HW4_Music Classification

Yanjie Liu

Abstract:

The goal of this report is to examine the mechanism of the Singular Value Decomposition and the Linear Discrimination Analysis. In this report, we will figure out how to use Singular Value Decomposition and Linear Discrimination Analysis to classify different music genre. We will write our own LDA algorithm to distinguish music piece from different bands and different genre.

Introduction and Overview

In order to analyze the music genre, we download some pieces with Creative Comment from SoundCloud website. Specifically, we download five kinds of music including epic, electric, relax, classical, pop, and meditation music. For the meditation music, the three pieces are from different bands in order to meet the specific requirement of the test case.

There are basically four steps to do the SVD and LDA. Firstly, we need to decompose the original audio signal into wavelet basis functions. This motivation is simply to provide an effective method for doing edge detection. Then we need to find the principal components associated with each music genre or band. The third step is to design a statistical decision threshold for the discrimination between different genres and bands. This step is also be called as the Linear Discrimination Analysis. The last step is to test the accuracy and efficiency of our own algorithm by taking sample into the algorithm.

Theoretical Background

Singular Value Decomposition

A singular value decomposition is a factorization of a matrix into a number of constitutive components all of which have a specific meaning in applications. Different from eigen decomposition, singular value decomposition can decompose the matrix with different number of columns and rows. If we define the singular value decomposition of matrix A which is $m \times \mathrm{n}$ is:

$$A = U\Sigma W^T \tag{1}$$

U is a matrix with $m \times \mathrm{n}$ and Σ is $m \times m$, which is a diagonal matrix. And V is $n \times n$. In this equation, U and V are both unitary matrices, which means:

$$U^T U = I \tag{2}$$

$$V^T V = I \tag{3}$$

Principal Component Analysis

Principal component analysis is one of the applications of the singular value decomposition. It is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. It is usually used in low dimension like this report. In this report, the dimension is 6 which is the number of measure dimensions. Then the covariance of the single matrix X is:

$$C_X = \frac{1}{n-1} XX^T \tag{4}$$

C_X is the variance of X along its vectors. Specifically, the diagonal of C_X is the variance of corresponding measurement. The off-diagonal elements of C_X are the covariance between two different measure dimensions

Multi-class Linear Discriminant Analysis

Since the inputs of LDA algorithm include three kind of music, we need to use Multi-class Linear Discriminant Analysis instead of the two-class Linear Discriminant Analysis. Multi-class LDA is based on the analysis of two scatter matrices: within-class scatter matrix and between-class scatter matrix. Given a set of samples

$x_1,....,x_n$,and their class labels $y_1,....y_n$. The within-class scatter matrix is defined as:

$$S_w = \sum_{i=1}^{n} (x_i - \mu_{y_i})(x_i - \mu_{y_i})^T \quad (5)$$

Here, μ_k is the sample mean of the k -th class.

The between-class scatter matrix is defined as:

$$S_b = \sum_{k=1}^{m} n_k (\mu_k - \mu)(\mu_k - \mu)^T \quad (6)$$

Here m is the number of classes, μ is the overall sample mean, and n_k is the number of samples in the k -th class.

Then, multi-class LDA can be formulated as an optimization problem to find a set of linear combinations(with coefficients w) that maximizes the ratio of the between-class scattering ot the within-class scattering, as

$$\hat{w} = \arg\max_{w} \frac{w^T S_b w}{w^T S_w w} \quad (7)$$

The solution is given by the following generalized eigenvalue problem:

$$S_b w = \lambda S_w w \quad (8)$$

The maximum eigenvalue λ an its associated eigenvector gives the quantity of interest and the projection basis. Thus once the scatter matrices are constructed, the generalized eigenvectors can easily be constructed.

Algorithm Implementation and Development

Set-up

Before the start of analysis, we need to download the mp3 and wav files into computer and load them into Matlab.

Apply Wavelet Decomposition

In order to do the wavelet transformation, we need to transform the audio file into matrix. We will use for loop to pick the specific audio file from the local folder ,

sum up the sampled data, and pick a relatively smaller sample rate for further computation. Then we will pick some 5 second piece from the audio file within its length and arrange them column by column in matrix. Then this process will repeat twice for other pieces from different band or music genre. Then, we will use wavelet decomposition function also written by ourselves to decompose the above matrix and use the result as the input of PCA and LDA.

