

AMATH 582 Homework 4

Shannon Dow

March, 2020

Abstract

This paper will demonstrate the usefulness and power of the Singular Value Decomposition (SVD) as a computational tool, first in creating low rank approximations of and analyzing faces, and then in classification. The first part of this assignment uses the SVD to compare low rank approximations and dominant modes of data sets of cropped and un-cropped faces. The second part uses the SVD as a part of training a program to recognize song data. To classify the music, Linear Discriminant Analysis (LDA) tools will be used to find an optimal subspace to separate the data classes.

1 Introduction and Overview

1.1 Part 1: Yale Faces

For this part, there are two datasets: a large one with many cropped pictures of people with a serious expression, and a smaller one of people's entire head with different expressions. In performing a Singular Value Decomposition (SVD) on both sets of images, I will determine the dominant modes needed to reconstruct the data, and form low rank approximations to the images.

1.2 Part 2: Music Classification

The goal of this section is to train a computer program to distinguish between different types of music.

Test 1 I will build a training set using 5 second clips of Mozart (Classical), The Eagles (Rock), and Snoop Dogg (Rap) and see how well the computer can identify other clips from the same artists

Test 2 For this test, I will repeat but build a training set with 3 bands (The Eagles, Fleetwood Mac, and Journey) from the same genre (Rock) and see if the program can identify bands.

Test 3 For this, I will use multiple bands from the same three genres as test 1 and evaluate the computer's ability to identify genre. To achieve these results, I will be using higher level MATLAB implementation to perform Linear Discriminant Analysis.

2 Theoretical Background

2.1 Part 1: Yale Faces

For this section, we are using the SVD to identify correlations between large data sets of people's faces. To create the A matrix to perform the SVD, I first reshape the image data into columns of numbers and add them into a matrix. The reason SVD is such a powerful computational tool for this overdetermined system is because all matrices have a SVD. Once we have the reduced SVD, it can be interpreted in the following way:

- U The columns of U represent the modes, or a set of characteristics of the face. The first column of U , corresponding to the largest singular value, is the most dominant feature. These are also called eigenfaces.
- Σ This is the singular values. The number of relevant (comparatively large) singular values tells you how many modes it takes to represent the images

- **V** The V values give the coefficients of the eigenfaces. They tell you how much of each eigenface you need to get the best approximation.

The main reason for using the SVD here is to look at low rank approximations as well as relevant modes. To obtain low rank approximations:

$$\text{Rank } n \text{ approximation} = \sum_{j=1}^n V_j * \Sigma_{jj} * U_j$$

2.2 Part 2: Music Classification

In order to perform supervised machine learning, the first goal is to take the data and make a training set, where the labels of the different categories are known. LDA finds a subspace for optimal separation of the data by identifying the projection that maximizes the distance between the different categories. Then, for linear analysis, it will create a line that provides the best separation of the different classes. If a quadratic analysis, it would choose the best quadratic to separate the data.

For this project, since we are working with 5 second segments of song data, we want to have the frequency and time information. For this we will need to find the spectrogram. To get the spectrogram, we will use a Gaussian Gabor transform, or a short time Fourier transform. This slides a filter along the music signal in time and gets localized data there about the frequency.

Once we have the spectrogram data from each song snippet, before we perform LDA, we want to take the reduced Singular Value Decomposition (SVD). To decrease computational time, we only want to keep the relevant modes, or columns of V, so it is helpful to look at the dominant, or largest singular values corresponding to those columns. Then we can use this truncated V matrix as the training data for our LDA. I will also use a similarly shaped truncated V from the test data when performing LDA.

3 Algorithm Implementation and Development

3.1 Part 1: Yale Faces

Before performing the analysis on the data using the SVD, it is first key to have the data in a usable format. The first section of the code takes the image files out of the folders of data. Then, each image is converted to a double data type and reshaped into a column vector. The result is then added to the matrix A.

The dimensions of A are: (size of image(height x width) x number of images) .

