AMATH 482- Computational Methods in Data Analysis: Building a Classifier

Gargi Kher

March 6th, 2020

Abstract: Classifiers are often used in Machine Learning, a technique that is broadly used to train programs to make conclusions from data based on certain patterns it is trained to recognize. Classifiers serve to recognize data and group them into their respective categories. This cuts down on a lot of manual time. I will explain how I built and trained a classifier using Principal Component Analysis and Linear Decomposition Analysis and tested it to recognize different music genres and artists.

Sec. I. Introduction and Overview

In this assignment, we were given three tests to try for our classifier. The first test involved training a classifier to recognize a song from one of three different artists/bands across genres. The second test was to train the classifier to recognize a song from one of three different artists/bands in the same genre. The third test was for the classifier to recognize a song from one of three different genres. I'll be showing one way to build a classifier, but there are many ways this can be done.

While we had the freedom to choose the music to be analyzed, each clip had to be cut down to five seconds to account for processing time. One line of starter code was also provided by the instructor, which is what I used to read in the audio.

Sec. II. Theoretical Background

This assignment had three steps for training the classifier: 1) Create the spectrogram for all of the song clips, 2) perform Principal Component Analysis (PCA) on the combined spectrograms and 3) perform Linear Discriminant Analysis (LDA) on this result to determine a threshold or boundary separating each of the groups.

The purpose of the first step was to convert each audio signal into Fourier space. Because a Singular Value Decomposition (SVD) needed to be applied to each song (as part of step 2: PCA), it was better to convert the audio to a spectrogram rather than perform the SVD directly on the audio. This is because spectrograms hold information in both the time and frequency domains. One result of the SVD is a matrix S, which represents all the singular values of the spectrogram. These singular values give the important modes associated with the piece.

One thing to keep in mind when dealing with any type of signal is the frequencies that are being captured. Generally, most audio is sampled at a rate of 44100 Hz (or 44100 samples per second). According to the Nyquist frequency rule, however, the maximum frequency that can be captured at this sampling rate is 22050 Hz, which is half of the sampling rate. This becomes clear after discretizing the audio interval, or splitting it up into parts. For any signal you receive, it may be useful to know the maximum frequency that is captured. This way, when obtaining data for the signal, you don't waste time and space storing frequencies you don't need.

Once you have the spectrograms representing each song, you can apply PCA to it by taking the SVD of the matrix. As a brief overview, applying the SVD will result in a matrix U and V, both of which are the left and right singular vectors of the spectrogram (respectively), and a diagonal matrix S that contains the singular values of the spectrogram. Applying S to V gives the projection of the data onto its principal components, which is what will ultimately help separate the groups. The next step is LDA. From all of the projections you have (each one representing some data in the training set, or in this case, one group of songs), the goal is to separate the basis of principal components. We want all of these points to have a certain variance so that the program can better classify the test data. To do this, we consider the between-class (Equation 1) and within-class (Equation 2) scatter matrices.

$S_B = \sum_{n=1}^{n} m_j(\vec{\mu}_n - \mu)(\vec{\mu}_n - \mu)^T$ (1) Between-class matrix

$S_w = \sum_{n=1}^{n} \sum_{\vec{x}} (\vec{x} - \vec{\mu}_n)(\vec{x} - \vec{\mu}_n)^T$ (2) Within-class matrix

We want the points in each song to be as close to each other as possible and have little variance amongst themselves, so we represent that by the within-class matrix being really close to zero. Conversely, we want all the points in one group of songs to be different from the points in another group, so we represent that by the between-class matrix being as large as possible. This way, we can have separation between each group. In the between-class matrix, we take the difference between the mean of the values across a group (or song) n and the mean of the values between all of the groups. This is multiplied by its transpose and weighed by the number of elements m_n in that group. This is done for all of the groups, and these values are summed together. The within-class matrix is similar in that it is the sum of the within-class matrices of all groups, but it subtracts the mean of the points in each group from the points themselves before multiplying by its transpose. Both of these matrices are used to solve for $\vec{w}$, the eigenvector to project on, in a generalized eigenvalue problem (Equation 3).

$S_B\vec{w} = \lambda\vec{w}S_w$ (3) Generalized Eigenvalue Problem

$\vec{w}$ represents the eigenvector corresponding to the largest eigenvalue, and serves as the best possible line (or vector) we want to project our data onto so that it is separated with high variance between groups and low variance within groups. Solving for this generalized eigenvalue problem allows you to project this onto the data points for each group. These projections will allow us to determine thresholds for each group. It is important to note that while the goal is to find a threshold so that we can classify our data, these values will not likely be perfect, as there may be certain data points in one group that overlap with another group. What we care about is a high success rate for our data rather than a perfect success rate.

