AMATH 582 Homework 3

Jesse Dumas

September 14, 2018

This is a discussion on the development of a music genre identification tool built using principal component analysis (PCA) and singular value decomposition (SVD) methods. Songs from the metal, jazz, and Korean pop (K-pop) genres were cut into 5-second samples which were converted to spectrograms after Gabor transforming. The SVD method was applied to the spectrograms to extract their principle components. These were used with a naive Bayes classifier to identify song clips by band and genre. When bands from three distinct genres were tested, the classifier had a 15.6% misclassification rate, as determined by k-fold cross-validation (with k=10). The classifier misidentified metal bands against each other at a rate of 37.8%, and genres were missclassified 24.4% of the time when all song clips were tested.

Introduction and Overview

Rather than feeding raw data to a classification algorithm, it is wise to first understand the structure of a data set. Principal component analysis[1] is useful for reducing high dimensional data to low rank data by finding the most important directions for use as basis vectors, and it permits building classifiers based on the most dominant features of the data. For this homework, the data came from 5-second song clips from metal, K-pop, and jazz artists.

[1] J.N. Kutz. *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*. OUP Oxford, 2013

This exploration will investigate the data set using the PCA method on three test cases:

- **(test 1) Band Classification:** Identify bands from three distinct genres. Bands: Meshuggah (metal), 2NE1 (K-pop), and Pat Metheny (jazz).
- **(test 2) The Case for Seattle:** Repeat case 1, but all three bands are from the same genre (metal).
- **(test 3) Genre Classification:** All bands are used in the training data; identify the genre of a given clip. Metal: Meshuggah, Animals as Leaders, and Gojira. K-pop: 2NE1, BIGBANG, Girls Generation. Jazz: Pat Metheny, Esperanza Spalding, and Snarky Puppy.

Theoretical Background

PCA

PCA is essentially a specific use of the SVD method. If **A** is a data matrix, like a collection of positions evolving in time, SVD will decompose it as

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

which can also be expressed as

$$[\mathbf{A}] = \begin{bmatrix} \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_n \\ \\ \end{bmatrix} \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_n \end{bmatrix}. \tag{2}$$

In equations 1 and 2, Σ is a diagonal matrix of eigenvectors comprising the singular values of **A**. The columns in **U** are orthonormal basis vectors for the space and **V** describes how each vector is projected onto the principal components in **U**[2,3]. Examining these principal components can reveal dominant features in data ideal for use with naive Bayes classifiers.

[2] J.N. Kutz. *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data.* OUP Oxford, 2013

[3] J Dumas. Amath 582 hw1, January 2017

Gabor Filter

Despite being incredibly useful in signal processing, Fourier transforms fail to capture time signals[4]. To analyze signals that vary in both the time and frequncy domains, the Gabor transform is the appropriate tool. The Gabor transform is defined as the following:

[4] J.N. Kutz. *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data.* OUP Oxford, 2013

$$G[f](t,\omega) = \tilde{f}_g(t,\omega) = \int_{-\infty}^{\infty} f(\tau)\bar{g}(\tau - t)e^{i\omega t}d\tau = (f, \bar{g}_{t,\omega}), \tag{3}$$

where $g_{t,\omega}$ is the Gabor filter kernel defined as

$$g_{t,\omega} = e^{i\omega t}g(\tau - t). \tag{4}$$

The main drawback of the Gabor filter is that, due to the Heisenberg uncertainty principle, some frequency information will be lost as a result of the selection of a time filtering window.

Naive Bayes

Naive Bayes classifiers are a set of supervised learning algorithms [5]. They assume that each pair of observations is independent, hence the "naive" term, and they utilize Bayes rule to identify clusters in data.

[5] Python sklearn naive bayes documentation. `http://scikit-learn.org/stable/modules/naive_bayes.html`. Accessed: 2017-02-20

Algorithm Implementation and Development

The song clips were made using a digital audio editor (Audacity) to ensure that all clips were the exact same length. However, there were still small discrepancies in length after they were imported to MATLAB, so the vectors storing the songs were cut to 2048 entries.