Perform PCA and LDA

The first step of PCA and LDA is the SVD decomposition of the wavelet generated data. Since there is only a limited number of useful principal components, we will set feature as 20, which already cover over 90 percent of accuracy. Then we will protect the result from the SVD onto the principal components. Then we come to the Linear Discrimination Analysis part.Since there are three classes in this case, we choose to perform multi-class LDA to solve this problem. The projection coefficient w has the same formula as the two-class LDA w. But the within-class scatter matrix and the between-class scatter matrix have slightly different formula in this case. Since there are three classes, we can not apply the method from two-class LDA to obtain the threshold. Here we will first obtain the arrangement of the mean of each classes. Then we will calculate the the threshold between the least two classes and the threshold between the largest two classes. Finally, we will get the arrangement of the three classes from lowest to highest and recognize the specific threshold between specific classes.

Test

We will repeat the previous process with different training data and do the wavelet transformation to the test data. The test data is ‘‘new’’ five second clips of music from the above genres or band. Then we will do PCA projection and LDA projection in order to compare with the threshold we get from previous step. Finally we will compute the accuracy of our algorithm.

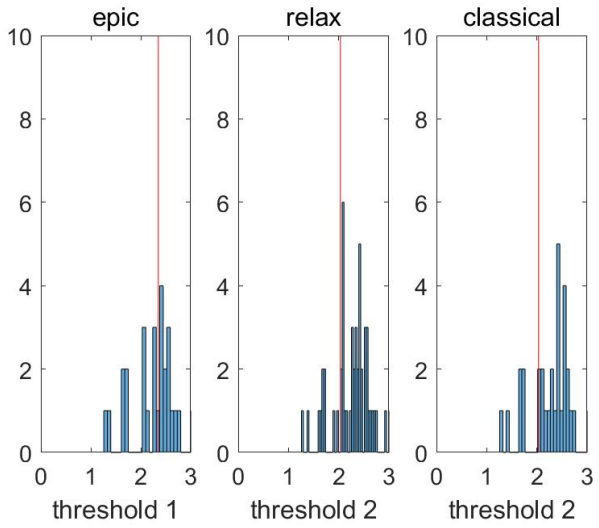

Figure 1: the histogram of the statistics associated with music data projected onto the LDA basis. The red line is the numerically computed decision threshold.

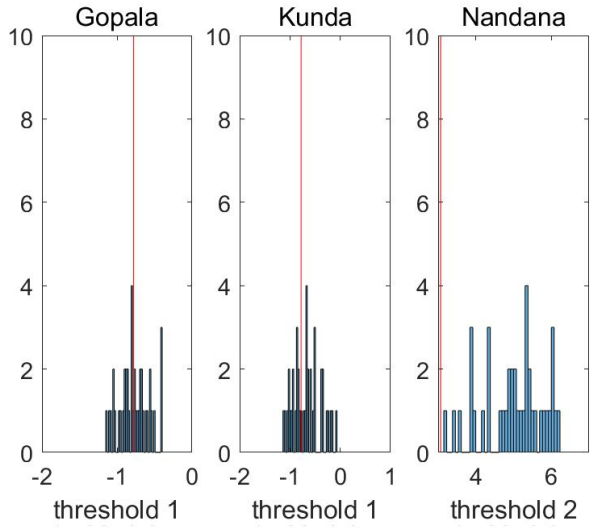

Figure 2: the histogram of the statistics associated with music data projected onto the LDA basis. The red line is the numerically computed decision threshold.

Computational Result

Test 1

In this case, we will classify three different band from three different genres. Our choices are epic music, classical music, and electric music. Figure 1 displays the result about the threshold from the specific LDA algorithm. The accuracy of this algorithm is 0.7685. The relatively poor result is heavily affected by the music genre we choose. Since the epic music, relax music, and the classical music are too similar, our algorithm can not figure out their slight differences very well.

Test 2

In this case, we will classify three different bands from the same music genre, meditation music. Figure 2 displays the result about the threshold from our own LDA algorithm. The accuracy of this test is 0.8032. This result is slightly better than test1. And we can explain this result due to the distinguishable style of one of the band we choose. From figure we can discover that the ‘music3’ is very easily recognizable from other two bands, although they all belong to meditation music.

