AMATH 582 Homework 4: Facial Principle Components and Music Classification using Supervised Learning Techniques

Gavin M. Chase

March 6, 2020

Abstract

This report primarily explores the SVD of two data sets: Yale Faces B and a selection of audio recordings from various music genres and artists. In each case, low rank approximations of the data are analyzed to determine the principle components. The music data is further classified as sets of 3 bands or 3 artists using a combination of supervised learning techniques to include Linear Discriminant Analysis, Naive Bayes, and Classification and Regression Trees.

1 Introduction and Overview

Machine learning has become an increasingly powerful tool for classification and recognition of new data. Several tools exist to perform these tasks, each with their own unique algorithmic approach. Collectively, they seek to take advantage of distinct features of subsets of data and either cluster data with similar features together or predict which cluster, or class, a new piece of data is most closely associated with.

Two classic problems for machine learning are facial recognition and music identification. Due to the complexity of the topic, this report only seeks to explore features that could be used in classification of faces from the Yale Faces B Database, but does not actually attempt to classify them. I do, however, employ three supervised learning algorithms in three trials on musical recordings to predict either the artist or the genre.

Trial 1 includes recordings from 3 bands each representing a different genre. Specifically, it includes music from AC/DC (hard rock), John Williams (classical), and the US Army Band (Songs of the Soldier, which are musical pieces officially recognized by the US Army. They include the division song of 2nd Infantry Division stationed at Ft. Lewis, WA and the *Alma Mater* of my alma mater, the United States Military Academy). Trial 2 includes recordings from 3 bands all from the same genre: 2000s Contemporary Christian Music (CCM). They include Third Day, Newsboys, and Barlow Girl. Trial 3 includes music from multiple artists that collectively represent a broad genre. The genres are CCM, classical, and Songs of the Soldier.

2 Theoretical Background

As will be discussed in the Algorithm Implementation section below, the preparation of both data sets for PCA and classification employs a number of techniques discussed in previous homework reports, which will not be discussed here. Specifically, I employ FFT[2], Gábor Transforms[3], Spectrograms[3], Step-functions[3], and Singular Value Decomposition[4]. The first new mathematical tool employed as part of this homework is the Haar wavelet[6][7]. This wavelet is exceptional at edge detection due to the sharp edges formed by its piecewise nature, and is very similar to employing the Step-function.

$$\psi(t) = \begin{cases} 1 & 0 \leq t < 1.2 \\ -1 & 1/2 \leq t < 1 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The remaining new techniques found in this report fall under statistical learning, known more popularly as machine learning. Statistical learning takes advantage of patterns in data to predict simple ways to cluster

or classify data into subsets. There are two dominant approaches to machine learning: unsupervised learning and supervised learning. Unsupervised learning involves statistically predicting if data can be subdivided into smaller groups known as clusters. This is often performed when little is known about the data being analyzed. Supervised learning, however, takes advantage of known information about the data to teach an algorithm how to recognize members of a set number of classes, and predict which class new data belongs to. I focused exclusively on supervised learning for music classification. Specifically, I employed Linear Discriminant Analysis (LDA), Naive Bayes, and Classification And Regression Trees (CART).

LDA seeks to construct projections onto which the variance amongst the intra-class data is minimized and the distance between the class means μ is maximized where $\mathbf{S}_B$ is the between class scatter matrix and $\mathbf{S}_W$ is the within class matrix [9][7][5].

$$\mathbf{S}_B = (\mu_n - \mu)(\mu_n - \mu)^{\mathrm{T}} \tag{2}$$

$$\mathbf{S}_W = \sum_{n=1}^{i} \sum_{\mathbf{x}} (\mathbf{x} - \mu_n)(\mathbf{x} - \mu_n)^{\mathrm{T}} \tag{3}$$

In Equations 2 and 3 μ is the mean of the entire data set, μ_n is the mean of the n^{th} class of data.

Naive Bayes classification takes advantage of Bayes' Rule

$$P(C = c_k|\mathbf{X} = \mathbf{x}) = P(C = c_k) \times \frac{P(\mathbf{X} = \mathbf{x}|C = c_k)}{P(\mathbf{x})} \tag{4}$$

where

$$P(\mathbf{X} = \mathbf{x}) = \sum_{k'=1}^{e_C} P(\mathbf{X} = \mathbf{x}|C = c_{k'}) \times P(C = c_{k'}) \tag{5}$$

and e_C is a given a set of classes C and X is a set of feature values. The key to minimizing the number of errors during classification is to assume that the features are statistically independent of each other. This allows users to model $P(c_k|\mathbf{x})$ with few inputs[8]. The derivation and explanation of this assumption can be found elsewhere[8], but it results in the following

$$\widehat{P(c_k}|\mathbf{x}) = \frac{\widehat{P(c_k)} \times \prod_{j=1}^{d} P(\widehat{x_j}|c_k)}{\widehat{P(\mathbf{x})}} \tag{6}$$

Classification and Regression Trees (CART) employ a series of binary choices at the conclusion of which a decision or "bin" is reached much in the same way a controlled game of plinko would result in a ball landing in a bin after bouncing off of several pins. Algorithmically, the goal is to select the moments of choice where the probability of a misclassification is lowest and establish the bins or terminal nodes at points where it no longer becomes possible to significantly reduces misclassifications[1].

3 Algorithm Implementation and Development

The objective is to train an algorithm to distinguish between classes of information and predict the class of new information based on its unique features. There are several key steps to this process:

1. Compile data and enforce uniformity in its size and structure. For images, this simply means lowering the resolution of all images to 64×64 pixels. For music, this involved resampling the music by removing every other point and applying a windowed step-function much like Algorithm 4 from the HW2 report to produce a spectrogram[3]. I then treat the spectrogram as an image and similarly rescale it.

