Computational Methods for Data Analysis

Homework 4

Due on March 16, 2020 at 5:00pm

Professor J. Nathan Kutz

Section A

Chandan Sharma Subedi

Abstract

In this paper Singular Value Decomposition (SVD) was analyzed as an unsupervised learning algorithm in the context of facial recognition and music genre classification. A Naive Bayes classification algorithm was implemented to classify the genre of a 5 sec sample of music.

1 Introduction and Overview

Singular Value Decomposition (SVD) is one of the most widely used unsupervised learning algorithm. It is also heavily used in optimization, signal analysis and numerical computation. This paper explores the fundamental idea behind SVD through its application as a learning algorithm. The first part of the paper discusses how SVD can be used for facial recognition. In this problem, SVD is applied to a large set of images to extract out dominant modes. The decomposition gives rise to a new coordinate system, also know as the set of eigenfaces. These eigenfaces are extremely effective for facial recognition.

The second part of the paper delves into the area of music genre classification. It is extremely easy for us to identify the genre of the music aftet we hear it. However, it is not clear as to how our mind processes all the information instantly. The objective is to implement and train a Naive Bayes classifier that can classify a 5 sec sample of music. The scope of the problem is limited to 3 genres of music: Classical Rock, Jazz and Grunge.

2 Theoretical Background

Singular Value Decomposition is a matrix decomposition of the form

$$\begin{bmatrix} & & \\ & \mathbf{A} & \\ & & \end{bmatrix} \begin{bmatrix} & & & & \\ \mathbf{v_1} & \mathbf{v_2} & \dots & \mathbf{v_n} & \\ & & & & \end{bmatrix} = \begin{bmatrix} & & & & \\ \mathbf{u_1} & \mathbf{u_2} & \dots & \mathbf{u_n} & \\ & & & & \end{bmatrix} \begin{bmatrix} \sigma_\mathbf{1} & & & \\ & \sigma_\mathbf{2} & & \\ & & \ddots & \\ & & & \sigma_\mathbf{n} \end{bmatrix}$$

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

The matrix $\mathbf{\Sigma}$ is an m x n diagonal matrix with positive entries provided the matrix $\mathbf{A}$ is of full rank. It stores the singular values, representing the stretching space, ordered from largest to smallest so that $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n$. The matrix $\mathbf{U}$ is a m x n unitary matrix with orthonormal columns (left-singular vectors). The matrix $\mathbf{V}$ is an n x n unitary matrix with orthonormal columns (right-singular vectors). Both of these vectors represent rotation. Furthermore, $\mathbf{v_1}$ is the vector that, when multiplied with the matrix A, strectches by a factor of $\sigma_\mathbf{1}$ and rotates so that it points along $\mathbf{u_1}$. Therefore columns of matrix $\mathbf{V}$ are the principal components of the matrix $\mathbf{A}$.

The discrete Gabor Tranform discretizes the signal through sliding window. Each window captures a piece of the signal, which is then fourier tranformed to gives its frequency spectrum. As we slide the window through the signal, we obtain a varying frequency spectrum. Combining these spectrums, we obtain a spectrogram that gives us a visual representation of how frequency spectrum varies over time in the signal.

The Gabor Transform of the function $f(t)$ is defined as

$$G_f(\tau, k) = \int_{-\infty}^{\infty} f(t)\overline{g}_a(\tau - t)e^{-ikt}dt$$

$g_a(\tau - t)$ is the sliding window obtained through the parameterization of function $g(t)$ over a scaling factor a and a shifting factor τ. The scaling factor a determines the width of the window. In this assingment, the gaussian sliding window ($g_a(\tau - t) = e^{-a(\tau - t)^2}$) was used.

When we mulitply the signal with these functions, the signal that lies inside the window remains more or less unchanged, while the signal that lies outside the window approaches zero. Thus, these functions act as a window that isolates a part of a signal at a particular instance of time.

A spectrogram is generated by combining frequency spectrums obtained through the fourier tranform of windowed signals. Thus, it gives a visual representation of how frequency spectrum is changing over time. The resolution of spectrogram, obtained through Gabor Transform, depends strongly on the sliding window and its scaling and sliding parameters. This stems from the fact that the fundamental idea of Gabor Transform is to make some compromise on spectral resolution so as to gain some temporal resolution.

