Music Classification

Eileen Chang

March 6, 2020

Abstract

Linear Discriminate Analysis is a machine learning technique that helps us classify music from several different bands and genres. An assortment of fifteen songs from each band/genre is compiled, and spectrograms of these data are made. An economy singular value decomposition of the spectrograms is performed. The total amount of songs is fifteen. Ten songs for training, five songs for tests. In this paper, we are going to test how well can the LDA performs by having different number of the feature.

1 Introduction and Overview

Music genres are instantly recognizable to us. We are always curious about how the brain classifies such information and how it makes a decision based upon hearing a new piece of music. Machine learning is a method to simulate the way brain works. It will extract meaningful features from data into distinct patterns that can be later used for decision making. It can learn from and make predictions on the given data. There is both a supervised and unsupervised machine learning method. In this paper, we are going to use supervised learning to classify the music. We will give labels for 5-second clip training data and use linear discriminant analysis to classify the songs into groups.

Three different analyses are conducted in this paper. Three bands/genres of music are chosen for each test. Fifteen songs of each band/genre are chosen and a 5-second sample from a song is selected after a spectrogram is constructed. The spectrogram is decomposed into its constituent principal components, after which certain rows from the unitary rotation matrix V are chosen as the training set and test set. We then apply LDA to these cases. The first test is three bands but with different genres. We have Ed Sheeran, Chainsmokers, and Charlie Puth. The second test is also three bands but with the same genre. I have Soundgarden, Alice in Chain, and Pearl Jam. The last test is three different genres with random bands. I have classical, jazz, and rock.

2 Theoretical Background

2.1 The Singular Decomposition

The Singular Decomposition(SVD) is a method to decompose the matrix into following:

$$A = U\Sigma V^* \tag{1}$$

where U is an m*m unitary matrix, $\sum$ is an m*n rectangular diagonal matrix with non-negative numbers on the diagonal, and V is an n*n unitary matrix. The matrices U and V^* are rotational matrices and $\sum$ is a stretching matrix. The SVD allows every matrix to be diagonal if the proper bases for the domain and the range are used. The SVD is also a least-square fitting algorithm. It allows us to project the matrix on to the low-dimensional representations.

On of the primary application of SVD is the Principle Component Analysis(PCA). Like its name, the PCA will help our analysis the most important part of the data. If the data is 2d, then the PCA will help us find the line that fits the data almost perfectly. If the data is 3d, then the PCA will give us a plane. The PCA allows the random sets of data to be reduced to lower dimensions of dynamics without any underlying behavior. Be able to help us find the bounding box is a useful part of the PCA. The key to analyzing the data is to consider the covariance matrix, which is

$$Cx = \frac{1}{n-1} X * X' = \frac{1}{n-1}\Sigma^2, \tag{2}$$

where the size is m*m. The diagonal terms of Cx are the variances for particular measurements. The key idea behind the diagonalization is there exists an ideal basis in which the Cx can be written so that in the basis, all redundancies have been removed, and the largest variances of particular measurements are ordered.

2.2 Linear Discriminant Analysis

Linear Discriminant Analysis(LDA) is a tool to lower the dimension of the data. Usually, it is a way to realize the supervised learning in machine learning. It is a way to classify the input data. So, the majority function of LDA is when project the data on the lower dimension, it will try to make the same category input data close to each other, and decrease the overlap area between different categories.

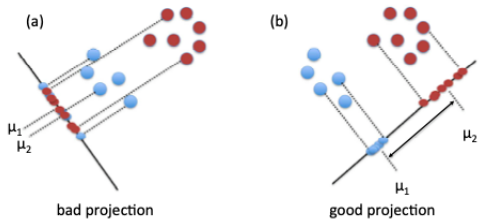

Figure 1: In the left figure, the projection produces highly mixed data and very little distinction or separation can be drawn between data sets. In the right figure, the projection produces the ideal, well-separated statistical distribution between the data sets. The goal is to mathematically construct the optimal projection basis which separates the data most effectively.

Thus the goal of LDA is two-fold: find a suitable projection that maximizes the distance between the inter-class data while minimizing the intra-class data.

