AMATH 582 Homework 4

Shabab Ahmed
Github link: https://github.com/sahmed95/AMATH582

March 06, 2020

Abstract

This project involves two parts. The first part of the project consists of producing eigenfaces using SVD and principal component analysis. The project uses datasets consisting of cropped faces and uncropped faces to carry out the analysis. The differences between the eigenfaces and rank of the face space for the cropped and uncropped faces are discussed along with their low rank approximations. The second part of the project involves using machine learning techniques to classify musical genres. We conduct three tests considering: 1) three different bands each of a different genre , 2)three bands from within the same genre and 3) various bands within different genre. For each test, we use the data to create training sets and use linear discrimination analysis and Naive Bayes algorithm to predict classification of new songs. In general, the Naive Bayes algorithm seems to be more successful in identify new music than linear discrimination analysis.

1 Introduction and Overview

Image recognition has become a growing are in the field of mathematics due to its practical applications. In particular, mathematical methods are being developed to help automate computers to recognize people, animals, and places. Image recognition is being used so widely that almost everybody encounters it on a daily basis. For example, websites automatically prompt to tag people it recognizes when a new picture is uploaded to social media. A lot of the methods for image recognition involve statistics, time-frequency analysis and the SVD. We have discussed the SVD in a previous project and that is exactly what we need in this case. We can image that working with images with thousands and thousands of pixels can be computationally expensive. Dimensionality reduction becomes key in these methods and that is exactly where the use of SVD in principal components analysis comes in. We need to find a way to able to represent an image with thousands of pixels in as few pixels as possible. In this project, we take datasets of cropped and uncropped images of people's faces and try to reduce the dimension of the space, that is, try to construct low rank approximations. We conduct principal component analysis using the SVD in order to figure out how many principal components we need to successfully recreate/identify the images.

The second part of the project involves music classification. Humans are able to instantly recognize musical genres and the question of how the brain classifies such information and how it comes to a decision based on new information is an interesting one to think about. This also raises the questions of whether we can 'teach' a computer the process to identify new music. This is what we seek to do in the second part, that is, classify a given 5 second piece of music sample. This can be made possible by conducting time-frequency analysis of the songs. We have dealt with time-frequency analysis in the past, particularly, the use of windowed Fourier transforms where we create a window and slide it across the signal to extract frequency components at each instant of time. This helps us categorize a piece of music by characterizing what frequency components arise in the signal at what time. Spectogram is an useful visual representation of this characterization and can be thought of as the signature of a music siganl. We do not discuss time-frequency analysis or the spectogram any further in this project as this has been discussed in the past.

We use time frequency analysis in conjunction with principal components and other statistical methods. We have already discussed principal component analysis in a previous project. The techniques used in this example is a simple example of the method known as *machine learning*. In this project, a supervised learning technique is implemented in which a prescribed set of data is known and classified beforehand. This prescribed set of data is the 5-second music samples we use to create the training set. In particular, we use linear discrimination analysis and Naive Bayes algorithm to classify new music samples based on the training set. These are discussed further in the Theoretical Background Section. In what follows, we give a brief overview of the process we undergo in this project.

1.1 Part 1

We have two datasets of cropped and uncropped images. The first step of the process involved extracting the pictures and loading them onto `MATLAB`. We use the `dir` command on MATLAB to find the folder and subfolder containing the images. We find the folders and subfolders by getting their name as an attribute from the struct created bt the command `dir`. Once we have found the files containing the images. Each image is read as a matrix of pixels and so we compute the size of the matrices. This size is then used to reshape the matrices into a column vector which are then appended into one matrix. The columns of this appended matrix corresponds to images we read onto `MATLAB`. We convert the matrix into a double precision matrix so that we are able to able to do math on it. After having completed all of these steps, we compute the SVD. We plot the spectrum of singular values and produce low rank approximations. We also plot the eigenfaces which can be thought of as the average represented by the all the images of the faces. The results are discussed in the computational results section. It can be noted that computing the SVD in `MATLAB` boils down to one command so the major task is extracting the data and reshaping it into a matrix.

1.2 Part 2

1.2.1 Test 1

For this test, we consider three different bands where each band belongs to a different genre. In particular, we look at the bands Coldplay, Pink Floyd and Linkin Park. Coldplay belongs to the genre of alternative rock, Pink Floyd belongs to the genre of psychedelic rock and Linkin Park falls under the genre of alternative metal. For each band, we use the software 'Audacity' to concatenate 10 of their music samples to form one large music sample. Each of the 10 music samples consists of 5-second pieces of songs of the bands. After constructing these datasets, we read them onto `MATLAB`. The music files we read in actually consists of two channels: left and right. We use an audio normalization technique to produce a vector out of the two channels. This process consists of summing the two channels and then normalizing by multipling the sum by the absolute value of the maximum in either of the channels. Then, we conver the vectors into single precision to decrease memory usage. After constructing a vector of the signal, we perform the same tasks as one of our previous projects to create the spectogram. One thing we are careful about is ensure that the each row in the spectogram corresponds to a song. This is easily done by appending the result of the Gabor transform to a matrix at times of multiples of 5. We also ensure that the songs of the bands are grouped together in the spectogram matrix. We use a Gábor transform with Gaussian window with width parameter $a = 250$ and translation parameter $dt = 1$. The translation parameter is large but we do this due to computation constraints. The computer took too long to compile or crashed for smaller values of dt. We found that even using this value of the translation parameter gave us expected results so we stuck with it.

We compute the economy matrix of the matrix used to produce the spectogram. This gives us the principal components and we use the first few projections onto the principal components to create our training set and test set. We have 10 songs for each band, and we randomly pick seven songs as the training set and the remaining 3 as our test set. We also create a vector with labels that help us classify the training set. We will use these classifications to classify the test data. We design a statistical decision threshold for discrimination between the bands using the methods of linear discriminant analysis and Naive Bayes algorithm. We compare it with the classifications we were supposed to get and compute the accuracy. We cross validate by repeating the process of randomly selecting test and training set and running the analysis for 500 trials and then computing the average accuracy of the trials. The results are described in the Computation Results section.

1.2.2 Test 2

For this test, we use three bands within the same genre. We focus on on the late 90s Seattle grunge bands: Soundgarden, Alice in Chains, and Pearl Jam. We use ten 5-second samples of songs for each of the bands and the rest of the process for this test is exactly the same as Test 1.

1.2.3 Test 3

This case considers three bands each from different musical genres. We consider the genre of alternative rock using the bands: Coldplay, U2 and Imagine Dragons. We also look at the grunge genre by consider the bands from Test 2. We also look at classical music by sampling music from Beethoven, Mozart and Bach. For each of the bands, we take five 5 second samples of their songs. So, for each genre we have 15 musical samples. We carry out the same process as above except we use a translation parameter of $dt = 1.5$ for computational constraints.

2 Theoretical Background

2.1 Singular Value Decomposition:

As we learned from our textbook [1], SVD of a $m \times n$ matrix A is the factorization of the form:

$$\mathbf{A} = \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\mathbf{V}^* \tag{1}$$

where $\hat{U}$ is a $m \times n$ matrix with orthonormal columns, $\hat{\Sigma}$ is a $n \times n$ diagonal matrix with non negative entries and V is a $n \times n$ unitary matrix. This is called the *reduced SVD* of $\mathbf{A}$ and this the form of SVD we use in the project. We also know that the SVD of a matrix always exists.

