Homework 4: Supervised Machine Learning with Music Classification

Audrey Shingleton

March 6, 2020

Abstract

This project uses supervised machine learning to classify five second music clips from various artists and genres. By performing linear discrimination analysis on the SVD of the spectrograms from songs clips, I was able to create classification algorithms from previously labeled training data and evaluate them on testing data.

I. Introduction and Overview

People are usually able to identify the artist or genre of music. I want to determine if a computer can be trained to do the same. The purpose of this project is to create an algorithm to best classify the genre or artist of various music clips given previously labeled training data. By using techniques from spectrograms, the Singular Value Decomposition, Principal Component Analysis, and Linear Discrimination Analysis, I will determine threshold values to classify music data. The tests of classification are using three artists from three different genres, three artists from the same genre, and three genres each with with three artists. I will be training and testing my classification algorithm using 20 five second song clips from each artist used in the various tests.

II. Theoretical Background

Singular Value Decomposition and Principal Component Analysis

The Singular Value Decomposition can be used to write any matrix X as

$$X = U\Sigma V^* \tag{1}$$

where $\Sigma^2 = \Lambda$, a diagonal matrix whose entries correspond to the eigenvalues of XX^T.[1] Applications of the SVD is Principal Component Analysis (PCA),

used to reduce data into lower dimensions. New variables can be generated with ΣV^* using the PCA modes from U.[1]

Linear Discrimination Analysis

Linear Discrimination Analysis (LDA) is a method which uses a statistical decision threshold for discrimination between classes. It uses use statistical information about the SVD decomposition to make a decision about the data. The goal of LDA is to find a suitable projection w that maximizes the distance between the inter-class data while minimizing the intra-class data.[1] This is so the data is well separated such that the means are well apart when projected. Sb and Sw are between-class and within-class scatter matrices for data with n classes. Sb is defined as

$$Sb = \sum_{j=1}^{n} (\mu_j - \mu)(\mu_j - \mu)^T m_j \tag{2}$$

where m_j is the number of samples in class j, μ is the average answer of all the data, and μ_j is the average answer of class j. Sw is defined as

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

such that

$$Sb(w) = \lambda Sw(w) \tag{4}$$

where the maximum eigenvalue λ and its associated eigenvector gives the quantity of interest and the projection basis.[1] The decision threshold level for classification can then be determined from the LDA basis for projecting new data.

III. Algorithm Implementation and Development

The method to create and test a classifier was very similar for each test case. I started by reading in five second samples of songs, 20 samples per artist, and downsampled the data. This ensured faster processing and that the data for each song clip would be the same size. Then I constructed the spectrogram of each song clip. The spectrogram was re-shaped and added to a matrix X where each column represented a five second song clip. I subtracted the mean from each row in X to center the data and performed the SVD on this matrix.

From this, I generated new song variables from the artists with $S * V'$ giving strength to the projection on the PCA/POD modes generated by U. Then I randomly split up the these songs into testing and training data.

Using the training data, I found the between-class and within-class scatter matrices to determine the LDA basis. I projected the training and test data onto

Table 1: Songs Correctly Classified for Varying Features

10 features: 8	44.44%
20 features: 9	50%
30 features: 9	50%

Table 2: Confusion Matrix from Multiple Artist Classification

	Ariana Grande	Luke Bryan	Bryce Vine
Predicted Ariana Grande	3	1	0
Predicted Luke Bryan	0	0	0
Predicted Bryce Vine	3	5	6

this basis. Then I determined the two appropriate threshold values for song classification based on the mean projection values for each artist/genre in the training data.

Finally, I tested the classification algorithm with the projection values of the test data, determining the predicted artist/genre based on if it was below, above, or in between the two previously determined threshold values. I then repeated this process using PCA with 10, 20, and 30 features to evaluate which gave the best classification results.

IV. Computational Results

Test 1: Band Classification

For test 1, I used 20 five second song clips from Ariana Grande (pop), Luke Bryan (country), and Bryce Vine (rap). I randomly selected 14 songs clips from each of them for training data and 6 for testing data. I found that using 20 features gave me the best classification results, predicting the artists of 50% of the testing songs correctly (Table 1).

Using the best classification method (20 features), I found that predicting Ariana Grande and Bryce Vine songs was the most accurate (Table 2). The three Ariana Grande songs incorrectly classified were "sometimes", "god is a woman", and "dangerous woman", all of which were classified as Bryce Vine. These are a few of her slower songs, which Bryce Vine tends to have more of. Interestingly enough, none of the songs were predicted to be from Luke Bryan, and all of his songs were classified as being from Bryce Vine.

