EECE 5644 Spring 2021

Assignment #1

Matthew Dottinger

Due: 2/22/21

Question 1

Part A

Code for this question can be found in Appendix A.

A set of 10,000 samples was generated and used for all parts of this problem. These samples were in a 4-dimensional real-valued random vector **X** following the probability density function $p(\boldsymbol{x}) = p(\boldsymbol{x}|L=0)P(L=0) + p(\boldsymbol{x}|L=1)P(L=1)$ where L indicates the true class label. The class conditional multivariate Gaussian pdfs $p(\boldsymbol{x}|L=i) = g(\boldsymbol{x}|\boldsymbol{m_i}, \boldsymbol{C_i})$ and class priors are summarized below:

$$\boldsymbol{m_0} = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix} \quad \boldsymbol{C_0} = \begin{bmatrix} 2 & -0.5 & 0.3 & 0 \\ -0.5 & 1 & -0.5 & 0 \\ 0.3 & -0.5 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \quad P(L=0) = 0.7$$

$$\boldsymbol{m_1} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad \boldsymbol{C_1} = \begin{bmatrix} 1 & 0.3 & -0.2 & 0 \\ 0.3 & 2 & 0.3 & 0 \\ -0.2 & 0.3 & 1 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix} \quad P(L=1) = 0.3$$

1. The minimum expected risk classification rule in the form of a likelihood-ratio test is:

$$\frac{p(\boldsymbol{x}|L=1)}{p(\boldsymbol{x}|L=0)} \gtrless \gamma = \frac{\lambda_{10} - \lambda_{00}}{\lambda_{01} - \lambda_{11}} \frac{P(L=0)}{P(L=1)} = \frac{\lambda_{10} - \lambda_{00}}{\lambda_{01} - \lambda_{11}} \frac{0.7}{0.3}$$

   To minimize probability of error in classification, a 0-1 loss matrix should be used, resulting in the following value for γ:

$$\Lambda = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$(D=1) \quad \frac{p(\boldsymbol{x}|L=1)}{p(\boldsymbol{x}|L=0)} \gtrless \gamma = \frac{1-0}{1-0} \frac{0.7}{0.3} = 2.\overline{3} \quad (D=0)$$

2. The classifier above was implemented and applied to the 10,000 generated samples. The threshold value γ was varied from 0 to ∞ to produce the ROC curve shown in Figure 1 on the next page.

3. Using the generated samples, the probability of error was tracked for each threshold γ and the minimum was found to be $P_e = 0.0808$ at a threshold of $\gamma = 2.1996$. This point is marked by the red cross on the ROC curve in Figure 1. The total probability of error is calculated by:

$$P_e = P(D=1|L=0)P(L=0) + P(D=0|L=1)P(L=1)$$

The theoretical minimum probability of error $P_e = 0.0814$ was also numerically calculated using the theoretical threshold $\gamma = 2.\overline{3}$ calculated above. This point is indicated by the green circle on the ROC curve in Figure 1.

The calculated and theoretical minimum probability of error are summarized in Table 1 below. As can be seen, both the threshold value and probability error are closely aligned between the theoretical and calculated cases.

Table 1. Comparison of calculated and theoretical error

	γ	Minimum P_e
Theoretical	2.3333	0.0814
Calculated from Data	2.1996	0.0808

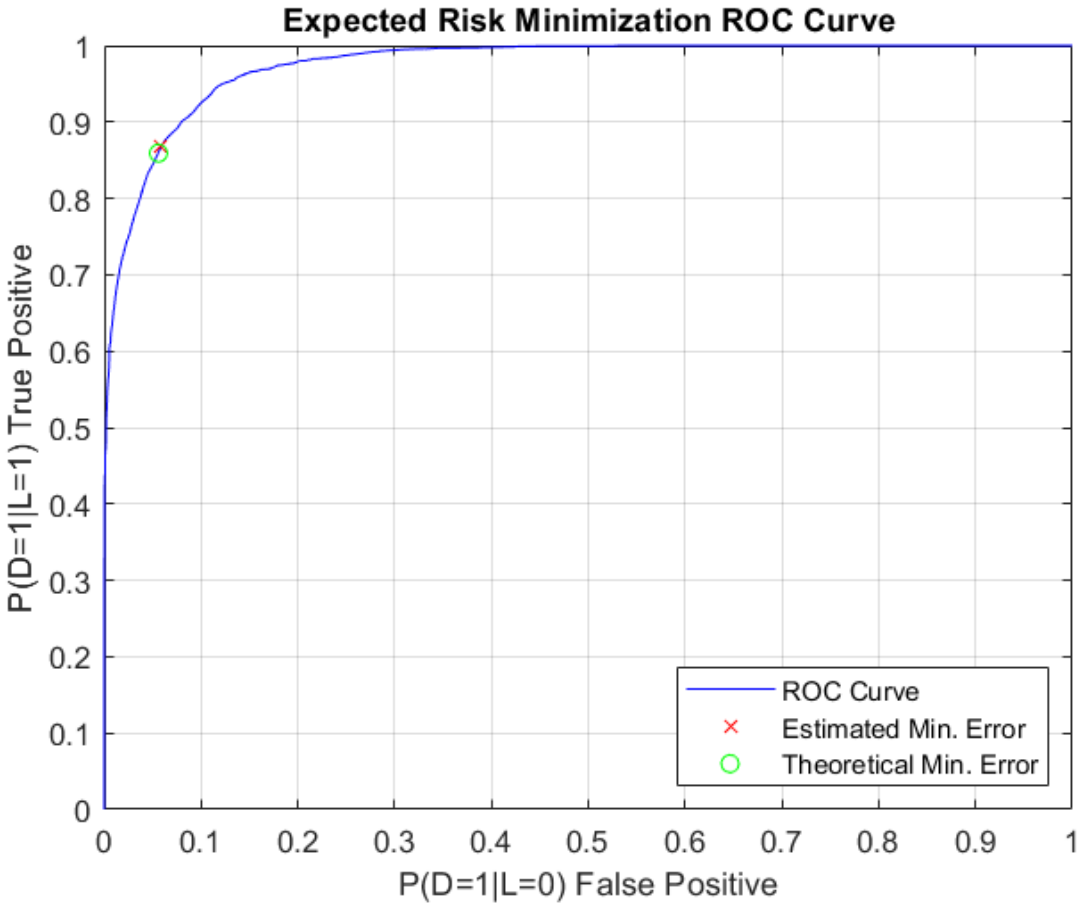

Figure 1. ROC Curve for ERM using true knowledge of class conditional pdfs

Part B

For Part B, the same data as generated above was used, however it was analyzed using a Naive Bayesian Classifier. In this case, true knowledge of the class priors and class conditional means was assumed. Class conditional covariance matrices were incorrectly assumed to be diagonal:

$$\boldsymbol{C_0} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

$$\boldsymbol{C_1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

1. The minimum expected risk classification rule in the form of a likelihood-ratio test remains the same as in Part A:

$$(D = 1) \quad \frac{p(\boldsymbol{x}|L=1)}{p(\boldsymbol{x}|L=0)} \gtrless \gamma = \frac{1-0}{1-0}\frac{0.7}{0.3} = 2.\overline{3} \quad (D = 0)$$

2. The classifier above was implemented and applied to the 10,000 generated samples. The threshold value γ was varied from 0 to ∞ to produce the ROC curve shown in Figure 2 on the next page.

3. Using the generated samples, the probability of error was tracked for each threshold γ and the minimum was found to be $P_e = 0.0913$ at a threshold of $\gamma = 2.5967$. This point is marked by the red cross on the ROC curve in Figure 1. The total probability of error is calculated by:

$$P_e = P(D = 1|L = 0)P(L = 0) + P(D = 0|L = 1)P(L = 1)$$

   The theoretical minimum probability of error $P_e = 0.0915$ was also numerically calculated using the theoretical threshold $\gamma = 2.\overline{3}$ calculated above. This point is indicated by the green circle on the ROC curve in Figure 1.

   The calculated and theoretical minimum probability of error are summarized in Table 2 below. As can be seen, both the threshold value and probability error are closely aligned between the theoretical and calculated cases.