Sec. III. Algorithm Implementation and Development

To load the songs into MATLAB, I used code provided by the instructor. I was using music from the Free Music Archive, so I was able to load the songs into MATLAB as mp3 files (Appendix B, Lines 3-34). This made cutting them down relatively simple. After I loaded the songs for all of the tests, I had to perform PCA and LDA on the spectrograms of my training data. Because these methods had to be done for each song, I made functions to reduce the amount of code I wrote.

I first called a function that would discretize the audio clip (Appendix B, Lines 253-284) and make the spectrogram. My audio came in the form of two channels (or columns), so the first step I took inside this function was to take the average of both of the columns and create an audio vector containing one channel. This shouldn't have affected the audio sample significantly, as each channel basically represents the sounds that come out of different speakers. There are a lot of audio samples within this variable, and knowing I had to analyze many songs, I resampled my data by taking every ten points. Fs, my sampling frequency, which was set at 44100 Hz for most of the songs, was cut by a factor of ten to account for this change. Every time I called this function for a song, I kept track of the first five seconds, last five seconds, and some five seconds before the end (which I designated as the "middle") of the song. This way, I had more clips for each test without having to get one clip from a single song. I then discretized the interval, designating the length L to be 5 seconds and the number of samples n to be the number of samples in the five second clip, or $Fs * L$. The frequency vector k was only multiplied by $\frac{1}{L}$ instead of $\frac{2\pi}{L}$ in this case because we needed to work in Hertz rather than angular frequency. Once the time and frequency vectors were defined, I could make the spectrogram. I applied a Gabor Filter to the signal and generated a spectrogram. Because this was a two-dimensional matrix, I reshaped it to output as a column vector. It's worth noting that I reused many of the songs for Test 3, which is why I didn't have to call my $discretizespec$ function as many times as I did for the other tests (Appendix B, Lines 36-50, 93-104, 147-151).

Once I obtained the output, I manually combined the Spectrograms into their respective vectors (so that there was one vector per group) before sending them off to my second function, $PCAandLDA$ (Appendix B, Lines 52-53, 106-107, 152-153). This function combined the spectrograms from all of the groups into one matrix and performed SVD on that. From this, I was able to get the projections for each group individually and then get the projections onto their principal components. Because we were comparing three groups, I obtained two projection vectors ($\vec{w}$ and $\vec{w}_2$). I also took the mean of the projection of each group to help during the testing step, so that I ended up with two mean values for each group (msN and msN_proj2, where N is the group number) (Appendix B, Lines 189-251).

When testing the classifier, I used the mean values for both projections as a "threshold". The test data was created into a spectrogram just like the training data and multiplied by U, which was modified to the size of the number of features I was looking at. This way, I obtained the projection of the data in each song onto the principal components. Because I had two projections, I subtracted this coordinate from all three means to find the value that represented the smallest distance to the mean. This determined the group that the test song belonged to (Appendix B, Lines 61-91, 115-15, 161-187).

Sec. IV. Computational Results

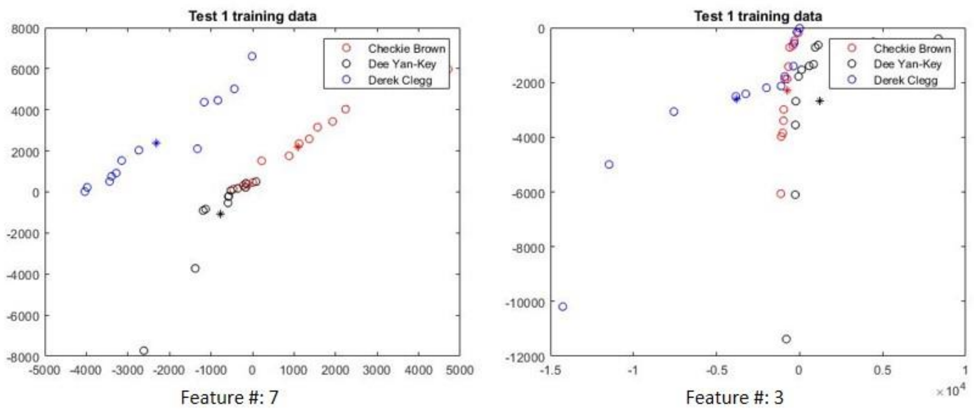

Figure 1: Plots depicting projection of each group corresponding to the second highest eigenvalue against projection of each group corresponding to the highest eigenvalue in Test 1 (different artists from different genres). Each group's mean is also plotted. From left to right, the plots represent features 7 and 3.

