Face and Song Recognition

Avery Pong - AMATH 482 - 03.08.2019

0. Overview: Put a Face to the Song

Collections of face images and song spectrograms can both be represented as matrices that exhibit a few unique, but altogether similar features. The SVD allows for a direct mapping of each face or song in a database to orthogonal basis vectors that best describe the eigen or song-space. Machine learning algorithms can then be applied on these common features to predict what other members of that facial group or song genre might look like in the principle component feature space.

I. Introduction: Face Images and Spectrogram

Humans can identify faces almost instantaneously, reducing a multitude of biological features to a registered name and face in the mind's eye. We can also identify individual songs, along with songs that might be similar enough to be a member of the song's same genre. This computation somehow takes an almost infinitely complex matrix of information and converts it into a low dimensional, manageable form. Computers are no humans, and this report seeks to demonstrate the ways in which supervised machine learning algorithms, in conjunction with common dimensionality reduction procedures, can begin to simplify faces, classify songs, and generally categorize high dimensional data.

First, we explore the singular value decomposition (SVD) and its applications to face deconstruction and reconstruction. We proceed by generating library of different faces from people in different lighting and poses, and end by generalizing a facial feature space that can, with relatively small amounts of data, begin to identify a human face as a face. Second, we dissect spectrograms of music from various artists with the hope of classifying their composer and genre. Spectrograms are developed, and the SVD is again applied in order to find the most important modes of information—the dominant features—that are represented in different types of music. Classification algorithms are then applied to the data in the principle component space in order to discriminate songs on the basis of their major feature differences.

II. Theoretical Background: the SVD, FFT, and different Classification Algorithms

This report attempts to identify the primary features that characterize particular facial images and musical songs. The primary features in the former might include nose and lip outlines, while those of the latter might include the basic rhythm of the piece or a repeated use of low frequency bass. Faces and songs, while totally analyzable by the naked eye, are difficult to parse with computational means as naked matrices. To best render these face and song matrices, the SVD may be applied to illuminate the optimal spatial coordinate system in which the matrices can be more efficiently represented. SVD is a matrix decomposition that is guaranteed to exist for any matrix, and is comprised of the following:

1) Matrix $X \in \mathbb{C}^{mxn}$ has an singular value decomposition $X = U\Sigma V^*$, where
2) $U \in \mathbb{C}^{mxm}$ is a unitary matrix
3) $V \in \mathbb{C}^{nxn}$ is a unitary matrix
4) $\Sigma \in \mathbb{R}^{mxn}$ is a diagonal matrix with values in descending order

Overall, data matrix X is broken down into constituent transformations, where V represents a rotation, Σ represents stretching by singular values of the matrix X, and U represents a final rotation in the basis space. One can also see the decomposition as a change in basis coordinates for the data: here, U represents the standard basis vectors of the optimal operational space for the data, V represents each data entry's particular coordinate location in the U coordinate system, and Σ represents coefficient weightings for each V in U. In the context of this report, putting all the images and spectrograms into this optimal U space allows for comparison along all major features that characterize the data—for this reason, this space defined by the U orthogonal basis matrix of the SVD can be referred to as the feature space. In the case of the music classification, using the U space defined by all of the music essentially allows for discrimination along the song's most dominant features.

For the music classification portions of this analysis, short-time Fourier Transform (Gabor Transform) was used to represent each song as a spectrogram—essentially the facial image of the music—to facilitate further analysis. The Gabor transform allows for time-specific capturing of frequency content from songs, and is of the following form:

$$G[f](t,\omega) = \int_{-\infty}^{\infty} f(\tau) \cdot g(\tau - t)\, e^{-i\omega t} d\tau$$

The Gabor transform uses a sliding window to localize input data with respect to time, and thus allows for time-localized Fourier Transformation. In other words, the Gabor transform provides information about the song with respect to time: this is important because songs have rhythm, and their rhythmic features go a long way to defining which genre to which songs belong. The Gabor transform, as a Fourier Transform variant, allow gives a full picture of the frequency content per time slice: this is essential because almost all music is defined by the pitches—the frequency of notes—that are played. A Gabor window of the following form was used:

$$Sliding\ Gabor\ Window = g(t) = \exp(-10t^{10})$$

Gabor Transforms can be culled into matrices to describe the time and frequency dynamics of the song, which represents a matrix that is commonly referred to as a spectrogram. These spectrogram contain information that are, as a generalization, specific to the artist.

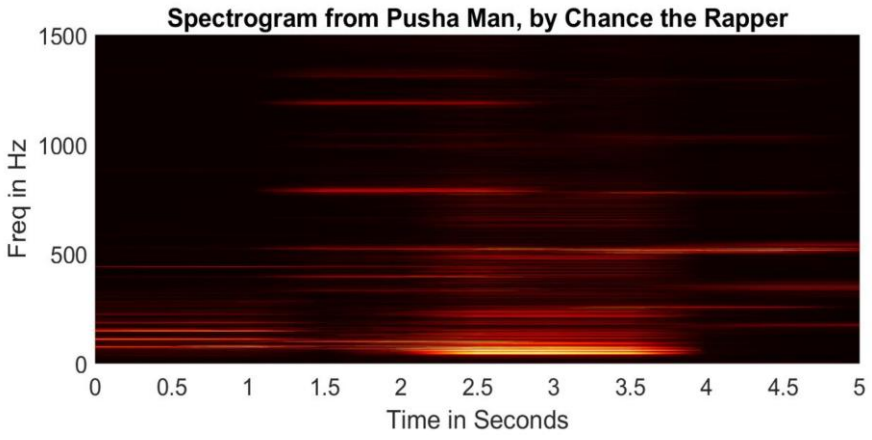


Figure 1. Spectrogram from 5s clip from Pusha Man.

The last piece of preliminary theory involves a discussion of the machine learning classification algorithms that allowed for song categorization. First, all song data for each test was mapped onto the principle component space that best allowed for their separation. This preliminary step was done by mapping separation amongst songs along the first 16 principle axes, and was performed by eye. However, this generalizable method aided in subsequent classification steps tremendously.

First, linear discriminate analysis (LDA) was implemented. Linear discriminate analysis essentially finds an optimal subspace in which different labeled data can best be distinguished. This algorithm makes use of a labeled training set, and finds the appropriate subspace on which the data can best be separated according to their labels. In essence, the LDA uses successive projections of each of the data sets to maintain each class's integrity, while best separating it from any others. The form of the two class LDA is as follows, and can be extended to higher dimensions as it was here:

$$v = \boldsymbol{arg}\max \frac{w^T(u_2 - u_1)(u_2 - u_1)^T w}{w^T \sum_{j=1}^{3} \sum_x (x - u_j)(x - u_j)^T w}$$

Project v is dependent on the inter and intra scatter between data, and quantify the variance associated between classes in the data and the difference in their means.[1] The second algorithm used was the Naïve Bayes, which classifies testing data based on prior class and predictor prior probabilities:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

Here, $P(c|x)$ is the posterior probability of a subset of data given a particular test data x and calculates the probability that c is true given x. Successive applications of this algorithm across the three classes of data used in the tests in this reports can yield discrimination between each class.