Table 2. Comparison of calculated and theoretical error

	γ	**Minimum P_e**
Theoretical	2.3333	0.0915
Calculated from Data	2.5967	0.0913

Both the ERM classifier with true knowledge of the class conditional pdfs and the Naive Bayesian classifier provided similar results with respect to probability of error. This is likely due to the distance between the means (2.83) of the classes being large with respect to the average standard deviation (1.25). Because the class means are farther than two standard deviations apart, there is little overlap between classes so not having complete knowledge of the data distributions causes little trouble with classification.

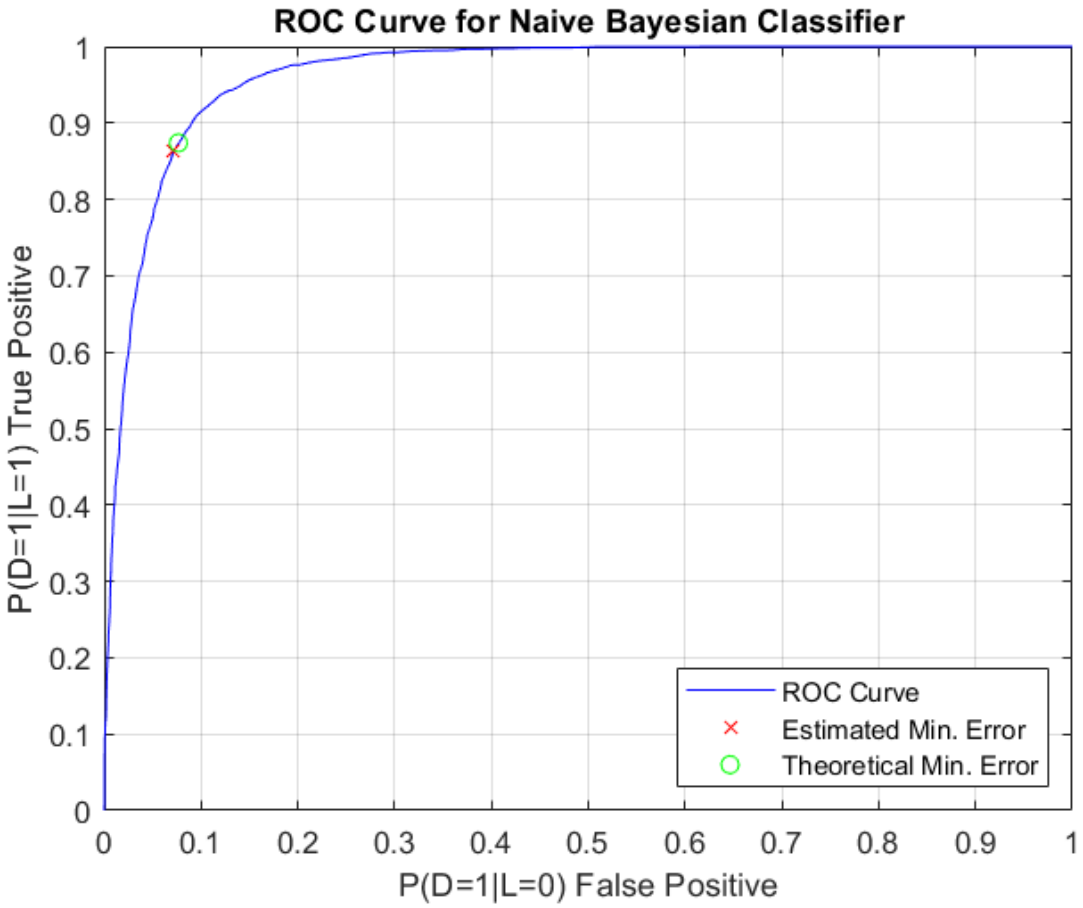

Figure 2. ROC curve for Naive Bayesian classifier

Question 2

Part A

Code for this question can be found in Appendix B.

1. A 10,000 sample 3-dimensional random vector **X** was generated and used for all parts of the problem. The data represents a mixture of four Gaussian distributions and three class labels. One of the Gaussians represents Class 1, the second Gaussian represents Class 2, and Class 3 comes from a mixture of the other two Gaussians with equal weights. To generate data for Class 3, half the points were taken from one Gaussian distribution and the other half were taken from another.

   To ensure significant overlap between the class conditional pdfs, the means were set with a distance of twice the average standard deviation of all the Gaussians. Four equidistant points in 3-dimensional space lie on the vertices of a tetrahedron, and using the Pythagorean theorem we can calculate the location of each point:

$$\boldsymbol{m_1} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad \boldsymbol{m_2} = \begin{bmatrix} 2\sigma \\ 0 \\ 0 \end{bmatrix} \quad \boldsymbol{m_3} = \begin{bmatrix} \sigma \\ \sqrt{3}\sigma \\ 0 \end{bmatrix} \quad \boldsymbol{m_4} = \begin{bmatrix} \sigma \\ \frac{\sqrt{3}}{3}\sigma \\ \frac{\sqrt{8}}{3}\sigma \end{bmatrix}$$

   The covariance matrices were set to be diagonal. Each of the diagonal entries was randomly selected between 0 and 1. The data generated fit these covariance matrices:

$$\boldsymbol{C_1} = \begin{bmatrix} 0.8700 & 0 & 0 \\ 0 & 0.6060 & 0 \\ 0 & 0 & 0.1773 \end{bmatrix}$$

$$\boldsymbol{C_2} = \begin{bmatrix} 0.1090 & 0 & 0 \\ 0 & 0.9424 & 0 \\ 0 & 0 & 0.3718 \end{bmatrix}$$

$$\boldsymbol{C_3} = \begin{bmatrix} 0.8565 & 0 & 0 \\ 0 & 0.4094 & 0 \\ 0 & 0 & 0.1639 \end{bmatrix}$$

$$\boldsymbol{C_4} = \begin{bmatrix} 0.7053 & 0 & 0 \\ 0 & 0.3675 & 0 \\ 0 & 0 & 0.0889 \end{bmatrix}$$

   The class priors were set as:

$$P(L = 1) = 0.3 \quad P(L = 2) = 0.3 \quad P(L = 3) = 0.4$$

After generating the sample data, it was plotted to show the true data distribution for each class, shown in Figure 3. Note that Class 3, shown in the figure with a green cross, is composed of two distributions, however most of the second distribution is occluded by the sample points in Classes 1 and 2.

