Music Classification

AMATH 482 Homework 4

Robin Yan

Mar. 3, 2020

Abstract

Classification, also known as supervised machine learning, can identify and predict the categorization of new data based its characteristics and those of the trainer data. This homework utilizes k-nearest neighbors (KNN) classification based on singular value decomposition (SVD) to identify music samples from various artists belonging to different genres. The music samples fall under three genres: classical, country and electronic. The script uses SVD to extract the characteristics of music samples to build trainers and classify new test data. Three sets of classification are carried out: bands cross genre, bands within genre and generalized genre classifier. All processes are repeated using v-fold cross validation to ensure the robustness of the configuration. The averaged accuracy of three tests are: 92.6%, 63.0% and 84.3%.

Section I – Introduction and Overview

The purpose of this homework is to explore classification techniques in data analysis through a series of tests on music samples: Test 1 - band classification from different genre; Test 2 - band classification from same genre; Test 3 - genre classifier with multiple bands.

The music samples being analyzed are 61 stereo mp3 files from YouTube music library. The use of the music fall under creative commons and is not intended as copyright infringement. There are three genres, each with three artists being analyzed: 1) Classical: 4 pieces from Bach, 4 pieces from Beethoven and 9 pieces from Mozart; 2) Country: 9 pieces from Chris Haugen, 9 pieces from Dan Lebowitz and 9 pieces from Nat Keefe; 3) Electronic: 4 pieces from Rondo Brothers, 4 pieces from Spence and 9 pieces from Quincas Moreira.

The music used in all three tests are: Test 1 – Mozart (9 pieces), Chris Haugen (9) and Quincas Moreira (9); Test 2 – Chris Haugen (9), Dan Lebowitz (9) and Nat Keefe (9); Test 3 – Classical all (17), Country all (27) and Electronic all (17). A detailed list of all music is attached in Appendix 3.

Upon importing, the script converts the stereo mp3 files into mono channel amplitude vector. The vector is then converted into spectrogram for overtone identification and reshaped back to a vector as one observation. All observations are combined to form the data matrix.

This script uses SVD to reduce the dimensionality of the data matrix and extract characteristic information. A random portion of the data is then selected as test data and the remaining as trainer data belonging to three categories. The script uses KNN algorithm to classify the test data into those three sets, compares the results with original tag to calculate the accuracy, and the averaged accuracy from all cross validation is considered to be the overall accuracy of the classification script.

Section II – Theoretical Background

Singular Value Decomposition

Standard singular value decomposition of an m by n matrix X takes the form of:

$$X = U * \Sigma * V^T \quad (E.1)$$

In which U and V are both unitary square matrices with dimensions m by m and n by n, and Σ is an m by n matrix consisting of a diagonal part of m by m filled with silent rows/columns to reach m by n dimension. By convention, m is the number of variables and n is the number of observations. In this application, the SVD

implemented is the “econ” version, which excludes the silent rows/columns for better performance, and the U matrix has dimension m by n.

Modal Scores

Based on results from SVD, and the idea of principal component analysis from Report 3, each column of the U matrix represent one mode in the original data and each diagonal entry of the Σ matrix denotes the magnitude of the mode in decreasing order.

Therefore, the modal scores are calculated by:

$$Y = U^* * X \quad (E.2)$$

In which U^* is the transpose of matrix U from SVD, for real numbers. Each column of the Y matrix contains the modal scores corresponding to all modes for each column/observation of the X matrix. Each modal score is the dot product between the observation and the mode. The original data X of m by n can be fully described with modal scores Y and mode matrix U.

In practice, the first few rows of Y would be sufficient to capture the main characteristics of the original data; hence the dimensionality of original data is reduced.

K-Nearest Neighbors

KNN classification determines the class of new data point based on the vote from the first k neighbors of the data point based on distance. In this script, the Euclidean distance is used, and is given by:

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2 + \cdots} \quad (E.3)$$

In which x, y, z are the first three dimensions of the data, followed by additional dimensions if present.