III. Algorithmic Implementation: EigenFaces

Faces of 38 different people were examined under 65 different lighting conditions. Grayscale JPG mages were collected together and resized o 50x50 matrices to reduce computational time; these smaller pictures were further reshaped to a column vector that was collected in a larger `Library` matrix that held each image considered. The `Library` file constituted, in essence, a space of facial features, which was extracted by applying the SVD to the entire matrix. This decomposition defined a U matrix that represented the optimal coordinate system in which to best see the successively important features in the face library. Here, the V matrix contained rows that, in conjunction with the diagonal Σ matrix, represent each image's scaled coordinates in the U space. Additionally, Σ describes the relative energy associated with each of the coordinate directions in U in descending order, and thus provides us with important insights as to the amount of energy is truly needed to best describe the dynamics of the system.

Data from the SVD provided the basis for two further insights. First, we could map each of the first few U modes as 50x50 matrix reconstructions to determine what the most important modes in a face actually look like—there's no surprise that the most important few modes are outlines of shadows of the head, and of prominent organ features like the nose and eyes. Second, this allowed for image reconstruction with successively more U modes, which allowed for a low dimensional reconstruction of the face. These steps were taken with both the cropped and uncropped datasets, and serve to demonstrate the limits of a computer's ability to perform meaningful data extraction and reconstruction on even slightly off-kilter (read tilted) data.

[1] Kutz and Brunton. Data-Driven Science and Engineering: ML, Dynamic Systems, and Control, 2019 (in press).

IV. Computational Results: Face Image Set

The SVD showed that for both the cropped and uncropped images, about 20% of storage was actually needed to capture 90% of the feature dynamics in the faces dataset.

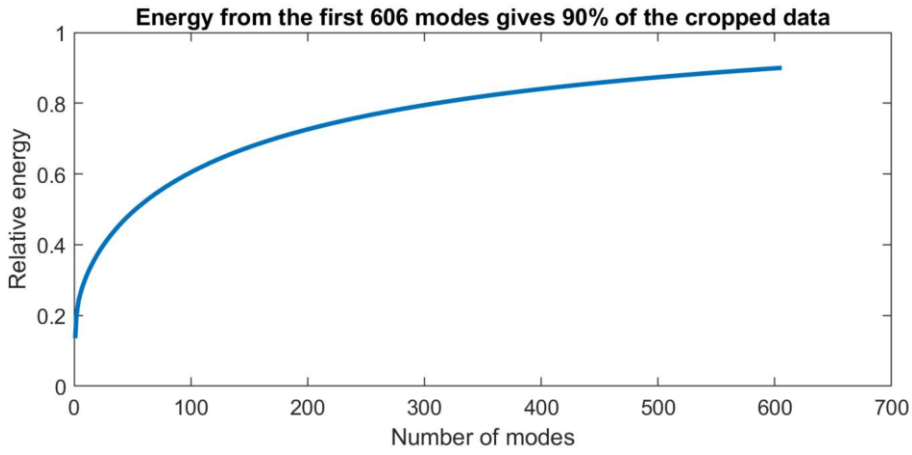


Figure 2, 606 modes of 2500 give 90% of the cropped data.

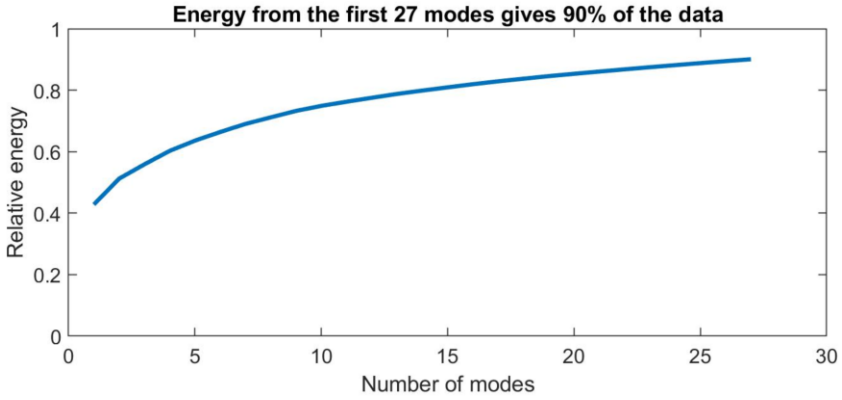


Figure 3, 27 modes of 62 give 90% of the uncropped data.

Interestingly, about 20% of the data was needed to describe 90% of the cropped data, while about 45% of the data was needed to describe 90% of the uncropped data set. This is likely due to the fact that a lot of the energy in the feature space of the uncropped data goes into the head tilts in the uncropped dataset. Because the cropped shots do not take the head tilt data into consideration, it doesn't need to store that extra data, and thus requires much less storage proportionally to the data it describes. This data importantly, illustrates that low rank approximations of the data can render images that are up to par with their full renditions.

Figure 4., First 25 Eigenface modes, cropped.

The cropped data's eigenspace primarily focuses on facial features that are prominent in the dataset. This is expected, since the only things that are present in the images are the facial structures. What's interesting is that the first mode shows the outline of the eyes, which is what humans usually identify on a face when they meet face to face. Eyes are our immediate first point of contact, and are generally how we acknowledge another's presence. It's no surprise that these prominent features dominate the eigenspace from the perspective of the computer as well. The first row, again, seems to highlight basic human facial structure, while subsequent modes seems to capture more detailed features, like nose size, lip shadows, and smirks.

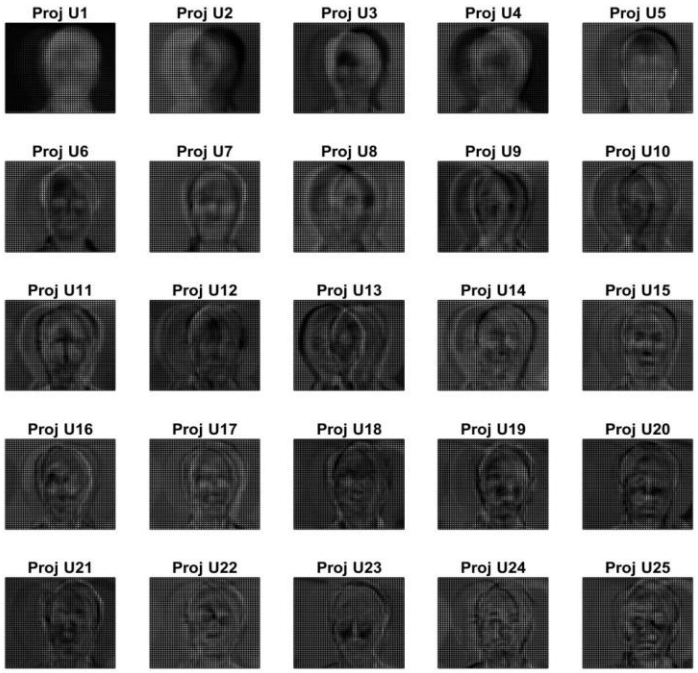


Figure 5., First 25 eigenface modes, uncropped.

