Amath 482 homework 4: Music Classification

Andy Zhang

March 6, 2020

Abstract

This report demonstrates the idea of Machine Learning on music classification, which is separated into 3 cases: Band Classification with different music genres, Band Classification with the same genre, and Genre Classification. To accomplish such goal, we are going to do spectrogram transform to our data first, then apply Principal Component Analysis (PCA) for dimension reduction and Linear Discriminant Analysis (LDA) to statistically determine threshold values. Finally, the result will be applied to new data to test the algorithm.

1 Introduction

As described in the report from Homework 3, PCA is an useful technique to extract lower-dimension data from the dynamic. Since machine learning commonly deals with a bunch of similar data, applying PCA makes it much more efficient (referring to energy captured by each mode) and provides a matrix U that we can project our data onto.

LDA is based on the idea of projecting two sets of data onto a new bases that separate the inter-class distance to the max extent while making the intra-class data as compact as possible. The corresponding eigenvector of the maximum eigenvalue between the within-class variance S_W and between-class variance S_B will give us a good one to project onto.

In this report, we are going to apply these techniques to several 5-seconds music clips for classification. In order to guarantee the success rate, different amounts of clips might be chosen for different cases, since it would definitely be harder to classify three artists performing the same genre than those performing different genres.

2 Theoretical Background

2.1 Spectrogram transform

The spectrogram is the visual representation of the spectrum of frequencies along the signal verses time. We can extract time-frequency component of the music clips by simply using the Matlab built-in function:

$$spectrogram(data(:,k)) \tag{1}$$

and then rearranging it into a single column vector that represents one specific music clip.

2.2 Principal Component Analysis (PCA)

Since we're dealing with similar spectrogram data that represent the frequency-time components of music clips, it's necessary and feasible to do PCA that helps dimension reduction, and thus making the algorithm much more efficient. PCA is based on the idea of singular value decomposition, as described in the report of **Homework 3**.

2.3 Linear Discriminant Analysis (LDA)

LDA is a statistic method that can classify different groups to the max extent. In this case of music classification, we have three groups of data, thus, we should apply formula of multi-group between-class variance S_B by:

$$S_B = \sum_{j=1}^{n} mj(\vec{\mu_j} - \vec{\mu})(\vec{\mu_j} - \vec{\mu})^T \tag{2}$$

, where n denotes the number of groups and $\vec{\mu}$ is the mean of all all $\vec{\mu_1}$ to $\vec{\mu_n}$. mj is the size of each group. In this case, we're going to use same amount of clips from each artist, so we can omit it when calculating S_B.

The within-class variance, S_W can be computed by:

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

, where $\vec{x}$ represents each column vector of the projection onto principal component matrix for each group of the data.

The projection vector w is defined as:

$$w = \arg\max_{w} \frac{w^T S_B w}{w^T S_W w} \tag{4}$$

We can interpret it as the vector w that maximizes the relation of S_B over S_W. To find what the projection basis, w exactly is, we can simply find the maximum eigenvalue and the corresponding eigenvector between $S_B w$ and $S_W w$, computed by:

$$S_B \vec{w} = \lambda S_W \vec{w} \tag{5}$$

Finally, we project the data onto vector w and then need to find threshold values that help to distinguish between groups. We are going to choose the boundary values from each group and taking average of the closest two as thresholds. Since we are dealing with three data groups, two threshold values are then needed.

3 Algorithm Implementation and Development

Prepare data

To start with each case, I first load the .wav files as data for machine learning. Since the clips I choose are stereos, I then take the average and make them mono types. I set up a function called **prepareTest** that's able to do the procedures above.

Do Spectrogram transform

Another function called **doSpec** is set up for this part. First, it will spectrogram transform each column vector of the given data, taking the absolute value, and then resize the result from a m*n matrix into a m*n by 1 column vector. This part ends up by arranging each column vector into a matrix called **spec**, with the size of m*n by the amount of .wav files chosen.

