Music Classifier

Aidan Sleavin

Humans can quickly and easily discern what genre of music a piece belongs to by just listening to a short clip. But how are computers able to classify where a piece of music belongs? To do this they must have a sample of many pieces from each genre of music it is trying to classify. This can be done by feeding a computer spectrograms of many pieces of music along with the label of a genre, with the goal of predicting the classification of a piece of music with no label.

Overview

The goal is to be able to take three genres of music then feed the computer one piece and have it classified in what genre it belongs to. To do this multiple 5 second samples from each genre are taken for classification. The three genres selected are, EDM, Rock, and Musical Oprah. For EDM all music used is from No Copyright Sounds (NCS), for Rock Tom Petty and the Heartbreakers is used, and for Musical Oprah Philip Glass's Einstein on the Beach is used. As a note all Tom Petty and Philip Glass music used was purchased as to not be used illegally. All NCS music is free to use for personal projects and does not need a license.

The clips that are selected are 5 second clips in three random places from three songs of each genre. The breakdown of the song goes as, for EDM, My Heart, Invincible, and Specter. For Rock, American Girl, I Need to Know, and Running Down a Dream. And for Musical Oprah, Knee Play 1, Spaceship, and Trial 1.

Theory

To best classify music the computer has an easier time classifying by the spectrogram which is a 2D item, rather than classifying by the 1D audio signal itself. This means that really what is being classified is the spectrograms of audio rather than the actual audio signal itself. For the classification process the Singular Value Decomposition (SVD) is taken of each spectrogram. The SVD is a process that takes a matrix A and creates three new matrices U, S, V such that $A = U * S * V'$. These matrices reconstruct A and can be used for dimensionality reduction. To get a close approximation of A only pieces of U, S, and V are needed. On the diagonal of S are the eigen values of A ordered such that the largest is top left and smallest bottom right. The value of the diagonal of S correspond to how important the corresponding eigenvector in V is to recreate the matrix. To show this figure 1 has four

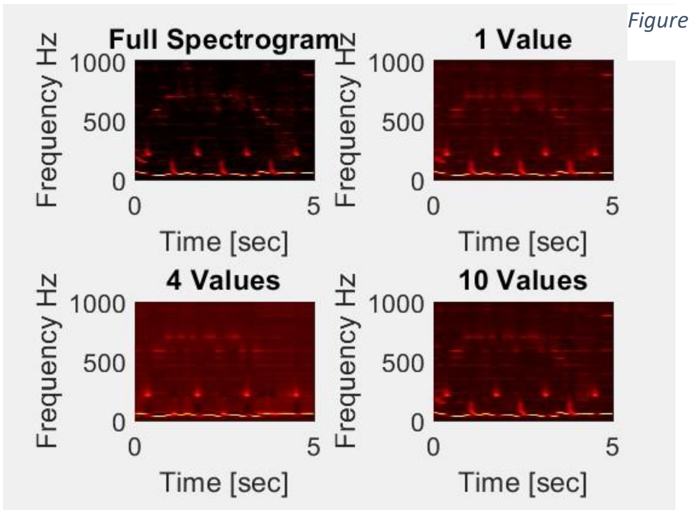


Figure 1

spectrograms, the first is the full A and each subsequent one uses only pieces of U, S, and V to reconstruct A with the formula A=U(:,1:V)*S(1:V,1:V)*V(:,1:V)' where V is the number of values taken, and a ":" represents taking all values in the row or column. It can be seen by just taking the first value almost the entirety of the spectrogram is stored in two vectors and one value.

The value and by default the importance of each of the 151 vectors can be seen in figure two which show the first value has a much larger importance than the rest. It can be seen that the first value is over twice as large as the next one and after about 20 the relevance of each value is severely decreased. Though it would take in this case all 151 values to completely reconstruct the spectrogram the first 10 do a great job.

With the principal components known the process of Linear Deskinment Analysis (LDA) to deem where is space these can be classified. For the most part similar spectrograms (and thus similar genres) will have components that lie close to each other, this is how the computer is able to classify. By setting bounds that when a new recording is analyzed it takes the area which is closest to that of a test genre.