Interestingly, the main eigen modes, or feature space, of the uncropped images shows outlines of the head, and only sparingly shows the details of the face. It makes sense that the primary modes in the first row would be head shape and head shadow, so that the computer, and indeed us humans, could at least identify the image as somewhat human. Something particularly compelling is that the feature space seems to distill the pictures to three different head tilts: center, left, right. There doesn't seem to be a gradient of head tilt direction, but rather three different "modes" of head position. It's very interesting to think about how this might be how humans perceive the image of other people: we don't necessarily see their overall position, we might just see that it's a head in a centered, or left/right off-center position. This would, in many ways, greatly reduce the amount of data we'd need to store about other people's position relative to ours, from an evolutionary point of view, and I think it's incredibly fascinating to think that these eigenface modes are how we might actually biologically embed million-dimensional facial and head position data to a mean dozen or so positional and facial modes.

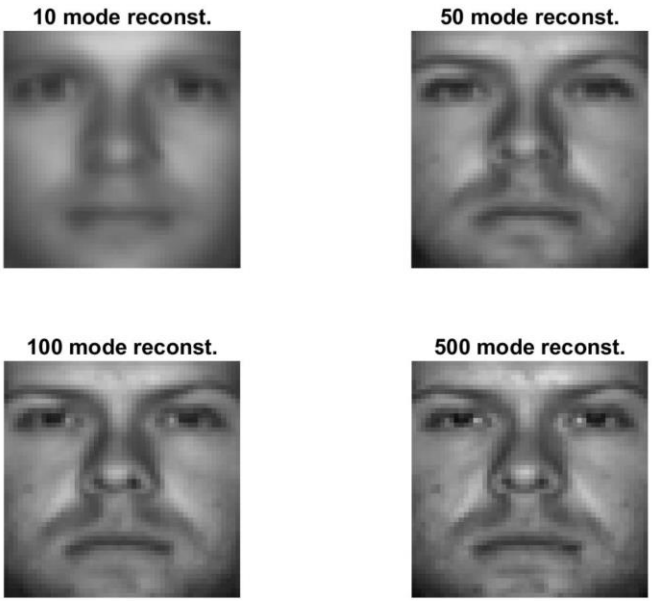


Figure 6., Low rank facial reconstructions.

Figure 6 demonstrates that low rank data from the library can, to a high degree of fidelity, reconstruct the face. A mere 50 modes can capture a truly compelling and realistic image. Interestingly, more detail like facial blemishes and pimples are rendered clearer with more modes. Maybe, in an ideal world, this means that we see people for who they truly are—not their faults, not their higher level eigenface features, and especially not their acne.

V. Algorithmic Implementation: Songs and Genres

Test 1 had the goal of making a training set of three different artists, and attempted to use this database as a basis for subsequent classification tests of other samples of their music. 60 songs from three artists were segmented to 5 second clips that were subsequently rendered as spectrograms using a Gabor transform, as described in Section II. 50 of these songs would be developed as a training set, while another 10 would be used to test different classifying algorithms under study. A few experimental precautions are noteworthy: first, each sublibrary of 60 songs was taken from at least 5 different songs to account for diversity in the artist's work; second, the first and last 10 seconds of each song were eliminated in order to account for data files lengths that might not have captured 5 seconds worth of data—this was also done to omit noiseless data that often comes at the beginning and end of songs, before they start and after they end; lastly, every other frequency reading from songs were read, in order to half computation times and prevent storage-related issues when saving the matrix for further downstream evaluation. Spectrograms were subsequently adjoined to a `Library` matrix that

contained all of the songs under consideration. For test 1, 60 clips from pieces by J.S. Bach, 60 clips from standards by Duke Ellington, and 60 clips from songs by Kendrick Lemar were used, making for a library of 180 song clips. The SVD was then applied to the `Library` matrix to find the dominant spectrogram modes–the primary musical features manifest in each artist's piece.

Next, an intermediate algorithmic step was undertaken to determine the principle axes on which the music by each artist differed most. This was done by plotting each artist's music along the first 12 principle modes, and the axes along which the artists' music showed demonstrable separation were then used for plotting and classification purposes. Three dimensional plotting of data–from test 1, 2 and 3–was performed along the three principle axes that demonstrated the best separation by eye, but seemed to work as a good, procedurally sound step for improving classifications along optimal principle axes.

VI. Computational Results: Song Set

Test 1 and 2 are presented in this results section, while test 3 is presented in Appendix C. This was done because of the high degree of procedural and experimental redundancy between test 1 and test 3, since both explored data sets from three different genres of music.

Test 1 was conducted using music by Bach (Classical, Baroque), Duke Ellington (Big band jazz), and Kendrick Lamar (rap). These musical genres sound markedly different, and many of the principle modes of the feature space demonstrated differences between the music.