Naive Bayes algorithm is a supervised classification algorithm based on Bayes theorem that assumes independence among features. The algorithm is most often used in the text classification and recommender systems. While the assumption of independence among features is highly undesirable, the algorithm benefits from its easy implementation and scalability.

3 Algorithm Implementation and Development

3.1 Eigenfaces

In order to study eigenfaces, Yale Faces B database was used. The database consisted of large sets of cropped and un-cropped grayscale images. The analysis was implemented through following steps:

- Each image was reshaped into a large column vector. The image was normalized by subtracting its mean from each element.
- The mean-subtracted image vectors were horizontally stacked to create the matrix X.
- SVD (economic) was performed on the matrix. Energy content of each mode and r-rank approximations were plotted.

3.2 Music Genre Classification

A music database was created, containing music clips from three different genre of music: Classic Rock, Grunge and Jazz. A naive bayes classifier was implemented to classify the music genre under 3 different training conditions:

- Test 1: Train NB classifier with music from three bands of different genres.
- Test 2: Train NB classifier with music from three bands of the same genre.
- Test 3: Train NB classifier with music from multiple bands of different genres.

The implementation involved three key steps:

Generate spectrogram of all music:
All music clips were loaded and re-sampled to reduce computation time. A Gabor filter, based on Gaussian sliding window, was implemented to generate the spectrogram of the clip. All spectrograms were stacked in a matrix for SVD decomposition.

Find principal components:
An economy-size SVD decompition was computed using MATLAB built in function *svd*. Energy contents of each mode were plotted to identify the number of modes n needed for classification.

Implement Naive Bayes classifier:
The music data was randomly split into testing and training datasets. A NB classifier object was created and trained using MATLAB in-built function *fitcnb*. The classifier object was used to predict the classification of test data. Error was estimated through cross-validation. Error distribution was computed by repeating the classification (with random data splitting) 100 times.

Figure 1: First 5 eigenfaces of the matrix X for cropped images dataset.

Figure 2: Rank-r approximations of the test image.

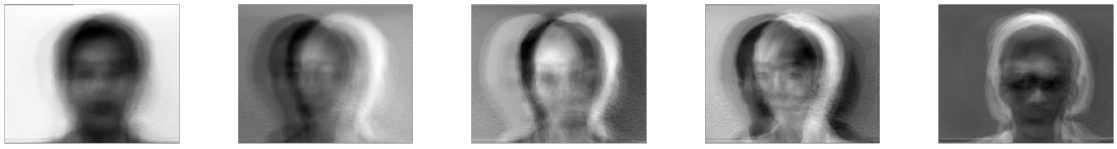


Figure 3: First 5 eigenfaces of the matrix X for un-cropped images dataset.

4 Computational Results

4.1 Eigenfaces

All the images, except one folder that was later used for test images, were transformed into the matrix as explain above. Figure 1 shows first 5 eigenfaces of cropped and un-cropped images respectively. Eigenfaces are the columns of the left-singular matrix $\mathbf{U}$. They define a new co-ordinate system. The diagonal matrix $\mathbf{\Sigma}$ contains singular values that represent the energy content of these eigenfaces, while $\mathbf{V}$ represent principal components of the matrix X.

A test image that was not in the training set was projected onto the eigenface subspace. Figure 2 shows r-rank approximations of the test image, i.e projection of test image onto the firsts r-PCA modes. It was found that for cropped image dataset, 128 modes were necessary to capture 50 % of the total energy. Similar analysis of un-cropped images produced eigenfaces as shown in the figure 3. However, they are drastically different. Due to misalignment, reflection and large background in images, SVD was forced to extract those non-facial feactures as well. Furthermore, it was found that for un-cropped image dataset, 19 modes were necessary to capture 50 % of the total energy. Those dominant modes may not represent facial features but instead, may represent (as an example) a round object in the image.