Section III – Algorithm Implementation and Development

Spectrogram and Data Matrix

Upon loading, the audio files are checked to determine if they are stereo (2 channels) or mono (1 channel), and stereo files are averaged between two channels to achieve consistency with mono files. The script then truncates the audio based on its sampling frequency to extract only audio information between 55 and 60 seconds. The excerpt is then resampled by a factor of 5 to reduce computation demand, and the spectrograms are generated using MATLAB’s built-in $spectrogram$ command. The window width is iterated and determined at 1000 to provide sufficient time and frequency resolution.

The frequency columns in the spectrogram matrix are concatenated to form a vector, as one observation. Multiple data vectors from different music samples are stacked together to form the data matrix.

A separate index vector is generated to tag the data.

Test Data Selection

Because the observations for each class is set at 9, this script uses 9-fold cross validation by randomly choosing one observation as test data and excluding it from the training set. This is implemented by using the command $randperm$ to randomly rearrange 1-9 and iterate through all of them as test data. This ensures all data points are used both as trainer and test data.

SVD and Modal Scores

SVD (E.1) on the training data is carried out with MATLAB's built-in function svd, which outputs three matrices U, Σ and V. Modal scores are calculated based on E.2, but only the first three rows are used, which corresponds to approximately 50% modal power. Since the goal is not to fully reconstruct the data, it is not necessarily beneficial to include more modal scores to achieve higher modal power. The choice of three rows helps extract the most distinct features from the data. The selected modal score matrix forms the trainer matrix.

The scores for each test data are also calculated based on E.2, using the U matrix from training data, and only the first three rows are chosen.

KNN

This script then uses built-in command $knnsearch$ to identify the k closest neighbors from the trainer to the test data. The distance calculation is based on Euclidean distance defined in E.3. The classification of the test data is voted by the neighbors. The variable k is iterated and determined at 3 for optimal performance.

The voted classification is compared with original data tag to determine the success rate with current configuration. The all success rates with the 9 choices of test data are averaged to compute the overall accuracy of the above algorithm.

Section IV – Computational Results and Discussions

Spectrograms

The spectrogram of each music sample being analyzed is generated. The spectrograms for 9 pieces of music from Mozart (Classical) and 9 from Chris Haugen (Country) during Test 1 are shown below in Fig. 1 and 2 for illustration.

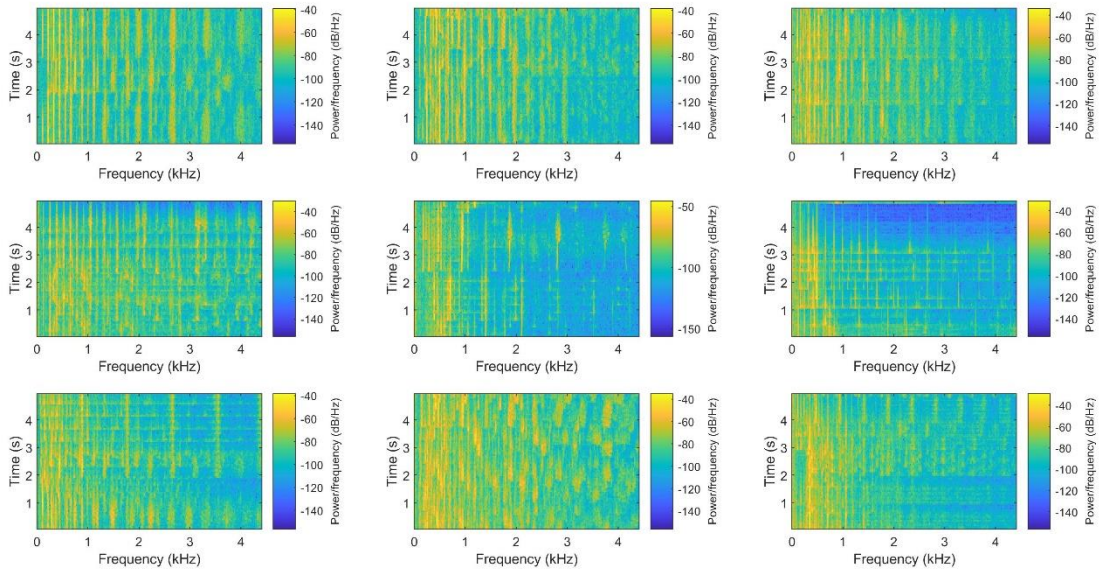

Fig. 1 Spectrograms of music samples from Mozart (between 55 sec to 60 sec)