The Algorithm

1. Read in a folder of song clips from one band.
2. Loop through the song clips, downsample the clips (keeping every 75th value).
3. Transpose to a column vector and cut length to 2048.
4. Build time and frequency space for Gabor transform.
5. Rescale wavenumbers by $\frac{2\pi}{L}$ since fft assumes 2π-periodic signals.
6. Take Gabor transform of the signal to build a spectrogram, and reshape it to a column vector.
7. Combine all songs in column form into a single matrix and save it for the next step.
8. Load all songs from all bands into a single matrix.
9. SVD that matrix of spectrograms.
10. Inspect singular value energies to determine principal components of the data.
11. Use the first 7 principal components as a basis.
12. Plot the projections of song clips in principal component space (components 2-4).
13. Split data into training and testing subsets.
14. Train a naive Bayes classifier on the training data.
15. Test the naive Bayes predictor on the testing data.
16. Cross validate!

The full code for the project is presented in Appendix B.

Computational Results

Spectrograms were successfully built using the Gabor transform. An example of a typical spectrogram for a K-pop song is shown in Figure 1. Using the first 7 principal components yielded the best results. Using fewer principal components decreased the accuracy of the algorithm, and using more did not improve the accuracy. Spectrograms of the first 9 principal components are shown in Figure 2.

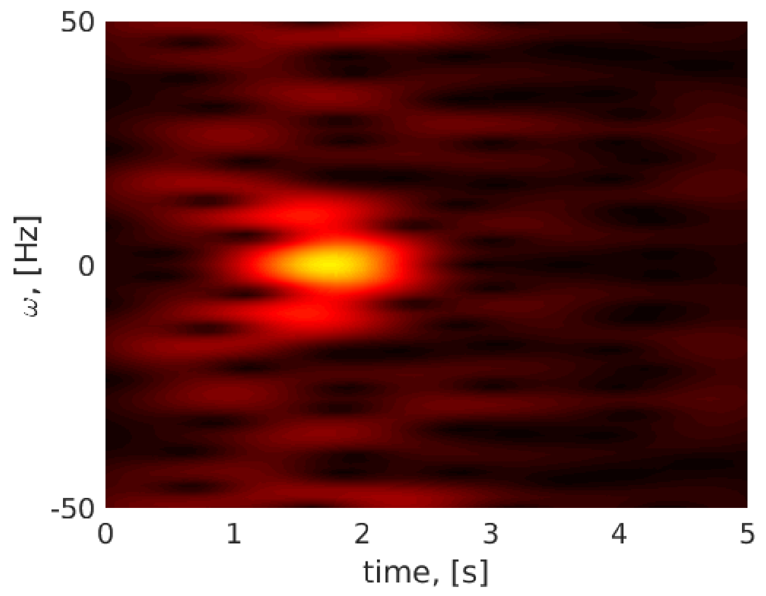

Figure 1: 2NE1's hit song "Come Back Home" in spectrogram form.

