Homework 4: Music Classification

AMATH 482: Computational Methods for Data Analysis

Rosemichelle Marzan

March 6, 2020

Abstract **– Principal component analysis (PCA) and linear discriminant analysis (LDA) were used to create a machine learning algorithm for classifying 5-second audio clips. Three different classifiers were used to classify songs from three artists of different genres, three artists from the same genre, and many artists from three genres. The classifiers were evaluated on their success rate. This report discussing the effects of training data variation and the number of PCA modes ("features") on classifier performance.**

1 Introduction and Overview

In this study, principal component analysis and linear discriminant analysis were used to classify sets of 5-second audio clips. Three experiments were performed: the first was to classify songs from three artists of different genres; the second was to classify songs from three artists within the same genre; and the third was to classify songs to three genres using a multitude of artists. Classifiers were built based on training sets for each experiment and then evaluated on their accuracy (success rate).

2 Theoretical Background

Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are commonly used dimensionality reduction techniques for machine learning applications. The key difference between them is that whereas PCA maximizes the separation (variance) between data points, LDA on the other hand seeks to maximize the separation of classes within the data set. As a result, PCA is often referred to as a "unsupervised" algorithm in that it ignores class labels, while LDA is considered a "supervised" algorithm because it takes labels into account. Recall that the goal of PCA is to find the principal component axes which maximize the variance in a data set. The goal of LDA is to maximize separation between classes, and to minimize separation within a class, represented as a projection $\mathbf{w}$.

If the goal is to classify songs, why is PCA necessary? The motivation for using both methods is that PCA can be thought of as enhancing the distinguishing features that describe a class, or amplifying a signal while attenuating the noise. In theory, LDA could be performed directly on the raw data but the classifier's success rate would be poorer because of the noise.

The LDA projection, $\mathbf{w}$, is the solution to an eigenvalue problem (Eq. 1):

$$\mathbf{S_B w} = \lambda \mathbf{S_W w} \tag{1}$$

where S_B and S_W are two scatter matrices: between-class (Eq. 2) and within-class (Eq. 3), respectively. The scatter matrices measure the variance in the data sets and the variance in the means:

$$\mathbf{S_B} = \sum_{\mathbf{j=1}}^{\mathbf{n}} \mathbf{m_j}(\vec{\boldsymbol{\mu_j}} - \vec{\boldsymbol{\mu}})(\vec{\boldsymbol{\mu_j}} - \vec{\boldsymbol{\mu}})^{\mathbf{T}} \tag{2}$$

$$\mathbf{S_W} = \sum_{\mathbf{j=1}}^{\mathbf{n}} \sum_{\vec{\boldsymbol{x}}} (\vec{\boldsymbol{x}} - \overrightarrow{\boldsymbol{\mu_j}})(\vec{\boldsymbol{x}} - \overrightarrow{\boldsymbol{\mu_j}})^{\mathbf{T}} \tag{3}$$

m_j is the number of samples from class j and $\vec{\boldsymbol{\mu}}$ is the mean of all the data.

In this study, PCA and LDA are performed on the spectrograms of the audio clips which contain key frequency and time information. Singular value decomposition (aka PCA) determines the dominant spectrogram modes associated with a song and artist which can be used to build the set of criteria for defining a class.

3 Algorithm Implementation and Development

3.1 Data Acquisition

Each experiment used a training data set with 30 audio clips (each clip is a different song) for each class, for a total of 90 training clips.

- Experiment 1: 3 artists, 3 genres (Electropop: MØ, Hip Hop: Tech N9ne, Rock: Fall Out Boy)
- Experiment 2: 3 artists, 1 genre (Rock: Five Finger Death Punch, the GazettE, Muse)
- Experiment 3: Many artists, 3 genres (Electronic Dance Music (EDM), Hip Hop, Alternative Rock)

Each experiment's testing set is made of 10 audio clips belonging to each class, for a total of 30 testing clips.