For a two-class LDA, the best projection(w) that discriminates two categories is trying to let two sample centers separate.

$$w = \frac{w^T S_B w}{w^T S_W w}, \tag{3}$$

where S_B is between-class scatter and S_W is within-class scatter matrix.
Here is the equation of S_B and S_W where μ_1 and μ_2 are the mean of the two class data.

$$S_B = (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T \tag{4}$$

$$S_W = \sum_{j=1}^{2} \sum_{x} (x - \mu_j)(x - \mu_j)^T \tag{5}$$

After doing calculation, we will get that

$$S_B w = \lambda S_W w \tag{6}$$

Once the scatter matrices are constructed, the generalized eigenvectors can be constructed with MATLAB.

3 Algorithm Implementation and Development

For three tests, the algorithms are all the same.

1. Find the music and download them. Cut them into 5 second and convert .mp3 to .wav.

2. Load the music and if the music has more than one channel, convert them into the single channel.

3. Create a spectrogram of each wav file by applying Gabor transform.
4. Sum up each column in spectrogram data to make the spectrogram data only have one row.
5. Split the data into the train and test data. In this paper, the train data has ten songs and test data has five songs.
6. Take an economy singular value decomposition of the traindata by using the command svd(). We will get the U, S, V matrix.
7. Set the number for the features.
8. Multiply the S and V' matrix which get from the command svd() and call it "music".
9. New variables for three songs are generated giving the strength of each song projection on the PCA/POD modes generated by U.
10. Calculate the mean of the music matrix and the new variables of each song.
11. Calculate the Sw and Sb.
12. Applying the linear discriminant analysis by using the command eig().
13. Calculate the w(projection).
14. Take the transpose of w and multiply with the three new variables.
15. Find two thresholds since we have three groups.
16. Project the testsongs onto the SVD/PCA/POD modes selected from the training algorithm.
17. Project this decomposition onto the LDA eigenvector.
18. Determine the projection value relative to the LDA thresholds determined in the training algorithm.
19. Calculate the error and the successful rate.
20. If the successful rate is low, then adjust the number of the feature until the accuracy is pretty high.

4 Computational Results

4.1 Test1: Three Band with Different Genres

Brand1: Ed Sheeran(Blue-eyed soul)
Brand2: Chainsmokers(EDM)
Brand3: Charlie Puth(Pop-rap)

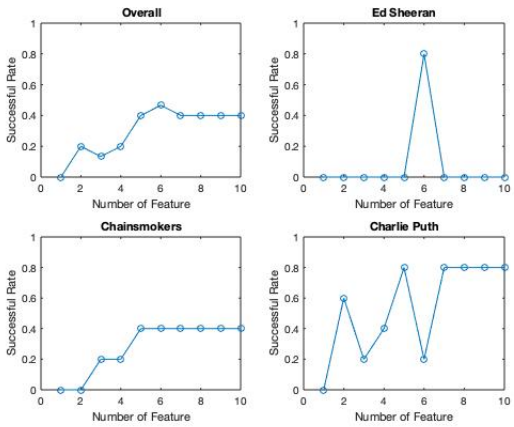

Figure 2: The number of feature versus the success rate of classification for test 1.

Figure 2 is the graph of the number of feature versus the success rate of classification. From the graph, we can tell that when we set the feature to 6 (take the first 6 charateristic of each song), we will get the highest success rate(0.4667) for this classify activity.

Figure 3 is the histogram of three songs with two thresholds when we have 1 to 6 features. The threshold that distinguish Charlie Puth and Chainsmokers is 1.8841e+04. And the second threshold that discriminate the Chainsmokers and Ed Sheeran is -4.385e+04. We can see that when we have the first six features, Chainsmokers' songs are kind of diverse since the characteristic data is spread widely and kind of uniformly. In training data, there are 7 songs of Chainsmokers that are identify correctly. Other three songs are identify as Charlie Puth's songs. So, it doesn't capture the Chainsmokers' style really well. That is the reason why we only have 0.4 accuracy for first six features. Moreover, the data in Ed Sheeran's histogram is clustering tightly. It means the these first features capture Ed Sheeran's style really well. So, when we plug in the test data, we can get pretty high accuracy. Last, for Charlie Puth, we can see that the accuracy for first six features in training data is 0.7. Same as Chainsmokers, it doesn't capture the characteristics well. So, when running the test data, we only get 0.2 for Charlie Puth's songs.

