Project 1:
A Comparison of Bayesian and Linear Discriminant Analysis Classifiers

Ian McAtee and Stein Wiederholt

EE5650 Object and Pattern Recognition
Professor Wright
University of Wyoming

October 14, 2021

Abstract

This project provides an introduction to Linear Discriminant Analysis Classifiers (LDA) also called Fisher's linear discriminant or Fisher Discriminant Analysis (FDA). A brief background of LDA theory is provided and a LDA classifier is constructed using MATLAB. The LDA results are compared to those of a Bayesian Classifier for different data sets. Finally, an analysis and discussion of the results is provided.

1 Introduction

A fundamental concept in the fields of machine learning and pattern recognition is the ability to analyze and compare different classification methods. Many classification problems can be solved using a variety of classification techniques, however, some may be better suited to a particular problem than others. Intuition into what classification methods may work can often be gleaned from an examination of the data set. However, in many instances, it may be beneficial to test multiple techniques on the same data in order to determine the best-performing method.

This project aims to compare the performance of two different classifiers, namely, Bayesian and Fisher LDA classifiers. The two classifiers were developed using the same set of training data and both were tested on the same two sets of test data. The differences in classification performance are analyzed and the reasoning behind the differences are discussed. Further, this project examines the training data relative to both sets of test data to determine what effects the data statistics have on the classification performance.

1.1 Bayesian Classifier

A Bayesian classifier is a foundational classifier that utilizes the data statistics in conjunction with Bayes rule to perform classification. The crux of the classification rests on calculating the conditional probability of the class given a set of observed features. One can develop discriminant functions from the training data as a method for evaluating these posterior probabilities. Classification can then be performed by selecting the class corresponding the discriminant function that returned the highest value given a test data point.

For brevity, the details of the discriminant function based Bayesian classifier are omitted here. For reference, McAtee and Stein [1] present a robust discussion of this type of classifier. The Bayesian classifier developed in this project utilizes this same approach.

1.2 Fisher's Linear Discriminant Classification

Classifiers based on the Fisher's linear discriminant analysis (LDA) aim to solve the issue of the "curse of dimensionality". That is, classification methods that work well on data of low dimensionality can become intractable in high dimensional spaces. The obvious solution to this is to somehow reduce the dimensionality of the problem. Fisher's LDA is a simple method to accomplish this dimensional reduction. In fact, Fisher's LDA methods reduce a d-dimensional classification problem simply to a problem of one dimension. This is achieved by noting that d-dimensional data can be projected onto a line in the feature space. However, there exists an infinite number of lines in the d-dimensional space, and projection of the data onto any random line will not guarantee that the class feature data will be well separated in the new one-dimensional space. The crux of Fisher LDA methods is to find the orientation of a line in the d-dimensional space such that after the projection of the d-dimensional class data onto the line, the class data exhibits the maximum possible separation in the new one-dimensional space.

1.2.1 Transformation from D-Dimensional to 1-Dimensional Space

Recall that d-dimensional data $\boldsymbol{x}$ can be projected onto any line defined in a direction of a vector $\boldsymbol{w}$ via the transformation:

$$y = \boldsymbol{w}^T \boldsymbol{x} \tag{1}$$

Here, y is the distance from the origin along the line defined by the direction of $\boldsymbol{w}$. If $||\boldsymbol{w}|| = 1$, that is $\boldsymbol{w}$ is a unit vector, then the projection of the data $\boldsymbol{x}$ falls orthogonally on the line. This provides for a easy visualization of the projection process, however, it is not strictly required for classification via Fisher's LDA. Suppose $||\boldsymbol{w}|| \neq 1$, the resulting data projections would not fall orthoganally on the line, but would instead be a scaled projection along line. However, if the line was in a direction such that the projection of the data onto the line provided maximum separability in the one-dimensional space, the scaling of the vector would not affect discriminability of the data.

1.2.2 Defining the Criterion Function for LDA

Determining the vector that defines the maximal separation of the data in the one-dimensional space requires the statistical analysis of the d-dimensional class data. Since one wishes to determine the line that maximizes the distance between the means of each class after projection relative to a measure of the standard deviation for each class. Fisher LDA utilizes the scatter of the projected data to accomplish this, instead of the sample variances. The scatter of the ith class is defined as:

$$\tilde{s}_i^2 = \sum_{y \in Y_i} (y - \tilde{m}_i)^2 \tag{2}$$

Where, $\tilde{s}_i^2$ is the scatter, y is a projected data point belonging to the set of class Y_i, and $\tilde{m}_i$ is the mean of the ith class that has been projected onto the line. It should be apparent that $1/(n-1)(\tilde{s}_1^2 + \tilde{s}_2^2)$ is an estimate of the unbiased variance of two-class data. Here, $\tilde{s}_1^2 + \tilde{s}_2^2$ is the total within-class scatter of the projected samples. Thus, Fisher's LDA seeks to find the linear function $\boldsymbol{w}^t \boldsymbol{x}$ that maximizes the ratio of the squared distance of the two projected means to the total within-class scatter:

$$J(\boldsymbol{w}) = \frac{|\tilde{m}_1 - \tilde{m}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2} \tag{3}$$

The maximization of the criterion function $J(\boldsymbol{w})$ will result in the Fisher vector $\boldsymbol{w}^*$ that provides for the best separation of the class data in the one-dimensional space. To maximize the criterion function with respect to $\boldsymbol{w}$, $J(\boldsymbol{w})$ needs to be rewritten in terms of scatter matrices, rather than the just the scatter. The in-class scatter matrix of the ith class is defined as:

$$\boldsymbol{S}_i = \sum_{\boldsymbol{x} \in \boldsymbol{D}_i} (\boldsymbol{x} - \boldsymbol{m}_i)(\boldsymbol{x} - \boldsymbol{m}_i)^T \tag{4}$$

Here, $\boldsymbol{x}$ is a d-dimensional data point belonging to class i and $\boldsymbol{m}_i$ is the mean vector of the features of the ith class. It is important to note here that $(n-1)\boldsymbol{S}_i$ is simply the unbiased

estimate of the covariance matrix for class i. Similarly, one can write the within-class scatter matrix as:

$$\boldsymbol{S}_W = \boldsymbol{S}_1 + \boldsymbol{S}_2 \tag{5}$$

Using this, the scatter can be rewritten as:

$$\begin{aligned} \tilde{s}_i^2 &= \sum_{\boldsymbol{x} \in \boldsymbol{D}_i} (\boldsymbol{w}^T \boldsymbol{x} - \boldsymbol{w}^T \boldsymbol{m}_i)^2 \\ &= \sum_{\boldsymbol{x} \in \boldsymbol{D}_i} \boldsymbol{w}^T (\boldsymbol{x} - \boldsymbol{m}_i)(\boldsymbol{x} - \boldsymbol{m}_i)^T \boldsymbol{w} \\ &= \boldsymbol{w}^T \boldsymbol{S}_i \boldsymbol{w} \end{aligned} \tag{6}$$

Similarly, the sum of the in-class scatters can be written:

$$\tilde{s}_1^2 + \tilde{s}_2^2 = \boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w} \tag{7}$$

And the separation between the two projected means is given by:

$$\begin{aligned} (\tilde{m}_1 - \tilde{m}_2)^2 &= (\boldsymbol{w}^T \boldsymbol{m}_1 - \boldsymbol{w}^T \boldsymbol{m}_2)^2 \\ &= \boldsymbol{w}^T (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^T \boldsymbol{w} \\ &= \boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w} \end{aligned} \tag{8}$$

Here, $\boldsymbol{S}_B$ is known as the between-class scatter matrix and is equal to the term $(\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^T$. The criterion function can now be rewritten in terms of the scatter matrices as:

$$J(\boldsymbol{w}) = \frac{\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}} \tag{9}$$

1.2.3 Maximization of the LDA Criterion Function as a Rayleigh Quotient

Note that the above form of the criterion function is in the form of a generalized Rayleigh quotient:

$$R(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{x}) = \frac{\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{B} \boldsymbol{x}} \tag{10}$$

Where $\boldsymbol{A}$ is an nxn symmetric matrix, $\boldsymbol{B}$ is an nxn positive definite matrix, and $\boldsymbol{x}$ is a nx1 vector. The optimization of generalized Rayleigh quotients is well studied in mathematics and can be done with eigenvector and eigenvalue analysis. Recall that the optimal Fisher vector is a vector that maximizes the criterion function:

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmax}}\, J(\boldsymbol{w}) \tag{11}$$

This can be solved as a constrained optimization problem such that:

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmax}}\, \boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w} \tag{12}$$

Subject to the constraint:

$$\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w} = 1 \tag{13}$$

Using the method of Lagrange multipliers, one can write the Lagrangian function as:

$$\mathcal{L} = \boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w} + \lambda(1 - \boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}) \tag{14}$$

To maximize, take the partial derivative with respect to $\boldsymbol{w}$:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = \boldsymbol{S}_B \boldsymbol{w} - \lambda \boldsymbol{S}_W \boldsymbol{w} \tag{15}$$

Set to zero:

$$\boldsymbol{S}_B \boldsymbol{w} = \lambda \boldsymbol{S}_W \boldsymbol{w} \tag{16}$$

Note that this can be rearranged into a general eigenvalue problem:

$$\boldsymbol{S}_W^{-1} \boldsymbol{S}_B \boldsymbol{w} = \lambda \boldsymbol{w} \tag{17}$$

Where the $\boldsymbol{w}^*$ that will maximize the criterion function is the generalized eigenvector of the largest eigenvalue. However, it is not actually necessary in this case to solve for the eigenvectors and eigenvalues. It can be noted that the term $\boldsymbol{S}_B \boldsymbol{w}$ inherently lies in the direction of $\boldsymbol{m}_1 - \boldsymbol{m}_2$. Also recall that it has been shown that the scaling of the optimal Fisher vector $\boldsymbol{w}^*$ is not relevant to maintain the separation of the data. Therefore, the solution can be written directly without eigenvector and eigenvalue analysis as:

$$\boxed{\boldsymbol{w}^* \propto \boldsymbol{S}_W^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2)} \tag{18}$$

1.2.4 Maximization of the LDA Criterion Function via Direct Derivation

The solution of the maximization of the LDA criterion function as a form of a Rayleigh quotient using eigenvector and eigenvalue analysis is a classic approach to this problem, however, may be difficult to comprehend for those without mathematical background in such techniques. For this reason, an alternate method for the maximization of the criterion function is presented here using direct derivation.

