Music Classification

Amanda Lee

March 6, 2020

Abstract

Three music classifiers are trained using linear discriminant analysis and then tested for their success rate. The first classifier tries to identify which of three bands of different genres produced the test music. The second classifier tries to identify which of three bands in the same genre produced the test music. The third classifier tries to identify which of three genres the test music belongs to.

Introduction

The goal of this assignment is to write code to attempt to classify music using a five second sample clip. I will be making three different classifiers. The first classifier will try to identify which of three artists (Visager, Dee Yan-Kee, or 23 and Beyond the Infinite) of different genres (Electronic, Jazz, and Psych-Rock) produced the music in the sample clip. The second classifier will try to identify which of three psychedelic rock artists (Third Mind Movement, The Agrarians, or 23 and Beyond the Infinite) made the sample clip. The third classifier will try to identify which of three genres (Halloween, Christmas, or Birthday) the music in the sample clip is in.

For both the training data and the test data, I will convert the music into spectrograms. Then, I will perform PCA on the training data. Finally, I will use linear discriminant analysis to find the thresholds that separate each category.

Background

Linear Discriminant Analysis

The goal of linear discriminant analysis (LDA) is find a projection that maximizes the inter-class data while minimizing the intra-class data. Mathematically, the problem can be formulated as the following. Construct a projection w such that

$$w = \text{argmax}\, w \frac{w^T S_B w}{w^T S_w w} \tag{1}$$

where the scatter matricies for between-class S_B and within-class S_w data for 3 classes are given by

$$S_B = \sum_{j=1}^{3} m_j(\mu_j - \mu)(\mu_j - \mu)^T \tag{2}$$

$$S_w = \sum_{j=1}^{3} \sum_{x} (x - \mu_j)(x - \mu_j)^T \tag{3}$$

where μ is the mean across all classes. The projection can be found by solving the generalized eigenvalue problem.

$$S_B w = \lambda S_w w \tag{4}$$

where the maximum eigenvalue λ and its associated eigenvector gives the quantity of interest and the projection basis. The separation between classes will be defined using the distributions of each class within this projection.

Algorithm Implementation

The training and test data will be composed of five second clips from the songs chosen. I collect urls of songs from `freemusicarchive.org` in text documents for each class. I repeat the process with different songs within the same class for the testing data. I then create the matrices of training and testing data. This is done by dividing each song from each category into five second clips, then taking a random sample of one hundred clips to represent each category. I then make a spectrogram out of each clip.

To train the classifier, the spectrograms of the training data for each class is reshaped into column vectors so that each category has a matrix with the columns being the spectrogram data of each music sample. Then, I take the reduced SVD of the matrices of each category put next to each other. I take the first twenty components of the SVD as the rest of the components correspond to less important aspects. I then compute the between-class and within-class scatter matrices and find the optimal projection for LDA by solving the generalized eigenvalue problem (4).

Results

The original signals of the music I used produced matricies that demanded more RAM than my computer had. As a result, I took only every 20th sample of the signal.

The classifier for the first test with three different bands of different genres had a success rate of 55.00%.

The classifier for the second test with three different bands of the same genre had a success rate of 52.33%

The classifier for the third test with three different genres only had a success rate of 22.33%.

Conclusion

The classifiers for the first two test cases were able to correctly classify a little more than half of the sample clips, which is a better performance than if the classier guessed at random. The classifier did not have too much trouble classifying bands within the same genre compared to classifying bands of different genres. This could be because the genre chosen for the second test case was psychedelic rock, which has a decent amount of diversity in the instruments used.

The classifier for the third case did much more poorly in comparison. In fact, the performance is worse than if the classifier guessed at random. This could be due to the diversity within the genres of Halloween, Christmas, and Birthday music. The genres are not defined by the instruments used and more defined by the moods created by the chords and other notes in the music. It is also possible that a five second clip is not sufficient for classifying a song within the genres.

