Music Genre Classification Using SVD and SVM

Connor Schleicher

March 6, 2020

Abstract

The singular value decomposition (SVD) is a powerful tool to separate out signals into components that can be used to classify unknown data of the same type. The first part of this analysis shows how different image sets are broken down by the SVD and how additional white space from around an uncropped image greatly distorts the SVD's projections. In the second part of this analysis the SVD showed how new music can be classified by artists in the same or different genres, or even determine which genre it is. The more variety between bands showed greater success than artists with similar music styles.

1 Introduction and Overview

Classification of unknown data is a common goal of many machine learning objectives. These classification problems are usually easily solved by a human observer but identifying the principal components to have a computer separate the data by is more challenging. In this analysis the principal components of two different groups of images are compared. Then a classification algorithm is tested for classifying music by artist, genre, and band.

The first part of this analysis looks at the SVD of two different sample groups from the YaleFaces dataset. The first group had the images cropped to just include the faces, while the second group was uncropped and contained more background in the image. The SVD was performed on each group and a comparison of the images can be made to see the differences between cropping the files before analysis.

The second part of the analyis was implementing a classification algorithm to classify music based on different conditions. The first part was to classify a music sample to see if it was from one of three bands from three different genres. The second task was to classify a music sample from three different artists within the same genre. The final task was to classify a music sample as being from one of three genres, when there there multiple different bands within each genre. For each of these tasks, the process was the same by performing the SVD and running the core features through a classification algorithm to fit some parameters to classify new data.

2 Theoretical Background

2.1 Singular Value Composition (SVD)

The singular value decomposition (SVD) is a way to represent any matrix as the product of three different matrix transformations. Any matrix performs a stretch or a rotation to the value it is applied to, as defined in Equation 1 [1]. In this definition, U and V are both unitary matrices, and Σ is a diagonal matrix. Sigma is also, equal to the square root of the eigenvalues of matrix `A`.

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

Due to the construction of the SVD the sigma matrix is made in a way so that each value is in decreasing order along the diagonal, therefore the largest and most significant sigma value is located in position one. Along with the sigma matrix, the U and V are constructed to obtain certain properties. U is constructed as an orthonormal set of basis vectors. All these properties come together to accurately describe the main components of a matrix without constructing the entire matrix, just the first couple of primary basis vectors will define the major components of the system.

2.2 Support Vector Machine (SVM)

The SVM algorithm attempts to separate different classes of data by creating a hyperplane or set of hyperplanes with the greatest margin between classes [2]. The SVM algorithm will look at separating data beyond the original plane where the data resides. This can lead to a better separation than a standard linear separation in the original space.

3 Algorithm Implementation and Development

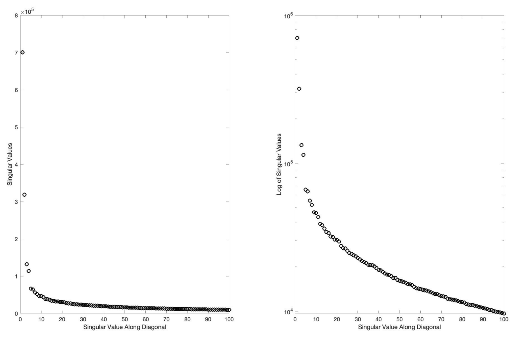
The two parts of this experiment were done in a similar manner. The Yale Faces dataset was analyzed by performing the SVD on the entire dataset and comparing the components of each. Algorithm 1 shows the steps used for each set of images. The music classification was a little more complex. After the SVD was performed the data was fed into a classification algorithm which creates the parameters to classify the data by. The total process for the music classification structure can be seen in Algorithm 2.

