Purdue University
Pattern Recognition and Decision Making
ECE662
Homework $\sharp$1

March 2'nd 2010

1 Introduction

Through out this homework several experiments have been designed to cover most of the topics that we have learned in the lecture, section 2 presents an experiment that clarifies Bayes Error Probability and Integral in one dimension, section 3 describes experiments for designing discriminant functions with normal density, and section 4 evaluate those functions by calculating the empirical training error and the true error, and showing how those error are bounded by both Chernoff and Bhattacharyya bounds, Finally, section 5 illustrate the Central Limit Theorem.

2 One Dimensional Error Probability and Integral

2.1 Brief Description

This section focuses on The concept of Bayes Decision Theory, and show how such a decision rule can minimize the classification error probability which will be calculated analytically by integrating a specific area as will be shown later in this section and finally the error probability will be compared to both Chernoff and Bhattacharyya error bounds.

The experiment in this section is as follows:
Assume that we have two-classes ω_1 and ω_2 in which our patterns belong, and suppose that we were given two statistical quantities the prior probabilities P(ω_1),P(ω_2) and the class conditional probability density functions p(x|ωi), i = *1,2* are to be gaussian distributed with variance σ_1^2, σ_2^2 and mean μ_1, μ_2, the

purpose of this experiment is to clarify the effect of each of the above parameters and to find the threshold x_0 that minimizes the error probability, and to achieve this let us study the cases below.

2.2 Case 1

Given that $\mu_1 = 0$, $\mu_2 = 1$, and $\sigma_1^2 = \sigma_2^2 = 0.5$, also suppose that our two classes are equiprobable such that $\mathrm{P}(\omega_1) = \mathrm{P}(\omega_2) = 0.5$.
To find the threshold x_0 which represents the decision boundary, we have to define our likelihood p.d.f's

$$\mathrm{p(x|}\omega_1) = \frac{1}{\sqrt{2\pi}} \exp(x^2)$$

$$\mathrm{p(x|}\omega_2) = \frac{1}{\sqrt{2\pi}} \exp((x-1)^2)$$

The decision Rule that will minimize the classification error is the bayes classification rule which can be stated as:

$$\text{if } \mathrm{p(x|}\omega_1)\mathrm{P}(\omega_1) > \mathrm{p(x|}\omega_2)\mathrm{P}(\omega_2), \text{ x is classified to } \omega_1$$

$$\text{if } \mathrm{p(x|}\omega_1)\mathrm{P}(\omega_1) < \mathrm{p(x|}\omega_2)\mathrm{P}(\omega_2), \text{ x is classified to } \omega_2$$

So the threshold is nothing but a decision surface with the following equation:

$$\mathrm{p(x|}\omega_1)\mathrm{P}(\omega_1) = \mathrm{p(x|}\omega_2)\mathrm{P}(\omega_2)$$

But since $\mathrm{P}(\omega_1) = \mathrm{P}(\omega_2) = 0.5$ we have:

$$\mathrm{p(x|}\omega_1) = \mathrm{p(x|}\omega_2)$$

$$\frac{1}{\sqrt{2\pi}} \exp(x^2) = \frac{1}{\sqrt{2\pi}} \exp((x-1)^2)$$

$$\ln\ (\exp(x^2)) = \ln\ (\exp((x-1)^2))$$

$$x^2 = (x-1)^2$$

$$x_0 = 0.5$$

Table 1: Bayesian classifier Error when $\mathrm{P}(\omega_1)$=$\mathrm{P}(\omega_2)$=0.5

P(error)	**Chernoff Bound**	**Bhattacharyya Bound**
0.2397	0.3894	0.3894

Notice that the total error probability is the area in the cyan as shown in Fig.1 which can be calculated as:

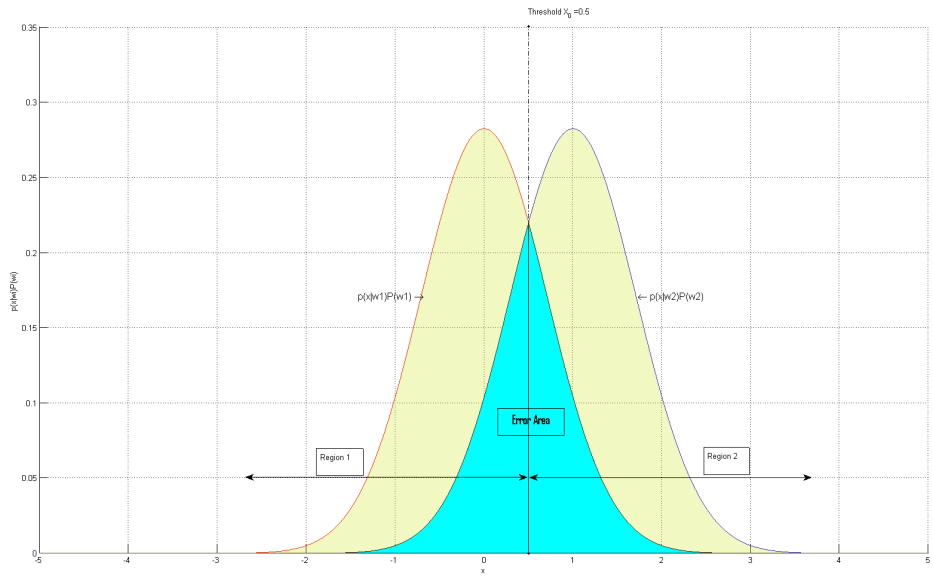


Figure 1: Two regions formed by the Bayesian classifier when P(ω_1)=P(ω_2)=0.5

P(error) = $\int_{-\infty}^{x_0}$ p(x|ω_2)P(ω_2).dx + $\int_{x_0}^{\infty}$ p(x|ω_1)P(ω_1).dx

where : $x_0 = 0.2397$.
This integration is evaluated automatically in the simulation program and the result is P(error) = 0.2397 which represents the true error, Table.1 compares this error with both Chernoff and Bhattacharyya error bounds,notice that both error bounds are equal and that is because $\sigma_1^2 = \sigma_2^2 = 0.5$, another observation is related to the threshold x_0, shifting this threshold to the left or to the right will increase the the area in cyan so the error probability will increase, so in this case the threshold is located exactly in the middle between the mean values of class 1 and class2 since P(ω_1)=P(ω_2)=0.5.

2.3 Case 2

Given that $\mu_1 = 0$, $\mu_2 = 1$, and $\sigma_1^2 = \sigma_2^2 = 0.5$, also suppose P($\omega_1$) = 0.3, P($\omega_2$) = 0.7.
To find the threshold x_0 which represents the decision boundary, we have to solve the following decision surface equation:

$$\text{p(x}|\omega_1)\text{P}(\omega_1) = \text{p(x}|\omega_2)\text{P}(\omega_2)$$

$$(0.3)\text{p(x}|\omega_1) = (0.7)\text{p(x}|\omega_2)$$

$$\frac{0.3}{\sqrt{2\pi}}\ \exp(x^2) = \frac{0.7}{\sqrt{2\pi}}\ \exp((x-1)^2)$$

$$\ln\ (\exp(x^2)) = \ln(\frac{0.7}{0.3}\ (\exp((x-1)^2)))$$

$$x_0 = 0.07635$$

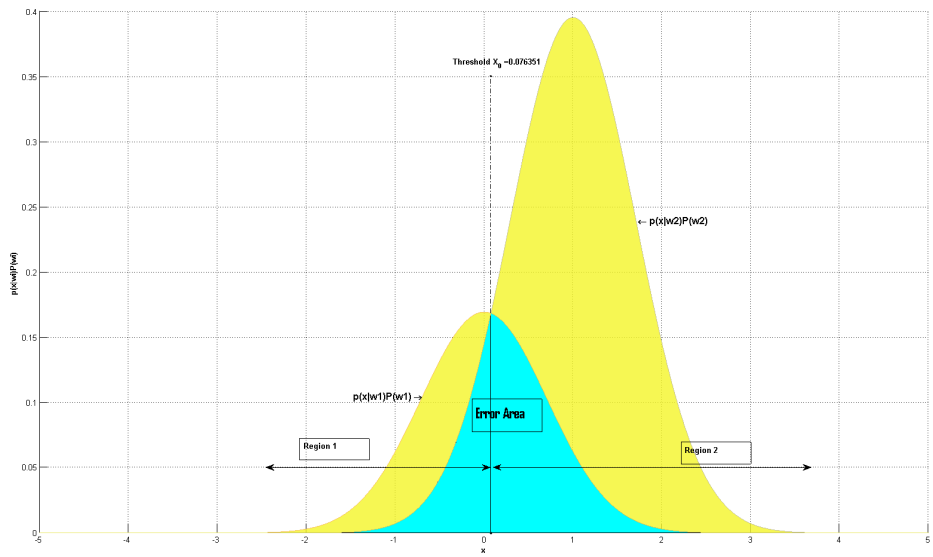


Figure 2: Two regions formed by the Bayesian classifier when P(ω_1)= 0.3,P(ω_2)=0.7

Table 2: Bayesian classifier Error when P(ω_1)= 0.3, P(ω_2)=0.7

P(error)	**Chernoff Bound**	**Bhattacharyya Bound**
0.07635	0.3569	0.3569

Notice that the total error probability is the area in the cyan as shown in Fig.2 which can be calculated as:
P(error) = P(error) = $\int_{-\infty}^{x_0}$ p(x|ω_2)P(ω_2).dx + $\int_{x_0}^{\infty}$ p(x|ω_1)P(ω_1).dx

where : $x_0 = 0.07635$.
This integration is evaluated automatically in the simulation program and the result is P(error) = 0.07635 which represents the true error, Table.2 compares this error with both Chernoff and Bhattacharyya error bounds,notice that both

error bounds are equal and that is because $\sigma_1^2 = \sigma_2^2 = 0.5$.
In this case we should expect that the threshold will be shifted to the left away from the mean of class2, which means that we are expanding the region in which we are deciding the most probable class which is in this case class2 and this is shown clearly in Fig.2.

2.4 Case 3

Given that $\mu_1 = 0$, $\mu_2 = 1$, and $\sigma_1^2 = \sigma_2^2 = 0.5$, also suppose P($\omega_1$) = 0.8 , P($\omega_2$) = 0.2.
To find the threshold x_0 which represents the decision boundary, we have to solve the following decision surface equation:

$$\text{p(x}|\omega_1)\text{P}(\omega_1) = \text{p(x}|\omega_2)\text{P}(\omega_2)$$

$$(0.8)\text{p(x}|\omega_1) = (0.2)\text{p(x}|\omega_2)$$

$$\frac{0.8}{\sqrt{2\pi}}\ \exp(x^2) = \frac{0.2}{\sqrt{2\pi}}\ \exp((x-1)^2)$$

$$\ln\ (\exp(x^2)) = \ln(\frac{0.2}{0.8}\ (\exp((x-1)^2)))$$

$$x_0 = 1.1931$$

Table 3: Bayesian classifier Error when P(ω_1)= 0.8, P(ω_2)=0.2

P(error)	**Chernoff Bound**	**Bhattacharyya Bound**
0.1581	0.3115	0.3115

Notice that the total error probability is the area in the cyan as shown in Fig.3 which can be calculated as:
P(error) = $\int_{-\infty}^{x_0}$ p(x|ω_2)P(ω_2).dx + $\int_{x_0}^{\infty}$ p(x|ω_1)P(ω_1).dx

where : $x_0 = 1.1931$.
This integration is evaluated automatically in the simulation program and the result is P(error) = 0.1581 which represents the true error, Table.3 compares this error with both Chernoff and Bhattacharyya error bounds,notice that both error bounds are equal and that is because $\sigma_1^2 = \sigma_2^2 = 0.5$.
In this case we should expect that the threshold will be shifted to the right away from the mean of class1, which means that we are expanding the region in which we are deciding the most probable class which is in this case class1 and this is shown clearly in Fig.3.
For more details about 1Dimensional Bayes error see [4].