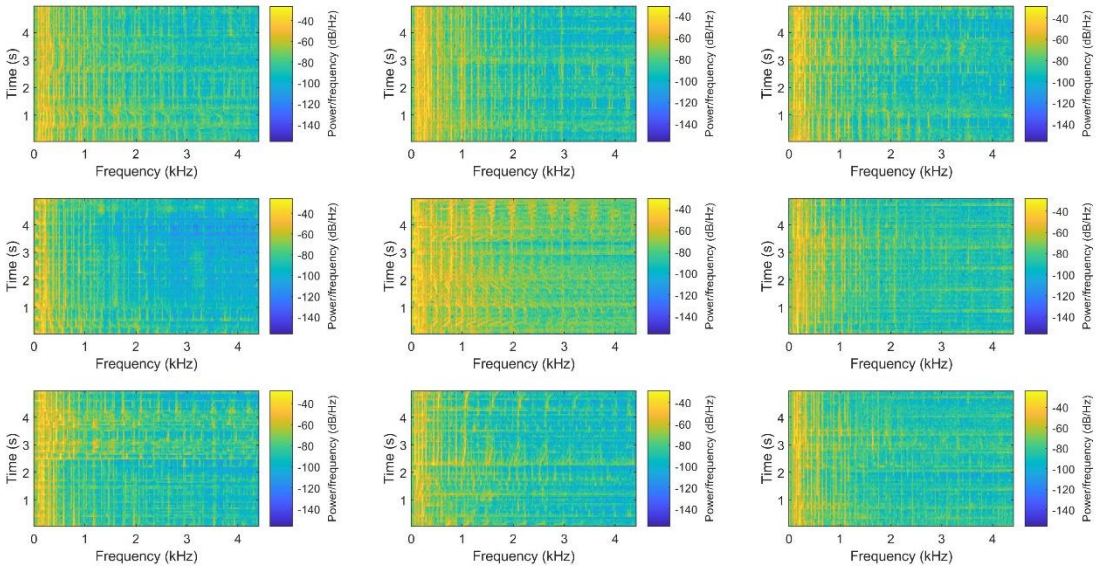

Fig. 2 Spectrograms of music samples from Chris Haugen (between 55 sec to 60 sec)

As seen above, there are some distinct features within each figure. Fig. 1 shows more complex overtones while Fig. 2 shows more periodic patterns. The complex overtones could be attributed to the large number of instruments used in orchestra where the overtones of all instrument are mixed to form the music. The periodic pattern in Fig. 2 could be attributed to the nature of country music, with emphasis on rhythm. The script aims at quantifying the above observations for music classification.

Modal Scores

Since the algorithm uses only first three modal scores, the data can be readily visualized with scatter plots. The modal scores from trainer during one random trail from Test 1 are shown below in Fig. 3.