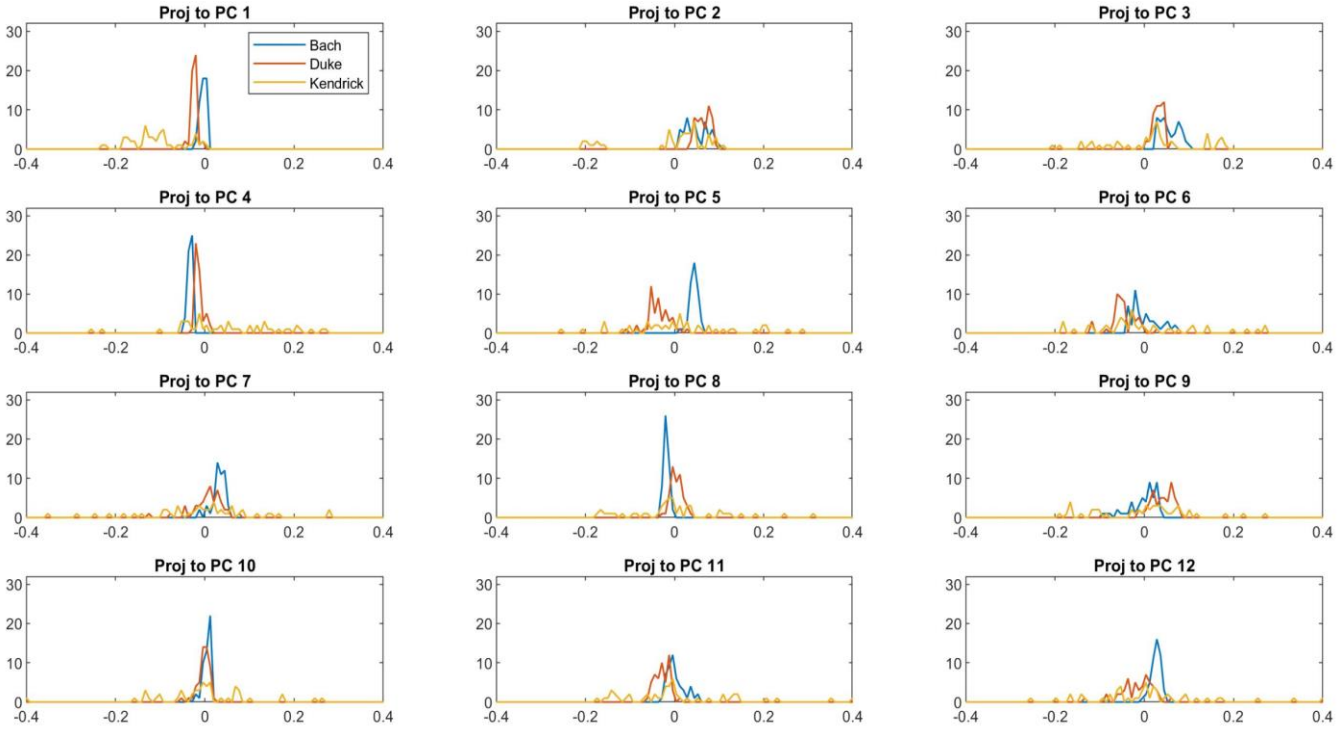


Figure 7. First 12 modes show that the music differs most on modes 1, 4, and 5.

These data were taken in order to show the principle modes on which the V matrix data describing each clip in the principle component space could best be plotted. Modes 1, 4, and 5 showed significant difference in the musical features between artists, and were thus used as the axes along which to plot data to best differential the subclasses in the dataset.

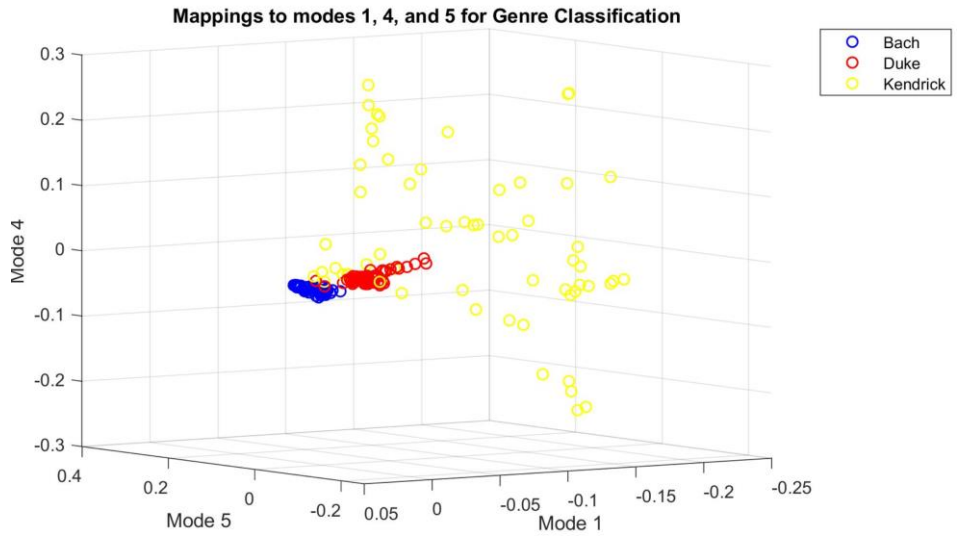


Figure 8. Mapping of Bach, Duke, and Kendrick on principle modes 1, 4, and 5.

These principle modes showed incredible separation between the different sublibraries of music. It's interesting to mention that Bach and Duke only use acoustic instruments, and should thus be limited in their frequency content. However, Kendrick is a modern rap artist that has access to synthetic music effects and computational methods that can give his music huge ranges of frequency content: the bass used in modern pop and rap is far lower in pitch than any 20th, let alone 17th century instruments, and the higher frequencies accessible from synthetic music software far exceeds the capabilities of the instruments employed by Bach and Duke Ellington. For this reason, it's not surprising that Bach and the Duke are close together, though slightly separate, with respect to the principle components that were used to plot them. Moreover, it's not surprising that Kendrick's music has a huge range of values in the feature space, since modern music innovations afford him a greater range of rhythms, instruments, and frequencies to operate with.

The music was classified using LDA and Naïve Bayes analysis, which had a 93% and 83% accuracy at classification, respectively. Here, 10 pieces by Bach, Duke, and Kendrick were tested against the training set–samples 1-10 are truly Bach, 11-20 are Duke, and 21-30 are Kendrick. Red circles show inaccuracies from the classifiers.

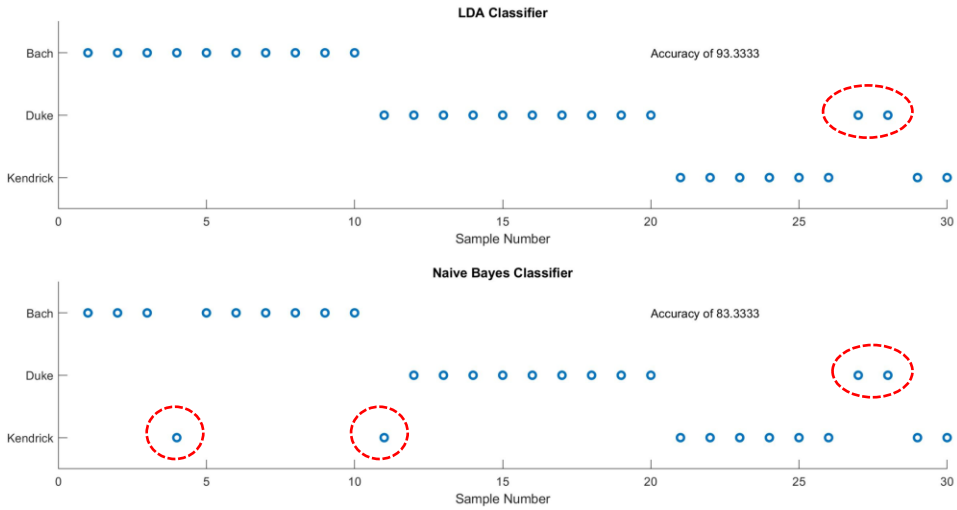


Figure 9. Classifying 10 new Bach, Duke, and Kendrick clips.