4.2 Music Genre Classification

All the music clips were downsampled by a factor of 2 to maintain a resonable computation time. The Gaussian sliding window, used for the Gabor Transform, was parameterized by the window width of $a = 100$ and sliding time-step $\tau = 0.1$. Figure 4 shows the spectrogram of all Nirvana music clips.

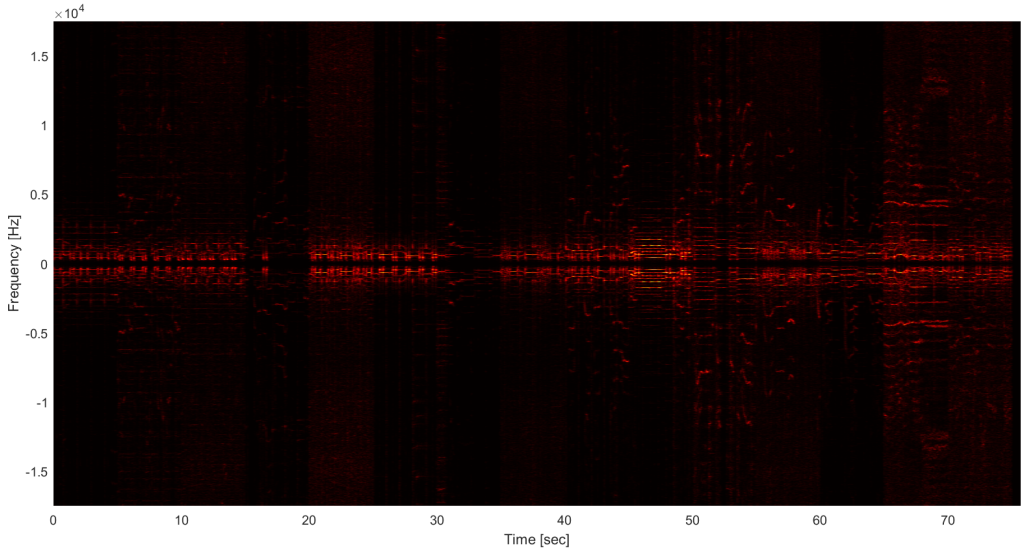
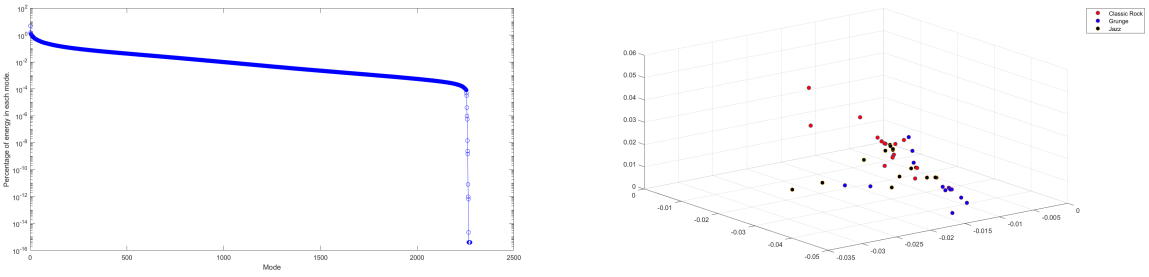


Figure 4: Spectrogram of all Nirvana music clips.

Test 1: Train NB classifier with music from three bands of different genres. Figure 5(a) shows the percentage of total energy contained in each modes. It was found that the classification error was smallest when 7 principal components were used. The accurancy of NB classifier was found to be $\approx 65\%$ with standard deviation of $\approx 10\%$. Figure 5(b) shows how much the first 3 principal components of the music of three bands from different genre are correlated. While one cluster seems to be distinct, the other two clusters were much overlap.

Test 2: Train NB classifier with music from three bands of same genre. Figure 6(a) shows the percentage of total energy contained in each modes. It was found that the classification error was smallest when 7 principal components were used. The accurancy of NB classifier was found to be $\approx 59\%$ with standard deviation of $\approx 14\%$. Figure 6(b) shows how much the first 3 principal components of the music of three bands from different genre are correlated. The clusters are very much overlapped.