Rather than use a Langrange multiplier to constrain the problem, one may take the partial derivative of the criterion function and set it to zero as in routine optimization problems:

$$\begin{aligned} \frac{\partial J(\boldsymbol{w})}{\partial \boldsymbol{w}} &= \frac{\partial}{\partial \boldsymbol{w}} \left[\frac{\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}} \right] \\ &= (\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}) \frac{\partial}{\partial \boldsymbol{w}} \left[\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w} \right] - (\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}) \frac{\partial}{\partial \boldsymbol{w}} \left[\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w} \right] \\ &= (\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}) 2 \boldsymbol{S}_B \boldsymbol{w} - (\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}) 2 \boldsymbol{S}_W \boldsymbol{w} \end{aligned} \tag{19}$$

Set derivative equal to zero:

$$\begin{aligned} 0 &= (\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}) 2\boldsymbol{S}_B \boldsymbol{w} - (\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}) 2\boldsymbol{S}_W \boldsymbol{w} \\ 0 &= (\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}) \boldsymbol{S}_B \boldsymbol{w} - (\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}) \boldsymbol{S}_W \boldsymbol{w} \end{aligned} \tag{20}$$

Divide through by the term $\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}$:

$$0 = \left(\frac{\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}} \right) \boldsymbol{S}_B \boldsymbol{w} - \left(\frac{\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}} \right) \boldsymbol{S}_W \boldsymbol{w} \tag{21}$$

Note that the term $(\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w})/(\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w})$ is simply the criterion function $J(\boldsymbol{w})$, thus one can simplify:

$$\begin{aligned} \boldsymbol{S}_B \boldsymbol{w} - J(\boldsymbol{w}) \boldsymbol{S}_W \boldsymbol{w} &= 0 \\ J(\boldsymbol{w}) \boldsymbol{S}_W \boldsymbol{w} &= \boldsymbol{S}_B \boldsymbol{w} \\ J(\boldsymbol{w}) \boldsymbol{w} &= \boldsymbol{S}_W^{-1} \boldsymbol{S}_B \boldsymbol{w} \end{aligned} \tag{22}$$

Now substitute in the definition of $\boldsymbol{S}_B$:

$$J(\boldsymbol{w}) \boldsymbol{w} = \boldsymbol{S}_W^{-1} (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^T \boldsymbol{w} \tag{23}$$

Note that the term $(\boldsymbol{m}_1 - \boldsymbol{m}_2)^T \boldsymbol{w}$ is merely a scaling factor. Let this term be denoted as a constant c:

$$J(\boldsymbol{w}) \boldsymbol{w} = \boldsymbol{S}_W^{-1} (\boldsymbol{m}_1 - \boldsymbol{m}_2) c \tag{24}$$

Solve for $\boldsymbol{w}$:

$$\boldsymbol{w} = \frac{c}{J(\boldsymbol{w})} \boldsymbol{S}_W^{-1} (\boldsymbol{m}_1 - \boldsymbol{m}_2) \tag{25}$$

Here, the term $c/J(\boldsymbol{w})$ simply scales the vector $\boldsymbol{w}$ in the direction of $\boldsymbol{S}_W^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2)$. However, it has been established that the scaling of the optimal Fisher vector $\boldsymbol{w}^*$ is immaterial. Thus, the final solution can be written as:

$$\boxed{\boldsymbol{w}^* \propto \boldsymbol{S}_W^{-1} (\boldsymbol{m}_1 - \boldsymbol{m}_2)} \tag{26}$$

Notice that this result is identical to the result obtained via analysis of the criterion function as a generalized Rayleigh quotient.

1.2.5 Classification with LDA

It has been shown that a vector can be found such that d-dimensional class data can be projected onto a line whose direction will result in maximal separation of the data in the new one-dimensional space. However, a decision boundary must still be defined in order to perform classification. This can be simply accomplished by setting the decision boundary such that it lies equidistant from the projected means of the two classes:

$$DecisionPoint = \frac{1}{2} (\boldsymbol{w}^{*T} \boldsymbol{m}_2 - \boldsymbol{w}^{*T} \boldsymbol{m}_1) \tag{27}$$

This leads to the intuitive classification rule: class 1 is chosen if the projected data sample falls to the side of the decision point that is associated with the projected mean of class 1 and class 2 is chosen if the projected data sample falls on the side that is closer to the projected mean of class 2.

2 Methods and Results

A discriminant function based Bayesian classifier and a Fisher LDA classifier were developed in MATLAB using provided two-featured training data for two classes. The classifiers were tested on two sets of training data for the purposes of individual classifier performance analysis and a direct comparison between the two designed classifiers. All developed MATLAB code is given in its entirety in the Code Listing section of the Appendix to this report. This section presents a detailed explanation of the design of each classifier as well as a presentation of the achieved results.

2.1 Analysis of the Data Sets

Three sets of data were provided via MATLAB *.mat* files containing training and two sets of test data for two classes. The data was organized as Nx2 matrices such that each row constituted a data sample with the elements in the two columns representing two features:

$$\boldsymbol{D_c} = \begin{bmatrix} d_{1,1} & d_{1,2} \\ \vdots & \vdots \\ d_{N,1} & d_{N,2} \end{bmatrix} \tag{28}$$

Here, $\boldsymbol{D_c}$ is data matrix for class c, and $d_{1...N,1}$ and $d_{1...N,2}$ are vectors containing the two features. The training data for class 1 contained N = 1000 data samples and class 2 contained N = 4000 data samples, for a total of 5000 samples of training data. The two sets of test data, denoted as test data A and B, both contained N = 100 data samples of class 1 and N = 400 data samples of class 2. The training data and both sets of test data for both classes is plotted in the two dimensional feature space in figures 1, 2, and 3.

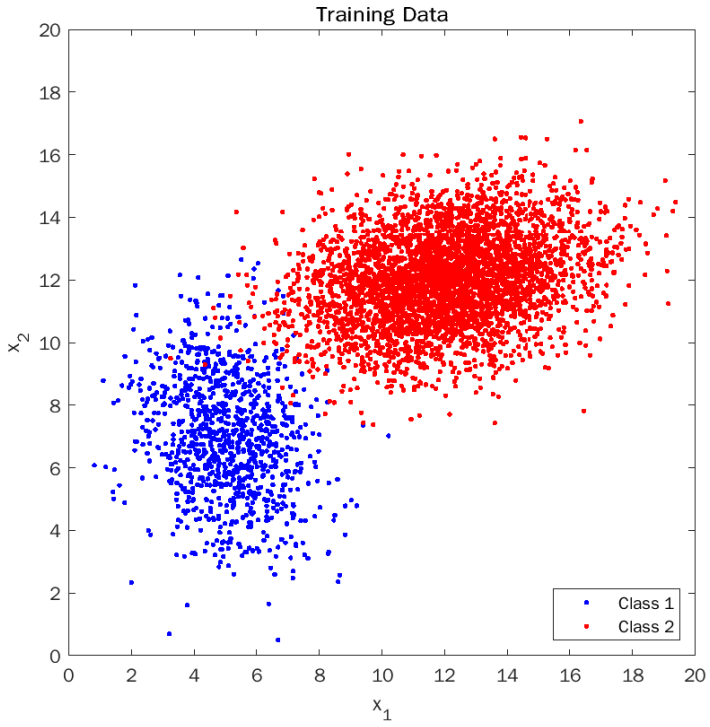

Figure 1: Scatter Plot of Training Data

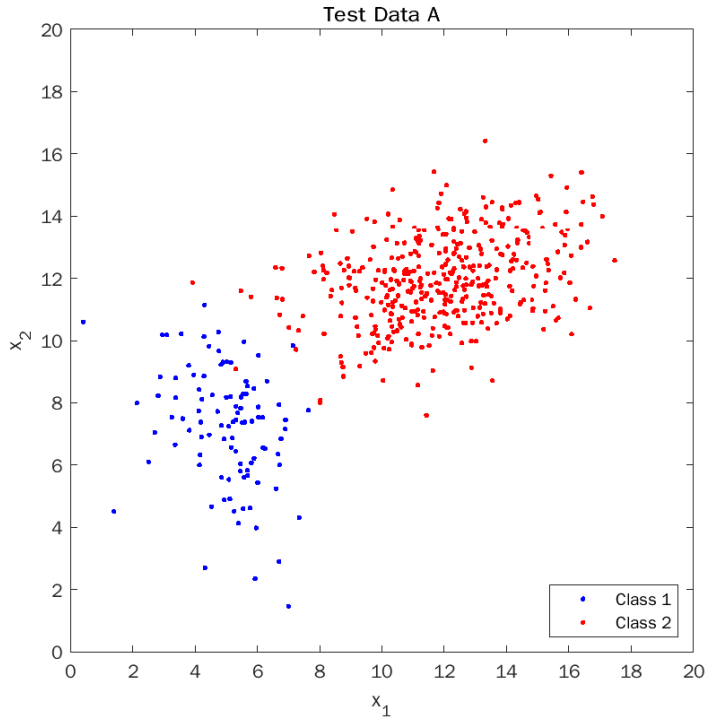

Figure 2: Scatter Plot of Test Data A

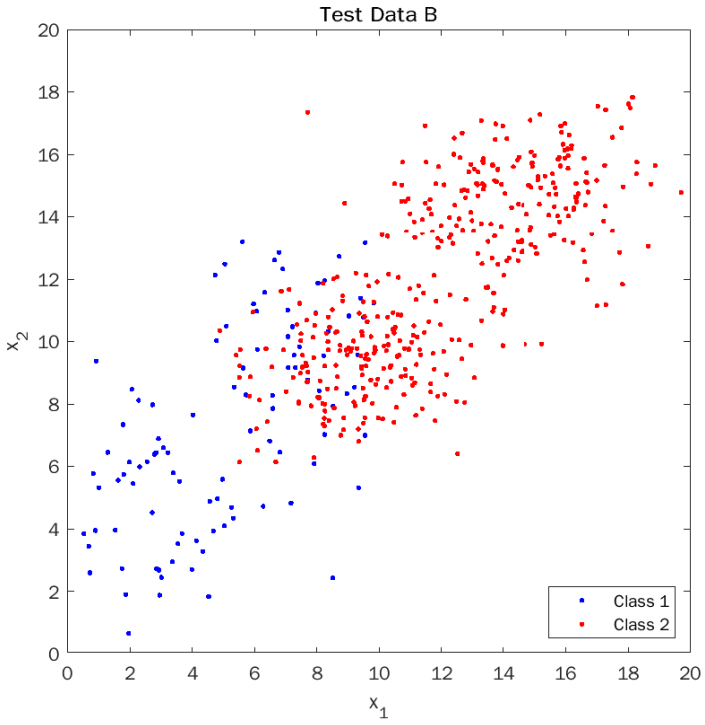

Figure 3: Scatter Plot of Test Data B

To better understand the underlying statistics and relationships between the training and test data sets, histograms were produced of both classes for all data sets. The histograms for the class 1 and class 2 data are shown in figures 4 and 5, respectively.