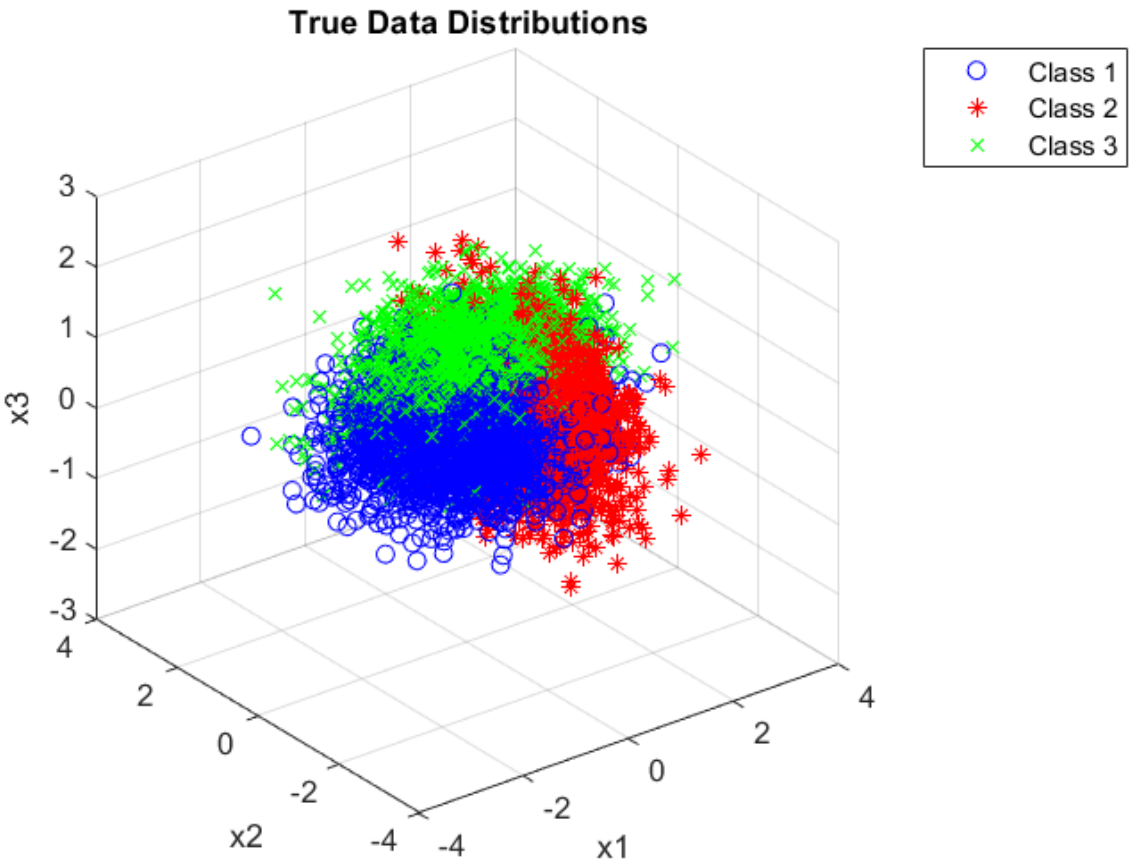

Figure 3. True data distributions for generated samples

2. Expected Risk Minimization was used to classify the data using the following decision rule:

$$D(x) = \min_{i\epsilon\{1,2,3\}} \sum_{j=1}^{C} \lambda_{ij}\, p(L = j|x)$$

Where C is the number of classes, λ_{ij} is the loss for classifying a sample in class j as label i, and $p(L = j|x)$ is the class posterior, defined by:

$$p(L = j|x) = \frac{p(x|L = j)P(L = j)}{p(x)}$$

Where $p(x|L = j)$ is the class conditional, $p(L = j)$ is the class prior, and $p(x)$ is:

$$p(x) = \sum_{j=1}^{C} p(x|L = j)P(L = j)$$

To minimize probability of error, 0-1 loss is selected:

$$\Lambda = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

After implementing the above decision rule, the confusion matrix was empirically estimated using the results of classification by comparing the decisions to the labels. The confusion matrix is shown in Table 3.

Table 3. Confusion matrix with 0-1 loss

		True Class Label		
		1	2	3
Decision	1	0.6462	0.0409	0.1044
	2	0.1676	0.7638	0.1363
	3	0.1862	0.1954	0.7592

3. The sample data was then plotted in Figure 4 on the next page to visualize the decisions made. Different shapes indicate each true class. Circles indicate Class 1, stars indicate Class 2, and crosses indicate Class 3. Color is used to indicate whether the point was correctly classified to its true class. Green and red indicate a correctly and incorrectly classified point, respectively.

   It is interesting to note that in the diagram, it is possible to see the dividing line between decisions in red. Mostly, however, the diagram is dominated by green correct decisions.

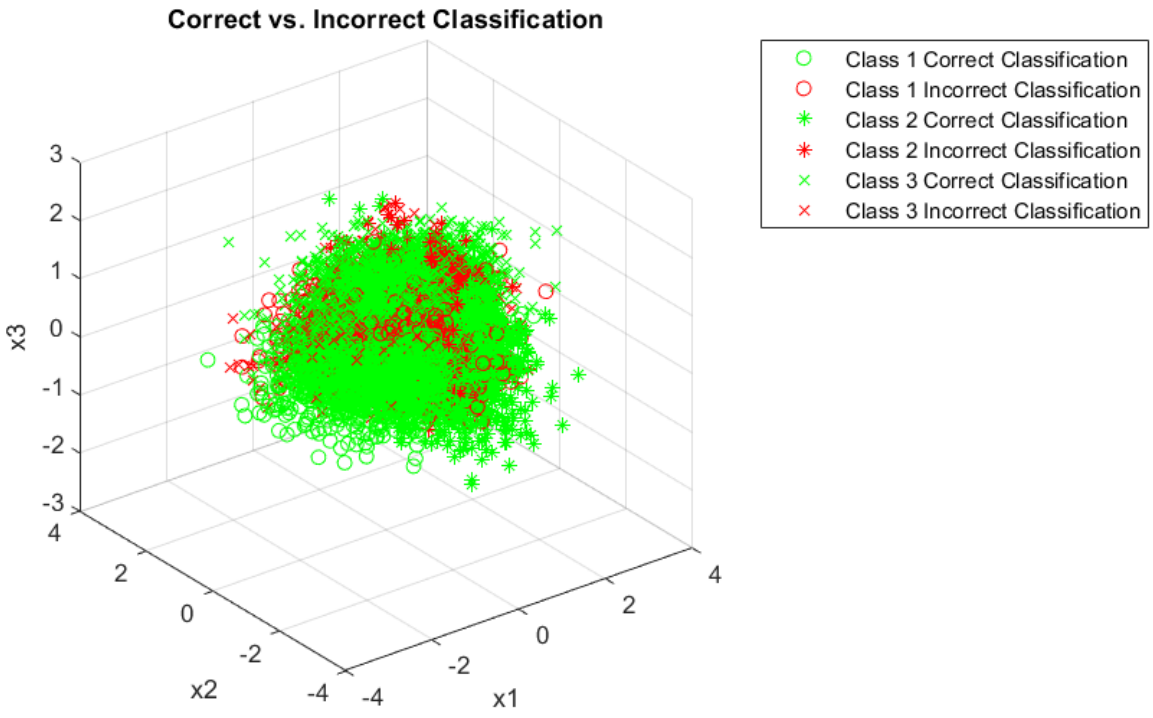

Figure 4. Correct vs incorrect classifications

Part B

The same data was used as in Part A for this part. The same ERM classification rule was used to classify the data, however loss matrices which respectively care 10 times and 100 times more about misclassification when L=3 were used:

$$\Lambda_{10} = \begin{bmatrix} 0 & 1 & 10 \\ 1 & 0 & 10 \\ 1 & 1 & 0 \end{bmatrix}$$

$$\Lambda_{100} = \begin{bmatrix} 0 & 1 & 100 \\ 1 & 0 & 100 \\ 1 & 1 & 0 \end{bmatrix}$$

After classifying the sample data the same visualizations were made for each run. Figures 5 and 6 show the correct and incorrect classifications for Λ_{10} and Λ_{100}, respectively. It can be seen that as the loss associated with misclassifying Class 3 is increased, the decision boundary moves further from the mean of Class 3. This results in more misclassification of Classes 1 and 2, causing the misclassed area to become wider.