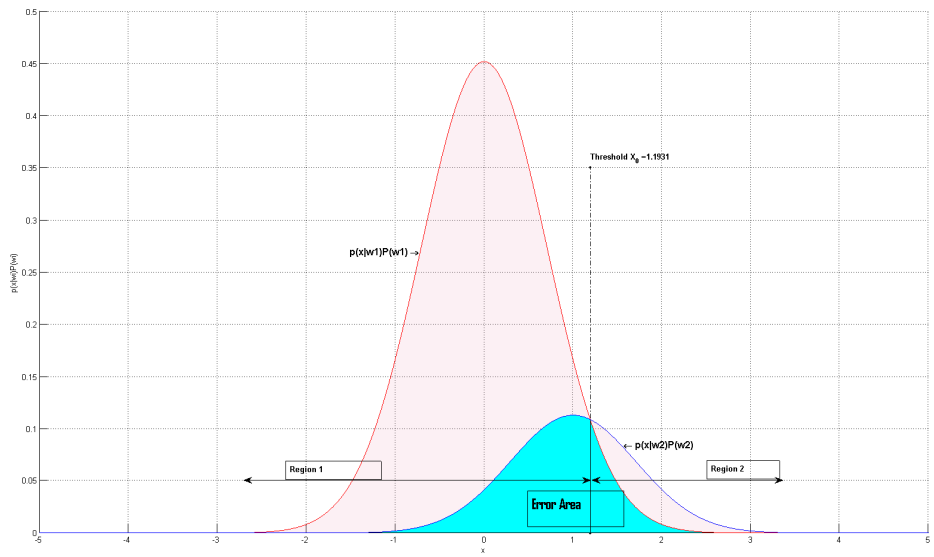


Figure 3: Two regions formed by the Bayesian classifier when P(ω_1)= 0.8,P(ω_2)=0.2

2.5 Case 4

Given that $\mu_1 = 0$, $\mu_2 = 1$, and σ_1^2 =1, $\sigma_2^2 = 3$, also suppose P(ω_1) = 0.5 , P(ω_2) = 0.5.
To find the threshold x_0 which represents the decision boundary, we have to solve the following decision surface equation:

$$\text{p(x}|\omega_1)\text{P}(\omega_1) = \text{p(x}|\omega_2)\text{P}(\omega_2)$$

$$(0.5)\text{p(x}|\omega_1) = (0.5)\text{p(x}|\omega_2)$$

$$\frac{0.5}{\sqrt{2\pi}}\ \exp(x^2) = \frac{0.5}{\sqrt{2\pi}}\ \exp(\frac{(x-1)^2}{4})$$

$$\ln\ (\exp(x^2)) = \ln((\exp((x-1)^2)))$$

$$x_0 = \text{-2.0485}$$

$$x_1 = 1.0485$$

Notice that the total error probability is the area in the cyan as shown in Fig.4 which can be calculated as:

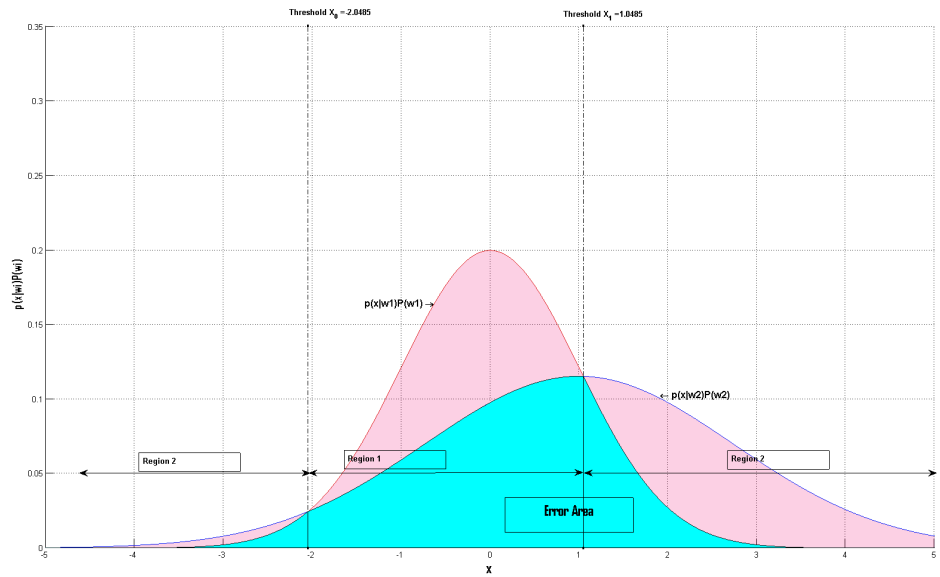


Figure 4: Two regions formed by the Bayesian classifier when P(ω_1)= 0.5, P(ω_2)=0.5 and σ_1^2 =1, σ_2^2 = 3

Table 4: Bayesian classifier Error when P(ω_1)=P(ω_2)=0.5 and σ_1^2 =1,σ_2^2 = 3

P(error)	**Chernoff Bound**	**Bhattacharyya Bound**
0.3197	0.4343	0.4371

P(error) = $\int_{-\infty}^{x_0}$ p(x|ω_1)P(ω_1).dx + $\int_{x_0}^{x_1}$ p(x|ω_2)P(ω_2).dx + $\int_{x_1}^{\infty}$ p(x|ω_1)P(ω_1).dx
This integration is evaluated automatically in the simulation program and the result is P(error) = 0.3197 which represents the true error, Table.4 compares this error with both Chernoff and Bhattacharyya error bounds.

This case differ than the previous three cases in the following:

1. The decision regions are not contiguous because we have 2 decision points x_0 and x_1.

2. The Chernoff and Bhattacharyya are no longer the same and this is because Chernoff bound is obtained by finding the argument β that minimizes Eq.73 in [3] which depend on both the means and variances of our classes and then apply Eq.72 in [3],and since σ_1^2 =1, σ_2^2 = 3 are no longer equal so this mean that we will get β not equal to 0.5 so the chernoff bound will not be equal to bhattacharrya bound, furthermore chernoff bound will be always less than or equal bhattacharyya.

3. The true error is less than both bounds, but greater than all the true error in the previous cases and that can be seen from Fig.4 which shows that the error area "in cyan" had been increased which result in increasing the error probability.

3 Discriminant Functions for Normal Distributions

3.1 Objectives

In section 2 we designed a bayes classifier for the case when our features space is one dimensional, in this section we will generate a synthetic random normal samples in two dimensional feature spaces, "for visualization issues", and finally a complete error analysis will be handled for each scenario in the next section.

3.2 Synthetic Normal Data Generator

The likelihood function of ω_i with respect to x in the n-dimensional feature space follow the general multivariate normal density described below:

$$\mathrm{p(x}|\omega_i) = \frac{1}{(2\pi)^{\frac{n}{2}}} \frac{1}{(|\Sigma_i|)^{\frac{1}{2}}} \exp((-0.5)(x-\mu_i)^T {\Sigma_i}^{-1}(x-\mu_i)) \ ,i= 1,...,M$$

For the purpose of this experiment a normal random generator was written to generate synthetic data according to the previous equation given the mean vectors, and the covariance matrices for each class ω_i, i= 1,...,M.
So for example suppose that we have 3 classes and we want to generate 10 samples for each class according to the following specifications.

$$\mu_1 = \begin{bmatrix} 4 \\ 4 \end{bmatrix}, \mu_2 = \begin{bmatrix} 3 \\ 3 \end{bmatrix}, \mu_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\Sigma_1 = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \Sigma_3 = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

Notice in Fig.5, in the contour plot, the black crosses represents the 10 samples of ω_1, the black circles represents the 10 samples of ω_2, and the black dots represents the 10 samples of ω_3.
In the following subsections we will design Baysian classifiers based on constructing decision functions of the form:

$$g_i(\mathrm{x}) = \ln \mathrm{p(x}|\omega_i) + \ln \mathrm{P}(\omega_i)$$

and then we will examine the classifiers under several parameter and feature dimensionality values, for this purpose three scenarios were designed and an error analysis for each case were performed as describe in the following subsections.

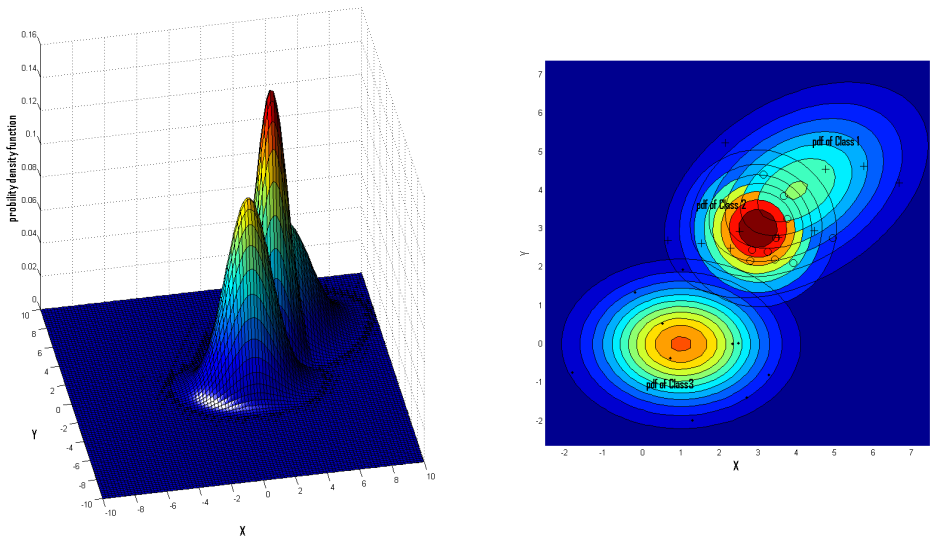


Figure 5: Likelihood p.d.f for three classes and their contours.

3.3 Case 1: 2-D with $\Sigma_i = \Sigma = \sigma^2 \boldsymbol{I}$

Assuming that the covariance matrices is the same for all classes i.e Σ_i=Σ=$\sigma^2 I$, and since our likelihood p.d.f's are a multivariate Normal density function, then the discriminant functions will have the following forms:

$$g_i(\mathrm{x}) = \mathbf{w}_i^T \mathrm{x} + w_{i0}$$

where

$$\mathbf{w}_i = \Sigma^{-1}\mu_i$$

$$w_{i0} = \ln \mathrm{P}(\omega_i) + (\text{-}0.5)\mu_i^T \Sigma^{-1}\mu_i$$

So our discriminant functions $g_i(\mathrm{x})$ are linear functions and this mean that the respective decision surfaces are hyperplanes, in this case we will concern with 2-Dimensional feature space and we will assume that we have only 2 classes, applying all the assumptions we end up with the following decision hyperplanes:

$$g_i(\mathbf{x}) \text{ - } g_j(\mathbf{x}) = g_{ij}(\mathbf{x}) = \mathbf{w}^T(\mathbf{x} \text{ - } \mathbf{x}_0) = \mathbf{0}$$

where

$$\mathbf{w} = \mu_i - \mu_j$$

$$\mathbf{x}_0 = (0.5)(\mu_i + \mu_j) \text{ - } \sigma^2 \ln(\tfrac{P(\omega_i)}{P(\omega_j)}) \ \tfrac{\mu_i - \mu_j}{||\mu_i - \mu_j||^2}$$

This mean that the decision surface is a hyperplane passing through the point $\mathbf{x}_0$, and if $P(\omega_i) = P(\omega_j)$, then $\mathbf{x}_0 = (0.5)(\mu_i + \mu_j)$, that is the hyperplane passes through the mean of μ_i, μ_j.
To illustrate this idea suppose that $\Sigma_i = \Sigma = I$, also the mean vectors $\mu_1 = [1, 0]^T$, and $\mu_2 = [0, 0]^T$ then the decision hyperplane in this case will be a straight line that is orthogonal to the vector $\mu_1 - \mu_2$ and passes through the mean of μ_1, and μ_2 since $\mathrm{P}(\omega_1) = \mathrm{P}(\omega_2) = 0.5$ as shown in Fig.6, where crosses represents data points from class1 and circles from class2 and the blue ellipses are just a single contour of the distributions.

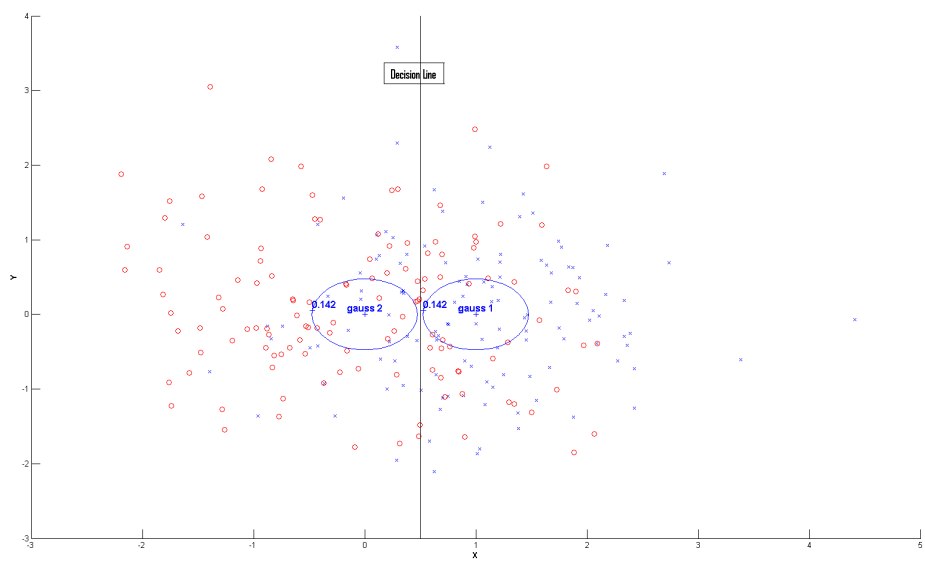


Figure 6: Decision line for two classes and normally distributed vectors with $\mathrm{P}(\omega_1)$=$\mathrm{P}(\omega_2)$=0.5 and Σ_1=$\Sigma_2 = I$ this figure generated using [2]

The effect of the Prior probability on the location of the decision surface is shown in Fig.7 and Fig.8, and can be stated such that the decision hyperplane is located closer to μ_1 if $\mathrm{P}(\omega_1) < \mathrm{P}(\omega_2)$ and the opposite is true.

Another interesting observation is that the location of the hyperplane is *insensitive* to the value of $\mathrm{P}(\omega_1)$, and $\mathrm{P}(\omega_2)$ if σ^2 is small w.r.t $||\mu_1 - \mu_2||$, and this is expected because small variance mean that most of the vectors will be clustered within a small radius around their mean, and this is shown in Fig.9

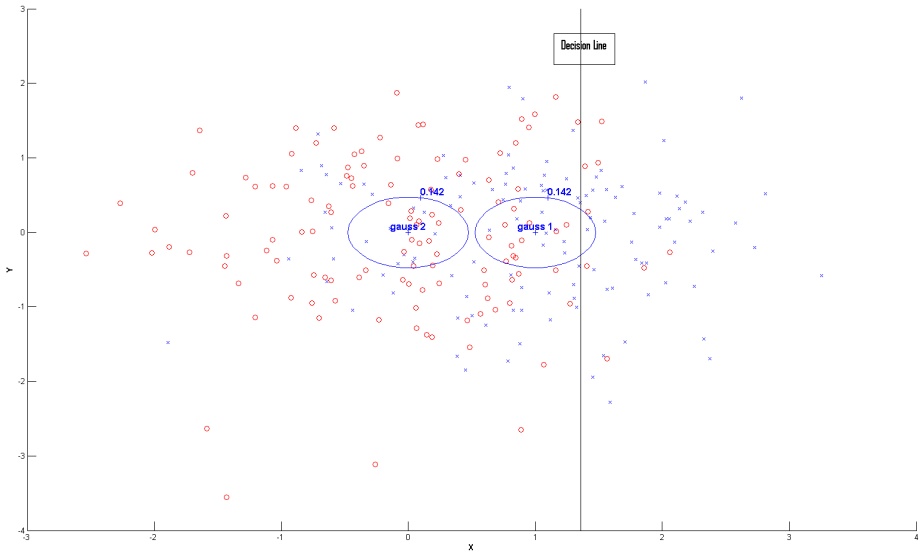


Figure 7: Decision line for two classes and normally distributed vectors with P(ω_1)=0.3 P(ω_2)=0.7 and Σ_1=Σ_2 = I

and Fig.10 for $\sigma^2 = 0.001$.