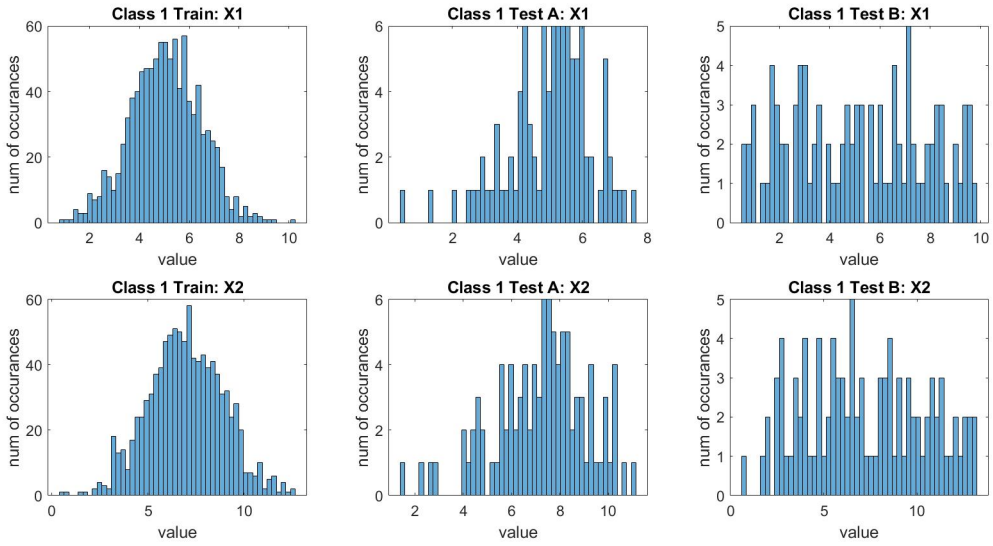

Figure 4: Histograms of Class 1 Data

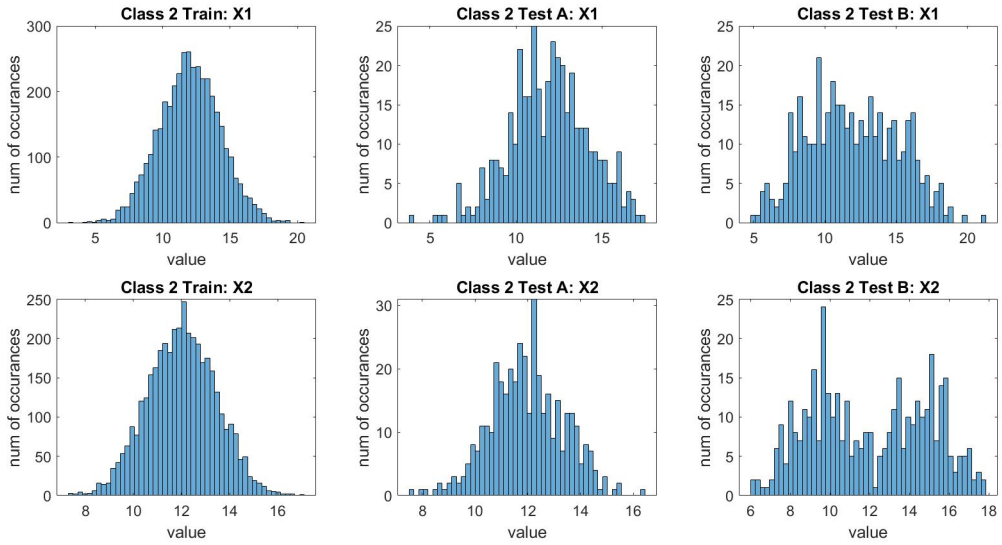

Figure 5: Histograms of Class 2 Data

An analysis of the histograms in figures 4 and 5 reveals that the training data closely follows a Gaussian distribution. The test data set A also appears to have a general Gaussian shape, however, less so than the training data, especially in the class 1 data. Further, the test data set B does not seem to be well defined by a Gaussian distribution. For class 1, test data B looks more akin to a random or noisy uniform distribution, while for class 2, the x_1 dimension may be Gaussian and the x_2 dimension appears to have a multi-modal distribution. Because the Bayesian classifier developed in this project assumed Gaussian distributed data and Fisher's LDA operates well on Gaussian data, it can be expected that the classifiers will perform worse on test data set B due to model error.

For a more exact comparison of the training data set and the test data sets, the sample means, unbiased covariance matrices, and the prior probabilities were computed for both classes of the training and test data sets. These statistics are summarized in figure 6.

	Class 1			Class 2		
	Training	**Test A**	**Test B**	**Training**	**Test A**	**Test B**
Sample Mean $\left(\hat{\mu} = \begin{bmatrix}\hat{\mu}_1 \\ \hat{\mu}_2\end{bmatrix}\right)$	$\begin{bmatrix}5.060 \\ 6.967\end{bmatrix}$	$\begin{bmatrix}5.006 \\ 7.236\end{bmatrix}$	$\begin{bmatrix}5.134 \\ 7.222\end{bmatrix}$	$\begin{bmatrix}12.00 \\ 12.01\end{bmatrix}$	$\begin{bmatrix}11.91 \\ 11.96\end{bmatrix}$	$\begin{bmatrix}11.96 \\ 12.01\end{bmatrix}$
Unbiased Covariance Matrix $(\hat{\Sigma})$	$\begin{bmatrix}1.984 & -0.552 \\ -0.552 & 3.753\end{bmatrix}$	$\begin{bmatrix}1.663 & -0.749 \\ -0.749 & 3.647\end{bmatrix}$	$\begin{bmatrix}7.115 & 5.024 \\ 5.024 & 10.35\end{bmatrix}$	$\begin{bmatrix}5.130 & 0.866 \\ 0.866 & 2.015\end{bmatrix}$	$\begin{bmatrix}5.223 & 1.126 \\ 1.126 & 1.963\end{bmatrix}$	$\begin{bmatrix}10.25 & 6.834 \\ 6.834 & 8.643\end{bmatrix}$
Prior Probability $(P(\omega_i))$	0.20	0.20	0.20	0.80	0.80	0.80

Figure 6: Training and Test Data Statistics

From figure 6, one can see that the statistics between the training data and the test data set A are quite similar. This, in addition to the histogram analysis, provides evidence that the training data is representative of the test data A. However, while the means of test data

set B are similar to that of the training data, the covariance matrices differ substantially. This is to be expected due to the histogram analysis of test data set B. It appears that the data of test data set B was drawn from a different distribution than that of the training data. Without proper investigation, one could be deceived by the fact that the means of the training data and the test data B are practically identical. However, it is clear that the training data data is not representative of test data B and this undoubtedly will inject a degree of model error into the classification.

2.2 Bayesian Classifier

2.2.1 Bayesian Classifier Design

The Bayesian classifier from Mcatee and Stein [1] was implemented using the provided training and test data used for this LDA project. This allows for a direct comparison in accuracy between the Bayesian classifier and the LDA classifier.

The Bayesian classification regions where each discriminant function prevails, also known as the decision regions, were plotted geographically as shown in Figure 7. Here, the training data has been superimposed onto to the decision regions to give an indication of how the decision regions relate to the data.

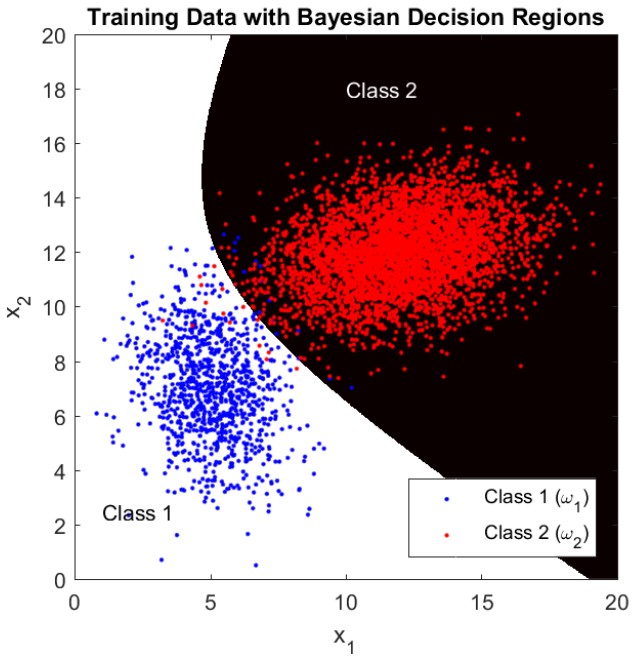

Figure 7: Bayesian Decision Regions with Training Data

2.2.2 Bayesian Classifier Testing

The developed Bayesian classifier was tested on test data sets A and B. Figures 8 and 9 show test data sets A and B superimposed over the decision regions, respectively.

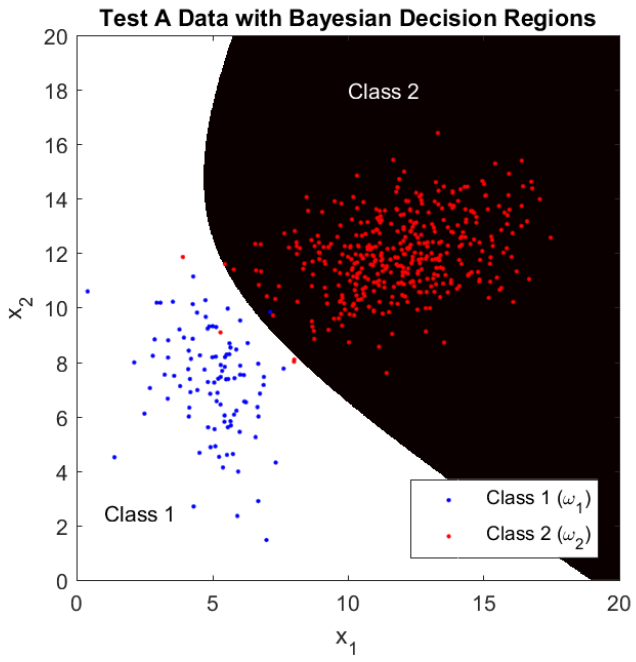

Figure 8: Bayesian Decision Regions with Test Data A

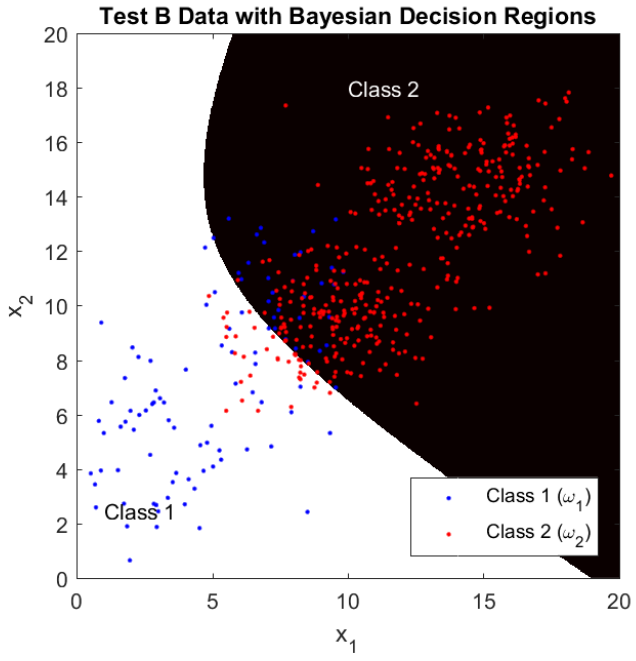

Figure 9: Bayesian Decision Regions with Test Data B

It is clear from examining figures 8 and 9, that the Bayesian classifier should perform quite well on test data A and less so on test data B. The quantitative classification results confirm this, with an accuracy of 99.0% on test data set A and an accuracy of 86.8% on test

data set B. Recall that it was expected that the classification accuracy would be lower for test data set B due to the model error introduced in this data set.

2.3 LDA Classifier

2.3.1 LDA Classifier Design

A Fisher's LDA classifier was constructed using the training data set. The in-class scatter matrices for both classes and the within class scatter matrix was calculated via the equations developed in section 1.2.2. To find the Fisher vector $\boldsymbol{w}^*$ that would provide the greatest separation of the class data after the projection, the method derived for equations 18 and 26 was employed. For purposes of graphical clarity of the LDA projections for these data sets, the Fisher vector $\boldsymbol{w}^*$ was scaled into a unit vector and oriented from the third quadrant into the first quadrant. The Fisher vector obtained after these transformations was:

$$\boldsymbol{w}^* = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0.5851 \\ 0.8110 \end{bmatrix} \tag{29}$$

Note that this forms a line from the origin to point $(x_1, x_2) = (0.5851, 0.8110)$ at an associated angle of 0.9458 radians (54.19 °) from the positive x_1 axis. Figure 10 shows the Fisher vector extended as a longer line and plotted with the training data for both classes:

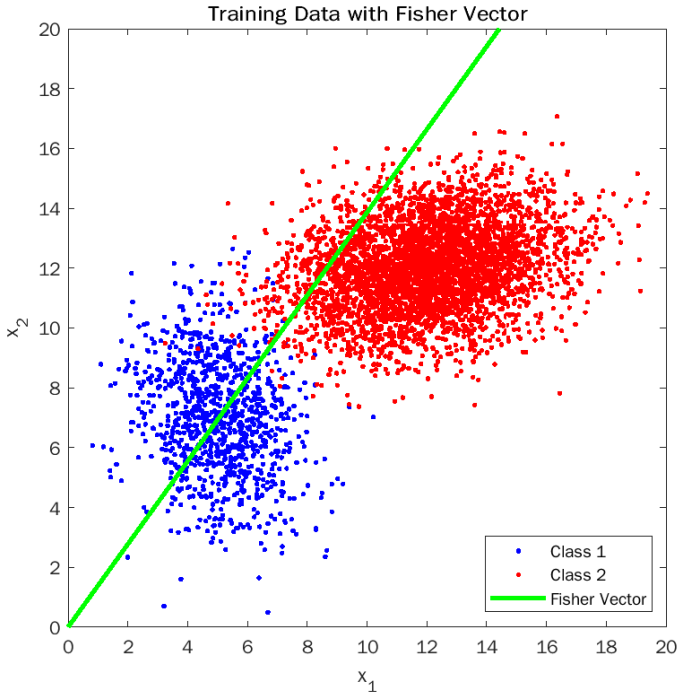

Figure 10: Line Extension of the Fisher Vector and Scatter Plot of the Training Data