2. Perform edge detection by decomposing the images into wavelet basis functions. In this report, the Haar wavelet is used. This is shown in Listing 3.

3. Use the wavelet expanded images to compute the SVD.

4. Based on analysis of the principal components found by the SVD, select features to be used for discrimination between classes.

5. Construct and cross-validate discrimination algorithms.

The data for this report's experiments are organized differently than for the paint can experiment in HW3. Instead of each row of $\mathbf{A}$ corresponding to a set of measurements x or y, $\mathbf{A}$ is constructed such that each column contains the reshaped pixel values for one wavelet expanded image. Thus, $\mathbf{A}$ becomes size $1024 \times n$ where n is the number of sample images used. This orientation is important because it means that in this case, $\mathbf{U}$ holds the features or principal components and $\mathbf{V}$ describes how much of $\mathbf{U}$ is needed to reproduce each element of $\mathbf{A}$. Its also worth noting that the classification portions of the list above only apply to the music classification trials. I do not seek to predict faces from the Yale Faces database. I simply analyze their principal components. For the music however, the following Algorithm is applied.

Algorithm 1: Training and Cross Validation

for $j = 1$: number of desired trials **do**
  Create a vector q of integers from 1 to n samples for each class and randomly shuffle it.
  Create a matrix for each class containing the number of elements of $\mathbf{V}$ to be included in training
  Assign i training components where $i < n$ and are the first i positions in q
  Assign the remaining members of each class's data as the testing components
  Create a vector of labels equal in size to the training set
  Train LDA, Naive Bayes, and CART prediction models
  Collect predictions of test data classes as columns
end for
Find the accuracy for each class within each model

4 Computational Results

4.1 Yale Faces

The Yale Faces database contains a set of cropped images and a set of uncropped images. The cropped set includes up to 64 images of 38 subjects taken with various lighting. There are 18 corrupted images spread throughout 7 subjects that were not included in this study. The uncropped set includes 165 images of 15 subjects with various lighting, facial expressions, and occasionally include glasses. Only 1 image, Subject 4's "sad" image was corrupted and replaced by Subject 4's "Normal" image.

Because the cropped images eliminate any influence from head shape or size, hair styles, or silhouettes, the SVD reveals principal components focused on the actual structure of a person's face and how the contours of that structure impact shadows and reflection. This is shown very clearly in Figure 1 where of the 12 modes shown, all but modes 2, 7, 9 and 12 characterize the eyes, nose, or mouth. Interestingly, shadows play only a marginally subordinate role, which is emphasized by modes 2 and 7.

The impact of $\mathbf{U}$ modes on each image are shown in Figure 2. Because each of the subjects was photographed with the same sequence of lighting options across the 64 shots, the lines for each mode of $\mathbf{V}$ are similar across the subjects. The finer differences between the plots are what would be needed to begin distinguishing between the subjects.

Finally, while the first three singular values do stand out in Figure 3a, they only account for 7.9% of the total value of $\mathbf{\Sigma}$. The first 25% (256) of the singular values account for 60.9% of $\mathbf{\Sigma}$, and the first 789 elements of $\mathbf{\Sigma}$ are needed to account for 95% of $\mathbf{\Sigma}$.

The addition of more information in the uncropped images very obviously diminished the value of facial structure in analyzing a subject's face. Because of the edge detection during pre-processing, the most important feature became the outline or silhouette of a person's head. The starkness of this feature was only enhanced by the choice of a white background. Thus, the first three modes of $\mathbf{U}$, shown in Figure 4 focus almost exclusively on isolating a subject's silhouette and whether or not there is a shadow on either

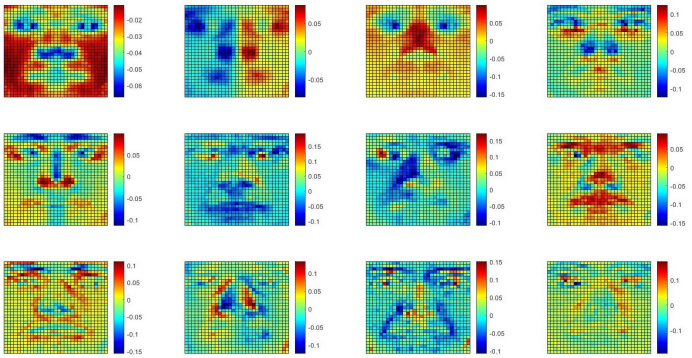


Figure 1: The first 12 modes $\mathbf{U}$ of the cropped faces. Mode 2 highlights the significance of shadows.