Here, the results show clearest separation between Bach and the Duke, since there's no crossover between the two. Kendrick, however, because of the diversity of his music's features, seems to muddle separation. Overall, it seemed fairly straightforward to separate music from different artists of very different genres. The same could not be said of test 2, where there was a significant amount of principle component feature overlap between rap artists under consideration.

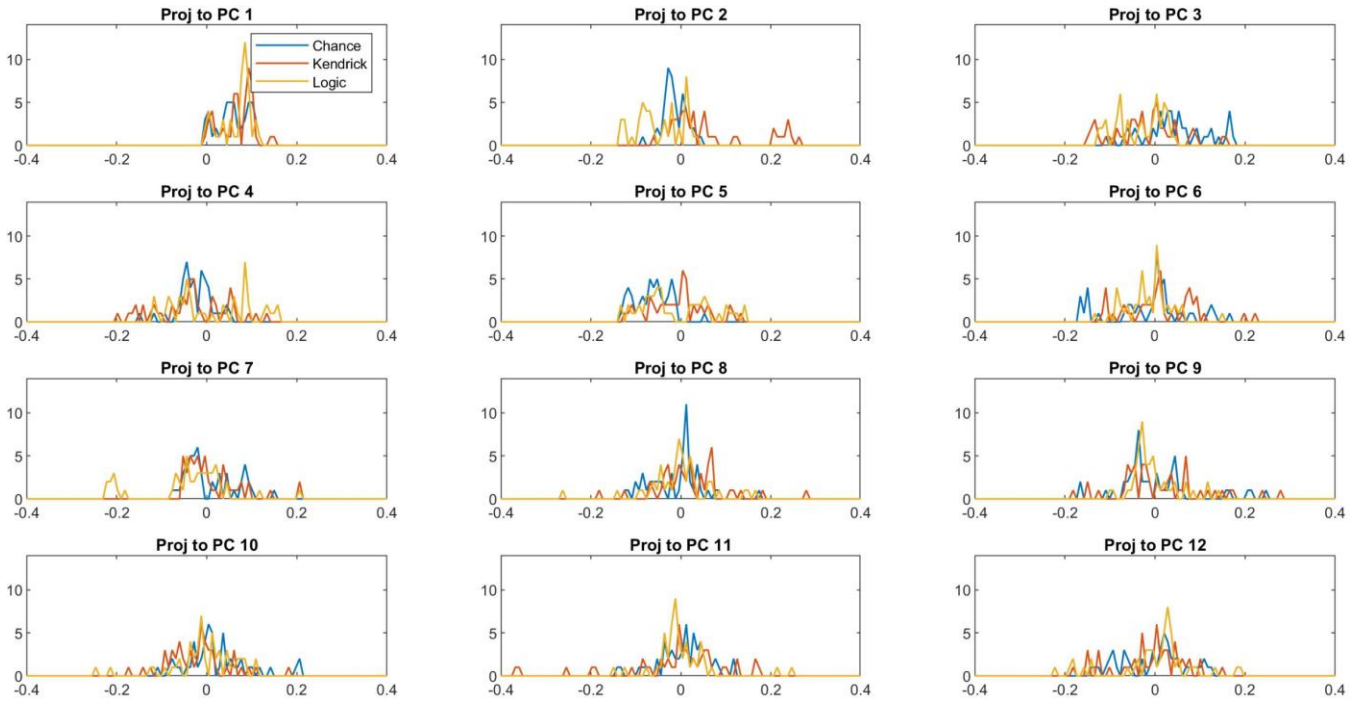


Figure 10. *First* 12 *modes show that the rappers differs most on modes* 2, 5, *and* 14 (*not shown*).

Chance the Rapper, Kendrick, and Logic all belong to the same genre–rap–and thus make use of the same tricks of the trade: low base, high frequency trills, lots of spoken word, and a very rhythmic pulse. For these reasons, it's not surprising that it is difficult to find principle modes on which they can be distinguished. In the end, modes 2, 5, and 14 were selected by eye, and showed decent separation.

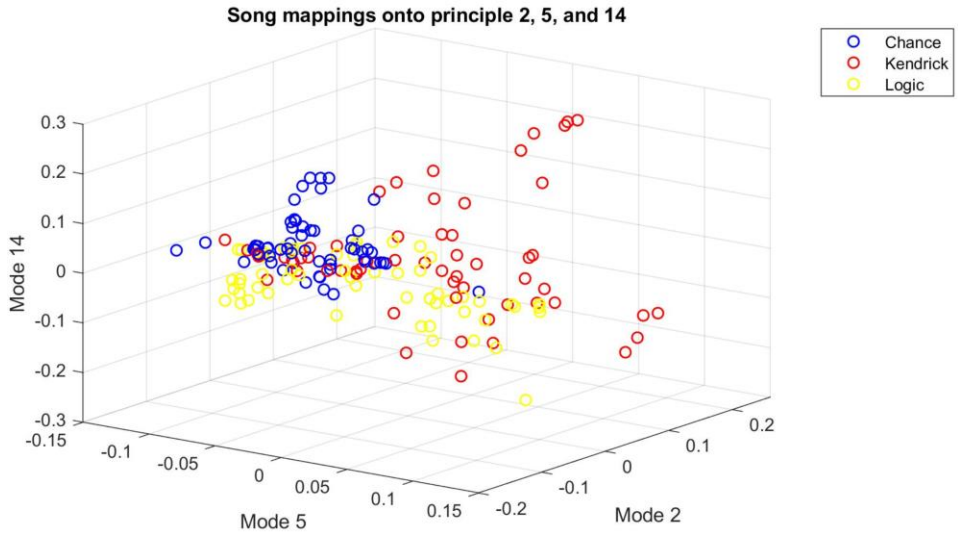


Figure 11. *Little separation along the optimal discrimatory modes s observed between rappers.*

Een plotting along the principle axes that show the most difference in the feature space cannot separate the rappers. In the subsequent classification steps, two, three, and four different modes were used in an attempt to improve the accuracy of the categorization. However, it was found that simply using the three abovementioned modes sufficed to give decent separation by the LDA and Naïve Bayes.