To visualize the projection of the training data onto the line defined by the Fisher vector, equation 1 was applied to 60 samples of the training data (30 samples each of class 1 and class 2) and the results were plotted. Figure 11 shows this subset of training data plotted in

the two-dimensional feature space along with the same subset of data after projection onto the Fisher vector:

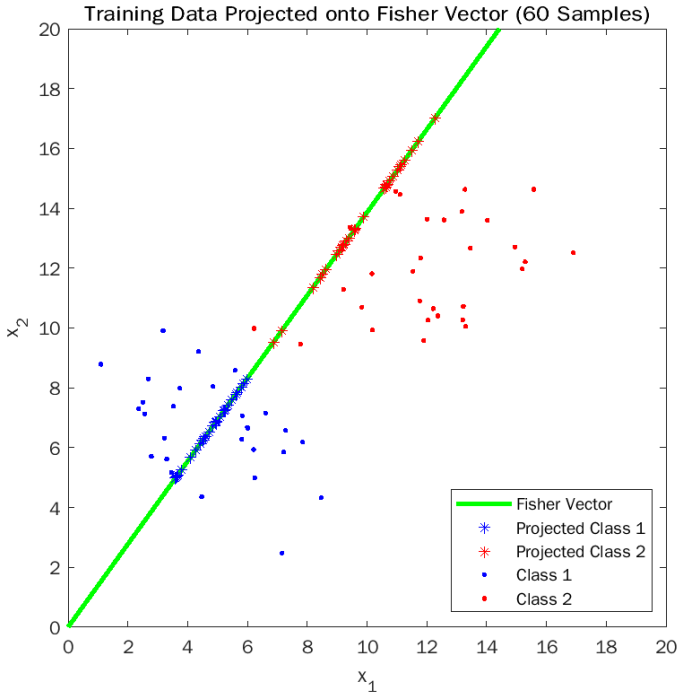

Figure 11: 60 Samples of Training Data Plotted in the Original Feature Space and After Projection onto the Fisher Vector

By examining Figure 11, one can see how the Fisher LDA method projects the data onto the line defined by the Fisher vector. Here, because the Fisher vector was described as a unit vector, the projection of the data occurs in an orthogonal manner.

The decision point for the LDA classifier was then calculated via application of equation 27. It was found that the decision point lies at a distance of 12.6891 from the origin along the line defined by the Fisher vector. Figure 12 shows this decision point on the Fisher vector along with all the training data after projection onto the Fisher vector.

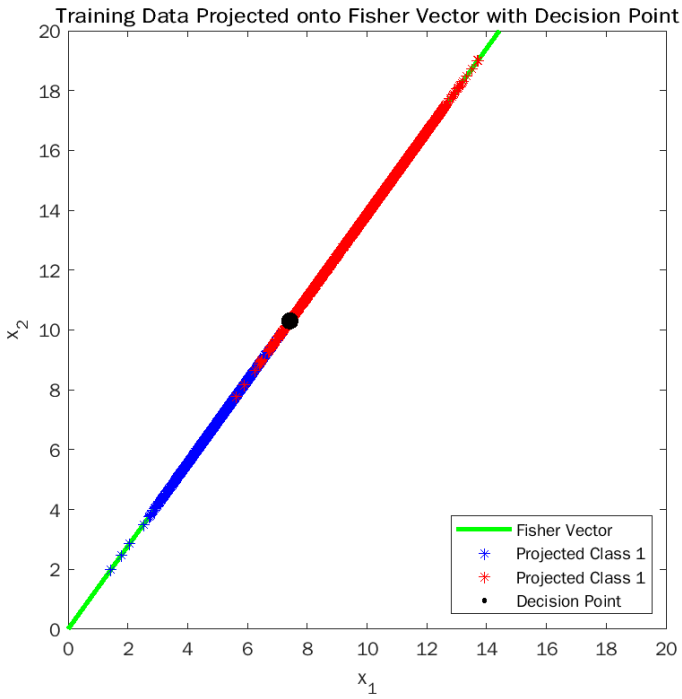

Figure 12: Projected Training Data onto Fisher Vector and the LDA Classifier Decision Point

Looking at Figure 12, it is not immediately apparent that the calculated Fisher vector actually provides for the most optimal separation of the data in the one-dimensional space. To graphically demonstrate this, an orthogonal line was plotted at the decision point along with a scatter plot of the training data in the original two-dimensional space, as shown in Figure 13. It is important to note that this decision line is purely for graphical illustration only, the actual classification of data is performed in the one-dimensional space via comparison with the decision point.

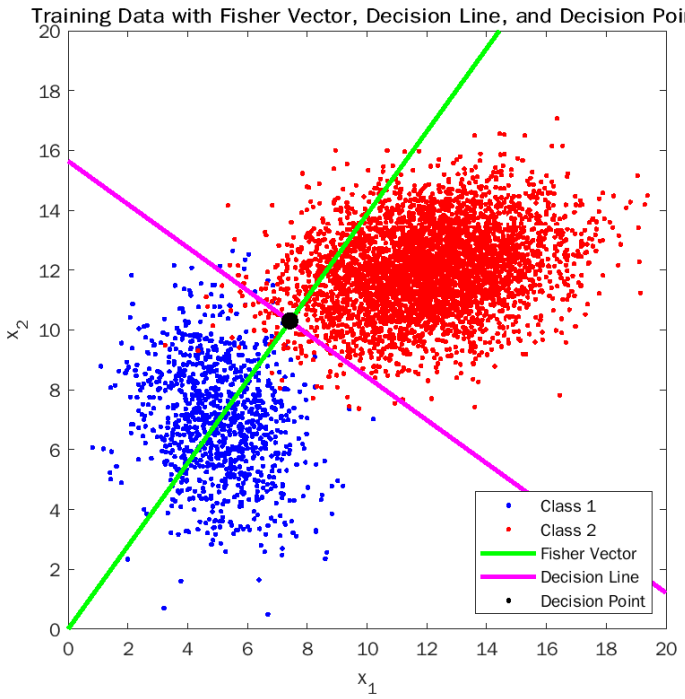

Figure 13: Training Data in the Two-Dimensional Space with Decision Line as Specified by the LDA Decision Point

As one can see from Figure 13, the decision line created from LDA shows good separation of the two classes. In fact, as proved in section 1.2.5, the decision point provides the optimal separation of the classes after projection onto the Fisher vector. To illustrate this, Figure 14 shows 60 samples the training data in the two-dimensional space and projected onto the Fisher vector along with the LDA decision point and line.

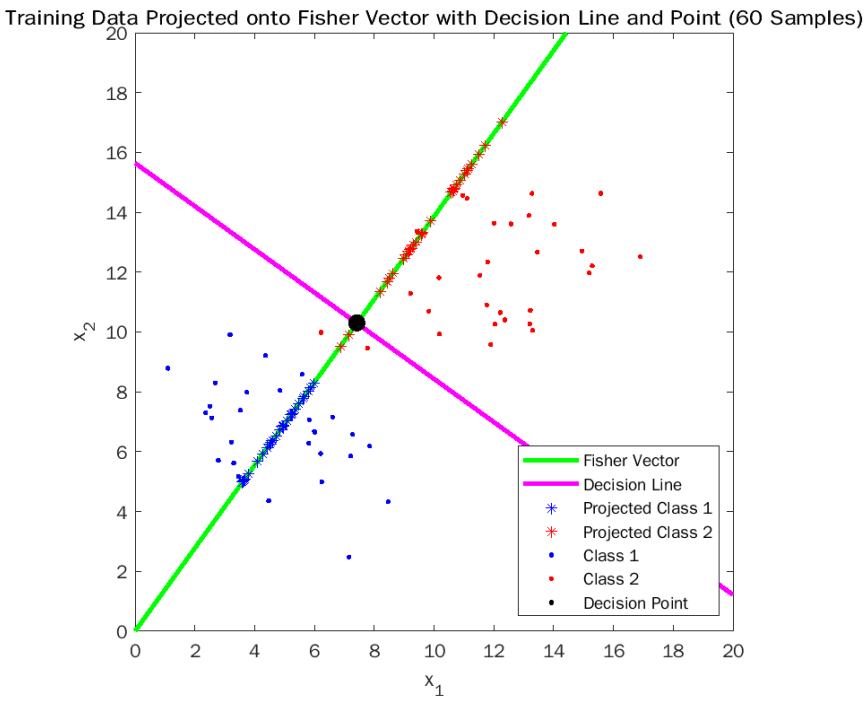

Figure 14: Training Data in the Two-Dimensional Space with Decision Line as Specified by the LDA Decision Point

2.3.2 LDA Classifier Testing

The designed LDA classifier was tested on both the test data set A and the test data set B. Figure 15 shows a scatter plot of test data set A along with the Fisher vector, decision point, and decision line. Figure 16 depicts the test data set A after projection onto the Fisher vector with the decision point.

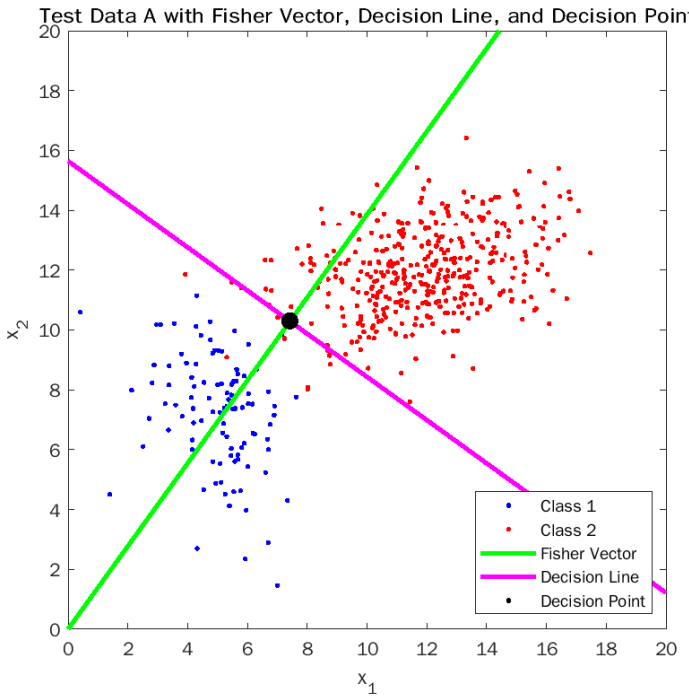

Figure 15: Test Data Set A in the Two-Dimensional Space with Decision Line as Specified by the LDA Decision Point

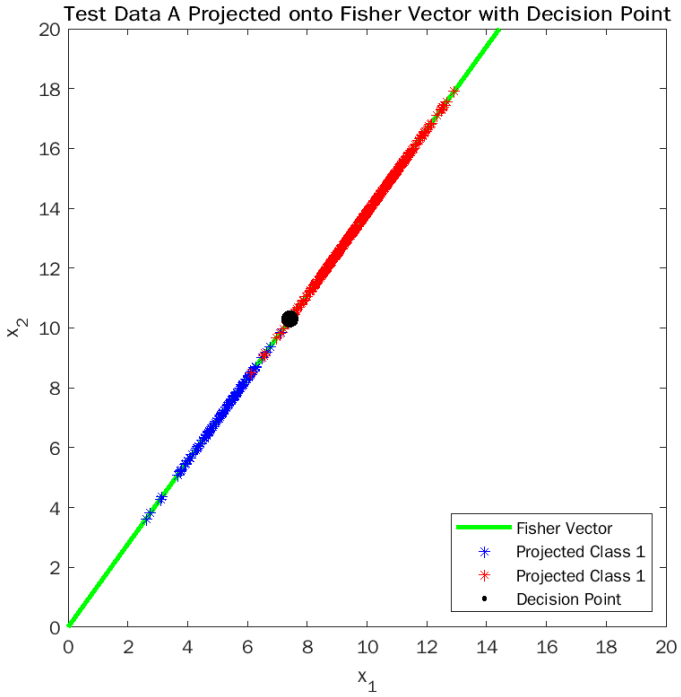

Figure 16: Projected Test Data Set A onto Fisher Vector and the LDA Classifier Decision Point

Analyzing figures 15 and 16, one can see that the Fisher vector and the associated decision point provide for good separation of the class data. Thus, it should be expected that the designed LDA classifier should achieve a high classification accuracy on test data set A.