The SVD is then obtained using MATLAB's reduced SVD function. To analyze the modes, you must reshape the columns of U back into images. Since U is a unitary matrix, the columns of U are orthonormal, so they are scaled to have the norm 1. In order to view the image, these columns need to be rescaled back into 0-255 scale. This is done by subtracting off the minimum and multiplying by $\frac{255}{max}$ or each column.

3.2 Part 2: Music Classification

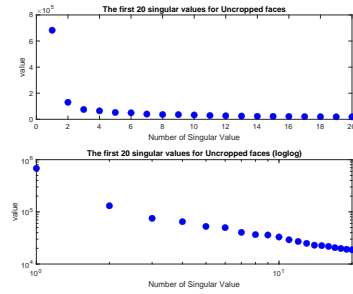
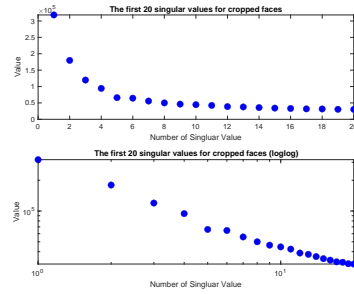
Again, the first part of this assignment is to get the data into a usable format. To do this, we must first get .wav files of the music samples we want to use. Then as the .wav files are very large, it is useful to re-sample the music data so that we decrease computational time. The goal is to form a matrix where each column is a vectorized spectrogram of a 5-second snippet of a song, one for the training data, and one for the test data, created the same way, (See Algorithm 1).

Once we have the two matrices of test and training data, we take the reduced SVD. By analyzing the number of relevant singular values, we know how many columns of V we need. Next, using the MATLAB function `classify`, we can perform LDA. `classify` takes in 3 main arguments, the test and training data, a vector containing the labels for the training data.

Once we have the computer's classifications of the test data, we can compare them to actual known labels and see how accurate the program is.

Algorithm 1: Updating the Matrix of Vectorized Song Data

```
Let count be the number of the column you want to start updating A on
for The number of desired additional columns do
    Read in a 5 second sample from the data
    Re-sample the 5 second sample
    Find the spectrogram using the Gabor Window
    Reshape the spectrogram into a vector
    Add that vector to A
    Update count
end for
```

Figure 1 Singular Values (Cropped)**Figure 2** Singular Values (Uncropped)

4 Computational Results

4.1 Part 1: Yale Faces

Figures 1 through 10 show the main differences between the SVD analysis on the cropped and un-cropped Yale faces. Comparing Figures 1 and 2, there are more large singular values for the cropped images. This is because it is easier for the SVD to pick up correlations in the images when everything is in roughly the same place. This can also be seen in later figures, that low rank approximations are more descriptive for the cropped images, and you can clearly different sets of features in the modes.

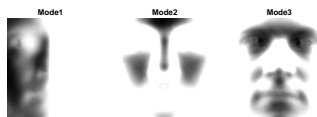
Figure 3 First 3 Modes All Faces (Cropped)**Figure 4** First 3 Modes All Faces (Uncropped)

Figure 5 First 3 Modes One Face (Cropped)

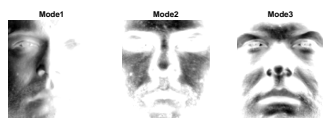


Figure 6 First 3 Modes One Face (Uncropped)



Figure 7 Rank Approximations One Face (Cropped)



Figure 8 Rank Approximations One Face (Uncropped)



Figure 9 Rank Approximations All Faces (Cropped)



Figure 10 Rank Approximations All Faces (Uncropped)



4.2 Part 2: Music Classification

4.2.1 Test 1: 3 Different Bands from 3 different genres

For this test, I used songs from Mozart (Classical), The Eagles(Rock), and Snoop Dogg(Rap) to create a training set. Then used other song data from the same bands to create a test set. This test has 81.6%accuracy.