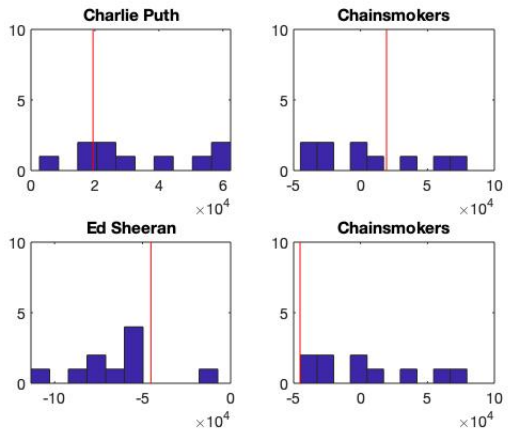

Figure 3: Test 1: The histogram of three songs with two thresholds when having first six features.

4.2 Test2: Three Band from Seattle with Same Genre

Brand1: Soundgarden
Brand2: Alice in Chains
Brand3: Pearl Jam
These three bands are all belongs to rock genre.

From figure 4, we know that when we have first two features, we will get the highest accuracy, 0.6.

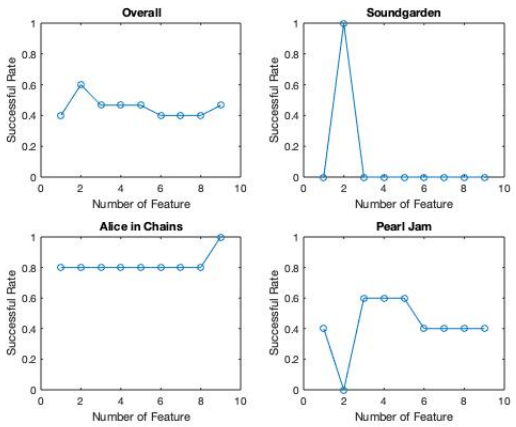

Figure 4: The number of feature versus the success rate of classification for test 2.

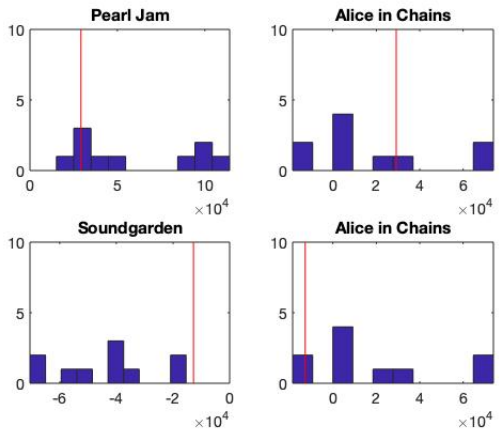

Figure 5: Test 2: The histogram of three songs with two thresholds when having first two features.

Figure 5 is the histogram of three songs with two thresholds in it when we have first two features. We can see that the threshold(2.9974e+04) that distinguish Pearl Jam and Alice in Chains doesn't perform really well. There are some overlap area for Pearl Jam and Alice in Chains. Oppositely, the second threshold(-1.131e+04) performs well to discriminate Soundgarden and Alice in Chains. From Pearl Jam's histogram, we can see that there are two groups in the graph, which indicates that it doesn't capture the characteristic well. Thus, in Pearl Jam's line graph, it has zero accuracy.

4.3 Test3: Genre Classification

Genre 1: Classical
Genre 2: Jazz
Genre 3: Rock

From figure 6, we can see that when we have first two, three, four, or five features, it will give us the highest accuracy, which is 0.6667. The threshold line between Rock and jazz is 1.6350e+04 in figure 7. And the threshold line between classical and jazz is 1.8088e+03.

From figure 7, the rock histogram has clustering tightly data, which means that it has strong characteristics of rock music. So, in the rock line graph, we have hundred percent of accuracy even though the accuracy is 0.7 in the training data. However, for jazz, it performs bad when having the first two to five features. We can verify from the jazz line graph, which the accuracy only has 0.4. Last, first two features capture classical music's characteristic fine. Even though some data are really close to each other, i.e. high bar in the histogram, there are some outliers and the range of the classical music is wide as well. So, in the classical line graph we have 0.6 accuracy for the testing data.