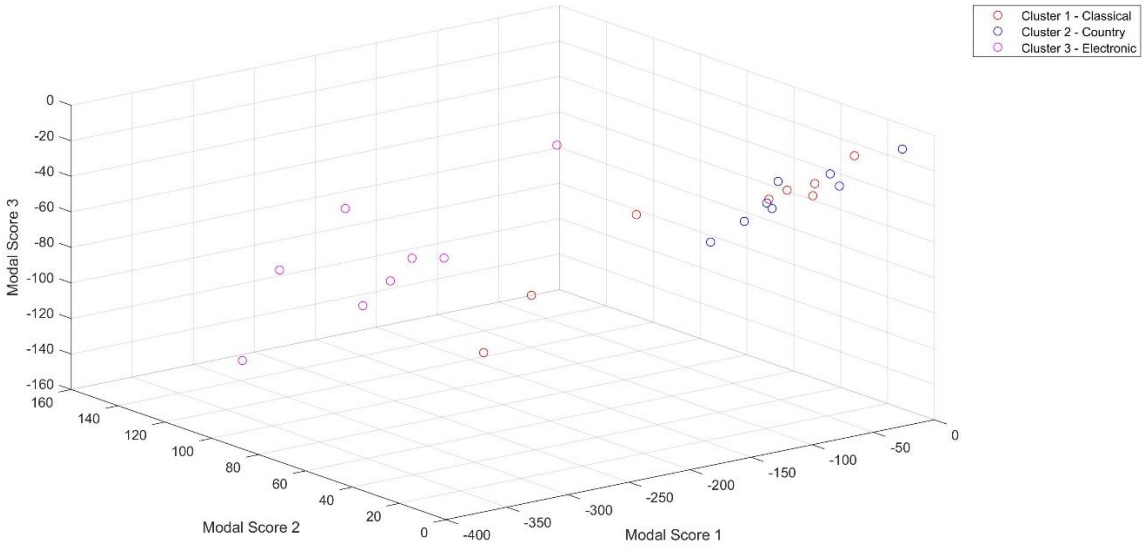

Fig. 3 Modal Scores of trainer data from one random trial during Test 1

As shown above, electronic music is highly diverse but distinct from the remaining samples, possibly due to the wide sound range possible by electronic mixer. Country music appears to be the least diverse, whilst there are commonalities between country music and classical music.

Classifier Accuracy

Test 1 – Cross Genre Band Classification

The classification results of all trials from Test 1 are summarized below in Table 1. Set 1 is Mozart from Classical, set 2 is Chris Haugen from Country, set 3 is Quincas Moreira from Electronic.

Table 1 Classification results from Test 1 (error highlighted)

	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7	Trial 8	Trial 9
Test Data 1	1	1	1	1	1	1	1	1	1
Test Data 2	2	2	2	2	2	2	2	2	2
Test Data 3	3	3	3	3	**2**	3	**2**	3	3

In which test data 1, 2, 3 are extracted from set 1, 2, 3 respectively. The algorithm incorrectly identified test data 3 two times, possibly due to the high variation within Electronic music.

Overall accuracy of Test 1 is 92.6%.

Test 2 – Same Genre Band Classification: Country

The classification results of all trials from Test 2 are summarized below in Table 2. Set 1 is Chris Haugen, set 2 is Dan Lebowitz and set 3 is Nat Keefe.

Table 2 Classification results from Test 2 (error highlighted)

	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7	Trial 8	Trial 9
Test Data 1	**2**	1	**2**	1	1	1	1	1	1
Test Data 2	2	**1**	2	2	2	**3**	2	**1**	2
Test Data 3	**2**	3	3	**2**	**2**	3	3	**2**	**2**

As expected, the algorithm is more prone to error when classifying music from within the same genre. There are mistakes in all three sets, due to the similarity in overtones from the same types instrument within genre.

Overall accuracy of Test 2 is 63.0%, much lower than Test 1, but still significantly better than brute guess (33%).

Test 3 – Multi-Band Genre Classification

The results from Test 3 are summarized below in Table 3. Set 1 is Classical, set 2 is Country and set 3 is Electronic.

Table 3 Classification results from Test 3 (error highlighted)

	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7	Trial 8	Trial 9
Test Data 1	1	1	1	1	1	1	1	1	1
Test Data 2	**1**	2	2	2	2	2	2	2	2
Test Data 3	3	**2**	3	**2**	3	3	3	**2**	3
	Trial 10	Trial 11	Trial 12	Trial 13	Trial 14	Trial 15	Trial 16	Trial 17	
Test Data 1	**2**	**2**	1	1	1	1	1	1	
Test Data 2	2	2	2	2	2	2	2	**1**	
Test Data 3	3	3	3	**2**	3	3	3	3	

The accuracy of Test 3 is slightly lower than Test 1, potentially due to the variation introduced by incorporating more bands within the same genre. But the overall accuracy is still satisfactory.