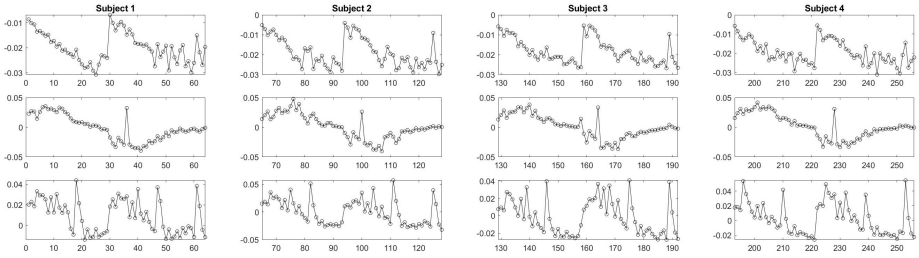


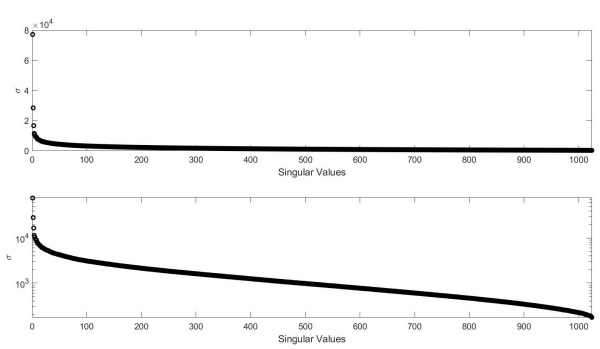
Figure 2: The values of the first 3 modes of $\mathbf{V}$ for the first 4 cropped face subjects. Note the parabolic-like curve for mode 2 across all subjects due to the rotation of the light source around the subjects' faces throughout the trials.

side of their head. Mode 4 introduces another interesting feature: long hair. This subtly gets reinforced throughout the subsequent modes. The final interesting observation about $\mathbf{U}$ is the fact that reflective or "high contour" points of facial structure makes their first appearance in mode 6 where the brow, nose, and chin appear. Unlike the cropped images, the progressions of $\mathbf{V}$ across the images for each uncropped subject are very distinct from each other. This is most likely due to how much more obvious distinctions like long hair v. baldness impact the role $\mathbf{U}$ plays in reconstructing each subject. Finally, analysis of the singular values shows that their decay when more information is available is much more gradual. Only the last 9 elements of $\mathbf{\Sigma}$ are near zero as shown in Figure 3b. The first 2 values account for 11.9% and the first 25% account for 57.1% of $\mathbf{\Sigma}$. 128 of the 165 elements of $\mathbf{\Sigma}$ are needed to achieve 95% reconstruction.

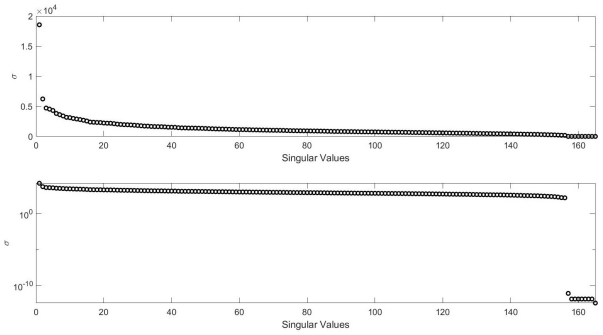
4.2 Music Classification

The accuracy scores for LDA, Naive Bayes, and CART during Trial 1 using the first 10 modes of $\mathbf{V}$ are shown in Table 1. After running 1000 iterations, the LDA algorithm proved most capable of correctly predicting bands within this trial. The strong performance of the US Army Band can be explained by the separation at mode 2 of $\mathbf{V}$ shown in Figure 8. The division songs selected for this trial are dominated by vocal content which is captured in modes 1 and 2 of $\mathbf{U}$. AC/DC and John Williams each perform without vocals during a number of samples, which leads them to be misclassified as each other as shown in Figure 7.

Unsurprisingly, the classification accuracy for Trial 2 was lower than it was for Trial 1. The accuracy ranges can be found in Table 2 for all three supervised learning techniques using the first 11 modes of $\mathbf{V}$. LDA continued to be the most reliable algorithm, but all struggled to classify Third Day correctly. In fact, CART misclassified Third Day samples as Barlow Girl more frequently than it correctly predicted they were performed by Third Day. LDA and Naive Bayes struggled in the same way, but only when the number of features was less than 6 and greater than 12. The poor performance in identifying Third Day samples most likely stems from the fact that Third Day represents a very "average" sound for its genre. They play no unique instruments, and have a male vocalist. Conversely, Barlow Girl employs female vocals, which are



(a) $\mathbf{\Sigma}$ values for the cropped face data.



(b) $\mathbf{\Sigma}$ values for the uncropped face data. Note the sharp drop off after the 157^{th} point.

Figure 3: $\mathbf{\Sigma}$ values for the Yale Face sets. The top panel depicts the singular values, and the bottom panel contains the same singular values plotted on a logarithmic scale to show the decay more clearly.

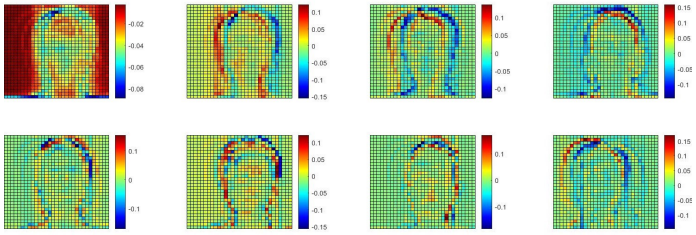


Figure 4: The first 8 modes $\mathbf{U}$ of the uncropped faces. Mode 1-3 highlight the significance of shadows, and mode 4 introduces long hair.

absent from the other 2 classes, and the Newsboys employ a fuller, technically produced sound.

For the Trial 3, the Naive Bayes method performed significantly better than LDA or CART. The accuracy results can be found in Table 3. US Army music remained exceptionally distinguishable from other genres. This is shown quite clearly in Figure 10. When CCM and classical music were misclassified, they were mostly misclassified as each other.

The evaluation of each trial ended at cross-validation. Further testing with hold-out data not used to compute the SVD may reveal further ways to improve the training sets which should then improve classification performance.