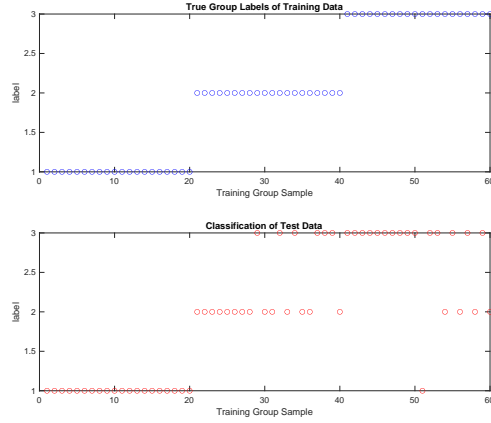


Figure 1: Mozart(1), The Eagles(2), and Snoop Dogg(3)

4.2.2 Test 2: 3 different Bands from the same genre

For this test, I used The Eagles, Journey, and Fleetwood Mac as soft rock bands from the 60s/70s. This test has 65%accuracy.

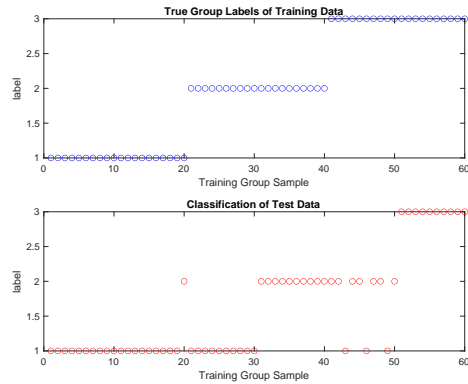


Figure 2: The Eagles(1), Journey(2), and Fleetwood Mac(3)

4.2.3 Test 3: Multiple Bands from three different genres

For this test, using the same three genres as above, I chose multiple artists from each.

- Classical Mozart, Bach, Beethoven, and Chopin
- Rock The Eagles, Journey, and Fleetwood Mac
- Rap Snoop Dogg, Eminem, 50 Cent

This test has has 88.33% accuracy.

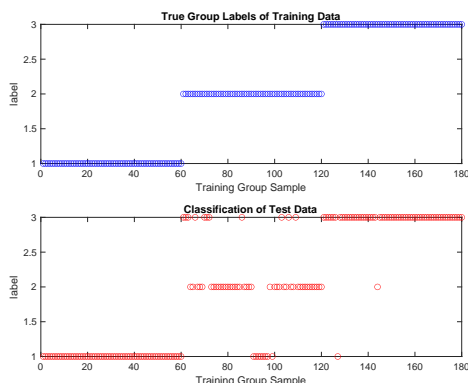


Figure 3: Classical(1), Rock(2), and Rap(3)

5 Summary and Conclusions

5.1 Part 1: Yale Faces

By analyzing the results from the SVD analysis, you can see that the SVD works better when the data is set up in a convenient way. For this, when all the faces are aligned and making the same facial expression, it is easier to find dominant shared characteristics. It also makes our low-rank approximations more accurate.

5.2 Part 2: Music Classification

From these classification tests, the most accurate test was the final one that distinguished between genres. One explanation for this is that I used a larger training and test data set. Since it had more, different information, it was better able to classify the test data. Additionally, I did choose relatively different sounding genres. The most often mis-labeled was the soft rock which is in the middle of classical and rap. As expected, it was the most difficult for the program to distinguish between artists in the same genre.

Appendix A MATLAB Functions

- `dir` Gets the directory

- `imread` Takes data and turns it into the uint8 image datatype
- `double` Converts to the double data type
- `reshape(a,x,y)` The reshape function takes data (a) and turns it into a matrix of the size (x by y).
- `imshow` This displays an image of type uint8
- `classify(Test, Train, Label)` Classify performs discriminant analysis on data once it receives training data with matching labels. It then classifies the test data into the categories listed in Label based on the training data.