Overall accuracy of Test 3 is 84.3%.

Section V – Summary and Conclusions

Based on the results, the script based on SVD and KNN can successfully classify music across genre and even within genre, all with satisfactory accuracy. Recommended next steps would be to incorporate more data for both training and testing, to improve the robustness of the method and finely tune certain arbitrarily determined parameters. Additionally, future project can use this script to classify music from other genres to further test the capability of this algorithm. In conclusion, classification is a powerful and versatile tool in data analysis.

Appendix A – MATLAB functions used

- $audioread$: read audio files into one (mono) or two (stereo) amplitude vectors
- $spectrogram$: generate a spectrogram matrix based on short-time Fourier transform
- svd: perform singular value decomposition on matrix
- $knnsearch$: search for the first k closest points in training set to the test data
- $mode$: find the mode in a vector/matrix

Appendix B – MATLAB codes


```
clear; close all; clc;
```

Loading

```
% Names of music files
% Classical
D = dir('Classical_Bach*.mp3');
for j = 1:length(D)
    CL.Bach{j} = D(j).name;
end
D = dir('Classical_Beethoven*.mp3');
for j = 1:length(D)
    CL.Beethoven{j} = D(j).name;
end
D = dir('Classical_Mozart*.mp3');
for j = 1:length(D)
    CL.Mozart{j} = D(j).name;
end
CL.All = [CL.Bach CL.Beethoven CL.Mozart];
% Country
D = dir('Country_Chris_Haugen*.mp3');
for j = 1:length(D)
    CO.CH{j} = D(j).name;
end
D = dir('Country_Dan_Lebowitz*.mp3');
for j = 1:length(D)
    CO.DL{j} = D(j).name;
end
D = dir('Country_Nat_Keefe*.mp3');
for j = 1:length(D)
    CO.NK{j} = D(j).name;
end
CO.All = [CO.CH CO.DL CO.NK];
% Electronic
D = dir('Electronic_Spence*.mp3');
for j = 1:length(D)
    EL.S{j} = D(j).name;
end
D = dir('Electronic_Quincas_Moreira*.mp3');
```

```
for j = 1:length(D)
    EL.QM{j} = D(j).name;
end
D = dir('Electronic_Rondo_Brothers*.mp3');
for j = 1:length(D)
    EL.RB{j} = D(j).name;
end
EL.All = [EL.S EL.QM EL.RB];
```

Part I: Different Genre Band Classification

```
% 1 - Classical.Mozart, 2 - Country.Chris Haugen, 3 - Electronic.QM

% Data - all pieces from each Band
data1 = generateData(CL.Mozart);
data2 = generateData(CO.CH);
data3 = generateData(EL.QM);

% Choose test data
sampleSize = min([size(data1,1) size(data2,1) size(data3,1)]);
number = randperm(sampleSize);
testSize = sampleSize; % cross validate with all data
number = number(1:testSize);
successRate = zeros(testSize,1);

for jj = 1:testSize
    test_i = number(jj);
    % Test data
    test1 = data1(test_i,:);
    test2 = data2(test_i,:);
    test3 = data3(test_i,:);

    % Preparing training data (exclude test file)
    trainer1 = data1; trainer1(test_i,:) = [];
    trainer2 = data2; trainer2(test_i,:) = [];
    trainer3 = data3; trainer3(test_i,:) = [];

    % Classification
    % SVD and projection
    clear trainerScores
    clear testScores
    n = 3;
    [trainerScores,ind,U] = projSVD(trainer1,trainer2,trainer3,n);
    testScores(:,1) = U(:,1:n)'*test1';
    testScores(:,2) = U(:,1:n)'*test2';
    testScores(:,3) = U(:,1:n)'*test3';

    % Classification by knn
    Class = zeros(3,1);
    for j = 1:3
        Idx = knnsearch(trainerScores',testScores(:,j)','k',3);
        ClusterInd = ind(Idx);
        Class(j) = mode(ClusterInd);
```

```
    end
    successRate(jj) = sum(Class == [1;2;3])/3;
end
P1_OverallAccuracy = mean(successRate);
```