Though a computer is being asked to classify the spectrograms it is also good to see how well a human eye can do. In figure three a spectrogram is taken from each of the three genres. And a difference can be seen. The EDM music (top left) which has a very strong bass and consistent beats can be seen to have a lot of frequencies in the lower region. The Oprah (top right) which has a lot of higher

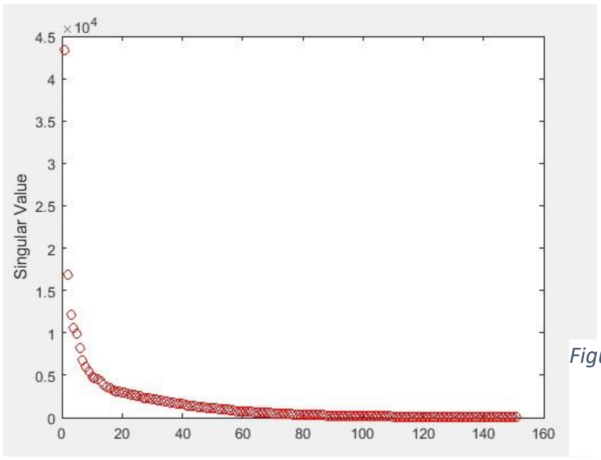
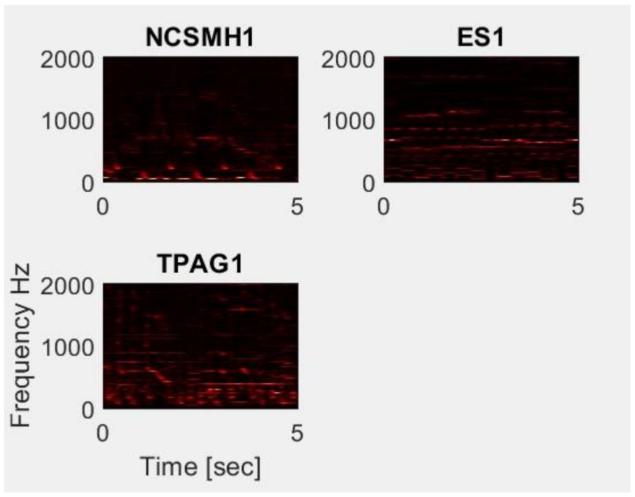


Figure 2



pitched music can be recognized as well. And the Tom Petty (bottom left) has many different frequencies as there is voice as well and my instruments that go into rock.

The final piece of theory is the sampling rate that went into the songs. Each song was taken at a rate of 44100 Hz which is standard. Due to the Shannon Nyquist principal the largest Hz than can be analyzed through this method is half of the sample rate, or 22050 Hz which is about as high as a human can hear. Though after looking at the spectrograms most of the data is held below the 5000 Hz region meaning that not all of the signal taken really needs to be sampled. Sampling less of the song means that all of the computation can go much faster, so at equal intervals two thirds of all the data in the music was picked out, leaving a lower audio quality but much more computationally easy. The new sample rate of the songs is 14700 Hz meaning audio up to 7350 Hz is being analyzed by the computer in this code.

Figure 3

Implementation

Lines 1-15 are based around reading in the audio file and setting it up to best manipulate. On line 4 the variable cut is introduced where the amount of the signal kept is 1/cut. Because the audio in an MP3 file is stereo (each speaker plays something different to get a surround sound) only the audio from one is taken as they are similar signals and enough to play the music. Lines 9-11 cut out three 5 second clips of audio at the one fourth, halfway, and three fourths marks in the song.

Lines 20-30 create the spectrogram by taking a Fourier transform for each point in time where the signal around that time point are magnified by a gaussian. The wider the gaussian is (as changed by a on line 20) the less data is known about time and the more is known about the overall frequency of the signal. The value was played with until a satisfactory spectrogram was created. This spectrogram is then saved to the name of a clip then repeated for each of the audio clips.

Lines 44-50 are optional lines left in the code than can display the spectrogram visually using the pcolor plotting function in MATLAB.

From line 51 on the code takes the SVD of all of the spectrogram together and orders them based on the genre, this is done to classify the data on lines 53-56. The singular values are then associated with each genre based on how many from clips from each genre are being analyzed and which parse of U, S, V correspond to each.

<u>Results</u>

Based on the data that was collected it did not seem that there was enough to accurately determine the genre a new piece belonged to. This could be due to a few factors which will be discussed. Number one is the amount of samples taken, more test data always lends to better accuracy as one anomaly will stand out much less about the others. I a best case more music clips would have been collected for the tests. But due to the difficulty in finding legal free music, or purchasing music this did not happen.

Another factor could be the similarity between each genre, in this test it seems by listening that Tom Petty and the Heartbreakers is the outlier from the other two genres but not by a lot. Much of the music across Einstein on the Beach and NCS was electronic in origin and had a steady beats per minute that was strong. As well as much less vocals than most music.

