AMATH 582 Homework 5: Music Genre Identification

Greg DeVore

March 11, 2012

Abstract

The objective of this homework assignment is to use several powerful mathematical techniques from the fields of signal processing (time-frequency analysis), linear algrbra (Singular Value Decomposition) and statistics (Linear Discrimination Analysis) to build an algorithm capable of accurately identifying specific genres of music.

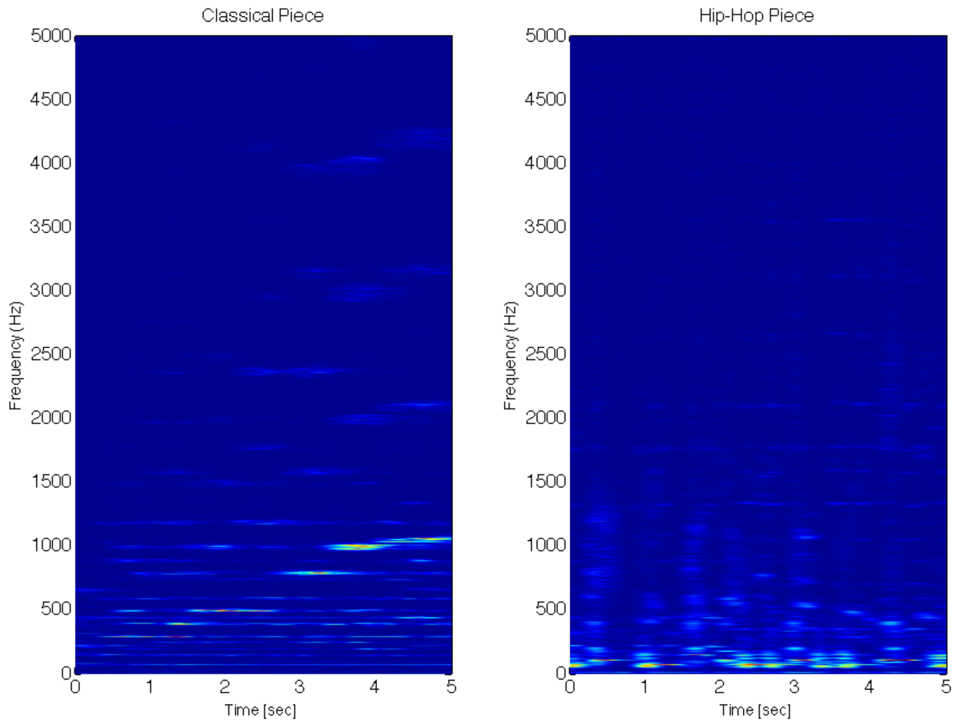

Figure 1: Spectrogram of a classical piece (Mendelssohn) compared to a hip-hop piece (Surreal). Note the differences in the dominant modes within each genre. These types of differences will be exploited in order to classify genres using a statistical testing algorithm.

Contents

1 Introduction and Overview **3**

2 Theoretical Background **3**

2.1 The Gábor Transform 3

2.2 The SVD 4

2.3 LDA 5

3 Algorithm Implementation and Development **6**

3.1 Forming the Training Sets 6

3.2 Creating the Classification Algorithm 6

3.3 Testing New Songs 6

4 Computational Results **7**

4.1 Test 1 - Band Classification (3 Genres) 8

4.2 Test 2 - Band Classification (1 Genre) 11

4.3 Test 3 - Genre Classification (3 Genres) 14

5 Summary and Conclusions **17**

6 Appendix A - Summary of Matlab Commands Used **18**

7 Appendix B - Matlab code **20**

7.1 Main Function 20

7.2 Training Function 23

7.3 Import Training Set Clips 24

7.4 Import Test Set Clips 24

7.5 Spectrogram Generation Function 25

1 Introduction and Overview

Over the past few years, one of the most exciting areas in the field of mathematics has been the topic of machine learning, or teaching computers to become aware of the world around them in order to recognize and identify people, objects, and sounds. Teaching computers to mimic what humans can easily do is extremely complicated, and unites many different topics from signal processing, linear algebra, and statistics.

In this assignment, training sets comprised of short clips from different musical genres will be used to construct an algorithm that is capable of identifying specific genres of music. This will be done by creating spectrograms of each piece (see Figure 1), and using the inherent differences between genres (in terms of the dominant modes captured within the spectrogram) to classify new music clips into their respective genre. The mathematical techniques used to build the algorithms include Gábor transforms, the Singular Value Decomposition (SVD), and Linear Discrimination analysis.

Problem Requirements

1. Choose three different bands from three separate genres. Build training sets using 5-second clips of different songs from each band and built an algorithm capable of correctly identifying new 5-second clips from each band
2. Repeat the first task, but now with three different bands from within a single genre. This will make identifying new clips much more challenging. Compare the accuracy to the algorithm from test 1.
3. Repeat the first task, but now broaden the algorithm to simply identify a certain genre of music. Training sets should consist of song clips from different bands from within the same genre. Test the algorithm by attemping to classify new clips from bands within the same genre (that were not part of the training set).

2 Theoretical Background

2.1 The Gábor Transform

Applying a Fourier transform to a time signal reveals all of its frequency content, which makes it a very powerful tool in signal analysis. However, during the transform *all* time content is removed. This means that when viewing a transformed signal in the frequency domain, it is impossible to tell at what moment in time various frequencies occured. The power of the Gábor transform is that it allows for the existence of both time and frequency data in signal analysis.

The Gábor transform, also known as the short-time Fourier transform, is given by

$$G[f](t,\omega) = \int_{-\infty}^{\infty} f(\tau)g(\tau - t)e^{-i\omega\tau}d\tau \tag{1}$$

The term inside the integral is identical to the Fourier transform except for the addition of the term $g(\tau - t)$. This term was introduced by Gábor, and it localizes both time and frequency by 'sliding' a window along a given time signal, isolating select pieces and providing the frequency content of just that piece. An example of a Gaussian type Gábor window is shown in Figure 2. Note the isolatation of a small piece of the time signal and the resulting frequency content. This technique will be used to create a spectrogram of each clip (see Figure 1), which contains the frequency content of the clip as a function of time.

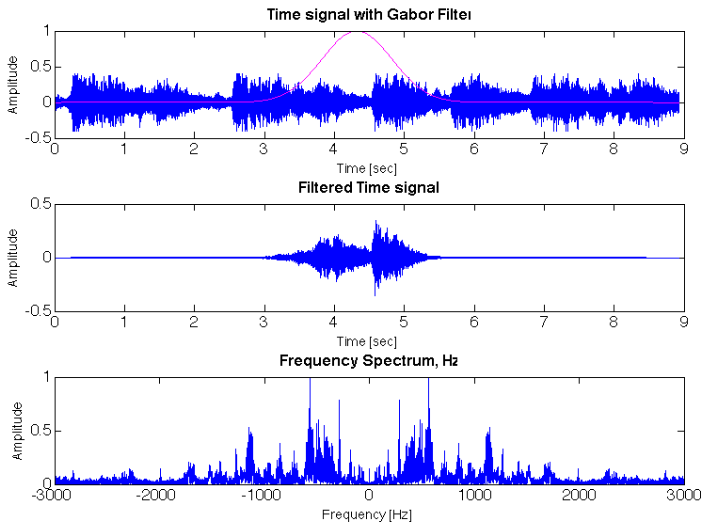

Figure 2: Example of a Gábor window (Gaussian type).