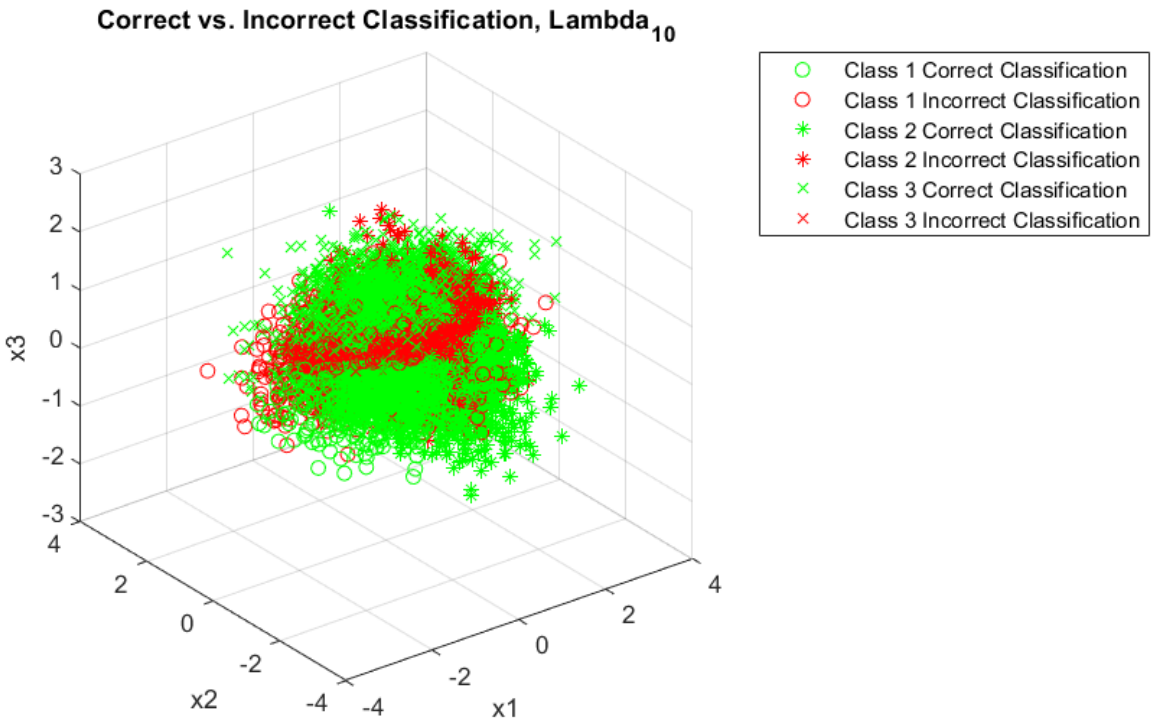

Figure 5. Correct vs incorrect classifications, 10 times sensitivity to misclassing Class 3

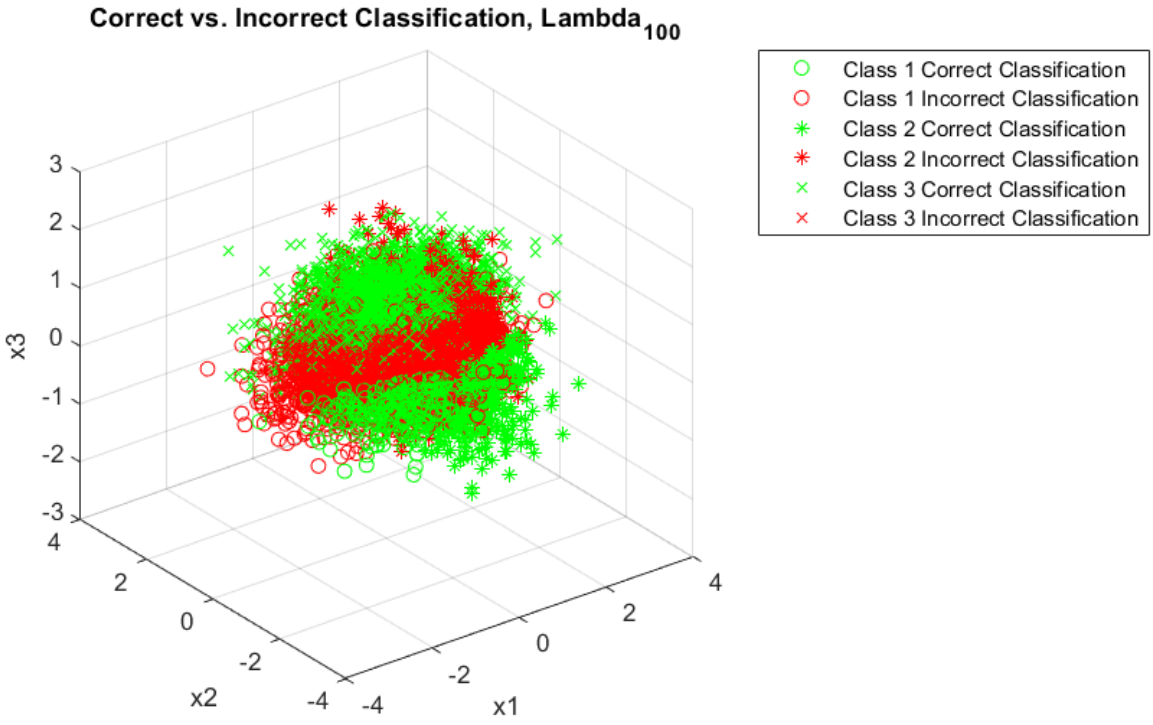

Figure 6. Correct vs incorrect classifications, 100 times sensitivity to misclassing Class 3

The confusion matrices were also empirically estimated for each case. Tables 4 and 5 contain the confusion matrices for Λ_{10} and Λ_{100}, respectively. Comparing these to the confusion matrix for 0-1 loss, the effect of increasing loss for misclassification of Class 3 becomes clear. With 10 times sensitivity to misclassification of Class 3, a majority of the points are classified as Class 3. Over 98% of Class 3 points are correctly classified, at the cost of misclassifying much of Classes 1 and 2 as Class 3. Increasing the sensitivity to 100 further magnifies this effect. The decision of the classifier is almost always to choose Class 3. This results almost every Class 3 point being classified correctly, but also most of Class 1 and Class 2 misclassified.

The overall probability of error increases from 27% for 0-1 loss, to 42% for 10 times increased sensitivity for misclassification of Class 3, to 54% for 100 times sensitivity. While increasing loss associated with misclassifying Class 3 ensures nearly all Class 3 samples are identified, the overall probability of error increases significantly and the classifier struggles with other classes.

Table 4. Confusion matrix, with 10 times sensitivity to misclassing Class 3

		True Class Label		
		1	2	3
Decision	1	0.2802	0.0135	0.0079
	2	0.0725	0.3535	0.0086
	3	0.6472	0.6329	0.9835

Table 5. Confusion matrix, with 100 times sensitivity to misclassing Class 3

		True Class Label		
		1	2	3
Decision	1	0.0755	0.0020	0.0003
	2	0.0222	0.1443	0.0003
	3	0.9023	0.8537	0.9995

Question 3

Part A

Code for this question can be found in Appendix C.

I've chosen to split the results for this question into 2 parts. Part A contains information about the first dataset about wine. Part B contains information about the second dataset about human activity.

After importing the wine dataset, the class conditional pdfs were estimated by finding the mean and covariance of the samples for each class. It was assumed that each feature followed a Gaussian distribution. Not every class came with data in the sample set, so careful checking for null values had to be implemented throughout the classifier.

Because the class covariance matrices were empirically estimated, some were ill conditioned. To ensure they were all positive definite, a regularization term was included:

$$\boldsymbol{C}_{Regularized} = \boldsymbol{C}_{SampleAverage} + \lambda \boldsymbol{I}$$

$$\lambda = \frac{\alpha Trace(\boldsymbol{C}_{SampleAverage})}{Rank(\boldsymbol{C}_{SampleAverage})}$$

Where α was set to 0.1.

The class priors were estimated using the sample count of each class.

Minimum probability of error classification was then implemented, using the same decision rule laid out in Question 2. Specifically, 0-1 loss was selected.

After performing classification, the total probability of error was found to be 77.81%. The confusion matrix was also created as shown in Table 6, shown on the next page.

The confusion matrix highlights some interesting points about the classifier. There are a lot of zeros in the matrix. There should be some, as Classes 0, 1, 2, and 10 don't come with any samples in the dataset. For that reason, a decision of 0, 1, 2, or 10 should never be made, and it is not per the confusion matrix. However, the classifier also never makes a decision of 4, 7, 8, or 9, for which there is sample data. This can mostly be explained by looking at the estimated class priors. Classes 4, 8, and 9 have priors of 0.0333, 0.0357, and 0.0010, respectively. They therefore don't make up a large proportion of the population and are unlikely to be chosen. Class 7, with a class prior of 0.1797, likely has a large amount of overlap with another class with a higher prior, and is therefore never chosen. In fact, most of the time, the classifier makes the decision of Class 6, which has the highest prior at 0.4488. The second most frequent decision of Class 5 corresponds to Class 5 having the second highest prior of 0.2975. Overall, because the decision seems to weigh so heavily on the class priors, it can be concluded that there is likely a very high amount of overlap between different

classes and the means are likely very close together. It is also possible that the initial assumption of Gaussian distributions was not correct.

