AMATH 582 Homework 4

Leon Yan

March 14, 2020

Abstract

This report serves to illustrate how we use the Singular Value Decomposition(SVD) techniques to extract the principle modes in human faces, and compare the effects caused by alignment of the data. Next, the SVD is promoted to analyze the modes in audio signal to classify the music among different genres, bands and artists. For the full materials including code and homework description, please go to url: `https://github.com/unitraveleryy/Amath582HW`

1 Introduction and Overview

In this homework, in the former part, we are going to extract the common features of different people's faces' data. And further, reconstruct the face image with just picking out the dominant elements in our SVD analysis. In the latter part, with the SVD technique, we analyze the spectrogram of music clips from different bands and in different genres, to extract the patterns that determine the category of the music. And further, we use the pattern to determine to which category the new unknown music belongs to.

1.1 Data Description

In Task I, eigen-faces task, we are provided with face data from the Yale University. In the uncropped face data, we have 11 different setting for 15 different people, in total we have 165 different images of face. In the cropped 192-by-168 face data, we have 64 different lighting conditions for 38 different people, in total we have 2432 different images of face. In Task II, classifying the music genres and bands, we downloaded several music file in different bands, artist and bands from the Internet, with the sampling rate 44100 hz. zuyingsong, jaychou and swordromance are the music data for doing the classification in case1 (different bands); soundgarden, aliceinchains and pearljam are the music data for doning the classification in case2 (different bands but same genre); classical, folk and rap are the music data data for doing the classification in case3 (different genre).

2 Theoretical Background

2.1 Singular Value Decomposition

The detailed explanation can be seen in Homework 3.

2.2 Distilling eigen-faces via SVD, PCA

By reshaping the face images into a long column vector, and construct a data matrix X to store the image information column wisely. Then subtract the mean row by row, and doing the SVD of X, $X = U\Sigma V^*$. We can obtain the dominant modes/pattern appearing in the face data we have. We can rearrange the columns in U into a 192-by-168 matrix and display them to see the dominant modes in human faces.

2.3 Faces reconstruction via reduced SVD, Matrix reconstruction

Given the fact that $X_{m\times n} = U\Sigma V^*$, we can reconstruct and approximate the face images with just important information denoted by the magnitude of the singular values, i.e., we can pick out the first dominant r modes, $r \ll n, m$, to give a rank-r approximation of the data matrix $X = U_{m\times r}\Sigma_{r\times r}V^*_{r\times n}$.

2.4 Music data feature extraction

Stack the audio wave data into a column vector, then randomly pick out a 5-second clip, apply the *spectrogram()* to derive the corresponding spectrogram S. Then do the SVD of S, $S = U\Sigma V^*$; Choose the first i columns (i can vary from 1 to the end column of the U), which depends on how many modes you think can represent the pattern/feature of that specific kind song; stack them into a column feature vector.

2.5 Linear Discriminant Analysis (LDA)

LDA is one of the standard methods for classification, of which the goal is to find a linear combination of features that characterizes or separates two or more classes of objects or events in the data. LDA finds a suitable projection that maximizes the distance between the interclass data while minimizing the interclass data. The problem is to construct a projection $\mathbf{w}$ such that

$$\mathbf{w} = \arg\max_{\mathbf{w}} \frac{\mathbf{w}\mathbf{S}_B\mathbf{w}}{\mathbf{w}\mathbf{S}_W\mathbf{w}} \tag{1}$$

where the scatter matrices for between-class S_B and within-class S_W data are

$$\mathbf{S}_B = n\sum_{i=1}^{n}(\mu_i - \mu)(\mu_i - \mu)^T \tag{2}$$

$$\mathbf{S}_W = \sum_{j=1}^{n}\sum_{x}(x - \mu_j)(x - \mu_j)^T \tag{3}$$

where μ_i is the mean of class i and μ is the mean of the class means.