The $g(\tau - t)$ term in Equation 1, also known as the *kernel*, for a Gaussian type window is

$$g = e^{-a(t-\tau)^2} \tag{2}$$

Where a is a parameter representing the width of the window, and τ represents the amount of the shift along the time domain.

2.2 The SVD

The SVD is one of the most important tools in the field of Linear Algebra. The fact that it is guaranteed to exist for any matrix, as well as its numerous useful properties, make it a go-to tool for a wide range of applications. The SVD decomposes a matrix, A, of size $m \times n$, into the following constituent matrices

$$A = U\Sigma V^* \tag{3}$$

Where

- U is an $m \times m$ unitary matrix
- Σ is an $m \times n$ diagonal matrix whose entries are the *singular values* of A
- V is an $n \times n$ unitary matrix

Graphically, the SVD illustrates the following transformation

$$AV = U\Sigma \tag{4}$$

Where the matrix A acts on a set of vectors V that are originally represented as a unit sphere and transformed into a hyperellipse ($U\Sigma$) under the image of the matrix. Since V is unitary, $V^* = V^{-1}$ and the result from Equation 3 is obtained.

This decomposition will be used to identify the dominant features from different genres of music. For this assignment, the matrix A will be the training set, where each column represents a different song. The dominant modes (or features) from the training sets will be contained in the columns of U, and the singular values from Σ represent the strength of the projection of each song onto these dominant modes. The matrix ΣV gives the actual projection of each song onto these dominant modes.

2.3 LDA

Thus far, techniques have been discussed for building training sets using spectrograms of song clips, and identifying the dominant modes of those clips using the SVD. The final piece of the algorithm is to use information from this decomposition to classify new song clips within a specific genre. Linear Discrimination Analysis (LDA) will make this possible.

The goal of LDA is to allow us to project the training set data (comprised of different musical genres, or *classes*) onto a new subspace, one where the distance between the classes is maximized, while the variance within each class is minimized. If C classes are present, the subspace will be of dimension $C-1$. For example, with two classes the projection is a line (1-dimensional subspace), and with three classes the projection is a plane (2-dimensional subspace). Once this projection is obtained, new song clips will be projected onto the same subspace, and their location within this subspace (specifically their proximity to the nearest genre from the training set) will be used to classify them.

To compute the desired projection, consider the following equation

$$W = arg\ max \frac{|W^T S_B W|}{|W^T S_W W|} \tag{5}$$

Where W is the desired projection matrix, S_B is the between class variance, and S_W is the variance within each class. For C classes, the variances are defined as

$$S_B = \sum_{j=1}^{C} (\mu_i - \mu)(\mu_i - \mu)^T \tag{6}$$

$$S_W = \sum_{j=1}^{C} \sum_{i=1}^{M} (x_i - \mu_j)(x_i - \mu_j)^T \tag{7}$$

Where μ_i is the mean for each class, μ is the mean for all classes, M is the number of items in each class, and x is a single data point. The solution for W is obtained from the eigenvalue problem

$$S_B W = \lambda S_W W \tag{8}$$

The eigenvectors associated to the non-zero eigenvalues (there are at most $C-1$) form the projection matrix W. For C classes, W will have $C-1$ columns. To project the training set data onto the LDA subpace, it will be necessary to pre-multiply the training set data by the transpose of W.

3 Algorithm Implementation and Development

This problem was solved using Matlab, and all Matlab code is contained in Appendix B.

3.1 Forming the Training Sets

To create training sets for use on the classification algorithm, the following approach will be taken

1. Load all song clips (5 second *.WAV files) into Matlab using the *wavread* function.
2. For each clip, use a Gábor transform with 100 time slices and a window width of 40 to capture its frequency spectrum as a function of time. Save the resulting spectrogram.
3. Reshape each spectrogram into a single column vector and concatenate them together to form the training set.

3.2 Creating the Classification Algorithm

1. Pass the training set, along with the desired number of features, into the training function.
2. Once inside this function, take the SVD of the training set and retain the first m columns of U, where m is the number of desired features.
3. Compute the projection of each song onto these features by calculating ΣV and grab the projection for each genre by taking the first m rows and the corresponding columns out of this new matrix (for example, if there are eight songs from each genre in the training set, the first eight columns correspond to the first genre, the next eight to the second genre, and so on).
4. Compute the mean for each class, along with S_B and S_W.
5. Solve the eigenvalue problem from the previous section and form W using the eigenvectors associated with the non-zero eigenvalues.
6. Project each class onto the subspace formed by W by multiplying them by its transpose.
7. Concatenate the results for all classes and return the result as the output of the training function.

3.3 Testing New Songs

To classify new songs using the algorithm, the following approach will be taken

1. Load a new song into Matlab and generate its spectrogram using the method described in the previous section.
2. Compute the SVD projection of this song onto the dominant features by multiplying its spectrogram by the transpose of U, where U is from the original SVD of the training set (recall that it contains exactly m columns, corresponding to the number of desired features).
3. Project the SVD projection of the song onto the subspace provided by the LDA by multiplying it by the transpose of W.
4. Find the nearest genre within this subspace using the L^2 norm (Euclidian distance).
5. Compare this with the known genre (answer key) and record the guess as either a success or failure.
6. Repeat for several new songs to assess the accuracy of the algorithm.

4 Computational Results

Computational and Testing Parameters

The following parameters were used in the computation of all results

- Length of all song clips is 5 seconds.
- For tests 1 and 2, 20 songs from three different bands were used to construct the training set (60 songs total). For test 3, 25 songs from a number of different bands were used to create training sets for a specific genre (75 songs total). For tests 1 and 3, each band belongs to a different genre (rock, hip-hop, and classical), whereas for test 2 all bands belonged to the same genre (hip-hop).
- When analyzing the frequency spectrum for each clip and generating spectrograms, a cutoff of 5 khZ was used due to the fact that most genres of music do not have significant modes above this range.
- After a full Fourier transform was taken, the resulting frequency spectrum was sampled at every 5 hZ to build a more compact signal and improve computing performance.
- Each spectrogram was of size 1000×100, for a total of 100,000 data points per song.
- 5 features were extracted from the SVD of the training set.
- For all tests, 15 songs were used to test the algorithm. For tests 1 and 2, new songs from the same bands used to create the training set were used to test the band classification abilities of the algorithm. For test 3, songs from new bands were used to test the genre classifying abilities of the algorithm.

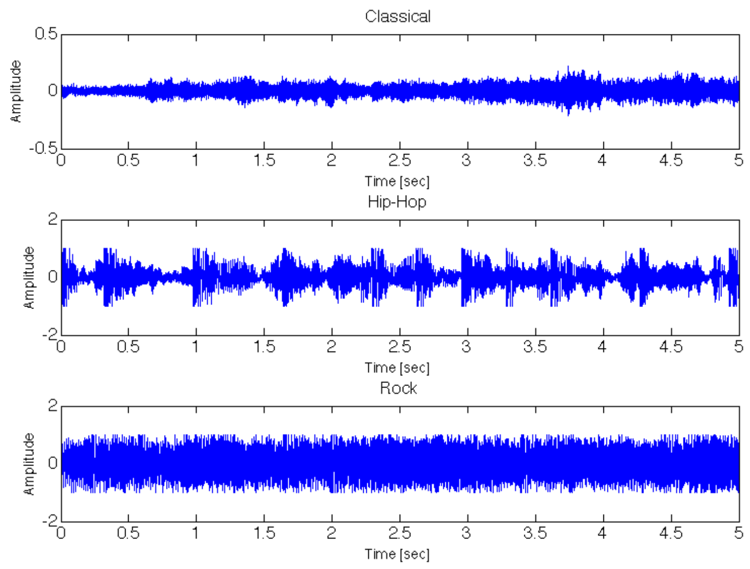

Figure 3: Sound clips from three different musical genres (classical, hip-hop, and rock). Note the differences between the genres.

