Facial Features Extraction and Music Genre Recognition with Machine Learning

Quinn Luo

March 2019

Abstract

This report illustrates how to extract main features from images and perform classification, in an example of music genre recognition. It is to show you how machine learning can be done in practice. I will start with introducing the related background knowledge, followed by an explanation of algorithms implemented, summarized by the results and conclusion from this report.

Introduction

In the contemporary machine learning, singular value decomposition are employed to extract out the main features of data so that classification can be done based on the analysis of SVD. However, the theory and algorithms of the method is elusive. This report gives two examples of how principal component analysis and classification can be done in practice.

Theoretical Background

Singular Value Decomposition(SVD)

A matrix in $\mathbb{C}^{m\times n}$ can be decomposed in the following way:

$$A = U\Sigma V^*$$

where $U \in \mathbb{C}^{m\times n}$, $V \in \mathbb{R}^{n\times n}$, and $V^* \in \mathbb{C}^{n\times n}$.

This decomposition comes from the idea of seeing matrix operation as a process of rotating and stretching. In detail, $A\vec{x} = \vec{b}$ can be visualized as rotating the vector $\vec{x}$ and stretching it to get to $\vec{b}$. SVD implements this idea of rotating and stretching by decomposing the matrix A into two unitary matrices U, V^* as rotating matrices and one diagonal matrix S as scaling matrix. The diagonal values of Σ are lined up in descending order and they correspond to the vector in U and V^*.

Generally speaking, singular value decomposition extracts the main features from the data.

Clustering and Classification

In machine learning, one of the tasks is to classify objects by their features. For example, suppose we take samples of two different types of birds measure their weight, and we find that one type has typically higher weight than the other. We conclude that there are correlation between weight and type. Hence, we are able to predict the type of bird from the measurement of their weight. In this example, indicator "weight" is the main feature of the birds. As for any unknown system, the SVD algorithm introduced above is able to extract the dominant features out of that system.

Mathematically, the classification process can be visualized as clusters. Based on the assumption that similar items have similar features, we expect that in the coordinate of these features, the studies objects would be close to similar objects, geographically, thus forming a cluster. In the example of the different types of birds and their weights above, if we represent each sample by their weight and lay these values of weight as point on the axis of weight, there would be two clusters, one for each type.

Especially for this report, I employ three algorithms of classification, including **K-nearest neighbors**, **Naive Bayes**, and **Linear Discriminant Analysis**, all of which are for supervised learning. The following is a detailed explanation of their implementation.

- **K-nearest neighbors** finds the number of **K** nearest neighbors in distance from the current dataset to the given point that is tested on. The most common label in these neighbor points is the label given by the model.
- **Naive Bayes** labels the test point based on an analysis of conditional probability.
- **Linear Discriminant Analysis** finds the linear combination of features that separates classes in the data. Assuming that data are normally distributed, LDA searches for a coordinate system that can maximize the difference between groups.

Algorithm Implementation and Results

Part I: Facial Features Extraction

1. **Loading data and building matrix**
   After downloading the compressed file and unzip them into a folder, I check the directory of the dataset and find that all data are divided into section and stored in sub-folders. Hence, I write a code to retrieve data by looping through all sub-folders and the files inside each sub-folders. The next step is to reshape these data, stored as matrix into a vector. And we build our matrix A which contains data from all faces.

$$A = \begin{bmatrix} Face1 & Face2 & Face3 & ... \end{bmatrix}$$

2. **SVD analysis on A**
   I center the mean of A at 0 by subtracting the mean value of all entries in A from each

entry. Subsequently, I perform the singular value decomposition on the matrix A to extract the main features out of all face pictures collected.

3. **Result**
   Plotting Σ, figure 1 shows the singular value spectrum of modes of SVD from both cropped images and uncropped images. By comparison, there are fewer modes with high energy levels for the cropped images; that is, modes of cropped images are either highly dominant or trivial but the modes of uncropped images displays a smoother curve. This is due to the fact that the cropped images are aligned and backgrounds are cut out thus less noise introduced. The difference in these two energy plots also indicate that cropped images are favored in terms of extracting main features of faces.