2.6 Naive Bayes Classifier

Naive Bayers classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features. A naive Bayes classifier considers each of these features to contribute independently to the probability regradless of any possible correlations between the features. Under this assumption, the naive Bayes classifier picks the hypothesis that is most probable; this is known as the maximum a posterior or MAP decision rule. The prediction of a naive Bayes classifier gives is

$$\hat{y} = \arg\max_{k\in\{1,....K\}} p(C_k)\prod_{i=1}^{n} p(x_i|C_k) \tag{4}$$

2.7 Decision Tree

The decision tree is a hierarchical construct that looks for optimal ways to split the data in order to provide a robust classification and regression. The algorithm of a basic decision tree is (i) scan through each component (feature) $x_k (k = 1, 2, \ldots, n)$ of the vector x_j to identify the value of x_j that gives the best labeling prediction for y_j. (ii) Compare the prediction accuracy for each split on the feature x_j. The feature giving the best segmentation of the data is selected as the split for the tree. (iii) With the two new branches of the tree created, this process is repeated on each branch. The algorithm terminates once the each individual data point is a unique cluster, known as a *leaf*, on a new branch of the tree. Because of its high variation for different data sets, decision trees are used as ensemble known as random forests.

3 Algorithm Implementation and Development

- **Algorithm implementation for PCA on the Yale cropped & uncropped face data.**

1. Gather all the images, reshaping them into a column vector, lining them up, storing in matrix X
2. Do the SVD for matrix $X_{m\times n} = U\Sigma V^*$, m is the number of pixels per image, n is the number of image
3. Plot the singular values and determine how many modes we need to approximate the original data properly
4. Pick out the relevant columns in U, pull them back into 2-d image matrix, and visualize them
5. Use the picked basis in U, reconstruct the image, and evaluate the goodness by computing the $\frac{||X_r - X||}{X}$
6. Compare the SVD results and image reconstruction properness between the cropped and uncropped image data

- **Algorithm implementation for music classification.**

1. Load all the music file, stack music belonging to the same category into a huge column vector
2. Randomly partition the audio data into 80% as the RAW training set, 20% as the RAW test set
3. Randomly pick 5 seconds clip from the RAW training set, do the SVD of its spectrogram, store the first 2 modes as the feature input.
4. By specify the repeat times for the above step as *ntrain*, we can apply it into the customized function *bootstrap()* (you can check the definition in the MATLAB appendix), and ge the formatted training data for model training
5. Construct the relevant labels for the formatted training data generated in the above step, and choose the method from LDA, Naive Bayes and Decision tree classifier to train the model
6. Generate the formatted testing set from the RAW test set through *bootstrap()*, evaluate the model performance on formatted test set by *predict()* provided by MATLAB.

4 Computational Results

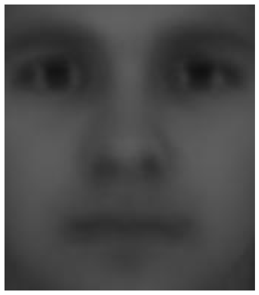
- In task to extract the eigen-faces in cropped & uncropped Yale face data.

For cropped data, in Fig. 1a is the mean image of all the different faces, and the distribution of the singular values can be seen in Fig. 2a, around the first 250 modes account for the nearly 99.9% of the total energy. Our data matrix X's dimension is 32256-by-2432. So roughly, there will be at most 2432 non-zero modes.

We use the `imagesc()` to visualize the eigen-faces due to the basis in U are no longer the uint8 type for image. In Fig. 3a are the first 9 principle modes(eigen-faces). In image reconstruction part, we varied the number of modes we use to re-generate the image and evaluating the goodness of the reconstruction by computing the error $\frac{||X_r - X||}{||X||}$; the less the error, the better the reconstructed images are. You can see in Fig. 4a, with just 10 modes; in Fig. 5b, with 100 modes. The error decreased significantly when we use 100 modes, as you can see in Fig. 6a.

The same process can be done for the uncropped yale face data. For uncropped data, in Fig. 1b is the mean image of all the different faces, and the distribution of the singular values can be seen in Fig. 2b, around the first 250 modes account for the nearly 99.9% of the total energy. Our data matrix X's dimension is 32256-by-2432. So roughly, there will be at most 2432 non-zero modes.