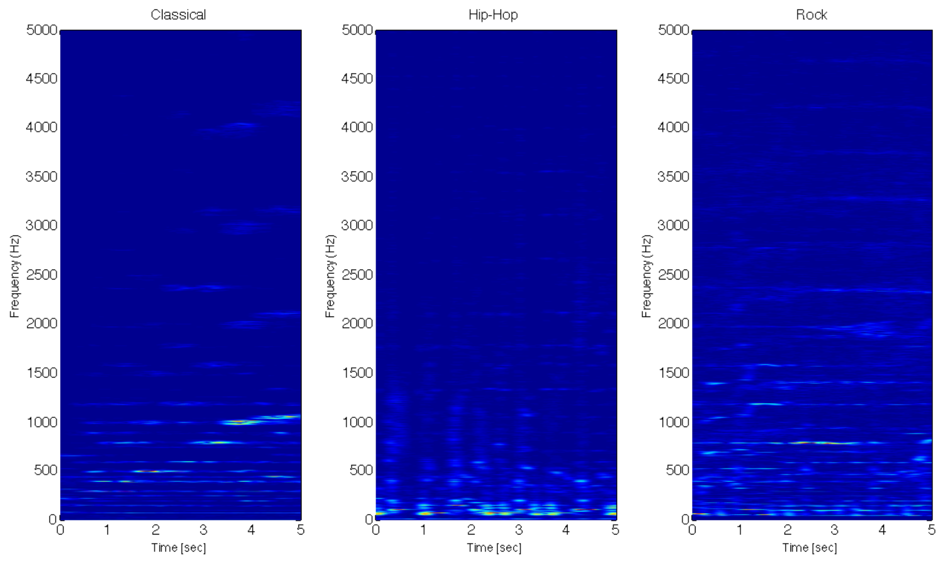

Figure 4: Spectrograms from three different musical genres (classical, hip-hop, and rock). Note the differences in the dominant modes for each genre.

4.1 Test 1 - Band Classification (3 Genres)

For the first test, three bands from three different genres were used to construct the training set. The three genres chosen were rock, hip-hop, and classical. Clips from each of these genres is shown in Figure 3. Note the distinct features of each genre. Classical music is very smooth, with no percussion; hip-hop is much more staccato in nature; and rock is a solid wall of sound. These types of features will help classify new songs into the correct genre.

Spectrograms of the same genres are shown in Figure 4. Now, the differences between the dominant modes for each genre can be directly observed. Note that for a classical piece, the dominant frequencies are very clean and distinct, and each note is observable. For a hip-hop piece, the distinct beats are indicated, and everything is occuring at much lower frequencies (due to the emphasis of bass). For a rock piece, a wide range of frequencies are present, but it is much harder to discern individual modes (this piece featured a lot of distorted guitar and drums, creating a wall of sound).

The singular values resulting from applying the SVD to the training set are shown in Figure 5. Note that one mode stands out above all others, and the rest, while not nearly as significant in terms of energy, appear to contain some information. To get a sense of which modes are the most significant, each of the songs in the training set was projected onto the first three SVD modes. This projection gives an indication of the strength of each of these modes in a given test song.

The result is shown in Figure 6. Note that for rock and classical songs, the first two modes are of the same sign and add together, but for hip-hop songs they are of opposite sign and thus try to cancel each other out. This suggests that the dominant mode might be of a higher frequency, meaning it is present in the classical and rock pieces, but not the hip-hop, which is dominanted by lower frequency modes.

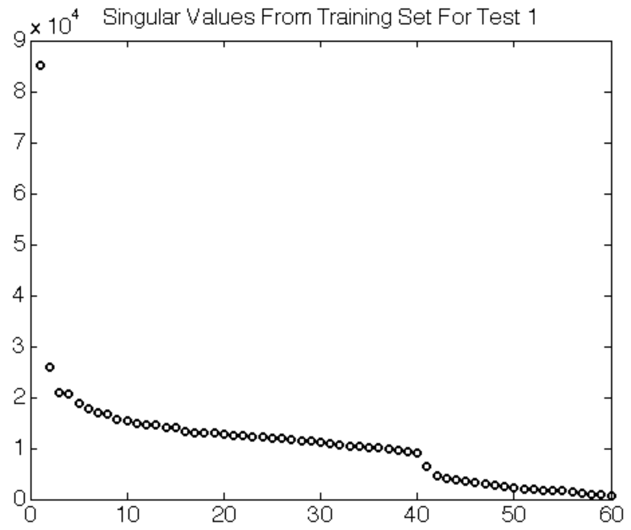

Figure 5: Singular values from the first training set (classical, hip-hop, and rock).

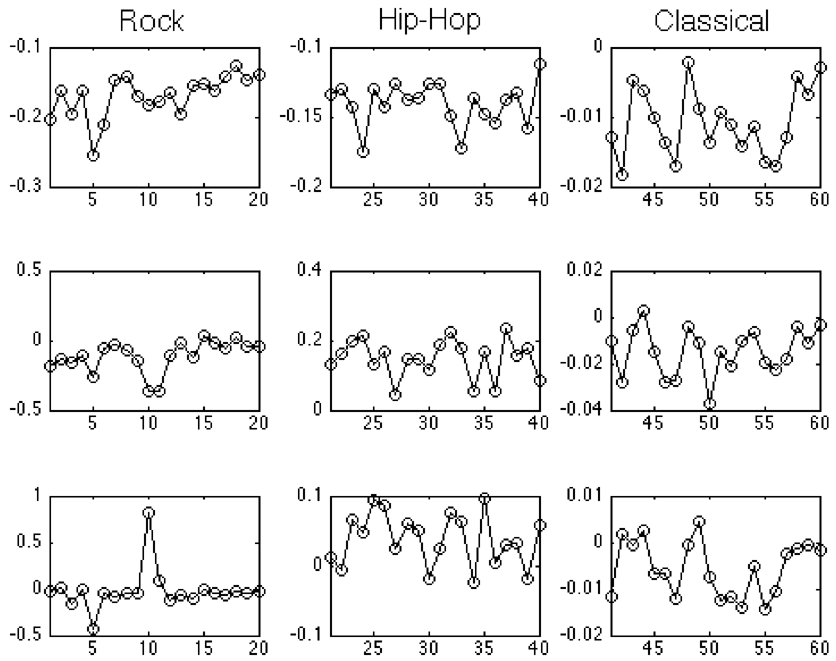

Figure 6: Projection of each first training set song onto the first three SVD modes.

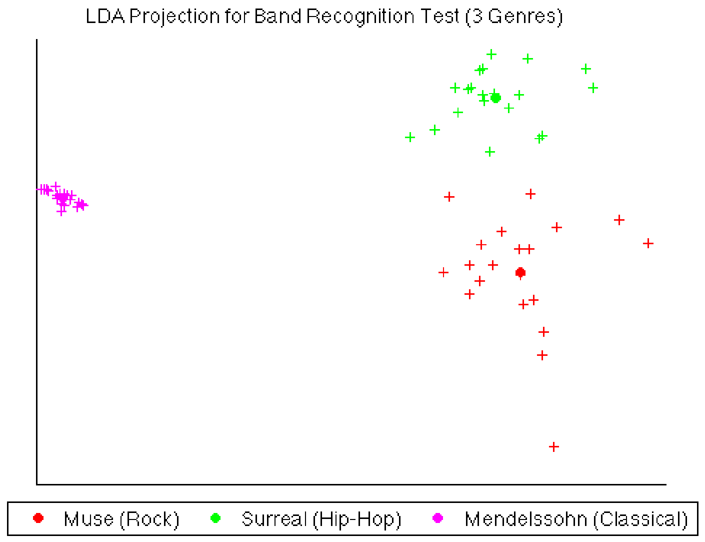

Figure 7: LDA projection for test 1 (classical, hip-hop, and rock).