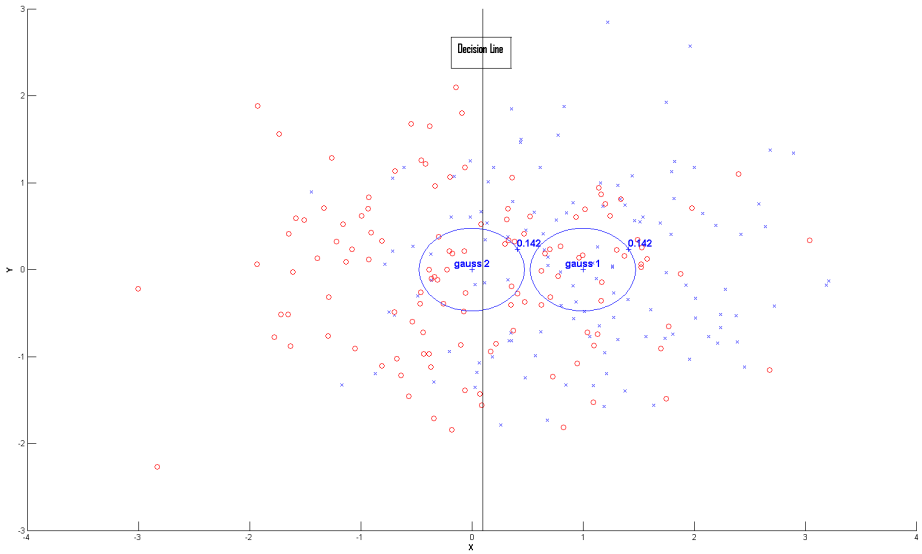


Figure 8: Decision line for two classes and normally distributed vectors with P(ω_1)=0.6 P(ω_2)=0.4 and Σ_1=Σ_2 = I

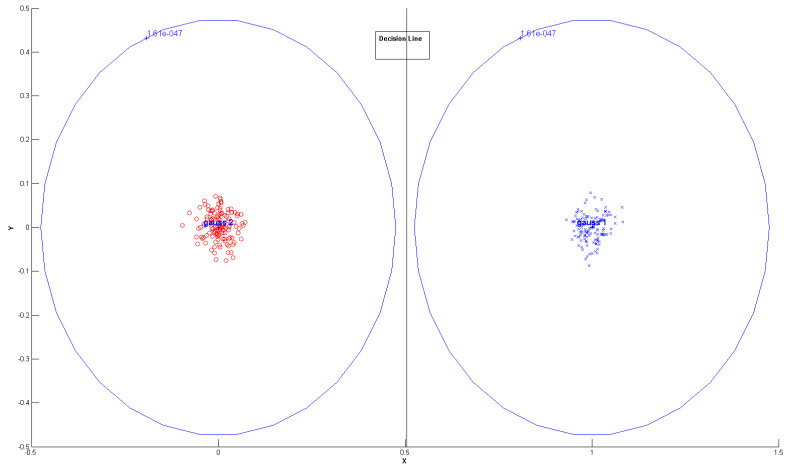


Figure 9: Decision line for two classes and normally distributed vectors with P(ω_1)=0.3 P(ω_2)=0.7 and Σ_1=Σ_2 = $0.001I$

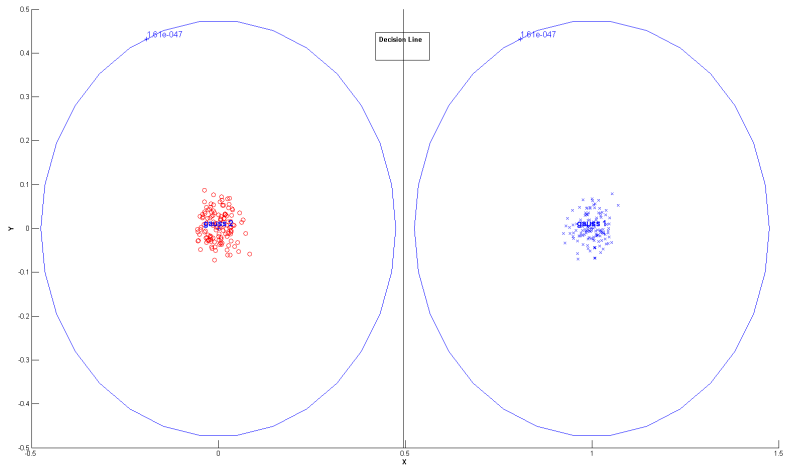


Figure 10: Decision line for two classes and normally distributed vectors with P(ω_1)=0.6 P(ω_2)=0.4 and Σ_1=Σ_2 = $0.001I$

3.4 Case 2: 2-D with $\Sigma_i = \Sigma$

Now if all the classes have the same arbitrary covariance matrix Σ, in this case the discriminant function still linear function of the form:

$$g_i(\mathbf{x}) \text{ - } g_j(\mathbf{x}) = g_{ij}(\mathbf{x}) = \mathbf{w}^T(\mathbf{x} \text{ - } \mathbf{x}_0) = \mathbf{0}$$

where

$$\mathbf{w} = \Sigma^{-1}\ \mu_i - \mu_j$$

$$\mathbf{x}_0 = (0.5)(\mu_i + \mu_j) \text{ - } \ln(\frac{P(\omega_i)}{P(\omega_j)})\ \frac{\mu_i - \mu_j}{||\mu_i - \mu_j||^2_{\Sigma-1}}$$

This case is similar to the previous case except that decision hyperplane is no longer orthogonal to the the vector $\mu_i - \mu_j$ but to its linear transformation Σ^{-1} $||\mu_i - \mu_j||$.
To illustrate this idea suppose that:

$$\Sigma_i = \Sigma = \begin{pmatrix} 3 & 1 \\ 1 & 2 \end{pmatrix}$$

also the mean vectors $\mu_1 = [1, 0]^T$, and $\mu_2 = [0, 0.5]^T$ then the decision hyperplane is shown in Fig.11, where crosses represents data points from class1 and circles from class2.

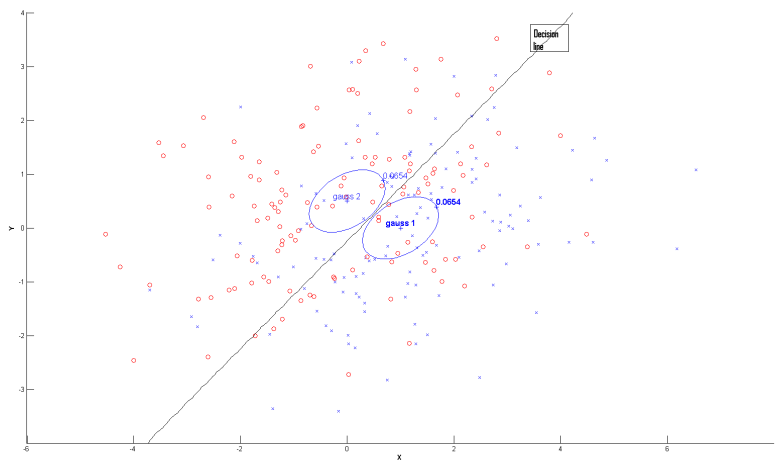


Figure 11: Decision line for two classes and normally distributed vectors with P(ω_1)=0.5 P(ω_2)=0.5 and Σ_i=Σ

3.5 Case 3: 2-D with $\Sigma_i \neq \Sigma_j$ "Quadratic Classifier"

Now if each class has different arbitrary covariance matrix Σ_i, in this case the discriminant function is quadratic of the form:

$$g_i(\mathbf{x}) = (\text{-}0.5)(\mathbf{x} - \mu_i)^T \Sigma_i^{-1} (\mathbf{x} - \mu_i) + \ln P(\omega_i) + c_i$$

where

$$c_i = -\ln(2\pi) \text{ - } (0.5) \ln |\Sigma_i|$$

In this case the decision surfaces will be one of the following hyperquadratic forms (hyperbolas, parabolas, ellipsoids).
To illustrate this idea suppose that:

$$\Sigma_1 = \begin{pmatrix} 3 & 2 \\ 2 & 2 \end{pmatrix}$$

$$\Sigma_2 = \begin{pmatrix} 4 & 0.3 \\ 0.3 & 1 \end{pmatrix}$$

also the mean vectors $\mu_1 = [1, 0]^T$, and $\mu_2 = [0, 0]^T$ then the decision hyperplane is shown in Fig.12 and Fig.13, where crosses represents data points from class1 and circles from class2 and blue ellipses represent a single contour of the distributions.

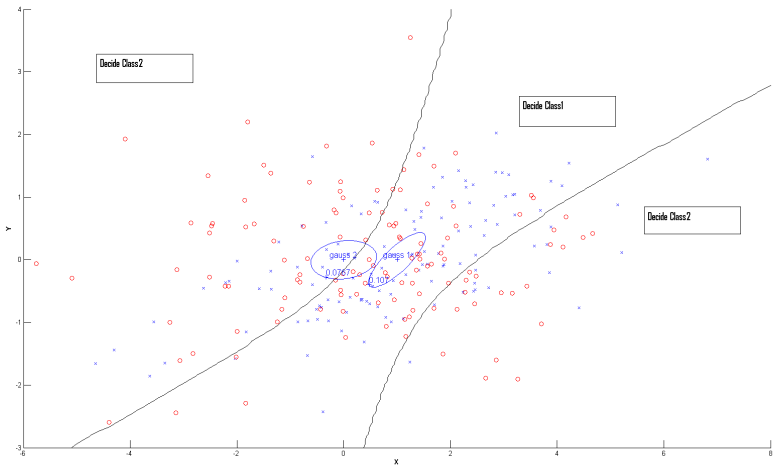


Figure 12: Decision hyperquadratic surface for two classes and normally distributed vectors with P(ω_1)=0.5 P(ω_2)=0.5 and $\Sigma_i \neq \Sigma_j$

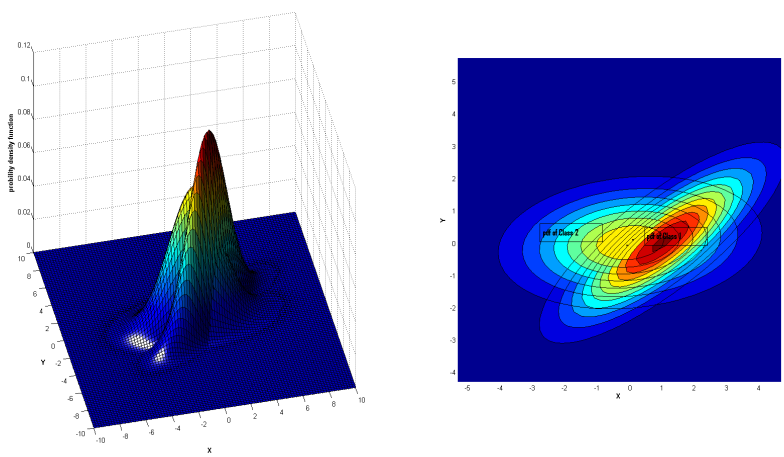


Figure 13: 3D representation of the Likelihood p.d.f's with P(ω_1)=0.5 P(ω_2)=0.5 and $\Sigma_i \neq \Sigma_j$

4 Error Analysis

In this section we will generate d-Dimensional normally distributed data for 2 classes, then the data will be classified and four error quantities will be computed

for different cases, the error quantities are listed below:

1. The training error rate which the average of misclassification of point for each class.

2. The Chernoff and Bhattacharyya error bounds.

3. The probability of error i.e "true error" evaluated by determining the the density of the likelihood ratio h($\mathbf{x}$) and integrate in the one dimensional h space as described in sec.3.3 in [1].
   And finally the P(error) = P(ω_1)ε_1 + P(ω_2)ε_2 .

In the following subsections, 1000 d-Dimensional normal samples was generated for each class with the same covariance matrix and that is to evaluate the true error, and the four error quantities are evaluated in the following way:
start by considering only the first feature in our feature space, and consider the first 100 samples for each class and evaluate the empirical training error, True error, Chernoff and Bhattacharyya Bounds, Then considering the first 200 samples for each class and evaluate the four error quantities, and so on until considering the whole 1000 samples.
Then we consider 2 features in our feature space and redo the experiment, and so on.

4.1 Consider d = 2

In this case the feature space is in $\boldsymbol{R}^2$, and $\mu_1 = [2, 1]^T$, $\mu_2 = [1, 3]^T$, and

$$\Sigma_1 = \Sigma_2 = \begin{pmatrix} 4 & 1 \\ 1 & 2 \end{pmatrix}$$

and the distribution of the points and the decision surface along with the likelihood p.d.f's for each class are shown in Fig.14, Fig.15.The results of the error analysis are shown in Fig.16.

<u>Observations</u>

1. The Chernoff and Bhattacharyya error bounds are equal due to the fact that the covariance matrices are equal.

2. Notice that both the empirical training error and the true error in this case have values less than the Bhattacharyya bound.

3. Notice that from Fig.14, and Fig.15 that given only the x component of any point, then this point can be classified as class1while it is belong to class2 and the opposite is true, so this will lead to a higher training error and true error as shown in Fig.16a.

4. Adding a second feature will completely specify each sample so this will decrease both the training error and the true error as shown in Fig.16b.

