ECE 8443: Pattern Recognition – Project 1

Minimum Risk Bayes Decision Theoretic Classifier

Ruperto Solis – Mississippi State University

Abstract — Design of a generic classification algorithm using Minimum Risk Bayes Decision theorem and testing its performance calculating the Error Estimates, based on Confusion Matrices through the re-substitution and cross validation on three different datasets.

I. INTRODUCTION

Data classification is one of the main subjects in Pattern Recognition; It consists on developing algorithms and tools capable of differentiate objects with D number of features, known as patterns, and classify them in different groups known as classes, based on behavior of the analyzed features on that particular object. One example of classification is to identify apples as class 1 and oranges as class 2 based on their shape and color.

There are many techniques and algorithms in the portfolio of Pattern Recognition, each one with specific advantages and disadvantages depending on the nature of the problem to solve, the data to classify, the resources, and the limitations of the particular approach. It is responsibility of the Data Scientist to determine with model is the more appropriate to use for the given application.

Minimum Risk Bayes Decision Theoretic Classifier is one of the most essential models in Pattern Recognition, which provides and optimal decision machine exploiting the statistical and probabilistic nature of the features extracted. The purpose of this project is to create a generic classifier using Bayes Decision Theory and Minimum Risk Analysis using MATLab.

II. TECHNICAL DESCRIPTION

SUPERVISED TRAINING

The general logic behind Classification under supervised training, is to select N number of samples, known as Training Data, that were already identified as part of specific classes and extract D number of features from them, which will help us to obtain the probabilistic components required to build a classification machine capable of identifying a future new sample and classify it in one of the previously defined classes by analyzing its features.

TOTAL PROBABILITY

Considering $\boldsymbol{\omega}$ as all the different M Classes identified in the universe of N patterns (samples); Each Class is defined as $\omega i = 1, 2, \dots, M$.

$P(\omega_i)$ is the probability of finding a sample identified as Class ω_i. The summation of all the different probabilities:

$$\sum_{i=1}^{M} \mathrm{P}(\omega_i) = 1$$

THE BAYES RULE

Per the Bayes rule; the probability of a random new pattern B with $\vec{X}$ set of features, known as posteriori probability, belonging to class Ai is defined by:

$$P\left(\omega_i \middle| \vec{X}\right) = \frac{P\left(\vec{X} \middle| \omega_i\right) P(\omega_i)}{P\left(\vec{X}\right)} \qquad Eq.\,1.$$

Where

$P\left(\vec{X} \middle| \omega_i\right)$, known as *Likelihood*, is the probability of finding B pattern with $\vec{X}$ set of features in probability distribution of class ω_i.

$P(\omega_i)$ as previously defined, it is the probability of finding a pattern of class ω_i in the Training Data.

$P(\vec{X})$ is the overall probability of finding a pattern with $\vec{X}$ features in the Training Data regardless of its class.

Once we have all the necessary components, we can calculate the posterior probability $P(\vec{X}|\omega_i)$ for each of the M classes and by simply comparing them, it will be apparent to realize what class is more probable to contain pattern B.

CALCULATING BAYES COMPONENTS

All the components are calculated using the Training data which contains N number of samples, each with D number of features and individually identified as part of 1 class in A.

A. Calculate $P(\omega_i)$

$\mathrm{N}(\omega_i) =$ total Number of samples classified as ω_i.

$$P(\omega_i) = \frac{\mathrm{N}(\omega_i)}{N} \qquad Eq.2.$$

B. Calculate $P(\vec{X}|\omega_i)$

To be able to calculate it, first we need to generate a PDF (Probability Density Function) for Ai using the information from those patterns with features $\vec{x}$ belonging to it. We will fall under the assumption that class ω_i has a normal distribution with respect to the D features of the patterns. Since the number of features (Dimensions) is unknown we need to use the Multivariate Normal Distribution:

$$\boldsymbol{P}(\vec{X}|\omega_i) = \qquad \text{Eq.3}$$

$$N(\vec{X}, \vec{\mu}, \Sigma) = \frac{1}{(2\pi)^{\frac{D}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\vec{X}-\vec{\mu})^T \Sigma^{-1} (\vec{X}-\vec{\mu})}$$

Where:

$\vec{\mathbf{X}}$ is the array of features of the new patter B to be evaluated.

$\vec{\boldsymbol{\mu}}$ is the array of Arithmetic Means for class ω_i of all the features, calculated using the training Data: $\vec{\mu} = [\mu_1 \ \mu_2 \ \ldots \ \mu_D]$.

$$\vec{\boldsymbol{\mu}} = \frac{1}{N} \sum_{k=1}^{N} \overrightarrow{X_k} \qquad Eq.4.$$

$\boldsymbol{\Sigma}$ Is the Covariance Matrix of DxD containing the covariance information of the standard deviation σ_i of each feature $i \in 1..D$ in relation to the standard deviation σ_j of all features.

$$\Sigma = \begin{bmatrix} {\sigma_i}^2 & \cdots & \sigma_{ji} \\ \vdots & \ddots & \vdots \\ \sigma_{ij} & \cdots & {\sigma_D}^2 \end{bmatrix}$$

Σ is computed as follows:

$$\Sigma = \frac{1}{N} \sum_{k=1}^{N} (\overrightarrow{X_k} - \vec{\mu})(\overrightarrow{X_k} - \vec{\mu})^T \qquad Eq.5.$$

$\mathrm{N} =$ Total Number of samples.

$\overrightarrow{X_k} =$ Set of features of each sample in the training data.

C. Calculate $P(\vec{X})$

It is done through the Marginalization of all the classes in the likelihoods $P(\vec{X}|\omega_i)$.

$$P(\vec{X}) = \sum_{i=1}^{M} P(\vec{X}|\omega_i) \qquad Eq.6.$$

Where M is the total number of classes in ω.

MINIMUM RISK DECISION

For several applications, knowing what's the highest classification probability is enough to take a decision. Such is the case of classifying apples and oranges when there's no particular preference when it comes to confusing an orange for an apple and vice versa.

However, there are applications where confusing a sample of Class ω_i as a pattern of Class ω_j can have catastrophic consequences in comparison with confusing a sample of class ω_j as a pattern of class ω_iA very common example is the analysis of an algorithm designed to diagnose the nature of a tumor based on D number of symptoms. Confusing a benign tumor with a cancerous one and doing the opposite have very different implications.