Training and testing songs were recorded via loopback from Spotify and into Audacity. Each recording begins at a random point around the 1:00 to 2:00 minute mark of each song. The recording is then saved as a MP3 file as a single (mono) channel at a sampling frequency of 44.1 kHz. The audio clips for each experiment's training set were loaded into MATLAB. The data points for each song make up a single column in the `train` matrix. The `train` matrix is then truncated to take only every 4th point in the original sample, cutting the matrix down to 55,125 rows and 90 columns, which helps to reduce processing time and memory use.

3.2 The Trainer

3.2.1 Creating the Spectrograms

The spectrograms were made by multiplying a sliding Gaussian filter window, `tslide` by the signal, then applying the fast Fourier transform to get the signal in the frequency domain. The window parameters, `a` and `dtau`, were set to 1000 and 0.05, respectively based on the (visual) quality of the first spectrogram (decent resolution in frequency and in time). In a previous study, `a` = 1000 was found to be the ideal parameter for analyzing the spectrogram of Handel's *Messiah* and so was accepted as the default parameter for all spectograms. Choosing a larger `dtau` would result in poorer time resolution, while a smaller `dtau` is computationally expensive, so the selected `dtau` was deemed appropriate for all spectrograms. The spectrogram for each song is in matrix form. This is vectorized so that each spectogram forms a column in the `spec` matrix.

3.2.2 Singular Value Decomposition

The reduced SVD is performed on the matrix of spectrograms. Multiplying the Σ (`S`) matrix by `V` form the projections which show the strength of the most dominant POD modes on each class. The `feature` variable dictates how many of the most dominant POD modes to consider, and is a parameter that is tuned to give the highest classifier success rate.

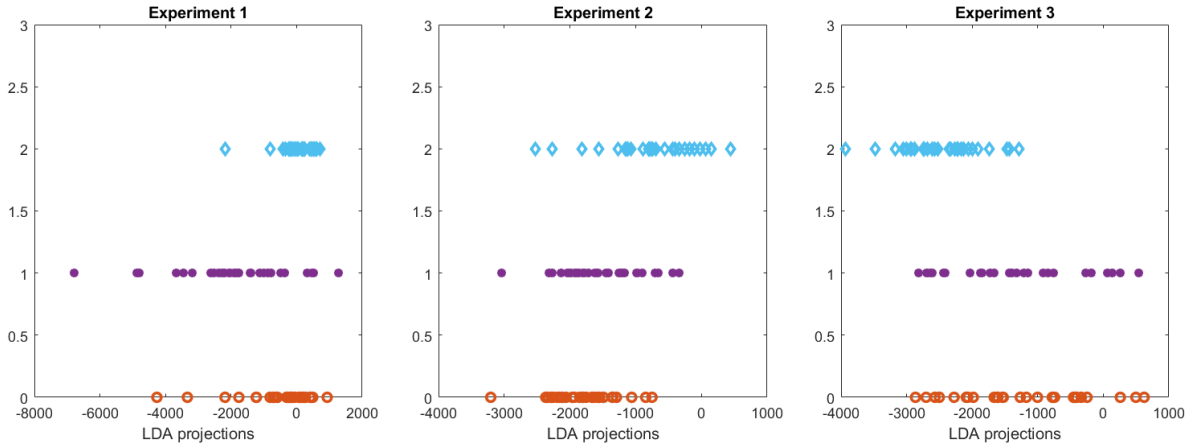

Figure 1: Plots of the LDA-projected classes for all experiments. Each point represents an audio clip. Orange markers = electropop (exp. 1), Five Finger Death Punch (exp. 2), EDM (exp. 3). Purple markers = hip hop (exp. 1 & 3), the Gazette (exp. 2). Light blue markers = Rock (exp. 1 3), Muse (exp. 2). MATLAB legend produced incorrect labels and was thus omitted. PCA features = 5 (exp. 1 & 3) and 10 (exp. 2). Points of each class are floated to best show overlap.

3.2.3 Linear Discriminant Analysis

The `eig` function solves the eigenvalue problem for the $\mathbf{w}$ projection by taking in the S_B and S_W scatter matrices. The `w` projection is the eigenvector associated with the maximum eigenvalue, normalized by the 2-norm. This eigenvector represents the most dominant LDA basis. Multiplying `w` by each class matrix, whose dimensions are (# of features $\times$ # of songs in the class) shows how each LDA-projected class can be separated along one dimension.

