Homework 4: Applications of Supervised Learning for Music Classification

Eli Miller

March 8, 2019

Abstract

In this analysis, we will first explor PCA representation of the Yale Faces Database (Eigenfaces). We will look at the differences that arise between cropped and uncropped datasets, and discuss low-rank projections of our data. We will then leverage this style of image processing routine to classify music. Small segments' spectrograms will be treatued as images. Again, we use SVD and PCA to create low-rank representations of our data. We then explore Supervised Machine Learning with the application of classifying a music sample's genre or artist.

1 Introduction and Overview

Machine Learning Algorithms are the hot topics of modern data science. We have explored using SVD and other tools to represent data in low-rank structures. We continue to expand on this idea through this analysis, and leverage this way of thinking for application with Machine Learning (ML) algorithms and techniques. First, we will explore using the Yale Faces database as an illustration of the differences in SVD outcomes with different levels of data cohesiveness. We then take these ideas of using the SVD on different types of data and use them for music classification. We are interested in representing data efficiently and effectively for ML implementation. As such, we leverage previous analyses and create spectrograms of audio segments. Now, treating these spectrograms as "faces" of different songs and genres, we are able to use our implementation of the Yale Faces to extend our background from facial to audio recognition. We then take the analysis one step further, training a model that we can use for both genre and artist classification. Discussion of specific implementation and analysis of the accuracy of our model are explored.

2 Theoretical Background

The Singular Value Decomposition aims to diagonal a data matrix A into orthogonal principal component directions. This takes the form of

$$A_{m,n} = U\Sigma V^* \tag{1}$$

where $U_{n,m}$ is the matrix of principal directions, $\Sigma_{m,m}$ is a diagonal matrix of principal components, and $V^*_{n,m}$ is a matrix of projections of principal components.

One thing important to note is that the SVD expects that our data has a mean zero. As a result, we need to subtract off the average of each row of A before preforming SVD analysis.

To preform a rank-r reconstruction of the data, we simply multiply out our decomposition with the first r components. Specifically:

$$A_{reconstructed} = U_{n,r}\Sigma_{r,r}V_{r,n} \tag{2}$$

To leverage this idea of low rank representation, we can look at the physical significance of each of the U, Σ, and V matrices.

- U: Collection of basis vectors for the Principal Component space
- Σ: Diagonal matrix of singular values in order of descending magnitude.
- V: Projections of our data into the Principal Component space

And if we want to obtain a low-rank representation X of our data in terms of the first r principal components, we can compute (using a pythonic index notation)

$$X = A * V^T(:, 0 : r) \tag{3}$$

The motivation for all of this is to leverage what we have learned in image and audio processing to create training and test data for machine learning algorithms. Specifically we will look into Linear Support Vector Machine and Gaussian Naive-Bayes as algorithms for classifying data. An introduction to these (and other) data-mining algorithms can be found in *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*.

3 A Building Block for ML: EigenFaces

Let's look at using PCA for image processing with the Yale Faces Database. We will explore both cropped and cetnered images, as well as uncropped images. We can load our images into a data matrix A where each row is the flattened array of our image data. Again, we take care to center the rows of A with mean zero for use in the SVD. Assuming that this occurs for each column we have:

$$A_{centered} = A - \texttt{mean}(A) \tag{4}$$

We can now take the SVD of our data, and plot the Singular Value Spectra. This is pictured in Figure 1. We also include cumulative sums of the relative energy of our spectra. This is useful to visualize at what rank r our representation contains a desired percentage of our original signal's energy.

We notice that the cropped signal's Singular Value spectrum decays much quicker than the uncropped. This makes sense intuitively; because the faces in the cropped case are centered, the SVD