Algorithm 1: Yale Faces SVD

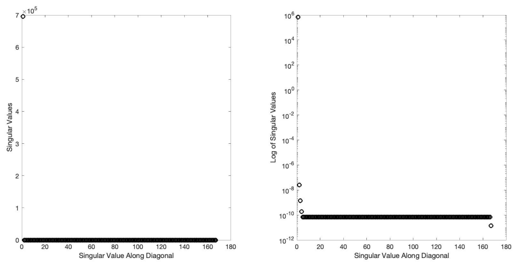
```
for $i = 1 : NumberofImages$ do
  Reshape image into vector of values
  Add to column in matrix
end for
Perform SVD
```

Algorithm 2: Music Classification

```
for $i = 1 : NumberofSongs$ do
  for $k = 1 : Numberofsnippetsfromeachsong$ do
    Import samples from a song
  end for
end for
Split samples from each set into test/train sets
for $j = 1 : NumberofTrainSamples$ do
  Compute spectogram of sample
  Convert and store spectogram in new matrix
end for
Compute the SVD of the train data set
Use the V matrix (feature space) to train and classify an algorithm
for $j = 1 : NumberofTestSamples$ do
  Compute spectogram of sample
  Convert and store spectogram in new matrix
end for
Project the spectrogram of the test set onto the principal components of the training space
Use the classification parameters to classify the new test data
Cross validate with different test data to verify results
```

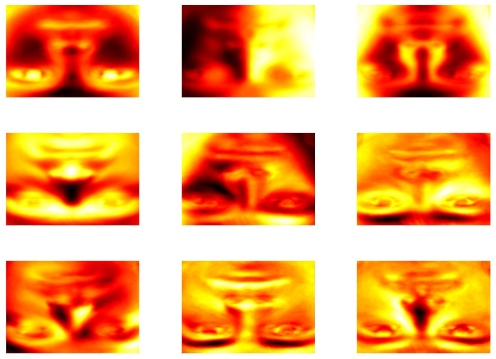


(a) Cropped Yale Faces Singular Values

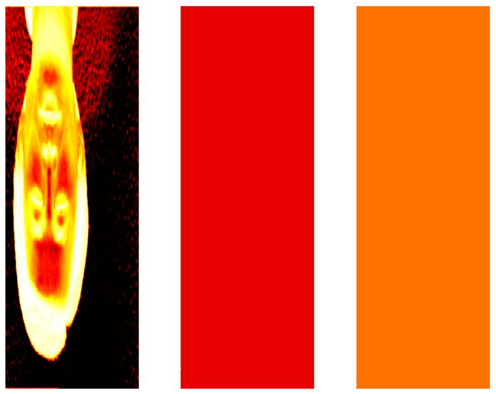


(b) Uncropped Yale Faces Singular Values

Figure 1: Comparison of the singular values for the cropped and uncropped Yale Faces dataset.



(a) Cropped Yale Faces Low Rank Approximations



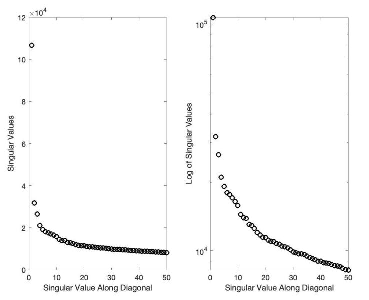
(b) Uncropped Yale Faces Low Rank Approximations

Figure 2: Comparison of low rank approximations for the Yale Faces dataset.

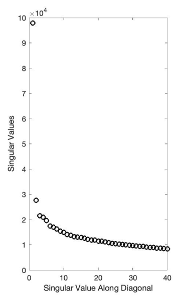
4 Computational Results

4.1 Part I Yale Faces

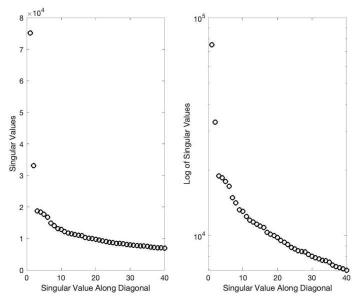
The first part of this analysis was comparing the cropped and uncropped Yale Faces datasets. Each image was converted to a column of values and stacked into one large matrix containing all the pictures from each set. The SVD of both sets was then done on each. The first thing I looked at was the singular values of both sets, the Σ matrix. Figure 1 shows the comparison between of the cropped and uncropped singular values. Figure 1a has the plot of the singular values along with the log of the singular values. From this you can see how there are few singular values that are significant. Compare this to the uncropped faces in Figure 1b, where there is only one singular value that stand out. Comparing the rank of each set, the cropped set has a low rank approximation of around 9, where the uncropped faces have a low rank approximation of 1, based off the singular values. The other two matrices in the SVD, the U and V, perform another rule. The U matrices showed the principal components of each data set that it can separate them by. The V matrix is instead the features themselves that can be separated into the various classes. In Figure 2 low rank approximations are shown. A low rank approximation is done by recombining the SVD matrices but with only first few singular values. In the cropped set, Figure 2a, the first 9 rank approximations portray a clear picture of the faces. On the contrary, the uncropped faces, Figure 2b only the first rank has any resemblance to a face. The next two approximations are presumable the extra space around the images.