Table 6. Confusion matrix for ERM of wine dataset

		True Class Label										
		0	1	2	3	4	5	6	7	8	9	10
Decision	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0	0	0	0
	3	0	0	0	0.2500	0.0123	0.0034	0.0036	0.0023	0.0229	0	0
	4	0	0	0	0	0	0	0	0	0	0	0
	5	0	0	0	0.2000	0.1963	0.1935	0.1128	0.0466	0.0343	0	0
	6	0	0	0	0.5500	0.7914	0.8030	0.8835	0.9511	0.9429	1.000	0
	7	0	0	0	0	0	0	0	0	0	0	0
	8	0	0	0	0	0	0	0	0	0	0	0
	9	0	0	0	0	0	0	0	0	0	0	0
	10	0	0	0	0	0	0	0	0	0	0	0

To explore these conclusions further, Principal Component Analysis was performed to visualize the dataset. The first two principal components were used, maintaining 98.9% of the variance. Each class was plotted using a different color. The results can be seen in Figure 7, shown on the next page.

Looking at the plot, two results become immediately apparent. First, the shape of the data distribution does not closely match that of a Gaussian distribution. The assumption of a Gaussian distribution is therefore not very robust, and likely introduces a good amount of error. In the future, if sample data is to represent the population, it should be plotted before assuming a certain distribution.

Second, it is easy to see that all the classes overlap a lot, with their means very close together. In fact, the average distance between the means of any two classes (7.65) is smaller than the average standard deviation of the features for each class (10.34). This numerically confirms what is shown visually in the plot.

All together, the rather poor performance of this classifier comes down to the facts that the different classes have an extremely high overlap and that the distributions are not Gaussian as assumed.

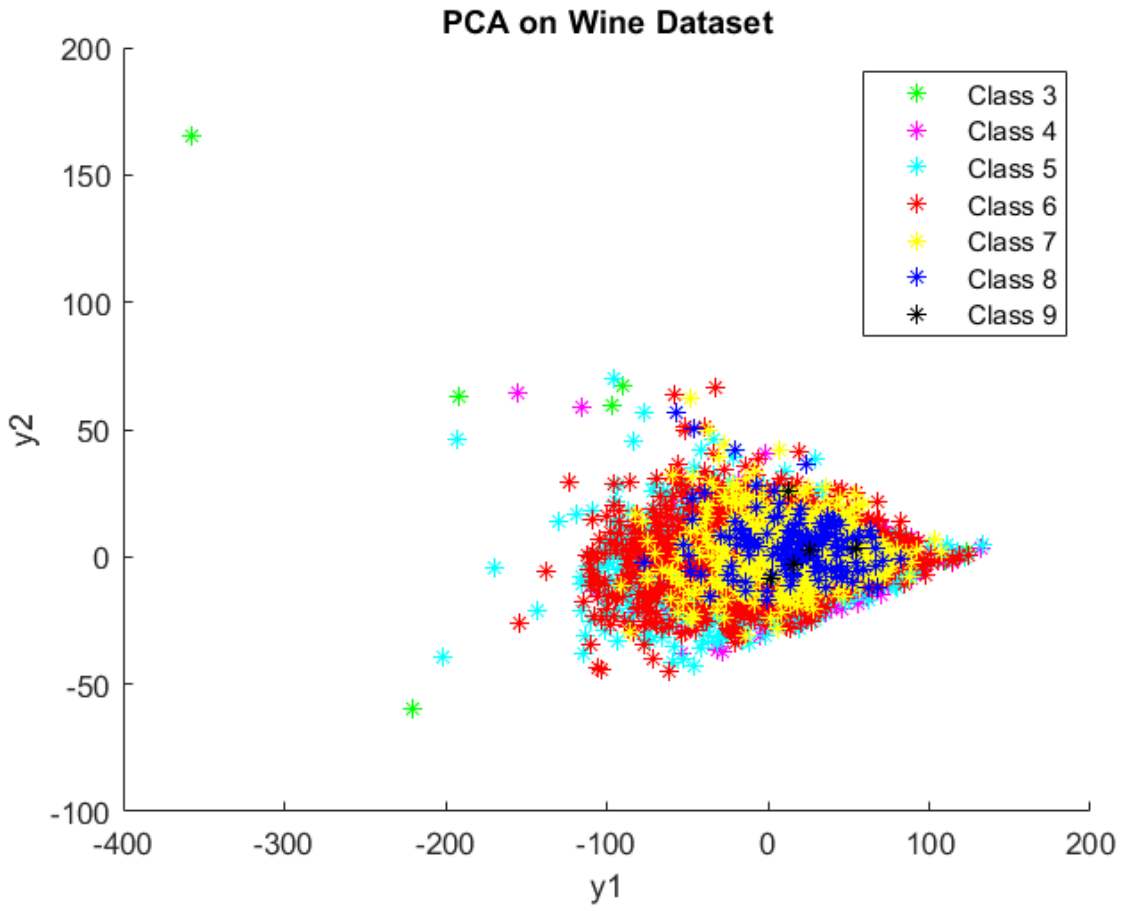

Figure 7. First two principal components of the wine dataset, plotted by class

Part B

As in Part A, for the human activity dataset, the sample data was imported and conditioned. The same minimum probability or error classifier was implemented, again with 0-1 loss.

After performing classification, the total probability of error was found to be 1.71%. The confusion matrix was also created as shown in Table 7, shown on the next page.

From the confusion matrix as well as the total probability of error, it is clear that this classifier was very effective for this sample dataset. Examining the confusion matrix a little more closely, the classifier was able to correctly classify every point in Classes 2 and 6, while also performing incredibly well for the other classes. There were additionally no incorrect decisions assuming another class was Class 6. Overall, this high of performance implies that the means of the class sample distributions for this dataset are well spread out with little overlap.

Table 7. Confusion matrix for ERM of wine dataset

		True Class Label					
		1	2	3	4	5	6
Decision	1	0.9983	0	0	0	0	0
	2	0.0012	1.000	0.0206	0	0	0
	3	0.0006	0	0.9794	0	0	0
	4	0	0	0	0.9240	0.0047	0
	5	0	0	0	0.0760	0.9953	0
	6	0	0	0	0	0	1.000

Again, to explore this conclusion further, Principal Component Analysis was performed to visualize the dataset. The first three principal components were used, maintaining 71% of the variance. Each class was plotted using a different color. The results can be seen in Figure 8.

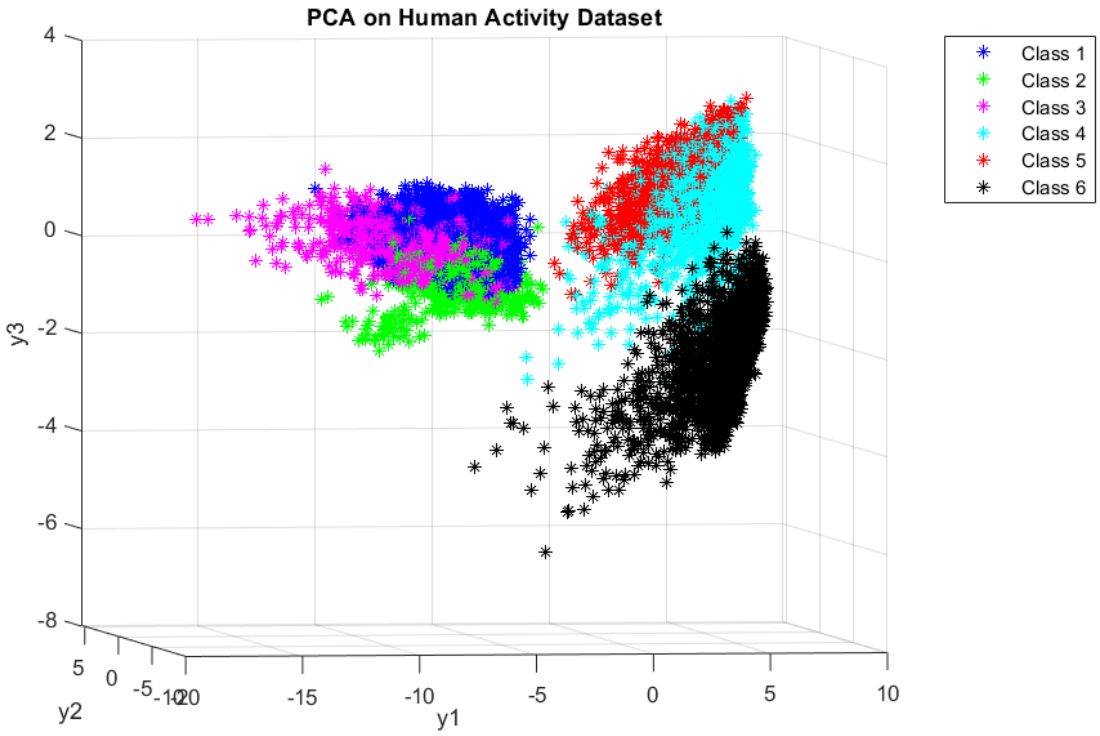

Figure 8. First three principal components of the human activity dataset, plotted by class