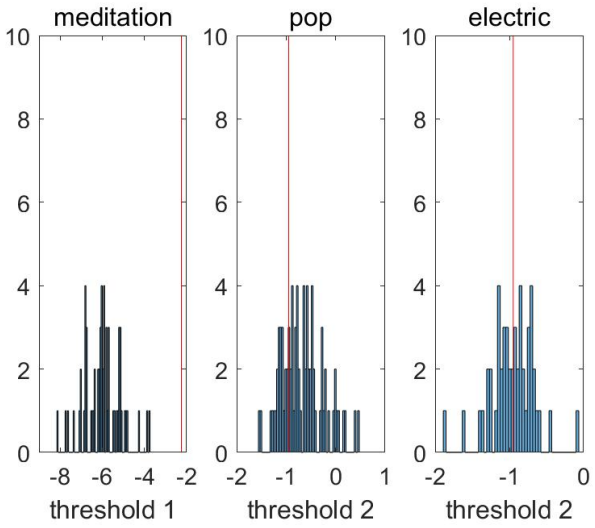

Figure 3: the histogram of the statistics associated with music data projected onto the LDA basis. The red line is the numerically computed decision threshold.

Test 3

In this case, we will classify different band from different genres. In this case, we will choose meditation music, pop music, and electric music. From figure 3 we can discover that the algorithm can easily recognize meditation music from others. But the pop music and electric music would confuse our algorithm a little bit. The accuracy in this case is 0.6032.

Summary and Conclusions

In this report , we review the principal component analysis and singular value decomposition. Also, we explore the application of linear decomposition analysis in music classification. Specifically we figure out how to build our own LDA algorithm to classify the different music genre. From the change of accuracy in three tests, we recognize that the accuracy of LDA will go down with the increasement of complexity.

Appendix A

s = strcat(s1,...,sN) horizontally concatenates s1,...,sN. Each input argument can be a character array, a cell array of character vectors, or a string array.

s = spectrogram(x) returns the short-time Fourier transform of the input signal, x. Each column of s contains an estimate of the short-term, time-localized frequency content of x.

y = resample(x,p,q) resamples the input sequence, x, at p/q times the original sample rate. If x is a matrix, then resample treats each column of x as an independent channel. resample applies an antialiasing FIR lowpass filter to x and compensates for the delay introduced by the filter.

[U,S,V] = svd(A,'econ') produces an economy-size decomposition of m-by-n matrix A:
m > n — Only the first n columns of U are computed, and S is n-by-n.
m = n — svd(A,'econ') is equivalent to svd(A).
m < n — Only the first m columns of V are computed, and S is m-by-m.

x = diag(A) returns a column vector of the main diagonal elements of A.

description from https://www.mathworks.com/help/matlab/

Appendix B

Test1

```
clear all; close all; clc;
music1 = [];
for i = 1:3
    [y, Fs] = audioread(strcat('a00',int2str(i),'.mp3'));
    y = y(:,1)+y(:,2);
    y = resample(y,20000,Fs);
    Fs = 20000;
    part = [];
    for j = 1:5:45
        five = y(Fs*j:Fs*(j+5),1);
        song_spec = abs(spectrogram(five));
        song_spec = reshape(song_spec,1,16385*8);
        part = [part song_spec'];
    end
    music1 = [music1 part];
end

music2 = [];
for i = 1:3
    [y, Fs] = audioread(strcat('e00',int2str(i),'.mp3'));
    y = y(:,1)+y(:,2);
    y = resample(y,20000,Fs);
    Fs = 20000;
    part = [];
    for j = 1:5:120
        five = y(Fs*j:Fs*(j+5),1);
        song_spec = abs(spectrogram(five));
        song_spec = reshape(song_spec,1,16385*8);
        part = [part song_spec'];
    end
    music2 = [music1 part];
end

music3 = [];
for i = 1:3
    [y, Fs] = audioread(strcat('l00',int2str(i),'.mp3'));
    y = y(:,1)+y(:,2);
    y = resample(y,20000,Fs);
    Fs = 20000;
    part = [];
    for j = 1:5:15
        five = y(Fs*j:Fs*(j+5),1);
        song_spec = abs(spectrogram(five));
        song_spec = reshape(song_spec,1,16385*8);
        part = [part song_spec'];
    end
    music3 = [music1 part];
end

musci1_wav = mu_wavelet(music1);
```

```
musci2_wav = mu_wavelet(music2);
musci3_wav = mu_wavelet(music3);

feature = 40;
[U,S,V,result,w,sort1,sort2,sort3] =
mu_trainer(musci1_wav,musci2_wav,musci3_wav,feature);
result;
[threshold1, threshold2] = mu_threshold(sort1,sort3,sort2); % adjust by hand
```

```
figure()
subplot(1,3,1)
histogram(sort1,27); hold on, plot([threshold1 threshold1],[0 10],'r')
set(gca,'Xlim',[0 3],'Ylim',[0 10],'Fontsize',14)
title('epic')
xlabel('threshold 1')
subplot(1,3,2)
histogram(sort2,51); hold on, plot([threshold2 threshold2],[0 10],'r')
set(gca,'Xlim',[0 3],'Ylim',[0 10],'Fontsize',14)
title('relax')
xlabel('threshold 2')
subplot(1,3,3)
histogram(sort3,30); hold on, plot([threshold2 threshold2],[0 10],'r')
set(gca,'Xlim',[0 3],'Ylim',[0 10],'Fontsize',14)
title('classical')
xlabel('threshold 2')
saveas(gcf,'t1_threshold.png');
```

```
test101 = [];
[y, Fs] = audioread('a004.mp3');
y = y(:,1)+y(:,2);
y = resample(y,20000,Fs);
Fs = 20000;
part = [];
for j = 1:5:45
    five = y(Fs*j:Fs*(j+5),1);
    song_spec = abs(spectrogram(five));
    song_spec = reshape(song_spec,1,16385*8);
    part = [part song_spec'];
end
test101 = [test101 part];

% TestNum=size(TestSet,2);

test101_wav = mu_wavelet(test101);
test101Mat = U'*test101_wav;  % PCA projection
pva101 = w'*test101Mat;  % LDA projection

TestNum=size(test101,2);

ResVec = (pva101>threshold1);

disp('Number of mistakes')
labels = [1 1 1 1 1 1 1 1];
errNum = sum(abs(ResVec-labels));
```

```
disp('Rate of success');
sucRate = 1-errNum/TestNum
```