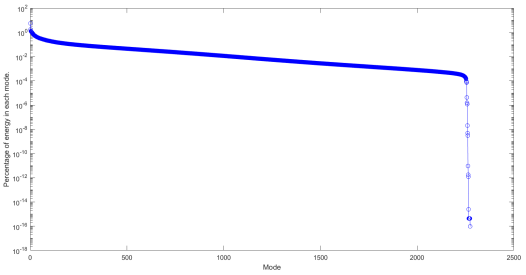
Test 3: Train NB classifier with music from multiple bands of different genres. Figure 7(a) shows the percentage of total energy contained in each modes. It was found that the classification error was smallest when 5 principal components were used. The accurancy of NB classifier was found to be $\approx 68\%$



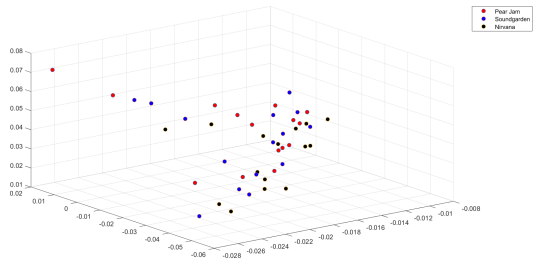
(a) Percentage of variance captured by each mode.

(b) First 3 principal components.

Figure 5: Principal Component Analysis for Test 1. NB clasifier was trained with music from three bands of different genre.



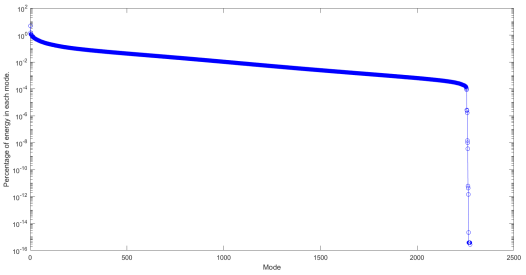
(a) Percentage of variance captured by each mode.



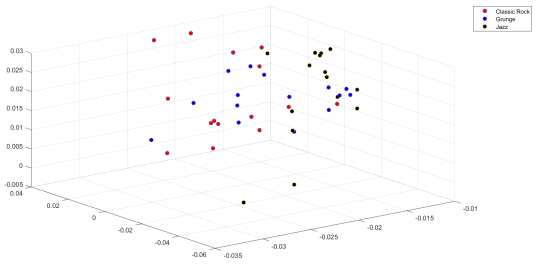
(b) First 3 principal components.

Figure 6: Principal Component Analysis for Test 2. NB clasifier was trained with music from three bands of same genre.

with standard deviation of $\approx 15\%$. Figure 7(b) shows how much the first 3 principal components of the music of three bands from different genre are correlated. Figure 8 compares the true classification with the predicted classification.

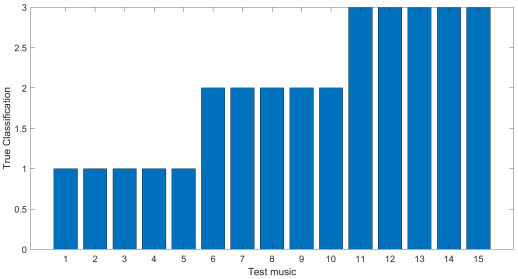


(a) Percentage of variance captured by each mode.

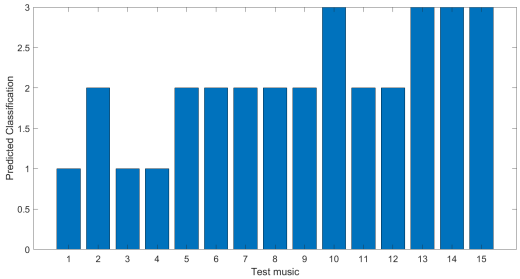


(b) First 3 principal components.

Figure 7: Principal Component Analysis for Test 1. NB clasifier was trained with music from three bands of different genre.



(a) True Classification.