Appendix A

`[y,Fs] = webread('url')` - Reads audio data from the specified url and returns the signal `y` and the sampling frequency `Fs`.

`[U,S,V] = svd(A,'econ')` - Performs the reduced SVD where `A=U*S*V'`.

`Y = fft(X)` - Computes the Fast Fourier Transform of `X`.

Appendix B

saveTrainingClips.m

```
% Get test 1 traning data
close all; clear; clc;
[trainClips1,Fs1] = getClips(loadTextFiles('visagertraining.txt'));
[trainClips2,Fs2] = getClips(loadTextFiles('dee_yan_keyTraining.txt'));
[trainClips3,Fs3] = getClips(loadTextFiles('32AndBeyondTraining.txt'));

test1specs1 = makeSpectrogram(trainClips1);
test1specs2 = makeSpectrogram(trainClips2);
test1specs3 = makeSpectrogram(trainClips3);

save('trainingData1.mat','-regexp','^test1')

%% Get test 1 testing data
close all; clear; clc;
[testClips1,Fs1] = getClips(loadTextFiles('visagerTesting.txt'));
[testClips2,Fs2] = getClips(loadTextFiles('dee_yan_keyTesting.txt'));
[testClips3,Fs3] = getClips(loadTextFiles('32AndBeyondTesting.txt'));

test1testspecs1 = makeSpectrogram(testClips1);
test1testspecs2 = makeSpectrogram(testClips2);
test1testspecs3 = makeSpectrogram(testClips3);
test1Ans = [ones(1,100) ones(1,100)*2 ones(1,100)*3];
save('testingData1.mat','-regexp','^test')
%% Get test 2 training data
close all; clear; clc;
[trainClips1,Fs1] = getClips(loadTextFiles('thirdMindMovementTraining.txt'));
[trainClips2,Fs2] = getClips(loadTextFiles('theAgrariansTraining.txt'));
[trainClips3,Fs3] = getClips(loadTextFiles('32AndBeyondTraining.txt'));

test2specs1 = makeSpectrogram(trainClips1);
test2specs2 = makeSpectrogram(trainClips2);
test2specs3 = makeSpectrogram(trainClips3);

save('trainingData2.mat','-regexp','^test2')
%% Get test 2 testing data
close all; clear; clc;
[testClips1,Fs1] = getClips(loadTextFiles('thirdMindMovementTesting.txt'));
[testClips2,Fs2] = getClips(loadTextFiles('theAgrariansTesting.txt'));
[testClips3,Fs3] = getClips(loadTextFiles('32AndBeyondTesting.txt'));

test2testspecs1 = makeSpectrogram(testClips1);
test2testspecs2 = makeSpectrogram(testClips2);
test2testspecs3 = makeSpectrogram(testClips3);
test2Ans = [ones(1,100) ones(1,100)*2 ones(1,100)*3];

save('testingData2.mat','-regexp','^test2')
%% Get test 3 training data
close all; clear; clc;
[trainClips1,Fs1] = getClips(loadTextFiles('halloweenTraining.txt'));
```

```
[trainClips2,Fs2] = getClips(loadTextFiles('christmasTraining.txt'));
[trainClips3,Fs3] = getClips(loadTextFiles('birthdayTraining.txt'));

test3specs1 = makeSpectrogram(trainClips1);
test3specs2 = makeSpectrogram(trainClips2);
test3specs3 = makeSpectrogram(trainClips3);
save('trainingData3.mat','-regexp','^test3')

%% Get test 3 testing data
close all; clear; clc;
[testClips1,Fs1] = getClips(loadTextFiles('halloweenTesting.txt'));
[testClips2,Fs2] = getClips(loadTextFiles('christmasTesting.txt'));
[testClips3,Fs3] = getClips(loadTextFiles('birthdayTesting.txt'));

test3testspecs1 = makeSpectrogram(testClips1);
test3testspecs2 = makeSpectrogram(testClips2);
test3testspecs3 = makeSpectrogram(testClips3);
test3Ans = [ones(1,100) ones(1,100)*2 ones(1,100)*3];
save('testingData3.mat','-regexp','^test')
```