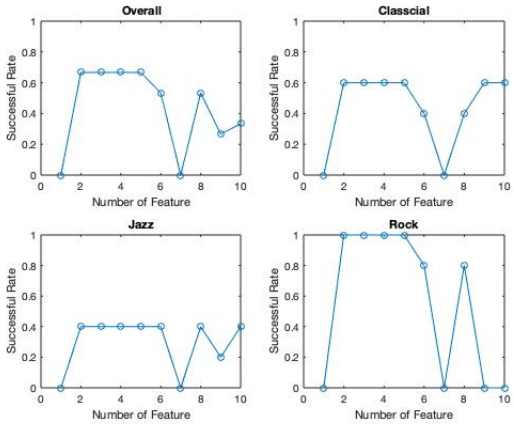

Figure 6: The number of feature versus the success rate of classification for test 2.

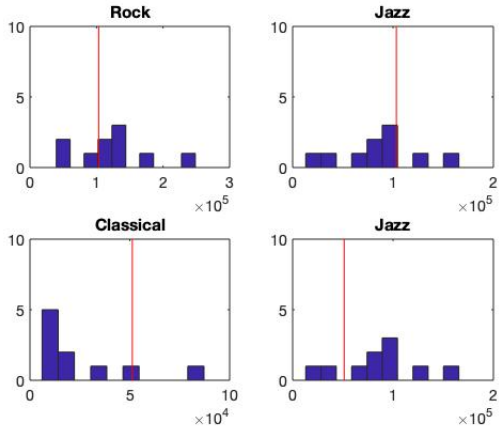

Figure 7: Test 3: The histogram of three songs with two thresholds when having first two features.

5 Summary and Conclusions

Applying the linear discriminate analysis to classify the songs is a useful tool. We calculate the projection and project the data on it to classify the songs into groups.

When we choosing different amount of feature, it will impact the success rate in some way. And it doesn't guarantee that choosing many features will perform well for the classification. For example, test 3 has two features but with

0.667 accuracy.

Since test 3 is based on the genre, not the band (if based on the band, the songs might not always have the same type), it performs the best in these three tests. The reason why the accuracy of test 1 is low is that even though the songs are from the same band but the style is varied.

6 Appendix A.

1. svd
   - Singular value decomposition
   - $[U, S, V] = svd(A)$ performs a singular value decomposition of matrix A, such that A = U*S*V'.
2. diag
   - Create diagonal matrix or get diagonal elements of matrix.
   - D = diag(v) returns a square diagonal matrix with the elements of vector v on the main diagonal.
3. max
   - Find the maxi value in the vector/matrix.
   - M = max(A) will return the max value in array A.
4. abs
   - Return an absolute value.
   - A = abs(a).
5. sort
   - Sort array elements
   - B = sort(A) sorts the elements of A in ascending order.
6. fftshift
   - Shift zero-frequency component to center of spectrum
   - Y = fftshift(X) rearranges a Fourier transform X by shifting the zero-frequency component to the center of the array.
7. length
   - Length of largest array dimension.
   - length(X) returns the length of the largest array dimension in X.
8. audioread

- Read audio file.
- $[y, Fs] = audioread(filename)$ reads data from the file named filename, and returns sampled data, y, and a sample rate for that data, Fs.

9. eig

   - Eigenvalues and eigenvectors
   - $[V, D] = eig(A)$ returns diagonal matrix D of eigenvalues and matrix V whose columns are the corresponding right eigenvectors, so that A*V = V*D.

10. fft

    - Fast Fourier transform
    - Y = fft(X) computes the discrete Fourier transform (DFT) of X using a fast Fourier transform (FFT) algorithm.