An important theorem involving SVD is the following:

Theorem 1: *If the rank of* $\mathbf{A}$ *is* r*, then there are* r *nonzero singular values.*

Anotherr important theorem involving the SVD:

Theorem 2: *A is the sum of* r *rank-one matrices*

$$\mathbf{A} = \sum_{\mathbf{j=1}}^{\mathbf{r}} \sigma_{\mathbf{j}}\mathbf{u_j}\mathbf{v_j^*} \tag{2}$$

where σ_j is the jth diagonal entry of Σ.

Theorem 1 can help us determine the rank of a system by just considering the singular values. Theorem 2 gives us a way of reconstructing $\mathbf{A}$ using low rank approximations. Many other important theorems regarding the SVD are given in [1].

SVD is used in principal component analysis as we use to diagonalize a covariance matrix in order to find a basis where each component is independent of one another. In our case, we take the SVD of the matrix where each column is an image. The columns of U gives us the modes of the images and the columns of V gives us the projection of the images onto these modes. We can then use the singular values whose magnitudes reflect their relative importance to find the more important modes and just use them to reconstruct the images. This helps us create low dimension space that best explains the variances in the face space.

2.2 Linear Discrimination Analysis

In our training algorithm we use statistical information about the SVD decomposition in order to make a decision about what categories the test songs belongs to. We essentially want to project the training set into a new basis function in a way that the projection produces well separated statistical distribution between the different groups of the training set. The idea of the LDA was first proposed by Fisher in the context of taxonomy.The goal of LDA is to find a suitable projection that maximizes the distance between the inter-class data while minimizing the intra-class data.

For a two-class LDA, the above idea results in consideration of the following mathematical formulation: Construct a project $\mathbf{w}$ such that

$$\mathbf{w} = \arg\max_{\mathbf{w}} \frac{\mathbf{w^T S_b w}}{\mathbf{w^T S_w w}} \tag{3}$$

where the scatter matrices for between-class $\mathbf{S_b}$ and within-class $\mathbf{S_w}$ data are given by

$$\mathbf{S_B} = (\mu_\mathbf{2} - \mu_\mathbf{1})(\mu_\mathbf{2} - \mu_\mathbf{1})^\mathbf{T} \tag{4}$$

$$\mathbf{S_w} = \sigma^\mathbf{2}_\mathbf{j=1}\sigma_\mathbf{x}(\mathbf{x} - \mu_\mathbf{j})(\mathbf{x} - \mu_\mathbf{j})^\mathbf{T} \tag{5}$$

.

These quantities basically measure the variance of the data sets as well as the variance of the difference in the means. In our example, the training set consisting of music samples are onto the first few principal modes obtained from the SVD and then the in-built `MATLAB` command is used to do the linear discrimination analysis.

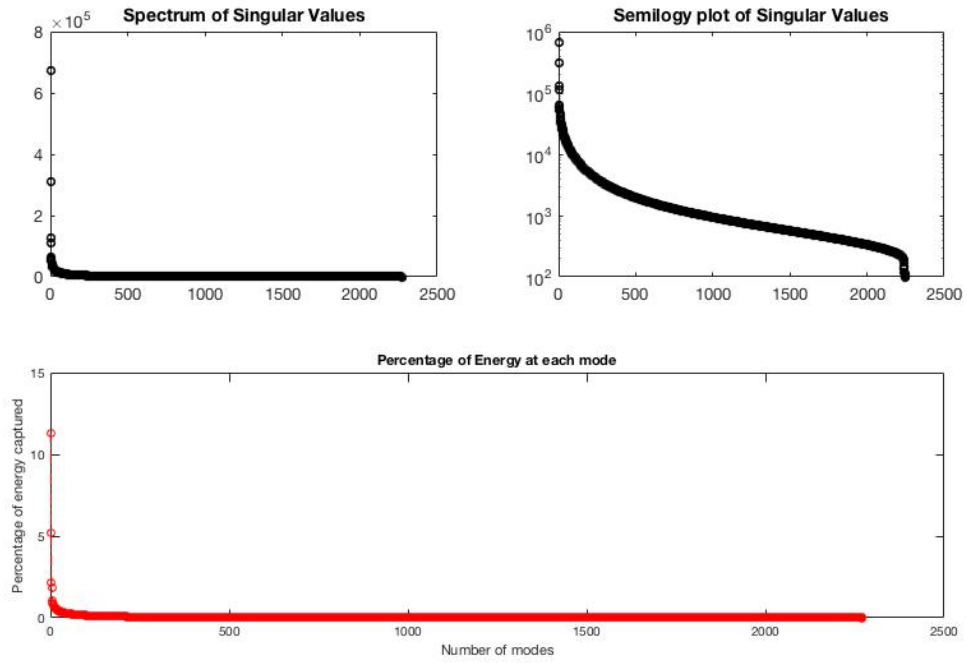

Figure 1: The top left panel contains the spectrum of singular values, the top right panel contains the semilogy plot of the singular values. The botton panel contains the percentage of energy obtained at each mode.

2.3 Naive Bayes Algorithm

This is a supervised algorithm that is based upon Bayes rule. Suppose we two classes of data that are 0 and 1 we create a score that gives us good separation. We construct the following ratio:

$$\frac{P(1|x)}{P(0|x)} \tag{6}$$

where $P(1|x)$ is the probability of having class 1 given the data x and $P(0|x)$ is the probability of having class 0 given the data x. This can be rewritten as:

$$\frac{P(1|x)}{P(0|x)} = \frac{P(x|1)P(1)}{P(x|0)P(0)} \tag{7}$$

These quantities are really use to compute and we involve prior information to create our score. Using the ratio, we can get a metric that gives us a good way of separting the data. In our case, we have three classes for each test and we use `MATLAB`'s in-built function to do this analysis.

3 Algorithm Implementation and Development

In what follows, we provide the steps undertaken in an orderly fashion. Part 1 involves computing the SVD which can be done in one line in `MATLAB`. Therefore, this is not a difficult task and the real task is in extracting the images from the files. For Part 2, we include the steps for Test 1 as the steps for the other tests are the same.

3.1 Part 1

1. List the contents of the folders that contains subfolders of images using `dir`
2. Initialize the matrix (`images`) whose columns will contain the images
3. For each of the item in the list of subfolders, obtain the attribute containing the name of the subfolder

4. Read the file onto `MATAB` using `imread`
5. Calculate the size (`sz`) of the matrix containing the pixels of the image
6. Reshape the matrix into a column vector (`image`)
7. Append it onto `images` as column so each column corresponds to an image
8. Convert the matrix (`images` to double precision
9. Compute the SVD of `images`
10. Produce plots of spectrum of singular values and eigenfaces
11. Reconstruct random images using a few modes using Equation(2)

3.2 Part 2

1. Read the music files onto `MATLAB` using `audioread` to produce a matrix containing the right and left channels of the signal
2. Set total time (`tr`) = 50, since for each band ten music samples of 5 seconds = 50 seconds
3. Normalize the audio signals for each band by summing the left and right channels and multiplying it by the absolute maximum value of the two channels to create a single vector
4. Produce the spectogram for each band using a Gaussian window of width parameter $a = 250$
   (a) We ensure that each song corresponds to a row in the spectogram matrix by ensuring that we append to the spectogram matrix for every time that is a multiple of 5 (sample consists of 5 seconds)
5. Concatenate the spectogram matrices for each band into one big spectogram matrix (`Spect`) in a way that the songs of bands are grouped together
6. Compute the SVD of `Spect`
7. Select the number of modes `features`), number of training (`trains`) and test (`tests`) songs
8. Create a histogram of the answer of the test songs- we know this because of the way we designed the spectogram matrix (songs by a band are grouped together)
9. Initialize accuracy vectors and set the number of trials
10. For each trial, create a equally spaced array (`q`) from 1 to the number of songs
11. Create a random permutation matrix (`q1`) containing values from 1 to number of songs and use the first `trains` values of the matrix to take projections of those particular songs onto the first few modes defined by `features`
12. Use `setdiff` to obtain `tests` values that are in `q` and not in `q1` - this will be test songs
13. Create the vector of classifications
14. Pass the training set, test set and the classifications into `MATLAB`'s in-builty `classify` and `fitcnb` commands
15. Calculate average accuracy over the trials

4 Computational Results

This section contains the computational results.