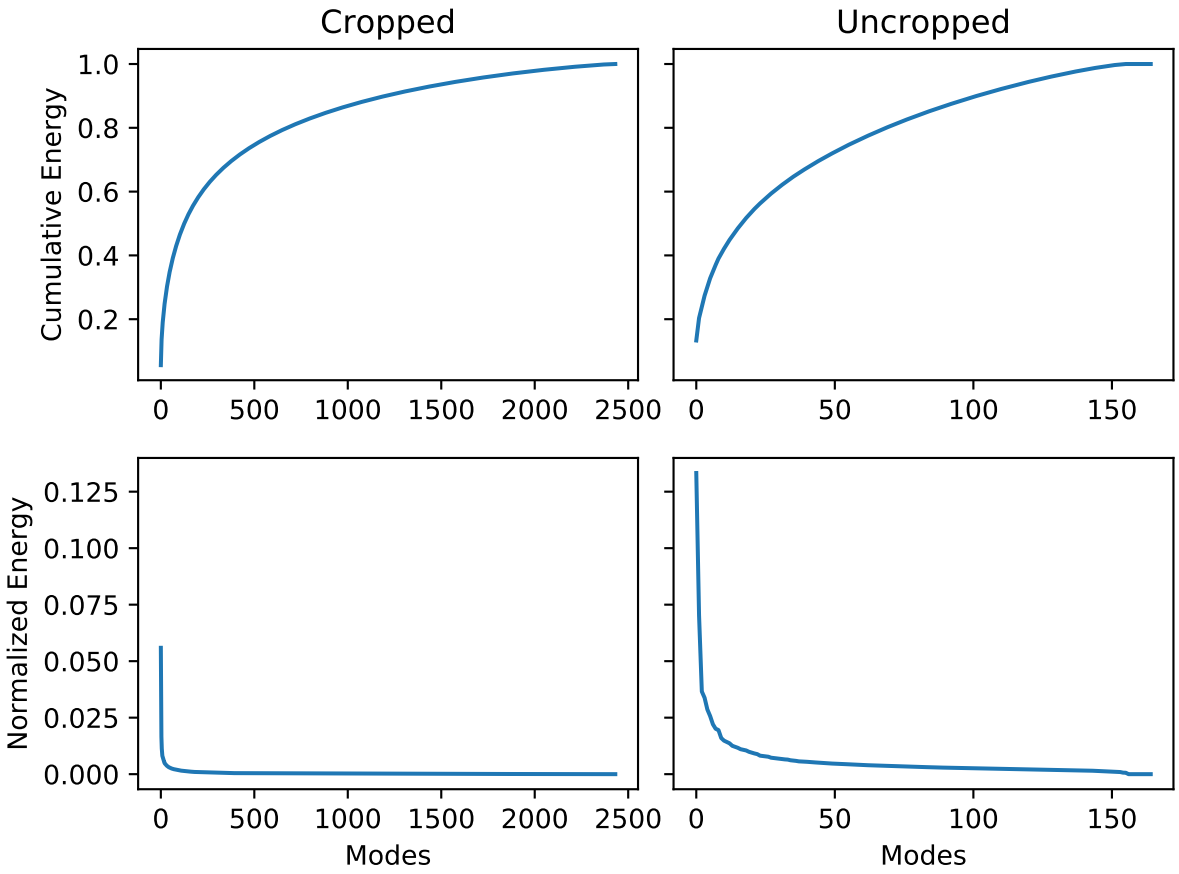

Figure 1: Summary of Yale Faces Singular Values

is simply picking out differences between each of the facial structures, whereas for the uncropped images, the SVD is getting a noisier signal. In the uncropped case, the SVD is having to reconcile differences in not only facial structure and lighting but also position. This explains the slower decay of the values of our Σ matrix.

It is important to note that the first mode of our uncropped test appears to contain much less energy than its cropped counterpart. The reason that this is occurring is because we are normalizing each case's singular values by their corresponding sum. We have many more samples for the cropped case. While most of these values are close to zero, when added together, these less significan singular values do accumulate to our overall sum.

If we were going to use this analysis to construct a low-rank feature representation of our data for interfacing with Machine Learning algorithms, we can look at Figure 1 to guide our choice of a rank r. Much like the way that we stop training on neural nets when the loss function stops decreasing rapidly (i.e. at the 'elbow' of our graph) we can get a good value of r for low-rank representation. For the example of the Yale Faces, we could use a rank of $r = 50$ for a representation that is reasonably accurate (assuming gaussian distribution, this puts us at above one standard deviation of the signal's content) and significantly compressed for use in combination with ML algorithms.

These ideas of low-rank representation as a precursor to Machine Learning will be explored and applied in the Music Recognition Section of this analysis.

4 Music Classification Implementation and Development

4.1 Data Preperation

Our data used for music recognition analysis will look at three genres: Jazz, Rock, and EDM. Within each of these genres we choose to use three seperate artists. The data for each of these artists is constructed by collecting randomized 5-second clips from one album for each artist. A summary of groups is included in Table 1.

