Pattern Recognition Assignment No:1
Analysis & Results
18MCMI09
Krishnan.S

1.
A) Random samples from a Gaussian distribution given µ and ∑ in d dimensions was produced by using the Box Mueller Transform. A value is obtained from the normal distribution with mean 0 and variance 1/I and then it is multiplied by the given variance/covariance and added to the mean to obtain a random sample from the given distribution.
A 2D plot is shown in the code/output document to show that the random points generated are spread like Gaussian Noise. Changing the covariance changes the direction in which the data is spread and the mean changes the center.
The blue points have mean (3,1) and covariance ((1,0) (0,1)) meaning it has equal spread in both directions (x,y). The red points have mean (-1,-3) and covariance ((-4,0), (0,2)) meaning that it's spread more along x than y.

B) There were two different functions written for calculating the discriminant function; one for univariate and one for multivariate inputs.
The DF is given as

$$g_i(x) = -\frac{1}{2}(x-\mu_i)^t \Sigma^{-1}(x-\mu_i) - \frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln|\Sigma_i| + \ln P(\omega_i)$$

The function takes the input vector we want to classify as an input, along with the mean, variance/covariance and prior of the i^{th} class as inputs.
When given x = (0.3, 0.5, 0.4) as an input vector for the DF with mean = (0.1,0.4,0.8) and covariance = ((3,0,0) (0,3,0) (0,0,3)) and prior = 0.25, it returned a value of -5.826, which probably means that this input vector doesn't fit very well with the class represented by this DF.

The DF use the next two procedures for calculating distance.

C) Euclidean distance is given by the square root of the sum of the squared differences between two points.
Given two points x and y having n dimensions, the Euclidean distance between them is given as

$$D(x,y) = \sqrt{(x_1-y_1)^2 + (x_1-y_1)^2 + ...+ (x}_n - y_n)^2$$

Two points (3,0) and (0,3) were given, and the Euclidean distance between them was given as 4.242
Calculating using the above, sqrt(9 + 9) = 3 * sqrt(2) = 4.242

D) Mahalanobis distance is the measure of the distance between a point and a distribution.
Given two points/vectors x and y belonging to the same distribution, and the covariance of their distribution, the Mahalanobis distance between the two points is given by

$$D(x,y) = \sqrt{(x-y)^t \Sigma^{-1}(x-y)}$$

We use this to find the distance of a point from the mean of the distribution. We put µ in the above equation instead of y.

The same inputs for x, the mean and the covariance from the first part of this question (A) was given to the function. It returned the value 0.26, meaning that this point x is close to the center of the distribution with the given covariance.
If the covariance matrix is a diagonal matrix with equal values along the diagonal, this reduces down to Euclidean distance.

---

2. A Dichotomizer is used to discriminate between two classes or categories. The dichotomizers were implemented by finding the difference in the result returned by the DF for the given two classes and checking that difference. If the difference is greater than 0, it belongs to the first class. Otherwise it belongs to the second class. The dichotomizers were implemented to return the difference, the classification was done by the driver code for some flexibility later. The mean and covariance of the classes are calculated in the function call of the Dichotomizers since we're taking different features for each subdivision.

The distributions for the classes in each subdivision were plotted to see how they relate, assuming they are Gaussian. All the class distributions overlap each other a lot. These are shown in the code/output document.

A) Both classes have equal priors of (0.5). Only the first feature of both classes are used by the dichotomizer called FirstUVDichotomizer.
For the first class,

$$\mu_1 = -0.44$$
$$\sigma_1^2 = 14.38$$

For the second class,

$$\mu_2 = -0.543$$
$$\sigma_2^2 = 36.82$$

These were given to the Univariate DF and the difference was found for a point x.
A sample point x = (-5.5) was said to belong to class ω_1
The dichotomizer does not classify the points very well with just feature x1. So this feature isn't sufficient to classify the points.

B) The empirical training error for the dichotomizers are calculated as the percentage of points misclassified. We'll refer to this as the ETE.
The ETE for the one feature dichotomizer was calculated to be **50.0%**, meaning that it classified points correctly half the time.