Appendix B MATLAB Code

B.1 GITHUB LINK

<https://github.com/shannondow/AMATH-582-Homeworks-2020>

```

1 %% Cropped Faces:
2
3 D = '/Users/shannondow/MATLAB/projects/582HW4/yalefaces_cropped/CroppedYale';
4 S = dir(fullfile(D, '*'));
5 N = setdiff({S([S.isdir]).name}, {}); % list of subfolders of D
6 A = [];
7 count = 1;
8 for ii = 1:numel(N)
9     T = dir(fullfile(D, N{ii}, '*.pgm')); % improve by specifying the file
        extension.
10    C = {T(~[T.isdir]).name}; % files in subfolder.
11    for jj = 1:numel(C)
12        F = fullfile(D, N{ii}, C{jj});
13        a = imread((T{jj}).name);
14        aa = double(a);
15        [m,n] = size(aa);
16        x = reshape(aa, m*n, 1);
17        x = x - mean(x);
18        A(:, count) = x;
19        count = count + 1;
20    end
21 end
22 %Take The SVD
23 [U, Sig, V] = svd(A, 0);
24 % Plot the Singular Values:
25
26 figure(1)
27 for i = 1:20
28     subplot(2,1,1)
29     plot(i, Sig(i,i), 'b.', 'MarkerSize', 30)
30     hold on;
31     title('The first 20 singular values for cropped faces')
32     xlabel('Number of Singular Value');
33     ylabel('Value')
34     subplot(2,1,2)
35     loglog(i, Sig(i,i), 'b.', 'MarkerSize', 30)
36     hold on;
37     title('The first 20 singular values for cropped faces (loglog)')

```

```

38     xlabel('Number of Singluar Value');
39     ylabel('Value')
40 end
41
42 % Low Rank Approximations:
43 figure(2)
44 for rank = 1:5
45     Approx = U(:,1:rank)*Sig(1:rank,1:rank)*V(:,1:rank)';
46     App = reshape(Approx(:,1), m,n);
47     subplot(1,5,rank)
48     imshow(uint8(App))
49     nam = strcat('Rank',num2str(rank));
50     title(nam);
51 end
52 % Modes (Columns of U)
53 % Rescale to 255
54 for j = 1:length(U(1,:))
55     minval = min(U(:,j));
56     maxval = max(U(:,j));
57     U(:,j) = U(:,j) - minval;
58     U(:,j) = U(:,j)*(255/maxval);
59 end
60 figure(3)
61 for mode = 1:3
62     yy= reshape(U(:,mode), m,n);
63     subplot(1,3,mode)
64     imshow(uint8(yy))
65     nam = strcat('Mode',num2str(mode));
66     title(nam);
67 end
68
69 %%
70 %% Just Subject 1:
71 [U1,Sig1,V1] = svd(A(:,1:64),0);
72
73 %Plot Singular Values
74 figure(4)
75 for i = 1:20
76     subplot(2,1,1)
77     plot(i,Sig1(i,i),'b.','MarkerSize',30)
78     hold on;
79     title('Singular Values Subject 1 Uncropped')
80     xlabel('Number of Singluar Value');
81     ylabel('Value')
82     subplot(2,1,2)
83     loglog(i,Sig1(i,i),'b.','MarkerSize',30)
84     hold on;
85     title('Singular Values Subject 1 Uncropped (loglog)')
86     xlabel('Number of Singluar Value');
87     ylabel('Value')
88 end
89 %Low Rank Approximations:
90 figure(5)
91 for rank = 1:5

