assumed that variance is the same for
both dimensions
    stdR = R_samples(R_id==i);
    std_R_aux = var(reshape(stdR,[2,length(stdR)/2]),0,2);
    std_R(:,i) = mean(std_R_aux); % It is assumed that variance is the same for
both dimensions
end

% Class calculation in Test Data
Means = [mean_G,mean_R]; % New means calculated from data
Vars = mean([std_G,std_R]); % New variances calculated from data (LDA assumes same
variance for all clusters)
test_LDA=zeros(length(grid)^2,1); % classification results on test data
for n1=1:length(grid)
    x1 = grid(n1);
    for n2=1:length(grid)
        x2 = grid(n2);
        p_x = exp(-0.5*(sum(([x1;x2]-Means).^2)./(Vars.^2))); % P(Class|X)
        class = p_x/sum(p_x); % Normalized P(Class|X)
        [max_value,class_predicted] = max(class); % Maximun[P(Class|X)] predicts
class.
        test_LDA(n1+length(grid)*(n2-1))=class_predicted; % store classification
    end
end
%% PLOT LDA
% identify location indices (in test grid) that are red and green
r_locations=find(test_LDA>=num_clusters+1); % Subclasses
[num_clusters+1,2*num_clusters]
g_locations=find(test_LDA<=num_clusters); % Subclasses [1,num_clusters]
% LDA plot
    figure(2),plot(samples(1,1:N/2),samples(2,1:N/2),'b*',... % green training
samples
    samples(1,N/2+1:N),samples(2,N/2+1:N),'ro',... % red training
    test_samples(1,g_locations),test_samples(2,g_locations),'b.',... % green test
    test_samples(1,r_locations),test_samples(2,r_locations),'r.'),... % red
title('Gaussian LDA')
axis([-3 3 -3 3]); % boundaries for figure aligned with grid
%% QDA (Section to solve 1d)
G_samples = samples(:,1:100);
R_samples = samples(:,101:200);
G_id = cluster_id(:,1:100);
R_id = cluster_id(:,101:200);

% Calculating Mean of the Classes
mean_G = zeros(2,num_clusters);
mean_R = zeros(2,num_clusters);
std_G = zeros(2,num_clusters);
std_R = zeros(2,num_clusters);
```

```
% Get means and variances from data
for i = 1:num_clusters
    meanG = G_samples(G_id==i);
    mean_G(:,i) = mean(reshape(meanG,[2,length(meanG)/2]),2);
    meanR = R_samples(R_id==i);
    mean_R(:,i) = mean(reshape(meanR,[2,length(meanR)/2]),2);
    stdG = G_samples(G_id==i);
    std_G(:,i) = var(reshape(stdG,[2,length(stdG)/2]),0,2);
    stdR = R_samples(R_id==i);
    std_R(:,i) = var(reshape(stdR,[2,length(stdR)/2]),0,2);
end

% Class calculation in Test Data
Means = [mean_G,mean_R]; % New means calculated from data
Vars = [std_G,std_R]; % New variances calculated from data
test_QDA=zeros(length(grid)^2,1); % classification results on test data
for n1=1:length(grid)
    x1 = grid(n1);
    for n2=1:length(grid)
        x2 = grid(n2);
        r = ([x1;x2]-Means).^2;
        px = r(1,:)./(Vars(1,:).^2);
        py = r(2,:)./(Vars(2,:).^2);
        p_x = exp(-0.5*(px+py)); % P(Class|X)
        class = p_x/sum(p_x); % Normalized P(Class|X)
        [max_value,class_predicted] = max(class); % Maximun[P(Class|X)] predicts
class.
        test_QDA(n1+length(grid)*(n2-1))=class_predicted; % store classification
    end
end
%% PLOT QDA
% identify location indices (in test grid) that are red and green
r_locations=find(test_QDA>=num_clusters+1);
g_locations=find(test_QDA<=num_clusters);
% LDA plot
    figure(3),plot(samples(1,1:N/2),samples(2,1:N/2),'b*',... % green training
samples
    samples(1,N/2+1:N),samples(2,N/2+1:N),'ro',... % red training
    test_samples(1,g_locations),test_samples(2,g_locations),'b.',... % green test
    test_samples(1,r_locations),test_samples(2,r_locations),'r.'),... % red
title('Gaussian QDA')
axis([-3 3 -3 3]); % boundaries for figure aligned with grid
```

Problem 2 Script