Examining the plot, the conclusion drawn above is easy to confirm. It is clear that the different classes are widely spread apart, with very little overlap. Classes 1, 2, and 3 are clustered near eachother, Classes 4 and 5 are near eachother, and Class 6 stands alone. Looking back to the

confusion matrix, the incorrect decisions were made only in each of these clusters. To confirm the spread of the data, we again look at the average distance between means (1.39) and the average standard deviation (0.20). The average distance between means of classes is about 7 standard deviations, more than enough to ensure little overlap.

As with the wine dataset, plotting the data shows that the data do not fit well to a Gaussian distribution as assumed. While this would normally be a source of misclassification and error, the classes are simply far enough away that an incorrect distribution assumption is acceptable for this case.

Appendix A – MATLAB Code for Question 1

```
%%%%%%
% Matthew Dottinger
% EECE5644
% Homework 1 Question 1
% Significant portions of this code were derived from these sources
% g/Practice/EECE5644_2020Spring/EECE5644_2020Spring_TakeHome1Solutions_v20200307.pdf
% g/Practice/EECE5644_2020Summer2/EECE5644_2020summer2_TakeHome1Solution_Matlab.pdf
%%%%%%

%% Setup and Sample Generation

clear all;
close all;

N = 10000; %number of samples
n = 4; %number of dimensions

% Class priors and class conditional distributions
p = [0.7, 0.3]; %class priors
mu(:,1) = [-1;  1;  -1;  1;];
mu(:,2) = [1; 1;  1;  1;];
sigma(:,:,1) = [2 -0.5 0.3 0
    -0.5 1 -0.5 0
    0.3 -0.5 1 0
    0 0 0 2];
sigma(:,:,2) = [1 0.3 -0.2 0
    0.3 2 0.3 0
    -0.2 0.3 1 0
    0 0 0 3];

% Data generation and labelling
label = rand(1,N) >= p(1); %generate labels based on class priors
NumClass = [sum(label==0),sum(label==1)]; %number of samples belonging to each class
x = zeros(n,N);
x(:, label==0) = mvnrnd(mu(:,1), sigma(:,:,1),  NumClass(1))';
x(:, label==1) = mvnrnd(mu(:,2), sigma(:,:,2),  NumClass(2))';

%% Part A - ERM with True Knowledge

% Calculate discriminant score
dscScr = log(mvnpdf(x', mu(:,2)', sigma(:,:,2)))' - log(mvnpdf(x', mu(:,1)',
sigma(:,:,1)))';

% Vary gamma from (effectively) 0 to inf, and take ln
logGamma = log(logspace(-18,18,1000));

% For each gamma, calculate TP, TN, FP and FN rates
for i = 1:length(logGamma)
    decision = dscScr > logGamma(i);
    pTP(i) = sum(decision==1 & label==1)/NumClass(2);
    pTN(i) = sum(decision==0 & label==0)/NumClass(1);
    pFP(i) = sum(decision==1 & label==0)/NumClass(1);
    pFN(i) = sum(decision==0 & label==1)/NumClass(2);
    pE(i) = pFP(i)*p(1) + pFN(i)*p(2); %error probability
end

% Plot ROC
figure
plot(pFP, pTP, 'b-', 'DisplayName', 'ROC Curve')
hold on
```

```
title("Expected Risk Minimization ROC Curve")
xlabel("P(D=1|L=0) False Positive")
ylabel("P(D=1|L=1) True Positive")
grid on

% Find minimum error threshold from above and add to plot
[minpE, minpEind] = min(pE);
plot(pFP(minpEind), pTP(minpEind), 'rx', 'DisplayName', 'Estimated Min. Error')

% Find theoretical minimum error threshold using class priors and plot
theoreticalLogGamma = log(p(1))-log(p(2));
decision = dscScr > theoreticalLogGamma;
TpTP = sum(decision==1 & label==1)/NumClass(2);
TpTN = sum(decision==0 & label==0)/NumClass(1);
TpFP = sum(decision==1 & label==0)/NumClass(1);
TpFN = sum(decision==0 & label==1)/NumClass(2);
TpE = TpFP*p(1) + TpFN*p(2);
plot(TpFP, TpTP, 'go', 'DisplayName', 'Theoretical Min. Error')
legend('Location', 'southeast')
hold off

%% Part B - Naive Bayesian Classifier

% New assumed covariances, data remains the same
NBsigma(:,:,1) = [2 0 0 0
    0 1 0 0
    0 0 1 0
    0 0 0 2];
NBsigma(:,:,2) = [1 0 0 0
    0 2 0 0
    0 0 1 0
    0 0 0 3];

% From here, the process is the same as the above, just using NBsigma
% Calculate discriminant score
dscScr = log(mvnpdf(x', mu(:,2)', NBsigma(:,:,2)))' - log(mvnpdf(x', mu(:,1)',
NBsigma(:,:,1)))';

% For each gamma, calculate TP, TN, FP and FN rates
for i = 1:length(logGamma)
    decision = dscScr > logGamma(i);
    NBpTP(i) = sum(decision==1 & label==1)/NumClass(2);
    NBpTN(i) = sum(decision==0 & label==0)/NumClass(1);
    NBpFP(i) = sum(decision==1 & label==0)/NumClass(1);
    NBpFN(i) = sum(decision==0 & label==1)/NumClass(2);
    NBpE(i) = NBpFP(i)*p(1) + NBpFN(i)*p(2); %error probability
end

% Plot ROC
figure
plot(NBpFP, NBpTP, 'b-', 'DisplayName', 'ROC Curve')
hold on
title("ROC Curve for Naive Bayesian Classifier")
xlabel("P(D=1|L=0) False Positive")
ylabel("P(D=1|L=1) True Positive")
grid on

% Find minimum error threshold from above and add to plot
[NBminpE, NBminpEind] = min(NBpE);
plot(NBpFP(NBminpEind), NBpTP(NBminpEind), 'rx', 'DisplayName', 'Estimated Min.
Error')

% Find theoretical minimum error threshold using class priors and plot
```

```
decision = dscScr > theoreticalLogGamma;
NBTpTP = sum(decision==1 & label==1)/NumClass(2);
NBTpTN = sum(decision==0 & label==0)/NumClass(1);
NBTpFP = sum(decision==1 & label==0)/NumClass(1);
NBTpFN = sum(decision==0 & label==1)/NumClass(2);
NBTpE = NBTpFP*p(1) + NBTpFN*p(2);
plot(NBTpFP, NBTpTP, 'go', 'DisplayName', 'Theoretical Min. Error')
legend('Location', 'southeast')
hold off
```