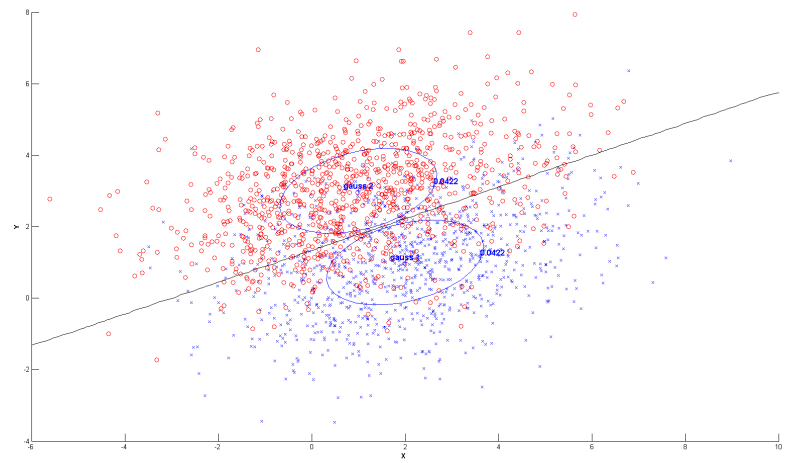


Figure 14: Decision line for two classes and normally distributed vectors with P(ω_1)=0.5 P(ω_2)=0.5 and Σ_1=Σ_2

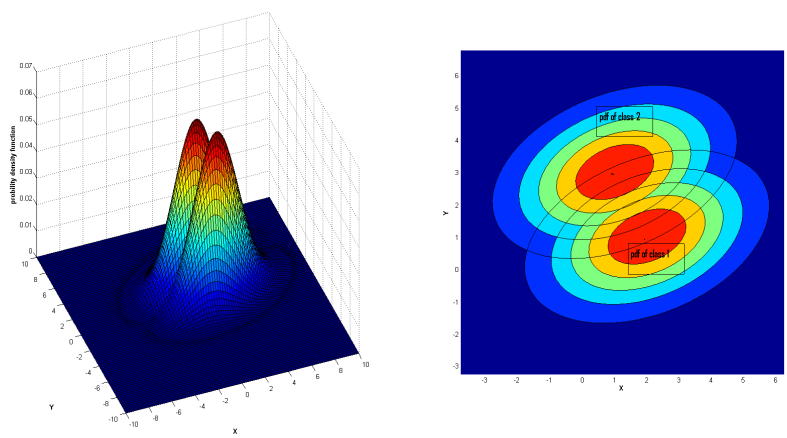
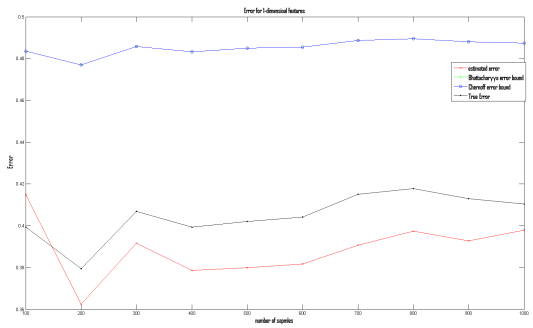
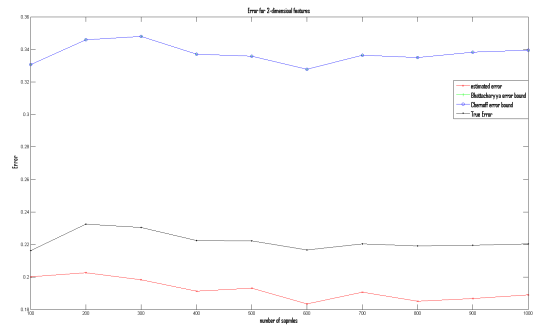


Figure 15: 3D representation of the Likelihood p.d.f's with P(ω_1)=0.5 P(ω_2)=0.5 and Σ_1=Σ_2

5. Notice that the true error is larger than the training error and this can be clarified by looking at Fig.14 which shows how the decision line split the points into two sets and the number of points that correspond to the opposite class in each region is very small, which leads to small training



(a) Error Analysis for single feature



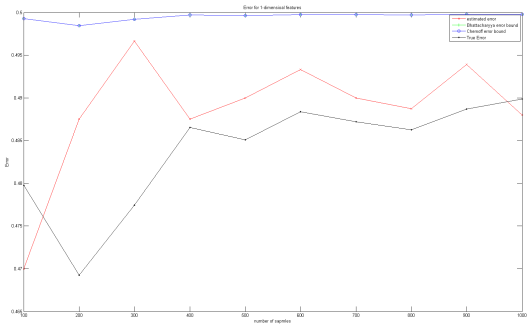
(b) Error Analysis for 2 features.

Figure 16: Error Analysis in 2D

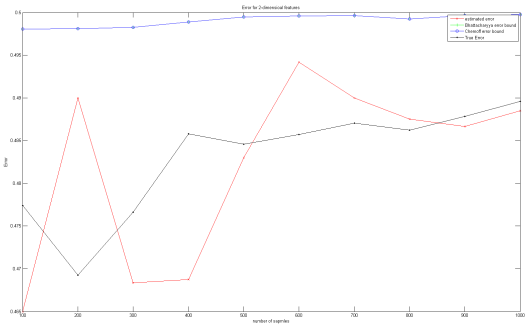
error, while the true error depends on the overlapping area which is shown in Fig.15.

4.2 d=10

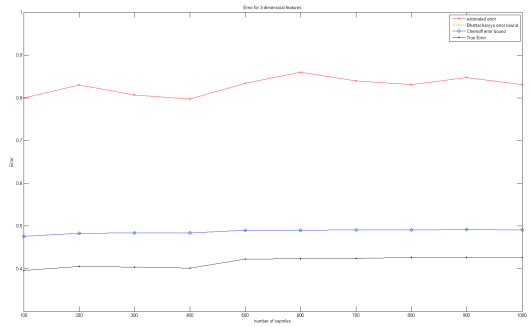
In this case the feature space is in $\boldsymbol{R}^{10}$, and the mean vectors were generated randomly so that each component is between 0 and 1, the results of the error analysis are shown in Fig.17,we can notice how the true error is still bounded by both Chernoff and Bhattacharyya bounds, while the training error is no longer bounded and this is expected because the dimension of the original data is 10 so using a number of features that is less than 10 will not allow us to classify a given point, also we can notice that the training error is decreasing as the number of features used in the classification increase with some exception that happen between Fig.17b,and Fig.17c and this mean that the 3'rd feature did not help in classification and thats why the training error has increased.



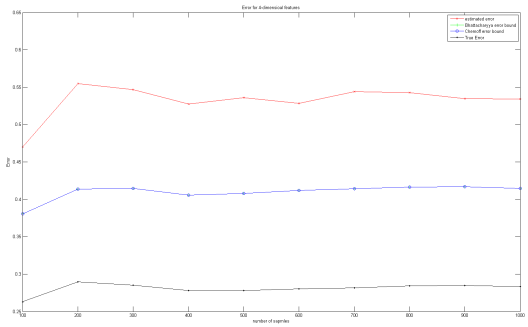
(a) Error Analysis for single feature



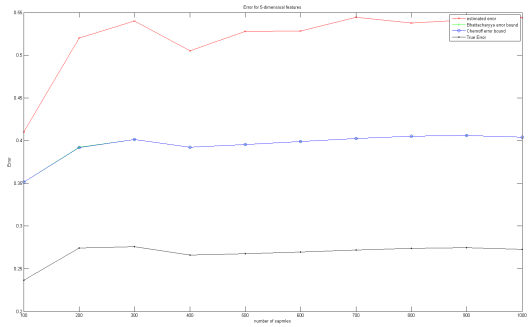
(b) Error Analysis for 2 features.



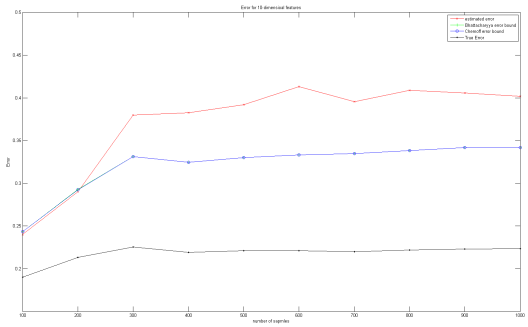
(c) Error Analysis for 3 features.



(d) Error Analysis for 4 features.



(e) Error Analysis for 5 features.



(f) Error Analysis for 10 features.

Figure 17: Error Analysis in 10D

5 Central Limit Theorem

The Central Limit Theorem says:

given n independent and identically distributed random variables $X_1, X_2, ..., X_n$ with mean μ and finite variance σ^2, Then the random variable

$$Z_n = \frac{\sum_{i=1}^{n} X_i - n\mu}{\sigma\sqrt{n}}$$

Converge in density to a normal random variable with zero mean and standard deviation equal one.

An alternative way to look at the Central Limit Theorem, is through the sample mean convergence, in other words The sample mean itself is a random variable represented by :

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

and as n increase the sample mean converges to a normal random variable with mean equal μ and variance equal $\frac{\sigma^2}{n}$.

To illustrate the Central Limit Theorem, we will consider a population of $\boldsymbol{N}$ i.i.d random values with mean μ and finite standard deviation σ, and suppose that we draw random samples of size $\boldsymbol{n}$ from this population, such that as we draw samples from the population each sample has a mean, and our objective is to show that the distribution of the sample means is normal regardless of the population distribution.
Now we will examine two cases to illustrate the Central limit Theorem.

5.1 Uniform Population

Assume that our population consists of 512000 i.i.d random values that were drawn from a uniform distribution in the range of $[1, 5]$ as shown in Fig.18, and assume that we start to draw 1000 samples in an increasing sizes such that the first 1000 samples each sample is of size n =1, then the second 1000 samples has n =2, and so on until we draw 1000 samples with n =9, as n increases the distribution of the sample means should approach a normal distribution with mean $\mu = 3$ and standard deviation $\overline{\sigma} = \frac{\sqrt{3}}{2\sqrt{n}}$.
and at n = 512 $\overline{\sigma} = \frac{\sqrt{3}}{2\sqrt{512}} = 0.03827$
To show this convergence, a histogram representation was used to represents the distribution of the points in 1000 samples for different sizes as shown in Fig.19.

Another excellent way to show the convergence of the sample means is by draw the normal plot, as shown in Fig.20, notice here that curve in blue rep-

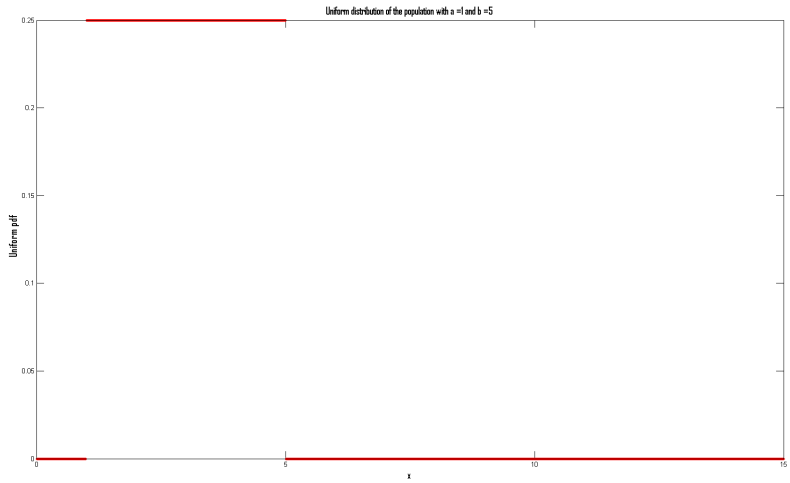


Figure 18: Uniform Probability density function U [1,5]

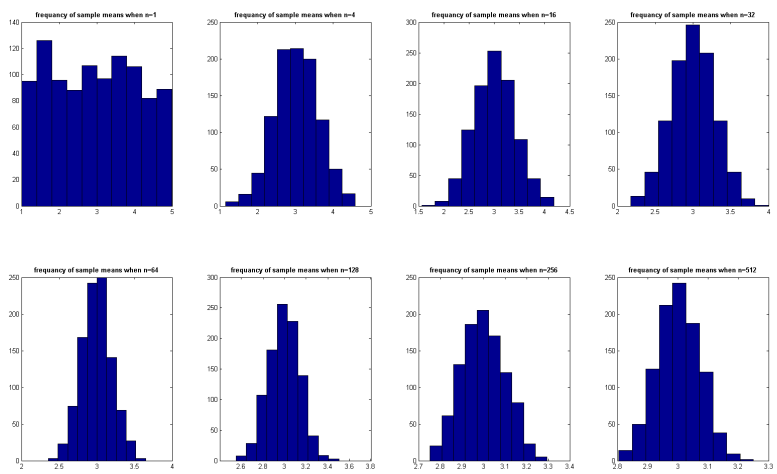


Figure 19: Histograms of Uniform sample means

resents the sample means and they are not coincide with the normal line "in red" because their original distribution was not normal, but as n increases their curve tend to coincide with the normal line.