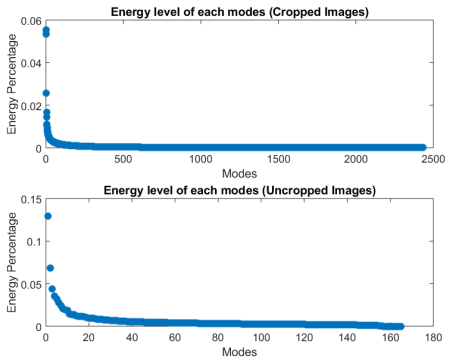

Figure 1: Energy level of modes

The matrix U is an orthonormal basis. Each column in U is a mode in the singular value decomposed A. By reshaping each column back to a matrix and plotting the matrix, figure 2 displays the first a few modes of cropped images, figure 3 for uncropped ones. In other words, these are the dominant features out of all faces, also referred to as eigenfaces. Comparing eigenfaces between cropped images and uncropped images, the conclusion above is confirmed that cropped and aligned ones are preferred for data analysis.

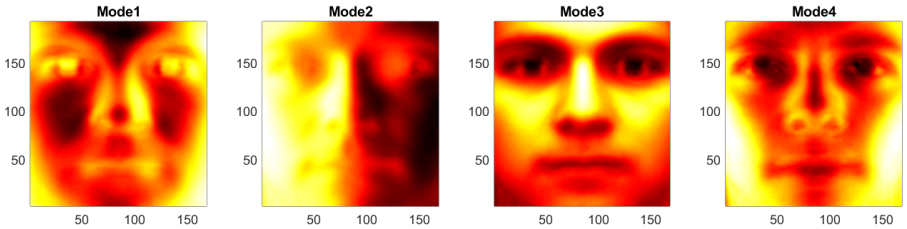

Figure 2: Dominant eigenfaces (cropped)

Lastly, V is also an orthonormal matrix. Each row of V^* is a projection of A onto the basis U.

4. **Reconstructing the matrix with truncation**
   Looking at the energy plot of the modes, any points with x-axis right to 100 approach 0, and for uncropped images I decide to cut off at 80. The process of truncating at

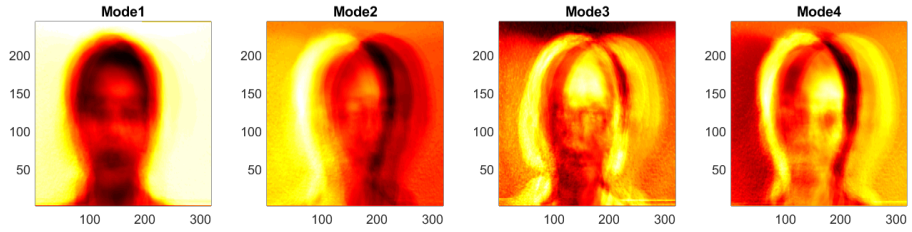

Figure 3: Dominant eigenfaces (uncropped)

rank r only preserves the first r columns of U, the first r values of the diagonal of Σ, and the first r rows of V^*. I am able to reconstruct the matrix A by computing

$$\hat{A} = U(:, 1:r) \cdot \Sigma(1:r, 1:r) \cdot V^*(1:r, 1),$$

Figure 4 compares one of the original face and the reconstructed face after truncation at rank r of cropped images. Figure 5 is similar only of uncropped image. Even with rank as small as 100 out of 2432, cropped images are easier to reconstruct than uncropped image which are truncated at a higher percentage of 80 out of 165.