Training the given data

First, we are going to choose an appropriate amount of features for dimension reduction by referring to the Energy-Mode plots for each of the three tests. Using the matrix **spec** and given feature number, we then do SVD to **spec** and choose feature amounts of columns of U and feature amounts of rows of the principal component projection matrix $S * V^T$. 1/3 of the columns of $S * V^T$ are then distributed to corresponding artist. Using the mean-of-rows vector $\vec{\mu_j}$ of each matrix, we are able to perform LDA that generate three sorted projection vectors that are used for finding 2 threshold values. This part is also stored in a function called **trainer**.

Test the algorithm on new data

To test the success rate of the algorithm, I choose one more clip of each used music for machine learning and two new clips that's not used for the algorithm. I then compute the exact labels for each of the music clip (3 being the artist with the largest mean value of the sorted projection vector, and 1 being the one with the least mean). To have the label generated by the algorithm, I first compute the spectrogram and project it onto matrix U and vector w following then. Naming the projection vector **pval**, I then call *label = (pval > threshold1) + (pval > threshold2) + 1*. Subtracting this label vector from correct labels gives us a vector of either 0 or not. If an entry is not 0, that means we fail to classify that clip. Using functions **check** generates the label and **pval**, and **checkRate** gives us the success rate in each test.

4 Computational Results

Test 1 - Band classification with different genres

Three artists, Dan Bodan, Density & Time, Dan Lebowitz are chosen for this test, each performing classical, electronics, and folk musics. 9 clips are chosen from each artist (3 clips per music piece). While for the test data, 5 clips are chosen from each, with a fourth clip from each of the three pieces used as algorithm data and 2 clips from new pieces. 8 features are applied, which results in a success rate of 0.8667, that is, 13/15 being correct. What's wrongly classified are two clips from Dan Bodan's "National Express" and "Kallimachou", and they are mis-classified to be folk music from Dan Lebowitz. The reason might be that, these two pieces are composed with instruments that generate high-frequency sounds. Since I decrease the sampling rate to 11025 Hz, they may then have similar pitches as folk musics that are mostly played by guitars.

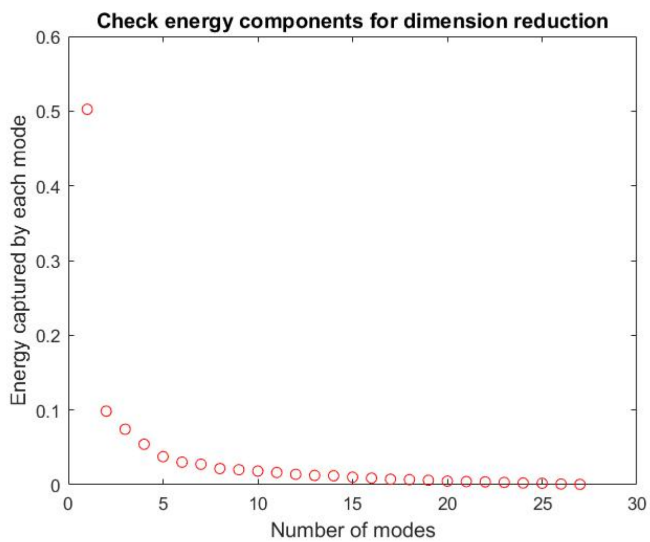

Figure 1: Energy captured by each mode in Test 1

Test 2 - Band classification with the same genre

The jazz & blues music pieces, from Quincas Moreira, Chris Haugen, and Freedom Trail Studio are used for this test. 4 clips from each of 4 pieces are chosen from each artist, which makes a total of 48 algorithm data. 6 clips from each artist (another 4 from the music pieces that are used as algorithm data and 2 from new pieces) are used for testing. In this case, I apply 20 features and the success rate is 0.7222, which means 13 out of 18 are rightly classified. The result tells us that 3 clips from Quincas Moreira and 2 from Freedom trauk studio are thought to be performed by Chris Haugen. In this case, this makes sense since jazz or blues are composed with similar chords.