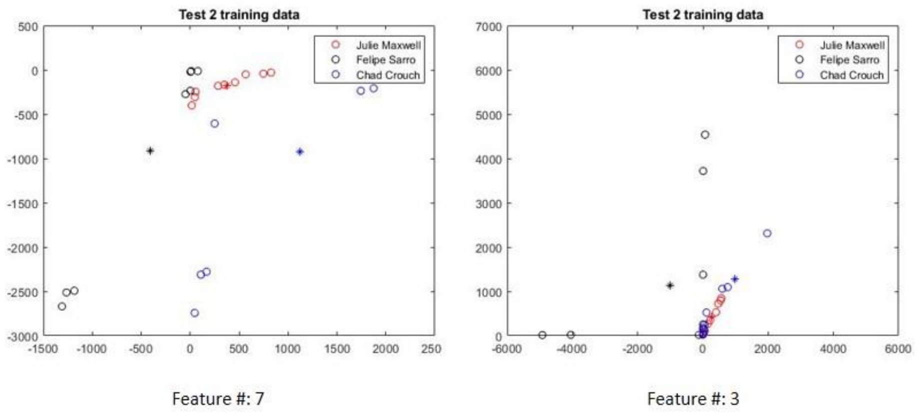

Figure 2: Plots depicting projection of each group corresponding to the second highest eigenvalue against projection of each group corresponding to the highest eigenvalue in Test 2 (different artists from same genre). Each group's mean is also plotted. From left to right, the plots represent features 7 and 3.

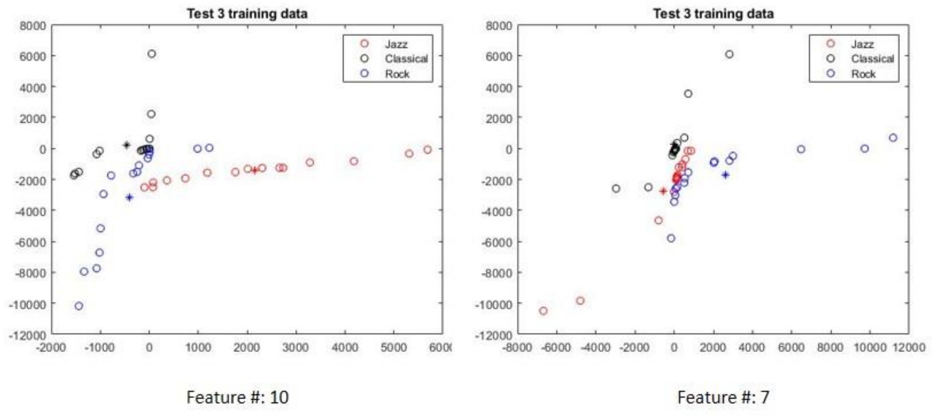

Figure 3: Plots depicting projection of each group corresponding to the second highest eigenvalue against projection of each group corresponding to the highest eigenvalue in Test 3 (different genres). Each group's mean is also plotted. From left to right, the plots represent features 10 and 7.

Test data: All of the songs for each test should have been classified as [1 1 1 2 2 2 3 3 3]. The songs for test one were classified as [1 3 3 3 1 3 3 2 3] for feature of 7, corresponding to a 33.33% success rate. They had an accuracy of 44.44% for a feature of 3, as they were classified as [1 3 1 1 3 2 3 1 1]. When the songs for test 2 were analyzed at a feature of 7, they were classified as [3 2 2 2 2 2 2 2 2] and had a success rate of 33.33%. At a feature of 3, they were classified as [1 2 1 1 1 1 2 1 1] and had a 22.22% success rate. For test 3, a feature of 10 gave the classification [2 1 2 2 2 2 3 3 3] and a success rate of 77.78%. A feature of 7 gave the classification [3 3 3 3 2 2 3 3 3] and a 55.56% success rate.

Sec. V. Summary and Conclusions

For each group, I plotted the projection corresponding to the second largest eigenvalue against the projection corresponding to the largest eigenvalue, but changed the number of features that I looked at in order to see how these influenced the separation of the groups. In Test 1, I had nine clips for each group. I plotted these projections when looking at seven features and three features (Sec.V., Figure 1). Overall, the group separation looks much better for seven features than that for three features. In the plot for three features, you can see that all of the groups overlap. From this, it doesn't seem like the test data would do very well if we looked at only three features. I had nine clips per group for Test 2 and plotted the projections for seven and three features as well (Sec.V., Figure 2). Based on the plots, I ended up deciding that looking at seven features was best, as it allowed for a good amount of separation. Finally, for Test 3, I had sixteen clips per group and plotted these projections for ten and seven features (Sec.V., Figure 3). The plot with ten features shows much better separation than the plot with seven features. For all three tests, it seemed that the more features I looked at, the better the separation between groups became. This makes sense, as increasing the number of features gets closer to the total amount of data points you have. However, because it isn't reasonable to look at the thousands or millions of data points you may have, finding the optimal amount of data points to analyze is important.