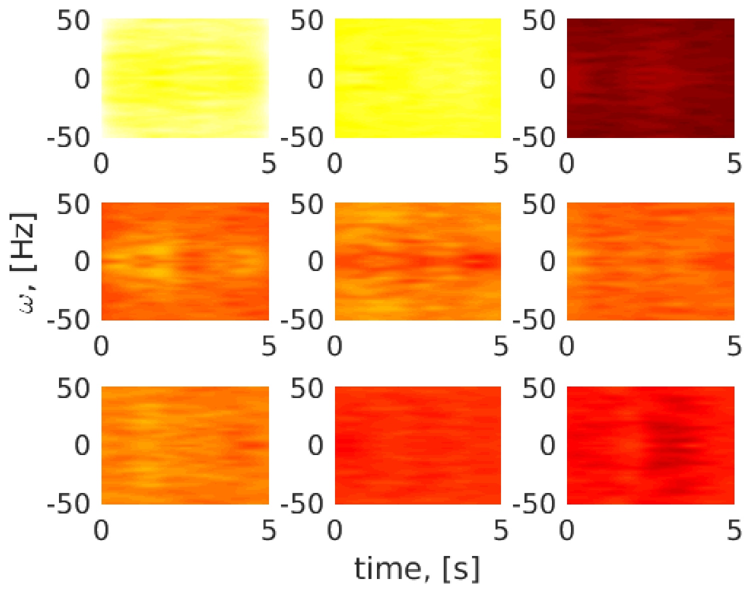

Figure 2: Spectrograms of the first 9 principal components. The spectrogram in the top left can be thought of as the "average" song. After the first 7, spectrograms start to show less unique information.

Case 1

Case 1, testing clips from Meshuggah, 2EN1, and Pat Metheny (labled 1, 2, and 3 in the data, respectively), was the most successful test case. This makes sense given the variability between bands from genres as different as metal, K-pop, and jazz. K-fold cross-validation (k=10, the default) showed that the naive Bayes classifier was only wrong about 15.6% of the time. Figure 3 shows one attempt at identifying the bands.

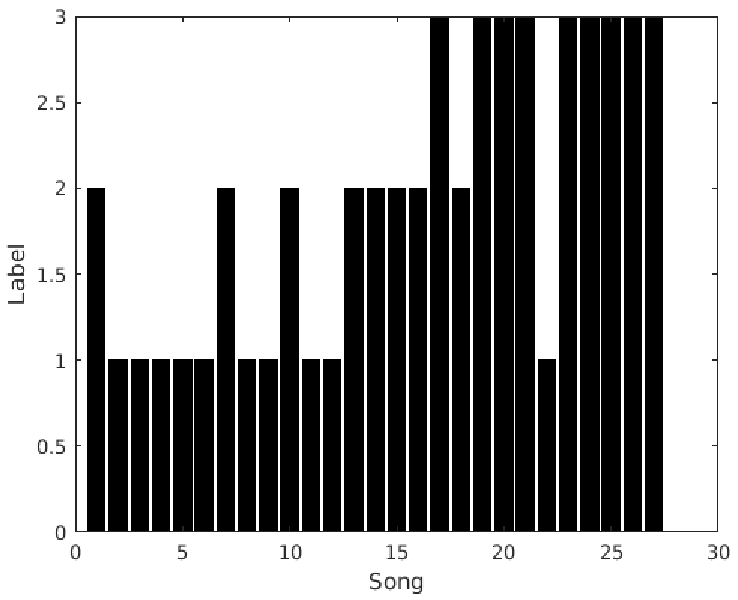

Figure 3: Naive Bayes attempt at identifying bands in 27 samples.

Case 2

Separating bands within the same genre proved much more difficult. When asked to identify whether clips were from Animals as Leaders, Meshuggah, or Gojira, the naive Bayes predictor failed 37.8% of the time, as revealed through cross-validation. Figure 4 shows how much overlap exists between the songs in principal component space.

Case 3

The performance of case 3 was between cases 1 and 2. The high variance between metal, K-pop, and jazz allowed the naive Bayes predictor to only misclassify 24.4% of the songs. Figure 9 demonstrates a high amount of overlap between the songs, but it is still somewhat clear that the jazz pieces form a fairly tight cluster.

