BM 59D Homework #3

Kaan Oktay, *2016701108*

I. Introduction

In this homework, we were given two datesets, X and Y. Compared to the previous homework, datasets were multivariate in this homework. These datasets have two class labels, C_0 and C_1. Three dimensional visualization of these datasets, including both training and validation parts, can be seen in the Figure (1) and (2).

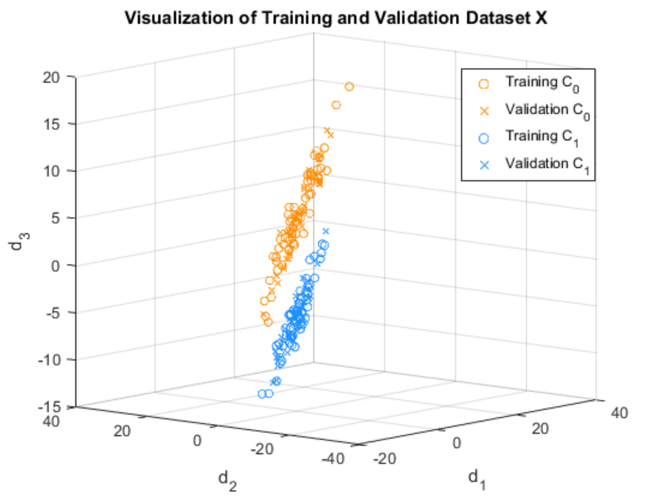

Fig. 1: Visualization of dataset X.

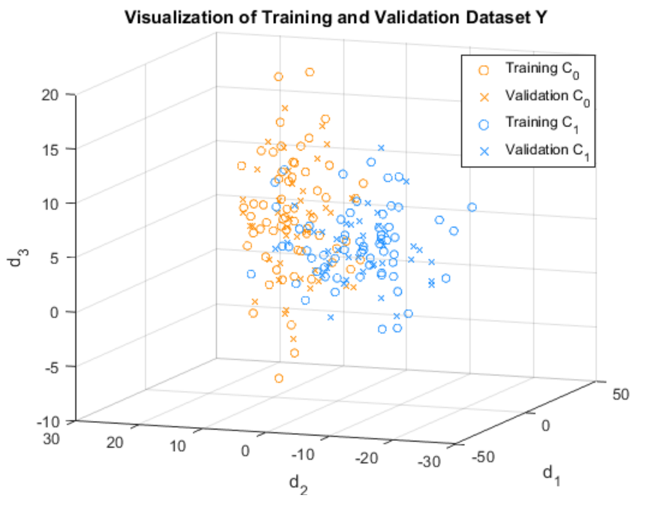

Fig. 2: Visualization of dataset Y.

As can be easily seen, without dimensionality reduction, discrimination of two classes is relatively easy in dataset X compared to dataset Y. In addition, training and validation sets lie approximately on the same plane for each class in dataset X. So we expect a better classification performance on this dataset without any assumptions for covariance matrices. However, when we apply dimensionality reduction or make assumptions about covariance matrices, we expect different results depending on the kind of reduction which will be discussed later.

Throughout this report, calculations for questions are performed and demonstrated firstly. Then confusion matrices for all questions are given in a separate section and these matrices are compared and discussed deeply in the discussion section.

II. Q1: Multivariate Classification

In this question, Bayesian decision theory is adopted to our two datasets for classification. Class priors are equal as below.

$$P(C_0) = P(C_1) = 0.5$$

Then class posteriors can be calculated as below.

$$P(C_i|\mathbf{x}) = \frac{P(C_i) \cdot p(\mathbf{x}|C_i)}{p(\mathbf{x})}$$

Then we can calculate the discriminant functions as below.

$$g_i(\mathbf{x}) = \log p(\mathbf{x}|C_i) + \log P(C_i) - \log p(\mathbf{x})$$

Here, we can omit evidence and prior terms because they do not vary between two classes. Thus it can be rewritten as below.

$$g_i(\mathbf{x}) = \log p(\mathbf{x}|C_i)$$

In addition, assuming Gaussian likelihoods for classes and using maximum likelihood estimators for the class parameters, we can express multivariate Gaussian distribution of a class likelihood as below

$$p(\mathbf{x}|C_i) = \frac{1}{\sqrt{(2\pi)^d |\mathbf{\Sigma}_i|}} \cdot \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T \mathbf{\Sigma}_i^{-1} (\mathbf{x}-\boldsymbol{\mu}_i)\right)$$

where $\boldsymbol{\mu}_i$ is class mean vector and $\mathbf{\Sigma}_i$ is class covariance matrix which can be calculated by using training datasets. Using the expression above, we can rewrite the discriminant function as below

$$g_i(\mathbf{x}) = -\frac{1}{2}\log|\mathbf{\Sigma}_i| - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T \mathbf{\Sigma}_i^{-1} (\mathbf{x}-\boldsymbol{\mu}_i)$$

where equal terms for both classes are omitted. Classification is performed using this discriminant function with ML estimated parameters of each of the following parts. If a sample $\mathbf{x}$ has a greater g_1 than g_0, it will be labelled as C_1 and vice versa.

A. Part (a)

In this part, we calculate distinct covariance matrices for each class and also class mean vectors using ML estimates. These are the parameters of multivariate Gaussian distribution. Calculations of them can be seen below. Each element of class mean vector is the mean of one column of that class data. Class covariance matrix contains variances and covariances of all variables within that class.

$$E[\mathbf{X}_i] = \boldsymbol{\mu}_i = [\mu_1, \ldots, \mu_d]^T$$

$$\boldsymbol{\Sigma}_i = E[(\mathbf{X}_i - \boldsymbol{\mu}_i)(\mathbf{X}_i - \boldsymbol{\mu}_i)^T]$$

Using these calculations for each class of each dataset, we can calculate and visualize multivariate Gaussian likelihood isoprobability surfaces of their training sets as below.