5 Summary and Conclusions

This report introduces basic machine learning methods in a way that collectively demonstrates the utility of most of the topics covered in this course. By transforming data into the right domain (Gàbor transforms, wavelets, and spectrograms) and highlighting unique features (SVD, and PCA), it becomes possible to

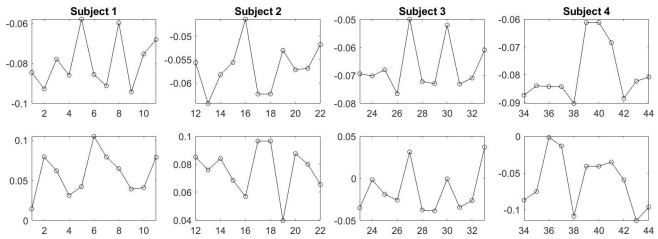


Figure 5: The values of the first 3 modes of $\mathbf{V}$ for the first 4 cropped face subjects.

(a) A 10% reconstruction of Subjects 38 (T) and 37 (B). (b) A 50% reconstruction of Subjects 15 (T) and 11 (B).

Figure 6: Low rank reconstructions of faces from both the cropped (left) and uncropped (right) face sets.

Classifier	AC/DC	John Williams	US Army Band
LDA	80-82%	67-70%	98-100%
NB	71-75%	69-72%	96-98%
CART	67-74%	57-60%	100%

Table 1: Cross-validation accuracy scores for each model on Trial 1 using the first 10 modes $\mathbf{V}$. The accuracy ranges include scores from 4 sets of 250 iterations for each model.

distinguish classes of information within a larger set of data. Even when data is considered to be from closely related classes such as those in Trial 2 of the music classification, it is still possible to amplify subtle differences to distinguish between them.

Beyond the mathematical power of the results shown here, the Yale faces study was validating for me as a military professional. Camouflage is a critical soldier skill, and many of the techniques we teach soldiers are reinforced by the PCA of the uncropped set of images. The human eye can detect the silhouette of a human head very easily, and it is essential to break up that signature to remain undetected. Similarly, the need to darken the brow, nose, and chin with camouflage paint is reinforced by later modes of $\mathbf{U}$.

References

[1] Leo Breiman. *Classification and regression trees*. Routledge, 2017.

[2] Gavin M. Chase. "AMATH 582 Homework 1: FFT Applications Including Filtering by Averaging and Gaussian Filtering". In: *AMATH 582 Portfolio* (2020).

[3] Gavin M. Chase. "AMATH 582 Homework 2: Time-Frequency Analysis of an Unlikely Couple- G.F. Handel and Mary the Shepherdess". In: *AMATH 582 Portfolio* (2020).

[4] Gavin M. Chase. "AMATH 582 Homework 3: The Principal Components of a Paint Can in it Various States of Motion Found Using SVD". In: *AMATH 582 Portfolio* (2020).

[5] Ronald A Fisher. "The use of multiple measurements in taxonomic problems". In: *Annals of eugenics* 7.2 (1936), pp. 179–188.

[6] Alfred Haar. *Zur theorie der orthogonalen funktionensysteme*. Georg-August-Universitat, Gottingen., 1909.

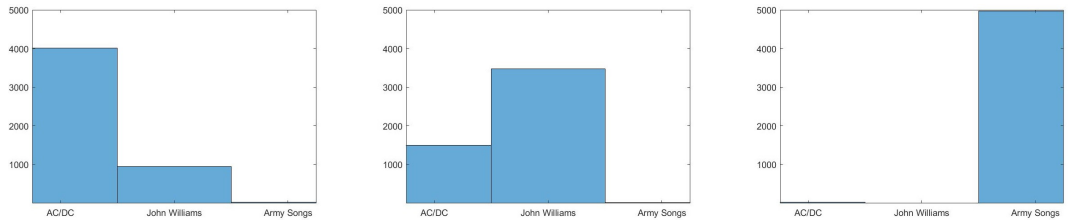


Figure 7: The LDA Classification results after 1000 iterations. The left panel shows predictions for AC/DC samples, the center shows results for John Williams samples, and the right panel shows predictions for the US Army Band.

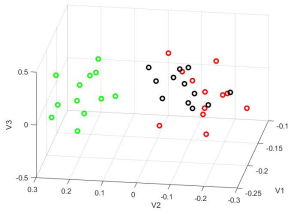


Figure 8: The first 3 modes of $\mathbf{V}$ for Trial 1. The green dots represent the US Army Band, the black represent AC/DC, and the red correspond to John Williams.

Classifier	Barlow Girl	Third Day	Newsboys
LDA	57-68%	43-45%	82-85%
NB	57-60%	35-40%	87-90%
CART	60-64%	27-31%	74-78%

Table 2: Cross-validation accuracy scores for each model on Trial 2 using the first 11 modes $\mathbf{V}$. The accuracy ranges include scores from 4 sets of 250 iterations for each model.

[7] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

[8] David D Lewis. "Naive (Bayes) at forty: The independence assumption in information retrieval". In: *European conference on machine learning*. Springer. 1998, pp. 4–15.

[9] sebastian raschka sebastian. Aug. 2014. URL: `https://sebastianraschka.com/Articles/2014_python_lda.html`.

Appendix A MATLAB Functions

What follows is a list of MATLAB functions not covered in the Homework 1, 2, or 3 Reports as they appear in Appendix B. Much of the text below with additional examples can be found in the Mathworks Documentation Library.