7 Appendix B.

This is the code for the test 1. Other tests have the similar code.

```
clc; close all; clear all;

Sgtvector1 = [];
a = 150;
for j = 1:15
    [y,Fs] = audioread(strcat('EdSheeran' , num2str(j) , '.wav'));
    %convert to single channel
    for i = 1:length(y)
        if (y(i,1) == 0 || y(i,2) == 0)
            monosong(i,1) = max(y(i,:));
        else
            monosong(i,1) = (y(i,1) + y(i,2))/2;
        end
    end

    v1 = monosong'/2;
    tfinal = length(v1)/Fs; t = (1:length(v1))/Fs;
    t_step = 0:0.05:tfinal;
    for i = 1:length(t_step)
        g = exp(-a*(t-t_step(i)).^2);
        Sg1 = g.*v1; Sgt1 = fft(Sg1); Sgt1 = fftshift(Sgt1);
        Sgtvector1 = [Sgtvector1; abs(Sgt1)];
    end
```

```
    edsheeran(:,j) = sum(Sgtvector1);
    Sgtvector1 = [];
end

Sgtvector1 = [];
for j = 1:15
    [y,Fs] = audioread(strcat('Chain' , num2str(j) , '.wav'));
    %convert to single channel
    monosong = [];
    for i = 1:length(y)
        if (y(i,1) == 0 || y(i,2) == 0)
            monosong(i,1) = max(y(i,:));
        else
            monosong(i,1) = (y(i,1) + y(i,2))/2;
        end
    end

    v1 = monosong'/2;
    tfinal = length(v1)/Fs; t = (1:length(v1))/Fs;
    t_step = 0:0.05:tfinal;
    for i = 1:length(t_step)
        g = exp(-a*(t-t_step(i)).^2);
        Sg1 = g.*v1; Sgt1 = fft(Sg1); Sgt1 = fftshift(Sgt1);
        Sgtvector1 = [Sgtvector1; abs(Sgt1)];
    end
    chainsmoker(:,j) = sum(Sgtvector1);
    Sgtvector1 = [];
end


Sgtvector1 = [];
for j = 1:15
    [y,Fs] = audioread(strcat('Charlie' , num2str(j) , '.wav'));
    if length(y(1,:)) == 1
        monosong = y;
    else
        %convert to single channel
        monosong = [];
        for i = 1:length(y)
            if (y(i,1) == 0 || y(i,2) == 0)
                monosong(i,1) = max(y(i,:));
            else
                monosong(i,1) = (y(i,1) + y(i,2))/2;
            end
        end
    end
```

```
    v1 = monosong'/2;
    tfinal = length(v1)/Fs; t = (1:length(v1))/Fs;
    t_step = 0:0.05:tfinal;
    for i = 1:length(t_step)
        g = exp(-a*(t-t_step(i)).^2);
        Sg1 = g.*v1; Sgt1 = fft(Sg1); Sgt1 = fftshift(Sgt1);
        Sgtvector1 = [Sgtvector1; abs(Sgt1)];
    end
    puth(:,j) = sum(Sgtvector1);
    Sgtvector1 = [];
end


traindata = [edsheeran(:,1:10), chainsmoker(:,1:10), puth(:,1:10)];
testdata = [edsheeran(:,11:15), chainsmoker(:,11:15), puth(:,11:15)];

[U,S,V] = svd(traindata, 0);

ne=length(edsheeran(1,1:10)); nc=length(chainsmoker(1,1:10)); np=length(puth(1,1
music = S*V';
suc = [];
succh = [];
succhar = [];
suced = [];
for feature = 1:9
    U1 = U(:, 1:feature);

    ed = music(1:feature, 1:ne);
    chain = music(1:feature, ne+1:ne+nc);
    char = music(1:feature, ne+nc+1:ne+nc+np);

    global_mean = mean(music(1:feature,:),2);

    med = mean(ed,2);
    mchain = mean(chain,2);
    mchar = mean(char,2);

    Sw = 0;
    for k = 1:ne
        Sw = Sw + (ed(:,k)-med)*(ed(:,k)-med)';
    end

    for k = 1:nc
        Sw = Sw + (chain(:,k)-mchain)*(chain(:,k)-mchain)';
    end
```

```
for k = 1:np
    Sw = Sw + (char(:,k)-mchar)*(char(:,k)-mchar)';
end

Sb = 1/3*(med-global_mean)*(med-global_mean)';