My test data (end of Sec.V.) clearly does not do well, as none of the tests have a success rate over 90%. For Test 1, it was surprising to me that that the success rate was higher for the analysis with a smaller number of features. Test 2 had the best success rate for a feature number of 7, but at 33.33%, this suggests that the classification was random. Test 3 had the best success rate of all three tests. This was likely because I had more samples for this test than the others. I might be able to improve the success rates of Tests 2 and 3 by adding more clips to them. Another reason Test 3 might have done significantly better was because it had more diverse training data. I could possibly improve the success rates of Tests 1 and 2 by adding more clips from different songs rather than taking multiple clips from the same song.

Sec. VI. Sources/References

Kutz, Nathan J. (2013). *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*

Free Music Archive. WMFU Radio Station, 2009.

Appendix A. MATLAB functions used and brief implementation explanation

$webread()$ - takes in audio as a url and outputs a sample size y and sample rate Fs.

$eig()$ - solves general eigenvalue problems.

$norm(X, 2) -$ takes the 2-norm of a vector X.

Appendix B. MATLAB codes

```
%% Test 1
%Checkie Brown
[y,Fs] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-org/music/no_curator/Checkie_Brown/hey/Checkie_Brown_-_12_-
_Rosalie_CB_29.mp3');
[y12,Fs12] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-org/music/no_curator/Checkie_Brown/hey/Checkie_Brown_-_01_-
_Susie_the_Cat_-_Boss_CB_25.mp3');
[y16,Fs16] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-org/music/no_curator/Checkie_Brown/hey/Checkie_Brown_-_06_-
_Gangster_Downhill_-_Hippie_Beatnix_CB_26.mp3');
[y24,Fs24] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-
org/music/no_curator/Checkie_Brown/Staubkeller_Live/Checkie_Brown_-_05_-
_Wirklich_Wichtig_CB_011.mp3');
%Dee Yan-Key
[y1, Fs1] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-org/music/ccCommunity/Dee_Yan-Key/Thalatta_Thalatta/Dee_Yan-
Key_-_03_-_Triton.mp3');
[y13,Fs13] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-org/music/ccCommunity/Dee_Yan-Key/Thalatta_Thalatta/Dee_Yan-
Key_-_02_-_Amphitrite.mp3');
[y17,Fs17] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-org/music/ccCommunity/Dee_Yan-Key/Perpetual_Peace/Dee_Yan-
Key_-_03_-_World_of_Brothers___Allegretto.mp3');
[y25,Fs25] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-org/music/ccCommunity/Dee_Yan-Key/Perpetual_Peace/Dee_Yan-
Key_-_01_-_Wars_End___Allegro_molto_moderato.mp3');
%Derek Clegg
[y2,Fs2] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-org/music/no_curator/Derek_Clegg/Solar/Derek_Clegg_-_12_-
_Annalise.mp3');
[y14,Fs14] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-org/music/no_curator/Derek_Clegg/Solar/Derek_Clegg_-_08_-
_Cest_La_Vie.mp3');
[y15,Fs15]=webread('https://files.freemusicarchive.org/storage-
freemusicarchive-org/music/no_curator/Derek_Clegg/Solar/Derek_Clegg_-_07_-
_Ill_Almost_Get_Us_There.mp3');
[y26,Fs26] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-org/music/no_curator/Derek_Clegg/Solar/Derek_Clegg_-_03_-
_Misunderstood.mp3');
%% Test 2- Classical
[y8, Fs8] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-
org/music/ccCommunity/Julie_Maxwells_Piano_Music/Farther_Than_All_The_Stars/J
ulie_Maxwells_Piano_Music_-_09_-__Sakura.mp3');
[y18,Fs18] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-
