Data Mining, Classification & Clustering

University of Washington
AMATH 582 Homework 4
Winter Quarter 2019

David Long
dmlong888@live.com

Abstract

This paper will explore some of the industry-leading techniques that are currently employed in the fields of data mining, clustering and classification. Topics addressed include Fourier transforms, singular value decompositions, *k*-means, agglomerative and divisive hierarchical clustering, and linear discriminant analysis.

I. Introduction and Overview

The science of statistics is relatively new. While scientists have been compiling large sets of measurements for hundreds of years, performing calculations on them was daunting. The sheer volume of arithmetic computations that had to be done by hand was prohibitively time consuming and prone to errors. The fact that Ronald Fisher (b.1890-d.1962) was able to lay the foundations in the pre-computer age is astonishing. The manipulations of data sets that mega-corporations such as Google and Kaiser Permanente casually employ on a routine basis were impractical even thirty years ago. Computational machines with the ability to process millions of mathematical calculations in fractions of seconds have made it possible for professionals and hobbyists alike to process and interpret remarkably large sets of data. The science of statistics is booming right now, motivated primarily by the desires of corporations to extract meaningful and actionable information from immense data sets with no apparent point of entry. The bleeding edge of these advancements are being made in the fields of data mining and clustering, through which mathematical algorithms are employed to parse data and classify new observations. This paper will explore some of the current techniques in widespread use today.

Data can be represented in many forms. The alternative forms are sometimes useful for various tasks. Media files are often given a Fast Fourier Transform, which facilitates data compression. Singular Value Decomposition transforms a data set to a different basis, which can reduce redundancy and dimensional rank. In the quest to classify and cluster data, data transformations can even increase accuracy. A classification algorithm applied to raw data which is no better than random guessing may work quite well after the data have been transformed. The science of data mining endeavors to identify the optimal representation of the data so that meaningful features can be extracted and used for classification and clustering. Data which is very high in dimension may thus be reduced to a low-dimensional feature space, which may not only improve classification accuracy, but also realize computational savings. To illustrate the analysis of a transformed feature space, the Yale Face Database will be explored. The methods illustrated in this process will then be applied to a sample of audio files and the extracted feature space will be used in classification algorithms.

II. Theoretical Background

While there are numerous classification techniques that are in use today, they can all be categorized as either *supervised learning* or *unsupervised learning*. In supervised learning, a set of labeled values are used to inform the algorithm. The labels are provided by someone with knowledge of the data. Most of the data is used as a training set, but some of it is reserved to cross-validate the informed algorithm. As the data are labeled, the accuracy of the algorithm can be assessed before applying it to new data that may be unlabeled. Many random subsets of the data should be used to train the algorithm and assess its long-run accuracy. Furthermore, several representations of the dataset (e.g. Fourier transforms, singular value decompositions, etc.) can

be compared under the rigor of the algorithm. In unsupervised learning, no labels are available. The data set is simply fed to the algorithm and clusters are output which may or may not have practical import. Whether or not a given algorithm is successful in identifying clusters is largely dependent on the structure of the dataset and the method that is chosen to parse it. In general, supervised learning is more successful, especially when the clusters have significant overlap, but sometimes labels are too expensive to obtain or simply unavailable. Unsupervised algorithms can still be informative in such cases.

To explore some of the various classification algorithms that are in widespread use today, samples of MP3 files from my music collection were imported into MATLAB. Various supervised and unsupervised techniques were used to cluster the samples. The knowledge of labels such as genre and artist enabled the assessment of accuracy for both techniques. While all of the algorithms described below were explored for all classification tasks, in the interest of brevity, only a representative example of each will be discussed.

The first unsupervised algorithm to be considered is the *k*-means algorithm. This method will separate the observations into *k* groups. Initially, *k* means are guessed. The distances between an observation and each of the means are calculated according to some metric and the observation is classified as belonging to the closest mean. After doing this for all observations, *k* clusters will exist. The mean of each cluster is calculated and the classification according to distance is repeated with the updated means. This process iterates until the change in the *k* means based on the new cluster set is within some tolerance, which indicates that the process has converged. As the convergence values are sensitive to the initial guesses, it is not uncommon to replicate the entire *k*-means process again using converged values from the previous iteration as the initial guesses in the new iteration.

Another algorithm is hierarchical clustering, which can be either agglomerative or divisive. The former is unsupervised while the latter is supervised. In agglomerative clustering, each observation begins as its own cluster and the algorithm is complete when all the data have been merged. Divisive clustering begins with all of the observations in one cluster and divides it into subgroups until each point is in its own cluster. Agglomerative clustering will be considered here, while the divisive decision tree algorithm will be considered in the supervised learning section. Assume that the dataset has *n* observations. The distance between all pairs of points is calculated according to some metric, which requires that ${}_nC_2$ distances be calculated. The points with the smallest distance between them are then replaced by one point represented by their center of mass. The process is repeated on the new set of $n-1$ points. At each stage, the pair of points that were merged are recorded so that the hierarchy may be summarized in a graphical display called a *dendrogram*, which shows both the pairs that are merged and the distance at which the merging occurs. The overall number of clusters can thus be controlled by setting a distance threshold.

Perhaps the earliest method for supervised learning was developed by the godfather of statistics himself, Ronald Fisher, who outlined the algorithm in 1936 to standardize the classification of

species, which up until then had been a very subjective process.[1] The objective of *linear discrimination analysis* is to project the data onto the basis that yields the optimal separation of the data distributions. Analogous to his *F*-statistic, the degree of separation was quantified as a ratio of variability between the distributions to variability within the distributions. The analysis boiled down to optimizing the quantity

$$\vec{w} = \arg\max_{\vec{w}} \frac{\vec{w}^T S_B \vec{w}}{\vec{w}^T S_W \vec{w}} \qquad \textbf{(Eqn. 1)}$$

where $\vec{w}$ is the vector onto which the data are projected and S_B and S_W are scatter matrices for the between-class and within class data.[1] When the ratio is low, the means of the distributions are close together and the data are intermingled. When the ratio is high, the means are well separated and the distributions are easier to distinguish from one another. As the problem is stated in terms of a projection vector, it should be no surprise that Eqn. 1 can be reduced to an eigenvalue problem $S_B \vec{w} = \lambda S_W \vec{w}$.

The next supervised learning algorithm that will be considered is a divisive hierarchical clustering scheme known as a *decision tree*. As previously stated, the observations are initially considered to belong to the same universal cluster. The algorithm then examines each variable one at a time and determines a cut-point on its axis that isolates a maximal number of observations in the same cluster from the rest of the data. Without labels on the data, this determination would not be possible. The algorithm then repeats the process on the remaining observations using a new variable. The end-result is a branching at binary nodes with limits on the variables that either classify an observation or pass it forward for further discrimination. The user has a lot of control over parameters such as which variables and how many nodes are used. Once the tree is established, new observations can be passed down the decision tree and classified accordingly. The implementation is tremendously easy in MATLAB, as the algorithm will automatically divide the data into training and cross-validation sets. Replication is important to ensure that the decision tree is not over-fit to the training data.

The last algorithm that will be employed in this investigation is the *naïve Bayes* test, which uses the statistician's classic formula for conditional probabilities. Given labels for *k* clusters of training data and a new observation vector $\vec{x}$, the probability of classifying it as belonging to C_k is given by

$$P\left(C_k \middle| \vec{x}\right) = \frac{P\left(\vec{x} \middle| C_k\right) P\left(C_k\right)}{P\left(\vec{x}\right)} \qquad \textbf{(Eqn. 2)}$$

Based solely on the given data, the algorithm calculates the probability that the vector belongs to each C_i and assigns the new observation to the cluster with the highest probability.

The algorithms described here were used to produce the data output in this report. For a discussion of other algorithms that were explored but not used, see **Appendix A**.

III. Algorithm Implementation and Development

The Yale Face Database has two sets of faces. The first consists of approximately 60 images taken under different lighting conditions of 38 different subjects. A handful that had the label ".bad" were discarded. The images had been well-cropped so as to fill the frame with the subject's face. The second set consists of 11 images of 15 subjects, each with a different expression such as "glasses", "sleepy", "rightlight", or "wink". A loop was constructed to import an image, convert it to data type *double*, reshape it as a vector, and add it to a vector that accumulated the total. The two data sets were arranged as matrices with one image per column. The average faces for both sets are illustrated in **Figure 1**. The average face was then subtracted from each image vector to center the data. The centered data were given singular value decompositions, which returned factorizations of the data matrices of the form $Y = U\Sigma V$.