C) The priors are kept the same. The features x1 and x2 are used by this dichotomizer, called FirstMVDichotomizer. It uses the Multivariate DF.
For the first class,

$$\mu_1 = \begin{bmatrix} -0.44 & -1.479 \end{bmatrix}$$
$$\Sigma_1 = \begin{bmatrix} 14.38 & 7.69 \\ 7.69 & 14.62 \end{bmatrix}$$

For the second class,

$$\mu_2 = \begin{bmatrix} -0.543 & -0.816 \end{bmatrix}$$
$$\Sigma_2 = \begin{bmatrix} 36.82 & 9.87 \\ 9.87 & 13.35 \end{bmatrix}$$

A sample point (0, 1.6) was said to belong to class ω_1

The ETE for the two feature dichotomizer was calculated to be **40.0%**. It does better than the dichotomizer that takes only one feature, but not by much.

D) Again, the priors are the same, but we use all three features x1, x2 and x3 to build the dichotomizer called SecondMVDichotomizer.

For the first class,

$$\mu_1 = \begin{bmatrix} -0.44 & -1.749 & -0.766 \end{bmatrix}$$

$$\Sigma_1 = \begin{bmatrix} 14.38 & 7.69 & 4.12 \\ 7.69 & 14.62 & 3.90 \\ 4.12 & 3.90 & 19.72 \end{bmatrix}$$

For the second class,

$$\mu_2 = \begin{bmatrix} -0.54 & -0.81 & -0.54 \end{bmatrix}$$

$$\Sigma_2 = \begin{bmatrix} 36.82 & 9.87 & -16.36 \\ 9.87 & 13.35 & 0.58 \\ -16.36 & 0.58 & 18.42 \end{bmatrix}$$

A sample point (-1, 3.4, 1) was said to belong to class ω_2

The ETE for the three feature dichotomizer was calculated to be **15.0%**. Using all three features improves the classification a lot compared to the other two features. I tried running the univariate DF using the other two features but they don't perform very well either. A combination of all three features manages to distinguish between class ω_1 and ω_2 well.

E) In this case, we're dealing with all three classes with uniform prior (0.33). Instead of using a dichotomizer, we compute the result of the DF for each of the three classes and then assign the input vector to the class whose discriminant function returned the greatest value. If we gave an input vector to the DF for classes ω_1, ω_2 and ω_3 and the DF for class ω_2 returned the maximum value, then we say that x belongs to ω_2. When implementing it, the Multivariate DF was called, passing the mean, covariance and prior of each class.

The mean and covariance of the first two classes have been given above. For the third class,

$$\mu_3 = \begin{bmatrix} 3.88 & 1.376 & 1.58 \end{bmatrix}$$

$$\Sigma_3 = \begin{bmatrix} 8.30 & 7.44 & 13.14 \\ 7.44 & 8.56 & 11.60 \\ 13.14 & 11.60 & 47.28 \end{bmatrix}$$

We gave the given points to the classifier called ThirdMVD. As a result,

Point (1,2,1) was classified as belonging to class ω_2

Point (5,3,2) was classified as belonging to class ω_3

Point (0,0,0) was classified as belonging to class ω_1

Point (1,0,0) was classified as belonging to class ω_1

This makes sense just by looking at the means of the three classes and their distances from the above points, since the priors are equal in this case. It is still dependent on covariance since that determines whether the point lies in the spread, but just by looking at the distance we can guess what class they belong to.

F) The priors were changed such that $P(\omega_1) = 0.8$, $P(\omega_2) = P(\omega_3) = 0.1$.
The result is that all points are now classified as belonging to class ω_1. Since the prior of the first class is much larger than the other two, the distance that this class occupies in the probability space is also much larger than the other two. The means of the second and third class lie much farther away from the first class, there is a lot of room for the first class, hence why all these points were classified as belonging to the first class. This makes sense as a higher prior means that the first class has a much higher probability of occurring.

---

3. The IRIS dataset consists of 150 four dimensional samples, corresponding to petal and sepal length and width. There are three classes, or three types of Iris, called Setosa, Versicolor and Virginica, each having 50 samples in the training set. The prior for the classes are hence uniform (0.33). The task is to build three different discriminant functions that take these features to classify inputs into one of these three classes, the first two being Linear (LDF) and the third one being Quadratic (QDF).