3.2.4 Determining the Thresholds

The thresholds to distinguish one class from another can be determined by plotting each LDA-projected class on a single one-dimensional plot as seen in Fig. 1. Test points were categorized by measuring the distance of its LDA-projection from each class mean. The test point would thus be assigned to the class where the distance is the shortest. The thresholds are essentially the midpoints between the means of each class.

3.3 The Classifier

In a separate script, each experimental test set was created by first importing all 30 test songs in MATLAB. Due to the file naming convention, the first 10 columns of the test song matrix belonged to the first class, the next 10 to the second class, and the final 10 to the third class. This non-random order should not have an effect on the classifier performance since it does not affect the training matrix, training SVD matrices, and training LDA projection. In the same script, each song was subsampled and then saved as a `.mat` file.

Similar to the trainer algorithm, the classifier takes in the `.mat` file and creates a spectrogram for each song, vectorized then saved as a column in the `spec` matrix. The PCA projection is taken by multiplying the principal components, `U`, by each spectrogram. Then, the LDA projection is taken by multiplying the `w` projection from earlier to the PCA projection. Each LDA-projected test point's distance from each of the class means were calculated and the minimum distance was determined by the classifier to be the class the test point belongs to.

	EXPERIMENT 1				EXPERIMENT 2				EXPERIMENT 3			
Features	Electro-pop	Hip Hop	Rock	Overall Accuracy Rate	Finger Death Punch	the GazettE	Muse	Overall Accuracy Rate	EDM	Hip Hop	Alt. Rock	Overall Accuracy Rate
5	90%	60%	80%	76.67%	90%	30%	60%	60.00%	20%	60%	100%	60.00%
10	80%	60%	80%	73.33%	90%	50%	70%	70.00%	20%	60%	100%	60.00%
15	20%	60%	40%	40.00%	40%	70%	50%	53.33%	30%	60%	40%	43.33%
20	20%	60%	30%	36.67%	40%	50%	50%	46.67%	20%	60%	90%	56.67%
25	40%	50%	80%	56.67%	40%	60%	50%	50.00%	30%	50%	70%	50.00%
30	50%	50%	90%	63.33%	70%	70%	10%	50.00%	30%	50%	60%	46.67%
35	40%	40%	80%	53.33%	90%	70%	20%	60.00%	50%	30%	80%	53.33%
40	40%	40%	80%	53.33%	90%	60%	20%	56.67%	20%	50%	80%	50.00%
45	30%	40%	80%	50.00%	90%	60%	10%	53.33%	20%	40%	80%	46.67%
50	20%	40%	80%	46.67%	90%	60%	10%	53.33%	20%	30%	80%	43.33%
55	30%	50%	90%	56.67%	90%	10%	30%	43.33%	20%	40%	90%	50.00%
60	30%	50%	90%	56.67%	90%	20%	20%	43.33%	0%	40%	90%	43.33%
65	40%	30%	90%	53.33%	90%	20%	20%	43.33%	0%	40%	80%	40.00%
70	40%	30%	90%	53.33%	90%	20%	20%	43.33%	0%	30%	80%	36.67%
75	40%	50%	90%	60.00%	90%	50%	10%	50.00%	0%	40%	80%	40.00%
80	40%	50%	100%	63.33%	20%	50%	10%	26.67%	50%	40%	0%	30.00%
85	40%	60%	100%	66.67%	20%	70%	10%	33.33%	0%	40%	80%	40.00%
90	0%	60%	20%	26.67%	0%	60%	30%	30.00%	0%	30%	10%	13.33%

Figure 2: Classifier accuracy rates for each experiment. Rates were also calculated for each class within an experiment. The overall accuracy rates are highlighted in yellow, with the highest rates highlighted in golden yellow.