We use the `imagesc()` to visualize the eigen-faces due to the basis in U are no longer the uint8 type for image. In Fig. 3b are the first 9 principle modes(eigen-faces). In image reconstruction part, we varied the number of modes we use to re-generate the image and evaluating the goodness of the reconstruction by computing the error $\frac{||X_r - X||}{||X||}$; the less the error, the better the reconstructed images are. You can see in


(a) The mean of all cropped face images

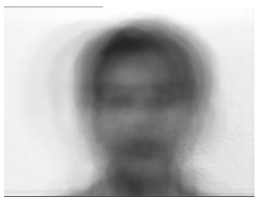

(b) The mean of all uncropped face images

Figure 1: plots of mean for corpped & uncropped face image data

Fig. 5a, with just 10 modes; in Fig. 4b, with 80 modes. The error decreased mildly when we increasing the number of modes, compared to the situation in cropped face data, as you can see in Fig. 6b.

To answer the question listed:

1. Do an SVD analysis of cropped & uncropped images.

You can check the materials above and pictures below for more information.

2. What is the interpretation of the U, Σ and V matrices?

U stores the orthonormal basis which we can use to construct the image in the data matrix. Σ stores the singular values for each basis in U, the larger the singular values, the larger the variance that the whole image data has on that direction. V represents the how the basis coefficients change when expressing different images.

3. What does the singular values spectrum look like and how many modes are necessary for good image reconstruction?

For the cropped face image, rank 100 will be a good approximation, as you can check the relevant reconstruction in Fig. 5b and the error rate in Fig. 6a. The 100 modes make up $\frac{100}{2432} \approx 4.2\%$ of all the modes.

For the cropped face image, rank 100 will be a good approximation, as you can check the relevant reconstruction in Fig. 4b and the error rate in Fig. 6b. The 80 modes make up $\frac{80}{165} \approx 48.5\%$ of all the modes.

4. compare the difference between the cropped (and aligned) versus uncropped images.

We can see that, for the cropped images, due to alignment, we can use fairly small set of the patterns to represent different people's faces. However, for the uncropped images, since SVD can not capture the translation. So the SVD can no give out proper extraction of the dominant features. We can also tell the difference in the distribution of the singular values. In cropped data, the energy is more focused while in the uncropped data, the energy is more dissipated.

- In task to classify music among different artists, bands and genres.

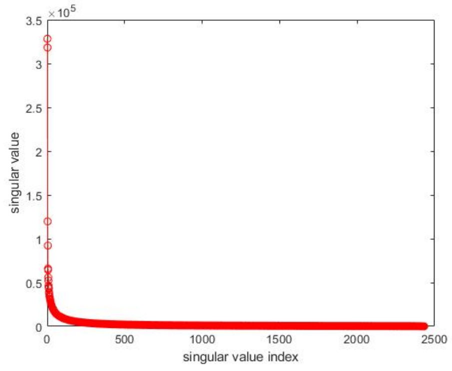
In case1, identifying different bands/artists: See in Fig. 7a for the performance of the model generated by three different training methods. And a 5-second clip's sprectrogram of each kind song are plotted in Fig. 7b.

In case1, identifying different bands/artists but in the same genre: See in Fig. 8a for the performance of the model generated by three different training methods. And a 5-second clip's sprectrogram of each kind song are plotted in Fig. 8b.

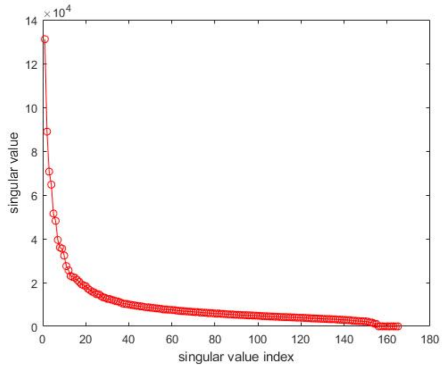
In case1, identifying different genres: See in Fig. 9a for the performance of the model generated by three different training methods. And a 5-second clip's sprectrogram of each kind song are plotted in Fig. 9b.

5 Summary and Conclusions

In distilling the eigenfaces in the Yale face data, we can see that the SVD can extract the eigenfaces successfully in the cropped data, and reconstruct the images with around 4.2% of all the modes. And the reason why SVD failed in the uncropped data is that SVD can not tackle the translation in images, i.e., the in-alignment in the incropped image impairs the SVD to extrct the pattern.