(a) Singular Values of Band Classification



(b) Singular Values of Artist Classification



(c) Singular Values of Genre Classification

Figure 3: Comparison of the singular values for each music classification task.

4.2 Part II Music Classification

4.2.1 Band Classification

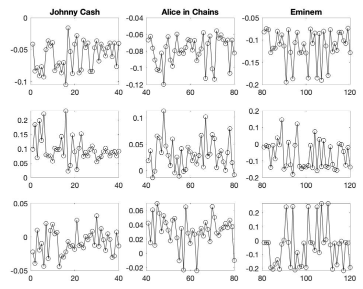
The first of the three music classification schemes was classifying a band from three separate genres. Using the information from the SVD we can start to see how certain features will be used to identify a band. Figure 3a shows how the first three singular values stand out the most with a large emphasis on the first one. The singular values came from the SVD of the spectrograms of the songs. Since there were three separate bands with each song having similar sounds within those bands, each of these correlates to one of the three main singular values. Then in Figure 4a the first three features of each band from the training set is plotted. This allows us to see how the features can be used to identify the band. For each band they tend to follow a similar feature space for the first feature since they're all negative values. Then starting in the second and third features, we can see some differences between the three bands. This training data was fed into MATLAB's SVM function to classify unknown song clips. With only using the first 10 features, the SVM classification was able to achieve an average of 93.7% accuracy after 20 different randomization's.

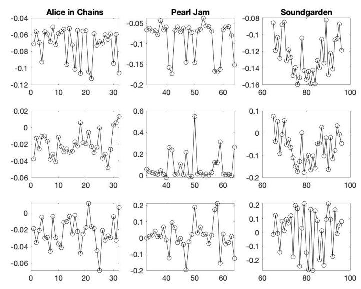
4.2.2 Artist Classification

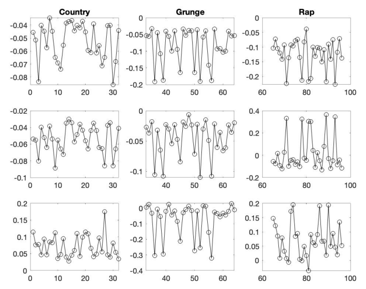
The artist classification was done on three different bands but within the same genre. The three bands were all within the same genre and all originated from Seattle. Starting with the singular values of the spectrogram in Figure 3b there are only two singular values that stand out. This would give a clue that the separation between these artists is more difficult. In the features shown in Figure 4b there are more similarities across the features than there were in above section. When MATLAB's SVM algorithm was applied to this training data the performance was significantly worse than the Band classification, at 71.2%. This was confirms the suspicions from the singular values, and features. This level of accuracy was only achieved with doubling the amount of features to 20.

4.2.3 Genre Classification

The final part of this analysis was to classify a song clip to one of three genres. There were five songs chosen from five different artists for each genre. The singular values, Figure 3c, have two very distinct singular values and a secondary set of four singular values. Further observation of Figure 4c shows some separation between features. There's a lot of switching of signs occurring at all but the first feature. The output of the SVM algorithm corresponded to an average accuracy rating of 91.7%. It also only took 10 features to achieve this level of accuracy.







(a) Features of Band Classification (b) Features of Artist Classification (c) Features of Genre Classification

Figure 4: Comparison of the singular values for each music classification task.