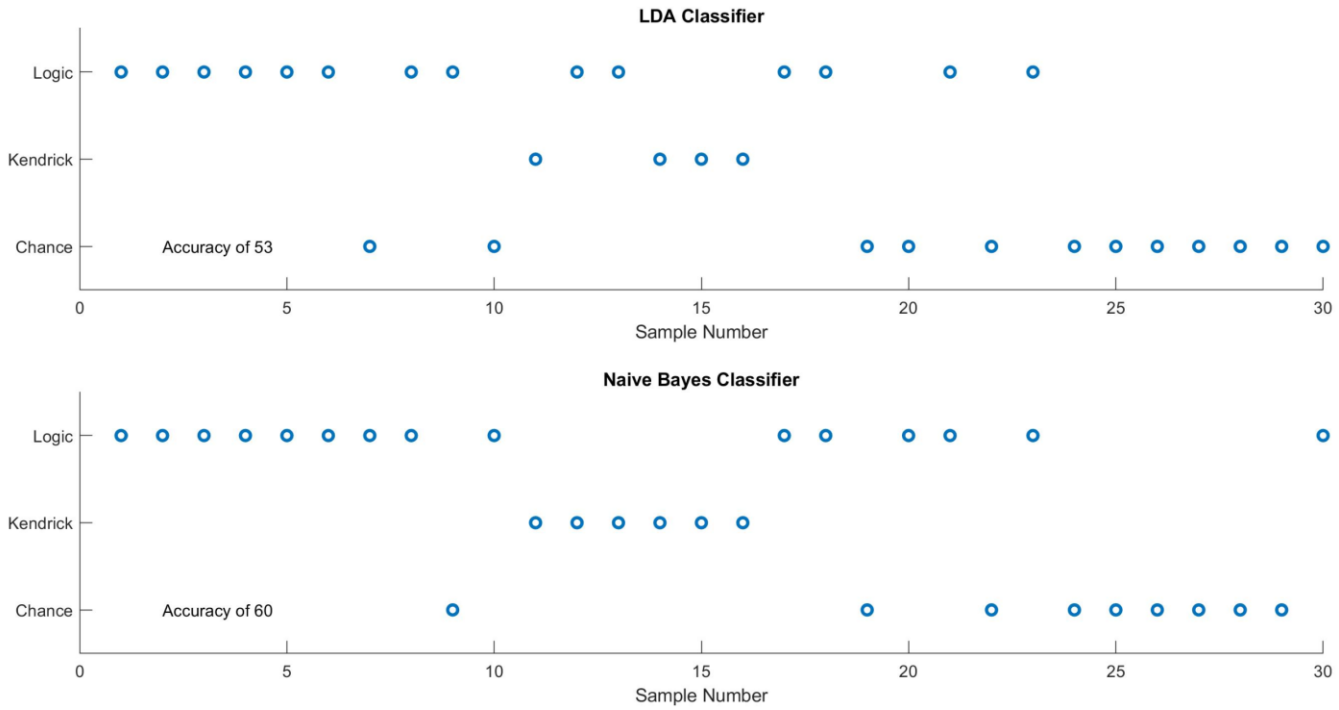


Figure 12. Classifying Logic, Kendrick, and Chance.

Here, like in test 1, 10 new Logic, Kendrick, and Chance clips were tested against the training dataset. LDA classified with a 53% accuracy, while the Naïve Bayes did so with a 60% accuracy. The results were obtained by classification on the abovementioned principle modes. The results are promising, especially considering the extent of the mixture in the principle component space, observed in figure 11. The results from figure 12 seem to indicate that Logic is not that unique, as both Kendrick and Chance seem to have produced similar enough music to be miss classified as Logic. Chance, however, seems to have four mistaken identifications by the LDA algorithm, and a mere two by the Naïve Bayes. This would demonstrate that Chance is producing fairly unique music. Maybe this goes to show why he's won so many awards, why Logic hasn't yet broken out on the scene, and why Kendrick just can't win that elusive Grammy.

VII. Conclusion

This exercise in SVD implementation showed that most data, be it in the form of facial images or musical clips, can largely be reduced to their low dimensional essence. This has a number of important applications. First, as demonstrated by the EigenFace example, only a fraction of the facial data is required to fully elucidate the picture of the person underneath. This has enormous consequences for data storage. Second, the SVD provides an excellent coordinate system by which data can be classified, and upon which subsequently developed machine learning algorithms can perform discriminatory tasks. The SVD, in short, can bring the data into a crystal clear light—the optimal coordinate system—in which a seemingly complex, muddled picture of data can be efficiently and accurately analyzed.

Appendix A: MatLab Functions and Implementation

audioread() – reads in audio data, both ears, as an n x 2 vector

dir() – changes directory

fft() – performs fast fourier transform

imshow() – displays matrix of data as image

reshape(data) – shapes an original data matrix to new dimensional specifications

pcolor() – displays checkboard matrix of data as image

save() – used extensively for the sake of saving large data matrices

svd(data) – performs singular value decomposition on matrix 'data'

Appendix B: MatLab Code Samples: Cropped EigenFace

```
% HW 4 Part 1
clear all; close all; clc;

%% Get files from Cropped

L = [];                                     % make library for image storage

for j = 1:38
    current_D = ['C:\Users\avery\Documents\AveryC\Academics\YEAR 4\Winter 2019\AMATH 482\Homework
Scripts\HW 4\Part 1 Faces\HW_Cropped\CroppedYale/yaleB',num2str(j,'%02d')];
    dir(current_D);
    cd(current_D);
    for k = 3:64
        image_file = dir(current_D);        % find image
        file = image_file(k).name;          % load in image to var
        image = imread(file);               % read in image
        image = imresize( double(image), [50 50] );     % resize matrix appropriately
        L = [L; reshape(image, 1, 50*50)];  % generate library

    end
    cd('C:\Users\avery\Documents\AveryC\Academics\YEAR 4\Winter 2019\AMATH 482\Homework Scripts\HW
4\Part 1 Faces\HW_Cropped\CroppedYale');
end

L = L';

%% Library development

[U, S, V] = svd(L, 'econ');             % perform efficient svd
d = diag(S);
Library_Cropped = L;

%% Energy

plot_energy = [];                   % initializes energy for plotting purposes
tol = 90;                           % break once 90% of data captured
for x = 1:length(d)                 % continue for all S
    energy = sum(d(1:x))/sum(d);    % find relative, cumulative energy
    plot_energy(x) = energy;        % facilitate plotting
    if energy > (tol/100)           % breaks at 90%
        break
    end
end
```

```
energy;
x;


figure(1)
plot(plot_energy, 'linewidth', [2])
xlabel('Number of modes')
ylabel('Relative energy')
title(['Energy from the first ', num2str(x),' modes gives ' , num2str(tol), '% of the cropped data'])



%% Finding dominant facial modes

i = 0;
figure(2)

for mode = [10 50 100 500]                                      % selected modes
    i = i + 1;
    subplot(2, 2, i)
    reconstruction = U(:,1:mode)*S(1:mode,1:mode)*V(:,1:mode)'; % low rank approx
    imshow(uint8(reshape(reconstruction(:,1), [50,50])));       % facilitates plot
    title([num2str(mode),' mode reconst.'])                     % gives title
end
```