And lastly the precision of the spectrograms might not have been enough. Of all the components that go into making the spectrogram, filter width, number of points, accuracy in time vs frequency domain, all were played with and adjusted to make a good-looking spectrogram. But there are only so many permutation feasible to try across all the components that it might just be that while the spectrograms might look good to the human eye it is not enough for a computer.

<u>Conclusion</u>

Though SVD is a powerful tool for machine recognition and classification the parameters in this test were not honed enough to result in consistent recognition of new music pieces. While the Spectrograms may be visually different and slightly possible to classify with the human eye, machines can be much faster tools of classification.

<u>Append A:</u>

- [y,Fs]=audioread: this will read in an audio file and turn it into a vector y and return Fs the frequency the audio is captured at. Such that the time of the audio file is equal to length(y)/Fs
- fft: this takes the Fourier transform of a vector returning values in frequency space
- pcolor: this is a plotting function that let data be visualized on a 2d plane with a colored intensity.
- Shading inter: sets the type of shading to do for the plot. It interpolates the color for all pixels based on known value.
- colormap(): changes what spectrum of colors is used. In this “hot” is used for the entire paper.

- [U,S,V] = svd(A): does the singular value decomposition of the matrix A returning the matrices U, S and V

<u>Append B:</u>

```
clear; close all; clc;
%%
[y,Fs] = audioread("ETrial1.MP3");
cut=3;
Fs=Fs/cut;
y=y(:,1);
y=y(1:cut:end);
time=5;
y1= y(  length(y)/4:  length(y)/4+time*Fs);
y2= y(2*length(y)/4:2*length(y)/4+time*Fs);
y3= y(3*length(y)/4:3*length(y)/4+time*Fs);
t=linspace(0,5,length(y1)+1); t=t(1:length(y1));t=t(:);
k=(1/5)*[0:(length(y1)-1)/2 -(length(y1)-1)/2:-1]; ks=fftshift(k);

%%
p8 = audioplayer(y2,Fs);
playblocking(p8);

%%
a = 150;
times=150;

tslide=0:time/times:time;
Vgt_spec = zeros(length(tslide),length(y1));

for j=1:length(tslide)
```

```
   g=exp(-a*(t-tslide(j)).^2);
   Vg=g.*y3;
   Vgt=fft(Vg);
   Vgt_spec(j,:) = fftshift(abs(Vgt)); % We don't want to scale it
end

ET3 =Vgt_spec(:,:);
%%
%subplot(2,2,1);
pcolor(tslide,ks,ES1.')
shading interp
set(gca,'Ylim',[0 2300],'Fontsize',16)
colormap(hot)
%xlabel("Time [sec]"); ylabel("Frequency Hz");
title("NCSMH1");
%%
feature = 20;
[U,S,V] = svd([
   NCSI3, NCSI2, NCSI1, NCSMH1, NCSMH2, NCSMH3,
   ES1, ES2, ES3 , ET1, ET2, ET3,
   TPAG1, TPAG2, TPAG3 , TPRD1, TPRD2, TPRD3   ], 'econ');
songs = S*V';
U = U(:, 1:feature);
nNCS = 6; nE= 6; nTP= 6;
NCS = songs(1:feature, 1:nNCS);
E = songs(1:feature, nNCS+1: nNCS+nE);
TP = songs(1:feature, nNCS+nE+1: nNCS+nE+nTP);
mNCS= mean(NCS,2);
mE= mean(E,2);
```

```
mTP= mean(TP,2);
hold on
plot(1:nNCS              , NCS(1:feature,1:nNCS )  ,'ro');
plot(nNCS+1:nE+nNCS       , E(1:feature,1:nE)         ,'bo');
plot(1+nE+nNCS:nTP+nE+nNCS, TP(1:feature,1:nTP)      ,'go');



%%
Sw=0;
for k=1:nNCS
    Sw = Sw + (NCS(:,k)-mNCS)*(NCS(:,k)-mNCS)';
end
for k=1:nE
    Sw = Sw + (E(:,k)-mE)*(E(:,k)-mE)';
end
for k=1:nTP
    Sw = Sw + (TP(:,k)-mTP)*(TP(:,k)-mTP)';
end

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

vNCS = w' * NCS;
vE = w' * E;
vTP = w' * TP;

```

```
%%
hold on
[U,S,V] = svd(TPAG1, 'econ');
plot(21:40,V(1:20,1:30),'bo')
[U,S,V] = svd(ES1, 'econ');
plot(41:60,V(1:20,1:30),'go')
```