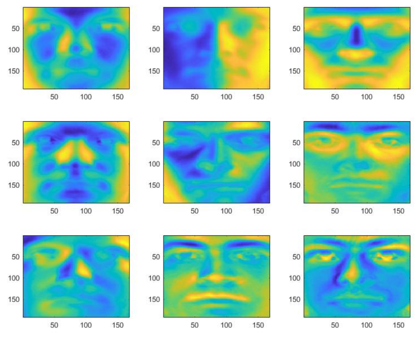


(a) Singular values analysis for all cropped face image

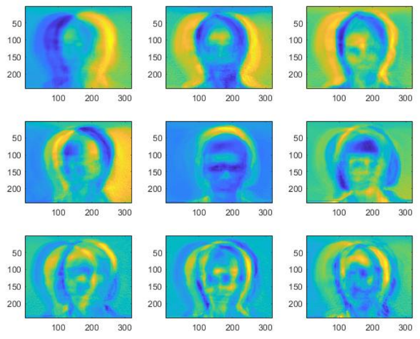


(b) Singular values analysis for all uncropped face image

Figure 2: plots of mean for corpped & uncropped face image data

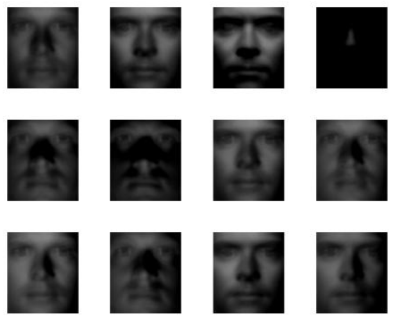


(a) First 9 eigen-faces of cropped images visualization

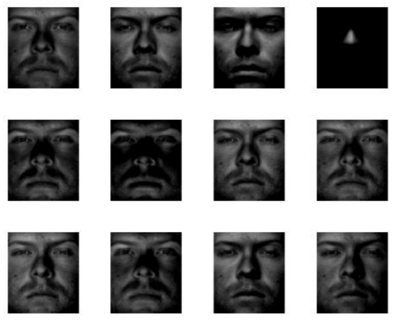


(b) First 9 eigen-faces of uncropped images visualization

Figure 3: plots of First 9 principle modes for corpped & uncropped face image data

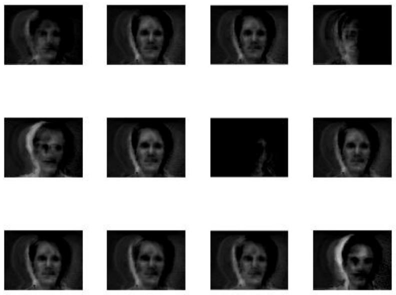


(a) Cropped image reconstruction by picking first 10 modes



(b) Demonstration of image reconstruction by first 100 modes

Figure 4: Demonstration of the cropped image reconstruction by picking first 10 & 100 modes


(a) Demonstration of image reconstruction by 10 modes

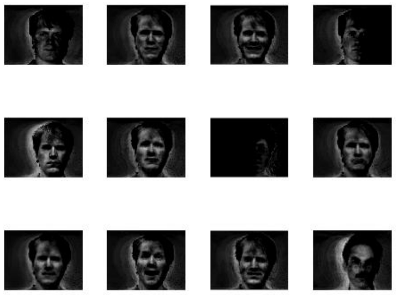
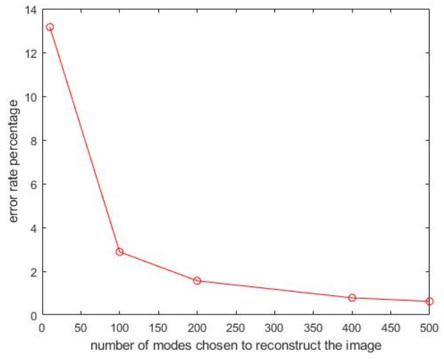
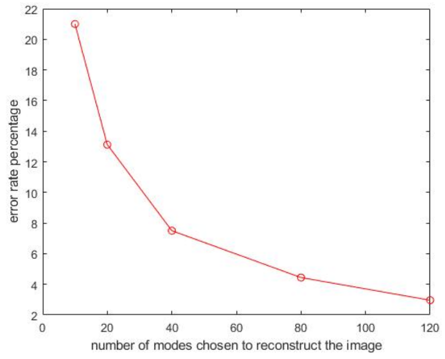