org/music/ccCommunity/Julie_Maxwells_Piano_Music/Classic_Piano_Collection_fro
m_the_Princess_of_Mars/Julie_Maxwells_Piano_Music_-_10_-__Starry_Sky.mp3');
```

```
[y19,Fs19] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-
org/music/ccCommunity/Julie_Maxwells_Piano_Music/Classic_Piano_Collection_fro
m_the_Princess_of_Mars/Julie_Maxwells_Piano_Music_-_11_-
_Labyrinth_Dreams.mp3');

[y9, Fs9] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-
org/music/Oddio_Overplay/Felipe_Sarro/Maurice_Ravel/Felipe_Sarro_-_15_-
_Ravel_-_Bolro_transcribed_by_Branga.mp3');
[y20,Fs20] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-
org/music/Oddio_Overplay/Felipe_Sarro/Bach_Original_works_and_transcriptions/
Felipe_Sarro_-_18_-_Bach_Prelude_BWV_855a_Siloti_transcription.mp3');
[y21,Fs21] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-
org/music/Oddio_Overplay/Felipe_Sarro/Bach_Original_works_and_transcriptions/
Felipe_Sarro_-_12_-
_Bach_Partita_2_BWV_1004_Chaconne_Siloti_transcription.mp3');

[y10, Fs10] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-
org/music/ccCommunity/Chad_Crouch/Vol_4_Satie_Rearranged_Furniture_Music/Chad
_Crouch_-_06_-_Gnossienne_3.mp3');
[y22,Fs22] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-
org/music/ccCommunity/Chad_Crouch/Field_Report_Vol_I_Oaks_Bottom_Instrumental
/Chad_Crouch_-_04_-_The_Spring_Instrumental.mp3');
[y23,Fs23] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-
org/music/ccCommunity/Chad_Crouch/Field_Report_Vol_I_Oaks_Bottom_Instrumental
/Chad_Crouch_-_The_Chorus_Ceases_Instrumental.mp3');
%%
[y3, Fs3] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-org/music/none_given/Scott_Holmes/Scott_Holmes_-
_Singles/Scott_Holmes_-_Driven_To_Success.mp3');
[y5,Fs5] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-
org/music/ccCommunity/Unheard_Music_Concepts/The_Lasso_of_Time/Unheard_Music_
Concepts_-_16_-_Dakota.mp3');
[y6,Fs6] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-
org/music/no_curator/J_Syreus_Bach/It_Rains__Abstract_Jazz/J_Syreus_Bach_-
_Goodbye_Grandmother.mp3');
[y7,Fs7] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-
org/music/ccCommunity/Chad_Crouch/Field_Report_Vol_I_Oaks_Bottom_Instrumental
/Chad_Crouch_-_The_Chorus_Ceases_Instrumental.mp3');
[y11,Fs11] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-
org/music/Decoder_Magazine/Lately_Kind_of_Yeah/Poindexter/Lately_Kind_of_Yeah
_-_14_-_Goner.mp3');
%% Test 1- Different artists over different genres
%Brown
[St_specb,St_specbmed, St_specbend] = discretizespec(y,Fs);
[St_specb1,St_specbmed1, St_specbend1] = discretizespec(y12,Fs12);
```

```
[St_specb2,St_specbmed2, St_specbend2] = discretizespec(y16,Fs16);
[St_specb3,St_specbmed3, St_specbend3] = discretizespec(y24,Fs24);
%Yan-Key
[St_specy,St_specymed, St_specyend] = discretizespec(y1,Fs1);
[St_specy1,St_specymed1, St_specyend1] = discretizespec(y13,Fs13);
[St_specy2,St_specymed2, St_specyend2] = discretizespec(y17,Fs17);
[St_specy3,St_specymed3, St_specyend3] = discretizespec(y25,Fs25);
%Clegg
[St_specc,St_speccmed, Stspeccend] = discretizespec(y2,Fs2);
[St_specc1,St_speccmed1, Stspeccend1] = discretizespec(y14,Fs14);
[St_specc2,St_speccmed2, Stspeccend2] = discretizespec(y15,Fs15);
[St_specc3,St_speccmed3, Stspeccend3] = discretizespec(y26,Fs26);
%%
feature = 7;
[U_t1,w,w2,sortgroupb, sortgroupy, sortgroupc, sortgroupb_proj2,
sortgroupy_proj2, sortgroupc_proj2,ms1,ms2,ms3,ms1_proj2,ms2_proj2,ms3_proj2]
= PCAandLDA(feature,[St_specb,St_specbmed,