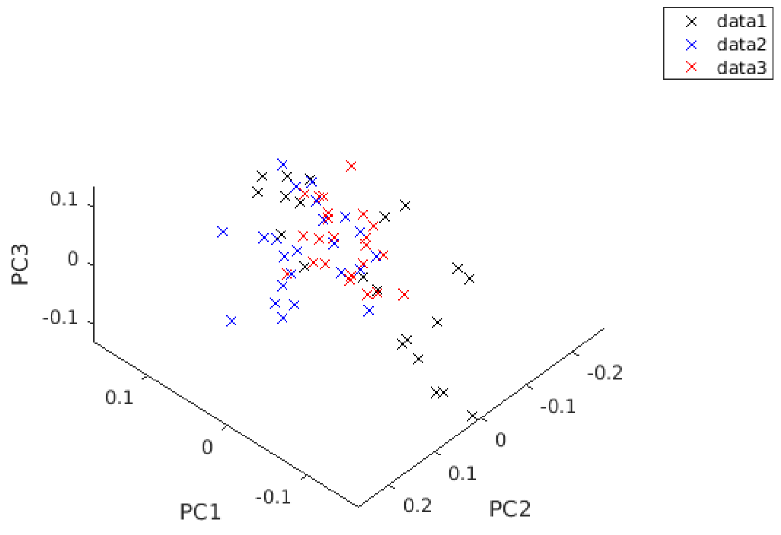

Figure 4: Metal songs projected on principal components 2-4.

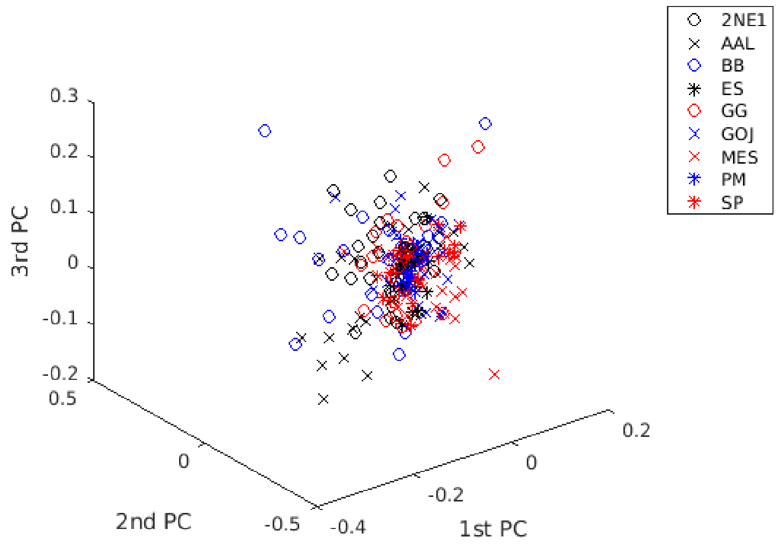

Figure 5: All song clips in principal component space. Songs with the same symbol share the same genre. A cluster of jazz songs (*) is noticeable.

Appendix C contains additional figures to save space in the actual report.

Summary and Conclusions

Building a naive Bayes classifier that could identify songs based on PCA of their spectrograms was successful. To improve performance in the future, more training data could be used, less stingy down-sampling could be employed, and more sophisticated classification methods (that do not assume feature independence) could be explored.

References

[1] J.N. Kutz. *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*. OUP Oxford, 2013.

[2] J Dumas. Amath 582 hw1, January 2017.

[3] Python sklearn naive bayes documentation. `http://scikit-learn.org/stable/modules/naive_bayes.html`. Accessed: 2017-02-20.

Appendix A: MATLAB Functions

```
double() -- converts data type to double precision floating point.
load() -- loads saved data.
size() -- returns dimensions of a matrix.
[U,S,V] = svd(A); -- performs SVD.
fft -- fast Fourier transform.
fftshift -- rescale by 2*pi/L.
downsample -- re-samples data by keeping every nth point.
audioread -- reads audio files.
pcolor -- makes contour plots used in spectrograms.
randperm(x) -- returns a random permutation of vector x.
fitcnb -- naive Bayes classifier.
kfoldloss -- k-fold loss function for cross-validation.
plot3 -- makes 3D plots.
```