(b) Cropped image reconstruction by picking 80 modes

Figure 5: Demonstration of the uncropped image reconstruction by picking first 10 & 80 modes

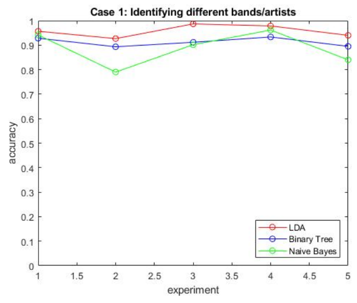


(a) Error change versus using different number of modes

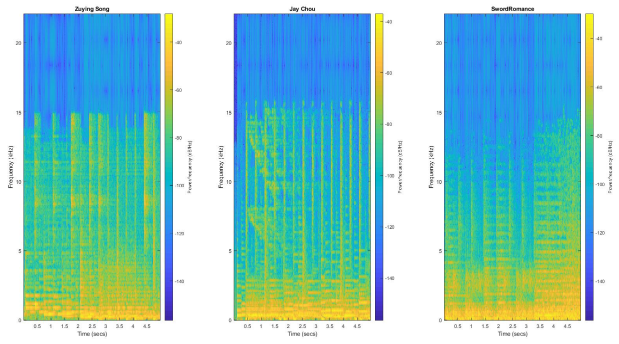


(b) Error change versus using different number of modes

Figure 6: Comparison of error decreasing when more modes are used in image reconstruction between cropped & uncropped face data

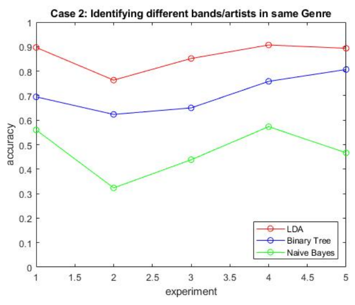


(a) Performance of model generated by different classification methods in case 1

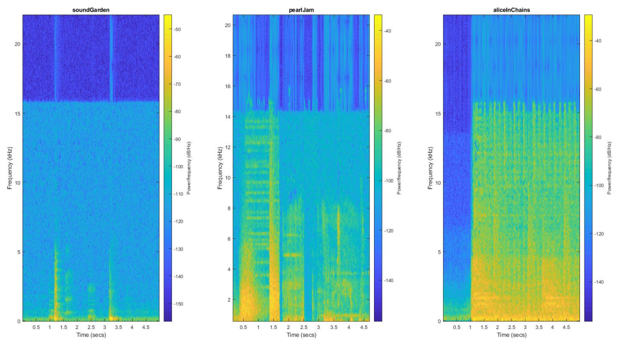


(b) Demonstration of the spectrograms of different kind songs in case 1

Figure 7

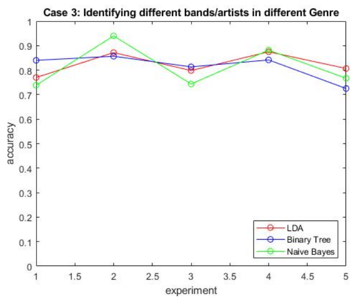


(a) Performance of model generated by different classification methods in case 2

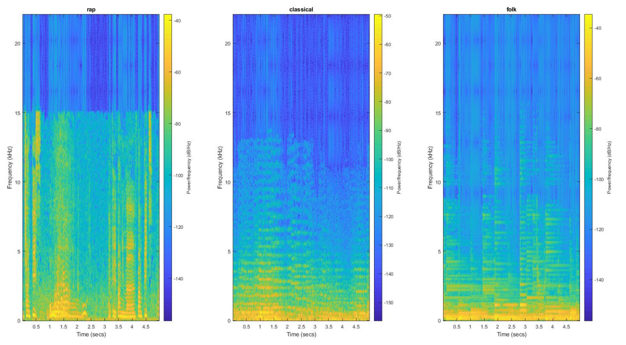


(b) Demonstration of the spectrograms of different kind songs in case 2

Figure 8



(a) Performance of model generated by different classification methods in case 3



(b) Demonstration of the spectrograms of different kind songs in case 3

Figure 9

In classifying music types, by comparing the accuracy in each cases, the case 2 classifying task is the hard to do, i.e., all the three training methods can not give out a good model in distinguish different bands within the same genre, which is understandable, because the music same genre will have more common features, and less distinguishable. Overall, the LDA is the best way to train the model in our specific scenario.