5 Summary and Conclusions

With the insight from the SVD, this analysis was able to illustrate how to classify objects and how that classification occurs. From the first part, there was a significant difference in the rank between the cropped and uncropped data sets. The addition blank space in the uncropped images greatly conributed to the principal comonent being only the faces. This could lead to more difficulty with classifying faces by smiling, sad, angry etc. For the second part, the singular values and the features were able to help predict how well the separation between the classes was. This was then verified by the accuracy of the SVM classification. With a small number of features we were able to achieve over 90% accuracy for two of the three music classification tasks. All of these music samples all underwent the same Fourier transform spectorgram. Further accuracy could have potentially been achieved by varying the type of transformation. A wavelet transformation could have provided better results and could be left for later research.

References

[1] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

[2] *Support-vector machine*. URL: `https://en.wikipedia.org/wiki/Support-vector_machine`.

`https://github.com/ConnorSch/GenreClassification`

Appendix A MATLAB Functions

- `svd(X)` computes the full Singular Value Decomposition of matrix X. Defined as `X = USV*`
- `fitecoc(V,ctrain)` computes a fully trained multiclass SVM model using the one-vs-one design. It uses the training data in Matrix V, with classes in vector ctrain to create the model.
- `predict(svm.mod, xtest)` returns a vector of class labels for the new data xtest. It uses the class labels and parameters in svm.mod to predict the classes of the new data
- `spectrogram(X)` returns the short-time Fourier transform of the input signal X
- `randperm(n)` random permutation of integers from 1 to n