3.3.1 Determining the number of features

As mentioned earlier, the number of PCA features is a parameter that can be tuned to see which produced the highest classifier success rate. The classifier was performed using `feature` $= 5$ all the way to 90 in increments of 5. The success rate was then calculated for the entire test set, as well as for each class within a test set (see Fig. 2).

4 Computational Results

The classifiers that performed the best had an success rate of **76.67% for experiment 1** (3 artists from 3 genres), **70% for experiment 2** (3 artists from the same genre), and **60% for experiment 3** (many artists from 3 genres). As seen in Fig. 2, these classifiers used either the first 5 or the first 10 modes of the PCA (# of features).

Considering how these classifiers performed to one another between experiments, these results are not surprising. The second and third experiments emulate how too little or too much diversity can affect success rates. A musical genre can be categorized by the kinds of instruments used (whose frequency signatures are able to be detected from the spectrogram), tempo, volume, to name a few parameters. These classifications do not necessarily have to be exact either, and is also largely subjective. If I were to classify these clips myself, I would also struggle unless the clip included vocals. The success rate between classes in experiment 2 showed that Five Finger Death Punch songs were consistently classified most correctly (mean success rate = 67.78%), which I personally agree with since they have a more "grunge-y"/"heavy metal" type of music compared to the other two artists. Their musical style is also much more consistent across all of their songs compared to the other two artists.

On the other hand, too much diversity can also contribute to poor performance, and in this case is worse than too little. This is because within-class variation can make it difficult to identify the criteria that defines a class. Fig. 1 shows that experiment 3 has the highest amount of within-class variance, experiment 2 has the lowest, and experiment 1 is the perfect balance between both.

The choice of genres for experiment 3 may also have an effect on the classifier's performance. The lowest

class mean success rate within experiment 3 was 18.33% for EDM. This could be because EDM as genre is quite diverse; songs can contain elements from a multitude of other genres such as pop and rap/hip hop, but EDM is typically defined by periods within the song which use more electronic sounds and beats for dancing. Most of the clips used in both the training and test sets were not recorded during these periods in the original song. It is also worth noting that for this experiment, the songs used for the EDM class were obtained from a Spotify user-curated playlist called "EDM Hits 2020", songs for the hip hop class were from a Spotify-curated (which could have been from a computer algorithm) "radio playlist" for artists and songs similar to Tech N9ne, and songs for the alternative rock class were from another Spotify-curated playlist called "Essential Alternative". The difference in success rates obtained for each class may also be an artifact of how the songs were selected (ie. computer versus user).

It is clear that the number of PCA features used in the classifier has a major impact on the success rate. As seen in Fig. 3, there appear to be approximately three major principal components that describe all 90 audio clips within a set. This makes sense because in the case of experiments 1 and 3, each class had three different genres. The fact that the only one singular value dominates experiment 2 indicates that the PCA algorithm recognizes that all three classes are within the same genre. The classifier for experiments 1 and 3 only needed 5 features to give the highest success rate. The experiment 2 classifier needed a few more (10 features) to best classify the test songs because the songs were more similar in this experiment than in the others.

The plots of the LDA-projected classes demonstrate a high degree of overlap because only a few features were used (Fig. 1). Using all 90 features showed no class overlap, yet produced the worst performing classifiers because of what is known as overfitting, wherein the classifier best correctly classifies the training data, but the lack of flexibility causes it to perform poorly when asked to classify new data. It is interesting that despite this high degree of overlap, the mean-thresholding algorithm classifies the best. Perhaps if a different thresholding scheme were used, such as that increases the separation between the classes with some overlap and setting the threshold at the center of the overlap, a greater number of features would be required to achieve a higher accuracy rate.