While this dataset is of limited diversity, these concepts and algorithms can easily scale to larger and broader datasets. Even with only 9 albums, we were able to generate over 3000 unique 5-second segments for use in training, test, and validation sets.

The next step is to transform each of these 5-second audio clips into spectrograms. This will create a set of visual audio signatures for use in training our model. With spectrogram data, we can treat each spectrogram as a sort of "face" corresponding to each of our genres and artists. With tools

Jazz	**Rock**	**EDM**
Count Basie *Chairman of the Board*	Aerosmith *Greatest Hits*	Odesza *A Moment Apart*
Miles Davis *Kind of Blue*	Def Leppard *Adrenalize*	Flume *Flume*
Sidney Bechet *Paris 1952*	Mötley Crüe *Dr. Feelgood*	Louis the Child *Kids at Play EP*

Table 1: Summary of artists used for training and test sets.

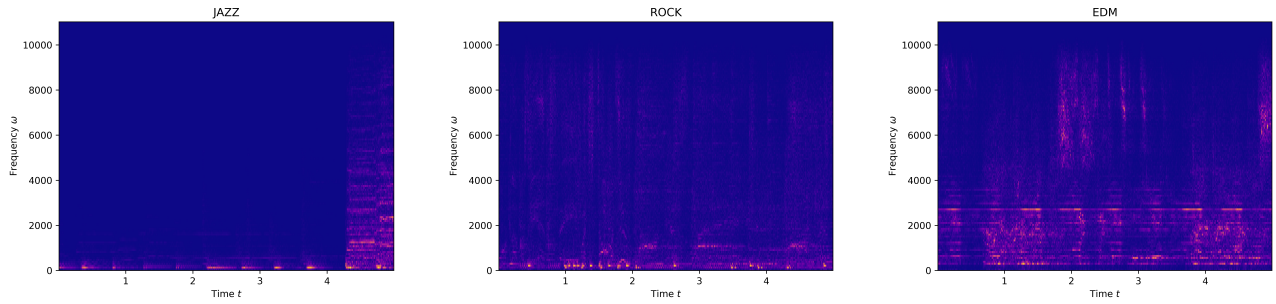

Figure 2: An example of each genre's spectrogram. Notice unique visual structures are present in both frequency and time

like PCA, we can build upon our work in low-rank representation of images to transform our audio recognition problem into an imag recognition problem. Sample spectrograms for each genre are show in Table 2

Because we know our samples' artist and genres are, we can construct labeled datatsets. As a result, we are able to use supervised machine learning for artist and genre classification. We construct arrays of reshaped spectrogram data, and a corresponding array of label data for the given spectrograms.

4.2 Creating Training and Test Data: PCA

While we could directly use our spectrogram data to preform classification, this would be a task suited more toward a Convolutional Neural Net (CNN). Because we are choosing to use more interpretable supervised machine learning classification algorithms, we want to reduce the order of datasets. As such, we will leverage the work that we have done on the Yale Faces (Eigenfaces) to represent our spectrogram images in a low-order combination of PCA basis vectors. First, we must preform SVD analysis of our large collection of mean-centered spectrogram data A. Now, for a given rank r, we can construct a data matrix X as shown in Equation 3.

This data matrix is then split into Training and Test data. We choose to allocate 80% of our data to training, and the remaining 20% to testing. This should allow enough data for training, yet still have enough to gain meaningful insight while testing our model.

We will be testing our model in three situations with the following amount of training data:

- Genre Classification (1 band per genre) : We will train our model on 3 bands, each from different genres. We test across these same bands, and cross validate against a new rock song.
- Band Classification (within 1 genre) : We will train our model on 3 bands, each from the same genre. We test across these same bands, and cross validate against a new song from one of the bands.
- Genre Classification (3 bands per genre) : We will train our model on 9 bands (3 from each genre). We test our model on correctly classifying the genre of any of these bands, and cross validate against a new rock song

4.3 Cross Validation

In addition to training, and test sets, an *essential* component of Machine Learning models is the use of a 3rd set called the Validation Set. It is important that our model never sees this information before it is tested against it. The preformance of the model on the Validation Set is our measure of how our model will preform for truly unknown data. For this analysis, we will choose a song from Aerosmith (genre: rock) that is not included in the album that our training and test sets are pulled from. We will use the song *Dude Looks Like a Lady* for validation, as it is not on the album *Aerosmith's Greatest Hits* that is used for training and testing. In the words of J. N. Kutz, "*Dude Looks Like a Lady* is really just a song about miss-classification," so it is only fitting that we use this 80's hair metal anthem for validation.