4.1 Part 1

We can recall that the columns of U are the modes of the images and the columns of V are the projection of the images onto these modes. Figure (1) contains the spectrum of singular values for the cropped images, the semilogy plot of singular values and the percentage of energy obtained at each mode. We can see from the top right panel that there are four singular values that are significantly higher than the rest. The rest just tail off to zero. This is also illustrated in the semilogy plot. However, the percentage of energy captured by these four modes is not that high. They capture about 25% of the energy. This maybe because there are so many images in the files that even though other singular values are close to zero, the sum of them contribute quite a bit to the total energy. We plot the eigenfaces using the first four features in Figure(2). We can see that we are able to observe facial features. The first features gives us the shape of the eyes and a bit of the nose. The second feature gives us the nose and the fourth feature gives a pretty good realization of a face. However, we want to check whether a 4-rank approximation gives us a good result.

A 4-rank reconstruction of randomly selected images is provided in Figure(3). The left images are the original images and the right images are the reconstructed images. We can see that we actually get pretty good approximations despite being blurry. We can make out the faces from the approximations. We do notice one problem, the second and the third reconstruction seems to have the same base for the face. However, we can see that the nose in the third reconstructed image is slightly off center just like the original image. As we increase the number of modes, the reconstructed images will look better and better in terms of its smoothness.

Figure(4) contains the spectrum of singular values for the uncropped images, the semilogy plot of singular values and the percentage of energy obtained at each mode. We can see that two singular values dominate. In fact, the first singular values capture about 30% of the energy. These might indicate that the rank is 2, but, we see that this is not really the case in Figure (5). Figure (5) contains the eigenfaces using the first four modes. The eigenfaces reflect that there is a lot shaking in the faces and it is hard to make out a face. In fact, Figure (6) reflects that we have to go eight modes to reduce this shaking effect. The nose is also hard to find in the eigenfaces. This shaking effect makes sense because we are using the background in these images and the background might make the pictures noisy in the sense of recognizing the face. We had a similar issue in tracking the paint can in Principal Component Analysis project.

Figure (7) contains the reconstruction of randomly selected uncropped images using the first four modes. We can see that these reconstructions are not as good as the reconstruction from the cropped image case. The images are not just blurry but they are also shaky and there are no discernible features of the face that can be pinpointed from the reconstructions. Even though the spectrum suggested the rank of the face space of uncropped images could be lower than cropped images, we find out this is not true. This, maybe, due to the presence of background (noise) in uncropped images.

4.2 Part 2

4.2.1 Test 1

Figure (8) contains the the spectrum of singular values. We can see that the 5 of the values dominate while the rest tail off to zero. So, we use 5 modes for the analysis. We also use 7 training songs and 3 test songs.

Figure(9) contains the classification used. We get this classification because of the way we set up the spectogram and the training and the test sets. Figure(10) contains the correct answers. Figure (11) contains the classifications we got using LDA and Figure(12) contains the classifications we got using Naive Bayes algorithm for one particular trial. We can see that in this trial the LDA does a poor job of classifying class 2. NB does a better job in classifying class 1. The average accuracy rate over 500 trials for LDA was about 35% and the average accuracy rate for Naive Bayes was about 55% for one simulation of the trials. So, overall the Naive Bayes algorithm fared better.

4.2.2 Test 2

Figure(13) contains the spectrum of singular values. We can already see that we do not have a nice separation for the first four singular values so we should expect to require more features to correctly identify test songs. However, we use 5 features and the same number of test and training songs to compare how it fares with Test 1.

Figure(14) contains the classification of test set using LDA and Figure (15) contains the classification of test set using Naive Bayes for one trial. We can see that for this particular trial, LDA cannot identify a single class 3 song. Even Naive Bayes is unable to identify a class 3 song incorrectly. This makes sense that LDA and Naive Bayes will have a hard time classifying cause the sogns from the bands chosen are so similar. They are chosen from the same genre, same era and same location and so their musics would be similar. The algorithms cannot statistically

distinguish the songs very well. This was reflected in the spectrum of the singular values too. Overall, Naive Bayes still fared better with an accuracy of 39% over 500 trials. LDA had an accuracy rate of 25%.

4.2.3 Test 3

In this case, we have 15 songs for each genre. We use 10 songs in the training set and 5 songs in the test set. We use 7 modes after considering the singular value spectrum which is not included in the project. The classifications for the training set will look really similar to Figure (9) but with 3 more songs added for each class. Similarly, the correct classifications for the test set will look like Figure(10) but with 2 more songs for each class.

Figure (16) contains the LDA classification and Figure (17) contains the Naive Bayes classification for one trial. Naive Bayes does an excellent job of classifying class 1 and also does a good job of classifying class 2. However, it struggles to classify class 3 correctly. LDA also seems to classify class 1 better and for this trial seems to better classify for this trial. However, after cross validating over 500 trials we see that Naive Bayes does a better job than LDA as it has an accuracy of 50% compared to 40% accuracy of LDA.

5 Summary and Conclusions

We have seen that low rank approximations can provide a good representation of faces that can be used in image recognition. We also observed that the rank of the face space of cropped images is smaller than that of uncropped images. This is because of the fact that the background of uncropped images can distort extracting the fact from the image. Also, the positioning of the face will determine the rank of approximations required. Lighting is also an important issue as we observed in Figure (3), where the first reconstructed image is harder to see than the other images. This makes sense if we compare Face ID technology on smart phones as they will not work properly in the dark or if we are not presenting our full face.

In Part 2 of the project, we created a machine learning algorithm using Linear Discrimination analysis and Naive Bayes Algorithm. We created training and test sets and tried to classify the test sets. Naive Bayes outperformed Linear discrimination analysis in all cases. We saw that Test 2 was the most difficult to identify given the homogenous nature of the songs being sampled. This project does not provide high confidence in such classification algorithms as the highest accuracy rate was about 50%. However, we did not use that many song samples in the training set and the number of training samples compared to test samples were not that high. We also used a large translation parameter to construct the spectogram matrix due to computation constraints. However, changing this parameter might affect the accuracy and decreasing might increasing accuracy because we can get a better characterization of the song.

References

[1] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data.* Oxford University Press, 2013.

Appendix A MATLAB Functions

This section contains important MATLAB functions used in the project with a brief implementation explanation.