```



```

92     Approx = U1(:,1:rank)*Sig1(1:rank,1:rank)*V1(:,1:rank)';
93     App = reshape(Approx(:,1), m,n);
94     subplot(1,5,rank)
95     imshow(uint8(App))
96     nam = strcat('Rank',num2str(rank));
97     title(nam);
98 end
99
100 % Modes (Columns of U)
101 for j = 1:length(U1(1,:))
102     minval = min(U1(:,j));
103     maxval = max(U1(:,j));
104     U1(:,j) = U1(:,j) - minval;
105     U1(:,j) = U1(:,j)*(255/maxval);
106 end
107 figure(6)
108 for mode = 1:3
109     yy= reshape(U1(:,mode), m,n);
110     subplot(1,3,mode)
111     imshow(uint8(yy))
112     nam = strcat('Mode',num2str(mode));
113     title(nam);
114 end
115 %% Uncropped Faces
116 %untar('yalefaces_uncropped.tar')
117 D = '/Users/shannondow/MATLAB/projects/582HW4/yalefaces';
118 S = dir(fullfile(D,'*'));
119 A = [];
120 count = 1;
121 for ii = 3:numel(S)
122     a = imread(S(ii).name);
123     aa = double(a);
124     [m,n]=size(aa);
125     x=reshape(aa,m*n,1);
126     %x = x-mean(x);
127     A(:,count) = x;
128     count = count+ 1;
129 end
130
131 %Take The SVD
132 [U,Sig,V] = svd(A,0);
133 % Plot the Singular Values:
134
135 figure(1)
136 for i = 1:20
137     subplot(2,1,1)
138     plot(i,Sig(i,i),'b.','MarkerSize',30)
139     hold on;
140     title('The first 20 singular values for Uncropped faces')
141     xlabel('Number of Singular Value')
142     ylabel('value')
143     subplot(2,1,2)
144     loglog(i,Sig(i,i),'b.','MarkerSize',30)
145     hold on;

```

```

146     title('The first 20 singular values for Uncropped faces (loglog)')
147     xlabel('Number of Singular Value')
148     ylabel('value')
149 end
150
151 % Low Rank Approximations:
152 figure(2)
153 for rank = 1:5
154     Approx = U(:,1:rank)*Sig(1:rank,1:rank)*V(:,1:rank)';
155     App = reshape(Approx(:,1), m,n);
156     subplot(1,5,rank)
157     imshow(uint8(App))
158     nam = strcat('Rank',num2str(rank));
159     title(nam);
160 end
161 % Modes (Columns of U)
162 for j = 1:length(U(1,:))
163     minval = min(U(:,j));
164     maxval = max(U(:,j));
165     U(:,j) = U(:,j) - minval;
166     U(:,j) = U(:,j)*(255/maxval);
167 end
168 figure(3)
169 for mode = 1:3
170     yy= reshape(U(:,mode), m,n);
171     subplot(1,3,mode)
172     imshow(uint8(yy))
173     nam = strcat('Mode',num2str(mode));
174     title(nam);
175 end
176
177 %% Just Subject 1:
178 [U1,Sig1,V1] = svd(A(:,1:11),0);
179
180 %Plot Singular Values
181 figure(4)
182 for i = 1:11
183     subplot(2,1,1)
184     plot(i,Sig1(i,i),'b.','MarkerSize',30)
185     hold on;
186     title('Singular Values Subject 1 Uncropped')
187     xlabel('Number of Singular Value')
188     ylabel('value')
189     subplot(2,1,2)
190     loglog(i,Sig1(i,i),'b.','MarkerSize',30)
191     hold on;
192     title('Singular Values Subject 1 Uncropped (loglog)')
193     xlabel('Number of Singular Value')
194     ylabel('value')
195 end
196 %Low Rank Approximations:
197 figure(5)
198 for rank = 1:5
199     Approx = U1(:,1:rank)*Sig1(1:rank,1:rank)*V1(:,1:rank)';