4.4 Machine Learning Implementation

Our choice to use Support Vector Machine was largely guided by the scikit-learn documentation article "Choosing the right estimator". As a result, I implemented Linear SVM with data projected into the first $r = 25$ modes of the SVD. I chose to use only 25 modes because it captured all of the major components of the singular value spectrum before the 'elbow' on the plot of σ values. Time was spent exploring Gaussian Naive Bayes. This was also successful, however I chose only to fully investigate and discuss SVM classification for this analysis.

5 Computational Results

Results of the Linear SVM with a rank $r = 25$ reconstruction of song spectrogram data are summarized in Table 2. We can see that in general, our model does very well. With an 80/20 split of training and test data, we are able to create a model that preforms well with both test and validation sets.

As expected, the accuracy of our classification of bands within the same genre (Case 2) is much lower than the other cases. This makes sense in the context of how our analysis was preformed. Bands who's songs are from the same genre will tend to have similar instruments (frequency content), tempo (x-coordinate feature spacing), and other features that will manifest themselves in the spectrograms, and subsequently SVD projections of the spectrograms. Contrary to the difficulty of classifying within the one genre, our validation set preforms quite well. Further investigation could look into the cause of this phenomenon. Possibilities include that the other two bands from the genre sound more similar than that used for validation, and our model has trouble telling them apart. Additionally the classification of "Aerosmith or not Aerosmith?" is simply less difficult than "What is this band?" as less possible combinations exist.

In addition, the overall performance of Case 3 is a bit lower than that of Case 1. While we have more data for training, we are limited by the inherent variability of a bands sound, even within one album. In fact, it could be argued that musical groups intent to have many styles of songs present throughout an album. Like the un-centered and un-cropped Yale Faces, this could increase variability in our data, and subsequently our model's ability to make accurate predictions.

Test	Training Size	Test Size	Accuracy	Cross Validation
Genre Classification (3 Bands)	720	180	88.89%	92.45%
Band Classification (1 Genre)	720	180	75.56%	90.57%
Genre Classification (9 Bands)	2160	540	87.22%	83.02%

Table 2: Summary of Linear Support Vector Machine Classification Accuracy

The next steps for this analysis could include introducing better and more sophisticated performance metrics for our classifier and working to interface our model to work with more genres, artists, albums, and ultimately entire music databases (did someone say "Big Data?"). Supervised Machine Learning models other than SVM could be used, or we could implement kernels other than linear. In additon, we could use an Unsupervised Machine Learning classifier to gain information into the correlation between different bands and genres. We could also use other methods than SVD to create low-rank structures for our data. We could leverage Independent Component Analysis, Wavelet Decomposition, and other signal processing tools. Another direction to take this analysis would be as an introduction to neural nets and the corresponding Python libraries (tensorflow and keras). In the workds of J. N. Kutz, the important thing to remember is "Deep Learn Responsibly."

6 References

Aerosmith, Greatest Hits. Columbia Records ,1980

Aerosmith, Dude Looks Like a Lady. Columbia Records, 1987

Brunton, S., & Kutz, J. (2019). *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge: Cambridge University Press.

Count Baise, Chairman of the Board. Roulette,1959

Def Leppard, Adrenalize. Mercury Records, 1992

Flume, Flume. Mom + Pop Music, 2012

J. N. Kutz, *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

Louis The Child, Kids at Play EP.Interscope, 2018

Miles Davis, Kind of Blue. Columbia 30th Street Studio, 1959

Mötley Crüe, Dr. Feelgood. Little Mountain Sound Studio , 989

Odesza, In Return. Counter Records, 2014

Sidney Bechet, Petit Fleur. Gennett Records, 1952

7 Appendix A: Functions Used

For this analysis, I relied heavily on the numpy and matplotlib libraries for Python. In addition, machine learning algorithms were implemented using the SckiKitLearn library. For loading the audio files, I used Librosa. Below are some specific functions used:

`librosa.load`: Load mp3 (or other audio) into Python with desired sampling rate, length, offset, and other useful options

`np.linalg.svd`: The heart of the math portion of this analysis. Computes the Singular Value Decomposition

`sklearn.model_selection.train_test_split`: Return training and test set data and labels with desired ratios.