The columns of *U* are the new basis vectors and are equal in length to the number of pixels in each image. The number of basis vectors (modes) is equal to the number of images in the data set. The objective of the data mining is to identify a small subset of these modes that drastically reduces the size of the data set while still capturing most of the variance. Each mode captures some of the defining facial structures in the data set, as illustrated in **Figure 2**.

The diagonal matrix Σ gives the singular values, which are the square roots of the variance captured by each mode of *U*. As such, they dilate or compress the axes of the rotated basis *U*. The cumulative proportion of variance captured by the first fifty singular values are shown in **Figure 3**.

The square matrix *V* gives the weighting coefficients of each mode. The first column, for instance, gives the coefficient of each image for the first mode. The rows of *V* then give a complete set of coefficients for each image in the data set.

The singular value spectrums of **Figure 3** illustrate how a subset of the modes can be used to reconstruct approximations of the data set. Over 80% of the variance can be captured by just the first five modes of both data sets. The first step in reconstructing the images is to take a subset of modes from U to form the matrix $\hat{U}$ and correspondingly truncate the matrices Σ and *V*. The number of included modes is the truncated rank *r* and the product $Y_r = \hat{U}\hat{\Sigma}\hat{V}$ is a dimensionally-reduced representation of the data set. To complete the approximation, each image vector needs to have the average face added to it.

For the cropped images, a sizable portion of the variance can be captured by the first nine modes, but **Figure 4** shows that the facial reconstruction still leaves a lot to be desired. The

reconstructions are blurry and difficult to identify as approximations of the original image. Faces are complex structures. Though adding successive modes contributes a relatively small amount to the overall variance that is captured in the data set, it does make a difference. The truncated basis of rank fifty yields a reconstruction that captures 95% of the variance and is probably good enough to identify as the person in the original image. With a rank of 200, fine details such as whiskers are starting to become apparent. While 200 modes may seem like overkill, the reconstruction is very good and represents a sizable reduction of the original 32,256 modes.

Almost all of the variance in the uncropped data set is captured by a truncated basis of rank ten, though **Figure 5** shows that the image reconstruction is poor. This is attributed to the poor cropping of the images. Misalignments of the facial features yield the poor approximation of an average face in **Figure 1**. Less than half of each image is occupied by the face. The rest contains relatively simple constructs such as backgrounds and shadows, which is a plausible explanation for the low rank necessary to capture all of the variance.

With some basic data mining techniques in hand, I moved on to analyzing the music samples. For each test, I created a cell array to store the labels. Each cell had a vector of strings that stored the artist, title, and genre of each track. Interested readers can peruse the catalog in **Appendix C**.

The tracks were stereo, so I averaged the channels to get a single signal, then took a five second sample of each signal. Since music is all about sound waves, the signals were given a Fourier transform into the frequency space with MATLAB's built-in `spectrogram` command, which returned a matrix of eight columns. The data were complex, so I took the absolute value as I reshaped it into a column vector. This was repeated for each of the thirty songs in each test and the column vectors were concatenated to form the data matrix.

Attempts to reduce the dimension of the frequency data by using a singular value decomposition were very successful. **Figure 6** shows that essentially all of the variance can be described by just one mode. This is not surprising, as the data is simply being projected onto the frequency axis. The components of the first mode were determined to be sufficient for feeding to the classification algorithms.

Structuring the data is the time-intensive chore of any data analysis. The clustering algorithms are all coded into MATLAB. The analyst need do nothing more than call them. For examples of calling the algorithms described in this paper, see **Appendix B**.

For Test 1, the objective was to classify the music of three different artists from three different genres. The results of four trials of a *k*-means classification are shown in **Figure 7**. Each trial was run with five replicates and produced the same convergence set each time. Note that the labels in the unsupervised algorithm are meaningless; all that matters is the grouping. I expected Miles Davis to be the easiest to identify, as Buddy Guy and Metallica are both heavy on the

electric guitar, but this cluster had the smallest success rate at just 60%. Buddy Guy and Metallica were correctly classified at rates of 100% and 80% respectively. The dendrogram in **Figure 8**, which will always produce the same results under replication, gave the exact same classification groups as the *k*-means classification. The dendrogram reveals that distances between clusters are very large compared to the distances within clusters.

In Test 2, the classification was between three different artists within the same genre. I chose to work with three heavyweights from the jazz world, namely Herbie Hancock, John Coltrane, and Miles Davis. I expected pretty poor results here due to the fact that Herbie and Trane both had stints with Miles. I deliberately chose some songs to represent that confounding. Further complicating the classification was the fact that Herbie and Miles both developed several distinct stylistic periods throughout their careers. I expect that the results might have been better if I had chosen artists whose styles were more consistent over their careers, but good statistical practice never seeks to tailor the data to achieve desired results. Drum roll, please. . .

The *k*-means and agglomerative clustering methods were entirely unable to distinguish between Trane and Miles. All twenty tracks were identified as belonging to the same cluster. Supervised methods had some success, though. One hundred replications of a linear discriminant analysis were run on the jazz data set. For each replication, a random subset of seven songs by each artist were put into the training set and the remaining three were reserved for cross-validation. The errors were calculated by determining whether or not the classification label was equal to the truth label. They were summed, converted to a percentage, and plotted as a time-series in **Figure 9**. The plot reveals apparent upper and lower bounds for the accuracy rate, as the only observed accuracy rates were 4/9, 5/9, and 6/9. The lower bound, however, is still greater than the binomial expectation of correctly guessing 3/9 songs at random. The average accuracy rate was about 60%.

For Test 3, ten different artists from blues, hard rock, and jazz were chosen. The unsupervised methods yielded results very similar to those in Test 1. All of the hard rock tracks were identified correctly and it was also the cluster chosen whenever a misclassification was made. The linear discrimination analysis had an average accuracy rate about 50%. I was very surprised by the accuracy of the divisive hierarchical method.

To construct a decision tree, I first had to create a cell array vector of string labels. I experimented with different training sets, such as the raw data, the raw spectrograms, and the components of the first mode of the singular value decomposition, which accounted for essentially all of the variance. The accuracy rate skyrocketed when I sent the entire matrix of components to the algorithm. **Figure 10** illustrates that ten replications had an average success rate of almost 80%. The decision tree for the final replication is shown in **Figure 11**. Surprisingly, only four branches are necessary. Couple the accuracy with its relatively simplistic approach and one can see why the decision tree is one of the favored methods in classification and clustering.

I didn't expect that I would be able to improve on the accuracy of the decision tree, but the naïve Bayes algorithm crushed it out of the park for all three tests. MATLAB tore through 100 replications of randomly selecting training sets and putting them to the test with the cross-validation sets. For Test 1, the average accuracy rate was about the same as the LDA at 80%. For Test 2, LDA was just under 60%. Naïve Bayes upped the average accuracy rate to almost 90%. Similar success rates were seen in Test 3, where LDA barely crossed the 50% threshold and naïve Bayes was pushing 90% **(Figure 12)**. The name sells it short, as it is clearly a very well-informed algorithm.

IV. Computational Results

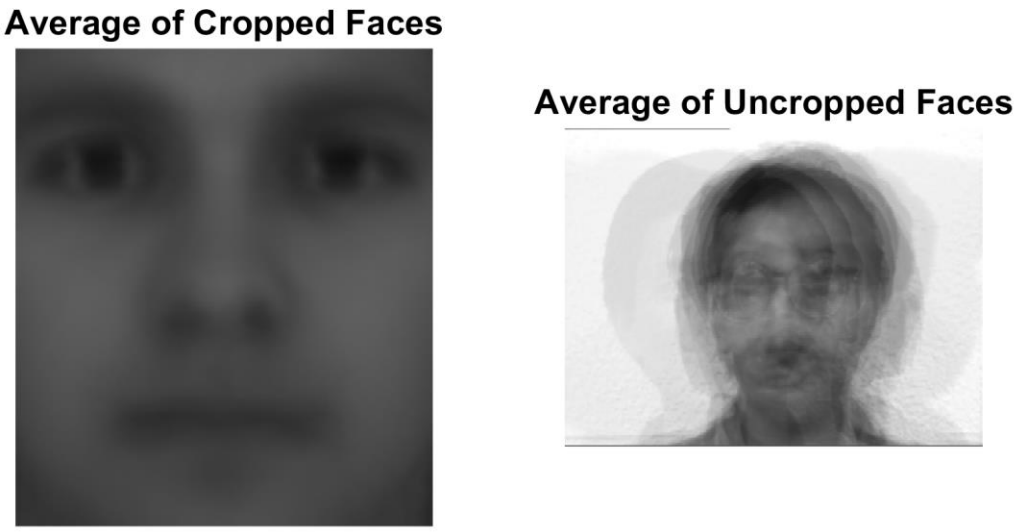

Figure 1: The cropped faces yield a composite image that looks like a person, possibly actor Colin Farrell, who apparently has a very generic face. The data in the uncropped images captures many features that are not pertinent to face reconstruction, such as clothing and background shadows. The composite image barely looks like a face.