```

```

200     App = reshape(Approx(:,1), m,n);
201     subplot(1,5,rank)
202     imshow(uint8(App))
203     nam = strcat('Rank',num2str(rank));
204     title(nam);
205 end
206
207 % Modes (Columns of U)
208 for j = 1:length(U1(1,:))
209     minval = min(U1(:,j));
210     maxval = max(U1(:,j));
211     U1(:,j) = U1(:,j) - minval;
212     U1(:,j) = U1(:,j)*(255/maxval);
213 end
214 figure(6)
215 for mode = 1:3
216     yy= reshape(U1(:,mode), m,n);
217     subplot(1,3,mode)
218     imshow(uint8(yy))
219     nam = strcat('Mode',num2str(mode));
220     title(nam);
221 end
222 %% Part 2: Test 1-
223 %Create Training Set
224 A = [];
225 count = 1;
226 [count, A] = updateA('classical.wav',count,A,30,20);
227 [count, A] = updateA('EaglesS1.wav',count,A,30,10);
228 [count, A] = updateA('EaglesS2.wav',count,A,30,10);
229 [count, A] = updateA('Snoop1.wav',count,A,30,10);
230 [count, A] = updateA('Snoop2.wav',count,A,30,10);
231 %A = A'
232
233 [UA,SA,VA] = svd(A,0);
234 VA = VA(:,1:10);
235 %Create Test Data
236 B = [];
237 count = 1;
238 [count, B] = updateA('classical.wav', count, B, 200,20);
239 [count, B] = updateA('EaglesS3.wav', count, B, 30,10);
240 [count, B] = updateA('EaglesS4.wav', count, B, 30,10);
241 [count, B] = updateA('Snoop2.wav', count, B, 200,10);
242 [count, B] = updateA('Snoop3.wav', count, B, 30,10);
243 %B = B';
244
245 [UB,SB,VB] = svd(B,0);
246 VB = VB(:,1:10);
247 % Make the group names
248 group = [];
249 for i = 1:20
250     group(i) = 1;
251 end
252 for i = 21:40
253     group(i) = 2;

```

```

254 end
255 for i = 41:60
256     group(i) = 3;
257 end
258 group = group';
259
260 %%
261 [class, err, POSTERIOR, logp, coeff] = classify(VB, VA, group, 'linear');
262
263 VA = VA';
264 classic = [];
265 eagles = [];
266 snoop = [];
267 for i = 1:60
268     if (0<i)&& (i<21)
269         classic = [classic; norm(VA(:, i))];
270     elseif (21 <= i)&&(i < 41)
271         eagles = [eagles; norm(VA(:, i))];
272     elseif (41 <= i) && (i<61)
273         snoop = [snoop; norm(VA(:, i))];
274     end
275 end
276 all = [classic; eagles; snoop];
277
278 %%
279 figure(4)
280 subplot(2,1,1)
281 plot(group, 'bo')
282 title('True Group Labels of Training Data')
283 xlabel('Training Group Sample')
284 ylabel('label')
285 subplot(2,1,2)
286 plot(class, 'ro')
287 title('Classification of Test Data')
288 xlabel('Training Group Sample')
289 ylabel('label')
290
291 %%
292 %calculate error
293 corr = 0;
294 for i = 1:length(group)
295     if group(i) == class(i)
296         corr = corr + 1;
297     end
298 end
299 error = (length(group)-corr)/(length(group))*100;
300 disp(error);
301
302 %%
303 figure(5)
304 plot(classic, 'bo')
305 hold on
306 plot(eagles, 'ro')
307 hold on

```

```

308 plot(snoop, 'go')
309 hold on
310 %% TEST 2-
311
312 A = [];
313 count = 1;
314 [count, A] = updateA('EaglesS1.wav', count, A, 30, 10);
315 [count, A] = updateA('EaglesS2.wav', count, A, 30, 10);
316 [count, A] = updateA('Journey1.wav', count, A, 30, 10);
317 [count, A] = updateA('Journey2.wav', count, A, 30, 10);
318 [count, A] = updateA('fleetwood1.wav', count, A, 30, 10);
319 [count, A] = updateA('fleetwood2.wav', count, A, 30, 10);
320 %A = A'
321
322 [UA, SA, VA] = svd(A, 0);
323 VA = VA(:, 1:10);
324 %Create Test Data
325 B = [];
326 count = 1;
327 [count, B] = updateA('EaglesS3.wav', count, A, 30, 10);
328 [count, B] = updateA('EaglesS4.wav', count, A, 30, 10);
329 [count, B] = updateA('Journey3.wav', count, A, 30, 10);
330 [count, B] = updateA('Journey4.wav', count, A, 30, 10);
331 [count, B] = updateA('fleetwood3.wav', count, A, 30, 10);
332 [count, B] = updateA('fleetwood4.wav', count, A, 30, 10);
333 %B = B';
334
335 [UB, SB, VB] = svd(B, 0);
336 VB = VB(:, 1:10);
337 % Make the group names
338 group = [];
339 for i = 1:20
340     group(i) = 1;
341 end
342 for i = 21:40
343     group(i) = 2;
344 end
345 for i = 41:60
346     group(i) = 3;
347 end
348 group = group';
349
350 %%
351 [class, err, POSTERIOR, logp, coeff] = classify(VB, VA, group, 'linear');
352
353 VA = VA';
354 eagles = [];
355 journey = [];
356 fleetwood = [];
357
358 for i = 1:60
359     if (0 < i) && (i < 21)
360         eagles = [eagles; norm(VA(:, i))];
361     elseif (21 <= i) && (i < 41)