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.
- `dir name` lists files and folders that match `name`. When `name` is a folder (in our case), `dir` lists the contents of the folder.
- `[U,S,V] = svd(X)` performs reduced SVD on the matrix `X` and returns `U`, `S` and `V` which are $\hat{U}$, $\hat{\Sigma}$ and V from Eq.(1)
- `Y = double(X)` converts the values in `X` to double precision.
- `Y = single(X)` converts the values in X to single precision.
- `p = randperm(n,k)` returns a row vector containing `k` unique integers selected randomly from 1 to `n`.
- `C = setdiff(A,B)` returns the data in `A` that is not in `B`, with no repetitions.

- `Mdl = fitcnb(Tbl,Y)` returns a multiclass naive Bayes model (Mdl), trained by the predictors in the table `Tbl` and class labels in the array `Y`. In our case, `Tbl` is the spectogram matrix for training samples of music and `Y` is the vector of classes.
- `classify(sample,training,group)`classifies each row of the data in `sample` into one of the groups in `training`. Each element of `group` defines the group to which the corresponding row of `training` belongs. The output class indicates the group to which each row of `sample` has been assigned, and is of the same type as `group`. In our case, sample was the spectogram matrix of the test music samples, training was the spectogram matrix of the test music samples and the groups were the classes.

Appendix B MATLAB Code

We provide the MATLAB code for this project. We provide full code for Part 1. We only provide code for Test 1 for Part 2. The codes for the other test are almost the same.

```
% Shabab Ahmed

clear all; close all; clc

% loading the images into Matlab

cropped = dir('CroppedYale'); % directory
cropped_f = cropped(4:end);

images = [];
for i = 1:length(cropped_f)
    if i<9
        strName = ['yaleB0' num2str(i)];
    else
        strName = ['yaleB' num2str(i)];
    end
    sub = dir(['CroppedYale', '/' ,strName]);
    sub_file = sub(4:end);
    for i = 1:length(sub_file)
        file = (sub_file(i).name); % get attribute name from struct
        image1 = imread(file);  % read the file
        sz= size(image1);
        sz1 = sz(1);
        sz2 = sz(2);
        sz = sz1*sz2;
        image = reshape(image1, sz,1); % reshaping image
        images = [images, image];
    end
end



images =  double(images);
[U,S,V] =svd(images,'econ');
sig = diag(S);



figure(1)

subplot(2,2,1)
plot(sig, 'ko','Linewidth',[1.5])
title('Spectrum of Singular Values')
set(gca, 'Fontsize', [13])

subplot(2,2,2)
semilogy(sig,'ko','Linewidth',[1.5])
ylim([10^(2); 10^(6)]);
set(gca,'Fontsize',[13],'Ytick', [10^(2) 10^(3) 10^(4),...
10^(5), 10^(6)])
title('Semilogy plot of Singular Values')
```

```
subplot(2,1,2)
plot((sig/sum(sig))*100, '-.or')
xlabel('Number of modes')
ylabel('Percentage of energy captured')
ylim([0, 15]);
title('Percentage of Energy at each mode')

figure(2)
title('First four modes')
set(gca, 'Fontsize', [16])
% eigenfaces
for j = 1:4
    subplot(2,2,j)
    ef = reshape(U(:,j),sz1, sz2);
    pcolor(ef), axis off, shading interp, colormap(hot)
end
%
m = randi([0 2269], 1, 3);

figure(3)
title('Reconstruction of Faces')
for j = 1:length(m)
    i = m(j);
    im = images(:,i);
    im = reshape(im, sz1, sz2);
    im = uint8(im);
    subplot(3,2,(2*j)-1)
    imshow(im);
    A = U*S(:,1:50)*V(:,1:50)';
    A = A(:,i);
    A = reshape(A, sz1, sz2);
    A = uint8(A);
    subplot(3,2,2*j)
    imshow(A)

end

%-------------------------------------------------------------------------

uncropped = dir('yalefaces_uncropped'); % directory
uncropped_f = uncropped(end);

images_u = [];
sub = [uncropped_f.name]; % get attribute name from struct
subfile =dir(['yalefaces_uncropped', '/', sub]);
sub_file = subfile(4:end);
for i = 1:length(sub_file)
        file = (sub_file(i).name); % get attribute name from struct
        image1 = imread(file);  % read the file
        sz= size(image1);
        sz1 = sz(1);
        sz2 = sz(2);
        sz = sz1*sz2;
        image = reshape(image1, sz,1); % reshaping image
        images_u = [images_u, image];
end
```

```
images_u =  double(images_u);
[U,S,V] =svd(images_u,'econ');
sig = diag(S);


figure(4)

subplot(2,2,1)
plot(sig, 'ko','Linewidth',[1.5])
title('Spectrum of Singular Values')
set(gca, 'Fontsize', [13])

subplot(2,2,2)
semilogy(sig,'ko','Linewidth',[1.5])
% ylim([10^(2); 10^(6)]);
% set(gca,'Fontsize',[13],'Ytick', [10^(2) 10^(3) 10^(4),...
% 10^(5), 10^(6)])
title('Semilogy plot of Singular Values')

subplot(2,1,2)
plot((sig/sum(sig))*100, '-.or')
xlabel('Number of modes')
ylabel('Percentage of energy captured')
% ylim([0, 15]);
title('Percentage of Energy at each mode')

figure(5)
title('First four modes')
set(gca, 'Fontsize', [16])


% eigenfaces
modes = [2,4,6,8];
for i = 1:length(modes)
    subplot(2,2,i)
    k = modes(i)
    ef = reshape(U(:,k),sz1, sz2);
    pcolor(ef), axis off, shading interp, colormap(hot)
end
%
m = randi([0 165], 1, 3);

figure(6)
title('Reconstruction of Faces')
for j = 1:length(m)
    i = m(j);
    im = images_u(:,i);
    im = reshape(im, sz1, sz2);
    im = uint8(im);
    subplot(3,2,(2*j)-1)
    imshow(im);
    A_u = U*S(:,1:4)*V(:,1:4)';
    A_u = A_u(:,i);
    A_u = reshape(A_u, sz1, sz2);
```

```
    A_u = uint8(A_u);
    subplot(3,2,2*j)
    imshow(A_u)

end


%Part 2
%Test 1

clear all; close all; clc
%%
x = audioread('Pink_Floyd.wav');
y = audioread('LP.wav');
z = audioread('CP.wav');
tr = 50;

% audio normalization
xsum = sum(x,2);
mLx = max(abs(x(:,1)));
mRx = max(abs(x(:,2)));
max_x = max([mLx mRx]);
x = xsum*max_x;

ysum = sum(y,2);
mLy = max(abs(y(:,1)));
mRy = max(abs(y(:,2)));
max_y = max([mLy,mRy]);
y = ysum*max_y;

zsum = sum(z,2);
peakAz = max(abs(zsum));
peakLz = max(abs(z(:,1)));
peakRz = max(abs(z(:,2)));
max_z = max([peakLz peakRz]);
z = ysum*max_z;

%
Fs = length(x);


x= x'/2;
y= y'/2;
z = z'/2;

r = mod(length(x),2);

% make even length
if  r == 1
    x = x(1:end-1);
    y = y(1:end-1);
    z = z(1:end-1);

end


n = length(x);
```

```
L = tr;    % final time
% discretized time vectors

t = linspace(0,L, n);

k = (2*pi/L)*[0:n/2-1 -n/2:-1];  %rescale
ks = fftshift(k); %shift
x=single(x); y=single(y); z =single(z);
%%
a = 250; % window width
dt = 1;
xgt_spec = []; ygt_spec = []; zgt_spec = [];
xg_spec =[]; yg_spec= []; zg_spec =[];
tslide = 0:dt:t(end-1);

for j=1:length(tslide)
    g=exp(-a*(t-tslide(j)).^2); % Gabor
    xg = g.*x; xgt =fft(xg);
    yg=g.*y; ygt=fft(yg);
    zg = g.*z; zgt = fft(zg);
    xgt =single(xgt); ygt =single(ygt); zgt=single(zgt);
    xgt_spec=[xgt_spec; abs(fftshift(xgt))];
    ygt_spec=[ygt_spec; abs(fftshift(ygt))];
    zgt_spec=[zgt_spec; abs(fftshift(zgt))];

    % make sure each row in spectogram is a song
    % if time is a multiple of 5s we have a new song
    if mod(j+50,50)== 0
        xg_spec = [xg_spec; xgt_spec];
        yg_spec = [yg_spec; xgt_spec];
        zg_spec = [zg_spec; xgt_spec];
        xgt_spec = [];
        ygt_spec = [];
        zgt_spec = [];
    end
end

Spect = [xg_spec; yg_spec; zg_spec];
%%
[U,S,V] = svd(Spect, 'econ');
sig = diag(S);
plot(sig/sum(sig)*100,  '-.or')

figure(2)
plot(sig, 'ko')
title('Spectrum of Singular Values')
set(gca, 'Fontsize', [14])



%%
features = 6;

train_s = 7;
test_s = 3;
```

```
% figure(3)
answer = [ones((test_s),1); 2*ones(test_s,1); 3*ones(test_s,1)];
% bar(answer)
% title('Correct Classification')
% xlim([0.5 9.5])

% initializing vectors for accuracy
acc_LDA = [];
acc_nb = [];

trials =500; %number of trials
for j = 1:trials
    % randomly choosing songs
    q = linspace(1,10,10);
    q1 = randperm(10);
    q2 = randperm(10);
    q3 = randperm(10) ;
    q1 = q1(1:train_s);
    q11 = setdiff(q,q1);
    q2 = q2(1:train_s)+10;
    q=linspace(11,20,10);
    q22 = setdiff(q,q2);
    q3 = q3(1:train_s)+20;
    q = linspace(21,30,10);
    q33 = setdiff(q,q3);

    % create training set

    xtrain = [];ytrain =[]; ztrain=[];
    for i = 1:length(q1)
        sl = 80000;
        xt = V(q1(i), 1:features);
        xtrain = [xtrain; xt];
        yt = V(q2(i), 1:features);
        ytrain = [ytrain; yt];
        zt = V(q3(i), 1:features);
        ztrain = [ztrain; zt];
    end
    % create test set

    xtest = []; ytest =[]; ztest = [];
    for k =1:length(q11)
        xt = V(q11(k), 1:features);
        xtest = [xtest; xt];
        yt = V(q22(k), 1:features);
        ytest = [ytest; yt];
        zt = V(q22(k), 1:features);
        ztest = [ztest; zt];
    end
    test = [xtest; ytest; ztest];
    train = [xtrain; ytrain; ztrain];

    % grouping
    ctrain = [ones(train_s,1); 2*ones(train_s,1); 3*ones(train_s,1)];
```

```
    % Linear Discriminant Analysis

    pre =classify(test, train, ctrain);
    % accuracy
    acc = 0;
    for l= 1:length(pre)
        if pre(l) == answer(l)
            acc =acc+1;
        end
    end
    acc_LDA = [acc_LDA;acc/length(pre)*100];

    % Naive Bayes
    nb = fitcnb(train, ctrain);
    pre1 = nb.predict(test);

    % accuracy

    acc_1 = 0;
    for p = 1:length(pre1)
        if pre1(p) == answer(p)
            acc_1=acc_1 +1;
        end
    end
    acc_nb = [acc_nb;acc_1/length(pre1)*100];

end

% Average accuracy

accuracy_LDA = sum(acc_LDA)/length(acc_LDA);
accuracy_nb = sum(acc_nb)/length(acc_nb);


%
% figure(4)
% bar(pre);
% title('Classification of Test set (LDA)')
% set(gca,'Fontsize', [14])
% ylim([0 3])
% xlim([0.5 9.5])
%
% figure(5)
% bar(pre1)
% title('Classification of Test set (Naive Bayes)')
% set(gca,'Fontsize', [14])
% ylim([0 3])
% xlim([0.5 9.5])
```