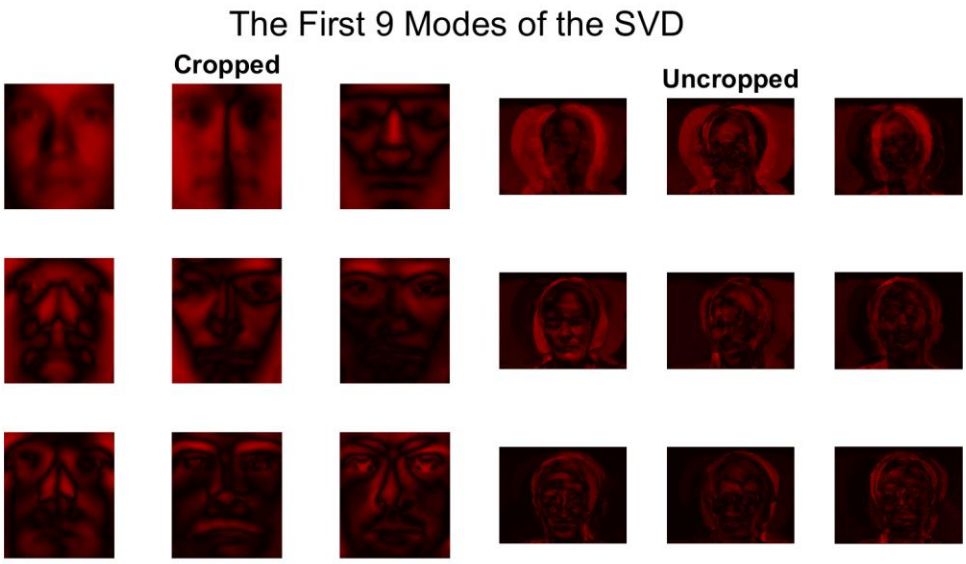

Figure 2: The first nine modes of the singular value decomposition for both the cropped and uncropped image sets. The modes for the cropped set clearly highlight defining facial structures. The modes for the uncropped sets are blurry by comparison. The first two seem to be capturing the state of the background shadows while the ninth seems to capture the glasses. Some of the modes clearly capture facial structures, though the fourth and seventh modes appear to be biased towards individuals from the data set rather representing generic facial structures.

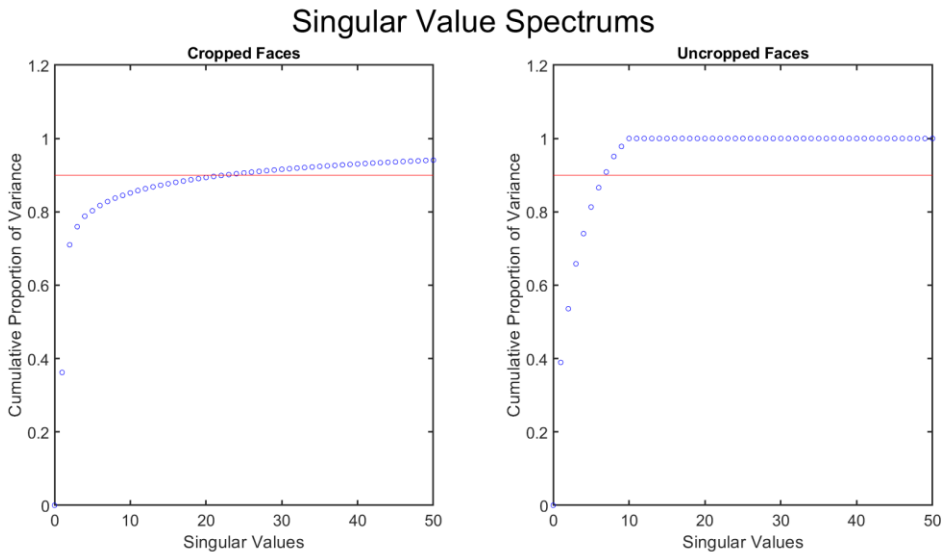

Figure 3: The red line indicates the threshold for capturing 90% of the total variance in the data sets. A sizable proportion of the variance is captured by just the first five modes for both data sets. Virtually all of the variance is captured by the first ten modes of the uncropped images.

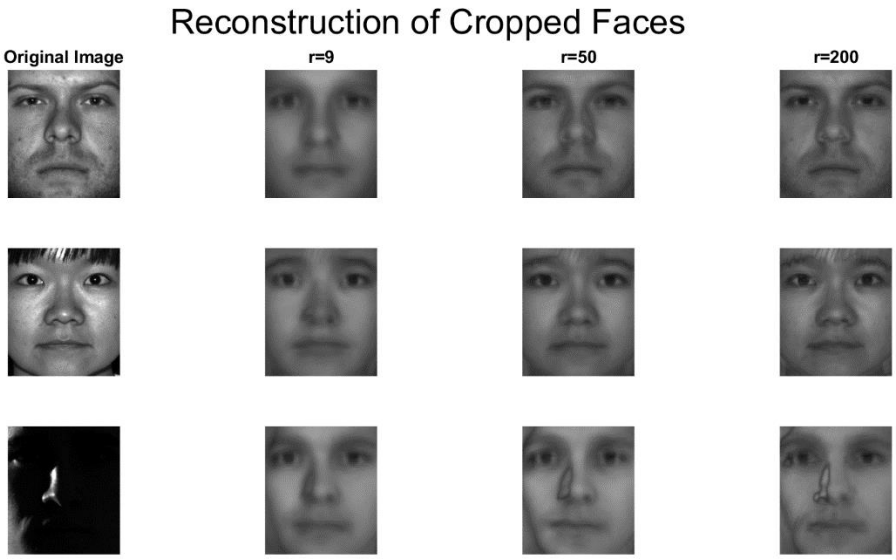

Figure 4: With nine modes, the reconstructed images seem blurry. Fifty modes sharpens the features and is probably good enough to recognize the individual. At two hundred modes, fine details such as whiskers are discernible. The reconstruction of the shadowed image is nothing more than a distorted version of the average face.

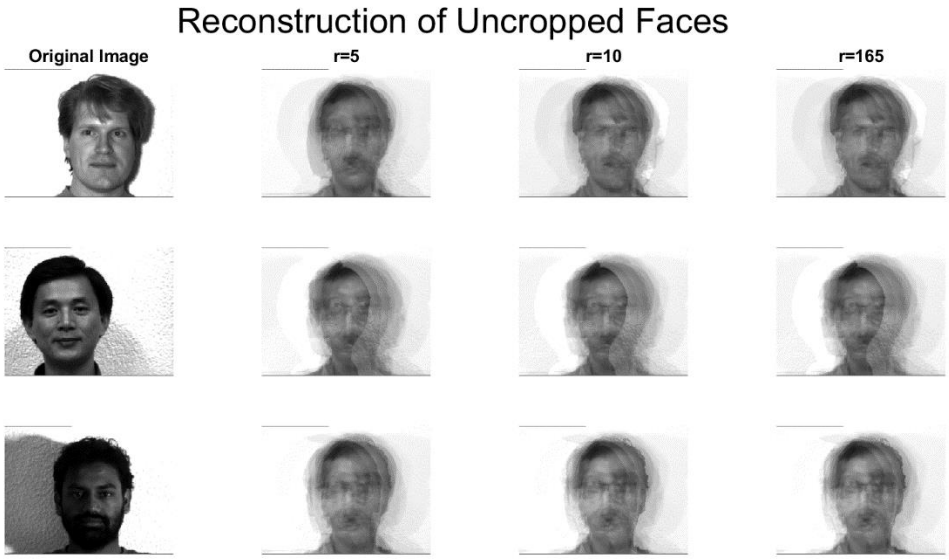

Figure 5: Though ten modes capture virtually 100% of the variance, the reconstructed images are unrecognizable even when all of the modes are used. In fact, there is little to be gained by using more than five modes. This illustrates the importance of removing unwanted features from the raw data before transforming it.