Appendix A MATLAB Functions

Below are the descriptions of the MATLAB function that I used in this report:

- `imagesc(C)` displays the data in array C as an image that uses the full range of colors in the colormap. Each element of C specifies the color for one pixel of the image. The resulting image is an m-by-n grid of pixels where m is the number of rows and n is the number of columns in C. The row and column indices of the elements determine the centers of the corresponding pixels.
- `unidrnd(N)` generates random numbers for the discrete uniform distribution with maximum N.
- `spectrogram(x,window,noverlap,[ ],fs)` Returns the spectrogram of the short-time Fourier transform of input x which uses window to divide the signal into segments and perform windowing and uses noverlap samples of overlap between adjoining segments, with the sample rate as fs.
- `classify(sample,training,group)` classifies each row of the data in sample into one of the groups in training using LDA classifier
- `fitctree(X,Y)` returns a fitted binary classification decision tree based on the input variables contained in matrix X and output Y.
- `fitcnb(Tbl,Y)` returns a multiclass naive Bayes model (Mdl), trained by the predictors in the table Tbl and class labels in the array Y
- `predict(Model, sample)` returns the prediction of input sample by Model.
- `x_train = bootstrap(ntrain,X_tr,length,Fs,npc)` returns the formatted training data *x_ train* though SVD of the Spectrogram generated by raw audio data *X_ tr*; ntrain specifies the number of training samples in x_train; Fs specifies the ; npc specifies the number of features extracted from SVD of the spectrogram.

Appendix B MATLAB Code

```
%% Construct training dataset
ntrain = 800;
ntest = 200;
npc = 2;
acc_lda = [];
acc_cnb = [];
acc_ctree = [];
for kk = 1:5

    testind = unidrnd(floor(0.8*L_x));

    % separate a fraction for extracting training data
    x_tr = x([1:testind-1, testind+ceil(0.2*L_x):end]);
    y_tr = y([1:testind-1, testind+ceil(0.2*L_y):end]);
    z_tr = z([1:testind-1, testind+ceil(0.2*L_z):end]);

    % separate a fraction for extracting test data
    x_te = x([testind:testind+floor(0.2*L_x)]);
    y_te = y([testind:testind+floor(0.2*L_y)]);
    z_te = z([testind:testind+floor(0.2*L_z)]);

    % extact training data
    x_train = bootstrap_construct(ntrain, x_tr, length(x_tr), Fs, npc);
    y_train = bootstrap_construct(ntrain, y_tr, length(y_tr), Fs, npc);
    z_train = bootstrap_construct(ntrain,z_tr, length(z_tr), Fs, npc);

    % %% Constructions
    % Construct labels
    labels = [ones(ntrain,1);2*ones(ntrain,1);3*ones(ntrain,1)];
    % Construct total training dataset
    training = abs([x_train';y_train';z_train']);
    % Construct test dataset
    test_labels = [ones(ntest,1);2*ones(ntest,1);3*ones(ntest,1)];
    x_test = bootstrap_construct(ntest, x_te, length(x_te), Fs, npc);
    y_test = bootstrap_construct(ntest, y_te, length(y_te), Fs, npc);
    z_test = bootstrap_construct(ntest, z_te, length(z_te), Fs, npc);
    sample = abs([x_test';y_test';z_test']);

    % %% Classification
    class = classify(sample, training, labels);
    accuracy = sum(class==test_labels)/length(class);
    Mdl_ctree = fitctree(training, labels);
    class_ctree = predict(Mdl_ctree, sample);
    accuracy_ctree = sum(class_ctree==test_labels)/length(class);
    Mdl_cnb = fitcnb(training, labels);
    class_cnb = predict(Mdl_cnb, sample);
    accuracy_cnb = sum(class_cnb==test_labels)/length(class);
    acc_lda = [acc_lda; accuracy];
    acc_ctree = [acc_ctree; accuracy_ctree];
    acc_cnb = [acc_cnb; accuracy_cnb];
end
```