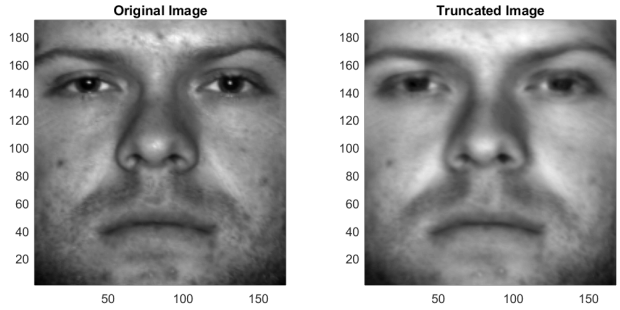

Figure 4: Reconstructed image truncated at rank r (cropped)

Figure 5: Reconstructed image truncated at rank r (cropped)

Part II: Music Genre Classification

1. **Data processing**
   Loading data is a similar fashion to the first part; however, this time an audio file that is stores as a two-column matrix Y and data-density FS, the number of data points

in one second. After summing two columns, I obtain the vector to work with.

I then cut each music file into many 5-second pieces by cutting $5 \cdot FS$, the number of data points in 5 seconds. However, all audio files do not have the same sampling rate, so I resample my data at ratio of $\frac{20000}{FS}$, i.e., taking 20000 points from one second. I call the resampled data x.

The next procedure of data processing is to obtain the spectrogram of these audio files. And on getting the spectrogram of each piece, I am able to reshape them into column, obtaining the matrix A and perform the singular value decomposition as the first part.

Because the audio files are in the order of genres, the columns of A that represent music pieces lay in order. In this report, there are 192 pieces of music in total and each genre takes up 64 out of 192.

2. **SVD and visualization of feature space**
After performing SVD on zero-centered A, I have the feature space basis U, the energy of mode Σ, and the projection of spectrogram V. Because each column of V stores information for a music piece and each row represent a dimension in the U basis, plotting the columns of V on the axes of the rows of V helps visualize how audio files differ in terms of the dominant features. I randomly divide the data into two parts and the first part is used for training. Figure 6 shows points in training set in the basis of first two modes, different genres in different colors. It turns out that two modes are not enough to separate my data.

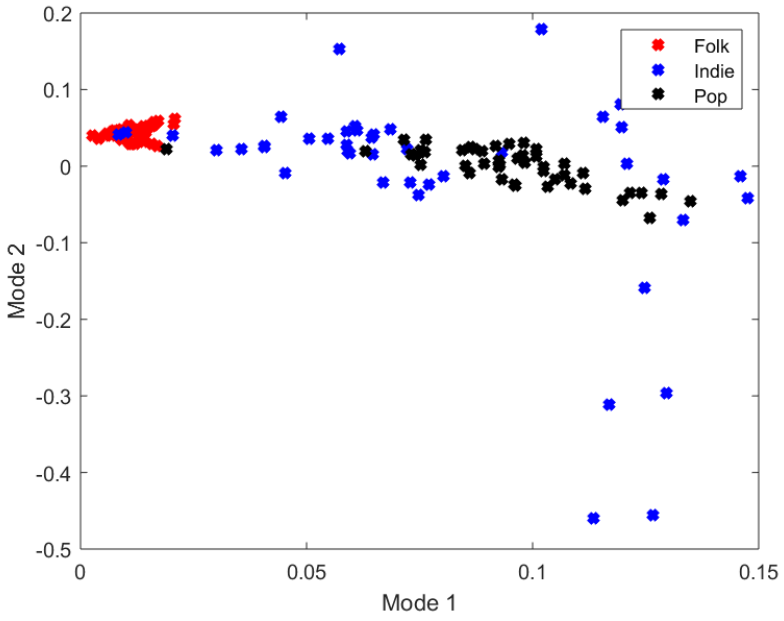

Figure 6: Clusters on two modes

Figure 7 sits the points onto a 3-D system with three modes. It separates the data

with different labels clearly.