Appendix B: MATLAB Code

The following script represents the code for one band, but it was changed for each band. It was used to build my data sets of spectrograms.

```
s1_1,4);
clear all; close all; clc
cd('snarky_puppy/')
files = dir('*.wav');
snk = [];
for file = files'
    [y, Fs] = audioread(file.name);

    z = downsample(y, 75);
    z2 = z(:,1);
    z3 = z2';
    z3=z3(1:2048);

    % This code is from the 582 notes by Kutz.
    L=5;
    n = length(z3);
    t2 = linspace(0,L,n+1);
    t=t2(1:n);
    k = (2.0*pi/L)*[0:n/2-1 -n/2:-1];
    ks = fftshift(k);

    Sgt_spec=[];
    tslide = 0:0.1:5;
    for j=1:length(tslide)
        g = exp(-2.5*(t - tslide(j)).^2);
        Sg = g.*z3;
        Sgt = fft(Sg);
        Sgt_spec=[Sgt_spec; abs(fftshift(Sgt))];
    end
    snk = [snk; reshape(Sgt_spec,1,104448)];
end
%
% save('/home/jes/Music/snk.mat', 'snk');
pcolor(tslide,ks, Sgt_spec.'), shading interp
set(gca,'Ylim',[-50,50],'Fontsize', [14])
colormap(hot)
%
% [u,s,v] = svd(Sgt_spec, 'econ');
%
% sigma = diag(s);
% energy = sigma/sum(sigma);
%
% % make more plots
% figure(2)
% subplot(2,1,1)
% plot(energy, 'ko','LineWidth', [1.4])
% title('Singular Values: Case 1')
% ylabel('Energy, %')
% subplot(2,1,2)
% semilogy(energy, 'ko','LineWidth', [1.4])
% ylabel('log(energy)')
% xlabel('Singular Values')
```