```

```

362     journey = [journey; norm(VA(:, i))];
363     elseif (41 <= i) && (i < 61)
364         fleetwood = [fleetwood; norm(VA(:, i))];
365     end
366 end
367 all = [eagles; journey; fleetwood];
368
369 %%
370 figure(4)
371 subplot(2,1,1)
372 plot(group, 'bo')
373 title('True Group Labels of Training Data')
374 xlabel('Training Group Sample')
375 ylabel('label')
376 subplot(2,1,2)
377 plot(class, 'ro')
378 title('Classification of Test Data')
379 xlabel('Training Group Sample')
380 ylabel('label')
381
382 %%
383 %calculate error
384 corr = 0;
385 for i = 1:length(group)
386     if group(i) == class(i)
387         corr = corr + 1;
388     end
389 end
390 error = (length(group) - corr) / (length(group)) * 100;
391 disp(error);
392
393 %%
394 figure(5)
395 plot(eagles, 'bo')
396 hold on
397 plot(journey, 'ro')
398 hold on
399 plot(fleetwood, 'go')
400 hold on
401 %% TEST 3-
402 A = [];
403 count = 1;
404 [count, A] = updateA('classical.wav', count, A, 30, 30);
405 [count, A] = updateA('classical2.wav', count, A, 30, 30);
406 [count, A] = updateA('EaglesS1.wav', count, A, 30, 10);
407 [count, A] = updateA('EaglesS2.wav', count, A, 30, 10);
408 [count, A] = updateA('Journey1.wav', count, A, 30, 10);
409 [count, A] = updateA('Journey2.wav', count, A, 30, 10);
410 [count, A] = updateA('fleetwood1.wav', count, A, 30, 10);
411 [count, A] = updateA('fleetwood2.wav', count, A, 30, 10);
412 [count, A] = updateA('Snoop1.wav', count, A, 30, 10);
413 [count, A] = updateA('Snoop2.wav', count, A, 30, 10);
414 [count, A] = updateA('Eminem1.wav', count, A, 30, 10);
415 [count, A] = updateA('Eminem2.wav', count, A, 30, 10);