Appendix B – MATLAB Code for Question 2

```
%%%%%%
% Matthew Dottinger
% EECE5644
% Homework 1 Question 2
% Significant portions of this code were derived from this source
% g/Code/ERMwithClabels.m
%%%%%%

%% Setup and Sample Generation

clear all;
close all;

N = 10000; %number of samples
n = 3; %number of dimensions
C = 3; %number of classes

% Class priors and class conditional distributions
p = [0.3, 0.3, 0.4]; %class priors
sigma(:,:,1) = [rand 0 0
    0 rand 0
    0 0 rand];
sigma(:,:,2) = [rand 0 0
    0 rand 0
    0 0 rand];
sigma(:,:,3) = [rand 0 0
    0 rand 0
    0 0 rand];
sigma(:,:,4) = [rand 0 0
    0 rand 0
    0 0 rand];
averageStdDev = trace(sum(sqrt(sigma),3))/16; %offset means by 2 std devs
mu(:,1) = [0; 0; 0];
mu(:,2) = [2*averageStdDev; 0; 0];
mu(:,3) = [averageStdDev; averageStdDev*sqrt(3); 0];
mu(:,4) = [averageStdDev; averageStdDev*(sqrt(3)/3); averageStdDev*sqrt(8/3)];

% Data generation and labelling
label = rand(1,N);
for i = 1:length(label)
    if label(i) < p(1)
        label(i) = 1;
    elseif label(i) < (p(2)+p(1))
        label(i) = 2;
    elseif label(i) < ((p(3)/2)+p(2)+p(1)) %two subclasses for the last class, will be
combined later
        label(i) = 3;
    else
        label(i) = 4;
    end
end
NumClass = [sum(label==1),sum(label==2),sum(label==3),sum(label==4)];
x = zeros(n,N);
x(:, label==1) = mvnrnd(mu(:,1), sigma(:,:,1),  NumClass(1))';
x(:, label==2) = mvnrnd(mu(:,2), sigma(:,:,2),  NumClass(2))';
x(:, label==3) = mvnrnd(mu(:,3), sigma(:,:,3),  NumClass(3))';
x(:, label==4) = mvnrnd(mu(:,4), sigma(:,:,4),  NumClass(4))';
% Combine labels 2 and 3 into one class under label 2
for i = 1:length(label)
    if label(i) == 4
```

```
        label(i) = 3;
    end
end
NumClass = [sum(label==1),sum(label==2),sum(label==3)];

% plot generated data
figure
scatter3(x(1, label==1),x(2, label==1),x(3, label==1),'bo')
hold on
scatter3(x(1, label==2),x(2, label==2),x(3, label==2),'r*')
scatter3(x(1, label==3),x(2, label==3),x(3, label==3),'gx')
title('True Data Distributions')
legend('Class 1','Class 2','Class 3')
xlabel('x1')
ylabel('x2')
zlabel('x3')
hold off

%% Part A - MAP Classifier with True Knowledge

% Evaluate class conditional pdfs
pxgivenl(1,:) = mvnpdf(x', mu(:,1)', sigma(:,:,1))';
pxgivenl(2,:) = mvnpdf(x', mu(:,2)', sigma(:,:,2))';
pxgivenl(3,:) = .5*mvnpdf(x', mu(:,3)', sigma(:,:,3))' + .5*mvnpdf(x', mu(:,4)',
sigma(:,:,4))'; %two distributions for class 3

% Find class posteriors
px = p*pxgivenl; %total probability
plgivenx = pxgivenl.*repmat(p',1,N)./repmat(px,C,1); %class posterior functions

% Loss matrix, 0-1 loss provides minimum probability of error
lossMatrix = ones(3,3)-eye(3);
expectedRisks = lossMatrix*plgivenx;
[~,decisions] = min(expectedRisks,[],1);

% Make confusion matrix and plot data
figure
shapes = ['o','*','x'];
for i = 1:C %each decision
    for j = 1:C %each class label
        confusionMatrix(i,j) = sum(decisions==i & label==j)/NumClass(j);
        if i == j
            scatter(i,j) = scatter3(x(1,decisions==i & label==j),x(2,decisions==i &
label==j),x(3,decisions==i & label==j),'g',shapes(j),'DisplayName', ['Class '
num2str(j) ' Correct Classification']);
            hold on
        else
            scatter(i,j) = scatter3(x(1,decisions==i & label==j),x(2,decisions==i &
label==j),x(3,decisions==i & label==j),'r',shapes(j),'DisplayName', ['Class '
num2str(j) ' Incorrect Classification']);
            hold on
        end
    end
end
title('Correct vs. Incorrect Classification')
legend([scatter(1,1) scatter(2,1) scatter(2,2) scatter(3,2) scatter(3,3)
scatter(1,3)])
xlabel('x1')
ylabel('x2')
zlabel('x3')
hold off

%% Part B - Higher Loss for L=3
```

```
% Repeat decision and plotting process as in Part A

% Loss matrix, Lambda_10
lossMatrix10 = [0 1 10
    1 0 10
    1 1 0];
expectedRisks10 = lossMatrix10*plgivenx;
[~,decisions10] = min(expectedRisks10,[],1);

% Make confusion matrix and plot data
figure
for i = 1:C %each decision
    for j = 1:C %each class label
        confusionMatrix10(i,j) = sum(decisions10==i & label==j)/NumClass(j);
        if i == j
            scatter(i,j) = scatter3(x(1,decisions10==i & label==j),x(2,decisions10==i
& label==j),x(3,decisions10==i & label==j),'g',shapes(j),'DisplayName', ['Class '
num2str(j) ' Correct Classification']);
            hold on
        else
            scatter(i,j) = scatter3(x(1,decisions10==i & label==j),x(2,decisions10==i
& label==j),x(3,decisions10==i & label==j),'r',shapes(j),'DisplayName', ['Class '
num2str(j) ' Incorrect Classification']);
            hold on
        end
    end
end
title('Correct vs. Incorrect Classification, Lambda_1_0')
legend([scatter(1,1) scatter(2,1) scatter(2,2) scatter(3,2) scatter(3,3)
scatter(1,3)])
xlabel('x1')
ylabel('x2')
zlabel('x3')
hold off

% Loss matrix, Lambda_100
lossMatrix100 = [0 1 100
    1 0 100
    1 1 0];
expectedRisks100 = lossMatrix100*plgivenx;
[~,decisions100] = min(expectedRisks100,[],1);

% Make confusion matrix and plot data
figure
for i = 1:C %each decision
    for j = 1:C %each class label
        confusionMatrix100(i,j) = sum(decisions100==i & label==j)/NumClass(j);
        if i == j
            scatter(i,j) = scatter3(x(1,decisions100==i &
label==j),x(2,decisions100==i & label==j),x(3,decisions100==i &
label==j),'g',shapes(j),'DisplayName', ['Class ' num2str(j) ' Correct
Classification']);
            hold on
        else
            scatter(i,j) = scatter3(x(1,decisions100==i &
label==j),x(2,decisions100==i & label==j),x(3,decisions100==i &
label==j),'r',shapes(j),'DisplayName', ['Class ' num2str(j) ' Incorrect
Classification']);
            hold on
        end
    end
end
```

```
title('Correct vs. Incorrect Classification, Lambda_1_0_0')
legend([scatter(1,1) scatter(2,1) scatter(2,2) scatter(3,2) scatter(3,3)
scatter(1,3)])
xlabel('x1')
ylabel('x2')
zlabel('x3')
hold off
```