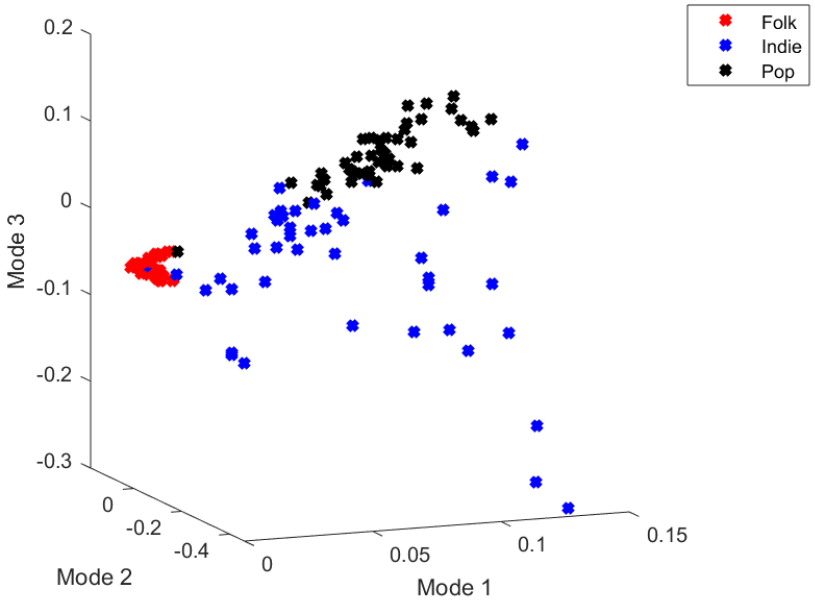

Figure 7: Clusters on three modes

3. **Classification**
   Here I employ three algorithms: K-nearest neighbors, Naive Bayes, and Linear Discriminant Analysis to classify the second part of my data, the testing set. Comparing the expected labels, the true labels of these points to the actual labels that the classifier gives, figure 8 shows how accurate these algorithms are.

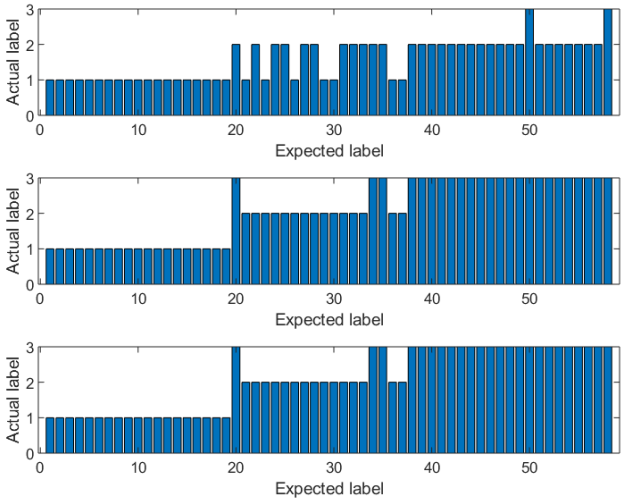

Figure 8: Efficacy of classifiers in one run (test 1)

In the spirit of cross-validation, figure 9 reports the accuracy of these algorithms of 100 random runs.

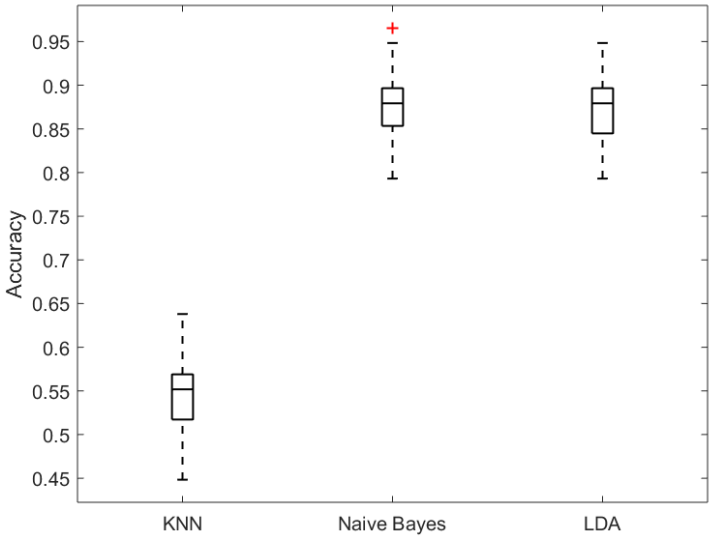

Figure 9: Accuracy of classifiers (test 1)

4. **Tests on different data**
The data used in test 1 are from three artists of different genres. However, test 2 performs the same algorithms on three artists in the same genre. The reported accuracy is shown in figure 10.