Similar plots were created for test data set B. Figure 17 shows a scatter plot of test data set B along with the Fisher vector, decision point, and decision line and figure 18 depicts the test data set B projected onto the Fisher vector with the decision point.

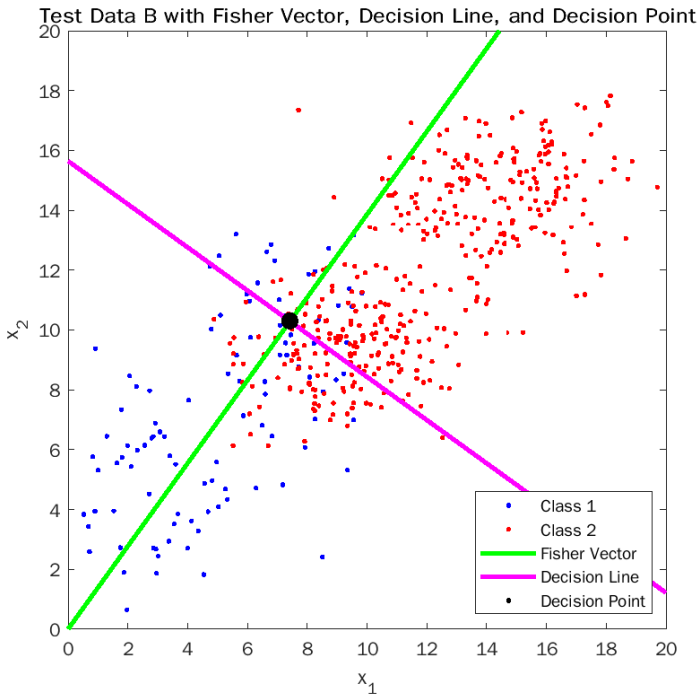

Figure 17: Test Data Set B in the Two-Dimensional Space with Decision Line as Specified by the LDA Decision Point

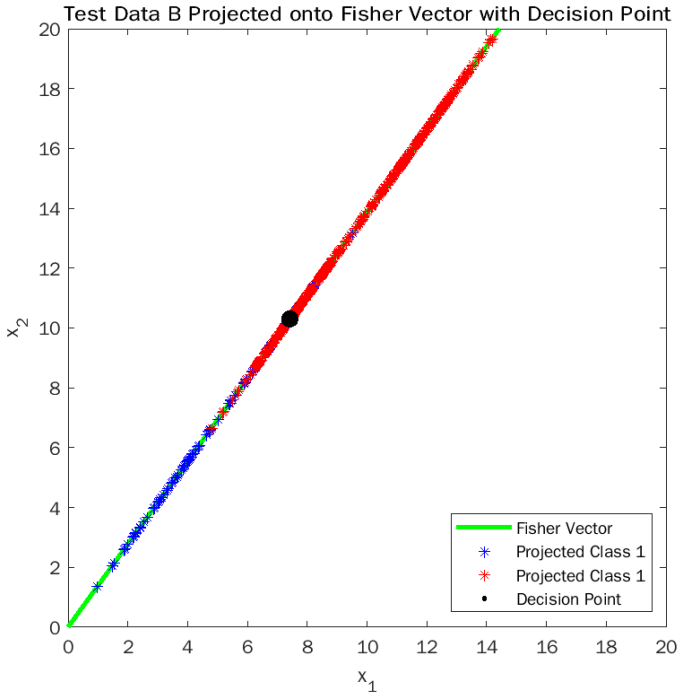

Figure 18: Projected Test Data Set A onto Fisher Vector and the LDA Classifier Decision Point

It can be clearly seen from figures 17 and 18 that the designed LDA classifier does not appear to separate the class data from test data set B as well as it does for test data set A. Therefore, one expects that the classification performance of the LDA classifier should not be as high for test data set B as opposed to test data set A. The final accuracy metrics of the LDA classifier confirm this, with the LDA classifier achieving a classifcation accuracy of 97.6% on test data set A and 82.4% on test data set B. Once again, this is an expected result as the training data was not representative of test data B, thus introducing model error.

3 Discussion and Conclusions

To compare the performance between the Bayesian and LDA classifier, the classification metrics for both classifiers are tabulated side by side in figure 19.

	Bayesian Error (%)	Bayesian Accuracy (%)	LDA Error (%)	LDA Accuracy (%)
Test Data A Class 1	1.00	99.0	0.00	100
Test Data A Class 2	1.00	99.0	3.00	97.0
Test Data A Overall	1.00	99.0	2.40	97.6
Test Data B Class 1	32.0	68.0	19.0	81.0
Test Data B Class 2	8.50	91.5	17.2	82.8
Test Data B Overall	13.2	86.8	17.6	82.4

Figure 19: Classifier Performance by Test Set

As one can see by examining figure 19, both classifiers perform well on test data set A and moderately worse on test data set B. It should be noted that the Bayesian classifier outperforms the LDA classifier on both test data sets to a small degree. This is to be expected as the Bayesian classifier is an optimal classifier that provides for the minimum amount of Bayes error. However, this optimality comes at the computational cost of having to compute and manipulate the statistics of the training data set. For problems of low-order dimensionality such as this one, these computations are not particularly difficult. However, many classification problems can be of quite high dimensionality and these calculations, particularly the covariance matrix inverse and determinant, can prove to be intractable. This leads to the main advantage of the Fisher LDA method, that is, the ability to reduce a multidimensional classification problem to a single dimension. The Fisher LDA method's reduction in dimensionality can transform a possibly intractable high dimensional classification problem to one that is less computationally expensive to train and which may still yield high classification accuracy, albeit less so than that of a Bayesian classifier.

The accuracy of a classifier can be better understood if one examines how errors may occur in the classification process. There will always be a degree of Bayes error in classification problems [1]. However, by selecting features such that the class PDFs exhibit means that are far apart and variances that are relatively narrow, Bayes error can be reduced or practically eliminated. Recall that Bayesian error occurs due to regions of overlap in the PDFs of the class data. Classification errors are likely to occur in this region of overlap. It can be shown that the Bayes error rate is the minimum error rate possible for a particular data set. One can examine the PDFs of the training data to get a cursory idea of how significant the Bayes error will be. Figure 20 below shows the Gaussian PDFs of the class 1 and class 2 training data shown as equiprobable contour lines.

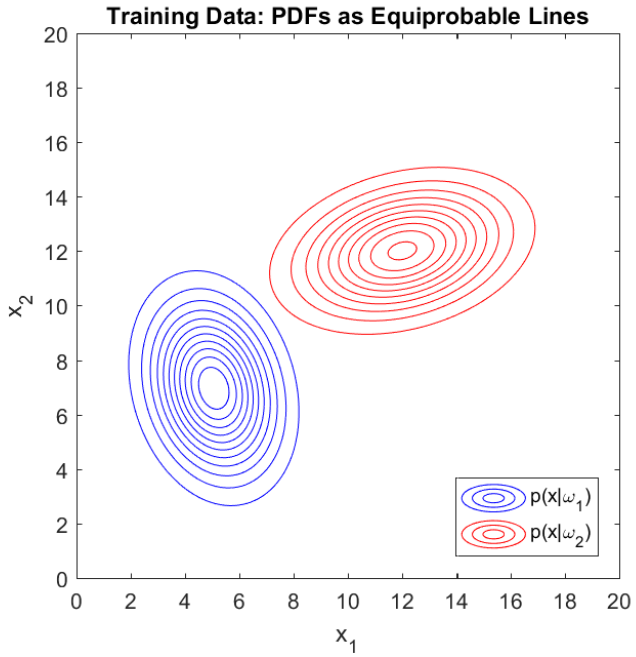

Figure 20: Equiprobable Contour Lines of Training Data Gaussian PDFs

Examining figure 20, one can see that there will be some Bayes error, however, it is difficult to determine how significant it is. For better clarity, the marginal PDFs $p(x_1|\omega_1)$, $p(x_1|\omega_2)$, $p(x_2|\omega_1)$, and $p(x_2|\omega_2)$ were calculated from the feature data. These are shown in figures 21 and 22.

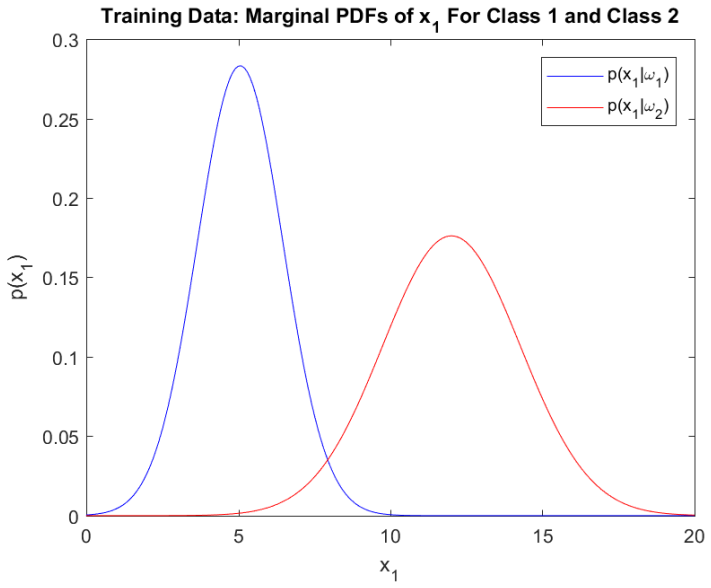

Figure 21: Marginal PDFs of the First Feature for Classes 1 and 2

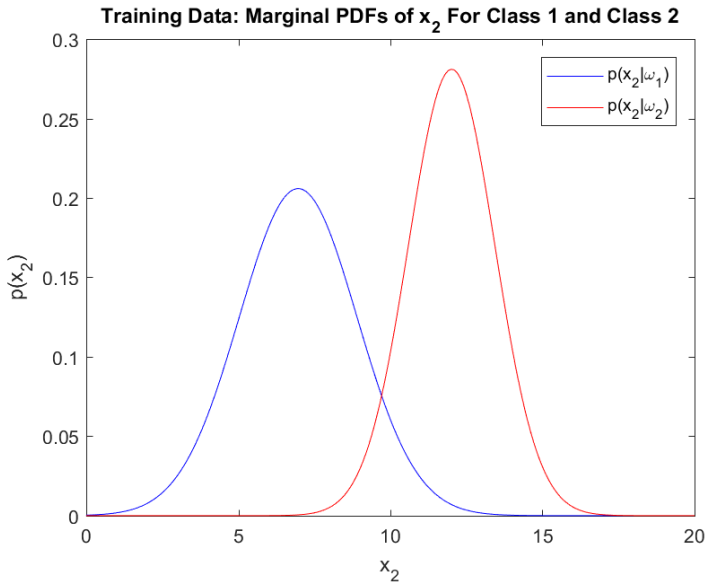

Figure 22: Marginal PDFs of the Second Feature for Classes 1 and 2

Looking at the regions of overlap in figures 21 and 22, one can clearly see that there will be an element of Bayes error in this classification problem. However, the degree of Bayes error present does not appear to be very significant. It is also important to note, that figures

21 and 22 depict PDFs of the training data. If the test data is not representative of the training data (i.e. not originating from the same distribution), one will see the introduction of model error into the classifier.