We can compensate this criticality by multiplying the posteriori probability obtained from Bayes rule by a weight factor λ known as loss. If we define action $\boldsymbol{\alpha_i}$ as the action of classifying pattern $\vec{X}$ as class ω_i then we can define $\boldsymbol{\lambda}\ (\boldsymbol{a_i}|\boldsymbol{\omega_j})$ as the loss of classifying pattern $\vec{X}$ as ω_i when it belongs to class ω_j.

The nature of the Minimum Risk Decision lies in the combination of the Bayes Rule with the loss values allowing us to determine what is the risk of classifying incorrectly a pattern as class ω_i for every class and taking a decision based on what risk is the minimum.

The first step to calculate the minimum risk decision is to consider all possible cases of $\lambda\ (a_i|\omega_j)$ which can be represented by the Loss Matrix.

A. Loss Matrix

$$\begin{bmatrix} \lambda\ (a_1|\omega_1) & \cdots & \lambda\ (a_1|\omega_M) \\ \vdots & \ddots & \vdots \\ \lambda\ (a_M|\omega_1) & \cdots & \lambda\ (a_M|\omega_M) \end{bmatrix}$$

The values of $\lambda\ (a_i|\omega_j)$ will widely depend on the application and a combination of factors such as experience on the field of interest and the limitations stablished for the classifier. The only values that are known for sure are the losses on the main diagonal which are always zero, since there is no risk on taking the correct decision.

$$\begin{bmatrix} 0 & \cdots & \lambda\ (a_1|\omega_M) \\ \vdots & \ddots & \vdots \\ \lambda\ (a_M|\omega_1) & \cdots & 0 \end{bmatrix}$$

$\lambda\ (a_i|\omega_j)$ represents the loss, hence whenever it is riskier to perform action a_i given ω_j, the value will be higher providing a higher loss whenever that combination occurs.

B. Risk Calculation

The risk of taking action a_i given pattern $\vec{X}$ is defined as the summation of: the loss of incorrectly classifying the pattern as class ω_i given that it is actually ω_j multiplied by the possibility of pattern $\vec{X}$ belonging to class ω_j, for all the M classes:

$$R\left(a_i\middle|\vec{X}\right) = \sum_{j=1}^{M} \lambda\ (a_i|\omega_j)\, P\left(\omega_j\middle|\vec{X}\right) \quad Eq.7.$$

C. Taking the Minimum risk decisions

The idea is simply to apply the previous formula to all different decisions a_i and take the action that generates the minimum value.

PERFORMANCE

The only way of knowing how good our classifier is doing its job is by testing it against data we already know its correct classification. Two methods that were used during the testing phase of this experiment are the re-substitution testing and the cross validation:

A. Re-substitution

It is done simply by using all the training data for testing, classifying every single one of the patterns initially used for the design of the classifier and compare the classification results against the real class of each sample.

B. Cross Validation

This method consists on dividing the training dataset in 2 groups, one for training data and one for testing data. On this project 90% of the data was allocated for training and 10% for testing. Cross validation involves creating several classifiers using different samples for testing on each one and the rest of the data is used for training. On this document, 10 different classifiers were created using all the 10% possibilities of test data available, the entire dataset was divided in 10 groups and each group was used for the testing of each classifier while the rest of the data was used for training.

Fig. 1. All 10 combinations for cross validation, red represents test data and blue represents training data.

Once the testing data has been classified, the performance of the classifier can be analyzed by using a tool known as Confusion Matrix which contains all the combinations of machine classification vs. actual class where all the elements in the main diagonal are correct classification and every other element in the matrix represent the errors ω_{ij}; Classifying pattern $\vec{X}$ as ω_i when it was actually ω_j:

C. Confusion Matrix

$$\text{Actual Class} \qquad \text{Classified as} \begin{bmatrix} \omega_{11} & \cdots & \omega_{1M} \\ \vdots & \ddots & \vdots \\ \omega_{M1} & \cdots & \omega_{MM} \end{bmatrix}$$

Using the confusion matrix, we can determine the number Average Classification Rates:

D. Correct Classification Rate

Number of samples correctly classified as ω_i over the total samples that are actually ω_i.

$$CCR_i = \frac{\omega_{ii}}{\sum_{j=1}^{M} \omega_{ji}} \qquad Eq.\,8.$$

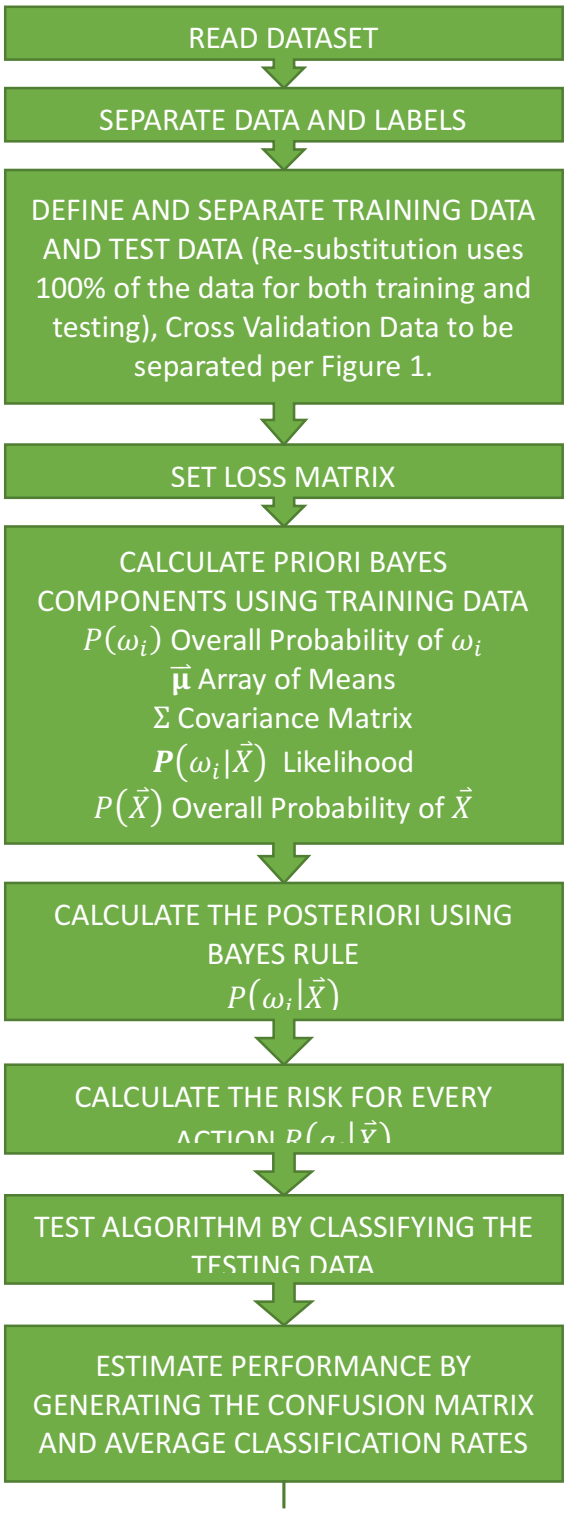
E. False Alarm Rate

Number of samples incorrectly classified as ω_i over the total number of times a sample was classified as ω_i.