loadTextFiles.m

```
function [urlList] = loadTextFiles(filename)
% Takes a list of urls in text form and creates a vector with the urls
fileID = fopen(filename, 'r');
urls = textscan(fileID,'%s','Delimiter','\n');
urls = urls{1,1};
urlList = strings(length(urls),1);

for j = 1:length(urls)
    urlList(j) = urls{j,1};
end

fclose(fileID);
end
```

getClips.m

```
function [pickedClips,Fs] = getClips(urls)
%getClips loads in music from a list of urls to songs, then cuts up each
%song into five second clips and returns 100 randomly selected clips
resampleN = 20;
sampleN = 100;
allClips = [];
targetFs = 2205;
fails = [];

for j = 1:length(urls)
    [y,Fs] = webread(urls(j));
    y = y';

    % Convert audio to mono
    if size(y,1) > 1
```

```
            y = (y(1,:) + y(2,:))./2;
        end

        % Resample data
        if mod(Fs,targetFs) == 0
            resampleN = Fs/targetFs;
            y = y(1:resampleN:end);
            Fs = Fs/resampleN;
        else
            fails = [fails j];
            continue
        end

        % Divide into 5 second clips
        nclips = round(length(y)/(Fs*5));
        clips = zeros(nclips-1,Fs*5);

        for jj = 1:nclips-1
            newClip = y((jj-1)*Fs*5+1:jj*Fs*5);
            clips(jj,:) = newClip;
        end

        allClips = [allClips; clips];
    end

    sampleInds = randsample(size(allClips,1),100);

    pickedClips = zeros(sampleN,Fs*5);
    for j = 1:length(sampleInds)
        ind = sampleInds(j);
        pickedClips(j,:) = allClips(ind,:);
    end


    end
```

makeSpectrogram.m

```
function [spectrograms] = makeSpectrogram(clips)
% Takes an array of music clips and returns an array of the corresponding
% spectrograms
Fs = 2205;
N = size(clips,1);
L = 5;
a = 50;
tslide = linspace(0,L,100);

spectrograms = zeros(length(tslide)*Fs*5,N);

for j=1:N
    v = clips(j,:);
    % Constuct time domain of signal
    t = (1:length(v))/Fs;
    % Number of samples
    n = length(v);
    % Construct Frequency Domain
    k=(2*pi/L)*[0:(n-1)/2 -(n-1)/2:-1];
    ks=fftshift(k);

    spec = zeros(length(tslide),n);

    for jj = 1:length(tslide)
        g = exp(-a * (t-tslide(jj)).^2);
        vg = g.*v;
        vgt = fft(vg);
        spec(jj,:) = fftshift(abs(vgt));
    end

    % Stretch spectrogram to a vector
    spec = reshape(spec, [length(tslide)*Fs*5,1]);

    spectrograms(:,j) = spec;
end

end
```