In examining figures 4, and 5 one can determine that there is going to be some model error in test data set B. As shown in the histograms, test data set B does not follow the same Gaussian distribution that the training data exhibits. Class 1 has a much more noisy, perhaps uniform distribution, while Class 2 may have a Gaussian or multi-modal Gaussian distribution. Whenever the assumption of the underlying PDF of the data is incorrect, the result is the introduction of model error into the classifier. This is especially evident in the Bayesian classification case. Recall that the discriminant functions are developed under the assumption of Gaussian distributed data. Therefore, if the data is not Gaussian in nature, the classifier will not generalize to another distribution and the error will increase. Because of this, it is very important to analyze the data carefully. Bayesian classifiers can be designed to operate on data originating from different distributions simply by developing discriminant functions based on that particular distribution.

The Fisher LDA classification method is also affected by model error, however, not quite in the same way as that of the Bayesian method. The techniques of Fisher LDA do not inherently assume Gaussian distributed data. Rather, LDA relies only on the means of the data to perform the classification. Therefore, LDA should generalize to different distributions, given that they are not multi-modal and the mean is a well-defined useful metric. However, this still assumes that the data for the training and the testing will exhibit the same underlying distribution. Model error affected the LDA classification in this project simply because the training data and test data B were of different distributions, not because the test data B simply was not Gaussian.

It has been shown that knowledge of the parameters of the data distribution are paramount for the design of classifiers. It is important to note that errors can also be introduced into the classifiers based on an inaccurate estimation of these model parameters. For example, in the case of a Bayesian classifier, an inaccurate estimation of the mean can affect the discriminability of class data, while errors in the covariance estimation could increase the Bayes error. It could be argued that a poor estimation of the mean would be more detrimental to the overall classifier performance as compared to a poor estimation of the covariance. Assuming equal priors, the decision region in a Bayesian classifier will be equidistant between the means. Therefore, an erroneous estimation of the means will lead to the incorrect placement of the decision regions and introduce errors into classification, as the decision region is not in the optimal position.

Similarly, because LDA methods rely on only the means of the data to determine the decision point, a poor estimation of the data means will also directly lead to non-optimal classification. In this project, the decision point was set equidistant between the two means in the one-dimensional space. If one or both means were erroneous, one can immediately see that the result would be a decision point that would cause a much greater number of misclassifications. A poor estimation of the scatter of the data may also affect the classifier performance, but not in such a direct or substantial way. Misrepresenting the scatter of the data will lead to a Fisher vector that may not provide for the best separation of the data in the one-dimensional space and will negatively affect classifier performance.

This project provides insight into how an analysis of the data is crucial to the imple-

mentation of classifiers. It is shown that both a Bayesian approach and an LDA approach achieve good classification of these specific data sets. However, the main take-away of this project is that different classifiers are better suited for problems of different types. Intuition on what classifier to use for a particular problem must be learned from an examination of the data. In this project, the data set is of low dimensionality and it is appropriate to utilize a Bayesian approach, as this would provide for the minimum error. If the data was of high dimensionality, perhaps the LDA approach would have been better suited. Lastly, it may sometimes be necessary to try a multitude of classifiers for a problem to determine which will give the best classification. However, there are myriad classification techniques, with more being developed every year. Therefore, an engineer or developer must have an intimate knowledge of the problem data and the benefits and drawbacks of many classification techniques in order to create an effective problem-specific classifier.

4 References

[1] I. McAtee and S. Wiederholt, "Project 0: Bayesian Classifier", University of Wyoming, 2021
[2] C. Wright, "Object and Pattern Recognition: Chapter 2 Parts 1-4", University of Wyoming, 2021
[3] R. O. Duda, P. E. Hart, and D. G. Stork, Pattern Classification, John Wiley & Sons, 2nd ed., 2001.

5 Appendix

Listing 1: Main Function for Project 1

```
%Project 1: Bayesian Discriminant and LDA Classifiers
%Authors: Ian McAtee and Stein Wiederholt
%Class: EE5650
%Professor: Dr. Wright
%Institution: University of Wyoming
%Date: 10/11/2021
%DESCRIPTION:
    %MATLAB program to perform the Bayesian Discriminant Function
    %classification and LDA classification of two classes of two
    %dimensional sample data

function McAtee_Wiederholt_proj01()
%% PRELIMINARY SETUP
%Load in the training and test data
load('test1.mat')
load('training1.mat')
axisOpt = [0,20,0,20];
```

```
%% PLOT DATA, STATISTICS, AND HISTOGRAMS
Title = 'Training Data';
genGaussStatisticsPlots(class1_train,class2_train,axisOpt,Title)
genHistograms(class1_train,class2_train,class1_test_a,
    [+]class2_test_a,class1_test_b,class2_test_b)

%% BAYESIAN CLASSIFIER
%Train Bayesian classifier
[W1,W2,w1,w2,w10,w20] = trainTwoClassBayesian(class1_train,
    [+]class2_train);

%Generate Plots of Trained Bayesian with Training data
genPlotsBayesian(class1_train,class2_train,W1,W2,w1,w2,w10,w20,
    [+]axisOpt,Title)

%Bayesian Classification of Test Data A and B
[classificationA_Bayes,errorsA_Bayes] = classifyTwoClassBayesian(
    [+]class1_test_a,class2_test_a,W1,W2,w1,w2,w10,w20);
[classificationB_Bayes,errorsB_Bayes] = classifyTwoClassBayesian(
    [+]class1_test_b,class2_test_b,W1,W2,w1,w2,w10,w20);

%Generate Plots of Trained Bayesian with Test data
Title = 'Test A Data';
genPlotsBayesian(class1_test_a,class2_test_a,W1,W2,w1,w2,w10,w20,
    [+]axisOpt,Title)
Title = 'Test B Data';
genPlotsBayesian(class1_test_b,class2_test_b,W1,W2,w1,w2,w10,w20,
    [+]axisOpt,Title)

%% LDA CLASSIFIER
%Train LDA Classifier
[w,decisionPoint,classOrder] = trainLDA(class1_train,class2_train)
    [+];

%Generate Plots of LDA with Training data
numSamps = 30;
genPlotsLDA(class1_train,class2_train,w,decisionPoint,axisOpt,
    [+]Title,numSamps)

%LDA Classification of Test Data A and B
[classificationA_LDA,errorsA_LDA] = classifyLDA(class1_test_a,
    [+]class2_test_a,w,decisionPoint,classOrder);
[classificationB_LDA,errorsB_LDA] = classifyLDA(class1_test_b,
    [+]class2_test_b,w,decisionPoint,classOrder);
```

```
%Generate Plots of LDA with Test Data
Title = 'Test Data A';
genPlotsLDA(class1_test_a,class2_test_a,w,decisionPoint,axisOpt,
    [+]Title,numSamps)
Title = 'Test Data B';
genPlotsLDA(class1_test_b,class2_test_b,w,decisionPoint,axisOpt,
    [+]Title,numSamps)

%% DISPLAY CLASSIFICATION RESULTS
%Bayesian Results
fprintf('BAYESIAN CLASSFICATION RESULTS\n')
fprintf('-----------------------------------\n')
fprintf('TEST DATA A:\n')
dispClassificationResults(errorsA_Bayes,2);
fprintf('\n')
fprintf('TEST DATA B:\n')
dispClassificationResults(errorsB_Bayes,2);

%LDA Results
fprintf('LDA CLASSFICATION RESULTS\n')
fprintf('-----------------------------------\n')
fprintf('TEST DATA A:\n')
dispClassificationResults(errorsA_LDA,2);
fprintf('\n')
fprintf('TEST DATA B:\n')
dispClassificationResults(errorsB_LDA,2);

end
```

Listing 2: Function to Train Bayesian Classifier

```
%FUNCTION: trainTwoClassBayesian
%AUTHOR: Stein Wiederholt
%DATE: 10/10/2021
%DESCRIPTION: Function to find the parameters for Gaussian
    [+]Discriminant
     %function classification
%INPUTS: (class1,class2)
     %class1: numSamples x 2 class 1 training data
     %class2: numSamples x 2 class 2 training data
%OUTPUTS: [W1,W2,w1,w2,w10,w20]
     %Equations 67-68 in Duda, Hart, and Stork's "Pattern
        [+]Classification"
     %Necessary terms to define Bayesian Discriminant function

function [W1,W2,w1,w2,w10,w20] = trainTwoClassBayesian(class1,
```

```
    [+]class2)

%find the size of the training data
lenClass1 = length(class1);
lenClass2 = length(class2);

%COMPUTE STATISTICS

%find the priors of the training data
pw1 = lenClass1 / (lenClass1 + lenClass2);
pw2 = lenClass2 / (lenClass2 + lenClass1);

%find the means of the training data
m1 = mean(class1).';
m2 = mean(class2).';

%find covariance matrix of training data
cov1 = cov(class1);
cov2 = cov(class2);

%Calculate necessary terms for class 1
detCov1 = det(cov1);  % determinant
invCov1 = inv(cov1);  % inverse

%calculate remaining class 2 variables
detCov2 = det (cov2);  % determinant
invCov2 = inv(cov2);  % inverse

% compute Bayesian discriminant function components using
    [+]equations 66, 67, 68, and 69 from text
%eq 67
W1 = -0.5 * invCov1;
W2 = -0.5 * invCov2;
%eq 68
w1 = invCov1 * m1;
w2 = invCov2 * m2;
%eq 69
w10 = -0.5 * m1' * invCov1 * m1 - 0.5 * log(detCov1) + log(pw1);
w20 = -0.5 * m2' * invCov2 * m2 - 0.5 * log(detCov2) + log(pw2);

end
```

Listing 3: Function to Train LDA Classifier

```
%FUNCTION: trainLDA
%AUTHOR: Ian McAtee
```

```
%DATE: 10/2/2021
%DESCRIPTION: Function to find the Fischer vector and decision [+]points based
    %on the two input class training data sets
%INPUTS: (class1_train,class2_train)
    %class1: numSamples x numFeatures class 1 training data
    %class2: numSamples x numFeatures class 2 training data
%OUTPUTS: [w,decisionPoint,classOrder]
    %w: Fischer Vector
    %descisionPoint: scalar value of decision point along Fischer [+]vector
    %classOrder: Integer value reflecting which class mean is [+]higher than
        %the decision point

function [w,decisionPoint,classOrder] = trainLDA(class1,class2)

%Find the number of training samples for each class
N1 = length(class1);
N2 = length(class2);

%Get training data means
m1 = mean(class1).';
m2 = mean(class2).';

%Get training data covariances
covW1 = cov(class1);
covW2 = cov(class2);

%Compute Scatter Matrices
S1 = (N1-1)*covW1;
S2 = (N2-1)*covW2;
Sw = S1 + S2;

%Find Fischer's Vector
w = (inv(Sw))*(m1-m2);

%Make w a unit vector
w = w./(sqrt((0-w(1))^2+(0-w(2))^2));

%Constrain w to first quadrant
w = abs(w);

%Project the training class means into the 1D space
m1Proj = w.'*m1;
```

```
m2Proj = w.'*m2;

%Decision Point is midway between the means in the 1D space
decisionPoint = (m1Proj + m2Proj)/2;

%Find which class is above decision point and which is below
if (m1Proj > m2Proj)
    classOrder = 1;
else
    classOrder = 2;
end

end
```