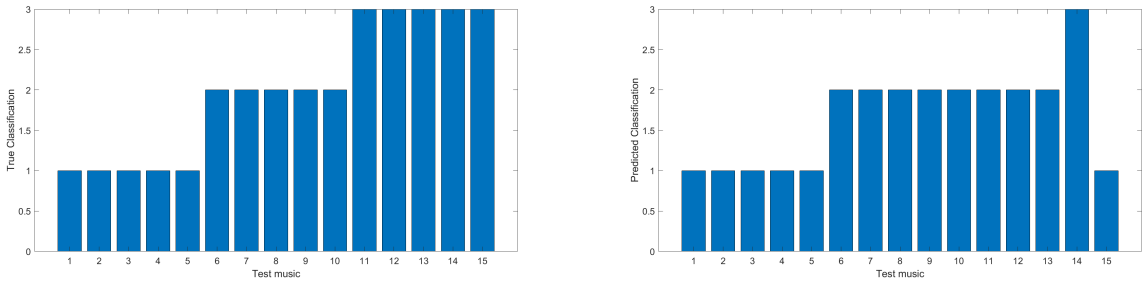


(b) Predicted Classification.

Figure 8: Naive Bayes Classification for Test 3.

5 Summary and Conclusions

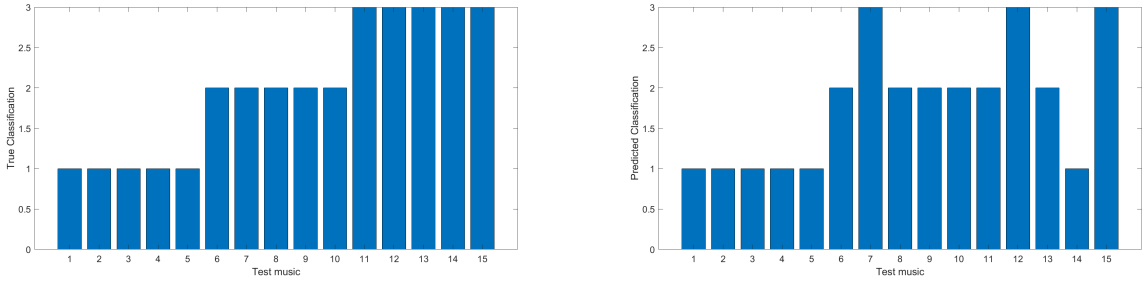
In this paper, Singular Value Decomposition (SVD) was analyzed as a learning algorithm. An image matrix, in which each column represents an image, was singularly decomposed to find the set of eigenfaces. These eigenfaces were then used to find low-dimensional approximation of a test image. Eigenfaces were generated for both cropped and un-cropped image datasets to study the effect of reflections, misalignment and background. SVD was also studied in the context of music genre classification. A Naive Bayes classification algorithm was implemented to classify a short piece of test music. The classifier was trained with three different datasets and its classification accuracy was compared. It was found that NB classifier was most acccurate when trained with music that was sampled from multiple bands across 3 genres of music. More clustered the principal components, poorer was the accuracy of classification.



(a) True Classification.

(b) Predicted Classification.

Figure 9: Naive Bayes Classification for Test 1.



(a) True Classification.

(b) Predicted Classification.

Figure 10: Naive Bayes Classification for Test 2.

Appendix A MATLAB Functions

Some important MATLAB functions used during the implementation.

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.
- `ks = fftshift(k)` rearranges FT by shifting zero frequency component to the center of the array. Exchanges two halves of the array.
- `Y = fftn(X)` returns multidimensional Fourier transform of an N-D array using FFT algorithm.
- `[U,S,V] = svd(A)` performs a singular value decomposition of matrix A, such that A = U*S*V'.
- `B = repmat(A,r1,...,rN)` specifies a list of scalars, r1,..,rN, that describes how copies of A are arranged in each dimension. When A has N dimensions, the size of B is size(A).*[r1...rN]. For example, repmat([1 2; 3 4],2,3) returns a 4-by-6 matrix.
- `plot3(X,Y,Z)` plots coordinates in 3-D space.
- `saveas(fig,filename,formattype)` creates the file using the specified file format, formattype
- `dir` lists files and folders in the current folder.
- `bar(y)` creates a bar graph with one bar for each element in y. If y is an m-by-n matrix, then bar creates m groups of n bars.
- `imagesc(C)` displays the data in array C as an image that uses the full range of colors in the colormap

Appendix B MATLAB Code

The code is published on the github repository AMATH-582 under Music Classification directory.