Listing 1: MATLAB code for paint can's position extraction

```
%% Load songs for case1
close all; clear all; clc;
x1 = audioread('songs/zuyingsong.wav');
x2 = audioread('songs/zuyingsong2.wav');
y1 = audioread('songs/jaychou.wav');
y2 = audioread('songs/jaychou2.wav');
z1 = audioread('songs/swordromance.wav');
z2 = audioread('songs/swordromance2.wav');
x1info = audioinfo('songs/zuyingsong.wav');
Fs = x1info.SampleRate;
x = [x1;x2];
y = [y1;y2];
z = [z1;z2];
L_x = length(x);
L_y = length(y);
L_z = length(z);
clear x1 x2 y1 y2 z1 z2 x1info;

%% model performance evaluation
figure()
plot(acc_lda(1:5), 'ro-');
hold on;
plot(acc_ctree(1:5), 'bo-');
hold on;
plot(acc_cnb(1:5), 'go-');
ylim([0 1])
legend({'LDA','Binary Tree', 'Naive Bayes'},'Location','southeast');
xlabel('experiment')
ylabel('accuracy')
title('Case 3: Identifying different bands/artists in different Genre')

%% demonstration of spectrogram of different categories
figure()
subplot(1,3,1)
pstart = 100000;
pend = pstart + 5*Fs;
clip = x(pstart:pend);
spectrogram(clip,gausswin(500),200,[],Fs,'yaxis');
title('rap')
subplot(1,3,2)
pstart = 100000;
pend = pstart + 5*Fs;
clip = y(pstart:pend);
spectrogram(clip,gausswin(500),200,[],Fs,'yaxis');
title('classical')
subplot(1,3,3)
pstart = 100000;
pend = pstart + 5*Fs;
clip = z(pstart:pend);
spectrogram(clip,gausswin(500),200,[],Fs,'yaxis');
title('folk')
```

Listing 2: MATLAB code for doing PCA

```
%% Yale cropped face SVD
% how much difference by comparing the norm(X-X_r)/norm(X)

% load data
rootdir = 'CroppedYale';
filelist = dir(fullfile(rootdir, '**\*.*'));  %get list of files and folders in any subfolder
filelist = filelist(~[filelist.isdir]);  %remove folders from list
numPic = length(filelist);
X = zeros(192*168,numPic);
% reshape image & store them in data Matrix X
for kk = 1:numPic
    data = filelist(kk).name;
    path = filelist(kk).folder;
    pic = imread([path '\' data]);
    X(:,kk) = double(pic(:));
end
% svd analysis
[m,n] = size(X);
mn = mean(X,2);
X = X - repmat(mn,1,n);
[u,s,v]=svd(X,'econ');
Y = u' * X;
sig = diag(s);
% plot avg face;
figure();
avg = reshape(mn,192,168);
imshow(uint8(avg));
% plot singular values
figure();
plot(sig,'ro-');xlabel('singular value index');ylabel('singular value');
% plot mode
figure();
for ii = 1:9
    subplot(3,3,ii);
    mode = reshape(u(:,ii),192,168);
    imagesc(mode);
    axis ij; title(['PC',num2str(ii)])
end
%% cropped face reconstruction
numMode = [10 100 200 400 500];
for ll = 1:length(numMode)
    feature = numMode(ll);
    cord = u.';
    X_recon = (u(:,1:feature)*cord(1:feature,:))*X;
    error = norm(X_recon-X)/norm(X)
    figure();
    for pp = 1:12
        subplot(3,4,pp);
        imshow(uint8(reshape(X_recon(:,pp),192,168)));
    end
end
%% plot error decreasing
error = [13.16 2.88 1.56 0.78 0.62];
x = [10 100 200 400 500];
figure();plot(x,error,'ro-');xlabel('number of modes chosen to reconstruct the image');
ylabel('error rate percentage');
```

Listing 3: MATLAB code for visualizing movement points