Appendix B MATLAB Code

```
%% SVD Exploration Images
% initialization
clear all, close all, clc

% Cropped Yale Faces
myMainDir = 'CroppedYale';
myFolds = dir(fullfile(myMainDir));

iter = 1;
for j = 4:length(myFolds)
     baseFoldname = myFolds(j).name;
     fullFoldname = fullfile(myMainDir,baseFoldname);
     myDir = fullfile(myMainDir,baseFoldname);
     myFiles = dir(fullfile(fullFoldname));
     for i = 1:length(myFiles)
          basefilename = myFiles(i).name;
          fullFileName = fullfile(myDir,basefilename);
          if isfile(fullFileName)
              im = imread(fullFileName);
              CF(:,iter) = im(:);
              iter = iter + 1;
          end
     end
end

[U,S,V] = svd(double(CF));

%% Playing with the SVD
figure()
sig = diag(S);
[M,N] = size(CF);

subplot(1,2,1), plot(sig(1:100),'ko','Linewidth',[1.5])
ylabel('Singular Values')
xlabel('Singular Value Along Diagonal')

subplot(1,2,2), semilogy(sig(1:100),'ko','Linewidth',[1.5])
ylabel('Log of Singular Values')
xlabel('Singular Value Along Diagonal')
%% Recreate an image
[r,c] = size(im);
rank = 10;
LRA = U(:,1:rank)*S(1:rank,1:rank) * V(:,1:rank)'; % Low Rank Approximation of images
image = reshape(LRA(:,1),r,c);

for j = 1:9
    subplot(3,3,j)
    ef = reshape(U(:,j),r,c);
    pcolor(ef),axis off, shading interp, colormap(hot)
end

%% Uncropped Images
MainDir = 'UnCroppedyalefaces';
Files = dir(fullfile(MainDir));
```

```
iter2 = 1;
for k = 1:length(Files)
     basefile = Files(i).name;
     fullFile = fullfile(MainDir,basefile);
     if isfile(fullFile)
         im2 = imread(fullFile);
         UCF(:,iter2) = im2(:);
         iter2 = iter2 + 1;
     end
end

[U2,S2,V2] = svd(double(UCF));

%% Recreate an image
[r2,c2] = size(im2);
rank = 1;
LRA2 = U2(:,1:rank)*S2(1:rank,1:rank) * V2(:,1:rank)'; % Low Rank Approximation of images
image2 = reshape(LRA2(:,1),r2,c2);
%imshow(uint8(image2))

figure()
sig2 = diag(S2);
[M,N] = size(UCF);

title('Singular Values and PCA Modes for Ideal Case')
subplot(1,2,1), plot(sig2(1:length(sig2)),'ko','Linewidth',[1.5])
ylabel('Singular Values')
xlabel('Singular Value Along Diagonal')

subplot(1,2,2), semilogy(sig2(1:length(sig2)),'ko','Linewidth',[1.5])
ylabel('Log of Singular Values')
xlabel('Singular Value Along Diagonal')

figure()
for j = 1:3
   subplot(1,3,j)
   ef = reshape(U2(:,j),r2,c2);
   pcolor(ef),axis off, shading interp, colormap(hot)
end

%% Music Artist Classification - Task 1
% initialization
clear all, close all, clc

mainDir = 'MusicClassification/BandClassification';


% Try looping through collecting numerous samples of music
%Country
fullFoldname = fullfile(mainDir,'Country');
myFiles = dir(fullfile(fullFoldname));

num_seg = 10;
iter = 1;
```

```
for i = 4:length(myFiles)
    basefilename = myFiles(i).name;
    fullFileName = fullfile(fullFoldname,basefilename);
    if isfile(fullFileName)
        info = audioinfo(fullFileName);
        mid = round(info.TotalSamples/2);
        step = 5*info.SampleRate;
        points = linspace(-num_seg/2,num_seg/2,num_seg+1);
        for j = 1:num_seg
            point = mid+points(j)*step;
            sample = double([point,point + step]);
            audio = audioread(fullFileName,sample);
            CM(:,iter) = audio;
            iter = iter + 1;
        end
    end
end

% Grunge
fullFoldname = fullfile(mainDir,'Grunge');
myFiles = dir(fullfile(fullFoldname));

iter = 1;
for i = 4:length(myFiles)
    basefilename = myFiles(i).name;
    fullFileName = fullfile(fullFoldname,basefilename);
    if isfile(fullFileName)
        info = audioinfo(fullFileName);
        mid = round(info.TotalSamples/2);