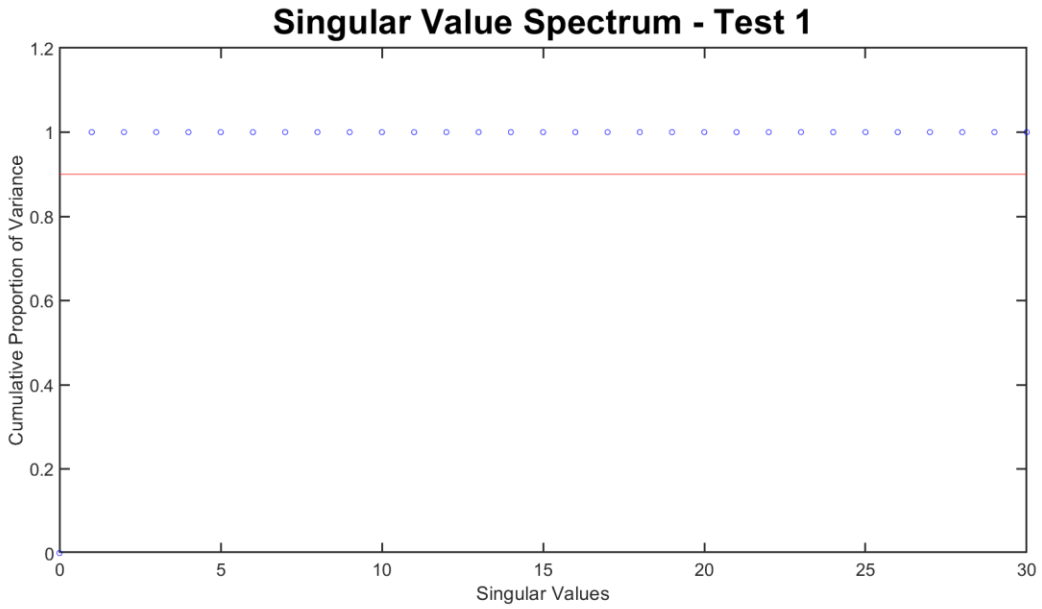

Figure 6: All of the variance in the frequency spectrograms can be captured by projecting it onto a single mode, which drastically reduces the dimension of the data set, realizing tremendous savings in computational time compared to classifying with the raw data.

Figure 7: Each trial was run with five replicates and converged to the same classification set. Buddy Guy was always identified correctly, but note that mistakes were always credited to him as well.

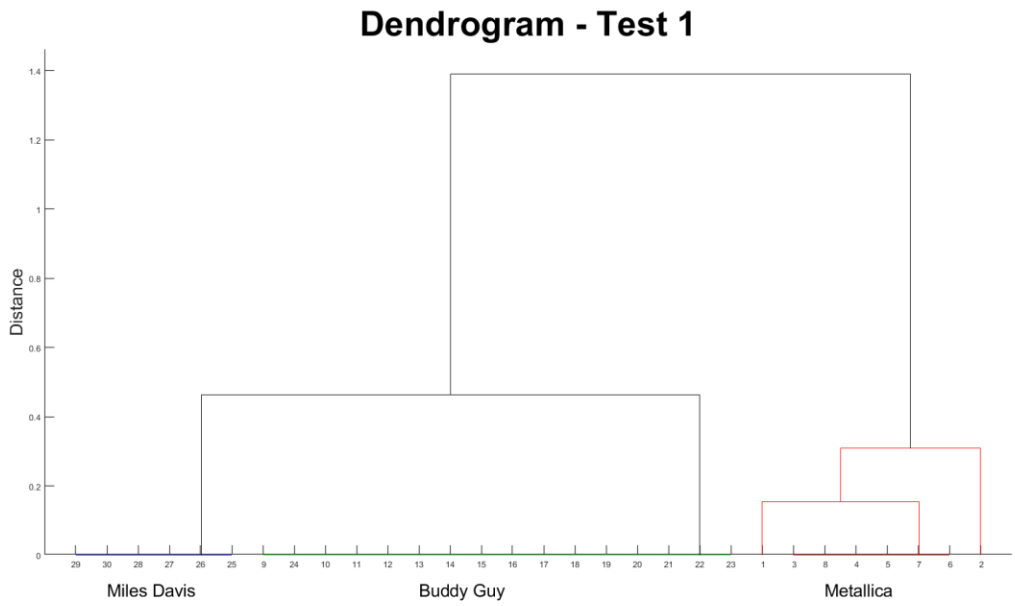

Figure 8: The agglomerative hierarchical clustering shown in the dendrogram gives the exact same clustering of tracks in Test 1 as the *k*-means algorithm. Note the large distances between clusters compared to the distances within clusters.

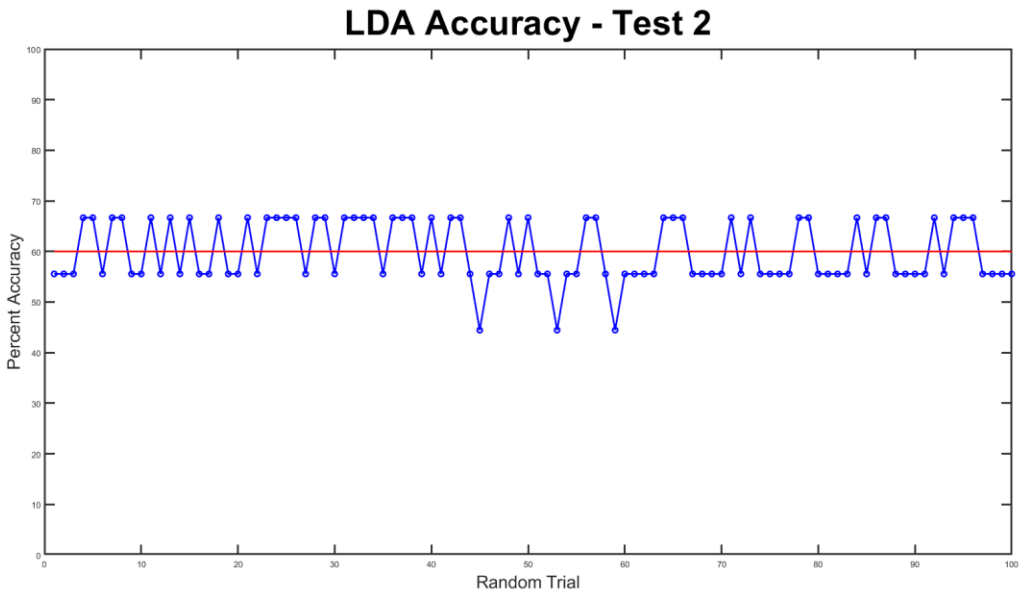

Figure 9: A time-series of the accuracy rate of 100 replications of a linear discriminant analysis for the artist within genre test. An average of 60% accuracy was achieved in classifying songs by Herbie Hancock, John Coltrane, and Miles Davis. There appear to be hard upper and lower bounds.

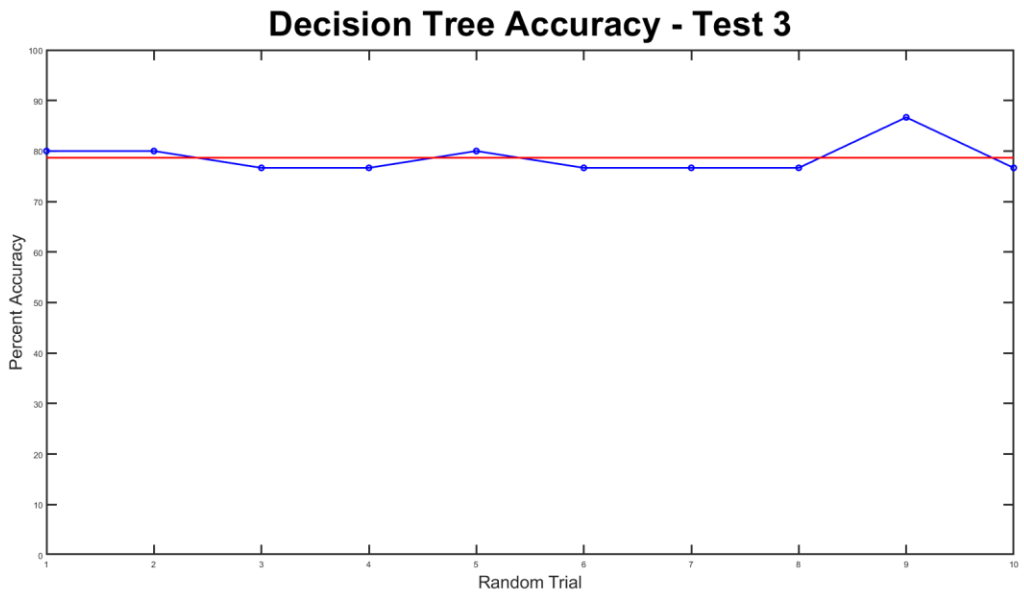

Figure 10: The divisive hierarchical decision tree algorithm has an average accuracy rate of almost 80% when classifying a diverse array of songs from blues, jazz, and hard rock. In addition to its staggering degree of accuracy, the method also appears to be very consistent.