The LDA projection for the training set songs is shown in Figure 7. Each individual song is shown as a cross, and the mean for each class is shown as a filled circle. The purpose of the LDA projection was to minimize within class variance while maximizing the distance between classes. This has been achieved for the classical pieces, but rock and hip-hop are quite spread out, suggesting more variance within the clips.

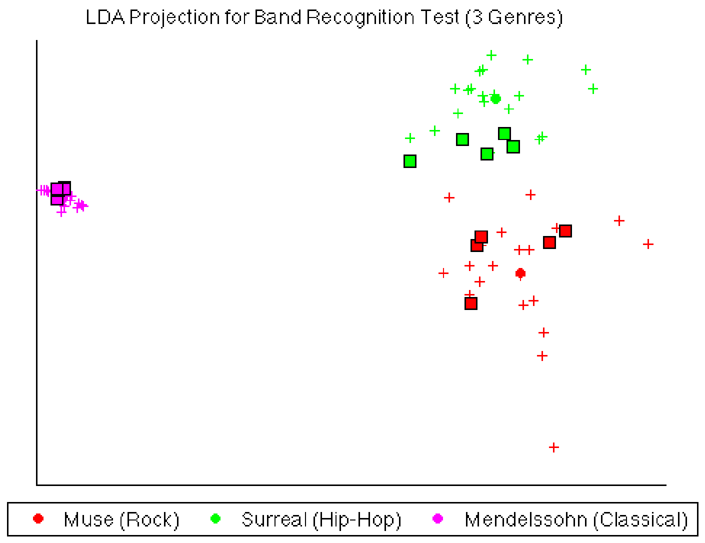

Figure 8: LDA projection for test 1 (classical, hip-hop, and rock) with new test songs included.

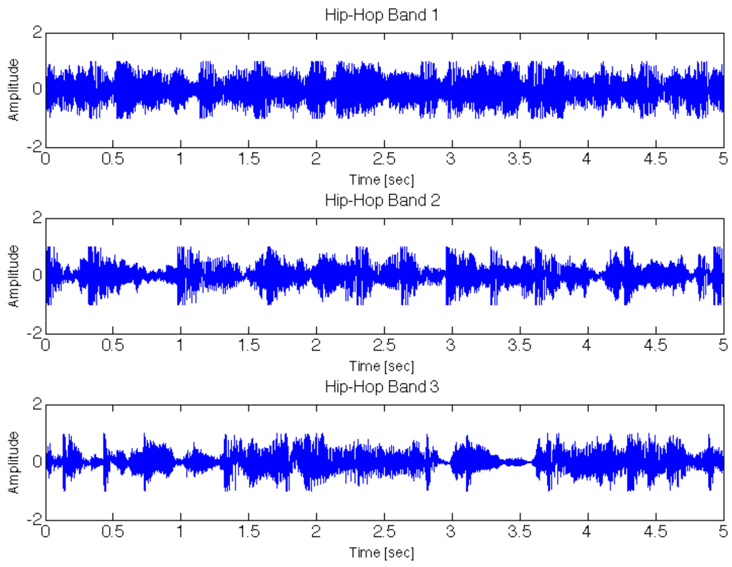

Figure 9: Sound clips from a single musical genre (hip-hop).

The result of projecting new songs from each band onto the LDA subspace is shown in Figure 8. The new songs are shown as filled squares, and observe that the new songs were projected in the same vicinity as the other songs from the same band. Desipite the proximity of the rock and hip-hop classes, all 15 test songs were located closer to their respective genres than to any other, giving a success rate of 100% to the first test.

4.2 Test 2 - Band Classification (1 Genre)

For the second test, three bands from a single genre were used to construct the training set. This time, only the hip-hop genre was used. Clips from each band within this genre is shown in Figure 9. Note that as opposed to test 1, it is much harder to pick out distinct features from within the same genre. All clips have the same staccato nature. This will make it much more difficult to classify new songs from the three bands.

Spectrograms from the single genre are shown in Figure 10. The differences between the dominant modes for this genre are harder to observe. For each band, distinct beats are clearly visisble, and as before, nearly all of the dominant modes occur at low frequencies.

The singular values resulting from applying the SVD to the training set are shown in Figure 11. Note that one mode stands out above all others (even more so than in the first test), and the rest do not appear to be nearly as important. Each of the songs in the training set was projected onto the first three SVD modes and the result is shown in Figure 12. Note that for the first two bands, the first two modes are of the same sign and add together, while for the third band, they are of the opposite sign and try to cancel each other out. This suggests that perhaps the first two bands possess more of the dominant mode.

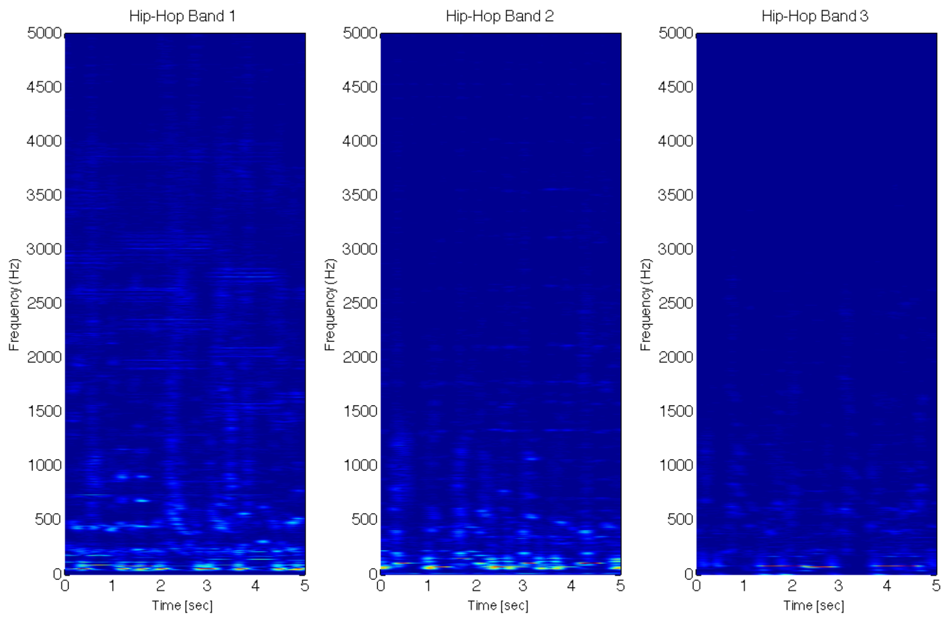

Figure 10: Spectrograms from a single musical genre (hip-hop).

The LDA projection for the training set songs is shown in Figure 13. Note that there is much more scatter within the data, and the classes are not nearly as clustered when compared to test 1. This makes sense, as there is a lot less variability between bands of a single genre compared to bands in different genres.