$$FAR_i = \frac{\sum_{j=1}^{M} \omega_{ij} \;\; \forall\, j \neq i}{\sum_{i=1}^{M} \omega_{ij}} \qquad Eq.\,9.$$

III. DESIGN OF THE ALGORITHMS

The following pipeline describes the sequential for design, test and performance estimate of the Minimum Risk Bayes Decision Theretic Classifier:



All the Algorithms are Pseudocode

READ DATASET AND SEPARATE LABELS

Common datasets are expect to be formatted with the following restrictions:

a. No headers in the data.
b. Every row represents a sample.
c. Every column represents a feature.
d. One of the columns represent the label of the class where the sample belongs.
e. Columns are separated by 1 or a series of characters in a consistent fashion.

The following Algorithm describes the logic behind the extraction of the samples in the dataset:

Algorithm 3.1: Reading dataset

```
function [Data,Labels] =
readDataset(fileLocation, labelColumn)
   text=fileread(fileLocation)
   %Separate columns by commas and
   rows by '|'
   ColSep=identColSeparator(text)
   text=replace(text,ColSep,',')
   text=replace(text,chr(10),'|')
   %Get number of rows and columns
   rows=len(text)-
        len(replace(text,'|',''')) + 1
   cols=1
   while (text(i:i) ~= '|')
       i=i+1;
       if (text(i:i)==',')
          cols=cols+1;
       end
   end
   %Get data and labels
   Data=zeros(rows,cols-1)
   Lables=zeros(rows,1)
   Samples=split(text,'|')
   Xindex=1;Yindex=1
   For each sample in Samples
      Features=split(sample,',')
      For each feature in Features
         If Yindex!= labelColumn
          Data(Xindex,Yindex)=feature
          Yindex++
         Else
          Labels(Xindex,1)=feature
         end
         Xindex++
      end
   end
end
```

DEFINE TRAINING AND TEST DATA

For the regular classifying machine and the Re-substitution example, selecting the training data is as simple as using the entire Data matrix from *Algorithm 3.1*. The Re-substitution will define the testing Data as the same matrix as training data.

Algorithm 3.2(a): re-substitution Data

```
testData=Data
trainingData=Data
trainingLbls=Labels
testLbls=Labels
```

In the other hand cross validation requires 10 different sets of test and training data where the test data is always 10% of the entire Dataset. Wenever the data cannot be evenly divided by 10, some of the cluster will have to be 1 sample longer than the rest.

Algorithm 3.2(b): Cross Validation Data

```
totParts=10
%all parts will have at least
%this amount of samples:
min=floor(totSamples/totParts)
%Parts 1 to extra will have an
extra sample
extra= mod(totSamples/totParts)
lastEnd=0;
for i=1: totParts
   newEnd=lastEnd+min+(i<=extra)
   indexes=(lastEnd+1:newEnd);
   %Array with the 10 data parts
   sData(i)=Data(indexes,:)
   sLabels(i)=Labels(indexes,1)
   %Update index of last sample
   lastEnd=newEnd
end
%Each of the 10 classifiers in
Cross Validation
for i=1: totParts
   testData=sData(i)
   testLbls=sLabels(i)
   trainingData=empty
   trainingLbls=empty
   for j=1: totParts
     if i!=j
      trainingData=(trainingData;
         sData(i))
      trainingLbls=(triningLbls;
         sLabels(i))
     end
   end
end
```

SET LOSS MATRIX

On this project we will assume there is no preference on misclassifications hence all the losses are set to 1 except the correct classifications that have no loss:

$$\begin{bmatrix} 0 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 0 \end{bmatrix}$$

Algorithm 3.3: lossMatrix

```
Loss=1-IdentMatrix(totClasses)
```

CALCULATE LIKELIHOOD $P(\vec{X}|\omega_i)$

Per *Equation 3*, in order to calculate the likelihood of a sample to have certain features, give it is clas ω_i first we need to calculate the Aray of means $\vec{\mu}$ and the covariance matrix Σ for a given class.

Algorithm 3.4: Likelihood

```
%currentClass contains the label of
the class to be evaluated
totTrSamp=size(trainingData,1)
features=size(trainingData,2)
classData=trainingData(Find(
   trainingLbls==currentClass),:)
classLbls=trainingLbls(Find(
   trainingLbls==currentClass),:)
classSmpls=size(classLbls,1)
%Refer to Equation 4
μ⃗=sum(classData)/classSmpls
%Refer to Equation 5
Σ=zeros(features,features)
for i=1:classSmpls
    temp2=classData(i)-μ⃗
    temp1=transpose(temp2)
    Σ= Σ+(temp1*temp2)
End
Σ= Σ/classSmpls
%Refer to Equation 3
%Patter is the new sample from the
test Data that we want to classify
D=features
Σdet=det(Σ)%Determinant of covariance
Σinv=inv(Σ)%Inverse of covariance
temp1=Pattern-μ⃗
```

```
temp2=transpose(temp1)
temp3=1/(((2PI)^(D/2))*sqrt(Σdet))
temp4=0.5*(temp1* Σinv*temp2)
Likelihood=temp3*exp(-temp4)
```

CALCULATE $P(\omega_i)$

It is the overall probability of finding a sample of class ω_i in the training Data, refer to *equation 2* for its calculation

Algorithm 3.5: Class Probability

```
%Total samples in the training data
totTrSamp=size(trainingData,1)
%Total samples in the class Data
%Refer to Algorithm3.4 for classLbls
classSmpls=size(classLbls,1)
classProb=classSmpls/totTrSamp
```

CALCULATE $P(\vec{X})$

It is the overall probability of finding a pattern $\vec{X}$ in any of the classes probability density functions, where $\vec{X}$ is the new Pattern to be classified from the test Data. Refer to *equation 6.*

Algorithm 3.6: Pattern Probability

```
%Likelihoods is an array with M
items, one for each of the
Likelihoods of each of the classes
in training data with respect to X⃗
P(X⃗)=0
 For each Likelihood in Likelihoods
    P(X⃗)+=Likelihood
 end
```