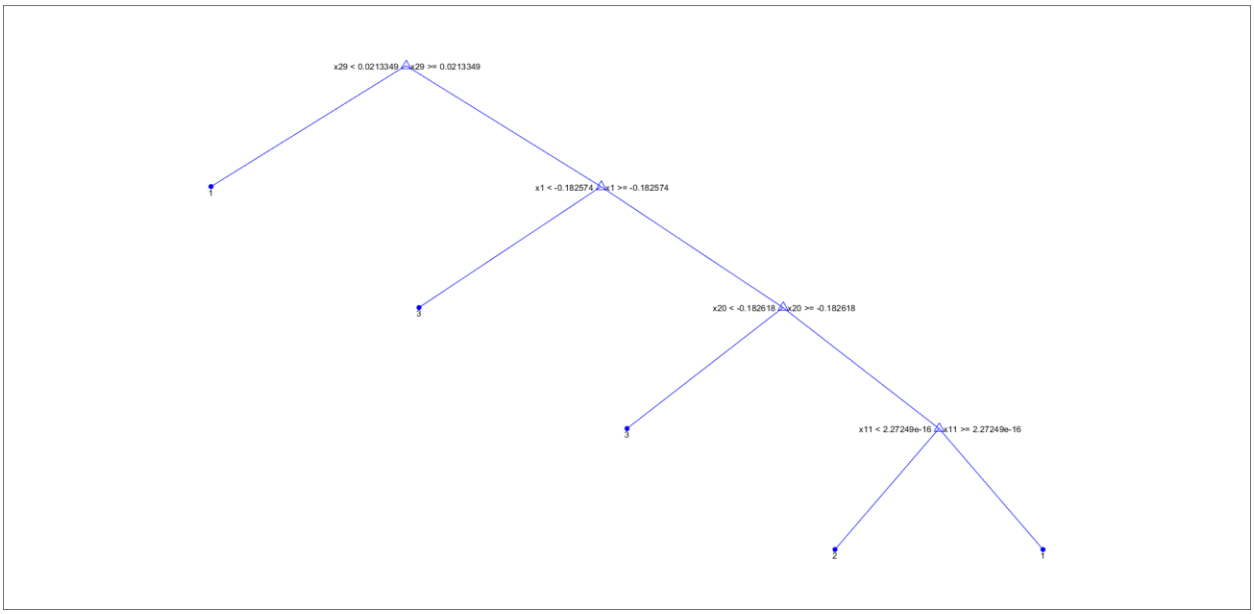

Figure 11: A four-stage decision tree for Test 3 based on the entire matrix of components from the singular value decomposition of the spectrograms.

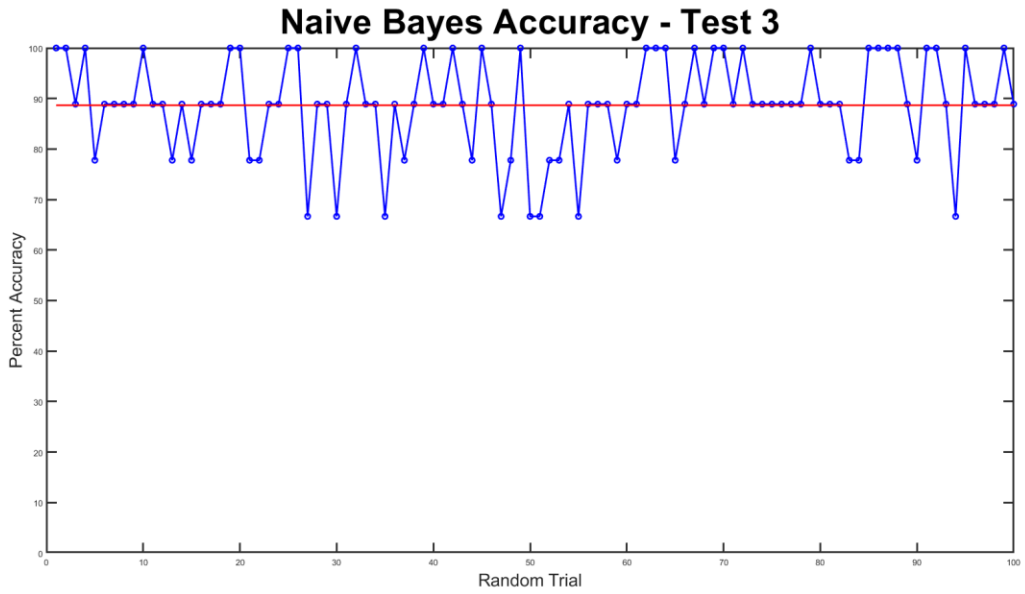

Figure 12: Score one for the Bayesians. The algorithm based on his posterior probability formula consistently outperformed the competition. The accuracy rate on Test 3 was close to 90%. It was not uncommon for the algorithm to correctly classify all of the songs to the correct genre.

V. Summary and Conclusions

One of the observations that I made during this process is that there really is no telling ahead of time which algorithm will work best. The ultimate success or failure of the algorithm has a lot to do with the samples that are chosen. The feature space that is used is also a huge determinant in how accurate the algorithms are. In the case of the music classifications, representing the data in the frequency spectrum realized huge gains in both accuracy and computational efficiency, which illustrates the importance of data mining in the classification process. Furthermore, multiple classification algorithms should be compared by feeding them different feature spaces. It is often the case that huge savings in computational time can be attained with minimal sacrifices in accuracy by simply reducing the rank of the feature space. If possible, supervised learning algorithms should be employed over unsupervised algorithms, but in the real world, unsupervised may be the only choice. As is often the case with statistics, sometimes we will never know whether or not our decisions are correct. All we can do is hope to maximize the probability of success. Rigorous mathematical models employed with conscientious technique go hand-in-hand towards achieving that goal.

A. Alternative Classification Algorithms

Linear discrimination analysis works fairly well, but it assumes that a polynomial function exists that adequately separates the clusters. *Support vector machine* analysis is an extension of this theory that attempts to project the data into higher dimensions, which may reveal a superior division boundary. If the data are expressed as $X = \langle x_1, x_2, .., x_n \rangle$, projections into higher dimensions may be accomplished by adding components of the form $\Phi(x_1, x_2, \ldots, x_n)$. The discrimination analysis can now be performed on the set of all $Z = \langle x_1, x_2, .., x_n, \Phi_1, \Phi_2, \ldots \rangle$. Graphical interpretations of *X* may give insight as to which functions Φ_i may be useful. If the original data may be separated by a hyper-spherical boundary, for instance, then $\Phi_1 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$ will likely give a clear separation of the transformed data. The calculus of SVM analysis seeks to maximize the buffer between the clusters and the decision boundary. A limitation of this classification algorithm in MATLAB is that it only allows for classification into two clusters, so it was not an option for the music genre classification tests of this investigation.

An unsupervised algorithm that is frequently employed is the *expectation maximization*. This method assumes that the data are random samples from distinct populations with known probability distributions. The distribution parameters are estimated using the regression technique of maximum-likelihood estimators. The most common distribution to assume is Gaussian, primarily because linear combinations of Gaussian variables are also Gaussian, which simplifies the math. In my humble opinion, one of the most common abuses of statistical theory is fitting Gaussians to data that are not normally distributed. The fit is often inappropriately imposed on discrete data and data with gross outliers. A normal probability plot was used to assess the fit of the mode components to a Gaussian model. Most failed badly, possibly due to the size of the components, which were small and identical to over thirteen places past the decimal. MATLAB spit out three distributions that were identical when the Gaussian mixture model was errantly imposed, which implies that all samples were from the same distribution.