```
%-------
% classification.m
% This script goes through examples in Hastie et al.'s book.
% Binary classification is performed in two ways: linear regression
% and nearest neighbors
% Dror Baron, 8.30.2016
%-------

clear % often useful to clean up the work space from old variables
%% Parameters
num_clusters=3; % number of components (clusters) in mixture model
N=200; % total number of samples of training data
grid=-3:0.1:3; % test data grid for each dimension
num_neighbors=10; % number of neighbors used in nearest neighbors
%% Mixture Models
Gmean=randn(2,num_clusters); % locations of centers of clusters for green class
Rmean=randn(2,num_clusters); % -"- red class
%% Training data
samples=zeros(2,N); % locations of samples in 2 dimensions
```

```
class_samples=zeros(N,1); % class of each one (green or red)
cluster_variance=0.1; % variance of each cluster around its mean
for n=1:N/2
    Gcluster=ceil(rand(1)*num_clusters); % select green cluster
    Rcluster=ceil(rand(1)*num_clusters); % -"- red
    samples(:,n)=Gmean(:,Gcluster)+sqrt(cluster_variance)*randn(2,1); % generate
green sample
    samples(:,n+N/2)=Rmean(:,Rcluster)+sqrt(cluster_variance)*randn(2,1); % -"- red
    class_samples(n)=1; % green
    class_samples(n+N/2)=0; % red
end
%% Test Data - basically a 2-D grid
test_samples=zeros(2,length(grid)^2); % locations of test samples
for n1=1:length(grid)
    for n2=1:length(grid)
        test_samples(1,n1+length(grid)*(n2-1))=grid(n1); % first coordinate
        test_samples(2,n1+length(grid)*(n2-1))=grid(n2); % second
    end
end
%% Nearest Neighbors (Clasical Approach)
test_NN=zeros(length(grid)^2,1); % classification results on test data
for n1=1:length(grid)
    for n2=1:length(grid)
        distances=(grid(n1)-samples(1,:)).^2+(grid(n2)-samples(2,:)).^2; %
distances to training samples
        [distances_sort,distances_index]=sort(distances);
        neighbors=distances_index(1:num_neighbors);
        class_predicted=(sum(class_samples(neighbors))/num_neighbors>0.5); % NN
classifier: 0 = RED, 1 = BLUE
        test_NN(n1+length(grid)*(n2-1))=class_predicted; % store classification
    end
end
%% PLOT
% identify location indices (in test grid) that are red and green
r_locations=find(test_NN==0);
g_locations=find(test_NN==1);
% NN plot
figure(1),plot(samples(1,1:N/2),samples(2,1:N/2),'b*',... % green training samples
    samples(1,N/2+1:N),samples(2,N/2+1:N),'ro',... % red training
    test_samples(1,g_locations),test_samples(2,g_locations),'b.',... % green test
    test_samples(1,r_locations),test_samples(2,r_locations),'r.') % red
title('NN Original')
axis([-3 3 -3 3]); % boundaries for figure aligned with grid
%% Nearest Neighbors (Problem 2) - MODIFICATION
test_NN_2=zeros(length(grid)^2,1); % classification results on test data
for n1=1:length(grid)
    for n2=1:length(grid)
        distances=(grid(n1)-samples(1,:)).^2+(grid(n2)-samples(2,:)).^2; %
distances to training samples
        [distances_sort,distances_index]=sort(distances);
        neighbors=distances_index(1:num_neighbors);
        NN_distances = 2*(num_neighbors:-1:1)./((num_neighbors)*(num_neighbors+1));
        Rsum = sum(NN_distances(class_samples(neighbors)==0)); % Sum K distances of
all neighbors in Red Class
        Gsum = sum(NN_distances(class_samples(neighbors)==1)); % Sum K distances of
all neighbors in Green Class
        class_predicted=Gsum>Rsum; % NN classifier: 0 = RED, 1 = BLUE
        test_NN_2(n1+length(grid)*(n2-1))=class_predicted; % store classification
    end
end
%% PLOT NN Modified
% identify location indices (in test grid) that are red and green
r_locations=find(test_NN_2==0);
g_locations=find(test_NN_2==1);
% NN plot
figure(2),plot(samples(1,1:N/2),samples(2,1:N/2),'b*',... % green training samples
    samples(1,N/2+1:N),samples(2,N/2+1:N),'ro',... % red training
    test_samples(1,g_locations),test_samples(2,g_locations),'b.',... % green test
    test_samples(1,r_locations),test_samples(2,r_locations),'r.') % red
title('NN Modified')
axis([-3 3 -3 3]); % boundaries for figure aligned with grid
```