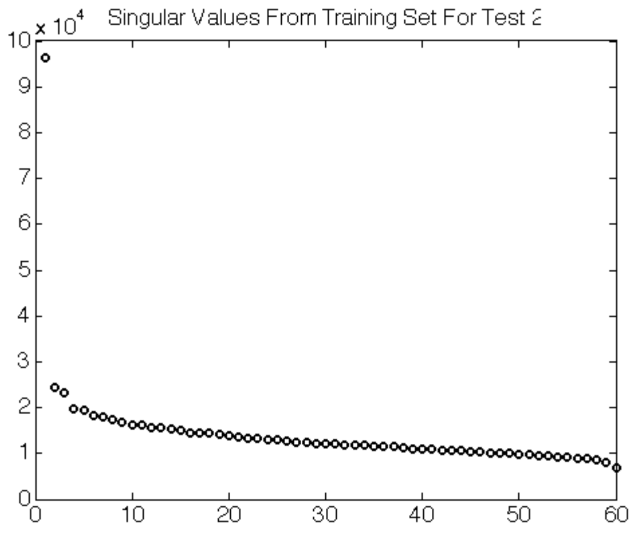

Figure 11: Singular values from the second training set (hip-hop only).

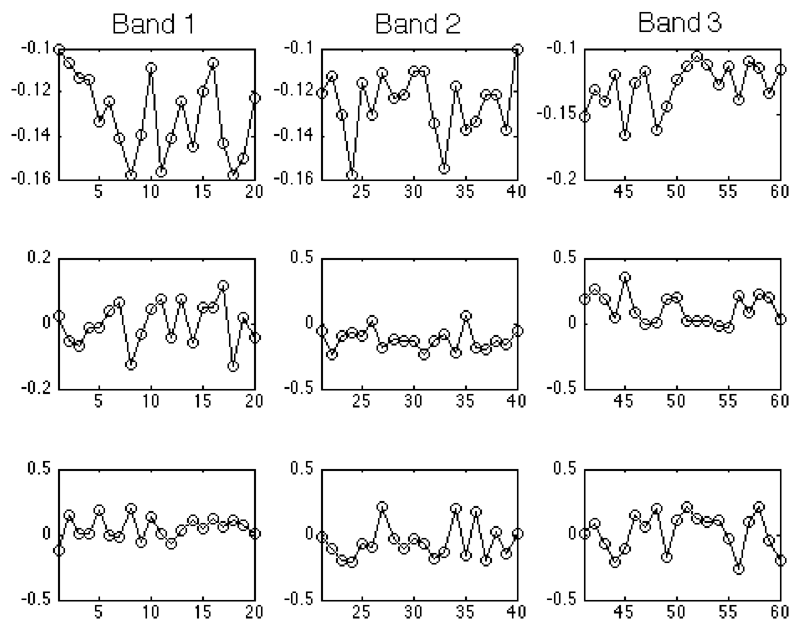

Figure 12: Projection of each second training set song onto the first three SVD modes.

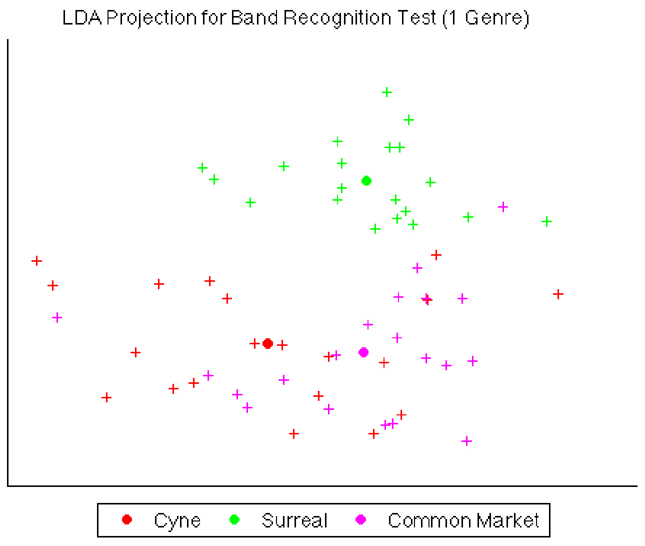

Figure 13: LDA projection for test 2 (hip-hop)

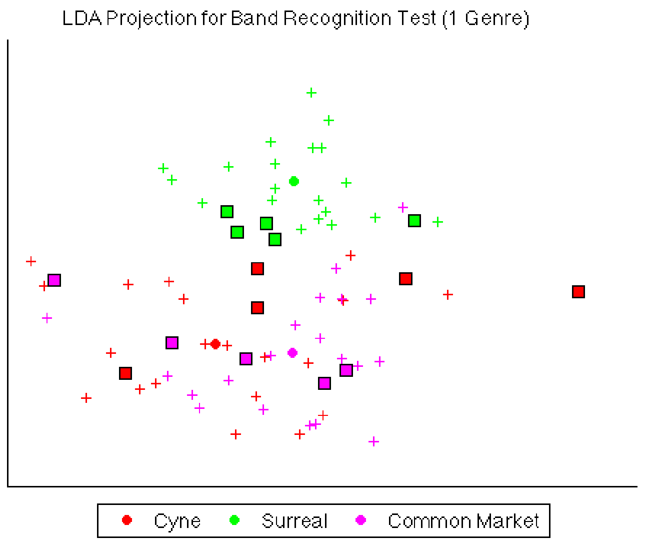

Figure 14: LDA projection for test 2 (hip-hop) with new test songs included.

The result of projecting new songs from each band onto the LDA subspace is shown in Figure 14. Observe that although the new songs were projected in the same vicinity as the other songs from the same band, it is much harder to classify the bands as compared to the first test. This is expected, as drawing distinct lines between bands of the same genre is much more difficult than between different genres. This is especially true between bands 1 and 3, whose training set songs are quite intermingled. Using the algorithm on a single genre is not nearly as accurate, as only 10 of the 15 bands were correctly identified, giving an accuracy rate of only 67%.

4.3 Test 3 - Genre Classification (3 Genres)

For the final test, the goal is to simply classify bands within a broad genre. Numerous bands from the same three genres as the first test (rock, hip-hop, and classical) were used to construct the training set.

The singular values resulting from applying the SVD to the training set are shown in Figure 15. Note that one mode stands out above all others, and none of the others are nearly as strong. To get a sense of which modes are the most significant, each of the songs in the training set was projected onto the first three SVD modes. This projection gives an indication of the strength of each of these modes in a given test song. The result is shown in Figure 16. Note that for rock and classical songs, the first two modes are of the same sign and try to add together. For hip-hop songs, the first two modes are of the opposite sign and try to cancel each other out. This is the same result that was observed in test 1, and it seems that diversifying the training set by adding more bands did not change the dominant modes generated by the SVD.

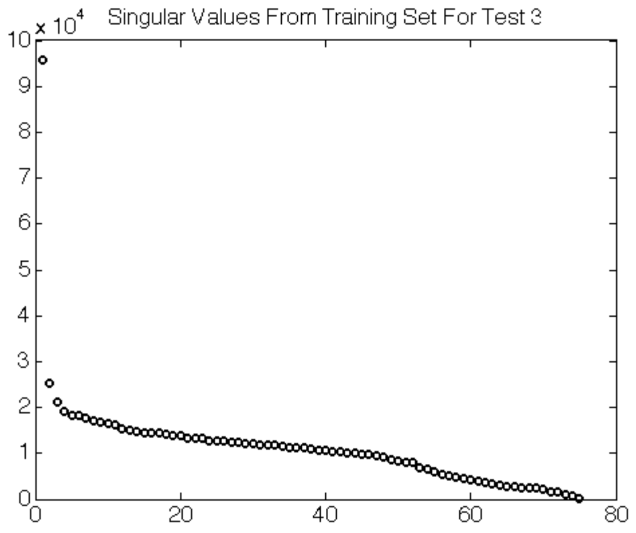

Figure 15: Singular values from the third training set (classical, hip-hop, and rock).