This is the code I used to combine all songs, perform SVD, and classify songs.

```
close all; clear all; clc

load 2ne1.mat;    % 2NE1, kpop, 1:24, ko
load aal.mat;     % Animals as Leaders, metal, 25:48, kx
load bigbang.mat; % BIGBANG, kpop, 49:72, bo
load esp.mat;     % Esperanza Spalding, jazz, 73:96, k*
load gg.mat;      % Girls Generation, kpop, 97:120, ro
load gojira.mat;  % Gojira, metal, 121:144, bx
load mes.mat;     % Meshuggah, metal, 145:168, rx
load pat.mat;     % Pat Metheny, jazz, 169:192, b*
load snk.mat;     % Snarky Puppy, jazz, 193:216, r*

% Need to use these again.
L=5;
n = 2048;
t2 = linspace(0,L,n+1);
t=t2(1:n);
k = (2.0*pi/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);

tslide = 0:0.1:5;

% Big data.
X = [twoNE1' AAL' bigbang' esp' gg' gojira' mes' pat' snk'];

% SVD
[u,s,v] = svd(X, 'econ');

% Plots
for j=1:4
    subplot(2,2,j)
    spcgrm=reshape(u(:,j),51,2048);
    pcolor(tslide, ks, spcgrm.'), axis off, shading interp, colormap(hot)
    set(gca,'Ylim',[-51,51],'Fontsize', [14])
end

pcolor(tslide, ks, reshape(X(:,77), 51, 2048).'), xlabel('time, [s]'),
ylabel('\omega, [Hz]'), shading interp, colormap(hot)
set(gca,'Ylim',[-50,50],'Fontsize', [14]) %xlabel('time, [s]'),
ylabel('\omega, [Hz]')

% Classifier
q1=randperm(24);
q2=randperm(24);
q3=randperm(24);

x2ne1= v(1:24,1:7);
xaal = v(25:48,1:7);
xbb  = v(49:72,1:7);
xesp = v(73:96,1:7);
xgg  = v(97:120,1:7);
xgoj = v(121:144,1:7);
xmes = v(145:168,1:7);
xpm  = v(169:192,1:7);
xsp  = v(193:216,1:7);

xmetal = [xaal xmes xgoj];
xkpop  = [x2ne1 xbb xgg];
xjazz  = [xesp xpm xsp];

xtrain=[xaal(q1(1:15),:); xmes(q2(1:15),:); xgoj(q3(1:15),:)];
xtest=[xaal(q1(16:end),:); xmes(q2(16:end),:); xgoj(q3(16:end),:)];

xtrain=[xmetal(q1(1:15),:); xkpop(q2(1:15),:); xjazz(q3(1:15),:)];
xtest=[xmetal(q1(16:end),:); xkpop(q2(16:end),:); xjazz(q3(16:end),:)];

ctrain=[ones(15,1); 2*ones(15,1); 3*ones(15,1)];

nb=fitcnb(xtrain,ctrain,'CrossVal','on');

pre=nb.predict(xtest);

bar(pre)

classErr = kfoldLoss(nb, 'LossFun','ClassifErr');
classErr
```

```
% Plotting code

% plot3(v(1:24,2),v(1:24,3),v(1:24,4), 'ko', 'LineWidth', [1])
% hold on
% plot3(v(25:48,2),v(25:48,3),v(25:48,4), 'kx', 'LineWidth', [1])
% hold on
% plot3(v(49:72,2),v(49:72,3),v(49:72,4), 'bo', 'LineWidth', [1])
% hold on
% plot3(v(73:96,2),v(73:96,3),v(73:96,4), 'k*', 'LineWidth', [1])
% hold on
% plot3(v(97:120,2),v(97:120,3),v(97:120,4), 'ro', 'LineWidth', [1])
% hold on
% plot3(v(121:144,2),v(121:144,3),v(121:144,4), 'bx', 'LineWidth', [1])
% hold on
% plot3(v(145:168,2),v(145:168,3),v(145:168,4), 'rx', 'LineWidth', [1])
% hold on
% plot3(v(169:192,2),v(169:192,3),v(169:192,4), 'b*', 'LineWidth', [1])
% hold on
% plot3(v(193:216,2),v(193:216,3),v(193:216,4), 'r*', 'LineWidth', [1])

% sigma = diag(s);
% energy = sigma/sum(sigma);
%
% % make more plots
% figure(2)
% subplot(2,1,1)
% plot(energy, 'ko','LineWidth', [1.4])
% title('Singular Values')
% ylabel('Energy, %')
% subplot(2,1,2)
% semilogy(energy, 'ko','LineWidth', [1.4])
% ylabel('log(energy)')
% xlabel('Singular Values')
```

Appendix C: Additional Figures

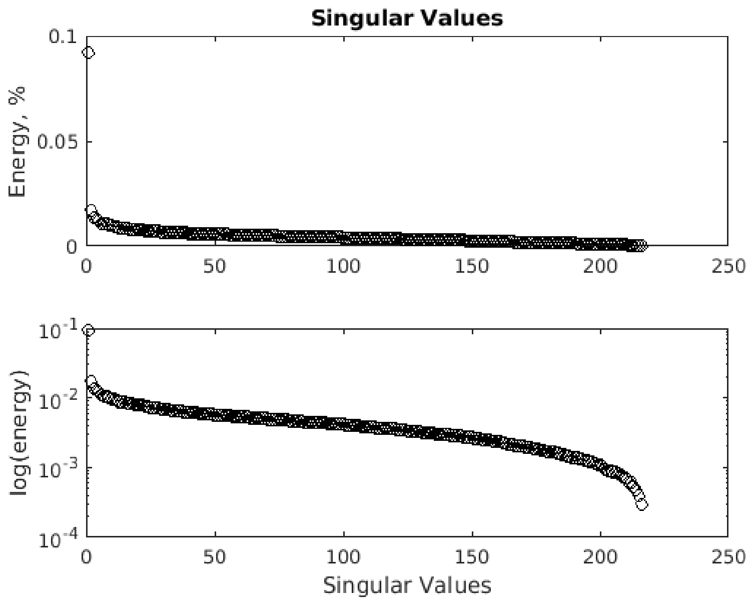

Figure 6: Singular values from the song spectrograms and their associated energies in normal (top) and semilog (bottom) plots.