Scatter Plot

```
% figure(1)
% scatter3(trainerScores(1,1:8)',trainerScores(2,1:8)',trainerScores(3,1:8)','r');
% hold on;scatter3(trainerScores(1,9:16)',trainerScores(2,9:16)',trainerScores(3,9:16)','b');
% scatter3(trainerScores(1,17:24)',trainerScores(2,17:24)',trainerScores(3,17:24)','m')
% xlabel('Projection onto 1st PC')
% ylabel('Projection onto 2nd PC')
% zlabel('Projection onto 3rd PC')
% legend('Cluster 1','Cluster 2','Cluster 3')
```

Spectrogram

```
% figure(2)
% title('Mozart')
% for j = 1:9
% filename = CL.Mozart{j};[y,Fs] = audioread(filename);
% % check mono or stereo
% if size(y,2) == 2 % double channel
%     y = mean(y,2);
% end
% % truncate data
% y = y(55*Fs+1:60*Fs); % extract only 55-60 sec
% % resample
% y = reshape(y,[5 5*Fs/5]); % one data point every five
% y = mean(y,1);
% Fs = Fs/5; % update sampling frequency
% % spectrogram
% subplot(3,3,j)
% spectrogram(y,1000,[],[],Fs); drawnow
% end
% figure(3)
% title('Country CH')
% for j = 1:9
% filename = CO.CH{j};[y,Fs] = audioread(filename);
% % check mono or stereo
% if size(y,2) == 2 % double channel
%     y = mean(y,2);
% end
% % truncate data
% y = y(55*Fs+1:60*Fs); % extract only 55-60 sec
% % resample
% y = reshape(y,[5 5*Fs/5]); % one data point every five
% y = mean(y,1);
% Fs = Fs/5; % update sampling frequency
% % spectrogram
% subplot(3,3,j)
```

```
% spectrogram(y,1000,[],[],Fs); drawnow
% end
% figure(4)
% title('Electronic QM')
% for j = 1:9
% filename = EL.QM{j};[y,Fs] = audioread(filename);
% % check mono or stereo
% if size(y,2) == 2 % double channel
%     y = mean(y,2);
% end
% % truncate data
% y = y(55*Fs+1:60*Fs); % extract only 55-60 sec
% % resample
% y = reshape(y,[5 5*Fs/5]); % one data point every five
% y = mean(y,1);
% Fs = Fs/5; % update sampling frequency
% % spectrogram
% subplot(3,3,j)
% spectrogram(y,1000,[],[],Fs); drawnow
% end
```

Part II: Same Genre Band Classification

```
% 1 - Classical.Mozart, 2 - Country.Chris Haugen, 3 - Electronic.QM

% Data - all pieces from each Band
data1 = generateData(CO.DL);
data2 = generateData(CO.CH);
data3 = generateData(CO.NK);

% Choose test data
sampleSize = min([size(data1,1) size(data2,1) size(data3,1)]);
number = randperm(sampleSize);
testSize = sampleSize; % cross validate with all data
number = number(1:testSize);
successRate = zeros(testSize,1);

for jj = 1:testSize
    test_i = number(jj);
    % Test data
    test1 = data1(test_i,:);
    test2 = data2(test_i,:);
    test3 = data3(test_i,:);

    % Preparing training data (exclude test file)
    trainer1 = data1; trainer1(test_i,:) = [];
    trainer2 = data2; trainer2(test_i,:) = [];
    trainer3 = data3; trainer3(test_i,:) = [];

    % Classification
    % SVD and projection
    clear trainerScores
    clear testScores
```

```
    n = 3;
    [trainerScores,ind,U] = projSVD(trainer1,trainer2,trainer3,n);
    testScores(:,1) = U(:,1:n)'*test1';
    testScores(:,2) = U(:,1:n)'*test2';
    testScores(:,3) = U(:,1:n)'*test3';

    % Classification by knn
    Class = zeros(3,1);
    for j = 1:3
        Idx = knnsearch(trainerScores',testScores(:,j)','k',3);
        ClusterInd = ind(Idx);
        Class(j) = mode(ClusterInd);
    end
    successRate(jj) = sum(Class == [1;2;3])/3;
end
P2_OverallAccuracy = mean(successRate);
```