Listing 4: Bayesian Classification Function

```
%FUNCTION: classifyTwoClassBayesian
%AUTHOR: Ian McAtee and Stein Wiederholt
%DATE: 10/10/2021
%DESCRIPTION: Function to find the classify test data based on
   [+]trained
    %Bayesian discriminant function parameters
%INPUTS: (class1_test,class2_test,w,decisionPoint,classOrder)
    %class1: numSamples x 2 class 1 test data
    %class2: numSamples x 2 class 2 training data
    %W1,W2,w1,w2,w10,w20: Discriminant function terms
%OUTPUTS: [classification,errors]
    %classification: (numSampsClass1+numSampsClass2)x1 vector
       [+]containing
        %the classification label (1 or 2) for each sample of test
           [+] data
    %errors: (numClasses+1)x1 vector containing the percent error
       [+]of the
        %ith class in the ith row with the last row containing the
        %overall percent error

function [classification,errors] = classifyTwoClassBayesian(class1
   [+],class2,W1,W2,w1,w2,w10,w20)

%Find the number of test samples for each class
N1 = length(class1);
N2 = length(class2);

%Create a labels matrix
labels = [ones(N1,1);2*ones(N2,1)];

```

```
testData = [class1;class2];

%Preallocate a classification vector
classification = zeros((N1+N2),1);

%Perform the classification using the discriminant function
for i = 1:(N1+N2)
    x = [testData(i,1);testData(i,2)];
    g1 = x.' * W1 * x + w1.' * x + w10;
    g2 = x.' * W2 * x + w2.' * x + w20;
    g = g1 - g2;
    if (g >= 0)
        classification(i,1) = 1;
    else
        classification(i,1) = 2;
    end
end

%Find classification errors
errorTotal = (sum((labels - classification).^2)/(N1+N2))*100;
errorClass1 = (sum((labels(1:N1,:) - classification(1:N1)).^2)/(N1
   [+]))*100;
errorClass2 = (sum((labels(N1+1:N1+N2,:) - classification(N1+1:N1+
   [+]N2,:)).^2)/(N2))*100;

errors = [errorClass1;errorClass2;errorTotal];

end
```

Listing 5: LDA Classification Function

```
%FUNCTION: classifyLDA
%AUTHOR: Ian McAtee
%DATE: 10/2/2021
%DESCRIPTION: Function to find the classify test data based on
   [+]trained LDA
    %parameters
%INPUTS: (class1_test,class2_test,w,decisionPoint,classOrder)
    %class1: numSamples x numFeatures class 1 test data
    %class2: numSamples x numFeatures class 2 training data
    %w: Fischer Vector
    %descisionPoint: scalar value of decision point along Fischer
       [+]vector
    %classOrder: Integer value reflecting which class mean is
       [+]higher than
        %the decision point
```

```
%OUTPUTS: [classification,errors]
    %classification: (numSampsClass1+numSampsClass2)x1 vector
       [+]containing
        %the classification label (1 or 2) for each sample of test
           [+] data
    %errors: (numClasses+1)x1 vector containing the percent error
       [+]of the
        %ith class in the ith row with the last row containing the
        %overall percent error

function [classification,errors] = classifyLDA(class1,class2,w,
   [+]decisionPoint,classOrder)

%Find the number of test samples for each class
N1 = length(class1);
N2 = length(class2);

%Form label vector for test data
labels = [ones(N1,1);2*ones(N2,1)];

%Project test data A into 1D space
yTest1 = (w.'*class1.').'; %Eq. from Chap3Part5 Slide 23
yTest2 = (w.'*class2.').';
yTest = [yTest1;yTest2];

%Classify test data
classification = yTest;
if (classOrder == 1)
    classification(yTest>=decisionPoint) = 1;
    classification(yTest<decisionPoint) = 2;
elseif (classOrder == 2)
    classification(yTest>=decisionPoint) = 2;
    classification(yTest<decisionPoint) = 1;
else
    error('Classification order must be set to 1 or 2. See help.')
end

%Find classification errors
errorTotal = (sum((labels - classification).^2)/(N1+N2))*100;
errorClass1 = (sum((labels(1:N1,:) - classification(1:N1)).^2)/(N1
   [+]))*100;
errorClass2 = (sum((labels(N1+1:N1+N2,:) - classification(N1+1:N1+
   [+]N2,:)).^2)/(N2))*100;

errors = [errorClass1;errorClass2;errorTotal];
```

```
end
```

Listing 6: Function to Display Classification Results

```
%FUNCTION: dispClassificationResults
%AUTHOR: Ian McAtee
%DATE: 10/2/2021
%DESCRIPTION: Function to display classification accuracies and errors for
    %each class
%INPUTS: (errors,numClasses)
    %numClasses: The number of classes in the classification
    %errors: (numClasses+1)x1 vector containing the percent error of the
            %ith class in the ith row with the last row containing the
            %overall error
%OUTPUT: Tabular display of classification erros and accuracies

function dispClassificationResults(errors,numClasses)

%Compute Classifier Accuracy Metrics
accuracy = 100*ones(1,numClasses+1) - errors;

fprintf('          \t ACCURACY: \t ERROR:\n')
for i = 1:numClasses
    fprintf('Class %u: \t %.2f%% \t %.2f%%\n',i,accuracy(i),errors(i));
end
fprintf('Overall: \t %.2f%% \t %.2f%%\n',accuracy(numClasses+1),errors(numClasses+1));

end
```

Listing 7: Function to Generate Plots of Data and Gaussian PDFs

```
%FUNCTION: genGaussStatisticsPlots
%AUTHOR: Ian McAtee and Stein Wiederholt
%DATE: 10/10/2021
%DESCRIPTION: Function to plot two class training data and the associated
    %bivariate and marginal gaussian distributions
%INPUTS: (class1,class2,axisOpt,Title)
    %class1: numSamples x 2 class 1 data
    %class2: numSamples x 2 class 2 data
    %axisOpt: row vector specifying the x and y axis for plotting
```

```
10      %Title: title for the particular data set being used
    %OUTPUT: Scatter plot of data, marginal gaussian distributions of
        [+]x_1
12      %and x_2, contour line plot of gaussian PDF, 3D plot of
            [+]gaussian PDF

14  function genGaussStatisticsPlots(class1,class2,axisOpt,Title)

16  x_ticks = [0:2:20];

18  %Get Statistics
    numFeat1 = size(class1,2);
20  numFeat2 = size(class2,2);

22  %Calculate means and covariences
    m1 = mean(class1).';
24  m2 = mean(class2).';
    cov1 = cov(class1);
26  cov2 = cov(class2);

28  %Pre-calculate univariate gaussian components
    sqrtTwoPiVar1_x1 = sqrt(2*pi*cov1(1,1)); %sqrt(2*pi*var)
30  sqrtTwoPiVar1_x2 = sqrt(2*pi*cov1(2,2));
    sqrtTwoPiVar2_x1 = sqrt(2*pi*cov2(1,1));
32  sqrtTwoPiVar2_x2 = sqrt(2*pi*cov2(2,2));
    oneHalf_var1_x1 = -0.5/cov1(1,1); %-1/(2var)
34  oneHalf_var1_x2 = -0.5/cov1(2,2);
    oneHalf_var2_x1 = -0.5/cov2(1,1);
36  oneHalf_var2_x2 = -0.5/cov2(2,2);

38  %Pre-calculate multivariate gaussian components
    twoPiD21 = (2*pi)^(numFeat1/2); %Class 1: 2pi^(d/2)
40  twoPiD22 = (2*pi)^(numFeat2/2); %Class 2: 2pi^(d/2)
    sqrtDetCov1 = sqrt(det(cov1)); %Class 1: |Sigma|^(1/2)
42  sqrtDetCov2 = sqrt(det(cov2)); %Class 2: |Sigma|^(1/2)
    invCov1 = inv(cov1); %Class 1: (Sigma)^(1/2)
44  invCov2 = inv(cov2); %Clsas 2: (Sigma)^(1/2)

46  %Create temp x data to evaluate over
    x = (-20:0.1:20); %Reduce the step size to for higher res. plots
48  L = length(x); %Get number of temp samples

50  %Preallocate matrices to hold probability
    %Multivariate distributions
52  pX1 = zeros(L,L);
```

```
pX2 = zeros(L,L);
%Marginal distributions
pX1_x1 = zeros(1,L);
pX1_x2 = zeros(1,L);
pX2_x1 = zeros(1,L);
pX2_x2 = zeros(1,L);

%Evaluate the Gaussian PDFs for training data of each class
for i=1:L  % loop columns
    %Marginal distributions of each dimension for each class
    pX1_x1(i) = (1/sqrtTwoPiVar1_x1)*exp(oneHalf_var1_x1*(x(i)-m1
       [+](1))^2);
    pX1_x2(i) = (1/sqrtTwoPiVar1_x2)*exp(oneHalf_var1_x2*(x(i)-m1
       [+](2))^2);
    pX2_x1(i) = (1/sqrtTwoPiVar2_x1)*exp(oneHalf_var2_x1*(x(i)-m2
       [+](1))^2);
    pX2_x2(i) = (1/sqrtTwoPiVar2_x2)*exp(oneHalf_var2_x2*(x(i)-m2
       [+](2))^2);
    for j=1:L  % loop rows
        x_vector = [x(i); x(j)]; %Form vector
        %Calculate Gaussian eq. terms
        xm1 = x_vector - m1;
        xm2 = x_vector - m2;
        arg1 = xm1.'*invCov1*xm1;
        arg2 = xm2.'*invCov2*xm2;
        %Calculate Multivariate Gaussian
        pX1(j,i)=(1/(twoPiD21*sqrtDetCov1))*exp(-0.5*arg1);
        pX2(j,i)=(1/(twoPiD22*sqrtDetCov2))*exp(-0.5*arg2);
    end
end

%Scatter plot of data
figure()
scatter(class1(:,1),class1(:,2),'b.')
hold on
scatter(class2(:,1),class2(:,2),'r.')
hold off
axis equal
axis(axisOpt)
xticks(x_ticks)
box on
xlabel('x_1')
ylabel('x_2')
legend('Class 1','Class 2','Location','southeast')
title(Title)
```

```
94
%Plot Marginal PDF of x1 Dimension for Both Classes
96 figure()
plot(x,pX1_x1,'b')
98 hold on
plot(x,pX2_x1,'r')
100 hold off
xlim([0,20])
102 box on
xlabel('x_1')
104 ylabel('p(x_1)')
legend('p(x_1|\omega_1)','p(x_1|\omega_2)','Location','northeast')
106 title(strcat(Title,': Marginal PDFs of x_1 For Class 1 and Class 2
    [+]'))

108 %Plot Marginal PDF of x2 Dimension for Both Classes
figure()
110 plot(x,pX1_x2,'b')
hold on
112 plot(x,pX2_x2,'r')
hold off
114 xlim([0,20])
box on
116 xlabel('x_2')
ylabel('p(x_2)')
118 legend('p(x_2|\omega_1)','p(x_2|\omega_2)','Location','northeast')
title(strcat(Title,': Marginal PDFs of x_2 For Class 1 and Class 2
    [+]'))
120

%Plot Contour Lines of Multivariate Gaussian of Both Classes
122 figure()
contour(x,x,pX1,'b')
124 hold on
contour(x,x,pX2,'r')
126 axis equal
axis(axisOpt)
128 xticks(x_ticks)
box on
130 hold off
xlabel('x_1')
132 ylabel('x_2')
legend('p(x|\omega_1)','p(x|\omega_2)','Location','southeast')
134 title(strcat(Title, ': PDFs as Equiprobable Lines'))

136 %Plot 3D PDFs of both classes
```

```
figure()
mesh(x,x,pX1)
hold on
mesh(x,x,pX2)
axis('tight')
xlim([0 20])
ylim([0 20])
hold off
xlabel('x_1')
ylabel('x_2')
zlabel('p(x|\omega)')
hold off
view(9,18)
title(strcat(Title,': PDFs of Class 1 and Class 2'))

end
```