MATLAB code for Eigenfaces problem.

```
%% Clear Workspace
close all; clear variables; clc

%% Analysis on cropped-images
folders = dir("./Data/yalefaces_cropped/CroppedYale/yaleB*");
X = [];
for ind=1:length(folders)-1    % use last folder for test images
    files = dir([folders(ind,1).folder '/' folders(ind,1).name '/*.pgm']);

    for file=files'
        img = imread([file.folder '/' file.name]);
        img_vec = double(reshape(img,[],1));
        X = [X img_vec];
    end
end

X = X-repmat(mean(X,1),192*168,1);
[U,S,V] = svd(X, 'econ');
%% Show Eigenfaces
for i=1:5
    pause(0.1);
    imagesc(reshape(U(:,i),192,168)); colormap gray;
    set(gca,'xtick',[],'ytick',[])
    saveas(gcf, ['./Figures/cropped' num2str(i) '.png'])
end

%% Reconstruct an image
image = imread("./Data/yalefaces_cropped/CroppedYale/yaleB39/yaleB39_P00A
   +000E+45.pgm");
img_vec = double(reshape(image, [],1));
mean_img = mean(img_vec);
img_vec = img_vec - mean_img;   % Subtract mean

ranks = [25 100 200 400 1000];
for r=ranks
    img_gen = U(:,1:r)*U(:,1:r)'*img_vec + mean_img;
    imagesc(reshape(img_gen,192,168)); colormap gray;
    set(gca,'xtick',[],'ytick',[])
    title(['Rank - ' num2str(r) ' approximation.'])
    saveas(gcf, ['./Figures/r' num2str(r) '.png'])
    pause(0.2)
end
imagesc(reshape(img_vec,192,168)); colormap gray;
set(gca,'xtick',[],'ytick',[])
title(['Test Image'])
saveas(gcf, './Figures/test1.png')

%% Number of components needed to capture 50% of energy.
sigma = diag(S);
```

```
plot((sigma/sum(sigma))*100, 'b*')
energy = (sigma./sum(sigma)).*100;

e = 0;
r = 1;
while true
    if e < 50
        e = e + energy(r);
        r = r + 1;
    else
        break
    end
end

%% Clear Workspace
close all; clear variables; clc

%% Analysis on Uncropped-images
files = dir("./Data/yalefaces/subject*");
X = [];
for file=files(1:end-1,1)'
    img = imread([file.folder '/' file.name]);
    imshow(img)
    img_vec = double(reshape(img,[],1));
    X = [X img_vec];
end

X = X-repmat(mean(X,1),243*320,1);
[U,S,V] = svd(X, 'econ');

%% Show Eigenfaces
for i=1:5
    pause(0.1);
    imagesc(reshape(U(:,i),243,320)); colormap gray;
    set(gca,'xtick',[],'ytick',[])
    saveas(gcf, ['./Figures/uncropped' num2str(i) '.png'])
end

%% Number of components needed to capture 50% of energy.
sigma = diag(S);
plot((sigma/sum(sigma))*100, 'b*')
energy = (sigma./sum(sigma)).*100;

e = 0;
r = 1;
while true
    if e < 50
        e = e + energy(r);
        r = r + 1;
    else
        break
    end
end
```