In the case of the LDF, there are two cases; one where all the classes have the same variance and the covariance matrix for all classes can be represented as $\sigma^2 I$, and the other case where all the classes have the same covariance matrix. The QDF is the third case where all classes have different covariance matrix.

For case II, the average of the three covariance matrices (one for each class) were taken and then made symmetric. And for case I, the average of the variances in this average covariance matrix was taken as σ^2.

The equation used for LDF Case I is given as,

$$g_i(x) = \frac{\mu_i^t x}{\sigma^2} - \frac{\mu_i^t \mu_i}{2\sigma^2} + \ln P(\omega_i)$$

Equation for LDF Case II is given as,

$$g_i(x) = (\Sigma^{-1}\mu_i)^t x - \frac{\mu_i^t \Sigma^{-1} \mu_i}{2} + \ln P(\omega_i)$$

Equation for QDF is given as,

$$g_i(x) = -\frac{x^t \Sigma_i^{-1} x}{2} + (\Sigma_i^{-1}\mu_i)^t x - \frac{\mu_i^t \Sigma_i^{-1} \mu_i}{2} - \frac{\ln|\Sigma_i|}{2} + \ln P(\omega_i)$$

A python class was made for each kind of DF. Three objects were made for each of the three cases of DFs that computed the DF (along with mean and covariance) for the three Iris classes. A ChangeCovariance method was implemented to change the covariance or variance to the average instead for case I and case II. Each class has a Result method which returns the value of the DF for a 4D input vector x. The Result method for each class implements the equations given above. It does not use the DF from 1(b).

For the Setosa class,

$$\mu_{Setosa} = \begin{bmatrix} 5 & 3.41 & 1.46 & 0.24 \end{bmatrix}$$

$$\Sigma_{Setosa} = \begin{bmatrix} 0.124 & 0.1 & 0.016 & 0.01 \\ 0.1 & 0.145 & 0.01 & 0.01 \\ 0.016 & 0.01 & 0.030 & 0.005 \\ 0.01 & 0.01 & 0.005 & 0.011 \end{bmatrix}$$

For the Versicolor class,

$$\mu_{Versicolor} = \begin{bmatrix} 5.93 & 2.77 & 4.26 & 1.32 \end{bmatrix}$$

$$\Sigma_{Versicolor} = \begin{bmatrix} 0.266 & 0.085 & 0.182 & 0.055 \\ 0.085 & 0.098 & 0.082 & 0.041 \\ 0.182 & 0.082 & 0.22 & 0.073 \\ 0.055 & 0.041 & 0.073 & 0.039 \end{bmatrix}$$

For the Virginica class,

$$\mu_{Virginica} = \begin{bmatrix} 6.58 & 2.97 & 5.55 & 2.02 \end{bmatrix}$$

$$\Sigma_{Virginica} = \begin{bmatrix} 0.404 & 0.093 & 0.303 & 0.049 \\ 0.093 & 0.104 & 0.071 & 0.047 \\ 0.303 & 0.071 & 0.304 & 0.048 \\ 0.049 & 0.047 & 0.048 & 0.075 \end{bmatrix}$$

The average covariance matrix for case II was calculated to be,

$$\Sigma_{avg} = \begin{bmatrix} 0.265 & 0.093 & 0.16 & 0.038 \\ 0.093 & 0.115 & 0.055 & 0.033 \\ 0.167 & 0.055 & 0.185 & 0.042 \\ 0.038 & 0.033 & 0.042 & 0.042 \end{bmatrix}$$

And the variance for case I was calculated to be,

$$\sigma^2 = 0.152$$

These were used to build the DFs.

The Empirical Training Error for each of the three classifiers were calculated.
For LDF Case I, the ETE was **7.3%**
For LDF Case II, the ETE was **66.6%**
For QDF, the ETE was **2.0%**
So the quadratic discriminant function manages to classify input vectors well compared to the other two. Curiously enough, case II seems to perform much worse compared to case I, which seems unintuitive. It's probably because the direction of the distribution changes a lot as compared to the covariances of the three classes individually.

The distributions of the three classes were plotted using the first three features to show how they relate (again, assuming they are normally distributed). They have no overlap at all and are neatly placed in three different regions. The plot is show in the code/output document.