Test2

```
clear all; close all; clc;
music1 = [];
[y, Fs] = audioread('m001.mp3');
y = y(:,1)+y(:,2);
y = resample(y,20000,Fs);
Fs = 20000;
part = [];
for j = 1:5:200
    five = y(Fs*j:Fs*(j+5),1);
    song_spec = abs(spectrogram(five));
    song_spec = reshape(song_spec,1,16385*8);
    part = [part song_spec'];
end
music1 = [music1 part];

music2 = [];
[y, Fs] = audioread('m002.mp3');
y = y(:,1)+y(:,2);
y = resample(y,20000,Fs);
Fs = 20000;
part = [];
for j = 1:5:200
    five = y(Fs*j:Fs*(j+5),1);
    song_spec = abs(spectrogram(five));
    song_spec = reshape(song_spec,1,16385*8);
    part = [part song_spec'];
end
music2 = [music2 part];

music3 = [];
[y, Fs] = audioread('m003.mp3');
y = y(:,1)+y(:,2);
y = resample(y,20000,Fs);
Fs = 20000;
part = [];
for j = 1:5:200
    five = y(Fs*j:Fs*(j+5),1);
    song_spec = abs(spectrogram(five));
    song_spec = reshape(song_spec,1,16385*8);
    part = [part song_spec'];
end
music3 = [music3 part];

musci1_wav = mu_wavelet(music1);
musci2_wav = mu_wavelet(music2);
musci3_wav = mu_wavelet(music3);

feature = 20;
[U,S,V,result,w,sort1,sort2,sort3] =
```

```
mu_trainer(musci1_wav,musci2_wav,musci3_wav,feature);
result;
[threshold1, threshold2] = mu_threshold(sort3,sort2,sort1); % adjust by hand
```

```
figure()
subplot(1,3,1)
histogram(sort3,40); hold on, plot([threshold1 threshold1],[0 10],'r')
set(gca,'Xlim',[-2 0],'Ylim',[0 10],'Fontsize',14)
title('Gopala')
xlabel('threshold 1')
subplot(1,3,2)
histogram(sort2,40); hold on, plot([threshold1 threshold1],[0 10],'r')
set(gca,'Xlim',[-2 1],'Ylim',[0 10],'Fontsize',14)
title('Kunda')
xlabel('threshold 1')
subplot(1,3,3)
histogram(sort1,40); hold on, plot([threshold2 threshold2],[0 10],'r')
set(gca,'Xlim',[3 7],'Ylim',[0 10],'Fontsize',14)
title('Nandana')
xlabel('threshold 2')
saveas(gcf,'t2_threshold.png');
```

```
test2 = [];
[y, Fs] = audioread('m003.mp3');
y = y(:,1)+y(:,2);
y = resample(y,20000,Fs);
Fs = 20000;
part = [];
for j = 200:5:240
   five = y(Fs*j:Fs*(j+5),1);
   song_spec = abs(spectrogram(five));
   song_spec = reshape(song_spec,1,16385*8);
   part = [part song_spec'];
end
test2 = [test2 part];

% TestNum=size(TestSet,2);

test2_wav = mu_wavelet(test2);
test2Mat = U'*test2_wav;  % PCA projection
pva2 = w'*test2Mat;  % LDA projection


TestNum=size(test2,2);

ResVec = (pva2>threshold1);

disp('Number of mistakes')
labels = [1 1 1 1 1 1 1 1];
errNum = sum(abs(ResVec-labels));

disp('Rate of success');
sucRate = 1-errNum/TestNum
```