MATLAB code for Music Classification.

```
%% Clear Workspace
close all; clear variables; clc

%% Global Variables
Test = 3;
Save = 0;

%% Load all music
if Test == 1
    datasets = dir("./Data/test1/*.wav");
elseif Test == 2
    datasets = dir("./Data/test2/*.wav");
elseif Test == 3
    datasets = dir("./Data/test3/*.wav");
end

spg_full = [];
for data=datasets'
    [Y, Fs] = audioread([data.folder '\' data.name]);
    t = (1:length(Y))/Fs;
    Y = downsample(Y,2)';
    t = t(1:2:end);
    N = length(Y);
    L = t(end);

    if mod(N,2) == 0
        k = (2*pi)/(L)*[0:N/2-1 -N/2:-1];      % Even N
    else
        k = (2*pi)/(L)*[0:(N-1)/2 -(N-1)/2:-1];
    end
    ks = fftshift(k);
    tslide = [0:0.1:L];
    spg = [];

    % Gabor Tranform
    for ind = 1:length(tslide)
        gauss = exp(-100*(t-tslide(ind)).^2);      % width~0.25sec
        Yf = gauss.*Y;
        Yft = fft(Yf);
        spg = [spg; abs(fftshift(Yft))];
    end
    spg_full = [spg_full; spg];
end

% SVD of the total spectrogram
[U,S,V] = svd(spg_full', 'econ');     % 45 principal modes
%% Save the spectrogram
% if Save
%     save(['./Data/test' num2str(Test) '/spectrogram.mat'], 'spg_full');
% end

%% Split the data into training and testing
V2 = V(:,1:5); % use first 5 modes
```

```
classification_errors = [];

for n=[1:100]
    group1_train = randsample(15,10);   % Training set of 30
    group2_train = randsample(15,10)+15;
    group3_train = randsample(15,10)+30;

    group1_test = setdiff(1:15,group1_train);   % Testing set of 15
    group2_test = setdiff(16:30,group2_train);
    group3_test = setdiff(31:45,group3_train);

    train = [V2(group1_train,:); V2(group2_train,:); V2(group3_train,:)];
    test = [V2(group1_test,:); V2(group2_test,:); V2(group3_test,:)];

    train_classification = [ones(10,1); 2*ones(10,1); 3*ones(10,1)];
    test_classification = [ones(5,1); 2*ones(5,1); 3*ones(5,1)];

    naive_base = fitcnb(train, train_classification);
    prediction = naive_base.predict(test);

    bar(prediction);
    set(gca, 'FontSize',14);
    xlabel('Test music'); ylabel('Predicted Classification');
    drawnow
    pause(0.1)

    error = 0;
    for ind=1:length(prediction)
        if prediction(ind) ~= test_classification(ind)
            error = error + 1;
        end
    end
    error = (error/15)*100;
    classification_errors = [classification_errors error];
end

mean_error = mean(classification_errors);
std_error = std(classification_errors);

%% Generate plots for the paper
% Energy captured in each POD modes
figure();
sigma = diag(S);
plot(sigma/(sum(sigma))*100,'-bo')
semilogy(sigma/(sum(sigma))*100,'-bo')
xlabel('Mode'); ylabel('Percentage of energy in each mode.')

% True classification bar graph
figure()
bar([ones(5,1); 2*ones(5,1); 3*ones(5,1)])
set(gca, 'FontSize',14);
xlabel('Test music'); ylabel('True Classification');

% First 3 principal components
```

```
figure();
plot3(V(1:15,1), V(1:15,2), V(1:15,3), 'o', 'MarkerFaceColor', 'r')
hold on
plot3(V(16:30,1), V(16:30,2), V(16:30,3), 'o', 'MarkerFaceColor', 'b')
hold on
plot3(V(31:45,1), V(31:45,2), V(31:45,3), 'o', 'MarkerFaceColor', 'k')
hold on
legend('Pear Jam', 'Soundgarden', 'Nirvana')
grid on

%% Spectrogram of Nirvana music
[Y, Fs] = audioread("./Data/test2/Nirvana.wav");
t = (1:length(Y))/Fs;
Y = downsample(Y,2)';
t = t(1:2:end);
N = length(Y);
L = t(end);

if mod(N,2) == 0
    k = (2*pi)/(L)*[0:N/2-1 -N/2:-1];      % Even N
else
    k = (2*pi)/(L)*[0:(N-1)/2 -(N-1)/2:-1];
end
ks = fftshift(k);
tslide = [0:0.1:L];
spg = [];

% Gabor Tranform
for ind = 1:length(tslide)
    gauss = exp(-100*(t-tslide(ind)).^2);      % width~0.25sec
    Yf = gauss.*Y;
    Yft = fft(Yf);
    spg = [spg; abs(fftshift(Yft))];
end

figure();
pcolor(tslide, ks, spg.')
shading interp
colormap(hot)
xlabel("Time [sec]")
ylabel("Frequency [Hz]")
```