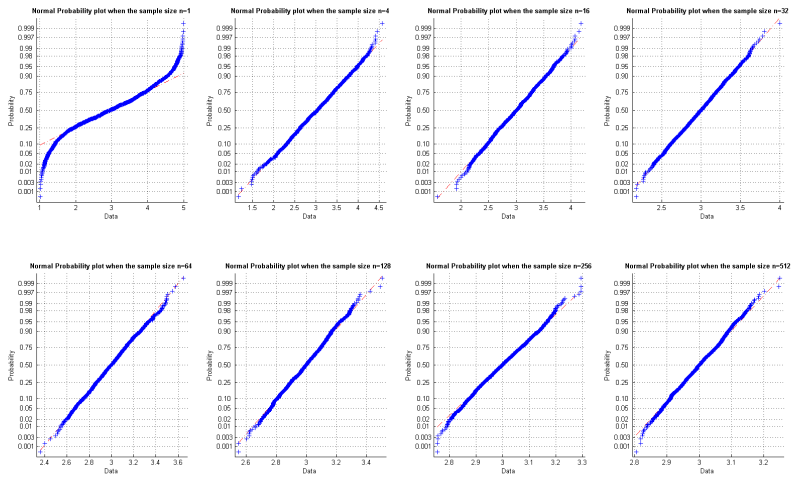


Figure 20: Normal Plot of Uniform sample means

5.2 Chi-Squared Population

Here we repeat the previous experiment but instead of using a uniform population we use a Chi-Squared population with 4 degree of freedom as shown in Fig.21 the histogram and the normal plots are shown in Fig.22, and Fig.23.

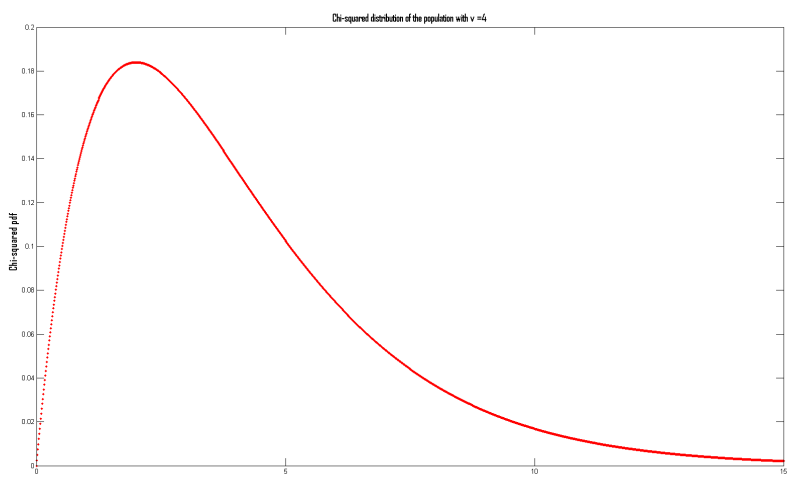


Figure 21: Chi-Squared Probability density function 4 DOF

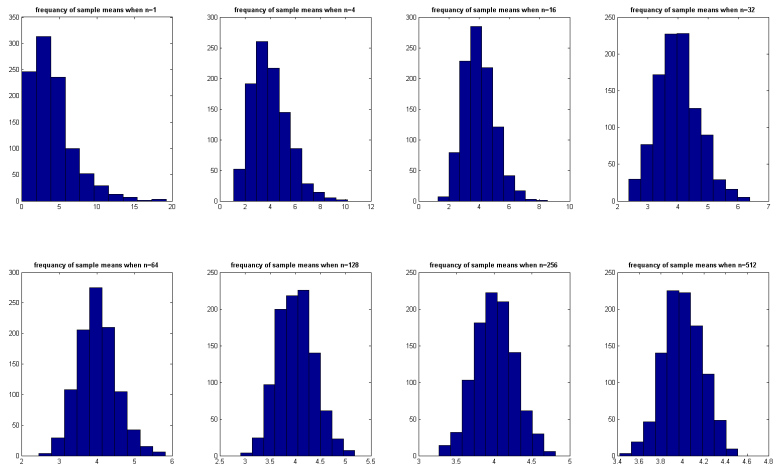


Figure 22: Histograms of Chi-Squared sample means

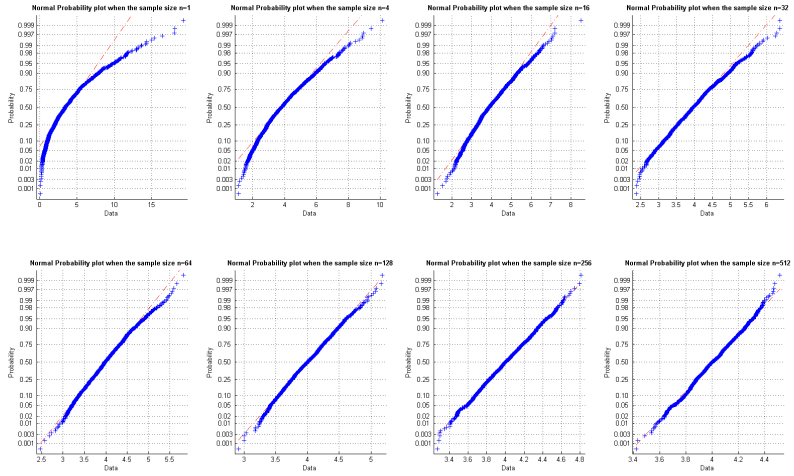


Figure 23: Normal Plot of Chi-Squared sample means

So no matter what was the original distribution the sample means will converge to a normal distribution, i have added a third case on Rhea that clarify the CLT when the population distribution is exponential.

6 Appendix A: source Code

6.1 Source code for Section2

```
function [out] = ChernoffBound(mu1,mu2,sigma1,sigma2,p1,p2)
objfunc = @(B) exp(-1*((B*(1-B)/2)*(mu2-mu1)'*...
    ((B*sigma1)+((1-B)*sigma2))^-1 ...
                    *(mu2-mu1)+ ...
    (0.5)*log(...
                (det(B*sigma1+(1-B)*sigma2))/...
                            ((det(sigma1)^B)*(det(sigma2)^(1-B)))...
              )));

% beta = fminbnd(objfunc,0,1,optimset('Display','iter'));%uncomment this
% line and comment the follwing line to see the details of the
% optimization.
beta = fminbnd(objfunc,0,1);

kBeta = exp(-1*((beta*(1-beta)/2)*(mu2-mu1)'*...
    ((beta*sigma1)+((1-beta)*sigma2))^-1 ...
                    *(mu2-mu1)+ ...
    (0.5)*log(...
                (det(beta*sigma1+(1-beta)*sigma2))/...
                            ((det(sigma1)^beta)*(det(sigma2)^(1-beta)))...
              )));
CherBound = (p1^beta)*(p2^(1-beta))*kBeta;

out = CherBound;
```

```
function [upperBound] = Bhattacharyya(mu1,Sigma1,P1,mu2,Sigma2,P2)
% compute the bhattacharyya error bound according to Eq 74 and 75 in DHS
% for classification of features drawn from 2 normal densities.
% Inputs:
%   mu1: mean vector of class 1
%   mu2: mean vector of class 2
%   Sigma1: covariance matrix of class1
%   Sigma2: covariance matrix of class2
%   P1:prior probability of class1
%   P2:prior probability of class2
% Output:
%   upperBound: Bhattacharyya upper bound on the error rate

mu1 = mu1(:);
mu2 = mu2(:);
% Eq.75
SumOfSigma = 0.5*(Sigma1+Sigma2);
SumOfSigmaInv = inv( SumOfSigma );
```

```
% evaluate k at beta = 0.5
k = (1/8)*( (mu2-mu1)' * SumOfSigmaInv * (mu2-mu1) ) + ...
           0.5*log( det(SumOfSigma)/sqrt(det(Sigma1)*det(Sigma2)) );

% Eq.74
upperBound = sqrt(P1*P2)*exp(-k);

function [out] = GenSampleGsussian(d,Mu,Sigma,nSamples)
% this function generate a normally distributed random vectors
% Inputs :
%           d: dimension of the vector
%           Mu: mean vector
%           Sigma: covariance matrix "symmetric positive definite"
%           nSamples: number of samples to generate
Mu = Mu(:);
% % Compute the Cholesky factorization:
% [R] = chol(Sigma);
% % Compute the samples:
% %this is according to matlab documentation.
% out = repmat(Mu,1,nSamples)+ R * randn(d,nSamples);
% get dimension
dim = length(Mu);

% compute eigen values and vectors
[U,L] = eig(Sigma);
% dewhitening transform
out = inv(U')*sqrt(L)*randn(dim,nSamples)+repmat(Mu,1,nSamples);
return;

% calculate the prob. of error for 1-D case at different values of the
% prior probability.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
close all;
% note here if any of these parameters is changed then change the
% correspondence parameter in the solve(...) equation.
mu1 = 0;
mu2 = 1;
var1 = 1;
var2 = 3;
p1 = 0.5;%prior of class1
p2 = 0.5;%prior of class2
% calculate the optimal descion boundary
s = solve('(1/sqrt(1))*exp((-1/2)*((x-0)^2)/1)*0.5 -(1/sqrt(3))*exp((-1/2)*((x-1)^2)/3)*0.5
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
```

```
if length(s) == 1%if the decision boundary equation has only one solution.
x_0 = [double(s(1));0];%threshold value .
x = -5:0.01:5;
posterior1 = (1/sqrt(2*pi))*(1/sqrt(var1))*exp((-1/2)*((x-mu1).^2)/var1)*p1;
posterior2 = (1/sqrt(2*pi))*(1/sqrt(var2))*exp((-1/2)*((x-mu2).^2)/var2)*p2;

% decide regions
t = x_0(1)-0.1;
post1 = (1/sqrt(2*pi))*(1/sqrt(var1))*exp((-1/2)*((t-mu1).^2)/var1)*p1;
post2 = (1/sqrt(2*pi))*(1/sqrt(var2))*exp((-1/2)*((t-mu2).^2)/var2)*p2;
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

figure;
hold  on
plot(x,posterior1,'r');
plot(x,posterior2,'b');
[ph,msg]=jbfill(x,posterior1,posterior2,rand(1,3),rand(1,3),0,rand(1,1));
grid on
line([x_0(1) x_0(1)],[x_0(2) .35],'Marker','.','LineStyle','-.','color','k')
colormap(cool)
if post2<post1%region 1 decide w1
    x = x_0(1):0.001:5;
    area(x,(1/sqrt(2*pi))*(1/sqrt(var1))*exp((-1/2)*((x-mu1).^2)/var1)*p1);
    x = -5:0.001:x_0(1);
    area(x,(1/sqrt(2*pi))*(1/sqrt(var2))*exp((-1/2)*((x-mu2).^2)/var2)*p2);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    disp('The prob. of error "True error"= ');
    % caculate the area in part 1 of the report.
    y1=@(x)(1/sqrt(2*pi))*(1/sqrt(var1))*exp((-1/2)*((x-mu1).^2)/var1)*p1;
    a1 = quad(y1,x_0(1),5);
    y2=@(x)(1/sqrt(2*pi))*(1/sqrt(var2))*exp((-1/2)*((x-mu2).^2)/var2)*p2;
    a2 = quad(y2,-5,x_0(1));
    Tarea = a1+a2;
    disp(Tarea);
else%region 2 decide w2
   x = x_0(1):0.001:5;
   area(x,(1/sqrt(2*pi))*(1/sqrt(var2))*exp((-1/2)*((x-mu2).^2)/var2)*p2);
   x = -5:0.001:x_0(1);
   area(x,(1/sqrt(2*pi))*(1/sqrt(var1))*exp((-1/2)*((x-mu1).^2)/var1)*p1);
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   disp('The prob. of error "True error"= ');
   % caculate the area in part 1 of the report.
   y1=@(x)(1/sqrt(2*pi))*(1/sqrt(var1))*exp((-1/2)*((x-mu1).^2)/var1)*p1;
   a1 = quad(y1,-5,x_0(1));
   y2=@(x)(1/sqrt(2*pi))*(1/sqrt(var2))*exp((-1/2)*((x-mu2).^2)/var2)*p2;
```

```
   a2 = quad(y2,x_0(1),5);
   Tarea = a1+a2;
   disp(Tarea);
end
xlabel('x');
ylabel('p(x|wi)P(wi)');
str = strcat('Threshold X_0 = ',num2str(x_0(1)));

%display information
text(x_0(1),0.359,str);
text(mu1-2*sqrt(var1), (1/sqrt(2*pi))*(1/sqrt(var1))*exp((-1/2)*(((mu1-sqrt(var1))-mu1)^2)/v
text(mu2+sqrt(var2),(1/sqrt(2*pi))*(1/sqrt(var2))*exp((-1/2)*(((mu2+sqrt(var2))-mu2)^2)/var2

% compute Chernoff and Bahtacharyya Error Bounds.
disp('The Chernoff Error Bound = ');
disp(ChernoffBound(mu1,mu2,var1,var2,p1,p2));

disp('The Battacharyya Error Bound = ');
disp(Bhattacharyya(mu1,var1,p1,mu2,var2,p2));

else % if the decision boundary equationhas 2 solutions
s1 = [double(s(1));double(s(2))];
x_0 = [min(s1);max(s1)];%threshold value .
x = -5:0.01:5;
posterior1 = (1/sqrt(2*pi))*(1/sqrt(var1))*exp((-1/2)*((x-mu1).^2)/var1)*p1;
posterior2 = (1/sqrt(2*pi))*(1/sqrt(var2))*exp((-1/2)*((x-mu2).^2)/var2)*p2;

% decide regions arround x_0(1)
t = x_0(1)-0.1;
post1 = (1/sqrt(2*pi))*(1/sqrt(var1))*exp((-1/2)*((t-mu1).^2)/var1)*p1;
post2 = (1/sqrt(2*pi))*(1/sqrt(var2))*exp((-1/2)*((t-mu2).^2)/var2)*p2;
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %

figure;
hold  on
plot(x,posterior1,'r');
plot(x,posterior2,'b');
[ph,msg]=jbfill(x,posterior1,posterior2,rand(1,3),rand(1,3),0,rand(1,1));
grid on
line([x_0(1) x_0(1)],[0 .35],'Marker','.','LineStyle','-.','color','k')
line([x_0(2) x_0(2)],[0 .35],'Marker','.','LineStyle','-.','color','k')
colormap(cool)
Tarea = 0;%total error area.
if post2<post1%region 1 decide w1
    x = x_0(1):0.001:x_0(2);
    area(x,(1/sqrt(2*pi))*(1/sqrt(var1))*exp((-1/2)*((x-mu1).^2)/var1)*p1);
```

```
    x = -5:0.001:x_0(1);
    area(x,(1/sqrt(2*pi))*(1/sqrt(var2))*exp((-1/2)*((x-mu2).^2)/var2)*p2);
    %%%%
```

```
   disp(Tarea);
end

xlabel('x');
ylabel('p(x|wi)P(wi)');
str1 = strcat('Threshold X_1 = ',num2str(x_0(1)));
str2 = strcat('Threshold X_2 = ',num2str(x_0(2)));

%display information
text(x_0(1),0.359,str1);
text(x_0(2),0.359,str2);
text(mu1-1.8*sqrt(var1), (1/sqrt(2*pi))*(1/sqrt(var1))*exp((-1/2)*(((mu1-sqrt(var1))-mu1).^2
text(mu2+sqrt(var2),(1/sqrt(2*pi))*(1/sqrt(var2))*exp((-1/2)*(((mu2+sqrt(var2))-mu2).^2)/var
% compute Chernoff and Bahtacharyya Error Bounds.
disp('The Chernoff Error Bound = ');
disp(ChernoffBound(mu1,mu2,var1,var2,p1,p2));
disp('The Battacharyya Error Bound = ');
disp(Bhattacharyya(mu1,var1,p1,mu2,var2,p2));
end
```