Table 3: Songs Correctly Classified for Varying Features

10 features: 8	44.44%
20 features: 10	55.56%
30 features: 8	44.44%

Table 4: Confusion Matrix of Multiple Artists from Same Genre

	Morgan Wallen	Luke Bryan	Eric Church
Predicted Morgan Wallen	2	0	0
Predicted Luke Bryan	0	2	0
Predicted Eric Church	4	4	6

Table 5: Songs Correctly Classified for Varying Features

10 features: 16	35.56%
20 features: 10	22.22%
30 features: 13	28.9%

Test 2: Same Genre Classification

For test 2, I used 20 five second song clips from Morgan Wallen, Luke Bryan, and Eric Church. These are all country artists, but each have a different style. For instance, Eric Church has a more rock-country vibe and Morgan Wallen has a higher voice than the other artists. I again randomly selected 14 songs from each of them for training data and 6 for testing data. I found that using 20 features gave me the best classification results, predicting the artists of 55.56% of the testing songs correctly (Table 3). This was surprisingly higher than the prediction accuracy using three different artists from different genres.

Eric Church was the most common prediction for each song (Table 4). Additionally, all of his songs were correctly classified. I feel like this makes sense considering his style of music is the most diverse.

Test 3: Different Genre Classification

For test 3, I used 60 five second song clips from country, pop, and rap music. Country artists included Morgan Wallen, Luke Bryan, and Eric Church. The pop artists were Ariana Grande, Camila Cabello, and Ed Sheeran. The rap artists were Bryce Vine, Mackelmore, and Travis Scott. I randomly selected 45 songs from each genre for training data and 15 for testing data. I found that using 10 features gave me the best classification results, predicting the genre of 35.56% of the testing songs correctly (Table 5).

Table 6: Confusion Matrix of Different Genres

	Country	Pop	Rap
Predicted Country	5	2	4
Predicted Pop	8	10	10
Predicted Rap	2	3	1

Pop was the most common guess from the classifier and the most accurate. For country songs, only five out of the 15 were correctly classified. However, this is much higher than rap which only had one song correctly classified (Table 6).

V. Summary and Conclusions

Overall, I noticed that artist classification was much more accurate than genre classification. I think this is potentially due to the fact that modern music genres aren't as black and white as they used to be. Pop artists Ariana Grande and Ed Sheeran feature rap in their songs, country singer Eric Church could also be considered rock, and rapper Mackelmore frequently makes use of trumpets which is very unique in rap music. Because of this, I think by separating the genres even to include more classes such as country-pop, hip-hop/rap, etc. would improve classification.

Additionally, it is important to evaluate classification methods using a range of features. More features didn't necessarily give more accurate predictions and should thus be taken into account when developing a classification algorithm.

Appendix A. MATLAB functions used and explanations

- svd(X): Performs the singular value decomposition on matrix X
- diag(X): Returns a vector with the diagonal values from a square matrix X
- fft: Transforms the function to the frequency domain using the Fast Fourier Transformation algorithm
- fftshift: Shifts the function back to its mathematically correct positions for plotting
- audioread: Opens a wav file as an array of signal data
- reshape: Reshapes an array into specified dimensions
- eig(A,B): Returns a diagonal matrix D of generalized eigenvalues and V with the corresponding eigenvectors so $AV = BVD$