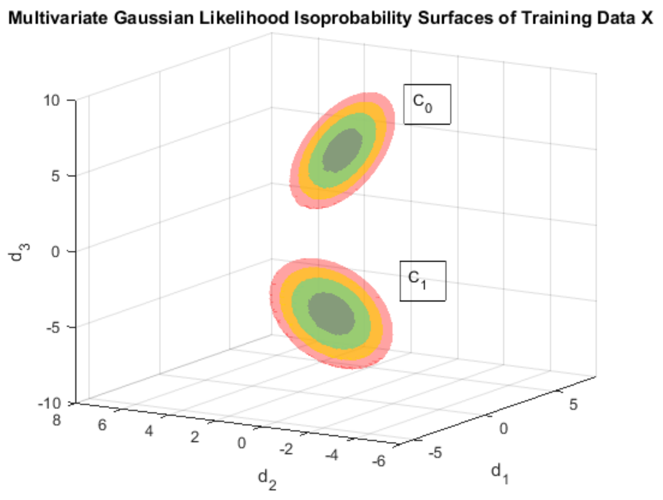

Fig. 3: Gaussian isoprobability surfaces of training set X.

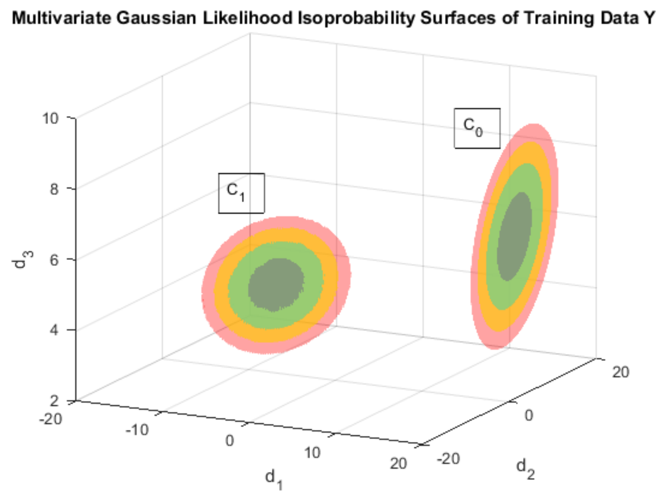

Fig. 4: Gaussian isoprobability surfaces of training set Y.

As can be seen from Fig. (3) and (4), isoprobability surfaces of two classes are not axis-aligned because of the covariance between variables and they have different orientations for both datasets because of the distinct covariance matrices.

In this part, the number of calculations are $K \cdot d = 6$ for the class means and $K \cdot \frac{d \cdot (d+1)}{2} = 12$ for the class covariance matrices where $K = 2$ is the number of classes and $d = 3$ is the number of variables. Complexity of covariance matrix calculation is $\mathcal{O}(d^2)$ in this case.

B. Part (b)

In this part, we use a common covariance matrix for both classes instead of distinct matrices. This common covariance matrix is calculated by pooling distinct covariance matrices as below.

$$\boldsymbol{\Sigma} = P(C_0) \cdot \boldsymbol{\Sigma}_0 + P(C_1) \cdot \boldsymbol{\Sigma}_1$$

Using this matrix for each class, we can calculate and visualize multivariate Gaussian likelihood isoprobability surfaces of training sets for both datasets, X and Y, as below.

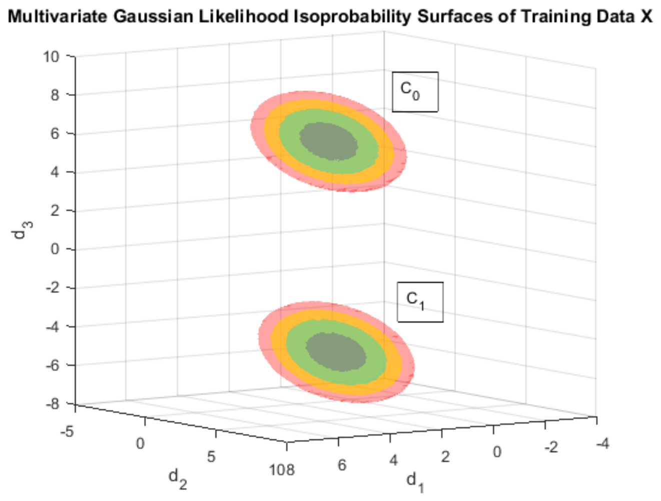

Fig. 5: Gaussian isoprobability surfaces of training set X.

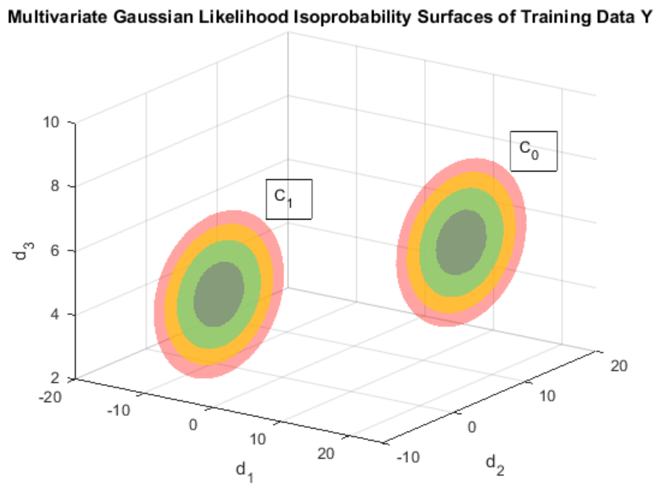

Fig. 6: Gaussian isoprobability surfaces of training set Y.

As can be seen from Fig. (5) and (6), isoprobability surfaces of two classes are not axis-aligned because of the covariance between variables and they have the same orientation for both datasets because of the same common covariance matrix (with different means).

In this part, the number of calculations are $K \cdot d = 6$ for the class means and $\frac{d \cdot (d+1)}{2} = 12$ for the class covariance matrices. Complexity of calculations is $\mathcal{O}(d^2)$ in this case.

C. Part (c)

In this part, in addition to the common covariance matrix for both classes, we assume a diagonal covariance matrix which means we make off-diagonal entries of covariance matrix zero. Using this matrix for each class, we can calculate and visualize multivariate Gaussian likelihood isoprobability surfaces of training sets for both datasets, X and Y, as below.