The *k*-nearest neighbors algorithm is a supervised learning technique that is based on a distance metric. The distance between an observation from the cross-validation set and all observations in the labeled training set are calculated. The *k* smallest distances each contribute a vote as to which group the new observation should be classified into. When applied to the music genre classification tests, the accuracy rate for this algorithm was too low to be considered informative.

B. MATLAB Codes

The code for exploring the Yale Face Database is given first. For the music genre classification tests, only the code for the first test is given, as it was simply relabeled for use on the other two.

```
% Homework 4 - Faces
% David Long
% 2.17.19

clear all; close all; clc

% Cropped faces

totalNcrop = 0;
mcrop = 192;
ncrop = 168;
numPeopleCrop = 38;
totalSumCrop = zeros(mcrop*ncrop, 1);
allFacesCrop = [];




% Load data

for j=1:(numPeopleCrop + 1)
    if (j ~= 14)
        N = size(dir([strcat('CroppedYale/yaleB', num2str(j,'%02d'))
'/*.pgm']), 1);
        totalNcrop = totalNcrop + N;
        currentStub = strcat('CroppedYale/yaleB', num2str(j,'%02d'),
'/yaleB', num2str(j,'%02d'), '_');
        [currentSum, currentFaces] = faceImport(N, mcrop, ncrop,
currentStub);
        totalSumCrop = totalSumCrop + currentSum;
        allFacesCrop = [allFacesCrop, currentFaces];
    end
end




% figure(1)
% for j=1:size(allFacesCrop, 2)
%     imshow(uint8(reshape(allFacesCrop(:,j), mcrop, ncrop)));
% end
```

```
% The average face

avgFaceCrop = totalSumCrop/(totalNcrop);




% Center faces

% figure(3)
allFacesCenteredCrop = zeros(size(allFacesCrop, 1), size(allFacesCrop, 2));

for j=1:size(allFacesCrop, 2)
    allFacesCenteredCrop(:,j) = allFacesCrop(:,j) - avgFaceCrop;
%     imshow(uint8(reshape(allFacesCenteredCrop(:,j), mcrop, ncrop)))
end




% SVD

[ucrop, scrop, vcrop] = svd(allFacesCenteredCrop, 'econ');
numPointsCrop = size(allFacesCenteredCrop, 1);
CY = (1/(numPointsCrop-1))*diag(scrop).^2;
contributionsCrop = CY/sum(CY);
cumContributionsCrop = [0; cumsum(contributionsCrop)];




%%
% Uncropped faces

m = 243;
n = 320;
numPeople = 15;
totalSum = zeros(m*n, 1);
allFaces = [];




% Load data

expressions = { "centerlight", "glasses", "happy", "leftlight",...
                "noglasses", "normal", "rightlight", "sad", "sleepy",...
                "surprised", "wink" };
N = length(expressions);
totalN = N*numPeople;

for j=1:numPeople
```

```
    for jj=1:N

        currentStub = strcat('UncroppedYale/subject', num2str(jj,'%02d'),
'.', expressions{jj});
        currentFile = double(imread(currentStub));
        r = reshape(currentFile, m*n, 1);
        totalSum = totalSum + r;
        allFaces = [allFaces, r];

    end

end




% figure(7)
% for j=1:(numPeople*N)
%     imshow(uint8(reshape(allFaces(:,j), m, n)));
% end




% The average face

avgFace = totalSum/(totalN);




% Center faces

% figure(9)
allFacesCentered = zeros(size(allFaces, 1), size(allFaces, 2));

for j=1:size(allFaces, 2)
    allFacesCentered(:,j) = allFaces(:,j) - avgFace;
%     imshow(uint8(reshape(allFacesCentered(:,j), m, n)))
end




% SVD

[u, s, v] = svd(allFacesCentered, 'econ');
numPoints = size(allFacesCentered, 1);
CY = (1/(numPoints-1))*diag(s).^2;
contributions = CY/sum(CY);
cumContributions = [0; cumsum(contributions)];
```

```
%%
% The average face

figure(2)

subplot(1, 2, 1)
imshow(uint8(reshape(avgFaceCrop, mcrop, ncrop)));
title('Average of Cropped Faces', 'Fontsize', 40);

subplot(1, 2, 2)
imshow(uint8(reshape(avgFace, m, n)));
title('Average of Uncropped Faces', 'Fontsize', 40);




%%
% Singular value spectrums

figure(4)

subplot(1, 2, 1)
plot(0:50, cumContributionsCrop(1:51), 'bo'), hold on
plot([0 50], [0.9 0.9], 'r')
xlabel('Singular Values'), ylabel('Cumulative Proportion of Variance')
set(gca, 'Fontsize', 20, 'Linewidth', 2, 'Ylim', [0 1.2])
title('Cropped Faces', 'Fontsize', 20)

subplot(1, 2, 2)
plot(0:50, cumContributions(1:51), 'bo'), hold on
plot([0 50], [0.9 0.9], 'r')
xlabel('Singular Values'), ylabel('Cumulative Proportion of Variance')
set(gca, 'Fontsize', 20, 'Linewidth', 2, 'Ylim', [0 1.2])
title('Uncropped Faces', 'Fontsize', 20)

sgtitle('Singular Value Spectrums', 'Fontsize', 40)




%%
% Visualize significant features of the SVD

figure(5)
nbcol = size(colormap(gray), 1);
plotNumberCrop = 1;
plotNumber = 1;

for j=1:3
    for jj=1:3
        subplot(3, 6, (j-1)*6 + jj)
        imshow(uint8(reshape(wcodemat(ucrop(:,plotNumberCrop), nbcol), mcrop,
ncrop))) % Rescale colormap
```

```
        colormap(hot)
        plotNumberCrop = plotNumberCrop + 1;
        if ((j==1)&&(jj==2))
            title('Cropped', 'Fontsize', 30)
        end

        subplot(3, 6, (j-1)*6 + jj + 3)
        imshow(uint8(reshape(wcodemat(u(:,plotNumber), nbcol), m, n)))
        colormap(hot)
        plotNumber = plotNumber + 1;
        if ((j==1)&&(jj==2))
            title('Uncropped', 'Fontsize', 30)
        end
    end

    for jj=1:3

    end
end
sgtitle('The First 9 Modes of the SVD', 'Fontsize', 40);




%%
% Reconstruction of cropped faces from truncated basis

figure(6)
plotNumberCrop = 1;
plotNumber = 1;

for jj=1:3

    imageLocation = [1, 257, 2000];

    subplot(3, 4, (jj-1)*4 + 1)
    imshow(uint8(reshape(allFacesCrop(:,imageLocation(jj)), mcrop, ncrop)))
    if (jj==1)
        title('Original Image', 'Fontsize', 20)
    end

    subplot(3, 4, (jj-1)*4 + 2)
    Y = wcodemat(ucrop(:,1:9)*scrop(1:9,1:9)*vcrop(imageLocation(jj),1:9)',
nbcol) + avgFaceCrop;
    imshow(uint8(reshape(Y, mcrop, ncrop)))
    if (jj==1)
        title(strcat('r=', num2str(9)), 'Fontsize', 20)
    end

    subplot(3, 4, (jj-1)*4 + 3)
    Y =
wcodemat(ucrop(:,1:50)*scrop(1:50,1:50)*vcrop(imageLocation(jj),1:50)',
nbcol) + avgFaceCrop;
    imshow(uint8(reshape(Y, mcrop, ncrop)))
```

```
    if (jj==1)
        title(strcat('r=', num2str(50)), 'Fontsize', 20)
    end

    subplot(3, 4, (jj-1)*4 + 4)
    Y =
wcodemat(ucrop(:,1:200)*scrop(1:200,1:200)*vcrop(imageLocation(jj),1:200)',
nbcol) + avgFaceCrop;
    imshow(uint8(reshape(Y, mcrop, ncrop)))
    if (jj==1)
        title(strcat('r=', num2str(200)), 'Fontsize', 20)
    end

end

sgtitle('Reconstruction of Cropped Faces', 'Fontsize', 40)




%%
% Reconstruction of uncropped faces from truncated basis

figure(7)

for jj=1:3

    imageLocation = [1, 50, 150];

    subplot(3, 4, (jj-1)*4 + 1)
    imshow(uint8(reshape(allFaces(:,imageLocation(jj)), m, n)))
    if (jj==1)
        title('Original Image', 'Fontsize', 20)
    end

    subplot(3, 4, (jj-1)*4 + 2)
    Y = wcodemat(u(:,1:5)*s(1:5,1:5)*v(imageLocation(jj),1:5)', nbcol) +
avgFace;
    imshow(uint8(reshape(Y, m, n)))
    if (jj==1)
        title(strcat('r=', num2str(5)), 'Fontsize', 20)
    end

    subplot(3, 4, (jj-1)*4 + 3)
    Y = wcodemat(u(:,1:10)*s(1:10,1:10)*v(imageLocation(jj),1:10)', nbcol) +
avgFace;
    imshow(uint8(reshape(Y, m, n)))
    if (jj==1)
        title(strcat('r=', num2str(10)), 'Fontsize', 20)
    end

    subplot(3, 4, (jj-1)*4 + 4)
    Y = wcodemat(u(:,1:165)*s(1:165,1:165)*v(imageLocation(jj),1:165)',