Appendix C: Sample Song Loading and Analysis

```
% HW 4 Part 2

%% Load Rap

sec = 5; % duration of all clips
x_spec = 11;                            % time res of spectrogram
y_spec = 220500/4;                      % modes divided by sampling
vgt_spec = zeros(y_spec,x_spec);        % initialize the storage for spectrogram
Library = zeros(y_spec*x_spec,180);     % initialize library

current_D = ['C:\Users\avery\Documents\AveryC\Academics\YEAR 4\Winter 2019\AMATH 482\Homework Scripts\HW
4\Part 2 Music\Music_Test1\Rap'];
dir(current_D);
cd(current_D);

for j = 3:4
    audio_file = dir(current_D);        % get audio file from current directory
    file = audio_file(j).name;          % read in file to var
    y = audioread(file);                % audiofile read in
    y = y(:,1);                         % take in right ear data
    y = y(1:4:end);                     % only sample every 4th point
    Fs = length(y)/sec;                 % take in appropriate sample rate
    v = y';                             % make row vector
    n = length(v);                      % total sampled modes
    L = n/Fs;                           % len of song clip
    t2 = linspace(0, L, n);             % n+1 because first point same as last
    t = t2(1:n);                        % gen time axis
    k = (2*pi/L)*[0:(n/2-1) -n/2:-1];   % freq comp for fft.
    ks = fftshift(k);                   % shift adjust to counter
    ksplot = max(ks);
    tslide = 0:0.5:5;                   % sliding window
    for i = 1:length(tslide)
        g = exp(-10.*(t-tslide(i)).^10);% Gabor transform
        vg = v.*g;                      % Getting Gabored data
        vgt = fft(vg);                  % fft of Gabor
        vgt_spec(:,i) = fftshift(abs(vgt));
    end
    column = reshape(vgt_spec, 1, y_spec*x_spec); % reshape spectrogram to huge column
    Library(:,j-2) = column';           % collect library

%     pcolor(tslide, ks/(2*pi), vgt_spec(1:55124,:)), shading interp;
%     set(gca, 'Ylim', [0 ksplot*(1/8)])
%     colormap hot
```

```
%     title('Spectrogram from Pusha Man, by Chance the Rapper')
%     xlabel('Time in Seconds')
%     ylabel('Freq in Hz')
%     ylim([0 1500])

end

% Rank Approx Library
Library_Chance = Library;
save('Library_Chance.mat', 'Library_Chance')




%% Load Library
% Analysis of Rap

clear all; close all; clc
load Library_Pop.mat
[u,s,v] = svd(Library_Pop-mean(Library_Pop(:)), 'econ') ;


%% Plot the actual good stuff

figure(1)

% plot along principle axes
plot3(v(1:60,5), v(1:60,2), v(1:60,14),'bo', 'linewidth', [1])
xlabel('Mode 5')
ylabel('Mode 2')
zlabel('Mode 14')
title('Song mappings onto principle 2, 5, and 14')
hold on
plot3(v(61:120,5), v(61:120,2), v(61:120,14),'ro', 'linewidth', [1])
hold on
plot3(v(121:180,5), v(121:180,2), v(121:180,14),'yo', 'linewidth', [1])
legend('Chance', 'Kendrick', 'Logic')
grid on
hold off

%% Plotting Based on Proj

% Plot histograms to observe separation along each principle mode
hold off
x = linspace(-.4, .4, 100);
figure(2)
graphs = 12;
for j = 1:12
    subplot(4, 3, j)
    pdf1 = hist(v(1:50, j), x);
    pdf2 = hist(v(61:110, j), x);
    pdf3 = hist(v(121:170, j), x);
    plot(x,pdf1,x,pdf2,x,pdf3,'Linewidth',[1]);
    ylim([0 14])
    title(['Proj to PC ', num2str(j)])
    if j == 1
        legend('Chance', 'Kendrick','Logic');
    end
end

%% Linear Disc Analysis and Naive Bayes


components = [2, 5, 14];            % select modes
xtrain = [v(1:50,components); v(61:110, components); v(121:170,components)]; % plot along axes
ctrain = [ones(50,1); zeros(50,1); -1*ones(50,1)];  % label dataset
test = [v(51:60,components); v(111:120,components); v(171:180,components)];

class = (classify(test,xtrain,ctrain))';    % LDA classify based on training
true = [ones(10,1); zeros(10,1); -1*ones(10,1)]';   % true labels for data
accuracy1 = 100 - sum(abs(class-true))/(length(true))*100;
accuracy1 = floor(accuracy1);               % label accuracy
xplot = (1:length(true));
```

```
% Naive Bayes
nb = fitcnb(xtrain,ctrain);                    % Naive Bayes
predict = nb.predict(test);                    % Pre set
accuracy2 = 100 - sum(abs(predict'-true))/(length(true))*100;
accuracy2 = floor(accuracy2);


figure(3)
subplot(2, 1, 1)
scatter(xplot, class, 'linewidth', [2]);
title('LDA Classifier')
xlabel('Sample Number')
ylim([-1.5 1.5])
artists = {'Chance' ; 'Kendrick'; 'Logic'};
set(gca, 'ytick' , [-1, 0, 1],'yticklabel',artists)
txt = {['Accuracy of ',num2str(accuracy1)]};
text(2,-1,txt)

subplot(2, 1, 2)
scatter(xplot, predict,'linewidth', [2]);
title('Naive Bayes Classifier')
xlabel('Sample Number')
ylim([-1.5 1.5])
artists = {'Chance' ; 'Kendrick'; 'Logic'};
set(gca, 'ytick' , [-1, 0, 1], 'yticklabel',artists)
txt = {['Accuracy of ',num2str(accuracy2)]};
text(2,-1,txt)
```