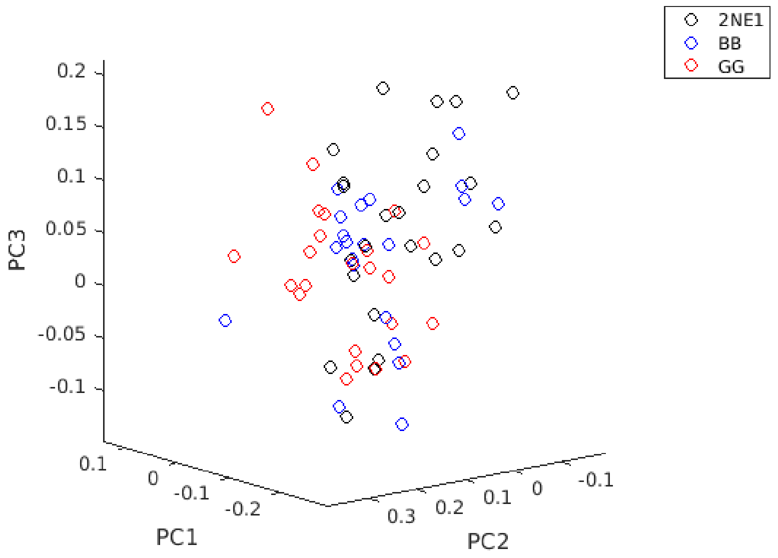

Figure 7: K-pop song clips in principal component space.

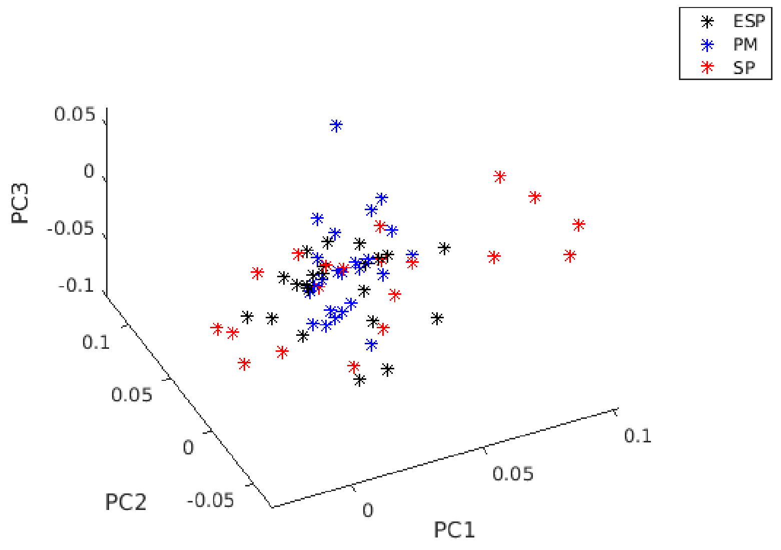

Figure 8: Jazz song clips in principal component space.