CLASSIFY TEST SAMPLES

If we create an array of $P(\omega_i)$, 1item for each of the classes using Algorithm 3.5, an array of Likelihoods, 1 item for each of the test samples using Algorithm 3.4, and 1 array of $P(\vec{X})$, 1 item for each of the test samples using Algorithm 3.6, then we can use the following algorithm to calculate the posteriori Bayes Rule for each of the combination of class|Test Sample $P(\omega_i|\vec{X})$ based on equation1:

```
%Likelihoods:Array of likelihoods
%ClassProb: Array of P(ω_i)
%Xprob: Array of P(X⃗)
```

Algorithm 3.7: Bayes Rule

```
% From Algorithm 3.4 we calculated the
total number of classes classSmpls
testSamp=size(testData,1)
%BAYES RULE
Pwx=zeros(testSamp,classSmpls)
For i=1:testSamp
   Pxw=Likelihoods(i)
   Px=Xprob(i)
   For j=1:classSmpls
      Pw=ClassProb(j)
      Pwx(i,j)=Pxw*Pw/Px
   end
end
```

Once we have all the posterior probabilities for all the test data, the only thing we need to do is to calculate the risk for selecting each of the classes as the owner of each of the samples in the test data and select the one with the less risk.

Algorithm 3.8: Classify

```
%From Algorith 3.3 we have Loss as the
Loss Matrix

%UniqueClass is the array of labels for
each class, 1 for class

Classifications=empty(testSamp,1)
For i=1:testSamp
   For j=2:classSmpls
      If j=1
        MinRiskIndex=1
        curRisk=Loss(j,:).*Pwx(i,:)
      else
        lastRisk=curRisk
        curRisk=Loss(j,:).*Pwx(i,:)
        if curRisk<lastRisk
            MinRiskIndex=j
        end
      end
   end
   Classification(i)=UniqueClass(
      MinRiskIndex)
end
```

PERFORMANCE

The first step to estimate the performance of the classifier is to generate the confusion Matrix using the results from *Algorithm3.8* then using its information we can calculate the CCR (Correct Classification Rate) and the FAR (False Alarm Rate) Using *equations 8 and 9.*

Algorithm 3.9: Performance

```
%CONFUSION MATRIX
Confusion=zeros(classSmpls)
For i=1:testSamp
  %Returns the index of the class
  where test sample i was classified
  cIndex=find(uniqueClass==
      Classification(i))
  %Returns the index of the actual
  class sample i belongs to
  rIndex= find(uniqueClass==
      testLbls(i))
  Confusion(cIndex,Index)+=1
end
%AVERAGE CLASSIFICATION RATES
    sumaCC=0;
    sumaFA=0;
    for i=1:classSmpls
        %For Each Class
        CorrectC=Confusion(i,i)
        ToC=sum(ConfusionMatrix(:,i))
        CAsClass=sum(Confusion(i,:))
        CIncorrect=CAsClass-correctC
        sumac+=(correctC/ToC);
        sumaFA+=(CIncorrect/CAsClass);
    end
    ACCR=sumaCC/classSmpls;
    AFAR=sumaFA/classSmpls;
```

IV. ANALYSIS OF RESULTS

To test the Minimum Risk Bayes Decision Theorim, an Algorithm was design in Matlab using the logic from the Algorithms in Section III, and was subjected to three different datasets obtaining the following results:

TWO CLASS FOUR GAUSSIANS DATSET

This dataset contains a total of 10,000 samples of two different classes and four features. *Figures 2 and 3* provide a visualization in a two dimensional cartesian space of the correlation between the features

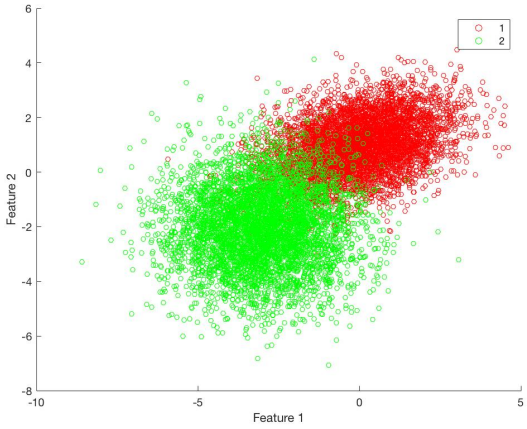


Fig. 2. Features 1 and 2 from the Two Class Four Gaussians Dataset.

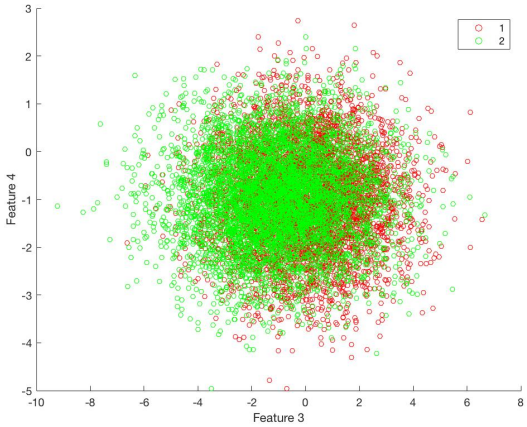


Fig. 3. Features 3 and 4 from the Two Class Four Gaussians Dataset.

As we can see on the 2 previous figures, it is evident that Features 1 and 2 although not very correlated, have such combination that allow us to easily identify what elements belong to each of the classes. In the other hand classes 3 and 4 are neither very correlated nor have a combination that can help us to separate the classes, if we were only to use these 2 classes, our classifier would present a very low performance.

If we analyze the covariance Matrix of class 1 shown below, we can confirm that there is no much covariance for most of the features, however we can see how features 1 and 2 have the highest covariance of 0.4955.

```
 1.9596    0.4955    0.0017   -0.0104
 0.4955    1.0270   -0.0133    0.0038
 0.0017   -0.0133    3.0482   -0.0048
-0.0104    0.0038   -0.0048    1.0336
```

From the classification testing using resubstitution we obtained the following results:

Error Estimate: 5.02%
Average CCR= 94.98%
Average FAR=5.02%

As we can tell from the previous images, with only these 4 features it is not easy to obtain a very low Error Estimate, even class 1 and 2 that show the best performance, have many items that are not linearly separable.

IRIS DATSET

The Iris Dataset contains a total of 150 samples identified under 3 classes, 50 for each class. For each of the samples, 4 features were selected: Sepal Length, Sepal Width, Petal Length and Petal Width. On *figure 4* below we have a view in the 3D Euclidean Space of the first three features of each sample, rotated in the angle where the different regions are more apparent.