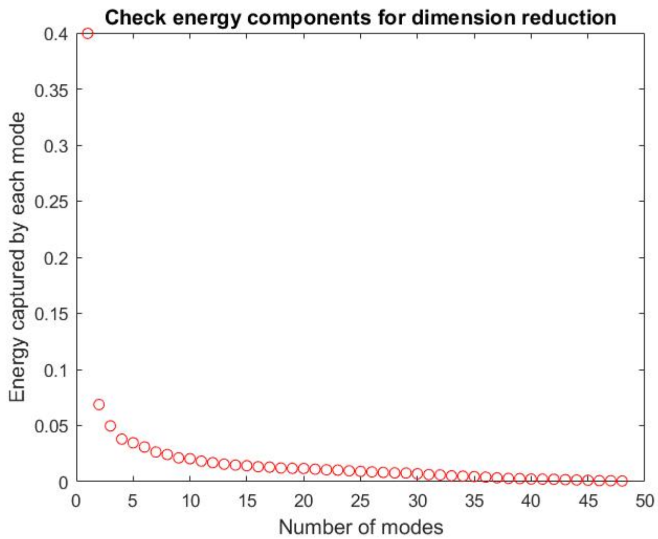

Figure 2: Energy captured by each mode in Test 2

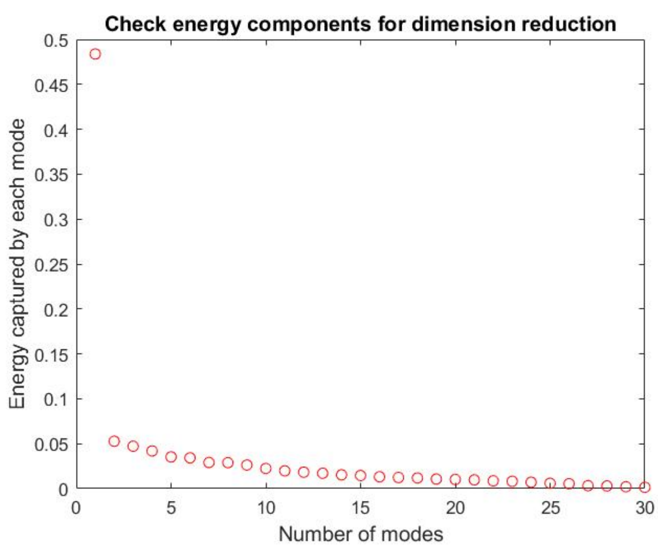

Figure 3: Energy captured by each mode in Test 3

Test 3 - Genre classification

In this case, 10 music clips are randomly chosen in the fields of Jazz & Blues, Electronics, and Punk. The test data are 4 new clips from each of the three genres. 8 features are used for the algorithm, yielding 0.8333 success rate, 10/12 being correct. One of jazz and one of electronics are thought to be punk, these two clips might also be wrongly classified due to the decreased sampling rate.

5 Summary and Conclusion

To sum up, we would see a much higher success rate when dealing with Band Classification (different genres) and Genre Classification than classifying different bands performing same genre of musics. We may also see a increase in all of the tests if we apply more input data and a higher sampling rate, however, this would also make our algorithm much slower, i.e, the tradeoff between efficiency and accuracy should be considered in the field of machine learning.

6 Appendix A

function spec = doSpec(data) returns the spectrogram of data with the right size.

function file = prepareTest(clip) reads stereo audio files and converts them to monos as a column vector.

function sucRate = checkRate(M1, M2, M3, label) checks means of each projection basis onto w and provides us a correct label. It also gives us a success rate determined by correct label and label determined by the algorithm.

function [label, pval] = check(U, w, threshold1, threshold2, spec) takes two threshold values, U and w for projections, and the spectrogram of test data and returns generated label and **pval**.

function [U, S, V, threshold1, threshold2, w, M1, M2, M3] = trainer(sp, feature) is the training function for machine learning by performing PCA and LDA.