Test3

```
clear all; close all; clc;
music1 = [];
[y, Fs] = audioread('m001.mp3');
y = y(:,1)+y(:,2);
y = resample(y,20000,Fs);
Fs = 20000;
part = [];
for j = 1:5:200
    five = y(Fs*j:Fs*(j+5),1);
    song_spec = abs(spectrogram(five));
    song_spec = reshape(song_spec,1,16385*8);
    part = [part song_spec'];
end
music1 = [music1 part];

[y, Fs] = audioread('m002.mp3');
y = y(:,1)+y(:,2);
y = resample(y,20000,Fs);
Fs = 20000;
part = [];
for j = 1:5:200
    five = y(Fs*j:Fs*(j+5),1);
    song_spec = abs(spectrogram(five));
    song_spec = reshape(song_spec,1,16385*8);
    part = [part song_spec'];
end
music1 = [music1 part];


music2 = [];
[y, Fs] = audioread('c001.wav');
y = y(:,1)+y(:,2);
y = resample(y,20000,Fs);
Fs = 20000;
part = [];
for j = 400:5:500
    five = y(Fs*j:Fs*(j+5),1);
    song_spec = abs(spectrogram(five));
    song_spec = reshape(song_spec,1,16385*8);
    part = [part song_spec'];
end
music2 = [music2 part];

[y, Fs] = audioread('c002.mp3');
y = y(:,1)+y(:,2);
y = resample(y,20000,Fs);
Fs = 20000;
part = [];
for j = 1:5:200
    five = y(Fs*j:Fs*(j+5),1);
    song_spec = abs(spectrogram(five));
    song_spec = reshape(song_spec,1,16385*8);
    part = [part song_spec'];
```

```
end
music2 = [music2 part];

music3 = [];
[y, Fs] = audioread('z001.mp3');
y = y(:,1)+y(:,2);
y = resample(y,20000,Fs);
Fs = 20000;
part = [];
for j = 1:5:200
    five = y(Fs*j:Fs*(j+5),1);
    song_spec = abs(spectrogram(five));
    song_spec = reshape(song_spec,1,16385*8);
    part = [part song_spec'];
end
music3 = [music3 part];

[y, Fs] = audioread('z002.mp3');
y = y(:,1)+y(:,2);
y = resample(y,20000,Fs);
Fs = 20000;
part = [];
for j = 1:5:50
    five = y(Fs*j:Fs*(j+5),1);
    song_spec = abs(spectrogram(five));
    song_spec = reshape(song_spec,1,16385*8);
    part = [part song_spec'];
end
music3 = [music3 part];

musci1_wav = mu_wavelet(music1);
musci2_wav = mu_wavelet(music2);
musci3_wav = mu_wavelet(music3);

feature = 20;
[U,S,V,result,w,sort1,sort2,sort3] =
mu_trainer(musci1_wav,musci2_wav,musci3_wav,feature);
result;
[threshold1, threshold2] = mu_threshold(sort2,sort3,sort1); % adjust by hand
```

```
figure()
subplot(1,3,1)
histogram(sort2,80); hold on, plot([threshold1 threshold1],[0 10],'r')
set(gca,'Xlim',[-9 -2],'Ylim',[0 10],'Fontsize',14)
title('meditation')
xlabel('threshold 1')
subplot(1,3,2)
histogram(sort1,61); hold on, plot([threshold2 threshold2],[0 10],'r')
set(gca,'Xlim',[-2 1],'Ylim',[0 10],'Fontsize',14)
title('pop')
xlabel('threshold 2')
subplot(1,3,3)
histogram(sort3,50); hold on, plot([threshold2 threshold2],[0 10],'r')
set(gca,'Xlim',[-2 0],'Ylim',[0 10],'Fontsize',14)
title('electric')
xlabel('threshold 2')
saveas(gcf,'t3_threshold.png');
```

```
test3 = [];
[y, Fs] = audioread('m003.mp3');
y = y(:,1)+y(:,2);
y = resample(y,20000,Fs);
Fs = 20000;
part = [];
for j = 200:5:240
    five = y(Fs*j:Fs*(j+5),1);
    song_spec = abs(spectrogram(five));
    song_spec = reshape(song_spec,1,16385*8);
    part = [part song_spec'];
end
test3 = [test3 part];

% TestNum=size(TestSet,2);

test2_wav = mu_wavelet(test3);
test2Mat = U'*test2_wav;  % PCA projection
pva2 = w'*test2Mat;  % LDA projection


TestNum=size(test3,2);

ResVec = (pva2>threshold1);

disp('Number of mistakes')
labels = [1 1 1 1 1 0 1 1 0];
errNum = sum(abs(ResVec-labels));

disp('Rate of success');
sucRate = 1-errNum/TestNum
```