6.2 Source code for section3

```
function [out] = TrueError(mu1,mu2,sigma1,sigma2,P1,P2)
%Error Probability for Quadratic classifier
% h(X) = ln(p(X|w2) - ln(p(X|w1)) - ln(P1/P2)
%      = (0.5)*(X - mu1)'*inv(sigma1)*(X-mu1)-(0.5)*(X -
%      mu2)'*inv(sigma2)*(X-mu2) + 0.5*log(det(sigma1)/det(sigma(2)) -
%      log(P1/P2)

% test normal distribution pT(X) =
%  => (1/(2*pi)^(n/2))*(1/sqrt(sigmaT))*exp((-0.5)*((x-muT)'*inv(sigmaT)*(x-muT))/newVar)



%
% % step 1 : muT = mu1, and sigmaT=sigma1
% % apply simultanious diagonalization
% % A'*sigma1*A = I,A'*sigma2*A = mu
% C(:,:,1) = sigma2;
% A = qdiag(sigma1,C);
% % apply coordinate shift
% L = A'*(mu2-mu1);
% CSig = A'*(inv(sigma1) - inv(sigma2))*A;
% % converting mu1,mu2 to D1,D2 and sigma1 ,sigma2 to K1,K2
% muT = mu1;
% D1 = A'*(mu1-mu1);
```

```
% D2 = A'*(mu2-mu1);
% K1 = A'*sigma1*A;
% K2 = A'*sigma2*A;
% V = inv(K1)*D1 - inv(K2)*D2;
% c = (0.5)*(D1'*inv(K1)*D1 - D2'*inv(K2)*D2) + (0.5)*(log(det(K1)/det(K2))) - (log(P1/P2));
%
%
% lamda1 = 1- 1/K2(1,1);
% lamda2 = 1- 1/K2(2,2);
% v1 = -L(1)/K2(1,1);
% v2 = -L(2)/K2(2,2);
% j = sqrt(-1);
%
% FT1 = @(x)(sqrt(1-j.*x.*lamda1).^-1).*(sqrt(1-j.*x.*lamda2).^-1) .* ...
%     exp(((-0.5).*(((v1.^2).*(x.^2))./(1- j.*x.*lamda1)))+...
%    ((-0.5).*(((v2.^2).*(x.^2))./(1- j.*x.*lamda2))+ ...
%     (j.*x.*c))).*((j.*x).^-1);
% % compute eps1
% a1 = quad(FT1,-10000,-0.000001);
% a2 = quad(FT1,0.000001,10000);
% eps1 = 0.5 + (1/(2*pi))*(a1+a2);
%
% % step2 : muT = mu2, and sigmaT = sigma2
% D1 = (-1)*(K2.^(-0.5))*L;
% D2 = [0;0];
% K1 = inv(A'*sigma2*A);
% K2 = A'*sigma1*A;
% V = -1*((A'*sigma2*A)^0.5)*L;
% c = (0.5)*(L'*L) - (0.5)*(log(det(A'*sigma2*A))) - (log(P1/P2));
% capSig = inv(A'*sigma2*A - A'*sigma1*A);
% lamda1 = (capSig(1,1));
% lamda2 = (capSig(2,2));
% v1 = V(1);
% v2 = V(2);
% j = sqrt(-1);
%
% FT2 = @(x)(sqrt(1-j.*x.*lamda1).^-1).*(sqrt(1-j.*x.*lamda2).^-1) .* ...
%     exp(((-0.5).*(((v1.^2).*(x.^2))./(1- j.*x.*lamda1)))+...
%    ((-0.5).*(((v2.^2).*(x.^2))./(1- j.*x.*lamda2))+ ...
%     (j.*x.*c))).*((j.*x).^-1);
%
% % compute eps2
%
% a1 = quad(FT2,-10000,-0.000001);
% a2 = quad(FT2,0.000001,10000);
% eps2 = 0.5 - (1/(2*pi))*(a1+a2);
```

```
%
% % total probability of error
% Perror = P1*abs(eps1)+ P2*abs(eps2);



% %
nu1 = (-0.5)*(mu2 - mu1)'*inv(sigma1)*(mu2 - mu1);% - nu this is the mean of p(h|w1)
nu2 = (0.5)*(mu2 - mu1)'*inv(sigma2)*(mu2 - mu1);% + nu this is the mean of p(h|w2)
newVar = (mu2 - mu1)'*inv(sigma1)*(mu2 - mu1);% this is the same as 2*nu ,this is the varian
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% calculate the error according to the following formula:
% P(error) = P(w1)*eps1 + P(w2)*eps2
%  where eps1 = 1- erf((ln(P1/P2) +nu)/sqrt(newVar))
%        eps2 = erf((ln(P1/P2) - nu)/sqrt(newVar))

eps1 = 1- erf((log(P1/P2) -nu1)/sqrt(newVar))/(2*sqrt(2));
eps2 = erf((log(P1/P2)+ nu1)/sqrt(newVar))/(2*sqrt(2));
Perror = P1*eps1 + P2*eps2;
out = Perror;

% find the decision surface
clear all;
close all;
P1 = 0.5;
P2 = 0.5;
mu1 = [1;0];
mu2 = [0;0.5];
% Sigma1 = [3,1;1,2];
% Sigma2 = [4,1;1,3];
Sigma1 = [3,2;2,2];
Sigma2 = [4,0.3;0.3,1];
% Class 1
d =2;
nSamples = 125;
Class1 = struct('dimension',d,'mean',mu1,'Sigma',Sigma1,'nSample',nSamples...
    ,'Samples',GenSampleGsussian(d,mu1,Sigma1,nSamples));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Class 2
Class2 = struct('dimension',d,'mean',mu2,'Sigma',Sigma2,'nSample',nSamples...
    ,'Samples',GenSampleGsussian(d,mu2,Sigma2,nSamples));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Y(1:125) = 1;
Y(126:250) = 2;
data = struct('X',[Class1.Samples(:,1:125),Class2.Samples(:,1:125) ],'name','Finite data set
```

```
Cova(:,:,1) = Class1.Sigma;
Cova(:,:,2) = Class2.Sigma;
model = struct('Mean',[Class1.mean Class2.mean],'Cov',Cova,'Prior',[P1,P2],'y',Y,'cov_type'

gauss_model = mlcgmm(data);
quad_model = bayesdf(model);
ypred = quadclass(data.X,quad_model);
cerror(ypred,data.y)
figure; ppatterns(data);pgauss( model ); pboundary(quad_model);


[Ax1,Ay1] = meshgrid(linspace(-10,10,100), linspace(-10,10,100));
y1 = MultiVariateGauss([Ax1(:)';Ay1(:)'],mu1,Sigma1);
figure;
subplot(1,2,1)
hold on
surf( Ax1, Ay1, reshape(y1,100,100));
grid on;
xlabel('X');
ylabel('Y');
zlabel('probility density function');
axis square
subplot(1,2,2)
hold on
contourf(Ax1,Ay1,reshape(y1,100,100));
axis square
xlabel('X');
ylabel('Y');

[Ax2,Ay2] = meshgrid(linspace(-10,10,100), linspace(-10,10,100));
y2 = MultiVariateGauss([Ax2(:)';Ay2(:)'],mu2,Sigma2);
% figure;
subplot(1,2,1)
surf( Ax2, Ay2, reshape(y2,100,100));
axis square
subplot(1,2,2)
contourf(Ax2,Ay2,reshape(y2,100,100));
axis square
```

6.3 Source Code for section4

6.3.1 2 Dimensional case

```
clear all;
close all;
```

```
d = 2;
nSamples = 1000;
P1 = 0.5;
P2 = 0.5;
% Class 1
% Mu1 = rand( d,  1 );
Mu1 = [2;1];
% A1  = rand( d, d );
% Sigma1 = A1' * A1;
Sigma1 = [4,1;1,2];
Class1 = struct('dimension',d,'mean',Mu1,'Sigma',Sigma1,'nSample',nSamples...
    ,'Samples',GenSampleGsussian(d,Mu1,Sigma1,nSamples));

% Class 2
% Mu2 = rand( d,  1 );
Mu2 = [1;3];
% A2  = rand( d, d );
% Sigma2 = A2' * A2;
% Sigma2 = [1,0;0,1];
Sigma2 = [4,1;1,2];
Class2 = struct('dimension',d,'mean',Mu2,'Sigma',Sigma2,'nSample',nSamples...
    ,'Samples',GenSampleGsussian(d,Mu2,Sigma2,nSamples));

Y(1:1000) = 1;
Y(1001:2000) = 2;
data = struct('X',[Class1.Samples(:,1:1000),Class2.Samples(:,1:1000) ],'name','Finite data s
Cova(:,:,1) = Class1.Sigma;
Cova(:,:,2) = Class2.Sigma;
model = struct('Mean',[Class1.mean Class2.mean],'Cov',Cova,'Prior',[P1,P2],'y',Y,'cov_type',

gauss_model = mlcgmm(data);
quad_model = bayesdf(model);
ypred = quadclass(data.X,quad_model);
% cerror(ypred,data.y)
figure; ppatterns(data);pgauss( model ); pboundary(quad_model);


[Ax1,Ay1] = meshgrid(linspace(-10,10,100), linspace(-10,10,100));
y1 = MultiVariateGauss([Ax1(:)';Ay1(:)'],Mu1,Sigma1);
figure;
subplot(1,2,1)
hold on
surf( Ax1, Ay1, reshape(y1,100,100));
grid on;
xlabel('X');
ylabel('Y');
```

```
zlabel('probility density function');
axis square
subplot(1,2,2)
hold on
contourf(Ax1,Ay1,reshape(y1,100,100));
axis square
xlabel('X');
ylabel('Y');

[Ax2,Ay2] = meshgrid(linspace(-10,10,100), linspace(-10,10,100));
y2 = MultiVariateGauss([Ax2(:)';Ay2(:)'],Mu2,Sigma2);
% figure;
subplot(1,2,1)
surf( Ax2, Ay2, reshape(y2,100,100));
axis square
subplot(1,2,2)
contourf(Ax2,Ay2,reshape(y2,100,100));
axis square

OriginalPatterns = [Class1.Samples,Class2.Samples];
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% Compute statistics
disp( 'Using One features...' );
% patterns = OriginalPatterns( 1:2, : );        % extract THE FIRST FEATURE
incr = 100;
start = 100;
ErrArr1 = zeros(1,1000/incr);
TrueErr1 = zeros(1,1000/incr);
bhaBoundArr1 = zeros(1,1000/incr);
cherBoundArr1 = zeros(1,1000/incr);
Mues1 = zeros(2,1000/incr);
for n=start:incr:1000
patterns = [Class1.Samples(1,1:n),Class2.Samples(1,1:n)];
patterns = reshape( patterns, [ n, 2 ] );


mus =  mean(patterns );
Mues1(:,n/incr) = mus;
sigmas = std( patterns).^2;
allPats = patterns(:).';
discV = DiscriminantFunNormalDensity(allPats,mus,sigmas);
[ maxdisc, classIndx ] = max( discV );

estErr = sum( [ classIndx(1:n)~=1, classIndx(n+1:2*n)~=2 ] )/(2*n);%size(patterns,2);
ErrArr1(n/incr)=estErr;
disp( [ 'The sample estimated error rate is given by: ', num2str(estErr) ] );
```

```
bhaBound = Bhattacharyya(mus(1),sigmas(1),0.5,mus(2),sigmas(2),0.5);
bhaBoundArr1(n/incr) = bhaBound;
% bhaBound = Bhattacharyya(Mu1,Sigma1,0.5,Mu2,Sigma2,0.5);
cherBound = ChernoffBound(mus(1),mus(2),sigmas(1),sigmas(2),0.5,0.5);
cherBoundArr1(n/incr) = cherBound;
disp( [ 'The Bhattacharyya error bound is given by: ', num2str(bhaBound) ] );
disp( [ 'The Chernoff error bound is given by: ', num2str(cherBound) ] );

trueError = TrueError(mus(1),mus(2),sigmas(1),sigmas(2),0.5,0.5);
TrueErr1(n/incr) = trueError;
disp( [ 'The error probability "The true Error "is : ', num2str(trueError) ] );
end
figure;

plot(start:incr:1000,ErrArr1,'-rx');
hold on
plot(start:incr:1000,bhaBoundArr1,'-g+');
hold on
plot(start:incr:1000,cherBoundArr1,'-bo');
hold on
plot(start:incr:1000,TrueErr1,'-k.');
title('Error for 1-dimensioal features');
xlabel('number of sapmles');
ylabel('Error');
legend('estimated error','Bhattacharyya error bound','Chernoff error bound','True Error');




% Compute statistics
disp( 'Using TWO features...' );
% patterns = OriginalPatterns( 1:2, : );              % extract THE FIRST TWO FEATURES
mus = zeros(2,2);
sigmas = zeros(2,2,2);
incr = 100;
start = 100;
ErrArr2 = zeros(1,1000/incr);
TrueErr2 = zeros(1,1000/incr);
bhaBoundArr2 = zeros(1,1000/incr);
cherBoundArr2 = zeros(1,1000/incr);
Mues2 = zeros(2,2,1000/incr);
for n=start:incr:1000
patterns = [Class1.Samples(1:2,1:n),Class2.Samples(1:2,1:n)];
newPat = zeros(n,2,2);
newPat(:,1,1) = Class1.Samples(1, 1:n).';
newPat(:,2,1) = Class1.Samples(2, 1:n).';
```

```
newPat(:,1,2) = Class2.Samples(1,1:n).';
newPat(:,2,2) = Class2.Samples(2,1:n).';


mus(:,1) = mean( newPat(:,:,1) ).';
mus(:,2) = mean( newPat(:,:,2) ).';
Mues2(:,:,n/incr) = mus;
sigmas(:,:,1) = cov( newPat(:,:,1) );
sigmas(:,:,2) = cov( newPat(:,:,2) );

discV = DiscriminantFunNormalDensity(patterns,mus,sigmas);
[ maxdisc, classIndx ] = max( discV );

estErr = sum( [ classIndx(1:n)~=1, classIndx(n+1:2*n)~=2 ] )/(2*n);%size(patterns,2);
ErrArr2(n/incr)=estErr;
disp( [ 'The sample estimated error rate is given by: ', num2str(estErr) ] );
bhaBound = Bhattacharyya(mus(:,1),sigmas(:,:,1),0.5,mus(:,2),sigmas(:,:,2),0.5);
bhaBoundArr2(n/incr) = bhaBound;

% bhaBound = Bhattacharyya(Mu1,Sigma1,0.5,Mu2,Sigma2,0.5);
disp( [ 'The Bhattacharyya error bound is given by: ', num2str(bhaBound) ] );

cherBound = ChernoffBound(mus(:,1),mus(:,2),sigmas(:,:,1),sigmas(:,:,2),0.5,0.5);
cherBoundArr2(n/incr) = cherBound;

disp( [ 'The Chernoff error bound is given by: ', num2str(cherBound) ] );
trueError = TrueError(mus(:,1),mus(:,2),sigmas(:,:,1),sigmas(:,:,2),0.5,0.5);
TrueErr2(n/incr) = trueError;
disp( [ 'The error probability "The true Error "is : ', num2str(trueError) ] );

end
figure;
plot(start:incr:1000,ErrArr2,'-rx');
hold on
plot(start:incr:1000,bhaBoundArr2,'-g+');
hold on
plot(start:incr:1000,cherBoundArr2,'-bo');
hold on
plot(start:incr:1000,TrueErr2,'-k.');

title('Error for 2-dimensioal features');
xlabel('number of sapmles');
ylabel('Error');
legend('estimated error','Bhattacharyya error bound','Chernoff error bound','True Error');
```