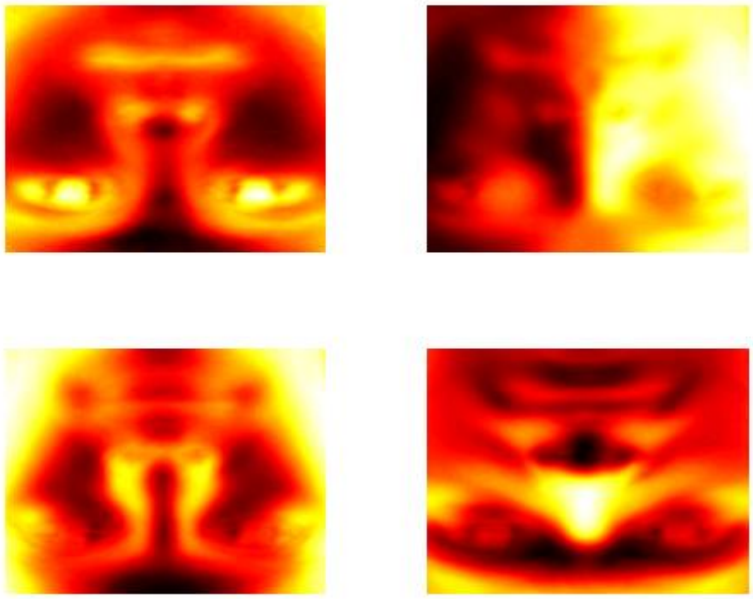

Figure 2: Eigenfaces using the first four features

Figure 3: Reconstructions of images using the first four modes

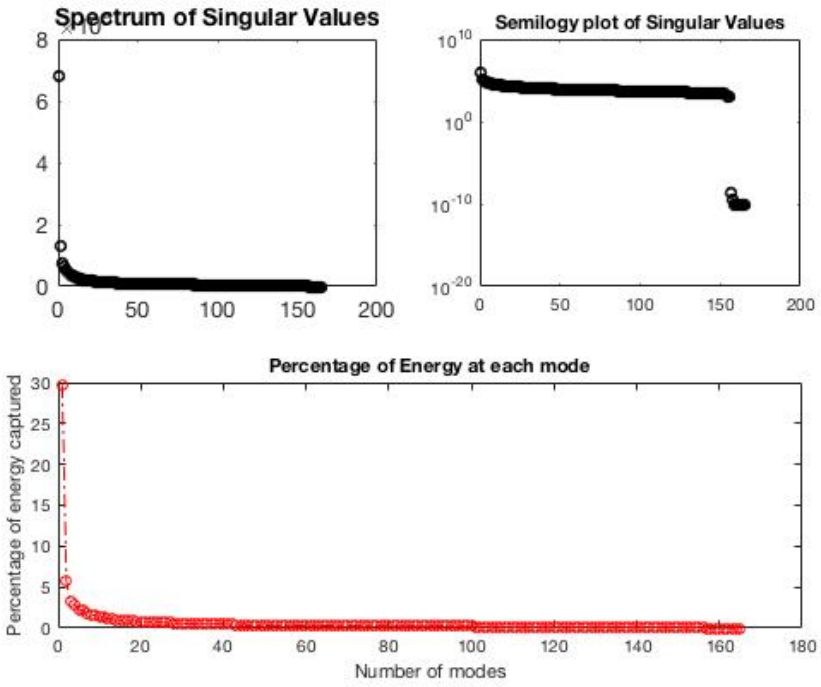

Figure 4: The top left panel contains the spectrum of singular values, the top right panel contains the semilogy plot of the singular values. The botton panel contains the percentage of energy obtained at each mode.