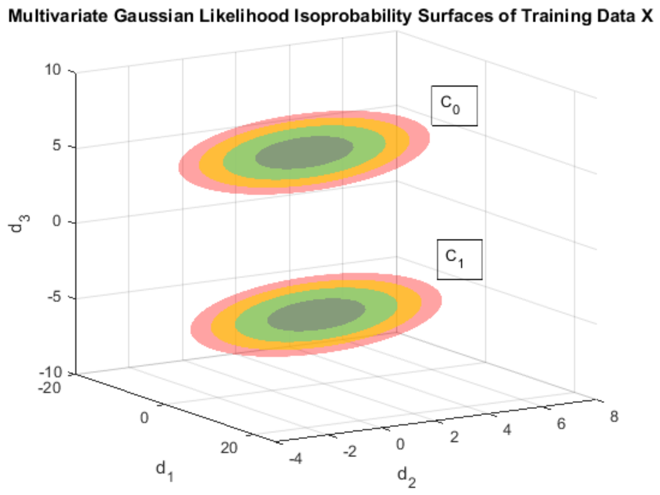

Fig. 7: Gaussian isoprobability surfaces of training set X.

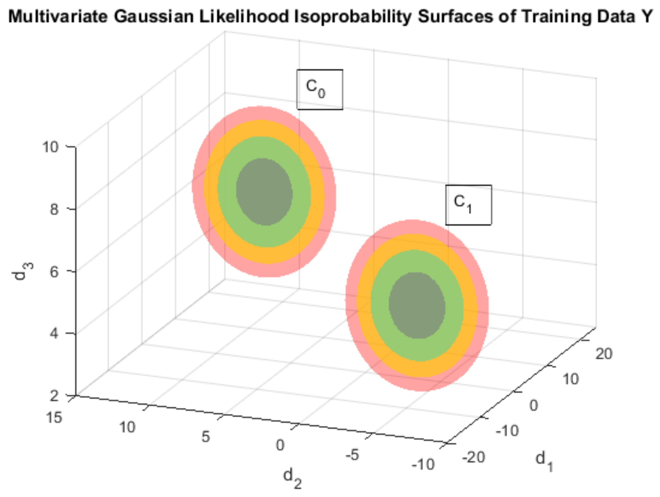

Fig. 8: Gaussian isoprobability surfaces of training set Y.

As can be seen from Fig. (7) and (8), isoprobability surfaces of two classes are axis-aligned because of the zero covariance assumption between variables.

In this case, the number of calculations are $K \cdot d = 6$ for the class means and $d = 3$ for the class covariance matrices. Complexity of calculations is significantly reduced from $\mathcal{O}(d^2)$ to $\mathcal{O}(d)$ in this case which lowers computation time.

D. Part (d)

In this part, in addition to the assumptions in previous parts, we assume variables have equal variances. This common variance is the mean of all diagonal entries of the diagonal covariance calculated in the previous part.

$$\sigma = \sum_{i=1}^{d} \Sigma_{ii}/d$$

Using this new covariance matrix for each class, we can calculate and visualize multivariate Gaussian likelihood isoprobability surfaces of training sets for both datasets, X and Y, as below.

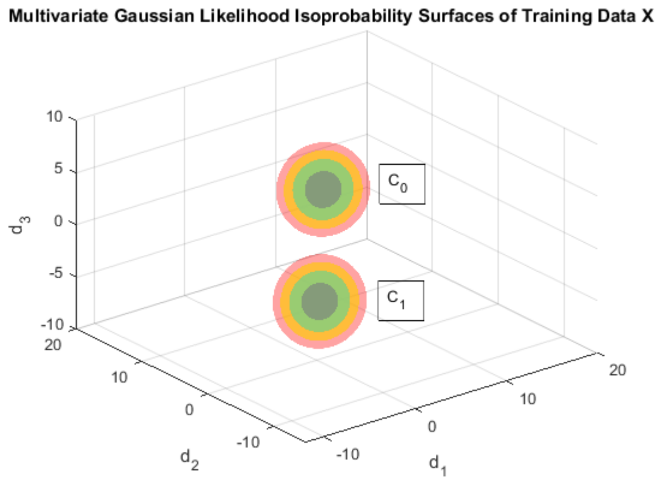

Fig. 9: Gaussian isoprobability surfaces of training set X.

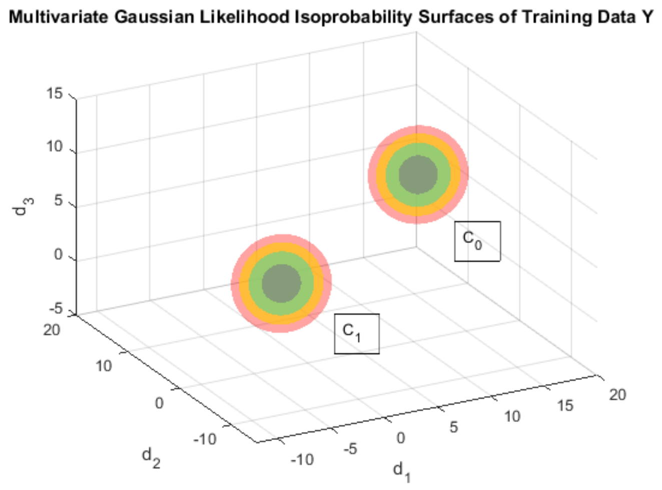

Fig. 10: Gaussian isoprobability surfaces of training set Y.

As can be seen from Fig. (9) and (10), isoprobability surfaces of two classes are axis-aligned as in the previous case because of the zero covariance assumption between variables. Also, isoprobability surfaces are hyperspheric in this case because of the equal variances assumption. In previous cases, isoprobability surfaces were hyperellipsoidal.

In this case, the number of calculations are $K \cdot d = 6$ for the class means and 1 for the class covariance matrices. Complexity of calculations is $\mathcal{O}(d)$ in this case.

III. Q2: Multivariate Classification with PCA

When the number of variables d is large and samples are small, $\mathbf{\Sigma}_i$ may be singular (zero determinant), and inverses may not exist. Or, $|\mathbf{\Sigma}_i|$ may be nonzero but too small, in which case it will be unstable; small changes in $\mathbf{\Sigma}_i$ will cause large changes in $\mathbf{\Sigma}_i^{-1}$. $|\mathbf{\Sigma}_i|$ goes to zero if variances are too small or absolute value of correlation between two variables are very close to 1. Therefore, two highly correlated variables or a variable with approximately zero variance may cause singularity and so we may need to decrease dimensionality, d.