nbcol) + avgFace;
```

```
    imshow(uint8(reshape(Y, m, n)))
    if (jj==1)
        title(strcat('r=', num2str(165)), 'Fontsize', 20)
    end

end

sgtitle('Reconstruction of Uncropped Faces', 'Fontsize', 40)
```

```
% Homework 4 - Test 1 Data Import
% David Long
% 2.23.19

clear all; close all; clc

for j=1:length(database)
    [track{j}, fs(j)] = audioread(strcat(database{j}(1,1), "/",
database{j}(1,2), ".mp3"));
    track{j} = sum(track{j}, 2)/size(track{j}, 2);
    sample{j} = track{j}(60*fs(j) : 65*fs(j));
end

save('test1database', 'database')
save('test1samples', 'sample')
save('test1sampleRates', 'fs')
```

```
% Homework 4 - Test 1 Spectrograms
% David Long
% 2.23.19

clear all; close all; clc

load('test1database.mat')
load('test1samples.mat')
load('test1sampleRates.mat')
```

```
%%
% Playback

% for j=1:length(database)
%     players{j} = audioplayer(sample{j}, fs(j));
%     playblocking(players{j})
% end
```

```
%%
% fft transform

for j=1:length(sample)
    sampleSpecs{j} = spectrogram(sample{j});
end




%%
% Resample

% for j=1:length(sampleSpecs)
%     currentSpec = sampleSpecs{j};
%     if (mod(size(currentSpec, 1), 2) == 1)
%         currentSpec = currentSpec(1:end-1,:);
%     end
%     sampleSpecs{j} = currentSpec;
% end




%%
% Export

save('E:/dxlong/Projects/tempMatlab/test1spectrograms', 'sampleSpecs', '-
v7.3');




% Homework 4 - Test 1 Classification
% David Long
% 2.23.19

clear all; close all; clc

load('test1database.mat')
load('E:/dxlong/Projects/tempMatlab/test1spectrograms')
X = [];
[m, n] = size(sampleSpecs{1});

for j=1:length(sampleSpecs)
    X = [X, reshape(abs(sampleSpecs{1}), m*n, 1)];
end
```

```
%%
% SVD

[u, s, v] = svd(X - mean(X(:)), 'econ');
numPoints = size(X, 1);
CY = (1/(numPoints-1))*diag(s).^2;
contributions = CY/sum(CY);
cumContributions = [0; cumsum(contributions)];



%%
% Singular value spectrums

figure(1)
plot(0:30, cumContributions, 'bo'), hold on
plot([0 30], [0.9 0.9], 'r')
xlabel('Singular Values'), ylabel('Cumulative Proportion of Variance')
set(gca, 'Fontsize', 20, 'Linewidth', 2, 'Ylim', [0 1.2])
title('Singular Value Spectrum - Test 1', 'Fontsize', 40)



%%
% kmeans

figure(2)

for j=1:4
    subplot(2, 2, j)
    [kmeansLabels, ~] = kmeans(v(:,1), 3, 'Replicates', 5);
    bar(kmeansLabels), hold on
    plot([10.5 10.5], [0 3], 'r')
    plot([20.5 20.5], [0 3], 'r')
    set(gca, 'Linewidth', 2, 'Ytick', [0 1 2 3])
    ylabel('Cluster')
    text(2.2, -0.3, 'Metallica', 'Fontsize', 20)
    text(12.2, -0.3, 'Buddy Guy', 'Fontsize', 20)
    text(22.2, -0.3, 'Miles Davis', 'Fontsize', 20)
    title(strcat('Trial ', num2str(j)), 'Fontsize', 20)
end

sgtitle('k-Means Clustering - Test 1', 'Fontsize', 40)



%%
% Dendrogram

figure(3)

vDistances = pdist(v(:,1), 'mahalanobis');
z = linkage(vDistances);
threshold = 0.4;
```

```
[h, t, o] = dendrogram(z, 30, 'ColorThreshold', threshold);
set(gca, 'Linewidth', 1)
ylabel('Distance', 'Fontsize', 20)
text(25, -0.1, 'Metallica', 'Fontsize', 20)
text(12, -0.1, 'Buddy Guy', 'Fontsize', 20)
text(2, -0.1, 'Miles Davis', 'Fontsize', 20)
title('Dendrogram - Test 1', 'Fontsize', 40)




%%
% Linear Discriminant Analysis

labels = [ones(7,1); 2*ones(7,1); 3*ones(7,1)];
truthLabels = [ones(3,1); 2*ones(3,1); 3*ones(3,1)];
accuracy = [];

for j=1:100

    % random selection
    r1 = randperm(10); r2 = randperm(10); r3 = randperm(10);
    ind1 = r1(1:7); ind2 = r2(1:7) + 10; ind3 = r3(1:7) + 20;
    ind1t = r1(8:10); ind2t = r2(8:10) + 10; ind3t = r3(8:10) + 20;
    vtrain = [v(ind1,1); v(ind2,1); v(ind3,1)];
    vtest = [v(ind1t,1); v(ind2t,1); v(ind3t,1)];
    classifyLDA = classify(vtest, vtrain, labels);

    % Error count
    count = 0;
    for jj=1:length(vtest)
        if ((classifyLDA(jj) - truthLabels(jj)) ~= 0)
            count = count + 1;
        end
    end

    accuracy(j) = (length(vtest) - count)/9*100;

end

% plot accuracy
figure(5)
plot(accuracy, 'bo-', 'Linewidth', 2), hold on
plot([1 100], [mean(accuracy) mean(accuracy)], 'r', 'Linewidth', 2)
xlabel('Random Trial', 'Fontsize', 20)
ylabel('Percent Accuracy', 'Fontsize', 20)
set(gca, 'Linewidth', 2, 'Ylim', [0 100])
title('LDA Accuracy - Test 1', 'Fontsize', 40)
```

```
%%
% Decision tree

accuracy = [];

labels = [ones(10,1); 2*ones(10,1); 3*ones(10,1)];
for j=1:length(labels)
    treeLabels{j} = num2str(labels(j));
end

for j=1:10
    decisionTree = fitctree(v, treeLabels', 'MaxNumSplits', 10, 'CrossVal',
'on');
    classError = kfoldLoss(decisionTree);
    accuracy(j) = (1 - classError)*100;
end

view(decisionTree.Trained{1}, 'Mode', 'graph');

% plot accuracy
figure(6)
plot(accuracy, 'bo-', 'Linewidth', 2), hold on
plot([1 10], [mean(accuracy) mean(accuracy)], 'r', 'Linewidth', 2)
xlabel('Random Trial', 'Fontsize', 20)
ylabel('Percent Accuracy', 'Fontsize', 20)
set(gca, 'Linewidth', 2, 'Ylim', [0 100])
title('Decision Tree Accuracy - Test 1', 'Fontsize', 40)




%%
% Gaussian Mixture Model

figure(4)
normplot(v(1:10,1))
set(gca, 'Fontsize', 20)
title('Normal Probability Plot - Test 1', 'Fontsize', 40)
gmModel = fitgmdist(v(:,1), 3, 'RegularizationValue', 1e-10)
aic = gmModel.AIC;




%%
% knn-search

labels = [ones(7,1); 2*ones(7,1); 3*ones(7,1)];
truthLabels = [ones(3,1); 2*ones(3,1); 3*ones(3,1)];
accuracy = [];

for j=1
```

```
    % random selection
    r1 = randperm(10); r2 = randperm(10); r3 = randperm(10);
    ind1 = r1(1:7); ind2 = r2(1:7) + 10; ind3 = r3(1:7) + 20;
    ind1t = r1(8:10); ind2t = r2(8:10) + 10; ind3t = r3(8:10) + 20;
    knntrain = [v(ind1,1); v(ind2,1); v(ind3,1)];
    knntest = [v(ind1t,1); v(ind2t,1); v(ind3t,1)];
    [ind, D] = knnsearch(knntrain, knntest, 'k', 5);
    knnClass = (sum(ind, 2)/5)
    count = 0;
    for j=1:3
        if (knnClass(j)>10)
            count = count + 1;
        end
        if ((knnClass(j+3)<=10) || (knnClass(j+3)>20))
            count = count + 1;
        end
        if (knnClass(j+6)<=20)
            count = count + 1;
        end
    end
    disp(count)

end


%%
% Naive Bayes

labels = [ones(7,1); 2*ones(7,1); 3*ones(7,1)];
truthLabels = [ones(3,1); 2*ones(3,1); 3*ones(3,1)];
accuracy = [];

for j=1:100

    % random selection
    r1 = randperm(10); r2 = randperm(10); r3 = randperm(10);
    ind1 = r1(1:7); ind2 = r2(1:7) + 10; ind3 = r3(1:7) + 20;
    ind1t = r1(8:10); ind2t = r2(8:10) + 10; ind3t = r3(8:10) + 20;
    nbtrain = [v(ind1,1); v(ind2,1); v(ind3,1)];
    nbtest = [v(ind1t,1); v(ind2t,1); v(ind3t,1)];
    nbModel = fitcnb(nbtrain, labels);
    nbPredict = predict(nbModel, nbtest);

    % Error count
    count = 0;
    for jj=1:length(nbPredict)
        if ((nbPredict(jj) - truthLabels(jj)) ~= 0)
            count = count + 1;
        end
    end

    nbaccuracy(j) = (length(nbPredict) - count)/9*100;

end
```

```
% plot accuracy
figure(7)
plot(nbaccuracy, 'bo-', 'Linewidth', 2), hold on
plot([1 100], [mean(nbaccuracy) mean(nbaccuracy)], 'r', 'Linewidth', 2)
xlabel('Random Trial', 'Fontsize', 20)
ylabel('Percent Accuracy', 'Fontsize', 20)
set(gca, 'Linewidth', 2, 'Ylim', [0 100])
title('Naive Bayes Accuracy - Test 1', 'Fontsize', 40)
```