Part III: Same Genre Band Classification

```
% 1 - Classical.Mozart, 2 - Country.Chris Haugen, 3 - Electronic.QM

% Data - all pieces from each Band
data1 = generateData(CL.All);
data2 = generateData(CO.All);
data3 = generateData(EL.All);

% Choose test data
sampleSize = min([size(data1,1) size(data2,1) size(data3,1)]);
number = randperm(sampleSize);
testSize = sampleSize; % cross validate with all data
number = number(1:testSize);
successRate = zeros(testSize,1);

for jj = 1:testSize
    test_i = number(jj);
    % Test data
    test1 = data1(test_i,:);
    test2 = data2(test_i,:);
    test3 = data3(test_i,:);

    % Preparing training data (exclude test file)
    trainer1 = data1; trainer1(test_i,:) = [];
    trainer2 = data2; trainer2(test_i,:) = [];
    trainer3 = data3; trainer3(test_i,:) = [];

    % Classification
    % SVD and projection
    clear trainerScores
    clear testScores
    n = 3;
    [trainerScores,ind,U] = projSVD(trainer1,trainer2,trainer3,n);
    testScores(:,1) = U(:,1:n)'*test1';
    testScores(:,2) = U(:,1:n)'*test2';
```

```
    testScores(:,3) = U(:,1:n)'*test3';

    % Classification by knn
    Class = zeros(3,1);
    for j = 1:3
        Idx = knnsearch(trainerScores',testScores(:,j)','k',3);
        ClusterInd = ind(Idx);
        Class(j) = mode(ClusterInd);
    end
    successRate(jj) = sum(Class == [1;2;3])/3;
end
P3_OverallAccuracy = mean(successRate);
```

Functions

```
function output = convSpec(filename)
% Input: filename - string
% Output: spectrogram of resampled audio data - horizontal vector

% load file
[y,Fs] = audioread(filename);
% check mono or stereo
if size(y,2) == 2 % double channel
    y = mean(y,2);
end
% truncate data
y = y(55*Fs+1:60*Fs); % extract only 55-60 sec
% resample
y = reshape(y,[5 5*Fs/5]); % one data point every five
y = mean(y,1);
Fs = Fs/5; % update sampling frequency
% spectrogram
spec = spectrogram(y,1000,[],[],Fs);
% flatten and absolute value
output = abs(reshape(spec,[size(spec,1)*size(spec,2) 1])');
end

function [scores,ind,U] = projSVD(A,B,C,n)
% Input: training sets A B C - horizontal vector/matrix
%        n - number of features to use
% Output: scores - match (dot product) of each feature n by m
%         ind - index of the feature scores
%         U - for projecting additional data points

trainer = [A;B;C]';
[U,~,~] = svd(trainer,'econ');
scores = U(:,1:n)'*trainer;
ind = [1*ones(1,size(A,1)) 2*ones(1,size(B,1)) 3*ones(1,size(C,1))];
end

function output = generateData(cellname)
% Stack entries in specified cell to form a combined data matrix
trial = convSpec(cellname{1});
```

```
output = zeros(length(cellname),size(trial,2));
output(1,:) = trial;
for j = 2:length(cellname)
    output(j,:) = convSpec(cellname{j});
end
end
```