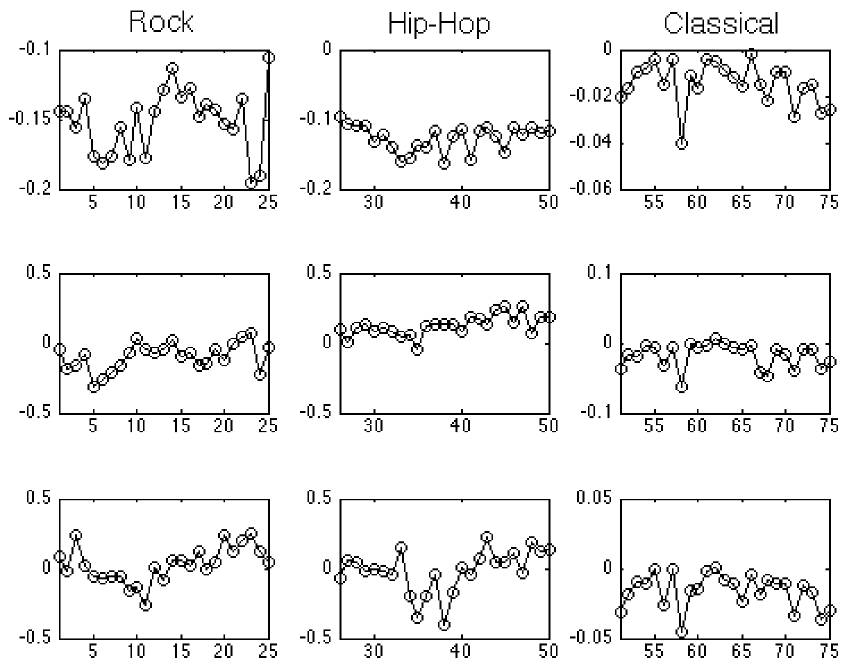

Figure 16: Projection of each third training set song onto the first three SVD modes.

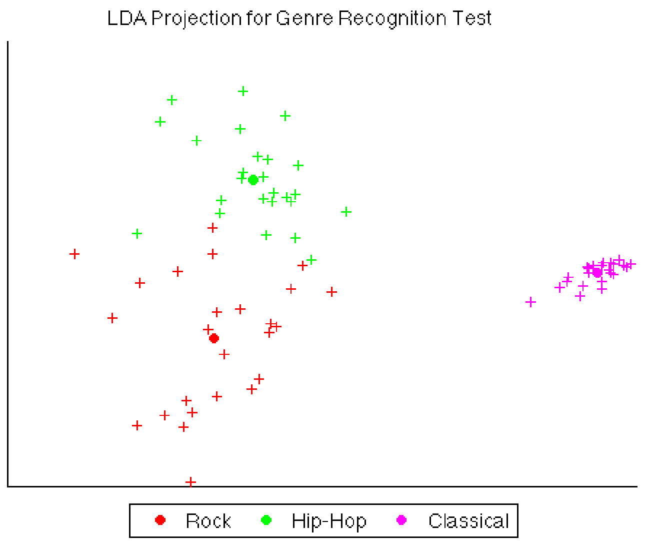

Figure 17: LDA projection for test 3 (classical, hip-hop, and rock).

The LDA projection for the training set songs is shown in Figure 17. Note that there is a bit less separation and clustering within each genre when compared to the first test. This makes sense because more bands were used to comprise the training set, which adds a greater variety of sounds. Despite the increase in scatter, the genres are still quite distinct on the LDA projection plane.

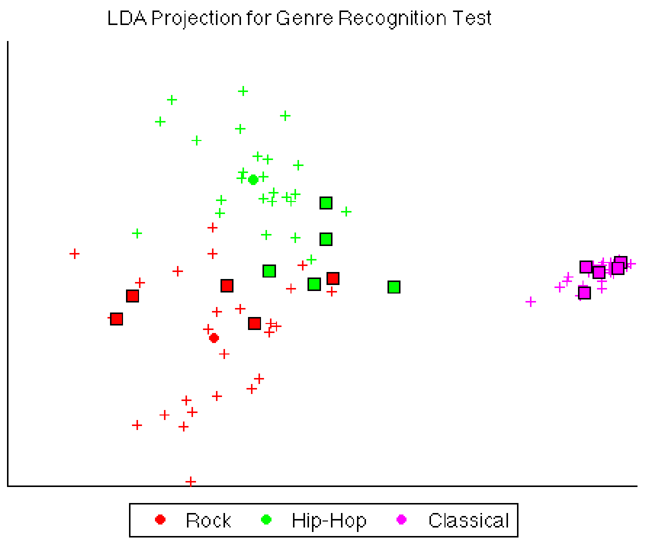

Figure 18: LDA projection for test 3 (classical, hip-hop, and rock) with new test songs included.

The result of projecting new songs onto the LDA subspace is shown in Figure 18. Observe that the new songs (from new bands within each genre) were projected in the same vicinity as the other songs from the same genre. However, due to the increase in diversity of the training set, the algorithm is not quite as effective as in test 1. This time, when classification is by genre, only 12 of the 15 test songs were located closest to their respective genre, giving a success rate of 80% to the third test.

This is still an impressive result, considering that the new bands, which were not included in the training set at all, were almost all correctly classified in terms of genre. It is important to note however, that the test songs should be somewhat similar to the training set songs, given the rather broad nature of some genres.

5 Summary and Conclusions

This assignment combined several powerful techniques from the area of signal processing (Gábor transforms), linear algebra (the SVD), and statistics (LDA) to create an algorithm capable of classifying songs into a specific genre.

To build the algorithm, training sets were constructed using spectrograms of songs from different bands. Once the algorithm was in place, test songs were used to determine its accuracy. For the first test, where bands from three different genres were used, the algorithm proved to be quite effective, as 100% of new test songs from the same bands were correctly identified. For the second test, where a single genre was used, the algorithm was not nearly as effective due to the difficulty of classifying bands who sounds are so similar. In this case, only 67% of the new test songs were correctly identified. For the third tets, where three different genres were used, but the goal was to classify new bands into each genre, the algorithm performed better than test 2, but not as well as test 1, as 80% of the new test songs from new bands were correctly identified.

Lessons learned from this assignment include the importance of building a large, diverse training set. This is especially true for test 2, when it is much harder to classify bands within the same genre due to the similar nature of their sounds. Also, there can be a high degree of variance within even a single genre. Because of this, it is important to either select training set songs that capture a wide range of sounds, or narrow the focus of the training set and classify the genre appropriately (using a single genre for rock songs versus separate genres for classic rock/modern rock, etc).

Overall, the algorithm appeared to be best at classifying new songs from individual bands from different genres (test 1 setup). This is due to the fact that this test setup offers a greater variety of sounds than multiple bands from a single genre (test 2 setup), but not as much variety as multiple bands across multiple genres (test 3 setup). For these types of tests, the similarity of bands from a single genre, and the wide variety of sounds within a single genre, make classification much more difficult.

6 Appendix A - Summary of Matlab Commands Used

Basic Commands

- CLEAR ALL $\rightarrow$ Remove all variables from the workspace.

- CLOSE ALL $\rightarrow$ Close all figure windows.