- `[cA,cH,cV,cD]=dwt2(X,'waveletname')` computes the single-level 2-D discrete wavelet transform (DWT) of the input data `X` using the `waveletname` wavelet. dwt2 returns the approximation coefficients matrix `cA` and detail coefficients matrices `cH`, `cV`, and `cD` (horizontal, vertical, and diagonal, respectively).
- `wcodemat(cH,n)` rescales an input matrix `cH` to a specified range `n` for display

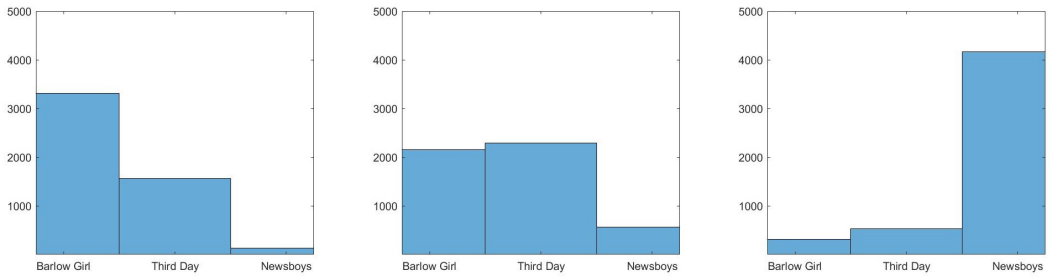


Figure 9: Trial 2 LDA Classification results after 1000 iterations. The left panel shows predictions for Barlow Girl samples, the center shows results for Third Day samples, and the right panel shows predictions for Newsboys samples.

Classifier	CCM	Classical	Army Songs
LDA	59-61%	45-46%	99-100%
NB	68-69%	70-73%	98-99%
CART	56-60%	50-54%	100%*

Table 3: Cross-validation accuracy scores for each model on Trial 3 using the first 10 modes $\mathbf{V}$. The accuracy ranges include scores from 4 sets of 250 iterations for each model.(*scores ranged from 99.8-100%)

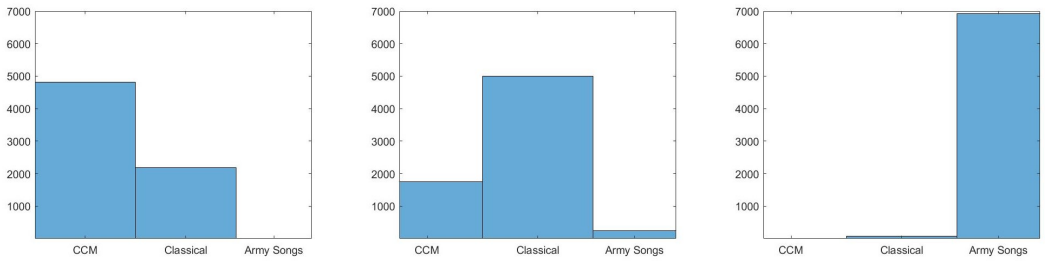


Figure 10: The Naive Bayes Classification results after 1000 iterations. The left panel shows predictions for Contemporary Christian Music samples, the center shows results for classical samples, and the right panel shows predictions for US Army song samples.

- `randperm(n)` returns a row vector containing a random permutation of the integers from 1 to n without repeating elements.
- `[predictions]=classify(test,train,labels)` classifies each row of the data in `test` into one of the groups in `training`.The output `predictions` indicates the group to which each row of `test` has been assigned, and is of the same type as `labels`.
- `fitcnb(train,labels)` returns a multiclass naive Bayes model (`Mdl`), trained by the predictors in `train` and class labels.
- `predict(Mdl, test)` returns a vector of predicted class labels for `test` based on the output `Mdl` from `fitcnb` or `fitctree`
- `fitctree(train,labels)` returns a fitted binary classification decision tree based on the input variables `train` and output `labels`

Appendix B MATLAB Code

This appendix includes the MATLAB code used to produce the results in this report. The code can also be found at `https://github.com/gchase15/AMATH-582`

```
%%Compile Faces and Organize into a single  matrix
clear all; close all; clc

fold=1:38;
mainpath = 'C:\Users\gavin\Documents\MATLAB\CroppedYale\yaleB';
cropface=[];
cropaveface=[];

%First for loop creates the path to the pictures
for k=1:length(fold)
    foldnum=string(k);
    newpath=strcat(mainpath,foldnum);
    D=dir(fullfile(newpath,'*.pgm'));

    %Second for loop pulls out the images, reduces the
    %resolution, turns it into a row, and adds it to a
    %collective matrix of all faces. Pixels are still uint8.
    for j=1:numel(D)
        face = imread(fullfile(newpath,D(j).name));
        blurface = imresize(face, [64,64]);
        aveface = double(blurface) - mean(double(blurface));
        fatface = reshape(blurface,1,4096);
        fataveface = reshape(aveface,1,4096);
        cropaveface = [fataveface' cropaveface];
        cropface = [fatface' cropface];
    end
end

%% Pull out first 4 face sets

face1 = cropaveface(:,1:64);
face2 = cropaveface(:,65:128);
face3 = cropaveface(:,129:192);
face4 = cropaveface(:,193:256);

% figure(1)
% for j=1:25
%     normface=reshape(face2(:,j),64,64);
%     subplot(5,5,j)
%     imshow(normface)
% end
```