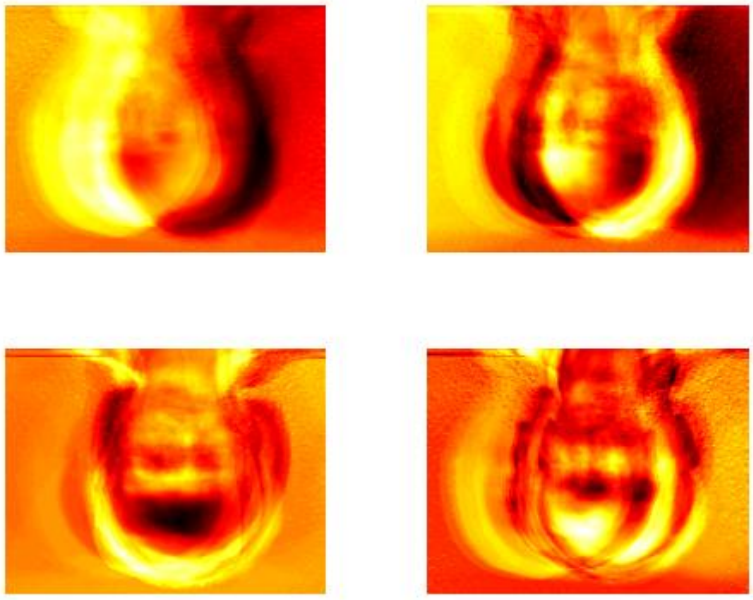

Figure 5: Eigenfaces of the uncropped images using the first four modes

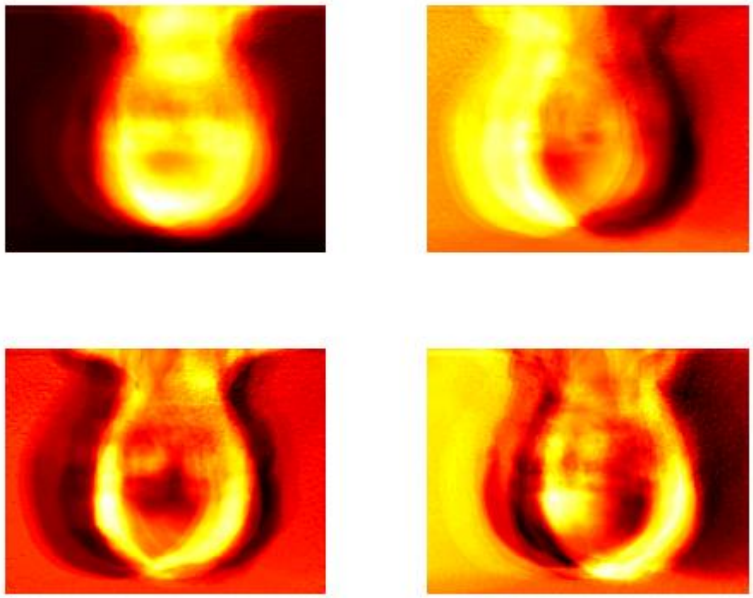

Figure 6: Eigenfaces of the uncropped images using the modes 2, 4, 6 and 8

Figure 7: Reconstructions of uncropped images using the first four modes

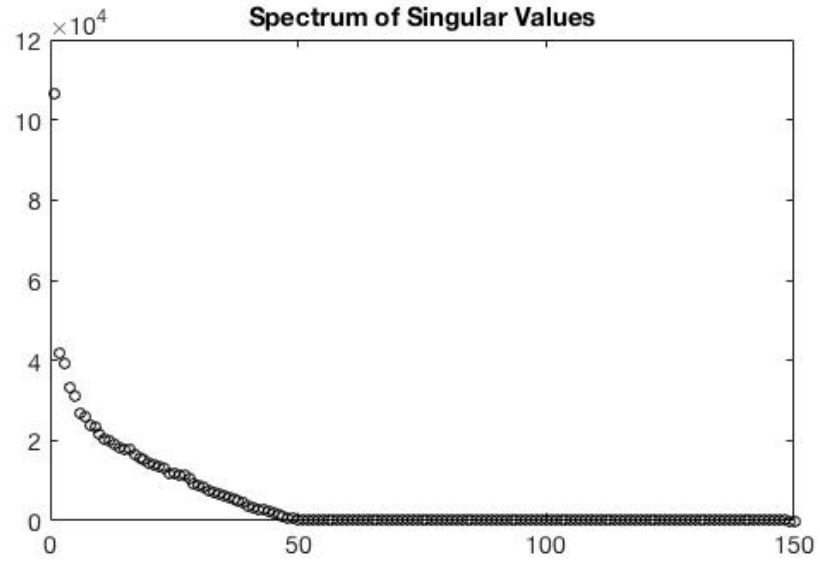

Figure 8: Spectrum of Singular Values (Test 1)