6.3.2 10 Dimensional case

```
clear all;
close all;
d = 10;
nSamples = 1000;
P1 = 0.5;
P2 = 0.5;
% Class 1
%  a + (b-a).*rand(100,1)
Mu1 = rand( d,  1 );
% Mu1 = [2;1];
A1  = rand( d, d );
Sigma1 = A1' * A1;
% Sigma1 = [4,1;1,2];
% Class1 = struct('dimension',d,'mean',Mu1,'Sigma',Sigma1,'nSample',nSamples...
%     ,'Samples',GenSampleGsussian(d,Mu1,Sigma1,nSamples));

% Class 2
Mu2 = rand( d,  1 );
% Mu2 = [1;3];
A2  = rand( d, d );
Sigma2 = A2' * A2;
% Sigma2 = [1,0;0,1];
Sigma2 = Sigma1;
% Class2 = struct('dimension',d,'mean',Mu2,'Sigma',Sigma2,'nSample',nSamples...
%     ,'Samples',GenSampleGsussian(d,Mu2,Sigma2,nSamples));
load('K:\Report#1\Data\goodData10Dim2.mat');
OriginalPatterns = [Class1.Samples,Class2.Samples];
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
% Compute statistics
disp( 'Using One features...' );
% patterns = OriginalPatterns( 1:2, : );        % extract THE FIRST FEATURE
incr = 100;
start = 100;
ErrArr1 = zeros(1,1000/incr);
TrueErr1 = zeros(1,1000/incr);
bhaBoundArr1 = zeros(1,1000/incr);
cherBoundArr1 = zeros(1,1000/incr);
Mues1 = zeros(2,1000/incr);
for n=start:incr:1000
patterns = [Class1.Samples(1,1:n),Class2.Samples(1,1:n)];
patterns = reshape( patterns, [ n, 2 ] );


mus =  mean(patterns );
```

```
Mues1(:,n/incr) = mus;
sigmas = std( patterns).^2;
allPats = patterns(:).';
discV = DiscriminantFunNormalDensity(allPats,mus,sigmas);
[ maxdisc, classIndx ] = max( discV );

estErr = sum( [ classIndx(1:n)~=1, classIndx(n+1:2*n)~=2 ] )/(2*n);%size(patterns,2);
ErrArr1(n/incr)=estErr;
disp( [ 'The sample estimated error rate is given by: ', num2str(estErr) ] );
bhaBound = Bhattacharyya(mus(1),sigmas(1),0.5,mus(2),sigmas(2),0.5);
bhaBoundArr1(n/incr) = bhaBound;
% bhaBound = Bhattacharyya(Mu1,Sigma1,0.5,Mu2,Sigma2,0.5);
cherBound = ChernoffBound(mus(1),mus(2),sigmas(1),sigmas(2),0.5,0.5);
cherBoundArr1(n/incr) = cherBound;
disp( [ 'The Bhattacharyya error bound is given by: ', num2str(bhaBound) ] );
disp( [ 'The Chernoff error bound is given by: ', num2str(cherBound) ] );

trueError = TrueError(mus(1),mus(2),sigmas(1),sigmas(2),0.5,0.5);
TrueErr1(n/incr) = trueError;
disp( [ 'The error probability "The true Error "is : ', num2str(trueError) ] );
end
figure;

plot(start:incr:1000,ErrArr1,'-rx');
hold on
plot(start:incr:1000,bhaBoundArr1,'-g+');
hold on
plot(start:incr:1000,cherBoundArr1,'-bo');
hold on
plot(start:incr:1000,TrueErr1,'-k.');
title('Error for 1-dimensioal features');
xlabel('number of sapmles');
ylabel('Error');
legend('estimated error','Bhattacharyya error bound','Chernoff error bound','True Error');


% Compute statistics
disp( 'Using TWO features...' );
% patterns = OriginalPatterns( 1:2, : );              % extract THE FIRST TWO FEATURES
mus = zeros(2,2);
sigmas = zeros(2,2,2);
incr = 100;
start = 100;
ErrArr2 = zeros(1,1000/incr);
TrueErr2 = zeros(1,1000/incr);
bhaBoundArr2 = zeros(1,1000/incr);
```

```
cherBoundArr2 = zeros(1,1000/incr);
Mues2 = zeros(2,2,1000/incr);
for n=start:incr:1000
patterns = [Class1.Samples(1:2,1:n),Class2.Samples(1:2,1:n)];
newPat = zeros(n,2,2);
newPat(:,1,1) = Class1.Samples(1, 1:n).';
newPat(:,2,1) = Class1.Samples(2, 1:n).';
newPat(:,1,2) = Class2.Samples(1,1:n).';
newPat(:,2,2) = Class2.Samples(2,1:n).';


mus(:,1) = mean( newPat(:,:,1) ).';
mus(:,2) = mean( newPat(:,:,2) ).';
Mues2(:,:,n/incr) = mus;
sigmas(:,:,1) = cov( newPat(:,:,1) );
sigmas(:,:,2) = cov( newPat(:,:,2) );

discV = DiscriminantFunNormalDensity(patterns,mus,sigmas);
[ maxdisc, classIndx ] = max( discV );

estErr = sum( [ classIndx(1:n)~=1, classIndx(n+1:2*n)~=2 ] )/(2*n);%size(patterns,2);
ErrArr2(n/incr)=estErr;
disp( [ 'The sample estimated error rate is given by: ', num2str(estErr) ] );
bhaBound = Bhattacharyya(mus(:,1),sigmas(:,:,1),0.5,mus(:,2),sigmas(:,:,2),0.5);
bhaBoundArr2(n/incr) = bhaBound;

% bhaBound = Bhattacharyya(Mu1,Sigma1,0.5,Mu2,Sigma2,0.5);
disp( [ 'The Bhattacharyya error bound is given by: ', num2str(bhaBound) ] );

cherBound = ChernoffBound(mus(:,1),mus(:,2),sigmas(:,:,1),sigmas(:,:,2),0.5,0.5);
cherBoundArr2(n/incr) = cherBound;

disp( [ 'The Chernoff error bound is given by: ', num2str(cherBound) ] );
trueError = TrueError(mus(:,1),mus(:,2),sigmas(:,:,1),sigmas(:,:,2),0.5,0.5);
TrueErr2(n/incr) = trueError;
disp( [ 'The error probability "The true Error "is : ', num2str(trueError) ] );

end
figure;
plot(start:incr:1000,ErrArr2,'-rx');
hold on
plot(start:incr:1000,bhaBoundArr2,'-g+');
hold on
plot(start:incr:1000,cherBoundArr2,'-bo');
hold on
plot(start:incr:1000,TrueErr2,'-k.');
```

```
title('Error for 2-dimensioal features');
xlabel('number of sapmles');
ylabel('Error');
legend('estimated error','Bhattacharyya error bound','Chernoff error bound','True Error');


% Compute statistics
disp( 'Using Three features...' );
% patterns = OriginalPatterns( :, : );              % extract first 3 FEATURES
mus = zeros(3,2); %size[dimension,# of classes]
sigmas = zeros(3,3,2); %size[dimension,dimension,# of classes]
ErrArr = zeros(1,10);%size[1,#of increments]
incr = 100;
start = 100;
ErrArr3 = zeros(1,1000/incr);
TrueErr3 = zeros(1,1000/incr);
bhaBoundArr3 = zeros(1,1000/incr);
cherBoundArr3 = zeros(1,1000/incr);
for n=start:incr:1000
patterns = [Class1.Samples(1:3,1:n),Class2.Samples(1:3,1:n)];

newPat = zeros(n,3,2);
newPat(:,1,1) = Class1.Samples(1, 1:n).';
newPat(:,2,1) = Class1.Samples(2, 1:n).';
newPat(:,3,1) = Class1.Samples(3, 1:n).';

newPat(:,1,2) = Class2.Samples(1,1:n).';
newPat(:,2,2) = Class2.Samples(2,1:n).';
newPat(:,3,2) = Class2.Samples(3,1:n).';


mus(:,1) = mean( newPat(:,:,1) ).';
mus(:,2) = mean( newPat(:,:,2) ).';

sigmas(:,:,1) = cov( newPat(:,:,1) );
sigmas(:,:,2) = cov( newPat(:,:,2) );

discV = DiscriminantFunNormalDensity(patterns,mus,sigmas);
[ maxdisc, classIndx ] = max( discV );

estErr = sum( [ classIndx(1:n)~=1, classIndx(n+1:2*n)~=2 ] )/n;%size(patterns,2);
ErrArr3(n/incr)=estErr;
disp( [ 'The sample estimated error rate is given by: ', num2str(estErr) ] );
bhaBound = Bhattacharyya(mus(:,1),sigmas(:,:,1),0.5,mus(:,2),sigmas(:,:,2),0.5);
bhaBoundArr3(n/incr) = bhaBound;
```

```
% bhaBound = Bhattacharyya(Mu1,Sigma1,0.5,Mu2,Sigma2,0.5);
disp( [ 'The Bhattacharyya error bound is given by: ', num2str(bhaBound) ] );
cherBound = ChernoffBound(mus(:,1),mus(:,2),sigmas(:,:,1),sigmas(:,:,2),0.5,0.5);
cherBoundArr3(n/incr) = cherBound;
disp( [ 'The Chernoff error bound is given by: ', num2str(cherBound) ] );
trueError = TrueError(mus(:,1),mus(:,2),sigmas(:,:,1),sigmas(:,:,2),0.5,0.5);
TrueErr3(n/incr) = trueError;
disp( [ 'The error probability "The true Error "is : ', num2str(trueError) ] );
end
figure;
plot(start:incr:1000,ErrArr3,'-rx');
hold on
plot(start:incr:1000,bhaBoundArr3,'-g+');
hold on
plot(start:incr:1000,cherBoundArr3,'-bo');
hold on
plot(start:incr:1000,TrueErr3,'-k.');


title('Error for 3-dimensioal features');
xlabel('number of sapmles');
ylabel('Error');
legend('estimated error','Bhattacharyya error bound','Chernoff error bound','True Error');


% Compute statistics
disp( 'Using four features...' );
% patterns = OriginalPatterns( :, : );              % extract first 4 FEATURES
mus = zeros(4,2); %size[dimension,# of classes]
sigmas = zeros(4,4,2); %size[dimension,dimension,# of classes]
ErrArr = zeros(1,10);%size[1,#of increments]
incr = 100;
start = 100;
ErrArr4 = zeros(1,1000/incr);
TrueErr4 = zeros(1,1000/incr);
bhaBoundArr4 = zeros(1,1000/incr);
cherBoundArr4 = zeros(1,1000/incr);
for n=start:incr:1000
patterns = [Class1.Samples(1:4,1:n),Class2.Samples(1:4,1:n)];

newPat = zeros(n,4,2);
for i=1:4
newPat(:,i,1) = Class1.Samples(i, 1:n).';
newPat(:,i,2) = Class2.Samples(i,1:n).';
end
```

```
mus(:,1) = mean( newPat(:,:,1) ).';
mus(:,2) = mean( newPat(:,:,2) ).';

sigmas(:,:,1) = cov( newPat(:,:,1) );
sigmas(:,:,2) = cov( newPat(:,:,2) );

discV = DiscriminantFunNormalDensity(patterns,mus,sigmas);
[ maxdisc, classIndx ] = max( discV );

estErr = sum( [ classIndx(1:n)~=1, classIndx(n+1:2*n)~=2 ] )/n;%size(patterns,2);
ErrArr4(n/incr)=estErr;
disp( [ 'The sample estimated error rate is given by: ', num2str(estErr) ] );
bhaBound = Bhattacharyya(mus(:,1),sigmas(:,:,1),0.5,mus(:,2),sigmas(:,:,2),0.5);
bhaBoundArr4(n/incr) = bhaBound;
% bhaBound = Bhattacharyya(Mu1,Sigma1,0.5,Mu2,Sigma2,0.5);
disp( [ 'The Bhattacharyya error bound is given by: ', num2str(bhaBound) ] );
cherBound = ChernoffBound(mus(:,1),mus(:,2),sigmas(:,:,1),sigmas(:,:,2),0.5,0.5);
cherBoundArr4(n/incr) = cherBound;
disp( [ 'The Chernoff error bound is given by: ', num2str(cherBound) ] );
trueError = TrueError(mus(:,1),mus(:,2),sigmas(:,:,1),sigmas(:,:,2),0.5,0.5);
TrueErr4(n/incr) = trueError;
disp( [ 'The error probability "The true Error "is : ', num2str(trueError) ] );
end
figure;
plot(start:incr:1000,ErrArr4,'-rx');
hold on
plot(start:incr:1000,bhaBoundArr4,'-g+');
hold on
plot(start:incr:1000,cherBoundArr4,'-bo');
hold on
plot(start:incr:1000,TrueErr4,'-k.');


title('Error for 4-dimensioal features');
xlabel('number of sapmles');
ylabel('Error');
legend('estimated error','Bhattacharyya error bound','Chernoff error bound','True Error');


% Compute statistics
disp( 'Using five features...' );
% patterns = OriginalPatterns( :, : );              % extract first 5 FEATURES
mus = zeros(5,2); %size[dimension,# of classes]
sigmas = zeros(5,5,2); %size[dimension,dimension,# of classes]
ErrArr = zeros(1,10);%size[1,#of increments]
incr = 100;
```

```
start = 100;
ErrArr5 = zeros(1,1000/incr);
TrueErr5 = zeros(1,1000/incr);
bhaBoundArr5 = zeros(1,1000/incr);
cherBoundArr5 = zeros(1,1000/incr);
for n=start:incr:1000
patterns = [Class1.Samples(1:5,1:n),Class2.Samples(1:5,1:n)];

newPat = zeros(n,5,2);
for i=1:5
newPat(:,i,1) = Class1.Samples(i, 1:n).';
newPat(:,i,2) = Class2.Samples(i,1:n).';
end

mus(:,1) = mean( newPat(:,:,1) ).';
mus(:,2) = mean( newPat(:,:,2) ).';

sigmas(:,:,1) = cov( newPat(:,:,1) );
sigmas(:,:,2) = cov( newPat(:,:,2) );

discV = DiscriminantFunNormalDensity(patterns,mus,sigmas);
[ maxdisc, classIndx ] = max( discV );

estErr = sum( [ classIndx(1:n)~=1, classIndx(n+1:2*n)~=2 ] )/n;%size(patterns,2);
ErrArr5(n/incr)=estErr;
disp( [ 'The sample estimated error rate is given by: ', num2str(estErr) ] );
bhaBound = Bhattacharyya(mus(:,1),sigmas(:,:,1),0.5,mus(:,2),sigmas(:,:,2),0.5);
bhaBoundArr5(n/incr) = bhaBound;
% bhaBound = Bhattacharyya(Mu1,Sigma1,0.5,Mu2,Sigma2,0.5);
disp( [ 'The Bhattacharyya error bound is given by: ', num2str(bhaBound) ] );
cherBound = ChernoffBound(mus(:,1),mus(:,2),sigmas(:,:,1),sigmas(:,:,2),0.5,0.5);
cherBoundArr5(n/incr) = cherBound;
disp( [ 'The Chernoff error bound is given by: ', num2str(cherBound) ] );
trueError = TrueError(mus(:,1),mus(:,2),sigmas(:,:,1),sigmas(:,:,2),0.5,0.5);
TrueErr5(n/incr) = trueError;
disp( [ 'The error probability "The true Error "is : ', num2str(trueError) ] );
end
figure;
plot(start:incr:1000,ErrArr5,'-rx');
hold on
plot(start:incr:1000,bhaBoundArr5,'-g+');
hold on
plot(start:incr:1000,cherBoundArr5,'-bo');
hold on
plot(start:incr:1000,TrueErr5,'-k.');
```

```
title('Error for 5-dimensioal features');
xlabel('number of sapmles');
ylabel('Error');
legend('estimated error','Bhattacharyya error bound','Chernoff error bound','True Error');




% Compute statistics
disp( 'Using 10 features...' );
% patterns = OriginalPatterns( :, : );              % extract all FEATURES
mus = zeros(d,2); %size[dimension,# of classes]
sigmas = zeros(d,d,2); %size[dimension,dimension,# of classes]
ErrArr = zeros(1,10);%size[1,#of increments]
incr = 100;
start = 100;
ErrArrd = zeros(1,1000/incr);
TrueErrd = zeros(1,1000/incr);
bhaBoundArrd = zeros(1,1000/incr);
cherBoundArrd = zeros(1,1000/incr);
for n=start:incr:1000
patterns = [Class1.Samples(1:d,1:n),Class2.Samples(1:d,1:n)];

newPat = zeros(n,d,2);
for i=1:d
newPat(:,i,1) = Class1.Samples(i, 1:n).';
% newPat(:,2,1) = Class1.Samples(2, 1:n).';
% newPat(:,3,1) = Class1.Samples(3, 1:n).';

newPat(:,i,2) = Class2.Samples(i,1:n).';
% newPat(:,2,2) = Class2.Samples(2,1:n).';
% newPat(:,3,2) = Class2.Samples(3,1:n).';
end

mus(:,1) = mean( newPat(:,:,1) ).';
mus(:,2) = mean( newPat(:,:,2) ).';

sigmas(:,:,1) = cov( newPat(:,:,1) );
sigmas(:,:,2) = cov( newPat(:,:,2) );

discV = DiscriminantFunNormalDensity(patterns,mus,sigmas);
[ maxdisc, classIndx ] = max( discV );

estErr = sum( [ classIndx(1:n)~=1, classIndx(n+1:2*n)~=2 ] )/n;%size(patterns,2);
```

```
ErrArrd(n/incr)=estErr;
disp( [ 'The sample estimated error rate is given by: ', num2str(estErr) ] );
bhaBound = Bhattacharyya(mus(:,1),sigmas(:,:,1),0.5,mus(:,2),sigmas(:,:,2),0.5);
bhaBoundArrd(n/incr) = bhaBound;
% bhaBound = Bhattacharyya(Mu1,Sigma1,0.5,Mu2,Sigma2,0.5);
disp( [ 'The Bhattacharyya error bound is given by: ', num2str(bhaBound) ] );
cherBound = ChernoffBound(mus(:,1),mus(:,2),sigmas(:,:,1),sigmas(:,:,2),0.5,0.5);
cherBoundArrd(n/incr) = cherBound;
disp( [ 'The Chernoff error bound is given by: ', num2str(cherBound) ] );
trueError = TrueError(mus(:,1),mus(:,2),sigmas(:,:,1),sigmas(:,:,2),0.5,0.5);
TrueErrd(n/incr) = trueError;
disp( [ 'The error probability "The true Error "is : ', num2str(trueError) ] );
end
figure;
plot(start:incr:1000,ErrArrd,'-rx');
hold on
plot(start:incr:1000,bhaBoundArrd,'-g+');
hold on
plot(start:incr:1000,cherBoundArrd,'-bo');
hold on
plot(start:incr:1000,TrueErrd,'-k.');


title('Error for d-dimensioal features');
xlabel('number of sapmles');
ylabel('Error');
legend('estimated error','Bhattacharyya error bound','Chernoff error bound','True Error');
```