In this section, we are supposed to project raw data onto one dimension using PCA. In PCA, we want to reduce dimensionality with minimum loss of information and maximize variance. For this purpose, covariance matrix of training data is calculated firstly. Then eigenvectors and eigenvalues of this matrix are calculated. Finally, the raw data (both training and validations parts) is projected onto the eigenvector associated with the maximum eigenvalue. This projection onto first principal component maximizes the proportion variance explained. In general, when λ_i are sorted in descending order, the proportion of variance explained by the k principal components is calculated as below where $k \leq d$. The proportions of variances explained by first principal components are 0.6762 for dataset X and 0.9056 for dataset Y.

$$\text{Explained Variance} = \frac{\lambda_1 + \cdots + \lambda_k}{\lambda_1 + \cdots + \lambda_k + \cdots + \lambda_d}$$

Also visualizations of datasets after dimensionality reduction can be seen below. By looking at Fig. (11) and (12), one can easily see that dimensionality reduction by PCA make dataset X difficult to classify compared to its raw version in Fig. (1) where one can discriminate two classes with naked eyes.

On the contrary, when we look at Fig. (13) and (14), we can easily see that the projection of dataset Y with PCA is much more beneficial than the projection of dataset X. We expect better classification performance on dataset Y than dataset X.

After PCA, Bayesian decision with Gaussian likelihoods for classes is performed on our new one-dimension sets for both datasets. Class likelihoods can be expressed as below where σ_i^2 is class variance and μ_i is class mean. These class parameters are maximum likelihood estimates.

$$p(x|C_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \cdot \exp(-\frac{(x-\mu_i)^2}{2\sigma_i^2})$$

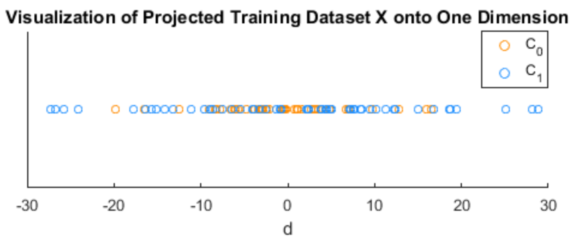

Fig. 11: Projected training part of dataset X using PCA.

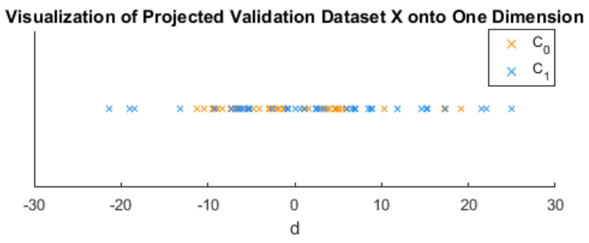

Fig. 12: Projected validation part of dataset X using PCA.

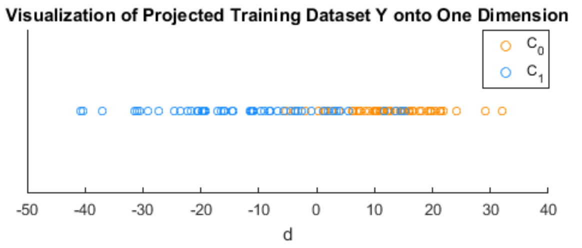

Fig. 13: Projected training part of dataset Y using PCA.

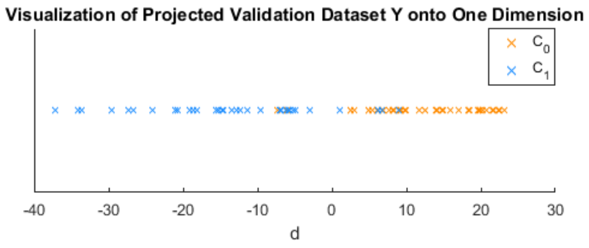

Fig. 14: Projected validation part of dataset Y using PCA.

Because both classes have the same priors, discrimination function can be written as below.

$$g_i(x) = \log p(x|C_i)$$

This discriminant can be further simplified as below. Classification is performed using this discriminant function. If a sample x has a greater g_1 than g_0, it will be labelled as C_1 and vice versa.

$$g_i(x) = -\log \sigma_i - \frac{(x-\mu_i)^2}{2\sigma_i^2}$$

IV. Q3: Multivariate Classification with LDA

Besides PCA, we also try LDA for dimensionality reduction in order to avoid possible singularity problems mentioned before. LDA is a supervised method which uses class information while PCA is an unsupervised method which do not use these labels.

In LDA, we want to find the direction, as defined by vector **w**, such that when the data is projected onto **w**, the examples from the two classes are as well separated as possible. After projection, for the two classes to be well separated, we would like the means to be as far apart as possible and the examples of classes be scattered in as small a region as possible. So we want to find **w** that maximizes the expression (Fisher's linear discriminant) below

$$J(\mathbf{w}) = \frac{(\mu_0 - \mu_1)^2}{\sigma_0^2 + \sigma_1^2}$$

where μ_i is the mean and σ_i^2 is the variance of class samples after projection. Also $\boldsymbol{\mu}_i$ is the mean of class samples before projection. Solution of this problem for our case is written below.

$$\mathbf{w} = \boldsymbol{\Sigma}_w^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

where $\boldsymbol{\Sigma}_w$ is the sum of within-class scatter matrices $\boldsymbol{\Sigma}_i$ for both classes. $\boldsymbol{\Sigma}_i$ can be calculated as below. Here, $r_i^t = 1$ when $\mathbf{x}^t$ belongs to class C_i and $r_i^t = 0$ otherwise.

$$\boldsymbol{\Sigma}_i = \sum_t r_i^t (\mathbf{x}^t - \boldsymbol{\mu}_i)(\mathbf{x}^t - \boldsymbol{\mu}_i)^T$$

After finding projection vector **w**, the raw data is projected onto this vector and its dimensionality is reduced to 1. Visualizations of datasets after projection can be seen below. By looking at Fig. (15) and (16), one can easily see that dimensionality reduction by LDA make dataset X very easy to classify even with naked eyes.

In addition, when we look at Fig. (17) and (18), we can see that the projection of dataset Y with LDA performs approximately the same as the projection with PCA. Dataset Y benefits less from LDA compared to dataset X. We expect better classification performance on dataset X than dataset Y with LDA.