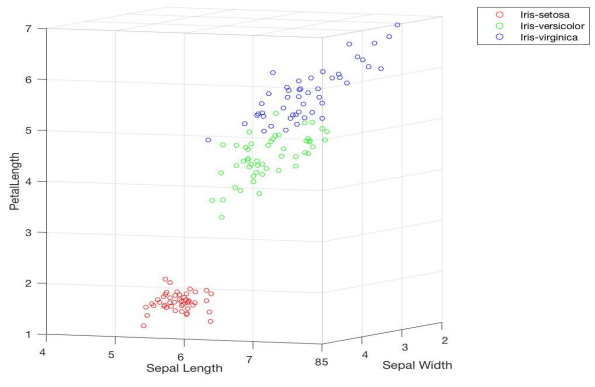


Fig. 4. Features 1, 2 and 3 from the Iris Dataset.

By looking at *figure 4* we can speculate that the Iris Setosa should have the minimum error during classification among all classes.

Let's take a look to the confusion Matrix to see if that's the case:

$$\begin{matrix} 50 & 0 & 0 \\ 0 & 48 & 1 \\ 0 & 2 & 49 \end{matrix}$$

The confusion Matrix show that all the 50 samples were correctly classified using the re-substitution testing, confirming our speculations. In addition to that we can see that the classification of the other 2 classes is very accurate as well as we can conclude with the following results:

Error Estimate = 2%
Average CCR = 98.00%
Average FAR = 1.99%

WINES DATSET

This Dataset contains 178 samples of different wines classified under 3 fundamental types. Each sample contains 13 different features including Alcohol, Malic Acid, Magnesium, Color Intensity among others.

By Analyzing figure 5 we can see that if we were to use only the first 3 features of the dataset the performance of our classifier would have a horrible performance due to the nonlinear separability of the vectors on these combinations.

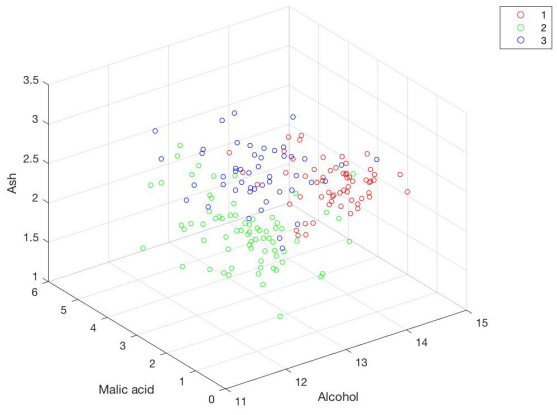


Fig. 5. Features 1, 2 and 3 from the Wines Dataset.

However when we take a look to the results of the resubstitution testing of the classifier, we can see that it's performance is the best among all 3 Datasets:

Error Estimate = 0.5618%
Average CCR = 99.53%
Average FAR = 0.56%

Obviously the high number of features used for the classification had a big impact on the performance of this classifier, it is hard to perceive when we are limitated to a 3 Dimensional visualization of the data. We would need to see all different combinations of features on the 2 or 3 Dimensional cartesian spaces to be able to visualize which particular features are the main contributors on this exceptional performance.

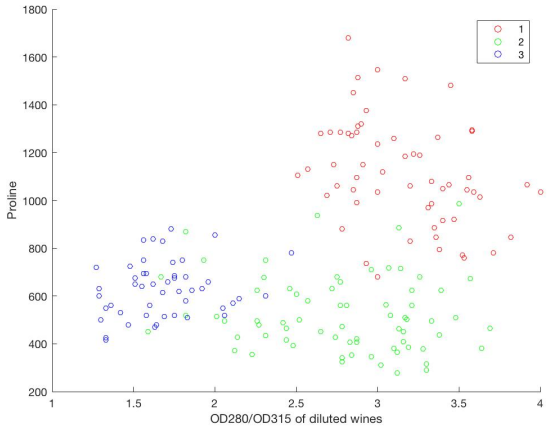


Fig. 6. Wines Datase features 12 and 13. This particular combination contributes to the separation of classes 1 and 2.

We can conclude that there are many factors that affect the performance of a classifier such as the number of samples, the number of features, the selection of features and the quality of the data.

APENDIX A - Test Results

TWO CLASS FOUR GAUSSIANS DATSET

a) Resubstitution Testing:

1) Covariance Matrices:

Class 1:

1.9596	0.4955	0.0017	-0.0104
0.4955	1.0270	-0.0133	0.0038
0.0017	-0.0133	3.0482	-0.0048
-0.0104	0.0038	-0.0048	1.0336

Class 2:

2.0390	0.0282	-1.0509	1.0061
0.0282	2.0301	1.0574	0.0271
-1.0509	1.0574	4.2168	0.0028
1.0061	0.0271	0.0028	0.9974

2) Mean Vectors:

Class 1:

[0.0206 0.9984 -0.0290 -0.9949]

Class 2:

[-2.9952 -2.0147 -1.0003 -0.9781]

3) Confusion Matrix:

4726	228
274	4772

4) Average CCR= 94.98%
Average FAR=5.02%

b) Cross Validation Testing:

1) Covariance Matrices:
One for each fold

Class 1:

1.9482	0.4909	0.0039	-0.0285
0.4909	1.0365	-0.0171	0.0021
0.0039	-0.0171	3.0992	-0.0107
-0.0285	0.0021	-0.0107	1.0254

1.9678	0.5160	0.0074	-0.0071
0.5160	1.0438	-0.0096	0.0012
0.0074	-0.0096	3.0682	0.0011
-0.0071	0.0012	0.0011	1.0273

1.9471	0.4795	-0.0150	0.0026
0.4795	0.9986	-0.0351	-0.0003
-0.0150	-0.0351	3.0253	0.0040
0.0026	-0.0003	0.0040	1.0382

1.9561	0.4904	0.0028	0.0078
0.4904	1.0279	0.0042	0.0185
0.0028	0.0042	3.0405	-0.0185
0.0078	0.0185	-0.0185	1.0287

1.9785	0.5003	0.0092	-0.0268
0.5003	1.0279	-0.0091	-0.0022
0.0092	-0.0091	3.0058	-0.0004
-0.0268	-0.0022	-0.0004	1.0482

1.9596	0.4955	0.0017	-0.0104
0.4955	1.0270	-0.0133	0.0038
0.0017	-0.0133	3.0482	-0.0048
-0.0104	0.0038	-0.0048	1.0336