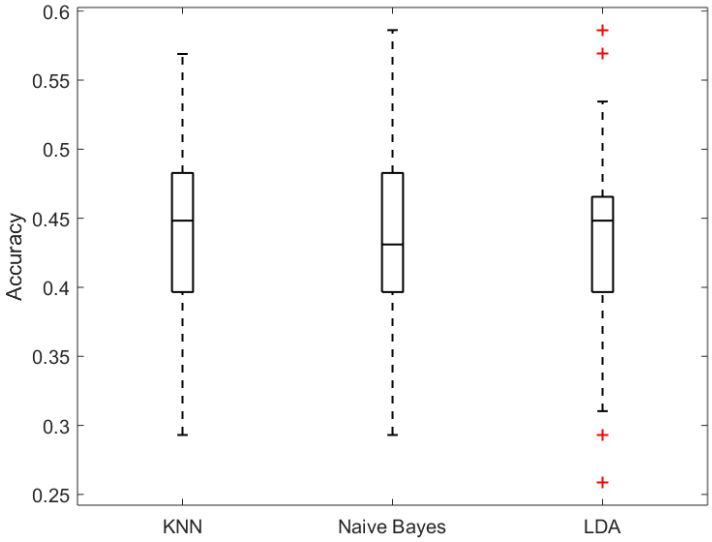

Figure 10: Accuracy of classifiers (test 2)

As for test 3, multiple artists are chosen in each genre, accuracy shown as figure 11.

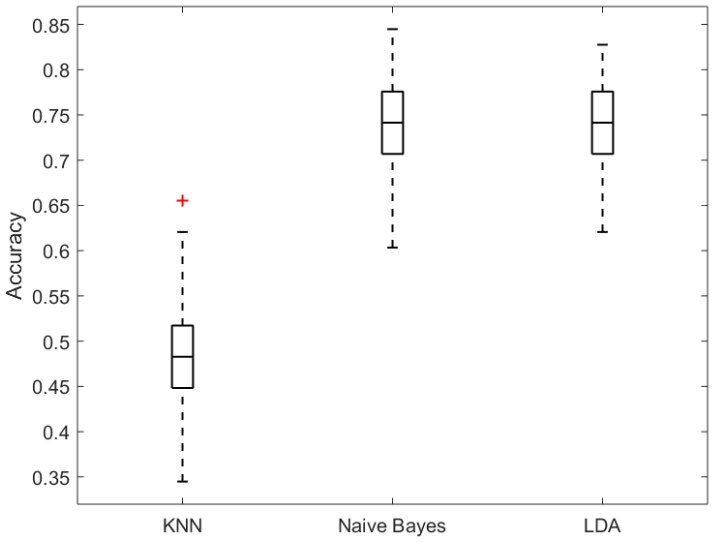

Figure 11: Accuracy of classifiers (test 3)

5. **Results**

   (a) The accuracy of each algorithm differs, depending on the nature of data: KNN performs generally worse than others. I assert that it is because data from one of the genre are not clustered even though they are separated from others.

   (b) From test 1 to test 2, as genres change from different one to the same one, the data become similar. Thus, the accuracy of classifier decreases. While test 3 works with data from different genres and different artists, the accuracy is between that of test 1 and 2.

Conclusion

In this report, images are processed to extract the main features. And I conclude that cropped and aligned images are better to analyze. With rigorous analysis and cross-validation, I am able to show an example of classification to illustrate the methods of modern machine learning. I find that the accuracy of classifiers highly depends on the nature of data.