Listing 1: MATLAB code covering compiling Yale Face images into a single matrix.

```
%% SVD of whole set

face_wave = dc_wavelet(cropaveface);

[U,S,V] = svd(face_wave,'econ');

figure(2)
for j=1:12
  subplot(3,4,j)
  ut1 = reshape(U(:,j),32,32);
  ut2 = ut1(32:-1:1,:);
  pcolor(ut2), colormap(jet)
  set(gca,'Xtick',[],'Ytick',[])
  colorbar
end
%% Plot and assess Singular Values

figure(3)
subplot(2,1,1)
plot(diag(S),'ko','Linewidth',[2])
set(gca,'Fontsize',[14])
axis([0 1024 0 8*10^4])
xlabel('Singular Values')
ylabel('\sigma')
subplot(2,1,2)
semilogy(diag(S),'ko','Linewidth',[2])
set(gca,'Fontsize',[14])
axis([0 1024 0 8*10^4])
xlabel('Singular Values')
ylabel('\sigma')

%Singular value assessment
sig=diag(S);
energy1=sig(1)/sum(sig)
energy2=sig(2)/sum(sig)
eng1perc=sum(sig(1:3))/sum(sig)
eng10perc=sum(sig(1:102))/sum(sig)
eng25perc=sum(sig(1:256))/sum(sig)
eng50perc=sum(sig(1:512))/sum(sig)
eng=sum(sig(1:789))/sum(sig)
```

Listing 2: MATLAB code covering SVD of Yale Faces along with analysis of $\mathbf{\Sigma}$.

```
function dcData = dc_wavelet(dcfile)
[m,n]=size(dcfile); % 4096 x 80
nw=32*32; % wavelet resolution

nbcol = size(colormap(gray),1);
for i=1:n
  X=double(reshape(dcfile(:,i),64, 64));
  [cA,cH,cV,cD]=dwt2(X,'haar');
  cod_cH1 = wcodemat(cH,nbcol);
  cod_cV1 = wcodemat(cV,nbcol);
  cod_edge=cod_cH1+cod_cV1;
  dcData(:,i)=reshape(cod_edge,nw,1);
end
```

Listing 3: MATLAB code covering edge detection with the Haar wavelet.

```
%% Analyze V
k=0;
figure(4)
for j=1:3
    k=k+1;
  subplot(3,4,k)
  plot(1:64,V(1:64,j),'ko-')
  k=k+1;
  subplot(3,4,k)
  plot(65:128,V(65:128,j),'ko-')
  k=k+1;
  subplot(3,4,k)
  plot(129:192,V(129:192,j),'ko-')
  k=k+1;
   subplot(3,4,k)
  plot(193:256,V(193:256,j),'ko-')
end

subplot(3,4,1), set(gca,'Xlim',[0 64],'Fontsize',[14]), title('Subject 1')
subplot(3,4,2), set(gca,'Xlim',[65 128],'Fontsize',[14]), title('Subject 2')
subplot(3,4,3), set(gca,'Xlim',[129 192],'Fontsize',[14]), title('Subject 3')
subplot(3,4,4), set(gca,'Xlim',[193 256],'Fontsize',[14]), title('Subject 4')
subplot(3,4,5), set(gca,'Xlim',[0 64],'Fontsize',[14])
subplot(3,4,6), set(gca,'Xlim',[65 128],'Fontsize',[14])
subplot(3,4,7), set(gca,'Xlim',[129 192],'Fontsize',[14])
subplot(3,4,8), set(gca,'Xlim',[193 256],'Fontsize',[14])
subplot(3,4,9), set(gca,'Xlim',[0 64],'Fontsize',[14])
subplot(3,4,10), set(gca,'Xlim',[65 128],'Fontsize',[14])
subplot(3,4,11), set(gca,'Xlim',[129 192],'Fontsize',[14])
subplot(3,4,12), set(gca,'Xlim',[193 256],'Fontsize',[14])
```

Listing 4: MATLAB code covering analysis of $\mathbf{V}$.

```
%% Reproduction
figure(6)
ff1=zeros(32,32);
ff2=ff1;
for j=1:61

    ut1 = reshape(U(:,j),32,32);
    ffo = ut1*sig(j)*V(24,j);
    ff1 = ff1 + ffo;
    ff = ut1*sig(j)*V(70,j);
    ff2 = ff2 + ff;
%     imshow(uint8(ffo))
    ff1a = ff1(32:-1:1,:);
    if j >= 50
    subplot(3,4,(62-j))
    pcolor(ff1a/j), colormap(gray)
  set(gca,'Xtick',[],'Ytick',[])
  colorbar
    end
end

figure(5)
ff1=zeros(32,32);
ff2=ff1;
for j=1:102
    ut1 = reshape(U(:,j),32,32);
    ffo = ut1*sig(j)*V(24,j);
    ff1 = ff1 + ffo;
    ffoa = ut1*sig(j)*V(70,j);
    ff2 = ff2 + ffoa;
%     imshow(uint8(ffo))
    ff1a = ff1(32:-1:1,:);
    ff2a = ff2(32:-1:1,:);
%     subplot(3,4,j)
%     pcolor(ff1a/j), colormap(gray)
%   set(gca,'Xtick',[],'Ytick',[])

end

figure(7)
subplot(2,2,1)
pcolor(ff1a/j), colormap(gray)
set(gca,'Xtick',[],'Ytick',[])
normface=reshape(cropface(:,24),64,64);
subplot(2,2,2)
imshow(normface)

subplot(2,2,3)
pcolor(ff2a/j), colormap(gray)
set(gca,'Xtick',[],'Ytick',[])
normface=reshape(cropface(:,70),64,64);
subplot(2,2,4)
imshow(normface)
```