- CLC $\rightarrow$ Clear all text from the command window.

- LOAD, SAVE $\rightarrow$ Load .MAT file or save Matlab variables to .MAT file

- WAVREAD $\rightarrow$ Read wave file.

Conditional Statements and Logical Operators

- IF ... ELSE ... END $\rightarrow$ Conditional statements which execute provided the logical statement contained within the IF line is true. The optional ELSE line provides a second statement in case the IF line fails.

- FOR ... END $\rightarrow$ Loop which will iterate the number of times specified in the FOR argument.

- SWITCH ... CASE ... END $\rightarrow$ Conditional statements where CASE takes on specific values of variable specified after SWITCH.

Mathematical Functions

- ABS $\rightarrow$ Returns the absolute value of the argument provided.

- DIAG $\rightarrow$ Returns the diagonal elements of an array.

- DSEARCHN $\rightarrow$ Used in the form $k = dsearchn(X, XI)$. Returns the indices of the nearest point in X for each point in XI.

- EIG $\rightarrow$ Returns eigenvalues and eigenvectors of input matrix.

- FFT $\rightarrow$ Fast Fourier transform.

- FFTSHIFT $\rightarrow$ Shifts zero-frequency component to center of spectrum. Necessary for viewing data in the spectral domain.

- INTERP1 $\rightarrow$ Used in the form $y_i = interp1(x, Y, x_i)$, which returns the interpolated values of the function Y at the points xi.

- LENGTH $\rightarrow$ Returns the length of an array.

- LINSPACE $\rightarrow$ Used in the form $linspace(a, b, n)$, which generates a linearly spaced vector of length n, with initial value a and final value b.

- MEAN $\rightarrow$ Returns the mean of an array. For a matrix, returns the mean along a dimension.

- NORM $\rightarrow$ Returns the norm of an array. The defualt is the 2-norm.

- SIZE $\rightarrow$ Returns the size of an array.

- SORT $\rightarrow$ Sort elements of an array.

- STRCMPI → Case insensitive string comparison. Returns a 1 if strings match, 0 otherwise.
- SUM → Summation of all entries in an array.
- SVD → Returns the singular value decomposition of a matrix. Second input argument of '0' returns reduced SVD.
- ZEROS → Used in the form $zeros(m, n, p)$, which generates an m by n by p array of zeros.

Plotting Functions

- FIGURE → Create a new figure window.
- PCOLOR → Pseudocolor checkerboard plot given a matrix as input.
- SUBPLOT → Used in the form $subplot(m, n, p)$, which generates the p^{th} subplot in an $m \times n$ array of subplots.
- TITLE, XLABEL, YLABEL → String for axes title and labels.
- XLIM, YLIM → Specify axes limits in the form $xlim([xmin\ xmax])$.

7 Appendix B - Matlab code

7.1 Main Function

```
clear all; close all; clc

% Test 1 - Band recognition (3 genres)
% Test 2 - Band recognition (1 genre)
% Test 3 - Genre recognition
test = 3;
feature = 5; % Number of features to extract

switch test
    case 1
        load museData, load surrealData, load mendData
        bands = {'Muse (Rock)';'Surreal (Hip-Hop)';'Mendelssohn (Classical)'};
        ns = length(muse(1,:)); % Number of songs from each band
        [result,W,U,S,V,evals] = song_trainer(muse,surreal,mend,5);

    case 2
        load cyneData, load surrealData, load cmData
        bands = {'Cyne';'Surreal';'Common Market'};
        ns = length(cyne(1,:)); % Number of songs from each band
        [result,W,U,S,V,evals] = song_trainer(cyne,surreal,cm,5);
    case 3
        load rockData, load hiphopData, load classicalData
        bands = {'Rock';'Hip-Hop';'Classical'};
        ns = length(rock(1,:)); % Number of songs from each band
        [result,W,U,S,V,evals] = song_trainer(rock,hiphop,classical,5);
end

vband1 = result(:,1:ns);
vband2 = result(:,ns+1:2*ns);
vband3 = result(:,2*ns+1:3*ns);

figure(1), hold on
plot(mean(vband1(1,:)),mean(vband1(2,:)),'ro','MarkerFaceColor','r')
plot(mean(vband2(1,:)),mean(vband2(2,:)),'go','MarkerFaceColor','g')
plot(mean(vband3(1,:)),mean(vband3(2,:)),'mo','MarkerFaceColor','m')
legend(bands,'Orientation','Horizontal','Location','SouthOutside')
plot(vband1(1,:),vband1(2,:),'r+')
plot(vband2(1,:),vband2(2,:),'g+')
plot(vband3(1,:),vband3(2,:),'m+')
switch test
    case 1
        titlestr = 'LDA Projection for Band Recognition Test (3 Genres)';
    case 2
        titlestr = 'LDA Projection for Band Recognition Test (1 Genre)';
    case 3
```

```
        titlestr = 'LDA Projection for Genre Recognition Test';
end
title(titlestr,'FontSize',14,'FontWeight','Bold')
set(gca,'FontSize',14,'Xtick',[],'Ytick',[])

% Test new songs
switch test
    case 1
        load testData1
        answers = vertcat(repmat({'Muse (Rock)'},5,1),...
            repmat({'Surreal (Hip-Hop)'},5,1),...
            repmat({'Mendelssohn (Classical)'},5,1));% New test songs
        testSVD = U'*testData1;
    case 2
        load testData2
        answers = vertcat(repmat({'Cyne'},5,1),...
            repmat({'Surreal'},5,1),...
            repmat({'Common Market'},5,1));% New test songs
        testSVD = U'*testData2;
    case 3
        load testData3
        answers = vertcat(repmat({'Rock'},5,1),...
            repmat({'Hip-Hop'},5,1),...
            repmat({'Classical'},5,1));% New test songs
        testSVD = U'*testData3;
end

pval = W'*testSVD; % LDA projection
colors = 'rrrrrgggggmmmmm';
for j = 1:15
    plot(pval(1,j),pval(2,j),[colors(j) 's'],'MarkerFaceColor',colors(j),...
        'MarkerEdgeColor','k','MarkerSize',8)
end

% Identify new songs
dsearchSet = [ mean(vband1,2) mean(vband2,2) mean(vband3,2) ]';
newsongs = pval';

% Find nearest song from training set
k = dsearchn(dsearchSet,newsongs);
testResults = cell(3,1);
for i = 1:length(k)
    switch k(i)
        case 1
            testResults{i,1} = bands{1};
        case 2
            testResults{i,1} = bands{2};
        case 3
            testResults{i,1} = bands{3};
    end
end

if all(strcmpi(answers,testResults))
    fprintf('Perfect Match!\n')
    fprintf('Success Rate = 100%%\n')
else
    fprintf('Not all songs correctly matched...\n')
    logical = strcmpi(answers,testResults);
    fprintf('Correct answers:\n\n')
    disp(answers(logical))
    fprintf('Wrong answers:\n\n')
    disp(answers(~logical))
    prc = sum(logical)/length(logical) * 100;
    fprintf('Success Rate = %2.0f%%\n',prc)
end

%Extra plots
figure(2)
plot(diag(S),'ko','LineWidth',2)
titlestr = [ 'Singular Values From Training Set For Test ' num2str(test) ];
title(titlestr,'FontSize',16), set(gca,'FontSize',16)

figure(3)
ns = size(V,1)/3;
for j=1:3
    subplot(3,3,3*j-2), plot(1:ns,V(1:ns,j),'ko-'), xlim([1 ns])
    subplot(3,3,3*j-1), plot(ns+1:2*ns,V(ns+1:2*ns,j),'ko-'), xlim([ns+1 2*ns])
    subplot(3,3,3*j), plot(2*ns+1:3*ns,V(2*ns+1:3*ns,j),'ko-'), xlim([2*ns+1 3*ns])
end
if test == 1 || test == 3
    subplot(3,3,1), title('Rock','FontSize',16)
    subplot(3,3,2), title('Hip-Hop','FontSize',16)
    subplot(3,3,3), title('Classical','FontSize',16)
else
    subplot(3,3,1), title('Band 1','FontSize',16)
    subplot(3,3,2), title('Band 2','FontSize',16)
    subplot(3,3,3), title('Band 3','FontSize',16)
end
```