Appendix I: MATLAB commands explained

`dir(folder)` returns the information of files stored in the folder

`extractfield(X, field)` returns the specified field value of X

`imread(X)` reads in the image file X

`strcat(v1, v2)` combines string v1 and v2 to a single string

`reshape(v, [dim])` returns a vector of dimension dim reshaped from v

`[Y, FS] = audioread(X)` returns the vector Y representing the audio file data and the sampling rate FS

`resample(X, P, Q)` returns a truncated X by only taking P points from every Q points

`spectrogram(X)` returns the short-time Fourier transform of X

`randperm(n)` shuffles the number from 1 to n

`knnsearch(train, test, 'K', k)` returns the k nearest neighbor of points in train, given points in test

`fitcnb(train, label)` returns a model with all training data labeled

`model.predict(test)` returns the label of test data from the model

`classify(train, test, label)` returns the label of test data based on the linear discriminant analysis classification of the train and its label

Appendix II: MATLAB code

Part I

```
clear all; close all; clc
%load data
A = [];
list = dir('CroppedYale');
folders = extractfield(list, 'name');
folders = folders(3:end, :);
for i = 1:length(folders)
    sublist = dir(strcat('CroppedYale/', folders{i}));
    subfolders = extractfield(sublist, 'name');
    subfolders = subfolders(3:end);

    for j = 1:length(subfolders(:, 1))
        img = imread(strcat('CroppedYale/', folders{i}, '/', subfolders{j}));
        img = reshape(img, [192*168, 1]);
        A = [A img];
    end
end
%%
A = double(A);
[u, s, v] = svd(A - mean(A(:)), 'econ');
plot(diag(s)/sum(diag(s)), 'x', 'LineWidth', 2)
xlabel('Principal Component(Dominant Feature)');
ylabel('Energy Percentage');

%%
for i = 1:4
    re = flip(u(:, i));
    re = reshape(re, [192, 168]);
```

```
    subplot(1, 4, i)
    pbaspect([1 1 1])
    pcolor(re); shading interp; colormap(hot);
    title(strcat('Mode', num2str(i)))
end
%%
figure(2)
for i = 1:4
    re = flip(u1(:, i));
    re = reshape(re, [243, 320]);
    subplot(1, 4, i)
    pbaspect([1 1 1])
    pcolor(re); shading interp; colormap(hot);
    title(strcat('Mode', num2str(i)))
end
%%
B = [];
list = dir('yalefaces');
faces = char(extractfield(list, 'name'));
faces = faces(3:end, :);
for i = 1:length(faces)
    img = imread(strcat('yalefaces/', faces(i, :)));
    img = reshape(img, [243*320, 1]);
    B = [B img];
end
B = double(B);

[u1, s1, v1] = svd(B - mean(B(:)), 'econ');
plot(diag(s1)/sum(diag(s1)), 'x')

%%
subplot(2, 1, 1)
plot(diag(s)/sum(diag(s)), '*', 'LineWidth', 2)
```

```
xlabel('Modes');
ylabel('Energy Percentage');
title('Energy level of each modes (Cropped Images)');

subplot(2, 1, 2)
plot(diag(s1)/sum(diag(s1)), '*', 'LineWidth', 2)
xlabel('Modes');
ylabel('Energy Percentage');
title('Energy level of each modes (Uncropped Images)');
%%
subplot(1, 2, 1)
o = flip(reshape(A(:, 1), [192 168]));
pcolor(o); shading interp; colormap(gray)
title('Original Image')
k = 100;
r = u(:, 1:k)*s(1:k, 1:k)*v(:, 1:k)';
r = r(:, 1);
r = flip(reshape(r, [192 168]));
subplot(1, 2, 2)
pcolor(r); shading interp; colormap(gray)
title('Truncated Image')
%%
figure(2)
subplot(1, 2, 1)
o = flip(reshape(B(:, 1), [243 320]));
pcolor(o); shading interp; colormap(gray)
title('Original Image')
subplot(1, 2, 2)
k = 80;
r = u1(:, 1:k)*s1(1:k, 1:k)*v1(:, 1:k)';
r = r(:, 1);
r = flip(reshape(r, [243 320]));
pcolor(r); shading interp; colormap(gray)
```

```
title('Truncated Image')
```