Listing 5: MATLAB code covering computing the reproduction of Yale Face images from a set number of PCA features.

```
clear all; close all; clc
mainpath = 'C:\Users\gavin\Documents\MATLAB\HW4Music\Bluegrass\newBlue';
D=dir(fullfile(mainpath,'*.m4a'));

%%  Construct Spectrogram images of music files
Lclip=5.4;
     for j=1:numel(D)
         musicfile = audioread(fullfile(mainpath,D(j).name));
         resampmusic = [];
         for i=1:(floor(length(musicfile)/2))
             resampmusic = [resampmusic musicfile(i*2)];
         end
         Fs = length(resampmusic)/Lclip;
         nclip = length(resampmusic);
         tp2=linspace(0,Lclip,nclip+1);
         tp=tp2(1:nclip);
         if rem(length(resampmusic),2) == 0
             kp=(2*pi/Lclip)*[0:nclip/2-1 -nclip/2:-1];
             kps=fftshift(kp);
         else
             kp=(2*pi/Lclip)*[0:nclip/2 -nclip/2:-1];
             kps=fftshift(kp);
         end
%          figure(13)
%          plot((1:length(resampmusic))/Fs,resampmusic);
%          xlabel('Time [sec]');
%          ylabel('Amplitude');
%          drawnow
     width=1214;
    step=zeros(1,length(tp));
    mask=ones(1,2*width+1);
    musicstept_spec=[];
    tslide=(width+1):800:(length(step)-width);
    for z=1:length(tslide)
         step=zeros(1,length(tp));
         step(tslide(z)-width:1:tslide(z)+width)=mask;
         musicstep=step.*resampmusic;
         musicstept=fft(musicstep);
         musicstept_spec=[musicstept_spec;
         abs(fftshift(musicstept))];
    end

tslidep=linspace(0,Lclip,length(tslide));
f=figure('visible', false);
pcolor(tslidep,kps/(2*pi),musicstept_spec.'),
shading interp
set(gca,'XTick',[],'YTick',[],'Ylim',[0 2000],'Fontsize',[14])
colormap(hsv)
newFileNameChar = 'Bluegrass' + string(j+13) + '.jpg';
print(newFileNameChar,'-djpeg');
close(f)
filescomplete = j
    end
```

Listing 6: MATLAB code covering the resampling of music files and conversion into spectrograms.

```
 %% Gets rid of white border and grayscales images

     mainpath = 'C:\Users\gavin\Documents\MATLAB\HW4Music\Test3Samples';
D=dir(fullfile(mainpath,'*.jpg'));
%     figure(8)
    for j=1:numel(D)
        thisfile = fullfile(mainpath,D(j).name);
        SpectroImg = imread(fullfile(mainpath,D(j).name));
        SpectroGray = double(rgb2gray(SpectroImg));
        SpectroCrop = SpectroGray(75:584,120:785);
%         imshow(uint8(SpectroCrop))
%         pause(.2)
        newFileName = split(thisfile,".");
        filename = string(newFileName(1,1));
        newFileNameChar = filename + '_cropped.jpg';
        imwrite(uint8(SpectroCrop),newFileNameChar);
    end
```

Listing 7: MATLAB code covering cropping spectrograms and converting them to grayscale.

```
%% Compiles elements of test data into individual matrices
clear all; close all; clc
addpath('HW4Music\Test3Samples')
     mainpath = 'C:\Users\gavin\Documents\MATLAB\HW4Music\Test3Samples';

    D=dir(fullfile(mainpath,'CCM*cropped.jpg'));
Band2Samp=[];
Band2AveSamp=[];

    for j=1:numel(D)
           song = imread(fullfile(mainpath,D(j).name));
         blurspectro = double(imresize(song, [64,64]));
%          imshow(blurspectro)
         avespectro = double(blurspectro) - mean(double(blurspectro));
         skinnyspectro =reshape(blurspectro,1,4096);
         skinnyavespectro = reshape(avespectro,1,4096);
         Band2AveSamp = [skinnyavespectro' Band2AveSamp];
         Band2Samp = [skinnyspectro' Band2Samp];
    end

    D=dir(fullfile(mainpath,'Classical*cropped.jpg'));
Band3Samp=[];
Band3AveSamp=[];

    for j=1:numel(D)
           song = imread(fullfile(mainpath,D(j).name));
         blurspectro = double(imresize(song, [64,64]));
%          imshow(blurspectro)
         avespectro = double(blurspectro) - mean(double(blurspectro));
         skinnyspectro =reshape(blurspectro,1,4096);
         skinnyavespectro = reshape(avespectro,1,4096);
         Band3AveSamp = [skinnyavespectro' Band3AveSamp];
         Band3Samp = [skinnyspectro' Band3Samp];
    end

        D=dir(fullfile(mainpath,'ArmySongs*cropped.jpg'));
Band4Samp=[];
Band4AveSamp=[];

    for j=1:numel(D)
           song = imread(fullfile(mainpath,D(j).name));
         blurspectro = double(imresize(song, [64,64]));
%          imshow(blurspectro)
         avespectro = double(blurspectro) - mean(double(blurspectro));
         skinnyspectro =reshape(blurspectro,1,4096);
         skinnyavespectro = reshape(avespectro,1,4096);
         Band4AveSamp = [skinnyavespectro' Band4AveSamp];
         Band4Samp = [skinnyspectro' Band4Samp];
    end
```