St_specbend,St_specb1,St_specbmed1, St_specbend1,St_specb2,St_specbmed2,
St_specbend2,St_specb3,St_specbmed3, St_specbend3], [St_specy,St_specymed,
St_specyend,St_specy1,St_specymed1, St_specyend1,St_specy2,St_specymed2,
St_specyend2,St_specy3,St_specymed3, St_specyend3], [St_specc,St_speccmed,
Stspeccend,St_specc1,St_speccmed1, Stspeccend1,St_specc2,St_speccmed2,
Stspeccend2,St_specc3,St_speccmed3, Stspeccend3]);
%%
plot(sortgroupb,sortgroupb_proj2,'ro',sortgroupy,sortgroupy_proj2,'ko',sortgr
oupc,sortgroupc_proj2,'bo')
hold on
plot(ms1,ms1_proj2,'r*',ms2,ms2_proj2,'k*',ms3,ms3_proj2,'b*');
title('Test 1 training data')
legend('Checkie Brown','Dee Yan-Key','Derek Clegg')
%% Test data
[yt1,Fst1] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-org/music/no_curator/Checkie_Brown/hey/Checkie_Brown_-_08_-
_Hippie_Bulle_-Stoned_Funghi_CB_28.mp3');
[yt2,Fst2] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-org/music/ccCommunity/Dee_Yan-Key/Thalatta_Thalatta/Dee_Yan-
Key_-_01_-_Poseidon.mp3');
[yt3,Fst3] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-org/music/no_curator/Derek_Clegg/Solar/Derek_Clegg_-_08_-
_Cest_La_Vie.mp3');
[Spectest1beg, Spectest1med, Spectest1end] = discretizespec(yt1,Fst1);
[Spectest2beg, Spectest2med, Spectest2end] = discretizespec(yt2,Fst2);
[Spectest3beg, Spectest3med, Spectest3end] = discretizespec(yt3,Fst3);
spectotal = [Spectest1beg, Spectest1med, Spectest1end,Spectest2beg,
Spectest2med, Spectest2end,Spectest3beg, Spectest3med, Spectest3end];
%minimize the distance between the pvals and means
hiddenlabels = [1,1,1,2,2,2,3,3,3];
Resvec = zeros(1,size(spectotal,2));
    TestNum = size(spectotal,2);
for i = 1:TestNum
    TestMat = U_t1'*spectotal(:,i);
    pval = w'*TestMat;
    pval2 = w2'*TestMat;
    dif1 = ([ms1,ms1_proj2]-[pval,pval2]);
    dif2 = ([ms2,ms2_proj2]-[pval,pval2]);
    dif3 = ([ms3,ms3_proj2]-[pval,pval2]);
    diftotal = [abs(dif1), abs(dif2), abs(dif3)];
    [~,col] = min(diftotal);
    if(col==1)
        Resvec(1,i) = 1;
    elseif(col==2)
        Resvec(1,i) = 2;
    else
        Resvec(1,i) = 3;
    end
end
    error = Resvec-hiddenlabels;
    count = find(error==0);
    succRate = length(count)/length(Resvec);
%% Test 2- Different artists over same genre
%JM
[St_specjm,St_specjmmed,St_specjmend] = discretizespec(y8,Fs8);
[St_specjm1,St_specjmmed1,St_specjmend1] = discretizespec(y18,Fs18);
[St_specjm2,St_specjmmed2,St_specjmend2] = discretizespec(y19,Fs19);
%SS
[St_specss, St_specssmed,St_specssend] = discretizespec(y9,Fs9);
[St_specss1, St_specssmed1,St_specssend1] = discretizespec(y20,Fs20);
[St_specss2, St_specssmed2,St_specssend2] = discretizespec(y21,Fs21);
%CC
[St_speccc, St_specccmed,Stspecccend] = discretizespec(y10,Fs10);
[St_speccc1, St_specccmed1,Stspecccend1] = discretizespec(y22,Fs22);
[St_speccc2, St_specccmed2,Stspecccend2] = discretizespec(y23,Fs23);
%%
feature = 3;
[U_t2,w,w2,sortgroupjm, sortgroupss, sortgroupcc,sortgroupjm_proj2,
sortgroupss_proj2,
sortgroupcc_proj2,ms1,ms2,ms3,ms1_proj2,ms2_proj2,ms3_proj2] =
PCAandLDA(feature,[St_specjm,St_specjmmed,St_specjmend,St_specjm1,St_specjmme
d1,St_specjmend1,St_specjm2,St_specjmmed2,St_specjmend2], [St_specss,
St_specssmed,St_specssend,St_specss1, St_specssmed1,St_specssend1,St_specss2,
St_specssmed2,St_specssend2], [St_speccc,
St_specccmed,Stspecccend,St_speccc1, St_specccmed1,Stspecccend1,St_speccc2,
St_specccmed2,Stspecccend2]);
%%
plot(sortgroupjm,sortgroupjm_proj2,'ro',sortgroupss,sortgroupss_proj2,'ko',so