Appendix D: Test 3 Data

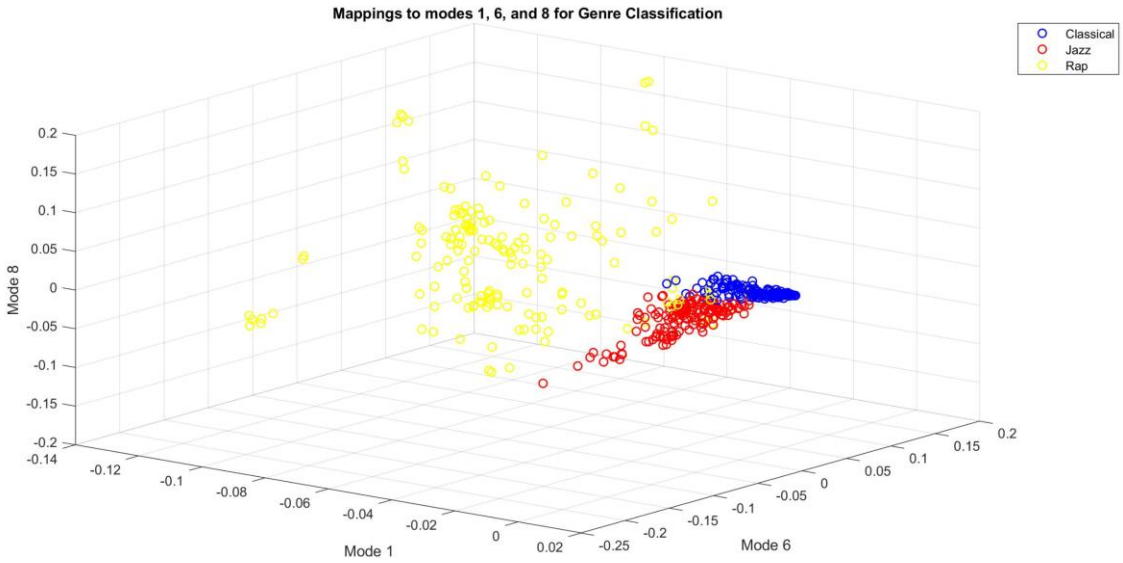
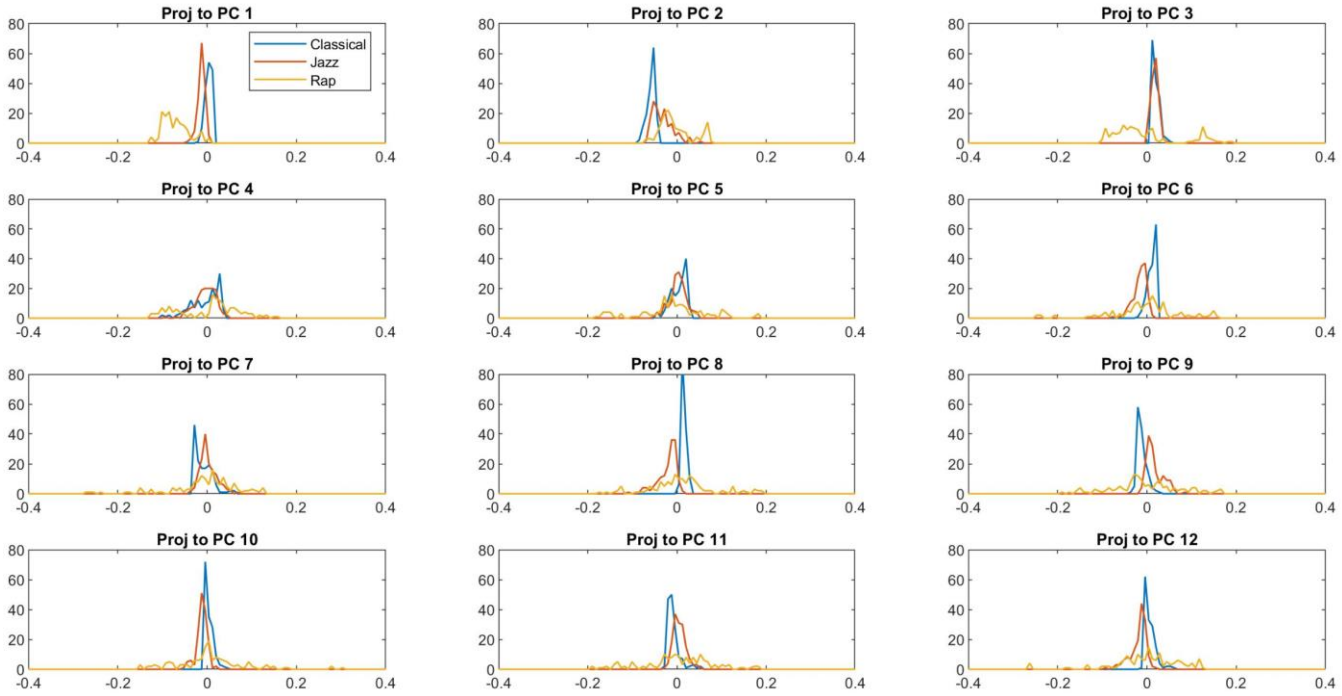
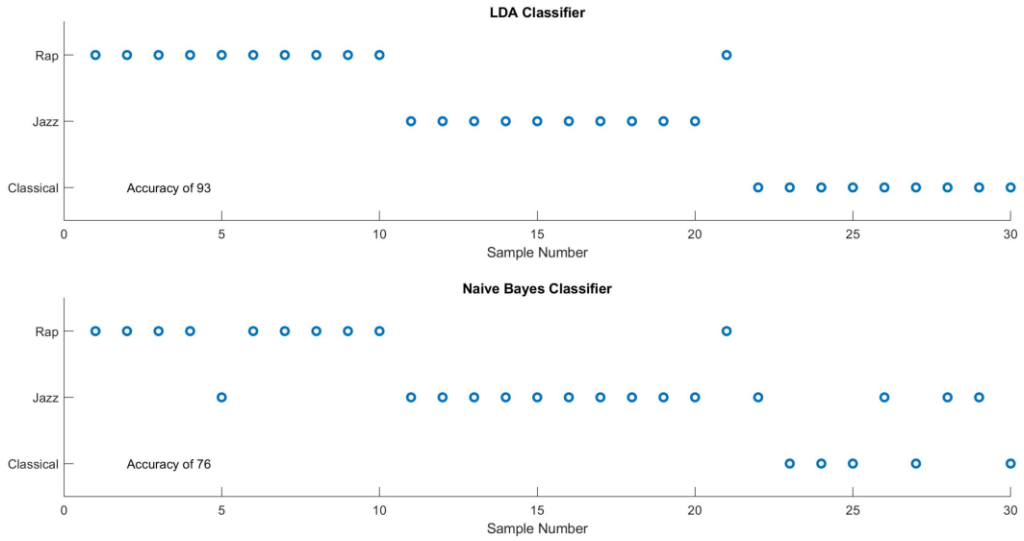


Figure showing distribution of Classical, Jazz, and Rap along principle axes 1, 6, and 8.



Modes that allowed for optimal plotting above, along modes 1, 6, and 8, which show good separation.



Analysis by LDA and NB show a relatively high degree of accuracy in classification, much more so than in the all Rap case. These algorithms were run with respect to the abovementioned modes 1, 6, and 8.