        step = 5*info.SampleRate;
        points = linspace(-num_seg/2,num_seg/2,num_seg+1);
        for j = 1:num_seg
            point = mid+points(j)*step;
            sample = double([point,point + step]);
            audio = audioread(fullFileName,sample);
            GM(:,iter) = audio;
            iter = iter + 1;
        end
    end
end

% RAP
fullFoldname = fullfile(mainDir,'RAP');
myFiles = dir(fullfile(fullFoldname));

iter = 1;
for i = 4:length(myFiles)
    basefilename = myFiles(i).name;
    fullFileName = fullfile(fullFoldname,basefilename);
    if isfile(fullFileName)
        info = audioinfo(fullFileName);
        mid = round(info.TotalSamples/2);
        step = 5*info.SampleRate;
        points = linspace(-num_seg/2,num_seg/2,num_seg+1);
        for j = 1:num_seg
            point = mid+points(j)*step;
            sample = double([point,point + step]);
            audio = audioread(fullFileName,sample);
            RM(:,iter) = audio;
            iter = iter + 1;
        end
    end
end

g_size = iter -1;

%% Pull Out Train and Test sets
test_runs = 20;
percentage = zeros(1,test_runs);
for l = 1:test_runs


q1 = randperm(g_size);
q2 = randperm(g_size);
q3 = randperm(g_size);

samplerate = 0.8;
tests = samplerate*g_size;
CM_train = CM(:,q1(1:tests));
CM_test = CM(:,q1(tests+1:end));

GM_train = GM(:,q2(1:tests));
GM_test = GM(:,q2(tests+1:end));

RM_train = RM(:,q3(1:tests));
RM_test = RM(:,q3(tests+1:end));

%% Create spectograms with Matlab
CM_sp = [];
GM_sp = [];
RM_sp = [];
for i = 1:g_size*samplerate
   cs = spectrogram(CM_train(:,i));
   gs = spectrogram(GM_train(:,i));
   rs = spectrogram(RM_train(:,i));

   CM_s(:,i) = abs(cs(:));
   GM_s(:,i) = abs(gs(:));
   RM_s(:,i) = abs(rs(:));
end

for j = 1:size(CM_test,2)
   cst = spectrogram(CM_test(:,j));
   gst = spectrogram(GM_test(:,j));
   rst = spectrogram(RM_test(:,j));

   CM_st(:,j) = abs(cst(:));
   GM_st(:,j) = abs(gst(:));
   RM_st(:,j) = abs(rst(:));
end
X = [CM_s, GM_s, RM_s];
Xtest = [CM_st,GM_st,RM_st];

%% Singular Values and Principal Components

figure()
sig = diag(S);
[M,N] = size(X);

subplot(1,2,1), plot(sig(1:g_size),'ko','Linewidth',[1.5])
ylabel('Singular Values')
xlabel('Singular Value Along Diagonal')

subplot(1,2,2), semilogy(sig(1:g_size),'ko','Linewidth',[1.5])
ylabel('Log of Singular Values')
xlabel('Singular Value Along Diagonal')

figure()
for j = 1:6
   subplot(2,3,j)
   plot(U(:,j))
end

figure(3)
for j=1:3
  V1 = [1,4,7];
  subplot(3,3,V1(j))
  plot(1:g_size*samplerate,V(1:g_size*samplerate,j),'ko-')
  V2 = [2,5,8];
  subplot(3,3,V2(j))
  plot(g_size*samplerate+1:2*g_size*samplerate,V(g_size*samplerate+1:2*g_size*samplerate,j
  V3 = [3,6,9];
  subplot(3,3,V3(j))
  plot(2*g_size*samplerate+1:3*g_size*samplerate,V(2*g_size*samplerate+1:3*g_size*samplera
end
subplot(3,3,1), title('Johnny Cash')
subplot(3,3,2), title('Alice in Chains')
subplot(3,3,3), title('Eminem')

%% Create training and test sets
    [U,S,V] = svd(X,'econ');
    numFeat = 10;

    samplesize = samplerate * g_size;
    xtrain = V(:,1:numFeat);
    xtest = U'*Xtest;

    ctrain = [repmat({'Johnny Cash'},[samplesize,1]);repmat({'Alice In Chains'},[samplesize
    %ctrain = [ones(samplesize,1);2*ones(samplesize,1);3*ones(samplesize,1)];
    truth = [repmat({'Johnny Cash'},[g_size-samplesize,1]);repmat({'Alice In Chains'},[g_si

    %xtrain = [CM_train,GM_train,RM_train];
    %xtest = [CM_test, GM_test, RM_test];
    %pre = classify(xtest, xtrain, ctrain, 'linear');

    svm_mod = fitcecoc(V, ctrain);
    pre = predict(svm_mod, xtest');

    num_correct = 0;
    for k = 1:length(truth)
       if strcmp(pre{k}, truth{k})