rtgroupcc,sortgroupcc_proj2,'bo')
hold on
plot(ms1,ms1_proj2,'r*',ms2,ms2_proj2,'k*',ms3,ms3_proj2,'b*');
title('Test 2 training data')
legend('Julie Maxwell','Felipe Sarro','Chad Crouch')
%%
[yt1,Fst1] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-
org/music/ccCommunity/Julie_Maxwells_Piano_Music/Classic_Piano_Collection_fro
m_the_Princess_of_Mars/Julie_Maxwells_Piano_Music_-_07_-_Sleeping_Rose.mp3');
[yt2,Fst2] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-
org/music/Oddio_Overplay/Felipe_Sarro/Bach_Original_works_and_transcriptions/
Felipe_Sarro_-_09_-_Bach_Prelude_8_BWV_853.mp3');
[yt3,Fst3] = webread('https://files.freemusicarchive.org/storage-
freemusicarchive-
org/music/ccCommunity/Chad_Crouch/Field_Report_Vol_I_Oaks_Bottom_Instrumental
/Chad_Crouch_-_Stratum_Instrumental.mp3');
[Spectest1beg, Spectest1med, Spectest1end] = discretizespec(yt1,Fst1);
```

```
[Spectest2beg, Spectest2med, Spectest2end] = discretizespec(yt2,Fst2);
[Spectest3beg, Spectest3med, Spectest3end] = discretizespec(yt3,Fst3);
spectotal = [Spectest1beg, Spectest1med, Spectest1end,Spectest2beg,
Spectest2med, Spectest2end,Spectest3beg, Spectest3med, Spectest3end];
%minimize the distance between the pvals and means
hiddenlabels = [1,1,1,2,2,2,3,3,3];
Resvec = zeros(1,size(spectotal,2));
    TestNum = size(spectotal,2);
for i = 1:TestNum
    TestMat = U_t2'*spectotal(:,i);
    pval = w'*TestMat;
    pval2 = w2'*TestMat;
    dif1 = ([ms1,ms1_proj2]-[pval,pval2]);
    dif2 = ([ms2,ms2_proj2]-[pval,pval2]);
    dif3 = ([ms3,ms3_proj2]-[pval,pval2]);
    diftotal = [abs(dif1), abs(dif2), abs(dif3)];
    [~,col] = min(diftotal);
    if(col==1)
        Resvec(1,i) = 1;
    elseif(col==2)
        Resvec(1,i) = 2;
    else
        Resvec(1,i) = 3;
    end
end
    error = Resvec-hiddenlabels;
    count = find(error==0);
    succRate = length(count)/length(Resvec);
%% Test 3 - Different genres
%using classical, rock, and jazz from above. Only loading some new ones in.
[St_specsh,St_specshmed,St_specshend] = discretizespec(y3,Fs3); %rock
[St_specy,St_specymed,St_specyend] = discretizespec(y11,Fs11); %rock
[St_specu,St_specumed,St_specuend] = discretizespec(y5,Fs5); %jazz
[St_specjb,St_specjbmed,St_specjbend] = discretizespec(y6,Fs6); %jazz
feature = 10;
[U_t3,w,w2,sortgroupjazz, sortgroupclass, sortgrouprock,sortgroupjazz_proj2,
sortgroupclass_proj2,
sortgrouprock_proj2,ms1,ms2,ms3,ms1_proj2,ms2_proj2,ms3_proj2 ] =
PCAandLDA(feature, [St_specb,St_specbmed, St_specbend,St_specb1,St_specbmed1,
St_specbend1,St_specb2,St_specbmed2,
St_specbend2,St_specu,St_specumed,St_specuend,St_specjb,St_specjbmed,St_specj
bend], [St_specy,St_specymed, St_specyend,St_specy1,St_specymed1,
St_specyend1,St_specy2,St_specymed2,
St_specyend2,St_specjm,St_specjmmed,St_specjmend,St_specss1,
St_specssmed1,St_specssend1], [St_specc,St_speccmed,
Stspeccend,St_specc1,St_speccmed1, Stspeccend1,St_specc2,St_speccmed2,
Stspeccend2,St_specsh,St_specshmed,St_specshend,St_specy,St_specymed,St_specy
end]);
plot(sortgroupjazz,sortgroupjazz_proj2,'ro',sortgroupclass,sortgroupclass_pro
j2,'ko',sortgrouprock,sortgrouprock_proj2,'bo')
hold on
plot(ms1,ms1_proj2,'r*',ms2,ms2_proj2,'k*',ms3,ms3_proj2,'b*');
title('Test 3 training data')
legend('Jazz','Classical','Rock')
%%
%using most data from above as "testing data"
[Spectest1beg, Spectest1med, Spectest1end] = discretizespec(y12,Fs12); %jazz
```

```
[Spectest2beg, Spectest2med, Spectest2end] = discretizespec(y11,Fs11); %rock