5 Summary and Conclusions

In this report, we explored why and how PCA and LDA are the dimensionality reduction techniques used in machine learning classification. Although similar, both have strengths that are useful in classification. Through SVD, PCA enhances the raw data by identifying the principal components that best separate the data into general groups. From there, LDA is able to maximize the separation between those groups as well

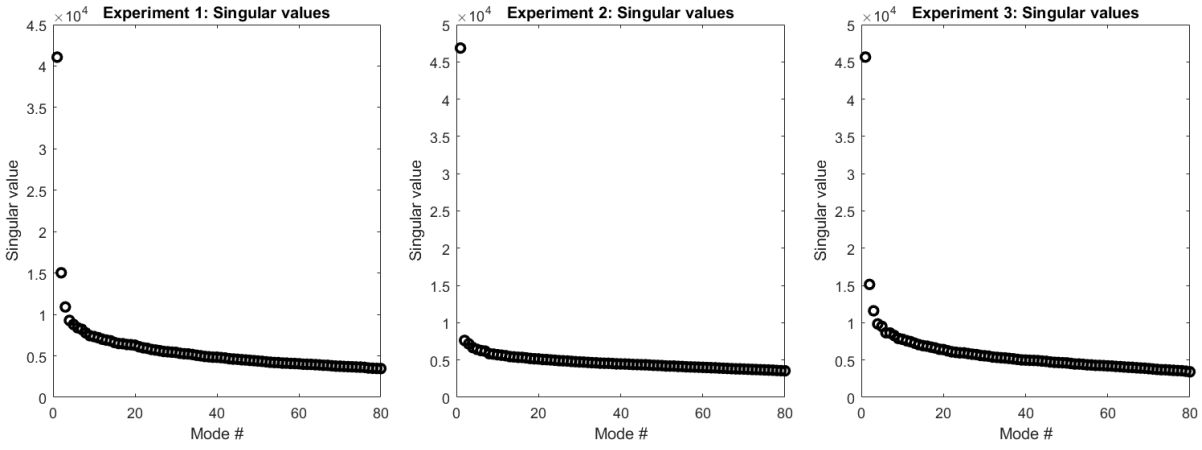

Figure 3: Plots of the first 80 singular values for each experiment.

as minimize the separation between data points within a group.

The algorithm for developing the trainer can be summarized into four key steps: (1) perform a wavelet transform (spectrogram) onto each audio clip in the training set; (2) perform a singular value decomposition (computational method for PCA) on the matrix containing all the spectrograms; (3) perform linear discriminant analysis; and (4) determine a statistical threshold for classification. The classifier is structured similarly. Test data are classified according to a thresholding scheme that finds the minimum distance between the data point and one of the class means, and assigns it to that class.

The experiments performed demonstrate the effect of varying degrees of diversity in the data used to train the classifier. When there is little variety in the spectrogram quality of the audio clips, using more features, or PCA modes, allows the algorithm to identify more, albeit subtle, differences between songs. Features thus function similar to the idea of classification "criteria". On the other end of the spectrum, too much variety, or greater variation within a class, broadens the criteria for defining a class.

6 Appendix A. MATLAB functions used and brief implementation explanation

- **fft**: Fast Fourier transform for converting time-domain signal of each audio clip into the frequency domain for spectrogram creation.
- **svd**: Decomposes the given matrix to three matrices: a unitary matrix of left singular vectors (principal components), a diagonal matrix of singular values, and a unitary matrix of right singular vectors
- **eig**: Returns the eigenvector and eigenvalue solutions to $\mathbf{S_B w} = \lambda \mathbf{S_W w}$.
- **diag**: Extracts the values along the diagonal of a square matrix.
- **norm**: Returns the 2-norm of the given matrix.

