# AMATH 582 Homework 4

Kyle Schultz*

March 6, 2020

**Abstract**

In this project, we first explore SVD for dimension reduction of faces. We used both cropped and uncropped photos from Yale, and computed the SVD for various subsets of the face data. We found that SVD could be used to reduce dimension significantly, up to 86% for cropped images of the same person. Next, we classify music using SVD to determine the dominant directions to classify along. We performed supervised learning to classify the music, achieving results significantly better than a baseline.

## 1 Introduction and Overview

### 1.1 Yale Faces

In this part of the project, we explore the use of SVD to reduce the dimension of a set of faces. We first consider the cropped data set and compute the SVD using 38 different subjects with the same light/expression. We then take the SVD of 64 photos of the same person with different lighting and facial expressions. Lastly, we explore the SVD on uncropped photos of subjects.

### 1.2 Music Classification

People can easily recognize the genre and band of songs. Our goal in this part of the project is to recreate this ability with software. We split this goal in to three parts:

1. **Classifying artists in different genres:** for this test, we classified the band of a given song, with the artists in different genres. We chose three bands: Mac Miller (Rap), Avicii (EDM), and Khalid (R&B).
2. **Classifying artists in the same genre:** here, we repeated the above test, but with artists in the same genre. Three rap artists were chosen: Mac Miller, Lil Yachty, and Drake.
3. **Classify genre from any artist:** in this last test, we classified genre instead of artist. We chose various music clips from different artists in rap, EDM, and classical genres.

## 2 Theoretical Background

### 2.1 Singular Value Decomposition

Singular value decomposition (SVD) is an algorithm that expresses any matrix in terms of three matrices: U, S, V (equation 1).

$$A = USV^* \tag{1}$$

In equation 1, matrices U and V are unitary and S has real, positive values on its diagonal. When we are performing the SVD on yalefaces, we used a 'fat' matrix: each row holds the pixel data of a different image. In this case, the columns of V are the dominant modes, U is how the data projects onto the modes in V,

*Code available on Github: https://github.com/kyleschultz0/

and S are the singular values. These give us a weighting of how important the dominant modes in V are. In the song classification, we used a 'skinny' matrix where the columns hold the song data. In this case the interpretation of U and V are flipped. We exploit this idea that certain modes are dominant to effectively classify music in part 2 of this project. If we compute $U^T X$ where X is the spectrum of our song, we can extract the dominant modes in songs giving us a way to effectively compare and classify them.

### 2.2 Gabor Transform

To classify the songs, we use the Gabor Transform which is a windowed Fourier Transform. The Gabor Transform gives us the spectral content at different time steps, which is the basis for how we compute the spectogram of our data. This is implemented using MatLab's `spectrogram` command (appendix A).

$$G_x(\tau, \omega) = \int_{-\infty}^{\infty} x(t) e^{-\pi(t-\tau)^2} e^{-j\omega t}\, dt \tag{2}$$

### 2.3 Supervised Learning

To classify songs, we project our training and test sets onto the principal components found by taking the SVD of the training set. By splitting our data into test and training sets, we can determine how accurate our algorithm is at classifying new data. The training set has no a posteriori knowledge of the test set from either the spectrogram or SVD. Since we know which songs we used to construct our test and training sets, we label our training data and are therefore performing supervised learning. We also know the true class of the test set which allows us to objectively determine performance.

We used the built in MatLab function `classify` to perform the supervised learning classification. This algorithm fits a normal multivariate density to each class in order to classify the test set. More details on `classify` are in appendix A.

## 3 Algorithm Implementation and Development

### 3.1 Music Classification

The method used in the algorithm to classify music is described here. The general outline of the algorithm is:

1. Import songs using `audioread` (see appendix A)
2. Sample each song 10 times for 5 seconds (algorithm 1)
3. Compute the spectrum of samples and store (algorithm 1)
4. Construct test and training data sets
5. Compute SVD of training set
6. Project test and training sets onto principles components from SVD, $U^T X$
7. Classify music using `classify` (see appendix A)

## 4 Computational Results

### 4.1 Yale Faces

We first consider the cropped data set and compute the SVD using 38 different subjects with the same light/expression. The singular values and faces projected onto the first 6 principle components is shown in figure 2 and figure 1. We see that the different components represent dominant facial structures, such as nose shape, facial hair, eye size, and jawline. We also note that there is one singular value that is significant

**Algorithm 1:** Creating frequency spectrum from song samples

```
for i = 1 : NumberSongs do
    Import song i
    for j = 1 : NumberSamples do
        Sample song i from 10j to 10j + 5 secs
        Compute spectrum of sample j
        Reshape spectrum j to column vector
        Store spectrum j in column j of matrix i
    end for
end for
```