C. Track Catalogs

```
database = {        ["Metallica",       "Blackened"],...
                    ["Metallica",       "Disposable Heroes"],...
                    ["Metallica",       "Hardwired"],...
                    ["Metallica",       "Hit The Lights"],...
                    ["Metallica",       "Leper Messiah"],...
                    ["Metallica",       "Ride The Lightning"],...
                    ["Metallica",       "Sad But True"],...
                    ["Metallica",       "Seek & Destroy"],...
                    ["Metallica",       "Spit Out The Bone"],...
                    ["Metallica",       "The Shortest Straw"],...
                    ["Buddy Guy",       "Champagne and Reefer"],...
                    ["Buddy Guy",       "Damn Right, I've Got The Blues"],...
                    ["Buddy Guy",       "First Time I Met The Blues"],...
                    ["Buddy Guy",       "I Gotta Try You Girl"],...
                    ["Buddy Guy",       "Just Playing My Axe"],...
                    ["Buddy Guy",       "Tramp"],...
                    ["Buddy Guy",       "Leave My Girl Alone"],...
                    ["Buddy Guy",       "She's Nineteen Years Old"],...
                    ["Buddy Guy",       "Someone Else Is Steppin' In"],...
                    ["Buddy Guy",       "Texas Flood"],...
                    ["Miles Davis",     "Airegin"],...
                    ["Miles Davis",     "Duke"],...
                    ["Miles Davis",     "E.S.P."],...
                    ["Miles Davis",     "Filles de Kilimanjaro"],...
                    ["Miles Davis",     "Miles Runs The Voodoo Down"],...
                    ["Miles Davis",     "Move"],...
                    ["Miles Davis",     "So What"],...
                    ["Miles Davis",     "Solea"],...
                    ["Miles Davis",     "Somethin' Else"],...
                    ["Miles Davis",     "Well You Needn't"]...
    };
```

```
database = {        ["Herbie Hancock",      "Both Sides Now"],...
                    ["Herbie Hancock",      "Chameleon"],...
                    ["Herbie Hancock",      "Cotton Tail"],...
                    ["Herbie Hancock",      "D Trane"],...
                    ["Herbie Hancock",      "Dis is da Drum"],...
                    ["Herbie Hancock",      "Mojuba"],...
                    ["Herbie Hancock",      "Nefertiti"],...
                    ["Herbie Hancock",      "Summertime"],...
                    ["Herbie Hancock",      "The Sorcerer"],...
                    ["Herbie Hancock",      "Watermelon Man"],...
                    ["John Coltrane",       "Africa"],...
                    ["John Coltrane",       "Blue Train"],...
```

```
                    ["John Coltrane",       "Chasin' Another Trane (Live)"],...
                    ["John Coltrane",       "Giant Steps"],...
                    ["John Coltrane",       "Greensleeves"],...
                    ["John Coltrane",       "I Hear A Rhapsody"],...
                    ["John Coltrane",       "Locomotion"],...
                    ["John Coltrane",       "Lush Life"],...
                    ["John Coltrane",       "My Favorite Things (Live)"],...
                    ["John Coltrane",       "Naima"],...
                    ["Miles Davis",         "Airegin"],...
                    ["Miles Davis",         "Duke"],...
                    ["Miles Davis",         "E.S.P."],...
                    ["Miles Davis",         "Filles de Kilimanjaro"],...
                    ["Miles Davis",         "Miles Runs The Voodoo Down"],...
                    ["Miles Davis",         "Move"],...
                    ["Miles Davis",         "So What"],...
                    ["Miles Davis",         "Solea"],...
                    ["Miles Davis",         "Somethin' Else"],...
                    ["Miles Davis",         "Well You Needn't"]...
    };
```

```
database = {        ["Blues", "Bo Diddley", "Bo Diddley"],...
                    ["Blues", "Tommy Tucker", "Drunk"],...
                    ["Blues", "Albert King", "Get out of My Life Woman"],...
                    ["Blues", "Hound Dog Taylor", "Give Me Back My Wig"],...
                    ["Blues", "Roy Gaines & The Crusaders", "Hell of a Night"],...
                    ["Blues", "John Lee Hooker", "House Rent Boogie"],...
                    ["Blues", "Albert Collins", "The Things I Used to Do"],...
                    ["Blues", "B.B. King", "The Thrill Is Gone"],...
                    ["Blues", "Big Walter Hornton", "Trouble In Mind"],...
                    ["Blues", "Howlin' Wolf", "Worried About My Baby"],...
                    ["Hard Rock", "Jeff Beck", "Come Dancing"],...
                    ["Hard Rock", "Jimi Hendrix", "Driving South"],...
                    ["Hard Rock", "Black Sabbath", "Fairies Wear Boots"],...
                    ["Hard Rock", "AC/DC", "If You Want Blood (You've Got It)"],...
                    ["Hard Rock", "Guns N' Roses", "It's So Easy"],...
                    ["Hard Rock", "Van Halen", "Panama"],...
                    ["Hard Rock", "Led Zeppelin", "Rock & Roll"],...
                    ["Hard Rock", "Ted Nugent", "Stranglehold"],...
                    ["Hard Rock", "James Gang", "Walk Away"],...
                    ["Hard Rock", "Lynyrd Skynyrd", "Workin' For MCA"],...
                    ["Jazz", "Weather Report", "Birdland"],...
                    ["Jazz", "Coleman Hawkins", "Body And Soul"],...
                    ["Jazz", "Stan Getz & Charlie Byrd", "Desafinado"],...
                    ["Jazz", "Charlie Parker", "Just Friends"],...
                    ["Jazz", "Count Basie's Kansas City Seven", "Lester Leaps In"],...
                    ["Jazz", "Charles Mingus", "Original Faubus Fables"],...
                    ["Jazz", "Django Reinhardt", "Shine"],...
                    ["Jazz", "Sonny Rollins", "St. Thomas"],...
                    ["Jazz", "Duke Ellington", "The Mooche"],...
                    ["Jazz", "Louis Armstrong", "West End Blues"]...
    };
```

D. References

[1] Brunton & Kutz. *AMATH 582: Chapter 5: Clustering and Classification.* Department of Applied Mathematics, University of Washington, Seattle, WA 98195.