Appendix B – MATLAB Code for Question 3

```
%%%%%%
% Matthew Dottinger
% EECE5644
% Homework 1 Question 3 Part A
% Significant portions of this code were derived from this source
% g/Code/ERMwithClabels.m
% g/Code/pca.m
%%%%%%

%% Wine Dataset Class Conditional/Prior Estimation

clear all;
close all;

% import data from excel
x = readmatrix('winequality-white.xlsx', 'Range', 'A2:K4899')';
label = readmatrix('winequality-white.xlsx', 'Range', 'L2:L4899')';
N = size(x,2); %number of samples
n = size(x,1); %number of dimensions
C = 11; %number of classes

alpha = 0.1; %for regularization
sigmaTotal = cov(x'); %for regularization

% Estimate class conditionals (gaussian) and class priors
for i = 1:C
    % estimate class prior
    p(i) = sum(label==i-1)/N;
    % estimate mean
    mu(:,i) = mean(x(:,label==i-1),2);
    % estimate covariance matrix
    sigma(:,:,i) = cov(x(:,label==i-1)');
    % regularize covariance matrix
    sigma(:,:,i) = sigma(:,:,i) +
eye(size(sigma,1))*alpha*trace(sigmaTotal)/rank(sigmaTotal);
end

%% Wine Dataset ERM

% Evaluate class conditional pdfs
for i = 1:C
    if sum(isnan(sigma(:,:,i)))==0
        pxgivenl(i,:) = mvnpdf(x', mu(:,i)', sigma(:,:,i))';
    else
        pxgivenl(i,:) = zeros(1,4898); %zero for classes without data
    end
end

% Find class posteriors
px = p*pxgivenl; %total probability
plgivenx = pxgivenl.*repmat(p',1,N)./repmat(px,C,1); %class posterior functions

% Loss matrix, 0-1 loss provides minimum probability of error
lossMatrix = ones(C,C)-eye(C);
expectedRisks = lossMatrix*plgivenx;
[~,decisions] = min(expectedRisks,[],1);
decisions = decisions-1; %because classes start at 0

% Total error probability estimate
countError = sum(label~=decisions);
```

```
pE = countError/N;

% Make confusion matrix
for i = 1:C %each decision
    for j = 1:C %each class label
        if sum(isnan(sigma(:,:,j)))==0
            confusionMatrix(i,j) = sum(decisions==i-1 & label==j-1)/sum(label==j-1);
        else
            confusionMatrix(i,j) = 0;
        end
    end
end

%% Wine Dataset PCA and Plotting

% Without classes, sample based estimates of distribution (gaussian)
muHat = mean(x,2);
% sigmaTotal is calculated above

% Make data zero-mean
xzm = x - muHat*ones(1,N);

% Get and sort eignvalues/eigenvectors
[Q,D] = eig(sigmaTotal);
[d,ind] = sort(diag(D),'descend');
Q = Q(:,ind);
D = diag(d);

% Calculate the first two principal components for visualization
y = Q(:,1:2)'*xzm;

% Percent of variance maintained
percentVar = trace(D(1:2,1:2))/trace(D);

% Plot data after PCA
figure
hold on
plot(y(1,label==3),y(2,label==3),'g*','DisplayName','Class 3')
plot(y(1,label==4),y(2,label==4),'m*','DisplayName','Class 4')
plot(y(1,label==5),y(2,label==5),'c*','DisplayName','Class 5')
plot(y(1,label==6),y(2,label==6),'r*','DisplayName','Class 6')
plot(y(1,label==7),y(2,label==7),'y*','DisplayName','Class 7')
plot(y(1,label==8),y(2,label==8),'b*','DisplayName','Class 8')
plot(y(1,label==9),y(2,label==9),'k*','DisplayName','Class 9')
title('PCA on Wine Dataset')
xlabel('y1')
ylabel('y2')
legend
hold off

% Average distance between means and average standard deviation
counterDist = 0;
counterSig = 0;
for i = 1:C
    if sum(isnan(sigma(:,:,i)))==0
        counterSig = counterSig+1;
        standardDev(i) = trace(sqrt(sigma(:,:,i)))/size(sigma,1);
    end
    for j = 1:C
        if sum(isnan(sigma(:,:,j)))==0
            if i < j
                counterDist = counterDist+1;
                distances(i) = sqrt(sum((x(:,i)-x(:,j)).^2));
```

```
            end
        end
    end
end
averageDistance = sum(distances)/counterDist;
averageStdDev = sum(standardDev)/counterSig;



%%%%%%
% Matthew Dottinger
% EECE5644
% Homework 1 Question 3 Part B
% Significant portions of this code were derived from this source
% g/Code/ERMwithClabels.m
% g/Code/pca.m
%%%%%%

%% Human Activity Dataset Class Conditional/Prior Estimation

clear all;
close all;

% import data from excel
x_test = readmatrix('activity-data.xlsx', 'Sheet', 'X_test', 'Range', 'B2:UP2948')';
x_train = readmatrix('activity-data.xlsx', 'Sheet', 'X_train', 'Range', 'B2:UP7353')';
label_test = readmatrix('activity-data.xlsx', 'Sheet', 'y_test', 'Range',
'A2:A2948')';
label_train = readmatrix('activity-data.xlsx', 'Sheet', 'y_train', 'Range',
'A2:A7353')';
x = horzcat(x_test,x_train);
label = horzcat(label_test,label_train);
N = size(x,2); %number of samples
n = size(x,1); %number of dimensions
C = 6; %number of classes

alpha = 0.1; %for regularization
sigmaTotal = cov(x'); %for regularization

% Estimate class conditionals (gaussian) and class priors
for i = 1:C
    % estimate class prior
    p(i) = sum(label==i)/N;
    % estimate mean
    mu(:,i) = mean(x(:,label==i),2);
    % estimate covariance matrix
    sigma(:,:,i) = cov(x(:,label==i)');
    % regularize covariance matrix
    sigma(:,:,i) = sigma(:,:,i) +
eye(size(sigma,1))*alpha*trace(sigmaTotal)/rank(sigmaTotal);
end

%% Human Activity Dataset ERM

% Evaluate class conditional pdfs
for i = 1:C
    if sum(isnan(sigma(:,:,i)))==0
        pxgivenl(i,:) = mvnpdf(x', mu(:,i)', sigma(:,:,i))';
    else
        pxgivenl(i,:) = zeros(1,4898); %zero for classes without data
    end
end
```

```
% Find class posteriors
px = p*pxgivenl; %total probability
plgivenx = pxgivenl.*repmat(p',1,N)./repmat(px,C,1); %class posterior functions

% Loss matrix, 0-1 loss provides minimum probability of error
lossMatrix = ones(C,C)-eye(C);
expectedRisks = lossMatrix*plgivenx;
[~,decisions] = min(expectedRisks,[],1);

% Total error probability estimate
countError = sum(label~=decisions);
pE = countError/N;

% Make confusion matrix
for i = 1:C %each decision
    for j = 1:C %each class label
        if sum(isnan(sigma(:,:,j)))==0
            confusionMatrix(i,j) = sum(decisions==i & label==j)/sum(label==j);
        else
            confusionMatrix(i,j) = 0;
        end
    end
end

%% Human Activity Dataset PCA and Plotting

% Without classes, sample based estimates of distribution (gaussian)
muHat = mean(x,2);
% sigmaTotal is calculated above

% Make data zero-mean
xzm = x - muHat*ones(1,N);

% Get and sort eignvalues/eigenvectors
[Q,D] = eig(sigmaTotal);
[d,ind] = sort(diag(D),'descend');
Q = Q(:,ind);
D = diag(d);

% Calculate the first three principal components for visualization
y = Q(:,1:3)'*xzm;

% Percent of variance maintained
percentVar = trace(D(1:3,1:3))/trace(D);

% Plot data after PCA

figure
scatter3(y(1,label==1),y(2,label==1),y(3,label==1),'b*','DisplayName','Class 1')
hold on
scatter3(y(1,label==2),y(2,label==2),y(3,label==2),'g*','DisplayName','Class 2')
scatter3(y(1,label==3),y(2,label==3),y(3,label==3),'m*','DisplayName','Class 3')
scatter3(y(1,label==4),y(2,label==4),y(3,label==4),'c*','DisplayName','Class 4')
scatter3(y(1,label==5),y(2,label==5),y(3,label==5),'r*','DisplayName','Class 5')
scatter3(y(1,label==6),y(2,label==6),y(3,label==6),'k*','DisplayName','Class 6')
title('PCA on Human Activity Dataset')
xlabel('y1')
ylabel('y2')
zlabel('y3')
legend
hold off

% Average distance between means and average standard deviation
```

```
counterDist = 0;
counterSig = 0;
for i = 1:C
    if sum(isnan(sigma(:,:,i)))==0
        counterSig = counterSig+1;
        standardDev(i) = trace(sqrt(sigma(:,:,i)))/size(sigma,1);
    end
    for j = 1:C
        if sum(isnan(sigma(:,:,j)))==0
            if i < j
                counterDist = counterDist+1;
                distances(i) = sqrt(sum((x(:,i)-x(:,j)).^2));
            end
        end
    end
end
averageDistance = sum(distances)/counterDist;
averageStdDev = sum(standardDev)/counterSig;
```