Part II

```
clear all; close all; clc
A = [];
loc = strcat('music_test2/');
m = 1;
list = dir(loc);
genres = extractfield(list, 'name');
genres = genres(3:end);
keep = [ones(64*m, 1); 2*ones(64*m, 1); 3*ones(64*m, 1)];
for o = 1:length(genres)
    subloc = strcat(loc, genres{o}, '/');
    list = dir(subloc);
    songs = extractfield(list, 'name');
    songs = songs(3:end);
    for i = 1:length(songs)
        [Y, FS] = audioread(strcat(subloc, songs{i}), 'double');
        time = 1;
        for j = 10:5:45
            x = Y(j*FS:(j+5)*FS, :);
            x = (x(:, 1) + x(:, 2))./2;
            x = resample(x, 20000, FS);

            x = abs(spectrogram(x));
            x = reshape(x, [16385*8, 1]);
            A = [A x];
        end
    end
end
%%
```

```
[u, s, v] = svd(A-mean(A(:)), 'econ');
plot(diag(s)./sum(diag(s)), 'o')

%%
p = v';
plot(p(1, 1:45), p(2, 1:45), 'rx', 'LineWidth', 3); hold on
plot(p(1, 65:109), p(2, 65:109), 'bx', 'LineWidth', 3); hold on
plot(p(1, 129:173), p(2, 129:173), 'kx', 'LineWidth', 3);
legend('Folk', 'Indie', 'Pop')
xlabel('Mode 1'); ylabel('Mode 2')

%%
p = v';
d = [1 2 3];
plot3(p(d(1), 1:45), p(d(2), 1:45), p(d(3), 1:45), 'rx', 'LineWidth', 3); hold on
plot3(p(d(1), 65:109), p(d(2), 65:109), p(d(3), 65:109), 'bx', 'LineWidth', 3);
plot3(p(d(1), 129:173), p(d(2), 129:173), p(d(3), 129:173), 'kx', 'LineWidth', 3);
legend('Folk', 'Indie', 'Pop')
xlabel('Mode 1'); ylabel('Mode 2'); zlabel('Mode 3')

%%
percentage = zeros(100, 3);
for o = 1:100
    di = 3;
    p1 = randperm(64*m); p2 = randperm(64*m)+64*m; p3 = randperm(64*m)+128*m;
    train_range = [p1(1:45*m) p2(1:45*m) p3(1:45*m)];
    train = p(1:di, train_range)';
    test_range = [p1(45*m+1:64*m) p2(45*m+1:64*m) p3(45*m:64*m)];
    test = p(1:di, test_range)';
    point = 3;
    label = knnsearch(train, test, 'K', point, 'IncludeTies', true);

    test_label = zeros(length(test(1, :)), 1);
```

```
    for i = 1:length(label)
        count = [0 0 0];
        x = label{i};
        for j = 1:point
            if keep(x(j)) == 1
                count(1)  = count(1) + 1;
            end
            if keep(x(j)) == 2
                count(2)  = count(2) + 1;
            end
            if keep(x(j)) == 3
                count(3)  = count(3) + 1;
            end
        end
        [M, I] = max(count);
        test_label(i) = I;
    end
    percentage(o, 1) = sum(test_label == keep(test_range))/length(test_range);

%     subplot(3, 1, 1)
%     bar(test_label);
%     xlabel('Expected label'); ylabel('Actual label');

    %
    train_label = keep(train_range);
    model = fitcnb(train, train_label);
    test_label = model.predict(test);
%     subplot(3, 1, 2)
%     bar(test_label);
%     xlabel('Expected label'); ylabel('Actual label');
    percentage(o, 2) = sum(test_label == keep(test_range))/length(test_range);

    %
```

```
    test_label = classify(test, train, train_label);
%     subplot(3, 1, 3)
%     bar(test_label);
%     xlabel('Expected label'); ylabel('Actual label');
    percentage(o, 3) = sum(test_label == keep(test_range))/length(test_range);
end
figure
boxplot(percentage, 'Labels', {'KNN','Naive Bayes', 'LDA'},
'Colors', 'k', 'Width', 0.1);
set(findobj(gca,'type','line'),'linew',1)
ylabel('Accuracy')
```

References

Kutz, J. N., (2013). Data-driven modeling scientific computation: Methods for complex systems big data. Oxford: Oxford University Press.