trainClassifier.m

```
% Train and test classifier for test 1
% Load case 1 traning data
close all; clear; clc;
load('trainingData1.mat')
%Train Classifier
nfeatures = 20;
[U,S,V,thresh1,thresh2,w,I] = trainer(test1specs1,test1specs2,test1specs3,
    nfeatures);

%Test classifier
load('testingData1.mat')
TestMat = U'*[test1testspecs1 test1testspecs2 test1testspecs3];
pval = w'*TestMat;
% 1 = Visager (electronic), 2 = Dee Yan-Kee (Jazz), 3 = 32 And Beyond
% (Psych-Rock)

ResVec = (pval<thresh1)*I(3) + (pval>thresh2)*I(1);
ResVec(ResVec == 0) = I(2);

errNum = sum(abs(ResVec-test1Ans))
sucRate = 1-errNum/300

%% Train and test classifier for test 2
% Load case 2 traning data
close all; clear; clc;
load('trainingData2.mat')
%Train Classifier
nfeatures = 20;
[U,S,V,thresh1,thresh2,w,I] = trainer(test2specs1,test2specs2,test2specs3,
    nfeatures);
```

```

%Test classifier
load('testingData2.mat')
TestMat = U'*[test2testspecs1 test2testspecs2 test2testspecs3];
pval = w'*TestMat;
% 1 = Third Mind Movement, 2 = The Agrarians, 3 = 23 and Beyond

ResVec = (pval<thresh1)*I(3) + (pval>thresh2)*I(1);
ResVec(ResVec == 0) = I(2);

errNum = sum(abs(ResVec-test2Ans))
sucRate = 1-errNum/300

%% Train and test classifier for test 3
% Load case 3 traning data
close all; clear; clc;
load('trainingData3.mat')
%Train Classifier
nfeatures = 20;
[U,S,V,thresh1,thresh2,w,I] = trainer(test3specs1,test3specs2,test3specs3,
    nfeatures);

%Test classifier
load('testingData3.mat')
TestMat = U'*[test3testspecs1 test3testspecs2 test3testspecs3];
pval = w'*TestMat;
% 1 = Halloween, 2 = Christmas, 3 = Birthday

ResVec = (pval<thresh1)*I(3) + (pval>thresh2)*I(1);
ResVec(ResVec == 0) = I(2);

errNum = sum(abs(ResVec-test3Ans))
sucRate = 1-errNum/300
```

trainer.m

```
function [U,S,V,thresh1,thresh2,w,I] = trainer(specs1,specs2,specs3,nfeatures)
N1 = size(specs1,2);
N2 = size(specs2,2);
N3 = size(specs3,2);

[U,S,V] = svd([specs1 specs2 specs3],'econ');

songs = S*V';
U = U(:,1:nfeatures);
rspecs1 = songs(1:nfeatures,1:N1);
rspecs2 = songs(1:nfeatures,N1+1:N1+N2);
rspecs3 = songs(1:nfeatures,N1+N2+1:N1+N2+N3);

m1 = mean(rspecs1,2);
m2 = mean(rspecs2,2);
m3 = mean(rspecs3,2);
m = mean([rspecs1 rspecs2 rspecs3],2);

```

```
Sw = 0;
for k=1:N1
    Sw = Sw + (rspecs1(:,k)-m1)*(rspecs1(:,k)-m1)';
end
for k=1:N2
    Sw = Sw + (rspecs2(:,k)-m2)*(rspecs2(:,k)-m2)';
end
for k=1:N3
    Sw = Sw + (rspecs3(:,k)-m1)*(rspecs3(:,k)-m3)';
end

% Make scatter matrices
Sb = (m1-m)*(m1-m)';
Sb = Sb + (m2-m)*(m2-m)';
Sb = Sb + (m3-m)*(m3-m)';

% Solve generalized eigenvalue problem
[v,D] = eig(Sb,Sw);

% Get the largest eigenvector
[~,ind] = max(abs(diag(D)));
w = v(:,ind);
w = w/norm(w,2);

% Project on to the eigenvetor
vspecs1 = w'*rspecs1;
vspecs2 = w'*rspecs2;
vspecs3 = w'*rspecs3;

% Compute threshold
vspecs = [vspecs1;vspecs2;vspecs3];
means = [mean(vspecs1) mean(vspecs2) mean(vspecs3)];
[~,I] = maxk(means,3);
right = sort(vspecs(I(1),:));
mid = sort(vspecs(I(2),:));
left = sort(vspecs(I(3),:));

tl1 = length(left);
tl2 = 1;
while left(tl1) > mid(tl2)
    tl1 = tl1-1;
    tl2 = tl2+1;
end
thresh1 = (left(tl1)+mid(tl2))/2;

tr1 = length(mid);
tr2 = 1;
while mid(tr1) > right(tr2)
    tr1 = tr1-1;
    tr2 = tr2+1;
end
thresh2 = (mid(tr1)+right(tr2))/2;
end
```