1.9596	0.4955	0.0017	-0.0104
0.4955	1.0270	-0.0133	0.0038
0.0017	-0.0133	3.0482	-0.0048
-0.0104	0.0038	-0.0048	1.0336

1.9596	0.4955	0.0017	-0.0104
0.4955	1.0270	-0.0133	0.0038
0.0017	-0.0133	3.0482	-0.0048
-0.0104	0.0038	-0.0048	1.0336

1.9596	0.4955	0.0017	-0.0104
0.4955	1.0270	-0.0133	0.0038
0.0017	-0.0133	3.0482	-0.0048
-0.0104	0.0038	-0.0048	1.0336

1.9596	0.4955	0.0017	-0.0104
0.4955	1.0270	-0.0133	0.0038
0.0017	-0.0133	3.0482	-0.0048
-0.0104	0.0038	-0.0048	1.0336

Class 2:

2.0339	0.0304	-1.0322	1.0049
0.0304	2.0321	1.0609	0.0260
-1.0322	1.0609	4.1380	-0.0055
1.0049	0.0260	-0.0055	0.9997

2.0339	0.0304	-1.0322	1.0049
0.0304	2.0321	1.0609	0.0260
-1.0322	1.0609	4.1380	-0.0055
1.0049	0.0260	-0.0055	0.9997

2.0339	0.0304	-1.0322	1.0049
0.0304	2.0321	1.0609	0.0260
-1.0322	1.0609	4.1380	-0.0055
1.0049	0.0260	-0.0055	0.9997

2.0339	0.0304	-1.0322	1.0049
0.0304	2.0321	1.0609	0.0260
-1.0322	1.0609	4.1380	-0.0055
1.0049	0.0260	-0.0055	0.9997

2.0339	0.0304	-1.0322	1.0049
0.0304	2.0321	1.0609	0.0260
-1.0322	1.0609	4.1380	-0.0055
1.0049	0.0260	-0.0055	0.9997

2.0310	0.0617	-1.0043	1.0011
0.0617	2.0208	1.0447	0.0442
-1.0043	1.0447	4.1375	0.0105
1.0011	0.0442	0.0105	1.0019

2.0357	0.0417	-1.0392	1.0002
0.0417	2.0584	1.0425	0.0304
-1.0392	1.0425	4.1058	-0.0121
1.0002	0.0304	-0.0121	0.9954

2.0444	-0.0030	-1.0603	1.0038
-0.0030	1.9949	1.0684	0.0004
-1.0603	1.0684	4.1274	-0.0207
1.0038	0.0004	-0.0207	0.9937

2.0194	0.0233	-1.0063	1.0133
0.0233	2.0557	1.0913	0.0277
-1.0063	1.0913	4.1022	-0.0081
1.0133	0.0277	-0.0081	1.0098

2.0390	0.0282	-1.0509	1.0061
0.0282	2.0301	1.0574	0.0271
-1.0509	1.0574	4.2168	0.0028
1.0061	0.0271	0.0028	0.9974

2) Mean Vectors:
One for each fold

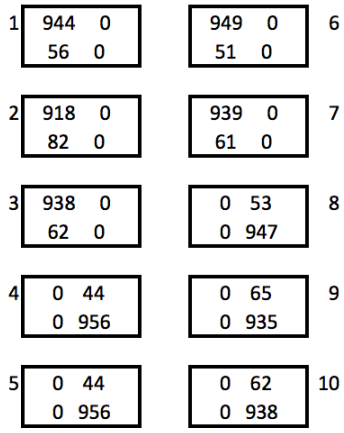
Class 1:

[0.0197 0.9974 -0.0230 -0.9962]
[0.0102 0.9897 -0.0416 -1.0004]
[0.0282 1.0072 0.0083 -0.9890]
[0.0206 0.9927 -0.0459 -0.9927]
[0.0242 1.0050 -0.0429 -0.9960]
[0.0206 0.9984 -0.0290 -0.9949]
[0.0206 0.9984 -0.0290 -0.9949]
[0.0206 0.9984 -0.0290 -0.9949]
[0.0206 0.9984 -0.0290 -0.9949]
[0.0206 0.9984 -0.0290 -0.9949]

Class 2:

[-2.9867 -2.0084 -1.0022 -0.9786]
[-2.9867 -2.0084 -1.0022 -0.9786]
[-2.9867 -2.0084 -1.0022 -0.9786]
[-2.9867 -2.0084 -1.0022 -0.9786]
[-2.9867 -2.0084 -1.0022 -0.9786]
[-2.9862 -2.0064 -0.9995 -0.9802]
[-2.9898 -1.9921 -0.9953 -0.9766]
[-2.9809 -2.0248 -1.0095 -0.9884]
[-2.9816 -2.0039 -1.0063 -0.9700]
[-2.9952 -2.0147 -1.0003 -0.9781]

3) Confusion Matrices:
One for each fold



4) Average CCR
One for each fold

94.4%
94.9%
91.8%
93.9%
93.8%
94.7%
95.6%
93.5%
95.6%
93.8%

Average FAR=
One for each fold

50%
50%
50%
50%
50%
50%
50%
50%
50%
50%

IRIS DATSET

a) Resubstitution Testing:

1) Covariance Matrices:

Iris-Setosa:

0.1218	0.0972	0.0160	0.0101
0.0972	0.1408	0.0115	0.0091
0.0160	0.0115	0.0296	0.0059
0.0101	0.0091	0.0059	0.0109

Iris-Versicolor:

0.2611	0.0835	0.1792	0.0547
0.0835	0.0965	0.0810	0.0404
0.1792	0.0810	0.2164	0.0716
0.0547	0.0404	0.0716	0.0383

Iris-Virginica:

0.4706	0.1262	0.3701	0.0458
0.1262	0.1189	0.0925	0.0463
0.3701	0.0925	0.3573	0.0577
0.0458	0.0463	0.0577	0.0726

2) Mean Vectors:

Iris-Setosa:

[5.0060 3.4280 1.4620 0.2460]

Iris-Versicolor:

[5.9360 2.7700 4.2600 1.3260]

Iris-Virginica:

[6.6171 2.9371 5.6257 1.9771]

3) Confusion Matrix:

50	0	0
0	48	1
0	2	49

4) Average CCR= 98.00%

Average FAR=1.99%

b) Cross Validation Testing:

1) Covariance Matrices:

One for each fold

Iris-Setosa:

0.1053	0.0871	0.0155	0.0079
0.0871	0.1498	0.0098	0.0047
0.0155	0.0098	0.0325	0.0050
0.0079	0.0047	0.0050	0.0120

0.1225	0.1021	0.0129	0.0059
0.1021	0.1465	0.0192	0.0035
0.0129	0.0192	0.0223	0.0063
0.0059	0.0035	0.0063	0.0099

0.1237	0.0924	0.0168	0.0130
0.0924	0.1150	0.0047	0.0150
0.0168	0.0047	0.0295	0.0047
0.0130	0.0150	0.0047	0.0087

0.1285	0.1014	0.0164	0.0115
0.1014	0.1461	0.0103	0.0109
0.0164	0.0103	0.0321	0.0067
0.0115	0.0109	0.0067	0.0118

0.1218	0.0972	0.0160	0.0101
0.0972	0.1408	0.0115	0.0091
0.0160	0.0115	0.0296	0.0059
0.0101	0.0091	0.0059	0.0109

0.1218	0.0972	0.0160	0.0101
0.0972	0.1408	0.0115	0.0091
0.0160	0.0115	0.0296	0.0059
0.0101	0.0091	0.0059	0.0109

0.1218	0.0972	0.0160	0.0101
0.0972	0.1408	0.0115	0.0091
0.0160	0.0115	0.0296	0.0059
0.0101	0.0091	0.0059	0.0109

0.1218	0.0972	0.0160	0.0101
0.0972	0.1408	0.0115	0.0091
0.0160	0.0115	0.0296	0.0059
0.0101	0.0091	0.0059	0.0109

0.1218	0.0972	0.0160	0.0101
0.0972	0.1408	0.0115	0.0091
0.0160	0.0115	0.0296	0.0059
0.0101	0.0091	0.0059	0.0109

0.1218	0.0972	0.0160	0.0101
0.0972	0.1408	0.0115	0.0091
0.0160	0.0115	0.0296	0.0059
0.0101	0.0091	0.0059	0.0109

Iris-Versicolor:

0.2611	0.0835	0.1792	0.0547
0.0835	0.0965	0.0810	0.0404
0.1792	0.0810	0.2164	0.0716
0.0547	0.0404	0.0716	0.0383

0.2611	0.0835	0.1792	0.0547
0.0835	0.0965	0.0810	0.0404
0.1792	0.0810	0.2164	0.0716
0.0547	0.0404	0.0716	0.0383

0.2611	0.0835	0.1792	0.0547
0.0835	0.0965	0.0810	0.0404
0.1792	0.0810	0.2164	0.0716
0.0547	0.0404	0.0716	0.0383

0.1990	0.0557	0.1457	0.0476
0.0557	0.0915	0.0680	0.0392
0.1457	0.0680	0.2132	0.0731
0.0476	0.0392	0.0731	0.0406

0.3066	0.0984	0.2136	0.0646
0.0984	0.0845	0.0920	0.0382
0.2136	0.0920	0.2368	0.0762
0.0646	0.0382	0.0762	0.0335

0.2708	0.0971	0.1892	0.0539
0.0971	0.0995	0.0857	0.0387
0.1892	0.0857	0.2147	0.0628
0.0539	0.0387	0.0628	0.0353

0.2669	0.0844	0.1678	0.0510
0.0844	0.1081	0.0789	0.0442
0.1678	0.0789	0.1984	0.0719
0.0510	0.0442	0.0719	0.0420

0.2611	0.0835	0.1792	0.0547
0.0835	0.0965	0.0810	0.0404
0.1792	0.0810	0.2164	0.0716
0.0547	0.0404	0.0716	0.0383

0.2611	0.0835	0.1792	0.0547
0.0835	0.0965	0.0810	0.0404
0.1792	0.0810	0.2164	0.0716
0.0547	0.0404	0.0716	0.0383

0.2611	0.0835	0.1792	0.0547
0.0835	0.0965	0.0810	0.0404
0.1792	0.0810	0.2164	0.0716
0.0547	0.0404	0.0716	0.0383

Iris-Virginica:

0.3963	0.0919	0.2972	0.0481
0.0919	0.1019	0.0700	0.0467
0.2972	0.0700	0.2985	0.0478
0.0481	0.0467	0.0478	0.0739

0.3963	0.0919	0.2972	0.0481
0.0919	0.1019	0.0700	0.0467
0.2972	0.0700	0.2985	0.0478
0.0481	0.0467	0.0478	0.0739

0.3963	0.0919	0.2972	0.0481
0.0919	0.1019	0.0700	0.0467
0.2972	0.0700	0.2985	0.0478
0.0481	0.0467	0.0478	0.0739

0.3963	0.0919	0.2972	0.0481
0.0919	0.1019	0.0700	0.0467
0.2972	0.0700	0.2985	0.0478
0.0481	0.0467	0.0478	0.0739

0.3963	0.0919	0.2972	0.0481
0.0919	0.1019	0.0700	0.0467
0.2972	0.0700	0.2985	0.0478
0.0481	0.0467	0.0478	0.0739

0.3963	0.0919	0.2972	0.0481
0.0919	0.1019	0.0700	0.0467
0.2972	0.0700	0.2985	0.0478
0.0481	0.0467	0.0478	0.0739

0.4164	0.0987	0.3223	0.0525
0.0987	0.1091	0.0715	0.0471
0.3223	0.0715	0.3184	0.0458
0.0525	0.0471	0.0458	0.0748

0.3052	0.0570	0.2011	0.0323
0.0570	0.0674	0.0494	0.0364
0.2011	0.0494	0.2130	0.0357
0.0323	0.0364	0.0357	0.0739

0.3836	0.0854	0.2853	0.0649
0.0854	0.1079	0.0702	0.0546
0.2853	0.0702	0.2920	0.0571
0.0649	0.0546	0.0571	0.0681

0.4706	0.1262	0.3701	0.0458
0.1262	0.1189	0.0925	0.0463
0.3701	0.0925	0.3573	0.0577
0.0458	0.0463	0.0577	0.0726

2) Mean Vectors:
One for each fold

Iris-Setosa:

[5.0457 3.4629 1.4800 0.2657]
[4.9486 3.3743 1.4343 0.2257]
[5.0171 3.4429 1.4714 0.2429]
[5.0111 3.4311 1.4622 0.2489]
[5.0060 3.4280 1.4620 0.2460]
[5.0060 3.4280 1.4620 0.2460]
[5.0060 3.4280 1.4620 0.2460]
[5.0060 3.4280 1.4620 0.2460]
[5.0060 3.4280 1.4620 0.2460]
[5.0060 3.4280 1.4620 0.2460]