[Spectest3beg, Spectest3med, Spectest3end] = discretizespec(y8,Fs8);
%classical
spectotal = [Spectest1beg, Spectest1med, Spectest1end,Spectest2beg,
Spectest2med, Spectest2end,Spectest3beg, Spectest3med, Spectest3end];
hiddenlabels = [1,1,1,2,2,2,3,3,3];
Resvec = zeros(1,size(spectotal,2));
    TestNum = size(spectotal,2);
for i = 1:TestNum
    TestMat = U_t3'*spectotal(:,i);
    pval = w'*TestMat;
    pval2 = w2'*TestMat;
    dif1 = ([ms1,ms1_proj2]-[pval,pval2]);
    dif2 = ([ms2,ms2_proj2]-[pval,pval2]);
    dif3 = ([ms3,ms3_proj2]-[pval,pval2]);
    diftotal = [abs(dif1), abs(dif2), abs(dif3)];
    [~,col] = min(diftotal);
    if(col==1)
        Resvec(1,i) = 1;
    elseif(col==2)
        Resvec(1,i) = 2;
    else
        Resvec(1,i) = 3;
    end
end
    error = Resvec-hiddenlabels;
    count = find(error==0);
    succRate = length(count)/length(Resvec);
%%
function[U,w,w2,sortgroup1, sortgroup2, sortgroup3, sortgroup1_proj2,
sortgroup2_proj2, sortgroup3_proj2,ms1,ms2,ms3,ms1_proj2,ms2_proj2,ms3_proj2]
= PCAandLDA(feature, St_spec1, St_spec2, St_spec3)
n1 = size(St_spec1,2); n2 = size(St_spec2,2); n3 = size(St_spec3,2);
[U S V] = svd([St_spec1, St_spec2,St_spec3],'econ');
proj = S*V.';
U = U(:,1:feature);
group1 = proj(1:feature,1:n1);
group2 = proj(1:feature,n1+1:n1+n2);
group3 = proj(1:feature,n1+n2+1:n1+n2+n3);

m = mean(proj(1:feature,:),2);
m1 = mean(group1,2);
m2 = mean(group2,2);
m3 = mean(group3, 2);

Sw = 0; %within class variances
for k = 1:n1
    Sw = Sw+ (group1(:,k)-m1)*(group1(:,k)-m1)';
end
for k = 1:n2
    Sw = Sw+ (group2(:,k)-m2)*(group2(:,k)-m2)';
end
for k = 1:n3
    Sw = Sw+ (group3(:,k)-m3)*(group3(:,k)-m3)';
end
```

```
Sb = 0;
Sb = Sb+(m1-m)*(m1-m)';
Sb = Sb+(m2-m)*(m2-m)';
Sb = Sb+(m3-m)*(m3-m)';
[V2, D] = eig(Sb,Sw); %LDA
[~,ind] = max(abs(diag(D))); %set max equal to zero and find eigenvector
w = V2(:,ind);
w = w/norm(w,2);

vgroup1 = w'*group1;
vgroup2 = w'*group2;
vgroup3 = w'*group3;

sortgroup1 = sort(vgroup1);
sortgroup2 = sort(vgroup2);
sortgroup3 = sort(vgroup3);

ms1 = mean(sortgroup1);
ms2 = mean(sortgroup2);
ms3 = mean(sortgroup3);

%proj 2
D(ind,ind) = 0;
[~,ind] = max(abs(diag(D))); %set max equal to zero and find eigenvector
w2 = V2(:,ind);
w2 = w2/norm(w2,2);

vgroup1_proj2 = w2'*group1;
vgroup2_proj2 = w2'*group2;
vgroup3_proj2 = w2'*group3;
sortgroup1_proj2 = sort(vgroup1_proj2);
sortgroup2_proj2 = sort(vgroup2_proj2);
sortgroup3_proj2 = sort(vgroup3_proj2);

ms1_proj2 = mean(sortgroup1_proj2);
ms2_proj2 = mean(sortgroup2_proj2);
ms3_proj2 = mean(sortgroup3_proj2);
end

function [St_spec,St_specmed,St_spec1] = discretizespec(y,Fs)
Fs = 44100/10;
v = y.';
sig = (v(1,:)+v(2,:))./2;
sig = sig(1:10:length(sig));
L = 5;
n = Fs*L; %chose my own n
v = sig(1:n); %make into five-second clip
vmed = sig(end-n+1-n:end-n);
v1 = sig(end-n+1:end);
t2 = linspace(0,L,n+1);
t = t2(1:n);
k = ((1)/L)*[0:n/2-1 -n/2:-1];
% Matlab spec function
tslide = 0:0.1:L;
a = 1;
```

```
St_spec = zeros(length(tslide),n);
St_specmed = zeros(length(tslide),n);
St_spec1 = zeros(length(tslide),n);
for b = 1:length(tslide)
g1 = exp(-a*(t-tslide(b)).^2);
St = fft(g1.*v); %St1 is in the frequency domain
Stmed = fft(g1.*vmed);
St1 = fft(g1.*v1);
St_spec(b, :) = fftshift(abs(St));
St_specmed(b,:) = fftshift(abs(Stmed));
St_spec1(b,:) = fftshift(abs(St1));
end
St_spec = reshape(St_spec,[length(tslide)*length(St_spec),1]);
St_specmed = reshape(St_specmed,[length(tslide)*length(St_specmed),1]);
St_spec1 = reshape(St_spec1,[length(tslide)*length(St_spec1),1]);
end
```