After dimensionality reduction with LDA, Bayesian decision with Gaussian likelihoods for classes is performed on our new one-dimension sets for both datasets. Discriminant calculations are the same as the previous section and the discriminant for classification can be seen below.

$$g_i(x) = -\log \sigma_i - \frac{(x - \mu_i)^2}{2\sigma_i^2}$$

Again, if a sample x has a greater g_1 than g_0, it will be labelled as C_1 and vice versa.

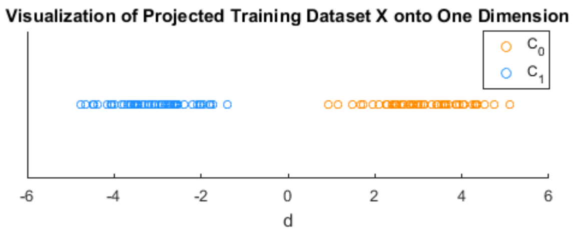

Fig. 15: Projected training part of dataset X using LDA.

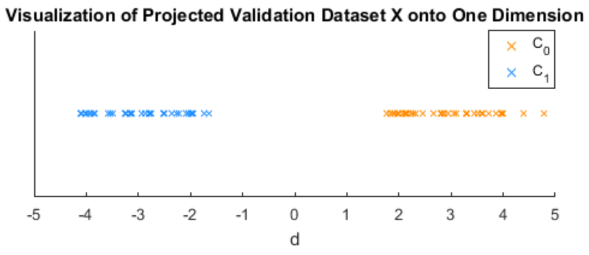

Fig. 16: Projected validation part of dataset X using LDA.

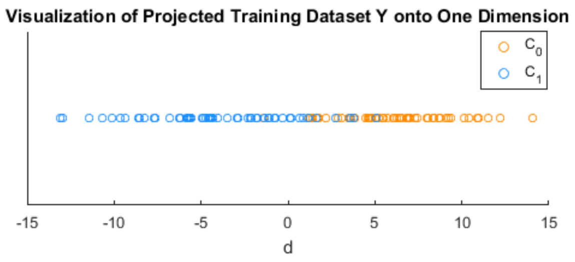

Fig. 17: Projected training part of dataset Y using LDA.

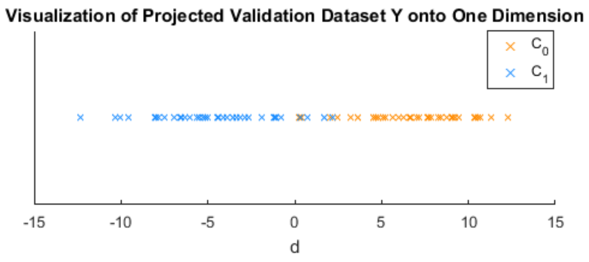

Fig. 18: Projected validation part of dataset Y using LDA.

V. Confusion Matrices

In this section, all the confusion matrices calculated after performing classification for all questions are shown. In these matrices, 1st row corresponds to actual class C_0 and 2nd row corresponds to actual class C_1. Also, 1st column corresponds to predicted class C_0 and 2nd column corresponds to predicted class C_1. These matrices will be compared and discussed for each datasets in the last section.

X		Y	
Training	Validation	Training	Validation
$\begin{bmatrix}60 & 0\\0 & 60\end{bmatrix}$	$\begin{bmatrix}40 & 0\\0 & 40\end{bmatrix}$	$\begin{bmatrix}58 & 2\\5 & 55\end{bmatrix}$	$\begin{bmatrix}39 & 1\\1 & 39\end{bmatrix}$

Table I: Confusion Matrices for Question 1(a).

X		Y	
Training	Validation	Training	Validation
$\begin{bmatrix}60 & 0\\0 & 60\end{bmatrix}$	$\begin{bmatrix}40 & 0\\0 & 40\end{bmatrix}$	$\begin{bmatrix}57 & 3\\6 & 54\end{bmatrix}$	$\begin{bmatrix}38 & 2\\2 & 38\end{bmatrix}$

Table II: Confusion Matrices for Question 1(b).

X		Y	
Training	Validation	Training	Validation
$\begin{bmatrix}53 & 7\\5 & 55\end{bmatrix}$	$\begin{bmatrix}36 & 4\\2 & 38\end{bmatrix}$	$\begin{bmatrix}56 & 4\\11 & 49\end{bmatrix}$	$\begin{bmatrix}38 & 2\\4 & 36\end{bmatrix}$

Table III: Confusion Matrices for Question 1(c).

X		Y	
Training	Validation	Training	Validation
$\begin{bmatrix}53 & 7\\5 & 55\end{bmatrix}$	$\begin{bmatrix}36 & 4\\2 & 38\end{bmatrix}$	$\begin{bmatrix}57 & 3\\11 & 49\end{bmatrix}$	$\begin{bmatrix}38 & 2\\4 & 36\end{bmatrix}$

Table IV: Confusion Matrices for Question 1(d).

X		Y	
Training	Validation	Training	Validation
$\begin{bmatrix}49 & 11\\35 & 25\end{bmatrix}$	$\begin{bmatrix}35 & 5\\28 & 12\end{bmatrix}$	$\begin{bmatrix}56 & 4\\10 & 50\end{bmatrix}$	$\begin{bmatrix}38 & 2\\3 & 37\end{bmatrix}$

Table V: Confusion Matrices for Question 2.

X		Y	
Training	Validation	Training	Validation
$\begin{bmatrix}60 & 0\\0 & 60\end{bmatrix}$	$\begin{bmatrix}40 & 0\\0 & 40\end{bmatrix}$	$\begin{bmatrix}55 & 5\\5 & 55\end{bmatrix}$	$\begin{bmatrix}38 & 2\\2 & 38\end{bmatrix}$

Table VI: Confusion Matrices for Question 3.

VI. Discussion

When we look at the confusion matrices of 1st question, we can easily see that classification performance on dataset X is better than dataset Y in part (a) as we expected by looking at Fig. (1) and (2). Classes in dataset X can be separated even with a plane while a more complex discriminator is needed to classify dataset Y.