S = spectrogram(data) produces the spectrogram by using STFT.

B = sort(A) sorts A in ascending order.

C = setdiff(A,B, 'rows') returns the values in A that are not in B with no repetitions and C will be sorted in rows.

7 Appendix B

```
%%
% Test 1: Band Classification (different genres)
clear; close all; clc;

% Dan Bodan
x1 = audioread('Anton1.wav');
x2 = audioread('Anton2.wav');
x3 = audioread('Anton4.wav');
x4 = audioread('City4.wav');
x5 = audioread('City2.wav');
x6 = audioread('City3.wav');
x7 = audioread('National1.wav');
x8 = audioread('National2.wav');
x9 = audioread('National3.wav');

x = [x1, x2, x3, x4, x5, x6, x7, x8, x9];
for i = 1 : 9
    x(:, i) = (x(:, 2*i - 1) + x(:, 2*i)) / 2;
end
x = x(:, 1 : end/2);



% Density & Time
y1 = audioread('Clean1.wav');
y2 = audioread('Clean2.wav');
y3 = audioread('Clean3.wav');
y4 = audioread('Dream4.wav');
y5 = audioread('Dream2.wav');
y6 = audioread('Dream3.wav');
y7 = audioread('Day4.wav');
y8 = audioread('Day1.wav');
y9 = audioread('Day3.wav');

y = [y1, y2, y3, y4, y5, y6, y7, y8, y9];
for i = 1 : 9
    y(:, i) = (y(:, 2*i - 1) + y(:, 2*i)) / 2;
end
y = y(:, 1 : end/2);

% Dan Lebowitz
z1 = audioread('Rag3.wav');
z2 = audioread('Rag1.wav');
```

```
z3 = audioread('Rag4.wav');
z4 = audioread('Opening4.wav');
z5 = audioread('Opening2.wav');
z6 = audioread('Opening1.wav');
z7 = audioread('Birds1.wav');
z8 = audioread('Birds2.wav');
z9 = audioread('Birds3.wav');

z = [z1, z2, z3, z4, z5, z6, z7, z8, z9];
for i = 1 : 9
    z(:, i) = (z(:, 2*i - 1) + z(:, 2*i)) / 2;
end
z = z(:, 1 : end/2);

Test1 = [x, y, z];
spec = doSpec(Test1);
feature = 8;

[U, S, V, threshold1, threshold2, w, M1, M2, M3] = trainer(spec, feature);

file(:, 1) = prepareTest('Anton3.wav');
file(:, 2) = prepareTest('City1.wav');
file(:, 3) = prepareTest('National4.wav');
file(:, 4) = prepareTest('Kallimachou1.wav');
file(:, 5) = prepareTest('Kallimachou2.wav');
file(:, 6) = prepareTest('Clean4.wav');
file(:, 7) = prepareTest('Dream1.wav');
file(:, 8) = prepareTest('Day2.wav');
file(:, 9) = prepareTest('Castlevania1.wav');
file(:, 10) = prepareTest('Castlevania2.wav');
file(:, 11) = prepareTest('Rag2.wav');
file(:, 12) = prepareTest('Opening3.wav');
file(:, 13) = prepareTest('Birds4.wav');
file(:, 14) = prepareTest('Cliffsides1.wav');
file(:, 15) = prepareTest('Cliffsides2.wav');


spec = doSpec(file);
[label, pval] = check(U, w, threshold1, threshold2, spec);
sucRate = checkRate(M1, M2, M3, label);


%%
% Test 2: Band Classification (same genre, Jazz & Blues)
```

```
clear all; close all; clc;

% Quincas Moreira
file(:, 1) = prepareTest('Brooklin1.wav');
file(:, 2) = prepareTest('Brooklin2.wav');
file(:, 3) = prepareTest('Brooklin3.wav');