6.4 Source code for Section5

```
function CLT(popDist,distParameters)
%
% Input:
%       popDist: a number represent the distribution of the population
%               1 : exponentioal distribution
%               2 : uniform distribution
%               3 : chi-square distribution
%               4 : normal distribution
%               5 : log normal distribuion
%
%        distParameters: this represents a vector that conatins the
%        parameters of the distribution
%               1 : exponentioal distribution [mean]
%               2 : uniform distribution  [min value; max value]
%               3 : Chi distribution [ v degrees of freedom]
%               4 : normal distribution [mean; standard deviation]
```

```
%               5 : log normal distribuion [mean; standard deviation]
% usage examples:
% example1: CLT(1,[5]);
% example2: CLT(2,[0;5]);
% example3: CLT(3,[5]);
% example4: CLT(4,[2;5])
% example5: CLT(5,[2;5]);
%


close all
nSamples=1000;

if(popDist == 1)%exponential
mu = distParameters(1);
nMax = 512;
setOfSamples = exprnd(mu,nSamples,nMax);
figure
X = 0:0.01:15;
plot(X, exppdf(X,mu),'r.');
strMu = num2str(mu);
str = strcat('Exponential distribution of the population with mean = ',strMu);
title( str);
xlabel('x');
ylabel('Exponential pdf');

elseif(popDist == 2)%uniform
a = distParameters(1);
b = distParameters(2);
nMax = 512;
setOfSamples = unifrnd(a,b,nSamples,nMax);
figure
X = 0:0.01:15;
plot(X, unifpdf(X,a,b),'r.');
stra = num2str(a);
strb = num2str(b);
str = strcat('Uniform distribution of the population with a = ',stra,' and b = ', strb);
title( str);
xlabel('x');
ylabel('Uniform pdf');

elseif(popDist == 3)%chi-squared
v =  distParameters(1);
nMax = 512;
setOfSamples = chi2rnd(v,nSamples,nMax);
figure
```

```
X = 0:0.01:15;
plot(X, chi2pdf(X,v),'r.');
strv = num2str(v);
str = strcat('Chi-squared distribution of the population with v = ',strv);
title( str);
xlabel('x');
ylabel('Chi-squared pdf');

elseif(popDist == 4)%normal
mu =  distParameters(1);
sigma = distParameters(2);
nMax = 512;
setOfSamples = normrnd(mu,sigma,nSamples,nMax);
figure
X = -4*sigma:0.01:4*sigma;
plot(X, normpdf(X,mu,sigma),'r.');
stra = num2str(mu);
strb = num2str(sigma);
str = strcat('Normal distribution of the population with mean = ',stra,' and standard deviat
title( str);
xlabel('x');
ylabel('Normal pdf');

elseif(popDist == 5)%log normal
mu =  distParameters(1);
sigma = distParameters(2);
nMax = 10000;
setOfSamples = normrnd(mu,sigma,nSamples,nMax);
figure
X = 0:0.001:4*sigma;
plot(X, normpdf(X,mu,sigma),'r.');
stra = num2str(mu);
strb = num2str(sigma);
str = strcat('LOG Normal distribution of the population with mean = ',stra,' and standard de
title( str);
xlabel('x');
ylabel('LOG Normal pdf');

end
sampleMeans = zeros(nSamples,1,9);
for i = 0:8
  sampleMeans(:,:,i+1) =   sum(setOfSamples(:,1:(2^(i))),2)/(2^(i));
end
figure
subplot(2,4,1), hist(sampleMeans(:,:,1)),title('frequancy of sample means when n=1')
subplot(2,4,2), hist(sampleMeans(:,:,3)),title('frequancy of sample means when n=4')
```

```
subplot(2,4,3), hist(sampleMeans(:,:,4)),title('frequancy of sample means when n=16')
subplot(2,4,4), hist(sampleMeans(:,:,5)),title('frequancy of sample means when n=32')
subplot(2,4,5), hist(sampleMeans(:,:,6)),title('frequancy of sample means when n=64')
subplot(2,4,6), hist(sampleMeans(:,:,7)),title('frequancy of sample means when n=128')
subplot(2,4,7), hist(sampleMeans(:,:,8)),title('frequancy of sample means when n=256')
subplot(2,4,8), hist(sampleMeans(:,:,9)),title('frequancy of sample means when n=512')
figure
subplot(2,4,1), normplot(sampleMeans(:,:,1)),title('Normal Probability plot when the sample
subplot(2,4,2), normplot(sampleMeans(:,:,3)),title('Normal Probability plot when the sample
subplot(2,4,3), normplot(sampleMeans(:,:,4)),title('Normal Probability plot when the sample
subplot(2,4,4), normplot(sampleMeans(:,:,5)),title('Normal Probability plot when the sample
subplot(2,4,5), normplot(sampleMeans(:,:,6)),title('Normal Probability plot when the sample
subplot(2,4,6), normplot(sampleMeans(:,:,7)),title('Normal Probability plot when the sample
subplot(2,4,7), normplot(sampleMeans(:,:,8)),title('Normal Probability plot when the sample
subplot(2,4,8), normplot(sampleMeans(:,:,9)),title('Normal Probability plot when the sample
```

References

[1] Keinosuke Fukunaga. *Statistical Pattern Recognition.* Academic press, 1990.

[2] Vaclav Hlavac Michal. I. Schlesinger. Statistical pattern recognition toolbox, 2000.

[3] David G. Stork Richard O. Duda, Peter E. Hart. *Patten Classification.* Wiely, 2000.

[4] Konstantinos Koutroumbas Sergios Theodoridis. *Pattern Recognition.* Elsevier, 2006.