As we go from part (a) to (d), we gradually simplify the calculations of covariance matrix and decrease the computational complexity and time. However at the same time, we can observe a decrease in classification accuracy at some simplification steps as can be seen from the confusion matrices. There is a trade-off between simplification (low complexity) and accuracy. The biggest advancement in decreasing complexity is between part (b) and (c). Also the biggest decrease in accuracy can be seen between these parts.

In the 2nd and 3rd questions, we applied dimensionality reduction and our aim is to simplify the complexity of classification calculations. Again in these cases, decrease in complexity generally decrease the classification accuracy.

We can now discuss the effect of PCA on classification performance. In classification problem, when the differentiating characteristics of the classes are not reflected in variance of the variables, PCA may not be a good choice of data processing. This is because PCA does not take into account class information when calculating the principal components. For the dataset X, PCA ruins perfect classification performance in the first question. This is because samples of both classes have maximum variance approximately on the same direction (eigenvector associated with maximum variance or eigenvalue) and their means are not far enough, then projection onto this direction make samples hard to classify.

PCA harms the classification of dataset Y less than the dataset X. Confusion matrices of dataset Y using PCA show better results compared to dataset X. This is because in this dataset, directions with maximum variances of classes noticeably differs from each other and their means are far enough. Therefore, when we project the raw data onto the direction with maximum variance of all data (both classes), we see an acceptable loss in the accuracy compared to the classification with raw dataset Y.

When we apply LDA, we can see much better classification performances on both datasets, especially on dataset X. This is because LDA consider class information data when it reduces the dimensionality. On dataset X, it can solve the problem in the PCA case. It projects both classes onto the direction where their means are well separated and their samples are tightly scattered.

LDA improve the classification performance on the dataset Y less than the dataset X. This is because it does not suffer from the problems which dataset X faces in the PCA case.

In overall, LDA performs well than PCA in classification. Throughout this homework, we assume that our raw data is normally distributed and Fisher's linear discriminant is optimal if the classes have multivariate Gaussian distribution.

VII. Appendix

A. Question 1

```
ClassifierQ1(X,'X','a');
ClassifierQ1(Y,'Y','a');
ClassifierQ1(X,'X','b');
ClassifierQ1(Y,'Y','b');
ClassifierQ1(X,'X','c');
ClassifierQ1(Y,'Y','c');
ClassifierQ1(X,'X','d');
ClassifierQ1(Y,'Y','d');
function [TrConfusion,ValConfusion] = ...
    ClassifierQ1(Data,Name,Part)