- norm(X,2): Returns the 2-norm of matrix X

Appendix B. MATLAB codes

```
%%
clear all; close all; clc;
%% TEST 1
% add more samples (14 training, 6 test)
alldata = [];
artists = ["ariana_grande", "luke_bryan", "bryce_vine"];
myDir = '/Users/audrey/Desktop/Music/';
songs = 20;
for i = 1:length(artists)
    for j = 1:songs % 5 5" clips from each artist
        [song, Fs] = audioread(strcat(myDir,artists(i),"/",num2str(i),".wav"));
        song = song(:,1); % make mono
        a = song(1:4:22050,:)'; % downsample song

        % get spectrogram
         Sgt_spec = [];
         L = 5;
         n = length(a);
         t2 = linspace(0,L,n+1);
         t = t2(1:n);
         tslide = 0:0.1:L;
          for k = 1:length(tslide)
              g = exp(-10*(t - tslide(k)).^2);
              Sg = g.*a;
              Sgt = fft(Sg);
              Sgt_spec = [Sgt_spec; abs(fftshift(Sgt))];
          end
        [m, n] = size(Sgt_spec);
        data = reshape(Sgt_spec, m*n, 1);
        alldata = [alldata data];
    end
end

[u, s, v] = svd(alldata - mean(alldata(:)), 'econ');
%plot(diag(s),'ko','Linewidth',2)
singers = u*v';
% generate random nums to split data
% q = randperm(songs);
% save permutaion
q = [16 15 12 14 13 5 2 6 8 17 18 1 4 20 9 11 7 10 19 3];
%%
```

```
features = 20;
tr_size = 14;
test_size = songs - tr_size;

ari = singers(1:features,1:20);
luke = singers(1:features,21:40);
bryce = singers(1:features,41:end);

%% training sets
ari_train = ari(:,q(1:tr_size));
luke_train = luke(:,q(1:tr_size));
bryce_train = bryce(:,q(1:tr_size));

% test sets
ari_test = ari(:,q(tr_size+1:end));
luke_test = luke(:,q(tr_size+1:end));
bryce_test = bryce(:,q(tr_size+1:end));

m1 = mean(ari_train,2);
m2 = mean(luke_train,2);
m3 = mean(bryce_train,2);
m_all = mean(singers);

% Within class
Sw = 0;
for i=1:tr_size
    Sw = Sw + (ari_train(:,i)-m1)*(ari_train(:,i)-m1)';
end
for i=1:tr_size
    Sw = Sw + (luke_train(:,i)-m2)*(luke_train(:,i)-m2)';
end
for i=1:tr_size
    Sw = Sw + (bryce_train(:,i)-m3)*(bryce_train(:,i)-m3)';
end

% Between class
Sb = 0;
Sb = Sb + (m1-m_all)*(m1-m_all)'*tr_size;
Sb = Sb + (m2-m_all)*(m2-m_all)'*tr_size;
Sb = Sb + (m3-m_all)*(m3-m_all)'*tr_size;
%% Project onto single vector
[V2,D] = eig(Sb,Sw);
[~,ind] = max(abs(diag(D)));
w = V2(:,ind); w = w/norm(w,2);

vari = w'*ari_train;
```

```
vluke = w'*luke_train;
vbryce = w'*bryce_train;

vari_test = w'*ari_test;
vluke_test = w'*luke_test;
vbryce_test = w'*bryce_test;
pval = [vari_test vluke_test vbryce_test];
answer = [ones(test_size,1); 2*ones(test_size,1); 3*ones(test_size,1)];

disp("means")
disp(mean([vari;vluke;vbryce],2));
disp(sort(mean([vari;vluke;vbryce],2)));
sort_ari = sort(vari);
sort_luke = sort(vluke);
sort_bryce = sort(vbryce);

%% 10 modes: ari (1) < luke < bryce
t1 = tr_size;
t2 = 1;
while sort_luke(t1) > sort_bryce(t2)
    t1 = t1-1;
    t2 = t2+1;
end
thresh_min = (sort_luke(t1)+sort_bryce(t2))/2';

% ari < luke
t1 = tr_size;
t2 = 1;
while sort_ari(t1) > sort_luke(t2)
    t1 = t1-1;
    t2 = t2+1;
end
thresh_max = (sort_ari(t1)+sort_luke(t2))/2';
test = [];
for i=1:test_size*3
    if pval(i) > thresh_max
        test = [test;3];
    elseif pval(i) < thresh_min
        test = [test;1];
    else
        test = [test;2];
    end
end

count_LDA = 0;
for i=1:test_size*3
```

```
    if test(i) == answer(i)
        count_LDA = count_LDA + 1;
    end
end
disp("Test 1: 10 modes")
disp(count_LDA) % 7 right
disp(count_LDA / (test_size*3)) % 38.89%

%% 20 modes: ari < bryce < luke
% bryce < luke
t1 = tr_size;
t2 = 1;
while sort_luke(t1) > sort_bryce(t2)
    t1 = t1-1;
    t2 = t2+1;
end
thresh_min = (sort_luke(t1)+sort_bryce(t2))/2';

% ari < luke
t1 = tr_size;
t2 = 1;
while sort_ari(t1) > sort_luke(t2)
    t1 = t1-1;
    t2 = t2+1;
end
thresh_max = (sort_ari(t1)+sort_luke(t2))/2';
test = [];
for i=1:test_size*3
    if pval(i) > thresh_max
        test = [test;3];
    elseif pval(i) < thresh_min
        test = [test;1];
    else
        test = [test;2];
    end
end

count_LDA = 0;
for i=1:test_size*3
    if test(i) == answer(i)
        count_LDA = count_LDA + 1;
    end
end
disp("Test 1: 20 modes")
disp(count_LDA) % 9 right
disp(count_LDA / (test_size*3)) % 50%
```

```
%% build matrix
counts = zeros(3,3);
for i=1:test_size*3
    t = test(i);
    a = answer(i);
    for j=1:3
        for k=1:3
            if (t==j) && (a==k)
                counts(j,k) = counts(j,k) + 1;
            end
        end
    end
end

%% 30 modes: ari < bryce < luke
% bryce < luke
t1 = tr_size;
t2 = 1;
while sort_bryce(t1) > sort_luke(t2)
    t1 = t1-1;
    t2 = t2+1;
end
thresh_min = (sort_bryce(t1)+sort_luke(t2))/2';

% ari < bryce
t1 = tr_size;
t2 = 1;
while sort_ari(t1) > sort_bryce(t2)
    t1 = t1-1;
    t2 = t2+1;
end
thresh_max = (sort_ari(t1)+sort_bryce(t2))/2';
test = [];
for i=1:test_size*3
    if pval(i) > thresh_max
        test = [test;2];
    elseif pval(i) < thresh_min
        test = [test;1];
    else
        test = [test;3];
    end
end

count_LDA = 0;
for i=1:test_size*3
```

```
    if test(i) == answer(i)
        count_LDA = count_LDA + 1;
    end
end
disp("Test 1: 30 modes")
disp(count_LDA) % 9 right
disp(count_LDA / (test_size*3)) % 50%
```

References

[1] Kutz, Nathan J. "Computational Methods for Data Analysis." *In Data-Driven Modeling Scientific Computation: Methods for Complex Systems Big Data.*. Oxford University Press, 2013.