Sb = Sb + 1/3*(mchain-global_mean)*(mchain-global_mean)';
Sb = Sb + 1/3*(mchar-global_mean)*(mchar-global_mean)';

[V2, D] = eig(Sb,Sw);
[~,ind] = max(abs(diag(D)));
w = V2(:,ind); w = w/norm(w,2);

ved = w'*ed;
vchain = w'*chain;
vchar = w'*char;

if mean(ved) > mean(vchain)
    w = -w;
    ved = -ved;
    vchain = -vchain;
end

if mean(vchain) > mean(vchar)
    w = -w;
    vchain = -vchain;
    vchar = -vchar;
end

sorted = sort(ved);
sortchain = sort(vchain);
sortchar = sort(vchar);

t1 = length(sortchain);
t2 = 1;

while sortchain(t1) > sortchar(t2)
    t1 = t1 - 1;
    t2 = t2 + 1;
    if (t1 == 0) || (t2 == 0)
        break
    end
end

 if (t1 == 0) || (t2 == 0)
     continue
```

```
 end

threshold1 = (sortchain(t1) + sortchar(t2))/2;


t3 = length(sorted);
t4 = 1;

while sorted(t3) > sortchain(t4)
    t3 = t3 - 1;
    t4 = t4 + 1;
    if (t3 == 0) || (t4 == 0)
        break
    end
end

if (t3 == 0) || (t4 == 0)
    continue
end

threshold2 = (sorted(t3) + sortchain(t4))/2;

if feature == 6
    figure(2)
    subplot(2,2,1)
    hist(sortchar); hold on; plot([threshold1 threshold1],[0 10],'r')
    set(gca,'Ylim',[0 10],'Fontsize',[14])
    title('Charlie Puth'); hold on;
    subplot(2,2,2)
    hist(sortchain); hold on; plot([threshold1 threshold1],[0 10],'r')
    set(gca,'Ylim',[0 10],'Fontsize',[14])
    title('Chainsmokers'); hold on;
    subplot(2,2,3)
    hist(sorted); hold on; plot([threshold2 threshold2],[0 10],'r')
    set(gca,'Ylim',[0 10],'Fontsize',[14])
    title('Ed Sheeran')
    subplot(2,2,4)
    hist(sortchain); hold on; plot([threshold2 threshold2],[0 10],'r')
    set(gca,'Ylim',[0 10],'Fontsize',[14])
    title('Chainsmokers')
end

testmat = U1'*testdata;
pval = w'*testmat;

hiddenlabels = [ones(5,1);ones(5,1)*2; ones(5,1)*3]';
```

```
ResVec = [];
for i = 1:length(pval)
    if pval(i) > threshold1
        ResVec(i) = 3;
    elseif pval(i) < threshold2
        ResVec(i) = 1;
    else
        ResVec(i) = 2;
    end
end
errNum = 0;
for i = 1:length(ResVec)
    if ResVec(i) ~= hiddenlabels(i)
        errNum = errNum + 1;
    end
end
sucRate = 1-abs(errNum)/15
suc(1,feature) = sucRate;

errNum = 0;
for i = 1:5
    if ResVec(i) ~= hiddenlabels(i)
        errNum = errNum + 1;
    end
end
sucRate = 1-abs(errNum)/5
suced(1,feature) = sucRate;

errNum = 0;
for i = 6:10
    if ResVec(i) ~= hiddenlabels(i)
        errNum = errNum + 1;
    end
end

sucRate = 1-abs(errNum)/5
succh(1,feature) = sucRate;

errNum = 0;
for i = 11:15
    if ResVec(i) ~= hiddenlabels(i)
        errNum = errNum + 1;
    end
end
```

```
    sucRate = 1-abs(errNum)/5
    succhar(1,feature) = sucRate;
end

figure(3)
subplot(2,2,1)
plot(suc,'-o');
xlabel('Number of Feature'); ylabel('Successful Rate');
title('Overall');
set(gca, 'Ylim', [0 1]);
subplot(2,2,2)
plot(suced,'-o');
xlabel('Number of Feature'); ylabel('Successful Rate');
title('Ed Sheeran');
set(gca, 'Ylim', [0 1]);
subplot(2,2,3)
plot(succh,'-o');
xlabel('Number of Feature'); ylabel('Successful Rate');
title('Chainsmokers');
set(gca, 'Ylim', [0 1]);
subplot(2,2,4)
plot(succhar,'-o');
xlabel('Number of Feature'); ylabel('Successful Rate');
title('Charlie Puth');
set(gca, 'Ylim', [0 1]);
```