```

```

416 [count , A] = updateA( '50cent1.wav' ,count ,A,30 ,10);
417 [count , A] = updateA( '50cent2.wav' ,count ,A,30 ,10);
418
419 %A = A'
420
421 [UA,SA,VA] = svd(A,0);
422 VA = VA(:,1:10);
423 %Create Test Data
424 B = [];
425 count = 1;
426 [count , B] = updateA( 'classical3.wav' ,count ,A,30 ,30);
427 [count , B] = updateA( 'classical4.wav' ,count ,A,30 ,30);
428 [count , B] = updateA( 'EaglesS3.wav' ,count ,A,30 ,10);
429 [count , B] = updateA( 'EaglesS4.wav' ,count ,A,30 ,10);
430 [count , B] = updateA( 'Journey3.wav' ,count ,A,30 ,10);
431 [count , B] = updateA( 'Journey4.wav' ,count ,A,30 ,10);
432 [count , B] = updateA( 'fleetwood3.wav' ,count ,A,30 ,10);
433 [count , B] = updateA( 'fleetwood4.wav' ,count ,A,30 ,10);
434 [count , B] = updateA( 'Snoop3.wav' ,count ,A,30 ,10);
435 [count , B] = updateA( 'Snoop4.wav' ,count ,A,30 ,10);
436 [count , B] = updateA( 'Eminem3.wav' ,count ,A,30 ,10);
437 [count , B] = updateA( 'Eminem4.wav' ,count ,A,30 ,10);
438 [count , B] = updateA( '50cent3.wav' ,count ,A,30 ,10);
439 [count , B] = updateA( '50cent4.wav' ,count ,A,30 ,10);
440 %B = B'
441
442 [UB,SB,VB] = svd(B,0);
443 VB = VB(:,1:10);
444 % Make the group names
445 group = [];
446 for i = 1:60
447     group(i) = 1;
448 end
449 for i = 61:120
450     group(i) = 2;
451 end
452 for i = 121:180
453     group(i) = 3;
454 end
455 group = group';
456
457 %%
458 [class ,err ,POSTERIOR,logp ,coeff] = classify(VB,VA,group , 'linear' );
459
460 VA = VA';
461 classical = [];
462 rock = [];
463 rap = [];
464
465 for i = 1:60
466     if (0<i)&& (i<61)
467         classical = [classical;norm(VA(:,i))];
468     elseif (61 <= i)&&(i < 121)
469         rock = [rock;norm(VA(:,i))];

```

```

470         elseif (121 <= i) && (i < 181)
471             rap = [rap; norm(VA(:, i))];
472         end
473     end
474     all = [classical; rock; rap];
475
476     %%
477     figure(4)
478     subplot(2, 1, 1)
479     plot(group, 'bo')
480     title('True Group Labels of Training Data')
481     xlabel('Training Group Sample')
482     ylabel('label')
483     subplot(2, 1, 2)
484     plot(class, 'ro')
485     title('Classification of Test Data')
486     xlabel('Training Group Sample')
487     ylabel('label')
488
489     %%
490     %calculate error
491     corr = 0;
492     for i = 1:length(group)
493         if group(i) == class(i)
494             corr = corr + 1;
495         end
496     end
497     error = (length(group) - corr) / (length(group)) * 100;
498     disp(error);
499
500     %%
501     figure(5)
502     plot(classical, 'bo')
503     hold on
504     plot(rock, 'ro')
505     hold on
506     plot(rap, 'go')
507     hold on
508
509     %% Functions:
510     function [yspec] = makeSpectrogram(y, Fs)
511         N = length(y); % sample length
512         slength = N/Fs; % total time span of audio signal
513         t = linspace(0, N/Fs, N);
514         %plot(t, y); % plots the audio
515         L = t(end);
516         k = (1/L) * [0:(N/2)-1 -(N/2):-1]; ks = fftshift(k);
517         y = y';
518         yspec = [];
519         tslide = 0:0.5:slength;
520         for j = 1:length(tslide)
521             g = exp(-(100)*(t - tslide(j)).^2);
522             yg = g.*y;
523             ygt = fft(yg);

```



```

524     yspec=[yspec;abs(fftshift(ygt))];
525 end
526 end
527
528 function [newc, newA] = updateA(file ,count ,A,start , numcol)
529     [y,Fs] = audioread(file);
530     for i = start :5:((start-5)+(5*numcol))
531         sample = [i*Fs,(i+5)*Fs];
532         [muz,fs] = audioread(file , sample);
533         muz = muz(:,1);
534         muz = muz(1:5:length(muz));
535         fs = fs/5;
536         sp = makeSpectrogram(muz,fs);
537         n = length(sp(1,:));
538         m = length(sp(:,1));
539         spv = reshape(sp,n*m,1);
540         A(:,count) = spv;
541         count = count+1;
542     end
543     newA = A;
544     newc = count;
545 end

```