file(:, 4) = prepareTest('Brooklin4.wav');
file(:, 5) = prepareTest('Infusion1.wav');
file(:, 6) = prepareTest('Infusion2.wav');
file(:, 7) = prepareTest('Infusion3.wav');
file(:, 8) = prepareTest('Infusion4.wav');
file(:, 9) = prepareTest('Malan1.wav');
file(:, 10) = prepareTest('Malan2.wav');
file(:, 11) = prepareTest('Malan3.wav');
file(:, 12) = prepareTest('Malan4.wav');
file(:, 13) = prepareTest('Paradox1.wav');
file(:, 14) = prepareTest('Paradox2.wav');
file(:, 15) = prepareTest('Paradox3.wav');
file(:, 16) = prepareTest('Paradox4.wav');


i = 16;
% Chris Haugen
file(:, 1+i) = prepareTest('Bleeker1.wav');
file(:, 2+i) = prepareTest('Bleeker2.wav');
file(:, 3+i) = prepareTest('Bleeker3.wav');
file(:, 4+i) = prepareTest('Bleeker4.wav');
file(:, 5+i) = prepareTest('Et1.wav');
file(:, 6+i) = prepareTest('Et2.wav');
file(:, 7+i) = prepareTest('Et3.wav');
file(:, 8+i) = prepareTest('Et4.wav');
file(:, 9+i) = prepareTest('Old1.wav');
file(:, 10+i) = prepareTest('Old2.wav');
file(:, 11+i) = prepareTest('Old3.wav');
file(:, 12+i) = prepareTest('Old4.wav');
file(:, 13+i) = prepareTest('Sunshine1.wav');
file(:, 14+i) = prepareTest('Sunshine2.wav');
file(:, 15+i) = prepareTest('Sunshine3.wav');
file(:, 16+i) = prepareTest('Sunshine4.wav');


% Freedom trauk studio
i = 32;
file(:, 1+i) = prepareTest('Crazy1.wav');
file(:, 2+i) = prepareTest('Crazy2.wav');
file(:, 3+i) = prepareTest('Crazy3.wav');
```

```
file(:, 4+i) = prepareTest('Crazy4.wav');
file(:, 5+i) = prepareTest('Current1.wav');
file(:, 6+i) = prepareTest('Current2.wav');
file(:, 7+i) = prepareTest('Current3.wav');
file(:, 8+i) = prepareTest('Current4.wav');
file(:, 9+i) = prepareTest('Elder1.wav');
file(:, 10+i) = prepareTest('Elder2.wav');
file(:, 11+i) = prepareTest('Elder3.wav');
file(:, 12+i) = prepareTest('Elder4.wav');
file(:, 13+i) = prepareTest('Mix1.wav');
file(:, 14+i) = prepareTest('Mix2.wav');
file(:, 15+i) = prepareTest('Mix3.wav');
file(:, 16+i) = prepareTest('Mix4.wav');


spec = doSpec(file);
feature = 20;

[U, S, V, threshold1, threshold2, w, M1, M2, M3] =
    trainer(spec, feature);

file2(:, 1) = prepareTest('Brooklin5.wav');
file2(:, 2) = prepareTest('Infusion5.wav');
file2(:, 3) = prepareTest('Malan5.wav');
file2(:, 4) = prepareTest('Paradox5.wav');
file2(:, 5) = prepareTest('Miles1.wav');
file2(:, 6) = prepareTest('Miles2.wav');

file2(:, 7) = prepareTest('Bleeker5.wav');
file2(:, 8) = prepareTest('Et5.wav');
file2(:, 9) = prepareTest('Front1.wav');
file2(:, 10) = prepareTest('Front2.wav');
file2(:, 11) = prepareTest('Old5.wav');
file2(:, 12) = prepareTest('Sunshine5.wav');

file2(:, 13) = prepareTest('Crazy5.wav');
file2(:, 14) = prepareTest('Current5.wav');
file2(:, 15) = prepareTest('Elder5.wav');