Listing 8: MATLAB code covering the compilation task of music samples to be tested for classification.

```
%% Wavelet Transform

%   band1_wave = dc_wavelet(Band1AveSamp);
    band2_wave = dc_wavelet(Band2AveSamp);
    band3_wave = dc_wavelet(Band3AveSamp);
    band4_wave = dc_wavelet(Band4AveSamp);

    [U,S,V]=svd([band2_wave,band3_wave,band4_wave],0);

    %% U Analysis
    figure(2)
for j=1:12
  subplot(3,4,j)
  ut1 = reshape(U(:,j),32,32);
  ut2 = ut1(32:-1:1,:);
  pcolor(ut2), colormap(jet)
  set(gca,'Xtick',[],'Ytick',[])
  colorbar
end

%% Analyze S
figure(3)
subplot(2,1,1)
plot(diag(S),'ko','Linewidth',[2])
set(gca,'Fontsize',[14])
axis([0 112 0 3*10^4])
xlabel('Singular Values')
ylabel('\sigma')
subplot(2,1,2)
semilogy(diag(S),'ko','Linewidth',[2])
set(gca,'Fontsize',[14])
axis([0 112 0 3*10^4])
xlabel('Singular Values')
ylabel('\sigma')

%Singular value assessment
sig=diag(S);
energy1=sig(1)/sum(sig)
energy2=sig(2)/sum(sig)
eng10perc=sum(sig(1:11))/sum(sig)
eng25perc=sum(sig(1:28))/sum(sig)
eng50perc=sum(sig(1:56))/sum(sig)
eng=sum(sig(1:64))/sum(sig)%find 95% of S
%% 3D version of V analyzation
figure(5)
plot3(V(1:28,1),V(1:28,2),V(1:28,3),'ko','Linewidth',[2])
hold on
plot3(V(29:56,1),V(29:56,2),V(29:56,3),'ro','Linewidth',[2])
hold on
plot3(V(57:84,1),V(57:84,2),V(57:84,3),'go','Linewidth',[2])
% hold on
% plot3(V(85:112,1),V(85:112,2),V(85:112,3),'bo','Linewidth',[2])
grid on, xlabel('V1'), ylabel('V2'), zlabel('V3')
```

Listing 9: MATLAB code covering wavelet transformation of music samples, SVD of the music, analysis of $\mathbf{U}$, and analysis of $\mathbf{\Sigma}$.

```
%% Classification Task

band1prelda = []; band2prelda = []; band3prelda = []; band4prelda = [];
band1prenb = []; band2prenb = []; band3prenb = []; band4prenb = [];
band1pretree = []; band2pretree = []; band3pretree = []; band4pretree = [];
n=250

for k=1:n
    %pick a new randomization for each test.
q1 = randperm(28); q2 = randperm(28); q3 = randperm(28); q4 = randperm(28);

xband1 = V(1:28,1:10); xband2 = V(29:56,1:10);
xband3 = V(57:84,1:10);
% xband4 = V(85:112,1:10);

xtrain = [xband1(q1(1:21),:); xband2(q2(1:21),:); xband3(q3(1:21),:)];
xtest = [xband1(q1(22:end),:); xband2(q2(22:end),:); xband3(q3(22:end),:)];
ctrain = [ones(21,1); 2*ones(21,1); 3*ones(21,1)];

%the classifiers (LDA,Naive Bayes, CART)
[predictlda] = classify(xtest,xtrain,ctrain);
nb = fitcnb(xtrain,ctrain);
predictnb = nb.predict(xtest);
tree=fitctree(xtrain,ctrain);
predicttree = predict(tree,xtest);

% Collect the results
band1prelda = [band1prelda predictlda(1:7)];
band2prelda = [band2prelda predictlda(8:14)];
band3prelda = [band3prelda predictlda(15:21)];
% band4prelda = [band4prelda predictlda(22:28)];
band1prenb = [band1prenb predictnb(1:7)];
band2prenb = [band2prenb predictnb(8:14)];
band3prenb = [band3prenb predictnb(15:21)];
% band4prenb = [band4prenb predictnb(22:28)];
band1pretree = [band1pretree predicttree(1:7)];
band2pretree = [band2pretree predicttree(8:14)];
band3pretree = [band3pretree predicttree(15:21)];
% band4pretree = [band4pretree predicttree(22:28)];

end
success1lda = sum(band1prelda(:) == 1)/(n*7)
success2lda = sum(band2prelda(:) == 2)/(n*7)
success3lda = sum(band3prelda(:) == 3)/(n*7)
% success4lda = sum(band4prelda(:) == 4)/(n*7)
success1nb = sum(band1prenb(:) == 1)/(n*7)
success2nb = sum(band2prenb(:) == 2)/(n*7)
success3nb = sum(band3prenb(:) == 3)/(n*7)
% success4nb = sum(band4prenb(:) == 4)/(n*7)
success1tree = sum(band1pretree(:) == 1)/(n*7)
success2tree = sum(band2pretree(:) == 2)/(n*7)
success3tree = sum(band3pretree(:) == 3)/(n*7)
% success4tree = sum(band4pretree(:) == 4)/(n*7)
```

Listing 10: MATLAB code covering classification of music samples.

```
%% Display classification Results

figure(6)
subplot(1,3,1)
histogram(band1prenb,3,'BinWidth',.8)
axis([.8 3 0 7000]), xticks([1.2 2 2.7]),xticklabels({'CCM', 'Classical', 'Army Songs'})
set(gca,'YTick',[1000 2000 3000 4000 5000 6000 7000], 'Fontsize',[12])
subplot(1,3,2)
histogram(band2prenb,3,'BinWidth',.8), axis([1 3 0 7000])
xticks([1.2 2 2.7]),xticklabels({'CCM', 'Classical', 'Army Songs'})
set(gca,'YTick',[1000 2000 3000 4000 5000 6000 7000], 'Fontsize',[12])
subplot(1,3,3)
histogram(band3prenb,3,'BinWidth',.8), axis([1 3 0 7000])
xticks([1.2 2 2.7]),xticklabels({'CCM', 'Classical', 'Army Songs'})
set(gca,'YTick',[1000 2000 3000 4000 5000 6000 7000], 'Fontsize',[12])
```

Listing 11: MATLAB code covering the Histogram construction after classification.