```

```matlab
% Arranging Data
TrueLabels_DataTr=[Data(1:60,4); ...
     Data(101:160,4)];
TrueLabels_DataVal=[Data(61:100,4); ...
     Data(161:end,4)];
DataTr=[Data(1:60,1:3);Data(101:160,1:3)];
DataVal=[Data(61:100,1:3);Data(161:end,1:3)];
N_tr=60; N_val=40; d=3;

% Visualization of Data
figure;  hold on; grid on;
scatter3(DataTr(1:60,1),DataTr(1:60,2), ...
     DataTr(1:60,3),25,[1 0.5469 0],'o', ...
     'DisplayName','Training C_0')
scatter3(DataVal(1:40,1),DataVal(1:40,2), ...
     DataVal(1:40,3),25,[1 0.5469 0],'x', ...
     'DisplayName','Validation C_0')
scatter3(DataTr(61:120,1),DataTr(61:120,2), ...
     DataTr(61:120,3),25,[0.1172 0.5625 ...
     1],'o', 'DisplayName','Training C_1')
scatter3(DataVal(41:80,1),DataVal(41:80,2), ...
     DataVal(41:80,3),25,[0.1172 0.5625 ...
     1],'x', 'DisplayName','Validation C_1')
view(-50,10)
legend('Show'); xlabel('d_1'); ...
     ylabel('d_2'); zlabel('d_3');
title(['Visualization of Training and ...
     Validation Dataset ',Name]);

% Bayesian Calculations for Data
DataMean_Class0=mean(DataTr(1:60,:));
DataMean_Class1=mean(DataTr(61:120,:));
Class_0=DataTr(1:60,:)-DataMean_Class0;
Class_1=DataTr(61:120,:)-DataMean_Class1;
Cov_Class0=(Class_0'*Class_0)./(N_tr);
Cov_Class1=(Class_1'*Class_1)./(N_tr);

if Part=='a'
    DataCov_Class1=Cov_Class1;
    DataCov_Class0=Cov_Class0;
end
if Part=='b'
    DataCov_Class0=0.5*Cov_Class0+0.5*Cov_Class1;
    DataCov_Class1=DataCov_Class0;
end
if Part=='c'
    DataCov_Class0=diag(diag(0.5*Cov_Class0+ ...
        0.5*Cov_Class1));
    DataCov_Class1=DataCov_Class0;
end
if Part=='d'
    DataCov_Class0=diag([1 1 ...
        1])*mean(diag(0.5*Cov_Class0+ ...
        0.5*Cov_Class1));
    DataCov_Class1=DataCov_Class0;
end

% Multivariate Gaussians for Class Likelihoods
s=0.2; x=(-12:s:23)';  y=(-13:s:18)';
z=(-10:s:15)'; [Y,X,Z]=meshgrid(y,x,z);
Class_0_Likelihood=zeros(length(x), ...
     length(y),length(z));
Class_1_Likelihood=zeros(length(x), ...
     length(y),length(z));

for i=1:length(x)
    for j=1:length(y)
        for k=1:length(z)
            Class_0_Likelihood(i,j,k)= ...
                (1/(sqrt(det(DataCov_Class0) ...
                *((2*pi)^d))))*exp((-0.5)* ...
                ([x(i) y(j) z(k)]- ...
                DataMean_Class0)* ...
                (DataCov_Class0\([x(i) y(j) ...
                z(k)]-DataMean_Class0)'));
            Class_1_Likelihood(i,j,k) ...
                =(1/(sqrt(det(DataCov_Class1) ...
                *((2*pi)^d))))*exp((-0.5)* ...
                ([x(i) y(j) z(k)]- ...
                DataMean_Class1)* ...
                (DataCov_Class1\([x(i) y(j) ...
                z(k)]-DataMean_Class1)'));
        end
    end
end

Isoprobability Surfaces for Data
figure; hold on; grid on;
isovalue = 0.82*(max(Class_0_Likelihood(:))- ...
     min(Class_0_Likelihood(:)))+ ...
     min(Class_0_Likelihood(:));
surf1=isosurface(X,Y,Z,Class_0_Likelihood, ...
     isovalue);
p1 = patch(surf1);
set(p1,'FaceColor','red','EdgeColor','none', ...
     'FaceAlpha',0.2);

isovalue = 0.87*(max(Class_0_Likelihood(:))- ...
     min(Class_0_Likelihood(:)))+ ...
     min(Class_0_Likelihood(:));
surf2=isosurface(X,Y,Z,Class_0_Likelihood, ...
     isovalue);
p2 = patch(surf2);
set(p2,'FaceColor','yellow','EdgeColor','none', ...
     'FaceAlpha',0.4);

isovalue = 0.92*(max(Class_0_Likelihood(:))- ...
     min(Class_0_Likelihood(:)))+ ...
     min(Class_0_Likelihood(:));
surf3=isosurface(X,Y,Z,Class_0_Likelihood, ...
     isovalue);
p3 = patch(surf3);
set(p3,'FaceColor','cyan','EdgeColor','none', ...
     'FaceAlpha',0.6);

isovalue = 0.97*(max(Class_0_Likelihood(:))- ...
     min(Class_0_Likelihood(:)))+ ...
     min(Class_0_Likelihood(:));
surf4=isosurface(X,Y,Z,Class_0_Likelihood, ...
     isovalue);
p4 = patch(surf4);
set(p4,'FaceColor','blue','EdgeColor','none', ...
     'FaceAlpha',1);

isovalue = 0.82*(max(Class_1_Likelihood(:))- ...
     min(Class_1_Likelihood(:)))+ ...
     min(Class_1_Likelihood(:));
surf1=isosurface(X,Y,Z,Class_1_Likelihood, ...
     isovalue);
p1 = patch(surf1);
set(p1,'FaceColor','red','EdgeColor','none', ...
     'FaceAlpha',0.2);

isovalue = 0.87*(max(Class_1_Likelihood(:))- ...
     min(Class_1_Likelihood(:)))+ ...
     min(Class_1_Likelihood(:));
surf2=isosurface(X,Y,Z,Class_1_Likelihood, ...
     isovalue);
p2 = patch(surf2);
set(p2,'FaceColor','yellow','EdgeColor','none', ...
     'FaceAlpha',0.4);

isovalue = 0.92*(max(Class_1_Likelihood(:))- ...
     min(Class_1_Likelihood(:)))+ ...
     min(Class_1_Likelihood(:));
surf3=isosurface(X,Y,Z,Class_1_Likelihood, ...
     isovalue);
p3 = patch(surf3);
```

```
set(p3,'FaceColor','cyan','EdgeColor','none', ...
    'FaceAlpha',0.6);

isovalue = 0.97*(max(Class_1_Likelihood(:))- ...
    min(Class_1_Likelihood(:)))+ ...
    min(Class_1_Likelihood(:));
surf4=isosurface(X,Y,Z,Class_1_Likelihood ...
    ,isovalue);
p4 = patch(surf4);
set(p4,'FaceColor','blue','EdgeColor','none', ...
    'FaceAlpha',1);

ylim([-15 20])
zlim([-5 15])
xlim([-12 20])
xlabel('d_1'); ylabel('d_2'); zlabel('d_3')
title(['Multivariate Gaussian Likelihood ...
    Isoprobability Surfaces of Training Data ...
    ',Name])

% Predicting Classes for Data
PredictedLabels_DataTr=zeros(120,1);
PredictedLabels_DataVal=zeros(80,1);
for i=1:(2*N_tr)
    g1_tr=-0.5*(log(det(DataCov_Class1))+ ...
        (DataTr(i,:)-DataMean_Class1)* ...
        (DataCov_Class1\(DataTr(i,:)- ...
        DataMean_Class1)'));
    g0_tr=-0.5*(log(det(DataCov_Class0))+ ...
        (DataTr(i,:)-DataMean_Class0)* ...
        (DataCov_Class0\(DataTr(i,:)- ...
        DataMean_Class0)'));
    if g1_tr>g0_tr
        PredictedLabels_DataTr(i)=1;
    end
end

for i=1:(2*N_val)
    g1_val=-0.5*(log(det(DataCov_Class1))+ ...
        (DataVal(i,:)-DataMean_Class1)* ...
        (DataCov_Class1\(DataVal(i,:)- ...
        DataMean_Class1)'));
    g0_val=-0.5*(log(det(DataCov_Class0))+ ...
        (DataVal(i,:)-DataMean_Class0)* ...
        (DataCov_Class0\(DataVal(i,:)- ...
        DataMean_Class0)'));
    if g1_val>g0_val
        PredictedLabels_DataVal(i)=1;
    end
end

TrConfusion=confusionmat(TrueLabels_DataTr, ...
    PredictedLabels_DataTr)
ValConfusion=confusionmat(TrueLabels_DataVal, ...
    PredictedLabels_DataVal)

end
```