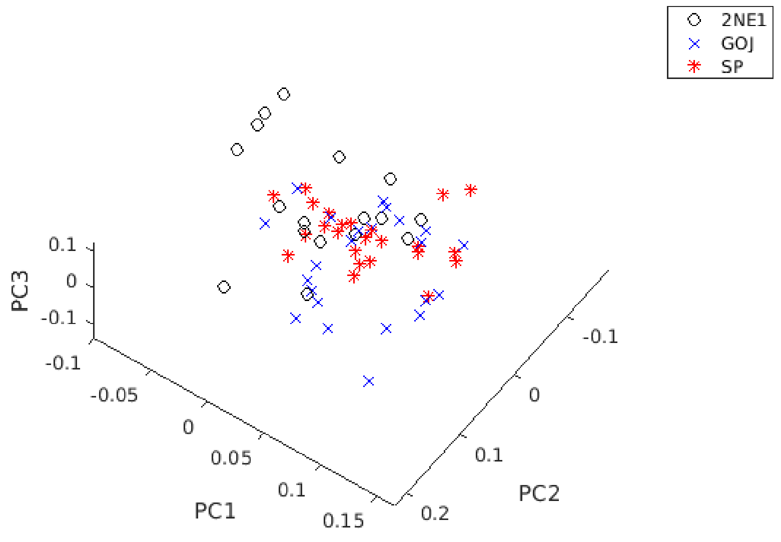

Figure 9: Song clips from 2NE1, Gojira, and Snarky Puppy in principal component space.

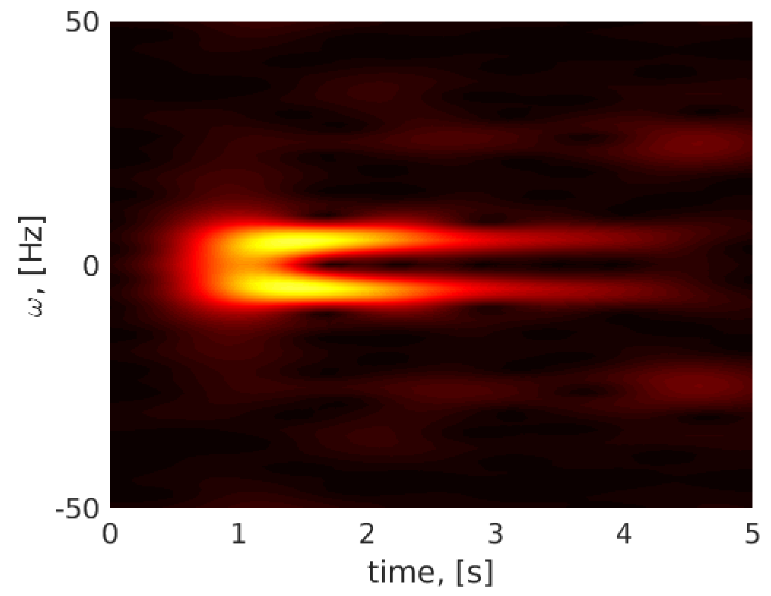

Figure 10: A song by Esperanza Spalding in spectrogram form.

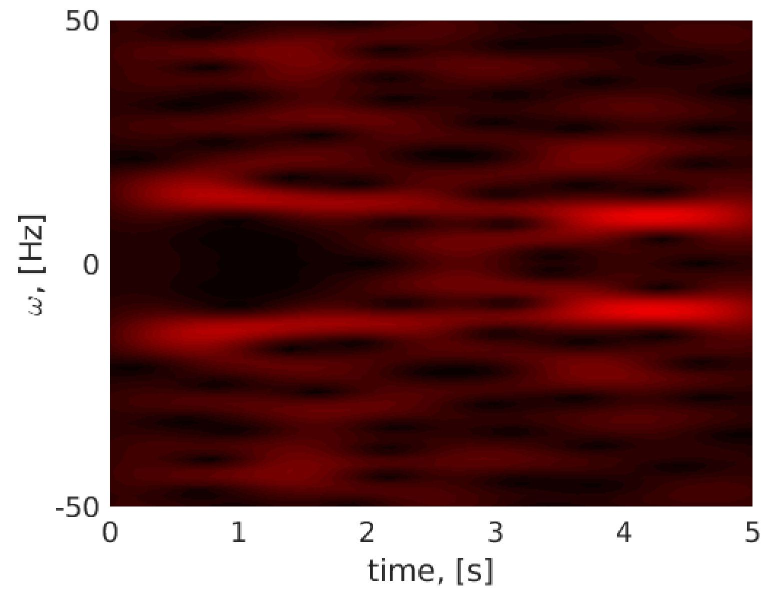

Figure 11: A song by Animals as Leaders in spectrogram form.