Published with MATLAB® R2019b

Appendix C – List of all music used

'Classical_Bach_C_Major_Prelude.mp3'
'Classical_Bach_G_Major_Prelude.mp3'
'Classical_Bach_Passion.mp3'
'Classical_Bach_Toccata_in_D_minor.mp3'
'Classical_Beethoven_9th_Symphony_Finale.mp3'
'Classical_Beethoven_Fur_Elise.mp3'
'Classical_Beethoven_Moonlight_Sonata.mp3'
'Classical_Beethoven_Symphony_No_5.mp3'
'Classical_Mozart_Adagio.mp3'
'Classical_Mozart_Adagio_Violin.mp3'
'Classical_Mozart_Allegro.mp3'
'Classical_Mozart_Allegro_Moderato.mp3'
'Classical_Mozart_Andantino.mp3'
'Classical_Mozart_Divertimento.mp3'
'Classical_Mozart_Eine_Kleine_Nachtmusik.mp3'
'Classical_Mozart_Hunt.mp3'
'Classical_Mozart_Rondo.mp3'
'Country_Chris_Haugen_Campfire_Song.mp3'
'Country_Chris_Haugen_Castleshire.mp3'
'Country_Chris_Haugen_Firefly.mp3'
'Country_Chris_Haugen_Ibiza_Dream.mp3'
'Country_Chris_Haugen_Mechanical_Bullride.mp3'
'Country_Chris_Haugen_Morning_Mandolin.mp3'
'Country_Chris_Haugen_Si_Senorita.mp3'
'Country_Chris_Haugen_Temptation.mp3'
'Country_Chris_Haugen_Tupelo_Train.mp3'
'Country_Dan_Lebowitz_5_O_July.mp3'
'Country_Dan_Lebowitz_Cliffsides.mp3'
'Country_Dan_Lebowitz_Hiiltop.mp3'
'Country_Dan_Lebowitz_Jeremiah_s_Song.mp3'
'Country_Dan_Lebowitz_Last_Train_to_Mars.mp3'
'Country_Dan_Lebowitz_Merry_Go_Round.mp3'
'Country_Dan_Lebowitz_Michigan_Greens.mp3'
'Country_Dan_Lebowitz_Parkside.mp3'
'Country_Dan_Lebowitz_Run_One_Down.mp3'
'Country_Nat_Keefe_Angeline_the_Baker.mp3'
'Country_Nat_Keefe_Arkansas_Traveler.mp3'
'Country_Nat_Keefe_Big_Sciota.mp3'
'Country_Nat_Keefe_Billy_in_the_Lowground.mp3'
'Country_Nat_Keefe_Cumberland_Gap.mp3'
'Country_Nat_Keefe_Daley_s_Reel.mp3'
'Country_Nat_Keefe_Dude_Where_s_My_Horse.mp3'
'Country_Nat_Keefe_Lost_Native.mp3'

'Country_Nat_Keefe_Sally_Goodin.mp3'
'Electronic_Spence_Cloud_Chaser.mp3'
'Electronic_Spence_Retrograde.mp3'
'Electronic_Spence_Right_Here_Beside_You.mp3'
'Electronic_Spence_Unstoppable.mp3'
'Electronic_Quincas_Moreira_Blue_Macaw.mp3'
'Electronic_Quincas_Moreira_Dragonfly.mp3'
'Electronic_Quincas_Moreira_Firefly.mp3'
'Electronic_Quincas_Moreira_Ladybug.mp3'
'Electronic_Quincas_Moreira_Mantis.mp3'
'Electronic_Quincas_Moreira_Moskito.mp3'
'Electronic_Quincas_Moreira_Red_Ant.mp3'
'Electronic_Quincas_Moreira_Scarab.mp3'
'Electronic_Quincas_Moreira_Silkworm.mp3'
'Electronic_Rondo_Brothers_Dolphin.mp3'
'Electronic_Rondo_Brothers_Magnum.mp3'
'Electronic_Rondo_Brothers_Mr_Tea.mp3'
'Electronic_Rondo_Brothers_Trophy_Wife.mp3'