            num_correct = num_correct + 1;
       end
    end
    percentage(l) = (num_correct/length(truth))*100;
end
mean(percentage)

%% Music Artist Classification - Task 2
% initialization
clear all, close all, clc

mainDir = 'MusicClassification/ArtistClassification';

myFiles = dir(fullfile(mainDir));
num_seg = 10;
iter = 1;
for i = 4:length(myFiles)
    fullFileName = fullfile(mainDir, myFiles(i).name);
    if isfile(fullFileName)
        info = audioinfo(fullFileName);
        mid = round(info.TotalSamples/2);
        step = 5*info.SampleRate;
        points = linspace(-num_seg/2, num_seg/2, num_seg+1);
        for j = 1:num_seg
            point = mid+points(j)*step;
            sample = double([point, point + step]);
            audio = audioread(fullFileName, sample);
            if size(audio,2) > 1
                audio = sum(audio,2)/size(audio,2);
            end
            AC(:, iter) = audio;
            iter = iter + 1;
        end
    end
end

a_size = (iter -1)/3;

%% Pull Out Train and Test sets
testruns = 20;
percentage = zeros(1, testruns);
for l = 1:testruns

q1 = randperm(a_size);
q2 = randperm(a_size);
```

```
q3 = randperm(a_size);

samplerate = 0.8;
tests = samplerate*a_size;
AIC = AC(:,1:a_size);
AIC_train = AIC(:,q1(1:tests));
AIC_test = AIC(:,q1(tests+1:end));

PJ = AC(:,a_size+1:2*a_size);
PJ_train = PJ(:,q2(1:tests));
PJ_test = PJ(:,q2(tests+1:end));

SG = AC(:,2*a_size+1:end);
SG_train = SG(:,q3(1:tests));
SG_test = SG(:,q3(tests+1:end));

%% Create spectograms with Matlab
AIC_sp = [];
PJ_sp = [];
SG_sp = [];
for i = 1:a_size*samplerate
    aics = spectrogram(AIC_train(:,i));
    pjs = spectrogram(PJ_train(:,i));
    sgs = spectrogram(SG_train(:,i));

    AIC_s(:,i) = abs(aics(:));
    PJ_s(:,i) = abs(pjs(:));
    SG_s(:,i) = abs(sgs(:));
end

for j = 1:size(AIC_test,2)
    aicst = spectrogram(AIC_test(:,j));
    pjst = spectrogram(PJ_test(:,j));
    sgst = spectrogram(SG_test(:,j));

    AIC_st(:,j) = abs(aicst(:));
    PJ_st(:,j) = abs(pjst(:));
    SG_st(:,j) = abs(sgst(:));
end

X = [AIC_s, PJ_s, SG_s];
Xtest = [AIC_st,PJ_st,SG_st];

%% Singular Values and Principal Components

% [U,S,V] = svd(X,'econ');
%
% figure()
% sig = diag(S);
% [M,N] = size(X);
%
% subplot(1,2,1), plot(sig(1:a_size),'ko','Linewidth',[1.5])
% ylabel('Singular Values')
% xlabel('Singular Value Along Diagonal')
```

```
%
% subplot(1,2,2), semilogy(sig(1:a_size),'ko','Linewidth',[1.5])
% ylabel('Log of Singular Values')
% xlabel('Singular Value Along Diagonal')
%
% figure()
% for j = 1:6
%     subplot(2,3,j)
%     plot(U(:,j))
% end
%
% figure(3)
% for j=1:3
%    V1 = [1,4,7];
%    subplot(3,3,V1(j))
%    plot(1:a_size*samplerate,V(1:a_size*samplerate,j),'ko-')
%    V2 = [2,5,8];
%    subplot(3,3,V2(j))
%    plot(a_size*samplerate+1:2*a_size*samplerate,V(a_size*samplerate+1:2*a_size*samplerate
%    V3 = [3,6,9];
%    subplot(3,3,V3(j))
%    plot(2*a_size*samplerate+1:3*a_size*samplerate,V(2*a_size*samplerate+1:3*a_size*sample
% end
% subplot(3,3,1), title('Alice in Chains')
% subplot(3,3,2), title('Pearl Jam')
% subplot(3,3,3), title('Soundgarden')

%% Create training and test sets
[U,S,V] = svd(X,'econ');

numFeat = 20;

samplesize = samplerate * a_size;
xtrain = V(:,1:numFeat);
xtest = U'*Xtest;

ctrain = [repmat({'Alice_In_Chains'},[samplesize,1]);repmat({'Pearl_Jam'},[samplesize,1]);r
truth = [repmat({'Alice_In_Chains'},[a_size-samplesize,1]);repmat({'Pearl_Jam'},[a_size-sam

svm.mod = fitcecoc(V,ctrain);
pre = predict(svm.mod,xtest');

num_correct = 0;
for k = 1:length(truth)
    if strcmp(pre{k},truth{k})
          num_correct = num_correct + 1;