Listing 8: Function to Generate Histograms of Data

```
%FUNCTION: genHistograms
%AUTHOR: Stein Wiederholt
%DATE: 10/11/2021
%DESCRIPTION: Function to plot the histograms of training and test
   [+] data
%INPUTS: (class1_train,class2_train,class1_test_a,class2_test_a,
        %class1_test_b,class2_test_b)
%OUTPUT: Subplots of the histograms of the training and test data

function genHistograms(class1_train,class2_train,class1_test_a,
   [+]class2_test_a,class1_test_b,class2_test_b)

%set parameters for near-full-screen figures.
%Credit BE_demo, Copyright (c) 2008-2012 Cameron H. G. Wright
scrsz = get(0,'ScreenSize'); % get the screen size
% Vista and Win7 have a 31 pixel task bar at bottom
Los=10; % left offset
Bos=55; % bottom offset
Ros=20; % right offset
Tos=145; % top ofset
FigPos=[scrsz(1)+Los scrsz(2)+Bos scrsz(3)-Ros scrsz(4)-Tos];

set(0,'DefaultAxesFontSize',14);
set(0,'DefaultTextFontSize',14);
set(0,'DefaultLineLineWidth',2);

nBins = 100;
```

```
%class 1 histograms
figure(1)
set(gcf,'Position',FigPos)  % set near max figure size
subplot(2,3,1)
histogram(class1_train(:,1),nBins)
xlabel('Value')
ylabel('Frequency')
title('Class 1 Train: x_1')
subplot(2,3,4)
histogram(class1_train(:,2),nBins)
xlabel('Value')
ylabel('Frequency')
title('Class 1 Train: x_2')
subplot(2,3,2)
histogram(class1_test_a(:,1),nBins)
xlabel('Value')
ylabel('Frequency')
title('Class 1 Test A: x_1')
subplot(2,3,5)
histogram(class1_test_a(:,2),nBins)
xlabel('Value')
ylabel('Frequency')
title('Class 1 Test A: x_2')
subplot(2,3,3)
histogram(class1_test_b(:,1),nBins)
xlabel('Value')
ylabel('Frequency')
title('Class 1 Test B: x_1')
subplot(2,3,6)
histogram(class1_test_b(:,2),nBins)
%axis('tight')
xlabel('Value')
ylabel('Frequency')
title('Class 1 Test B: x_2')

%class 2 histograms
figure(2)
set(gcf,'Position',FigPos)  % set near max figure size
subplot(2,3,1)
histogram(class2_train(:,1),nBins)
xlabel('Value')
ylabel('Frequency')
title('Class 2 Train: x_1')
subplot(2,3,4)
histogram(class2_train(:,2),nBins)
```

```
xlabel('Value')
ylabel('Frequency')
title('Class 2 Train: x_2')
subplot(2,3,2)
histogram(class2_test_a(:,1),nBins)
xlabel('Value')
ylabel('Frequency')
title('Class 2 Test A: x_1')
subplot(2,3,5)
histogram(class2_test_a(:,2),nBins)
xlabel('Value')
ylabel('Frequency')
title('Class 2 Test A: x_2')
subplot(2,3,3)
histogram(class2_test_b(:,1),nBins)
xlabel('Value')
ylabel('Frequency')
title('Class 2 Test B: x_1')
subplot(2,3,6)
histogram(class2_test_b(:,2),nBins)
xlabel('Value')
ylabel('Frequency')
title('Class 2 Test B: x_2')

%reset default figure properties
set(0,'DefaultAxesFontSize','remove');
set(0,'DefaultTextFontSize','remove');
set(0,'DefaultLineLineWidth','remove')

end
```

Listing 9: Function to Generate Plots for Bayesian Classifier

```
%FUNCTION: genPlotsBayesian
%AUTHOR: Ian McAtee and Stein Wiederholt
%DATE: 10/10/2021
%DESCRIPTION: Function to plot data on Bayesian discriminant
   [+]region plot
%INPUTS: (class1,class2,axisOpt,Title)
    %class1: numSamples x 2 class 1 data
    %class2: numSamples x 2 class 2 data
    %W1,W2,w1,w2,w10,w20: Discriminant function terms
    %axisOpt: row vector specifying the x and y axis for plotting
    %Title: title for the particular data set being used
%OUTPUT: Scatter plot of data superimposed on discriminant
   [+]function
```

```
12      %classification regions

14  function genPlotsBayesian(class1,class2,W1,W2,w1,w2,w10,w20,
        [+]axisOpt,Title)

16  %Set region to evaluate discriminant funtion over
    x = (-20:0.05:20);
18  L = length(x);

20  %Preallocate discriminant functions
    g1 = ones(L,L);
22  g2 = ones(L,L);

24  for i=1:L
        for j=1:L
26          xx = [x(i);x(j)];
            g1(j,i) = xx' * W1 * xx + w1' * xx + w10;
28          g2(j,i) = xx' * W2 * xx + w2' * xx + w20;
        end
30  end

32  %dichotimize function eq 29
    g = g1 - g2;
34
    %find decision regions for plot
36  gRegion = ones(L,L);
    index = find(g < 0);
38  gRegion(index) = 0;

40  %Plot the decision boundary with training data superimposed
    figure()
42  imagesc(x,x,gRegion,'AlphaData',1.0)
    axis('xy')
44  colormap('hot')
    hold on
46  scatter(class1(:,1),class1(:,2),'b.');
    hold on
48  scatter(class2(:,1),class2(:,2),'r.');
    axis equal
50  axis(axisOpt)
    box on
52  hold off
    xlabel('x_1')
54  ylabel('x_2')
    text(10,18,'Class 2','color','w')
```

```
text(1,2.5,'Class 1','color','k')
legend('Class 1 (\omega_1)','Class 2 (\omega_2)','Location','
    [+]southeast')
title(strcat(Title,' with Bayesian Decision Regions'))

end
```

Listing 10: Function to Generate Plots for LDA Classifier

```
%FUNCTION: genPlotsLDA
%AUTHOR: Ian McAtee
%DATE: 10/2/2021
%DESCRIPTION: Function to plot a variety of parameters based on a
    [+]trained
     %LDA classifier and two input data sets
%INPUTS: (class1_test,class2_test,w,decisionPoint,axisOpt,Title,
    [+]numSamps)
     %class1_test: numSamples x numFeatures class 1 test data
     %class2_test: numSamples x numFeatures class 2 training data
     %w: Fischer Vector
     %descisionPoint: scalar value of decision point along Fischer
        [+]vector
     %axisOpt: row vector specifying the x and y axis for plotting
     %Title: title for the particular data set being used
     %numSamps: number of samples to be plotted for certain plots
%OUTPUT: Plots of the various parameters of the LDA classifier to
        %illustrate operation

function genPlotsLDA(class1,class2,w,decisionPoint,axisOpt,Title,
    [+]numSamps)

%Set preliminary variables
lWidth = 2.5;
x_ticks = [0:2:20];
ww = [0,0;w(1),w(2)];
ww = ww*100; %off the plot

%Plot the Training Data with the Fisher Vector
figure()
scatter(class1(:,1),class1(:,2),'b.')
hold on
scatter(class2(:,1),class2(:,2),'r.')
hold on
plot(ww(:,1),ww(:,2),'g','LineWidth',lWidth);
hold off
axis equal
```

```
axis(axisOpt)
xticks(x_ticks)
box on
xlabel('x_1')
ylabel('x_2')
legend('Class 1','Class 2','Fisher Vector','Location','southeast')
title(strcat(Title,' with Fisher Vector'))

%Project test data A into 1D space
y1 = (w.'*class1.').'; %Eq. from Chap3Part5 Slide 23
y2 = (w.'*class2.').';

%Find angle of fisher vector
angle = atan(w(2)/w(1));

%Find x_1, x_2 coordinates of projections for plottng
class1Proj2D = [cos(angle).*y1,sin(angle).*y1];
class2Proj2D = [cos(angle).*y2,sin(angle).*y2];

%Plot the Training Data Projected onto Fischer Vector
figure()
plot(ww(:,1),ww(:,2),'g','LineWidth',lWidth);
hold on
scatter(class1Proj2D(1:numSamps,1),class1Proj2D(1:numSamps,2),'b*'
    [+])
hold on
scatter(class2Proj2D(1:numSamps,1),class2Proj2D(1:numSamps,2),'r*'
    [+])
hold on
scatter(class1(1:numSamps,1),class1(1:numSamps,2),'b.')
hold on
scatter(class2(1:numSamps,1),class2(1:numSamps,2),'r.')
hold off
axis equal
axis(axisOpt)
xticks(x_ticks)
box on
xlabel('x_1')
ylabel('x_2')
legend('Fisher Vector','Projected Class 1','Projected Class 2','
    [+]Class 1','Class 2','Location','southeast')
title(strcat(Title, ' Projected onto Fisher Vector (',num2str(2*
    [+]numSamps),' Samples)'))

%Find the decision line that is orthogonal to decision point
```

```
decisionLine = [decisionPoint*cos(angle)+((decisionPoint*sin(angle
    [+]))/(tan(pi/2-angle))),0;
                0,decisionPoint/(cos(pi/2-angle))];

%Scatter plot of data with fisher vector, decision point, and line
figure()
scatter(class1(:,1),class1(:,2),'b.')
hold on
scatter(class2(:,1),class2(:,2),'r.')
hold on
plot(ww(:,1),ww(:,2),'g','LineWidth',lWidth);
hold on
plot(decisionLine(:,1),decisionLine(:,2),'m','LineWidth',lWidth)
hold on
scatter(decisionPoint*cos(angle),decisionPoint*sin(angle),1000,'k.
    [+]')
hold off
axis equal
axis(axisOpt)
xticks(x_ticks)
box on
xlabel('x_1')
ylabel('x_2')
legend('Class 1','Class 2','Fisher Vector','Decision Line','
    [+]Decision Point','Location','southeast')
title(strcat(Title,' with Fisher Vector, Decision Line, and
    [+]Decision Point'))

%Plot the data projected onto fisher vector with decision point
figure()
plot(ww(:,1),ww(:,2),'g','LineWidth',lWidth);
hold on
scatter(class1Proj2D(:,1),class1Proj2D(:,2),'b*')
hold on
scatter(class2Proj2D(:,1),class2Proj2D(:,2),'r*')
hold on
scatter(decisionPoint*cos(angle),decisionPoint*sin(angle),1000,'k.
    [+]')
hold off
axis equal
axis(axisOpt)
xticks(x_ticks)
box on
xlabel('x_1')
ylabel('x_2')
```

```
legend('Fisher Vector','Projected Class 1','Projected Class 1','
    [+]Decision Point','Location','southeast')
title(strcat(Title, ' Projected onto Fisher Vector with Decision
    [+]Point'))

%Plot numSamps of data projected on fisher vector with decision
    [+]point/line
figure()
plot(ww(:,1),ww(:,2),'g','LineWidth',lWidth);
hold on
plot(decisionLine(:,1),decisionLine(:,2),'m','LineWidth',lWidth)
hold on
scatter(class1Proj2D(1:numSamps,1),class1Proj2D(1:numSamps,2),'b*'
    [+])
hold on
scatter(class2Proj2D(1:numSamps,1),class2Proj2D(1:numSamps,2),'r*'
    [+])
hold on
scatter(class1(1:numSamps,1),class1(1:numSamps,2),'b.')
hold on
scatter(class2(1:numSamps,1),class2(1:numSamps,2),'r.')
hold on
scatter(decisionPoint*cos(angle),decisionPoint*sin(angle),1000,'k.
    [+]')
hold off
axis equal
axis(axisOpt)
xticks(x_ticks)
xlabel('x_1')
ylabel('x_2')
legend('Fisher Vector','Decision Line','Projected Class 1','
    [+]Projected Class 2','Class 1','Class 2','Decision Point','
    [+]Location','southeast')
title(strcat(Title, ' Projected onto Fisher Vector with Decision
    [+]Line and Point (',num2str(2*numSamps),' Samples)'))

end
```