`sklearn.svm.LinearSVC`: Linear Support Vector Machine Classifier object. Used in conjunction with fit and score functions

`LinearSVC.fit`: Fit the desired model to the given training data with labels

`LinearSVC.score`: Return score of model on test data's performance, given known labels.

8 Appendix B: Code

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Mar  4 15:16:07 2019

@author: elimiller
"""

import numpy as np
import matplotlib.pyplot as plt
import scipy.signal
import librosa
import sklearn



def get_features(data, rate, samples, label, plot_spectrograms=False):

    data = np.array(data, dtype='float64')
```

```
    sample_length = 5 * rate
    #indiceis for 5 second clips
    num_samples = int(np.floor(len(data) / sample_length))
    #round so that we can reshape without residuals

    trunc_index = sample_length * num_samples

    data_mat = np.reshape(data[0:trunc_index], (num_samples, sample_length)
    np.random.shuffle(data_mat)

    if samples == 'max':
        samples = num_samples
    data_set = data_mat[0:samples, :]

    feature = [label] * np.shape(data_set)[0]
    #make a list of input genre for the album data
    for j in range(samples):
        f, t, Sxx =  scipy.signal.spectrogram(
                data_set[j,:], rate, scaling='spectrum', mode='magnitude')
        if plot_spectrograms:
            plt.figure()
            plt.pcolormesh(t, f, Sxx, cmap='plasma')
            plt.title(label.upper())
            plt.xlabel('Time $t$')
            plt.ylabel('Frequency $\omega$')

        if j ==0:
            spectrogram_data = np.zeros((samples, np.size(Sxx.flatten())))

        spectrogram_data[j, :] = Sxx.flatten()
    return spectrogram_data, feature


genres = np.array(['jazz', 'rock', 'edm'])
#genres = np.array(['rock'])
samplenums = np.array(['0', '1', '2'])
#samplenums = np.array(['0'])


MAKEFILES = True
SAVEFILES = False
LOADFILES = False


short_length = 60*25
#duration=short_length
```

```
#samples_per_file = int(short_length / 5)
samples_per_file = 'max'


A = []
labels = []



for genre in genres:
    for samplenum in samplenums:
        if MAKEFILES:
            path = '/Users/elimiller/Desktop/AMATH482/HW4/Audio_Files/' + g
            data, rate = librosa.load(
                    path, res_type='kaiser_fast', offset=0, duration
= 25*60)
        if LOADFILES:
            rate = 22050
            data = np.load(genre+samplenum+'.npy')

        if SAVEFILES:
            np.save('%s%s'%(genre, samplenum),data)

        print('genre: %s, sample: %s loaded' %(genre, samplenum))

        x_data, y_data = get_features(
                data,
                rate,
                samples_per_file,
                genre,
                plot_spectrograms=False)

        A.append(x_data)
        labels.append(y_data)

A_array = np.array(A)
A_new = np.reshape(
        A_array,
        (np.shape(A_array)[0]*np.shape(A_array)[1], np.shape(A_array)[2]))


labels_array = np.array(labels)
labels_new = np.ravel(np.reshape(labels_array, (np.size(labels_array),1)))


A_new_centered = A_new - A_new.mean(axis=1, keepdims=True)
U, S, V = np.linalg.svd(A_new_centered, full_matrices =False)
plt.figure()
```

```
plt.plot(S/np.sum(S), 'o-')
plt.title('Singular Value Spectrum of Audio Data')

import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
#suppress warnings for printouts
from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVC

rank = 25

representation = A_new_centered @ V.T[:, 0:rank]

x_train, x_test, y_train, y_test = train_test_split(
        representation, labels_new, train_size=.80)

svm_clf = LinearSVC()
svm_clf.fit(x_train, y_train)
#y_predict = clf.predict(x_test)
svm_score = svm_clf.score(x_test, y_test)
print('SVM Accuracy %f' %svm_score)


path = '/Users/elimiller/Desktop/AMATH482/HW4/Audio_Files/DudeLooksLikeALad
val_data, val_rate = librosa.load(
                    path, res_type='kaiser_fast', offset=0)

validation_set, validation_labels = get_features(
                val_data,
                val_rate,
                'max',
                'rock')

validation_centered = validation_set - validation_set.mean(
        axis=1, keepdims=True)
validation_rep = validation_centered @ V.T[:, 0:rank]
val_score = svm_clf.score(validation_rep, validation_labels)
print('Validation Score: %f' %val_score)
```