B. Question 2

```
ClassifierQ2(X,'X');
ClassifierQ2(Y,'Y');

function [TrConfusion,ValConfusion] = ...
    ClassifierQ2(Data,Name)

% Arranging Data
TrueLabels_DataTr=[Data(1:60,4); ...
    Data(101:160,4)];
TrueLabels_DataVal=[Data(61:100,4); ...
    Data(161:end,4)];
DataTr=[Data(1:60,1:3);Data(101:160,1:3)];
DataVal=[Data(61:100,1:3);Data(161:end,1:3)];
N_tr=60; N_val=40;

% PCA
DataTrMean=mean(DataTr);
Inter=DataTr-DataTrMean;
Cov_DataTr=(Inter'*Inter)./(2*N_tr);
[V,U] = eig(Cov_DataTr);
V=V(:,diag(U)==max(diag(U)));
Variance_Explained=max(diag(U))/sum(diag(U));
disp('Explained variance: ')
disp(Variance_Explained)
ProjectedDataTr=V'*(DataTr-DataTrMean)';
ProjectedDataVal=V'*(DataVal-DataTrMean)';

figure;
hAxes = axes('NextPlot','add',...
           'DataAspectRatio',[1 1 1],...
           'YLim',[-9 9],...
           'Color','white');
scatter(ProjectedDataTr(1:60),zeros(1,60), ...
    25,[1 0.5469 0],'o','DisplayName','C_0')
scatter(ProjectedDataTr(61:120),zeros(1,60), ...
    25,[0.1172 0.5625 ...
    1],'o','DisplayName','C_1')
yticks([])
xlabel('d')
legend('Show','Location','northeast')
title(['Visualization of Projected Training ...
    Dataset ',Name,' onto One Dimension'])

figure;
hAxes = axes('NextPlot','add',...
           'DataAspectRatio',[1 1 1],...
           'YLim',[-9 9],...
           'Color','white');
scatter(ProjectedDataVal(1:40),zeros(1,40), ...
    25,[1 0.5469 0], 'x','DisplayName','C_0')
scatter(ProjectedDataVal(41:80),zeros(1,40), ...
    25,[0.1172 0.5625 1], ...
    'x','DisplayName','C_1')
yticks([])
xlabel('d')
legend('Show','Location','northeast')
title(['Visualization of Projected ...
    Validation Dataset ',Name,' onto One ...
    Dimension'])


mean0=mean(ProjectedDataTr(1:60));
mean1=mean(ProjectedDataTr(61:120));
std0=std(ProjectedDataTr(1:60));
std1=std(ProjectedDataTr(61:120));

% Predicting Classes for Data

PredictedLabels_DataTr=zeros(120,1);
PredictedLabels_DataVal=zeros(80,1);
for i=1:(2*N_tr)
    g1_tr=-log(std1)-((ProjectedDataTr(i)- ...
        mean1)^2)/(2*std1^2);
    g0_tr=-log(std0)-((ProjectedDataTr(i)- ...
        mean0)^2)/(2*std0^2);
    if g1_tr>g0_tr
        PredictedLabels_DataTr(i)=1;
    end
end

for i=1:(2*N_val)
    g1_val=-log(std1)-((ProjectedDataVal(i)- ...
        mean1)^2)/(2*std1^2);
    g0_val=-log(std0)-((ProjectedDataVal(i)- ...
        mean0)^2)/(2*std0^2);
    if g1_val>g0_val
```

```
        PredictedLabels_DataVal(i)=1;
    end
end

TrConfusion=confusionmat(TrueLabels_DataTr, ...
    PredictedLabels_DataTr);
ValConfusion=confusionmat(TrueLabels_DataVal, ...
    PredictedLabels_DataVal);

end
```

C. Question 3

```
ClassifierQ3(X,'X');
ClassifierQ3(Y,'Y');

function ClassifierQ2(Data,Name)

% Arranging Data
TrueLabels_DataTr=[Data(1:60,4); ...
    Data(101:160,4)];
TrueLabels_DataVal=[Data(61:100,4); ...
    Data(161:end,4)];
DataTr=[Data(1:60,1:3);Data(101:160,1:3)];
DataTr_Class0=DataTr(1:60,:);
DataTr_Class1=DataTr(61:120,:);
DataVal=[Data(61:100,1:3);Data(161:end,1:3)];
N_tr=60; N_val=40;

% LDA
DataTr0Mean=mean(DataTr_Class0);
Class0=DataTr_Class0-DataTr0Mean;
Cov_DataTr0=(Class0'*Class0);
DataTr1Mean=mean(DataTr_Class1);
Class1=DataTr_Class1-DataTr1Mean;
Cov_DataTr1=(Class1'*Class1);
Cov_DataTr=Cov_DataTr0+Cov_DataTr1;
w=Cov_DataTr\(DataTr0Mean-DataTr1Mean)';
w=w/norm(w);
ProjectedDataTr=w'*(DataTr)';
ProjectedDataVal=w'*(DataVal)';

figure;
hAxes = axes('NextPlot','add',...
             'DataAspectRatio',[1 1 1],...
             'YLim',[-1.6 1.6],...
             'Color','white');
scatter(ProjectedDataTr(1:60),zeros(1,60), ...
    25,[1 0.5469 0],'o','DisplayName','C_0')
scatter(ProjectedDataTr(61:120),zeros(1,60), ...
    25,[0.1172 0.5625 ...
    1],'o','DisplayName','C_1')
yticks([])
xlabel('d')
legend('Show','Location','northeast')
title(['Visualization of Projected Training ...
    Dataset ',Name,' onto One Dimension'])

figure;
hAxes = axes('NextPlot','add',...
             'DataAspectRatio',[1 1 1],...
             'YLim',[-1.6 1.6],...
             'Color','white');
scatter(ProjectedDataVal(1:40),zeros(1,40), ...
    25,[1 0.5469 0], 'x','DisplayName','C_0')
scatter(ProjectedDataVal(41:80),zeros(1,40), ...
    25,[0.1172 0.5625 1], ...
    'x','DisplayName','C_1')
yticks([])
xlabel('d')
legend('Show','Location','northeast')
title(['Visualization of Projected ...
    Validation Dataset ',Name,' onto One ...
    Dimension'])

mean0=mean(ProjectedDataTr(1:60));
mean1=mean(ProjectedDataTr(61:120));
std0=std(ProjectedDataTr(1:60));
std1=std(ProjectedDataTr(61:120));

% Predicting Classes for Data

PredictedLabels_DataTr=zeros(120,1);
PredictedLabels_DataVal=zeros(80,1);
for i=1:(2*N_tr)
    g1_tr=-log(std1)-((ProjectedDataTr(i)- ...
        mean1)^2)/(2*std1^2);
    g0_tr=-log(std0)-((ProjectedDataTr(i)- ...
        mean0)^2)/(2*std0^2);
    if g1_tr>g0_tr
        PredictedLabels_DataTr(i)=1;
    end
end

for i=1:(2*N_val)
    g1_val=-log(std1)-((ProjectedDataVal(i)- ...
        mean1)^2)/(2*std1^2);
    g0_val=-log(std0)-((ProjectedDataVal(i)- ...
        mean0)^2)/(2*std0^2);
    if g1_val>g0_val
        PredictedLabels_DataVal(i)=1;
    end
end

TrConfusion=confusionmat(TrueLabels_DataTr, ...
    PredictedLabels_DataTr);
ValConfusion=confusionmat(TrueLabels_DataVal, ...
    PredictedLabels_DataVal);

end
```