    end
end
percentage(l) = (num_correct/length(truth))*100;

end
mean(percentage)

%% Music Genre Classification
```

```
% initialization
clear all, close all, clc

mainDir = 'MusicClassification/ArtistClassification';

myFiles = dir(fullfile(mainDir));
num_seg = 10;
iter = 1;
for i = 4:length(myFiles)
    fullFileName = fullfile(mainDir,myFiles(i).name);
    if isfile(fullFileName)
        info = audioinfo(fullFileName);
        mid = round(info.TotalSamples/2);
        step = 5*info.SampleRate;
        points = linspace(-num_seg/2,num_seg/2,num_seg+1);
        for j = 1:num_seg
            point = mid+points(j)*step;
            sample = double([point,point + step]);
            audio = audioread(fullFileName,sample);
            if size(audio,2) > 1
                audio = sum(audio,2)/size(audio,2);
            end
            AC(:,iter) = audio;
            iter = iter + 1;
        end
    end
end

a_size = (iter-1)/3;

%% Pull Out Train and Test sets
testruns = 20;
percentage = zeros(1,testruns);
for l = 1:testruns

q1 = randperm(a_size);
q2 = randperm(a_size);
q3 = randperm(a_size);

samplerate = 0.8;
tests = samplerate*a_size;
AIC = AC(:,1:a_size);
AIC_train = AIC(:,q1(1:tests));
AIC_test = AIC(:,q1(tests+1:end));

PJ = AC(:,a_size+1:2*a_size);
PJ_train = PJ(:,q2(1:tests));
PJ_test = PJ(:,q2(tests+1:end));

SG = AC(:,2*a_size+1:end);
SG_train = SG(:,q3(1:tests));
SG_test = SG(:,q3(tests+1:end));

%% Create spectograms with Matlab
```

```
AIC_sp = [];
PJ_sp = [];
SG_sp = [];
for i = 1:a_size*samplerate
    aics = spectrogram(AIC_train(:,i));
    pjs = spectrogram(PJ_train(:,i));
    sgs = spectrogram(SG_train(:,i));

    AIC_s(:,i) = abs(aics(:));
    PJ_s(:,i) = abs(pjs(:));
    SG_s(:,i) = abs(sgs(:));
end

for j = 1:size(AIC_test,2)
    aicst = spectrogram(AIC_test(:,j));
    pjst = spectrogram(PJ_test(:,j));
    sgst = spectrogram(SG_test(:,j));

    AIC_st(:,j) = abs(aicst(:));
    PJ_st(:,j) = abs(pjst(:));
    SG_st(:,j) = abs(sgst(:));
end

X = [AIC_s, PJ_s, SG_s];
Xtest = [AIC_st,PJ_st,SG_st];

%% Singular Values and Principal Components

% [U,S,V] = svd(X,'econ');
%
% figure()
% sig = diag(S);
% [M,N] = size(X);
%
% subplot(1,2,1), plot(sig(1:a_size),'ko','Linewidth',[1.5])
% ylabel('Singular Values')
% xlabel('Singular Value Along Diagonal')
%
% subplot(1,2,2), semilogy(sig(1:a_size),'ko','Linewidth',[1.5])
% ylabel('Log of Singular Values')
% xlabel('Singular Value Along Diagonal')
%
% figure()
% for j = 1:6
%     subplot(2,3,j)
%     plot(U(:,j))
% end
%
% figure(3)
% for j=1:3
%   V1 = [1,4,7];
%   subplot(3,3,V1(j))
%   plot(1:a_size*samplerate,V(1:a_size*samplerate,j),'ko-')
%   V2 = [2,5,8];
%    subplot(3,3,V2(j))
%    plot(a_size*samplerate+1:2*a_size*samplerate,V(a_size*samplerate+1:2*a_size*samplerate
%    V3 = [3,6,9];
%    subplot(3,3,V3(j))
%    plot(2*a_size*samplerate+1:3*a_size*samplerate,V(2*a_size*samplerate+1:3*a_size*sample
% end
% subplot(3,3,1), title('Alice in Chains')
% subplot(3,3,2), title('Pearl Jam')
% subplot(3,3,3), title('Soundgarden')

%% Create training and test sets
[U,S,V] = svd(X,'econ');

numFeat = 20;

samplesize = samplerate * a_size;
xtrain = V(:,1:numFeat);
xtest = U'*Xtest;

ctrain = [repmat({'Alice In Chains'},[samplesize,1]);repmat({'Pearl Jam'},[samplesize,1]);r
truth = [repmat({'Alice In Chains'},[a_size-samplesize,1]);repmat({'Pearl Jam'},[a_size-sa

svm.mod = fitcecoc(V,ctrain);
pre = predict(svm.mod,xtest');

num_correct = 0;
for k = 1:length(truth)
   if strcmp(pre{k},truth{k})
        num_correct = num_correct + 1;
   end
end
percentage(l) = (num_correct/length(truth))*100;

end
mean(percentage)
```