file2(:, 16) = prepareTest('Mix5.wav');
file2(:, 17) = prepareTest('Swing1.wav');
file2(:, 18) = prepareTest('Swing2.wav');

spec2 = doSpec(file2);
[label, pval] = check(U, w, threshold1, threshold2, spec2
    );
sucRate = checkRate(M1, M2, M3, label);
```

```
%%
% Test 3: Genre Classification
clear; close all; clc;

% Jazz & Blues
file(:, 1) = prepareTest('Et5.wav');
file(:, 2) = prepareTest('Miles2.wav');
file(:, 3) = prepareTest('Malan2.wav');
file(:, 4) = prepareTest('Malan3.wav');
file(:, 5) = prepareTest('Bleeker3.wav');
file(:, 6) = prepareTest('Swing2.wav');
file(:, 7) = prepareTest('Current3.wav');
file(:, 8) = prepareTest('Front2.wav');
file(:, 9) = prepareTest('Sunshine1.wav');
file(:, 10) = prepareTest('Swing1.wav');

i = 10;

% Electronics
file(:, 1+i) = prepareTest('Day3.wav');
file(:, 2+i) = prepareTest('Dream4.wav');
file(:, 3+i) = prepareTest('Clean2.wav');
file(:, 4+i) = prepareTest('Macaw1.wav');
file(:, 5+i) = prepareTest('Dragonfly1.wav');
file(:, 6+i) = prepareTest('Stranger2.wav');
file(:, 7+i) = prepareTest('Tea1.wav');
file(:, 8+i) = prepareTest('Tea2.wav');
file(:, 9+i) = prepareTest('Clean4.wav');
file(:, 10+i) = prepareTest('Macaw2.wav');

i = 20;

% Punk
file(:, 1+i) = prepareTest('Beach.wav');
file(:, 2+i) = prepareTest('Fail1.wav');
file(:, 3+i) = prepareTest('Fail2.wav');
file(:, 4+i) = prepareTest('Girl1.wav');
file(:, 5+i) = prepareTest('Girl2.wav');
file(:, 6+i) = prepareTest('Runner2.wav');
file(:, 7+i) = prepareTest('Pumps1.wav');
file(:, 8+i) = prepareTest('Radio1.wav');
```

```
file(:, 9+i) = prepareTest('Radio2.wav');
file(:, 10+i) = prepareTest('Runner1.wav');

spec = doSpec(file);
feature = 8;

[U, S, V, threshold1, threshold2, w, M1, M2, M3] =
    trainer(spec, feature);

file2(:, 1) = prepareTest('Infusion2.wav');
file2(:, 2) = prepareTest('Paradox5.wav');
file2(:, 3) = prepareTest('Elder3.wav');
file2(:, 4) = prepareTest('Old4.wav');

file2(:, 5) = prepareTest('Firefly1.wav');
file2(:, 6) = prepareTest('Cubic.wav');
file2(:, 7) = prepareTest('Operatic2.wav');
file2(:, 8) = prepareTest('Anniversary2.wav');

file2(:, 9) = prepareTest('Boy.wav');
file2(:, 10) = prepareTest('Give1.wav');
file2(:, 11) = prepareTest('Grassy.wav');
file2(:, 12) = prepareTest('Outlet1.wav');


spec2 = doSpec(file2);
[label, pval] = check(U, w, threshold1, threshold2, spec2
    );
sucRate = checkRate(M1, M2, M3, label);




%%
function spec = doSpec(data)
    for k = 1 : size(data, 2)
        temp = spectrogram(data(:, k));
        temp = abs(temp);
        [m, n] = size(temp);
        spec(:, k) = reshape(temp, m * n, 1);
    end
end

function file = prepareTest(clip)
    temp = audioread(clip);
    file = (temp(:, 1) + temp(:, 2))/2;
```

```
end