7.2 Training Function

```
function [result,W,U,S,V,evals]=song_trainer(band1_0,band2_0,band3_0,feature)

ns = length(band1_0(1,:)); % Number of songs from each band
[U,S,V] = svd([band1_0,band2_0,band3_0],0); % reduced SVD

songs = S*V';
U = U(:,1:feature);
band1 = songs(1:feature,1:ns);
band2 = songs(1:feature,ns+1:2*ns);
band3 = songs(1:feature,2*ns+1:3*ns);
m1 = mean(band1,2); m2 = mean(band2,2); m3 = mean(band3,2);

Sw=0; % within class variances
for i=1:ns
    Sw = Sw + (band1(:,i)-m1)*(band1(:,i)-m1)';
    Sw = Sw + (band2(:,i)-m2)*(band2(:,i)-m2)';
    Sw = Sw + (band3(:,i)-m3)*(band3(:,i)-m3)';
end

m = sum([m1 m2 m3],2)/ns; % Mean of class means
% between class variance
Sb = (m1-m)*(m1-m)' + (m2-m)*(m2-m)' + (m3-m)*(m3-m)';

[V2,D] = eig(Sb,Sw); % linear discriminant analysis
[evals,I] = sort(diag(D));

% Want 2 largest eigenvalues and eigenvectors
ind1 = I(end); ind2 = I(end-1);
w1 = V2(:,ind1); w1 = w1/norm(w1,2);
w2 = V2(:,ind2); w2 = w2/norm(w2,2);

W = [w1 w2];
vband1 = W'*band1; vband2 = W'*band2; vband3 = W'*band3;
result = [vband1, vband2, vband3];
```

7.3 Import Training Set Clips

```
clear all; close all; clc

% Create spectrograms for test 1 (band classification - 3 genres)
num_sample = 20;
data_pts = 100000;
surreal = zeros(data_pts,num_sample);
for i = 1:num_sample
    fprintf('Loading sample %s\n',num2str(i))
    str = [ 'surreal' num2str(i) ];
    y = wavread(str);
    spec = createSpectrogram(y,0);
    surreal(:,i) = spec(:);
end
save surrealData surreal
```

7.4 Import Test Set Clips

```
clear all; close all; clc

% Create spectrograms for test 1 (band classification - 3 genres)
num_sample = 5;
data_pts = 100000;
testData1 = zeros(data_pts,num_sample*3);
% Test band 1
for i = 1:num_sample
    fprintf('Loading sample %s from band 1\n',num2str(i))
    str = [ 'muse_TEST' num2str(i) ];
    y = wavread(str); spec = createSpectrogram(y,0);
    testData1(:,i) = spec(:);
end
% Test band 2
for i = 1:num_sample
    fprintf('Loading sample %s from band 2\n',num2str(i))
    str = [ 'surreal_TEST' num2str(i) ];
    y = wavread(str); spec = createSpectrogram(y,0);
    testData1(:,num_sample+i) = spec(:);
end
% Test band 3
for i = 1:num_sample
    fprintf('Loading sample %s from band 3\n',num2str(i))
    str = [ 'mend_TEST' num2str(i) ];
    y = wavread(str); spec = createSpectrogram(y,0);
    testData1(:,2*num_sample+i) = spec(:);
end
save testData1 testData1
```

7.5 Spectrogram Generation Function

```
function spec_out = createSpectrogram(y,plotflag)

stime = 5;
% Create time signal
y = y(:,1);
y = y';
Fs=length(y)/stime;
t = (1:length(y))/Fs;

if plotflag
    figure(1)
    set(gcf,'Units','Normalized','Position',[0.2 0.2 0.6 0.7])
    subplot(2,1,1), plot(t,y);
    xlabel('Time [sec]','FontSize',14); ylabel('Amplitude','FontSize',14);
    set(gca,'FontSize',16)
    ylim([-2 2])
end

% Create frequency signal
L = t(end);
n = length(t);
t2 = linspace(0,L,n+1); t = t2(1:n);
k = (2*pi/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k); ksHz = ks/(2*pi);
yt = fft(y,n);

if plotflag
    subplot(2,1,2)
    plot(ksHz,abs(fftshift(yt)))
    xlabel('Frequency [Hz]','Fontsize',14);
    ylabel('Amplitude','Fontsize',14);
    set(gca,'FontSize',16)
end

% Create Gabor Filter
if plotflag
    figure(2)
    set(gcf,'Units','Normalized','Position',[0.2 0.2 0.6 0.7])
end
width = 40; % Width of Gabor filter (larger width = tighter filter)
slices = 100;
slide = linspace(0,L,slices);
cutoff = 5000; % Limit in Hertz Range
sample = 1000; % Sample rate (5Hz)
spec_rough = zeros(length(slide),sample);
ksHzRed = ksHz(ksHz >=0 & ksHz <= cutoff);
xi = linspace(ksHzRed(1),ksHzRed(end),sample);
```

```
for j = 1:length(slide)
    f = exp(-width*(t-slide(j)).^2);  % Create Gaussian filter
    yf = f.*y;
    yft = fft(yf,n);
    if plotflag
        subplot(3,1,1), plot(t,y,'b',t,f,'m')
        xlabel('Time [sec]','Fontsize',14);
        ylabel('Amplitude','Fontsize',14);
        set(gca,'FontSize',16)
        subplot(3,1,2), plot(t,yf,'b')
        ylim([-1 1])
        xlabel('Time [sec]','Fontsize',14);
        ylabel('Amplitude','Fontsize',14);
        title('Filtered Time Signal','Fontsize',16,'FontWeight','Bold');
        set(gca,'FontSize',16)
        subplot(3,1,3), plot(ksHz,abs(fftshift(yft))./max(abs(fftshift(yft))),'b')
        xlabel('Frequency [Hz]','Fontsize',14);
        ylabel('Amplitude','Fontsize',14);
        title('Frequency Spectrum, Hz','Fontsize',16,'FontWeight','Bold');
        set(gca,'FontSize',16)
        drawnow
        pause(0.01)
    end
    spec = abs(fftshift(yft));
    spec = spec(ksHz >= 0 & ksHz <= cutoff); % Keep between 0 & 10,000 Hz
    % Reduce number of frequencies for pcolor plot
    yi = interp1(ksHzRed,spec,xi);
    spec_rough(j,:) = yi;
end

spec_out = spec_rough';
% Plot spectrogram
if plotflag
    figure(3)
    set(gcf,'Units','Normalized','Position',[0.1 0.2 0.7 0.8])
    pcolor(slide,xi,spec_out), shading interp
    ylim([0 5000])
    drawnow
    pause(0.01)
    xlabel('Time [sec]','FontSize',14)
    ylabel('Frequency (Hz)','FontSize',14)
    set(gca,'FontSize',16)
end
```