Build a function that takes in the music piece

and returns a matrix with the cod_edge for each

```
function muData = mu_wavelet(mufile)

    [m,n] = size(mufile);
    nw = m/2; % wavelet resolution
    muData = zeros(nw,n);

    for k = 1:n
       x = mufile(:,k);
       [cA,cD] = dwt(x,'haar');
       cod_cA1 = rescale(abs(cA));
       cod_cD1 = rescale(abs(cD));
       cod_edge = cod_cA1+cod_cD1;
       muData(:,k) = reshape(cod_edge,nw,1);
    end

end


function [U,S,V,result,w,sort1,sort2,sort3] = mu_trainer(mu1,mu2,mu3,feature)
```

```
    n1 = size(mu1,2); n2 = size(mu2,2); n3 = size(mu3,2);

    [U,S,V] = svd([mu1 mu2 mu3],'econ');

    genre = S*V'; % projection onto principal components
    U = U(:,1:feature);
    genre1 = genre(1:feature,1:n1);
    genre2 = genre(1:feature,n1+1:n1+n2);
    genre3 = genre(1:feature,n1+n2+1:n1+n2+n3);

    m1 = mean(genre1,2);
    m2 = mean(genre2,2);
    m3 = mean(genre3,2);

    Sw = 0; % within class variances
    for k=1:n1
        Sw = Sw + (genre1(:,k)-m1)*(genre1(:,k)-m1)';
    end
    for k=1:n2
        Sw = Sw + (genre2(:,k)-m2)*(genre2(:,k)-m2)';
    end
    for k=1:n3
        Sw = Sw + (genre3(:,k)-m3)*(genre3(:,k)-m3)';
    end
    mA = mean([m1 m2 m3]);
    Sb = length(genre1)*(m1-mA)*(m1-mA)' + length(genre2)*(m2-mA)*(m2-mA)'+
length(genre3)*(m3-mA)*(m3-mA)'; % between class

    [V2,D] = eig(Sb,Sw); % linear discriminant analysis
    [~,ind] = max(abs(diag(D)));
    w = V2(:,ind); w = w/norm(w,2);

    v1 = w'*genre1;
    v2 = w'*genre2;
    v3 = w'*genre3;


    array = sort([mean(v1) mean(v2) mean(v3)]);
    result=[0 0 0];
    if array(1) == mean(v1)
        result(1) = 1;
        if array(2) == mean(v2)
            result(2) = 2;
            result(3) = 3;
        elseif array(2) == mean(v3)
            result(2) = 3;
            result(3) = 2;
        end
    elseif array(1) == mean(v2)
        array(1) = 2;
        if array(2) == mean(v1)
            result(2) = 1;
            result(3) = 3;
        elseif array(2) == mean(v3)
            result(2) = 3;
            result(3) = 1;
        end
    else
```

```
        array(1) = 3;
        if array(2) == mean(v2)
            result(2) = 2;
            result(3) = 1;
        elseif array(2) == mean(v1)
            result(2) = 1;
            result(3) = 2;
        end
    end

    sort1 = sort(v1);
    sort2 = sort(v2);
    sort3 = sort(v3);
end
function [threshold1, threshould2] = mu_threshold(temp1,temp2,temp3)
    t1 = length(temp1);
    t2 = 1;
    while temp1(t1)>temp2(t2)
        t1 = t1-1;
        t2 = t2+1;
    end
    threshold1 = (temp1(t1)+temp1(t2))/2;

    t3 = length(temp2);
    t4 = 1;
    while temp2(t3)>temp3(t4)
        t3 = t3-1;
        t4 = t4+1;
    end
    threshould2 = (temp2(t3)+temp3(t3))/2;
end
```