function sucRate = checkRate(M1, M2, M3, label)
    Mean = [M1, M2, M3];
    Mean = sort(Mean);
    for i = 1 : 3
        if Mean(i) == M1
            ind1 = i;
        end
        if Mean(i) == M2
            ind2 = i;
        end
        if Mean(i) == M3
            ind3 = i;
        end
    end
    correctLabel = [];
    siz = length(label)/3;
    for i = 1 : siz
        correctLabel(i) = ind1;
        correctLabel(i + siz) = ind2;
        correctLabel(i + 2*siz) = ind3;
    end

    fail = 0;
    result = correctLabel - label;
    for i = 1 : length(result)
        if result(i) ~= 0
            fail = fail + 1;
        end
    end

    sucRate = 1 - fail/length(result);
end

function [label, pval] = check(U, w, threshold1,
    threshold2, spec)
    TestMat = U' * spec;
    pval = w' * TestMat;

    label = (pval > threshold1) + (pval > threshold2) +
        1;
end

function [U, S, V, threshold1, threshold2, w, M1, M2, M3]
```

```
    = trainer(sp, feature)
[U, S, V] = svd(sp, 'econ');

PCA_proj = S * V';
U = U(:, 1 : feature);

plot(diag(S).^2/sum(diag(S).^2), 'ro')
xlabel('Number of modes', 'FontSize', 12)
ylabel('Energy captured by each mode', 'FontSize', 12)
title('Check energy components for dimension reduction',
    'FontSize', 12)

Art1 = PCA_proj(1 : feature, 1 : end/3);
Art2 = PCA_proj(1 : feature, end/3 + 1 : end * 2/3);
Art3 = PCA_proj(1 : feature, end * 2/3 + 1 : end);

m1 = mean(Art1, 2);
m2 = mean(Art2, 2);
m3 = mean(Art3, 2);

Sw = 0;
for k = 1 : size(Art1, 2)
    Sw = Sw + (Art1(:, k) - m1) * (Art1(:, k) - m1)';
end
for k = 1 : size(Art2, 2)
    Sw = Sw + (Art2(:, k) - m2) * (Art2(:, k) - m2)';
end
for k = 1 : size(Art3, 2)
    Sw = Sw + (Art3(:, k) - m3) * (Art3(:, k) - m3)';
end

mu = (m1 + m2 + m3) / 3;
Sb = 0;
Sb = Sb + (m1 - mu) * (m1 - mu)';
Sb = Sb + (m2 - mu) * (m2 - mu)';
Sb = Sb + (m3 - mu) * (m3 - mu)';

[V2, D] = eig(Sb, Sw);
[~, ind1] = max(diag(D));

w = V2(:, ind1);
w = w / norm(w, 2);

vArt1 = w' * Art1;
vArt2 = w' * Art2;
vArt3 = w' * Art3;
```

```
M1 = mean(vArt1);
M2 = mean(vArt2);
M3 = mean(vArt3);

sortArt1 = sort(vArt1);
sortArt2 = sort(vArt2);
sortArt3 = sort(vArt3);
sortArt = [sortArt1; sortArt2; sortArt3];

Mean = [M1, M2, M3];
Mmax = max([M1, M2, M3]);

IndMax = Mean == Mmax;
sortMax = sortArt(IndMax, :);

sortArt = setdiff(sortArt, sortMax, 'rows');


sortNext = sortArt(2, :);
sortLeast = sortArt(1, :);


sizeMax = length(sortArt1);
t1 = sizeMax;
t2 = 1;
while sortLeast(t1) > sortNext(t2)
    t1 = t1 - 1;
    t2 = t2 + 1;
end
threshold1 = (sortLeast(t1) + sortNext(t2)) / 2;

t3 = sizeMax;
t4 = 1;
while sortNext(t3) > sortMax(t4)
    t3 = t1 - 1;
    t4 = t4 + 1;
end
threshold2 = (sortNext(t3) + sortMax(t4)) / 2;

end
```