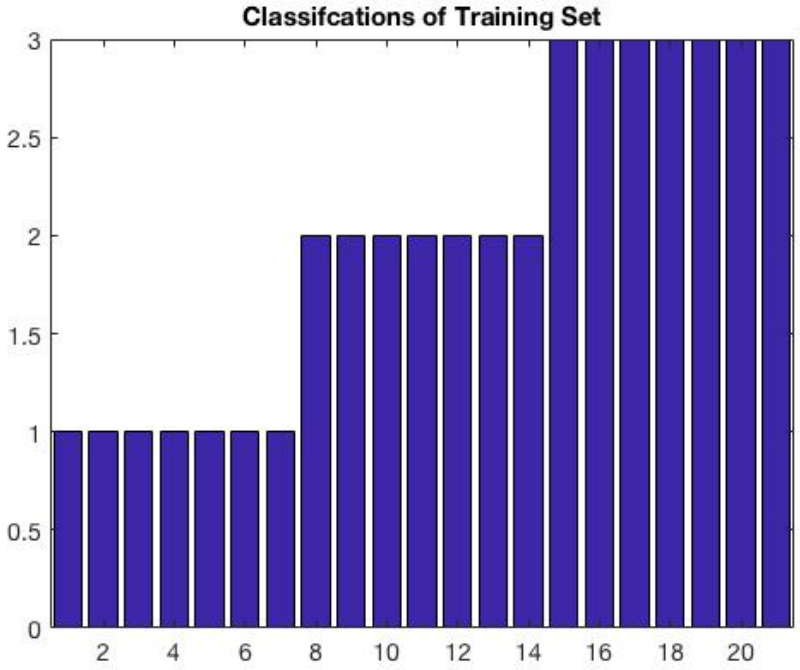

Figure 9: Classification for Training Set

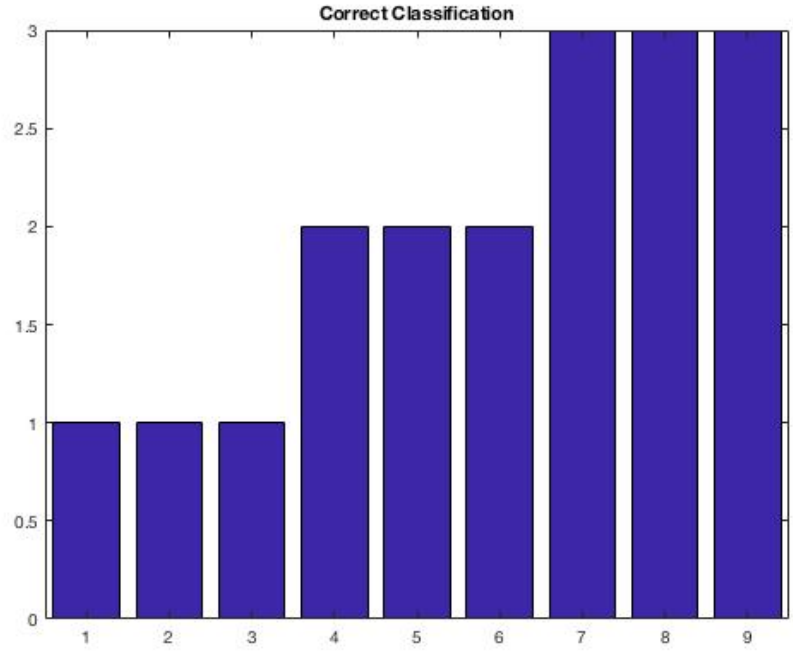

Figure 10: Correct Classification of Test Set

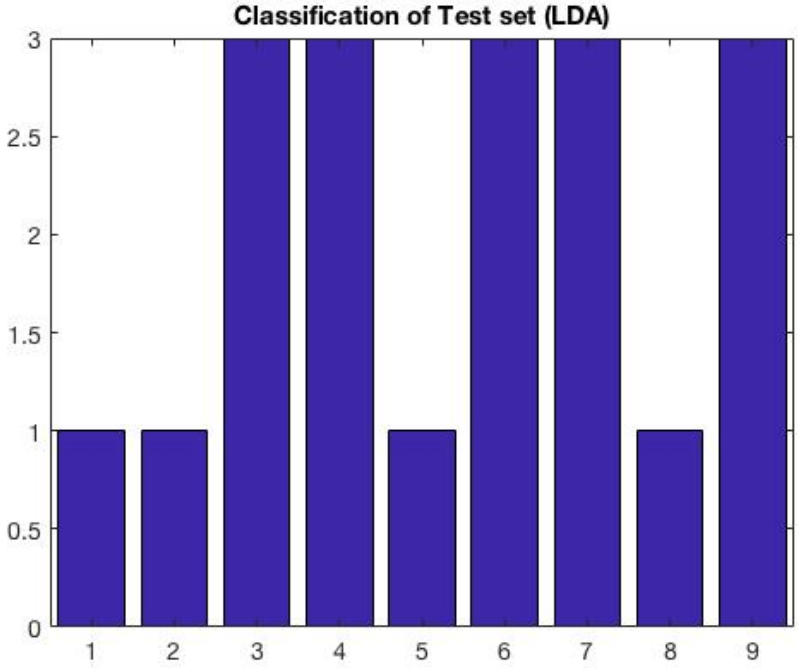

Figure 11: Classification for Test 1 Set (LDA)

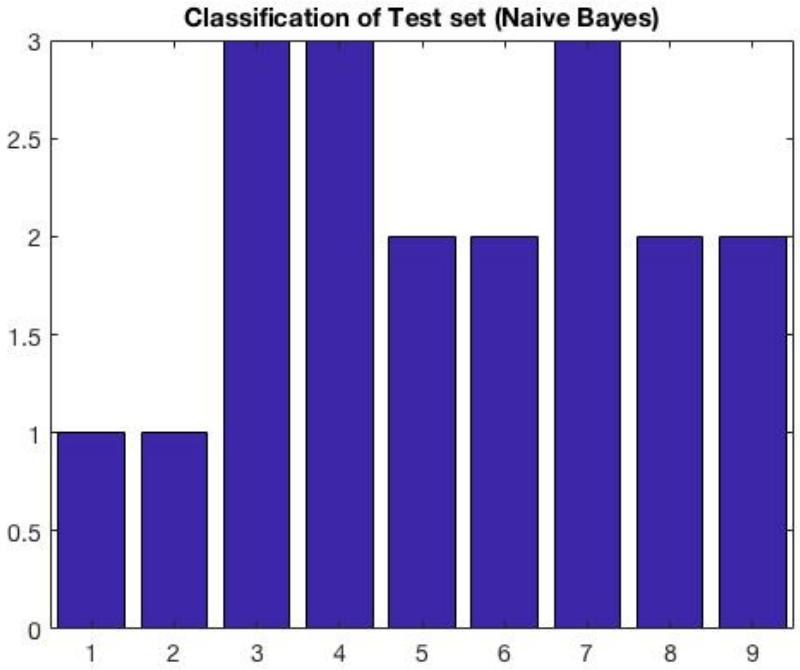

Figure 12: Classification for Test 1 Set (NB)

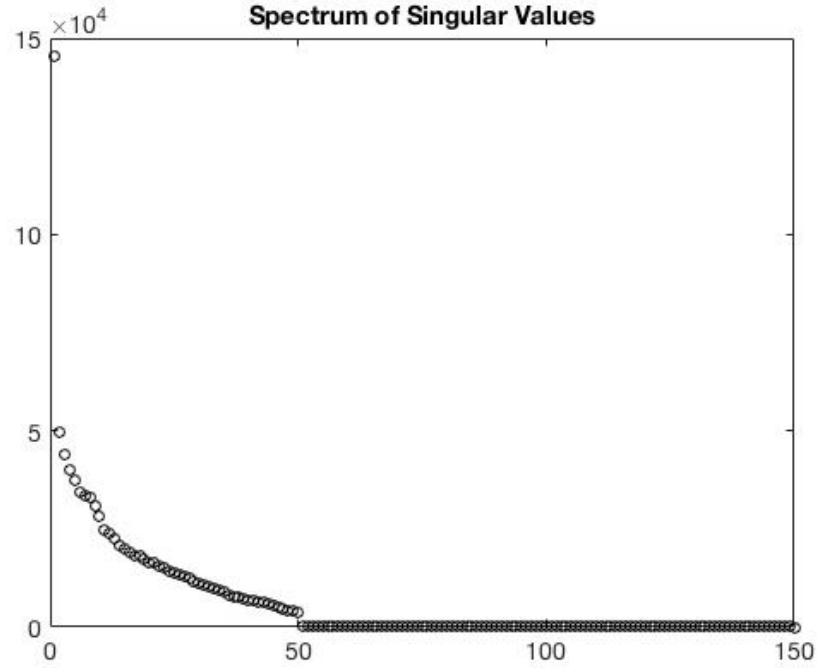

Figure 13: Spectrum of Singular Values (Test 2)