7 Appendix B. MATLAB code

```
clear all; close all; clc;

%%% Name: ROSEMICHELLE MARZAN
%%% Course: AMATH 482
%%% Homework 4, Due 3/6/2019

%% Training

files = dir('*.mp3');
numSongs = 90;
% train rows: arbitrary #; all clips have slightly varying # of data points
% depending on recording duration
train = zeros(230000,numSongs);
for k = 1:numSongs
    A = audioread(files(k).name);
    train(1:length(A),k) = A;
end

Fs = 44100;      % sampling rate
L = 5;           % want 5 s songs
train = train(1:(Fs*L),:);    % truncate training data
train = train(1:4:end,:); % take every fourth point

% Create spectrograms
n = length(train3(:,1));           % # of data points/clip
t = linspace(0,L,n);               % vector of time points
a = 1000; dtau = 0.05;             % sliding filter parameters
tslide = 0:dtau:L;
spec = zeros(length(tslide),n,numSongs);    % matrix of spectrograms
for i = 1:numSongs
    for j = 1:length(tslide)
        g = exp(-a*(t - tslide(j)).^2);
        Sg = g.*(train3(:,i).');
        Sgt = fft(Sg);
        spec(j,:,i) = fftshift(abs(Sgt));
    end
end

% separate matrix by classes
% 5567625 = # of data points * tslide length
class1spec = reshape(spec(:,:,1:30), [5567625,30]);
class2spec = reshape(spec(:,:,31:60), [5567625,30]);
class3spec = reshape(spec(:,:,61:90), [5567625,30]);

feature = 5;
[U,S,V,w,v1,v2,v3] = trainer(class1spec,class2spec,class3spec,feature)

%% Classification

clc;
% load test songs (.mat file)
load exp1testsongs.mat
numTestSongs = 30;

% get spectrogram
a = 1000; dtau = 0.05;
tslide = 0:dtau:L;
test_spec = zeros(length(tslide),n,numTestSongs);
for i = 1:numTestSongs
    for j = 1:length(tslide)
        g = exp(-a*(t - tslide(j)).^2);
        Sg = g.*(test1(:,i).');
        Sgt = fft(Sg);
```

```
        test_spec(j,:,i) = fftshift(abs(Sgt));
    end
end
test_spec = reshape(test_spec(:,:,:), [5567625,numTestSongs]);

% PCA projection
test_pca = zeros(90,30);
for i = 1:30
    test_pca(:,i) = U'*test_spec(:,i);
end

% LDA projection
test_lda = zeros(1,30);
for i = 1:30
    test_lda(1,i) = w'*test_pca(1:feature,i);
end

% mean of projected training points
m1_proj = mean(v1);
m2_proj = mean(v2);
m3_proj = mean(v3);

% Mean-based thresholding/classification
for i = 1:30
    distances = [abs(m1_proj - test_lda(i));
                 abs(m2_proj - test_lda(i));
                 abs(m3_proj - test_lda(i))];
    [~,index] = min(distances);

    % CHANGE LABELS
    if index == 1
        disp('Class 1')
    elseif index == 2
        disp('Class 2')
    else
        disp('Class 3')
    end
end


function [U,S,V,w,v1,v2,v3] = trainer(class1spec,class2spec,class3spec,feature)
    n1 = size(class1spec,2);
    n2 = size(class2spec,2);
    n3 = size(class3spec,2);

    [U,S,V] = svd([class1spec, class2spec, class3spec],'econ');

    allsongs = S*V'; % projection onto principal components
    class1 = allsongs(1:feature,1:n1);
    class2 = allsongs(1:feature,n1+1:n1+n2);
    class3 = allsongs(1:feature,n1+n2+1:n1+n2+n3);

    m1 = mean(class1,2);
    m2 = mean(class2,2);
    m3 = mean(class3,2);
    mu = mean([class1,class2,class3],2);

    % within-class scatter matrix
    Sw = 0;
    for k=1:30
        Sw = Sw + (class1(:,k)-m1)*(class1(:,k)-m1)';
    end
    for k=1:30
        Sw = Sw + (class2(:,k)-m2)*(class2(:,k)-m2)';
    end
    for k=1:30
```

```
        Sw = Sw + (class3(:,k)-m3)*(class3(:,k)-m3)';
    end

    % between-class scatter matrix
    Sb = ((m1-mu)*(m1-mu)' + (m2-mu)*(m2-mu)' + (m3-mu)*(m3-mu)');

    % linear discriminant analysis: maximize Sb, minimize Sw
    [V1,D1] = eig(Sb,Sw);
    [~,ind] = max(abs(diag(D1))); % w = eigenvec of largest eigenval
    w = V1(:,ind); w = w/norm(w,2);

    % LDA-projected class matrices
    v1 = w'*class1;
    v2 = w'*class2;
    v3 = w'*class3;
end
```