Iris-Versicolor:

[5.9360 2.7700 4.2600 1.3260]
[5.9360 2.7700 4.2600 1.3260]
[5.9360 2.7700 4.2600 1.3260]
[5.8950 2.7450 4.2325 1.3125]
[5.9286 2.7943 4.2543 1.3286]
[5.9057 2.7629 4.2314 1.3114]
[6.0100 2.7800 4.3175 1.3500]
[5.9360 2.7700 4.2600 1.3260]
[5.9360 2.7700 4.2600 1.3260]
[5.9360 2.7700 4.2600 1.3260]

Iris-Virginica:

[6.5880 2.9740 5.5520 2.0260]
[6.5880 2.9740 5.5520 2.0260]
[6.5880 2.9740 5.5520 2.0260]
[6.5880 2.9740 5.5520 2.0260]
[6.5880 2.9740 5.5520 2.0260]
[6.5880 2.9740 5.5520 2.0260]
[6.6089 2.9733 5.5378 2.0178]
[6.5771 3.0057 5.5114 2.0257]
[6.5429 2.9800 5.5371 2.0857]
[6.6171 2.9371 5.6257 1.9771]

3) Confusion Matrices:
One for each fold

1
15	0	0
0	0	0
0	0	0

2
15	0	0
0	0	0
0	0	0

3
15	0	0
0	0	0
0	0	0

4
5	0	0
0	10	0
0	0	0

5
0	0	0
0	12	0
0	3	0

6
0	0	0
0	14	0
0	1	0

7
0	0	0
0	10	0
0	0	5

8
0	0	0
0	0	0
0	0	15

9
0	0	0
0	0	1
0	0	14

10
0	0	0
0	0	0
0	0	15

4) Average CCR
One for each fold

100%
100%
100%
100%
80%
93.33%
100%
100%
93.33%
100%

Average FAR=
One for each fold

0%
0%
0%
0%
50%
50%
0%
0%
50%
0%

WINE DATSET

a) Resubstitution Testing:

1) Covariance Matrices:

Class 1:

0	0	0	0	1	0	0	0	0	0	0	0	36
0	0	0	0	1	0	0	0	0	0	0	0	-56
0	0	0	0	1	0	0	0	0	0	0	0	-1
0	0	0	6	6	0	0	0	0	-1	0	0	-68
1	1	1	6	108	1	1	0	0	2	0	0	-338
0	0	0	0	1	0	0	0	0	0	0	0	22
0	0	0	0	1	0	0	0	0	0	0	0	33
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	13
0	0	0	-1	2	0	0	0	0	2	0	0	159
0	0	0	0	0	0	0	0	0	0	0	0	9
0	0	0	0	0	0	0	0	0	0	0	0	-27
36	-56	-1	-68	-338	22	33	0	13	159	9	-27	48240

Class 2:

0	0	0	0	0	0	0	0	0	0	0	0	4
0	1	0	1	-1	0	0	0	0	0	0	0	-35
0	0	0	1	1	0	0	0	0	0	0	0	2
0	1	1	11	0	0	1	0	0	0	0	1	-8
0	-1	1	0	277	1	0	0	3	1	0	-1	1297
0	0	0	0	1	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	-13
0	0	0	0	0	0	0	0	0	0	0	0	-3
0	0	0	0	3	0	0	0	0	0	0	0	12
0	0	0	0	1	0	0	0	0	1	0	0	15
0	0	0	0	0	0	0	0	0	0	0	0	4
0	0	0	1	-1	0	0	0	0	0	0	0	-9
4	-35	2	-8	1297	1	-13	-3	12	15	4	-9	24367

Class 3:

0	0	0	0	0	0	0	0	0	0	0	0	-5
0	1	0	0	-2	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	-3
0	0	0	5	4	0	0	0	0	1	0	0	-25
0	-2	0	4	116	0	2	-1	1	3	0	-1	233
0	0	0	0	0	0	0	0	0	0	0	0	2
0	0	0	0	2	0	0	0	0	0	0	0	-8
0	0	0	0	-1	0	0	0	0	0	0	0	3
0	0	0	0	1	0	0	0	0	1	0	0	9
0	0	0	1	3	0	0	0	1	5	0	0	31
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	-1	0	0	0	0	0	0	0	6
-5	0	-3	-25	233	2	-8	3	9	31	0	6	12971

2) Mean Vectors:

Class 1:

[0.0137 0.0020 0.0025 0.0170 0.1063 0.0028 0.0030 0.0003 0.0019 0.0055 0.0011 0.0032 1.1157]

Class 2:

[12.2787 1.9327 2.2448 20.2380 94.5493 2.2589 2.0808 0.3637 1.6303 3.0866 1.0563 2.7854 519.5070]

Class 3:

[13.1537 3.3338 2.4371 21.4167 99.3125 1.6788 0.7815 0.4475 1.1535 7.3962 0.6827 1.6835 629.8958]

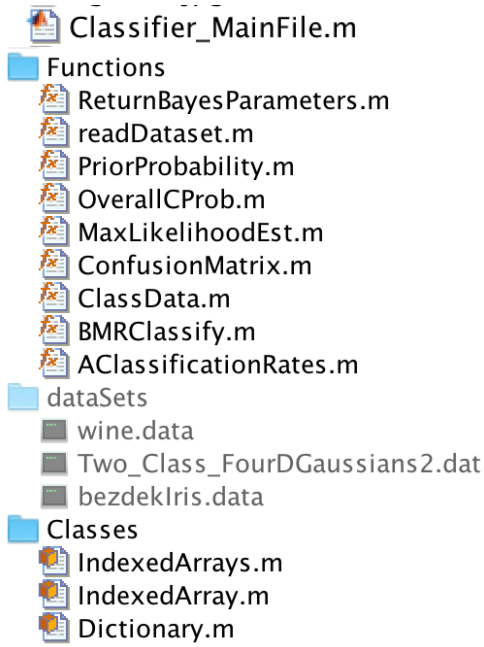
3) Confusion Matrix:

59	1	0
0	70	0
0	0	48

4) Average CCR= 99.53%
   Average FAR=0.56%

APENDIX B - Source Code

The following Program was created in matlab it consists in 1 Main file namesd Classifier_MainFile.m located in the root folder. Inside the root folder there are other 2 folders: 'Functions' which contains all the functions called by the main files, and 'Classes' which contain 3 different classes used during the development of the project. The datasets are included in the dataSets folder as shown in the following figure:



you can find the Final Code.zip file attached to this file.