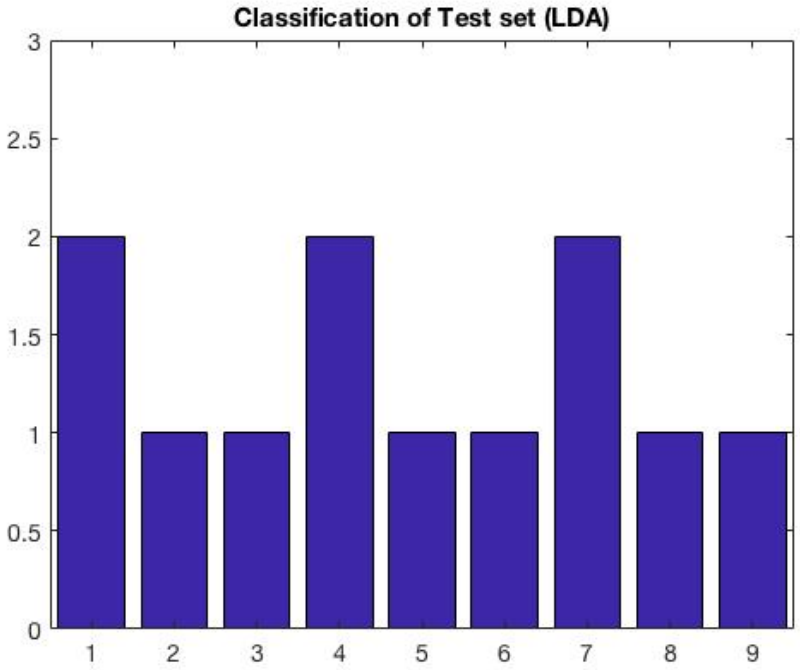

Figure 14: Classification for Test 2 Set (LDA)

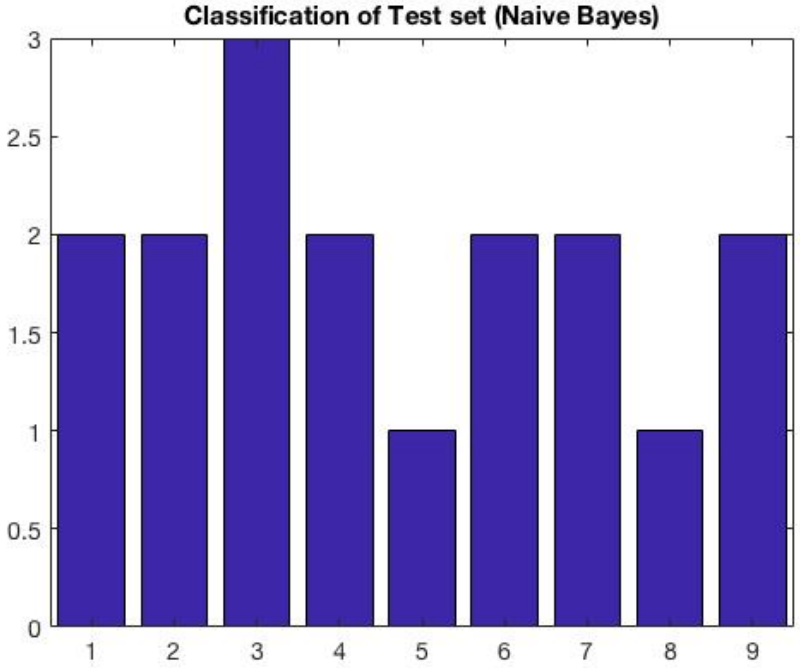

Figure 15: Classification for Test 2 Set (NB)

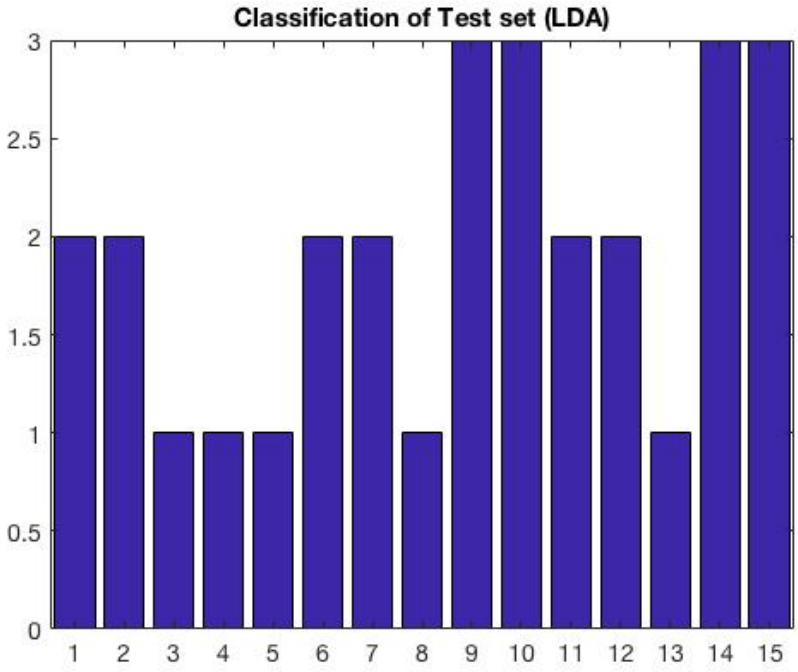

Figure 16: Classification for Test 3 Set (LDA)

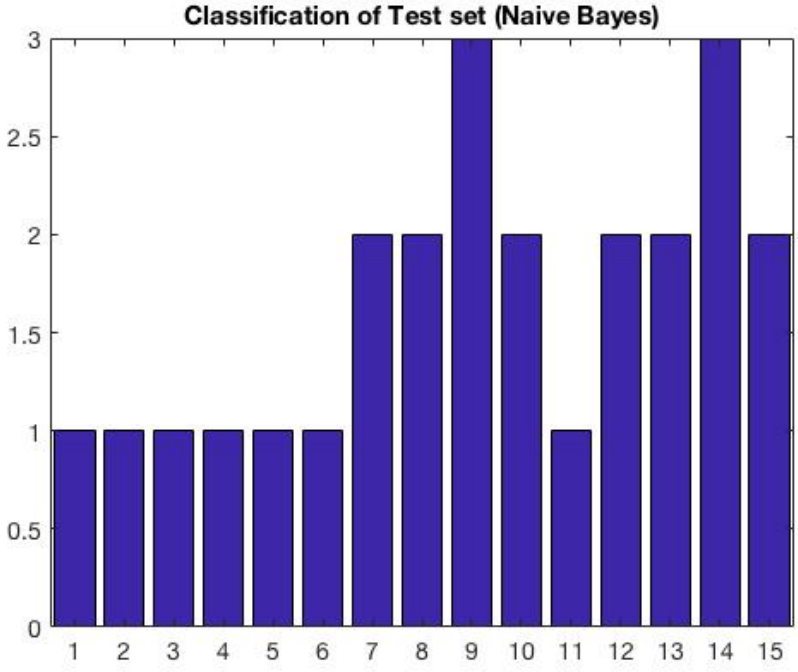

Figure 17: Classification for Test 3 Set (NB)