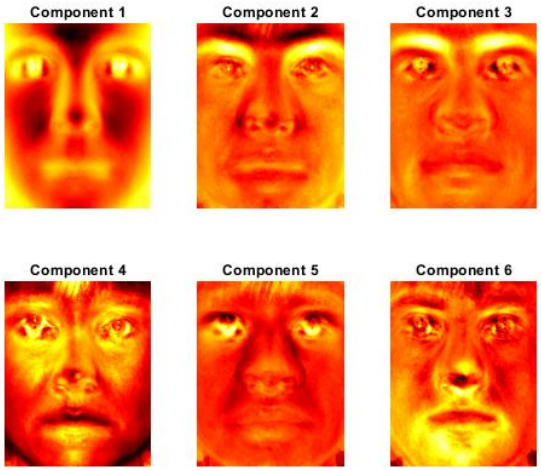


Figure 1: Projection of faces on first 6 principle components

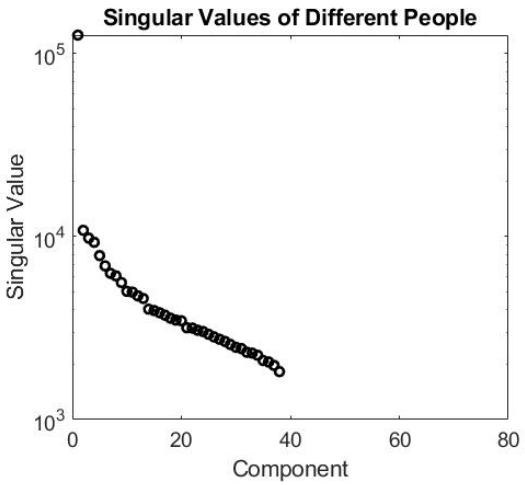


Figure 2: Singular values of faces (log-scale)

compared to the others. The first component represents the very fundamental facial structures: we all have mouths, noses, eyes et cetera. However, we also see that the singular values to not diminish quickly. If we

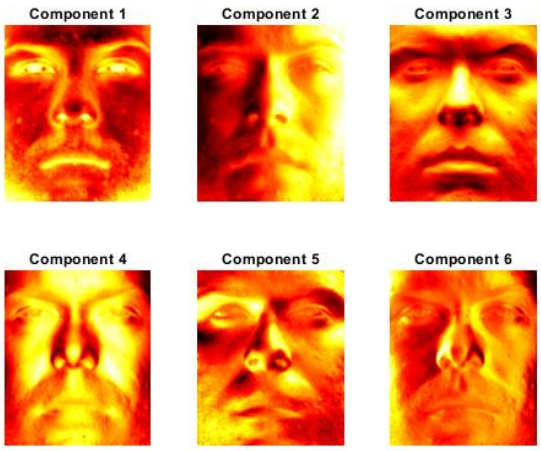


Figure 3: Projection of faces on first 6 principle components

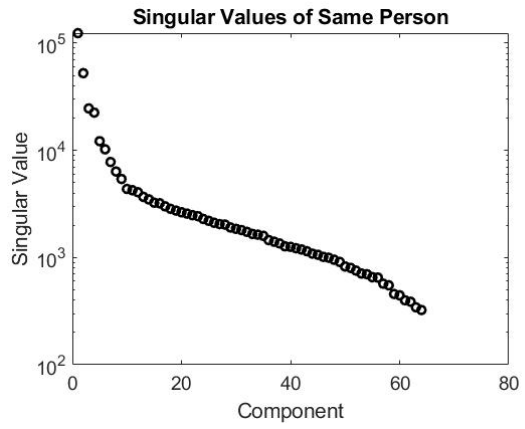


Figure 4: Singular values of faces (log-scale)

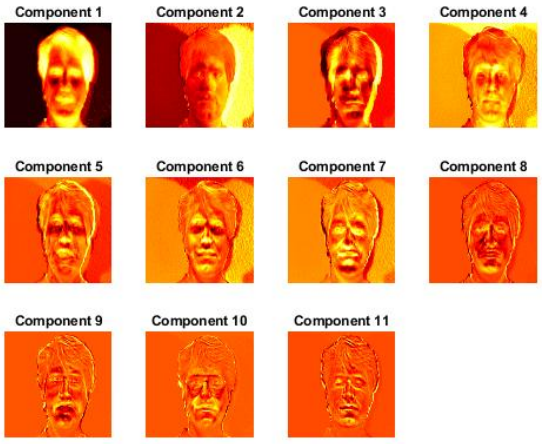


Figure 5: Projection of onto all 11 components

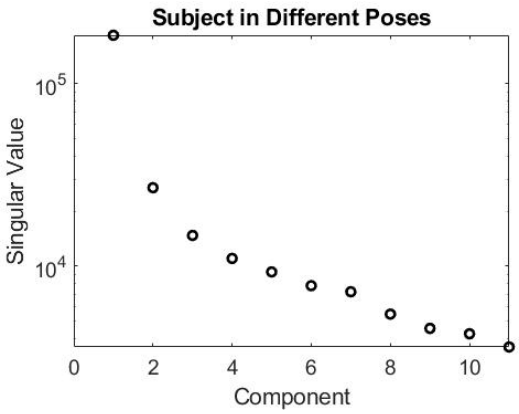


Figure 6: Singular values of faces (log-scale)

plot reduced rank approximations of the face space, we see that almost all 38 modes are required to recreate the faces accurately. This is a testament to how unique the human face is, and hence why biometric security is used.

Now, we consider the same face in different lighting/expressions. We can see from figure 4 that the values decrease more rapidly than the those in figure 2. The components in figure 3 represent a range of lighting conditions and expression: component 2 represents only 1 half of the face illuminated, component 3 represents closing of eyes and so on. We were able to accurately represent the faces with 9 modes, a reduction in rank of 86%!

We now compute the SVD for the uncropped faces. We do this for two data sets, the first consisting of 11 different scenarios of the same person. The second data set is 15 different subjects in the same pose. To capture the images correctly in the first and second case, we require all of the SVD modes. This shows that these face spaces are truly full rank.

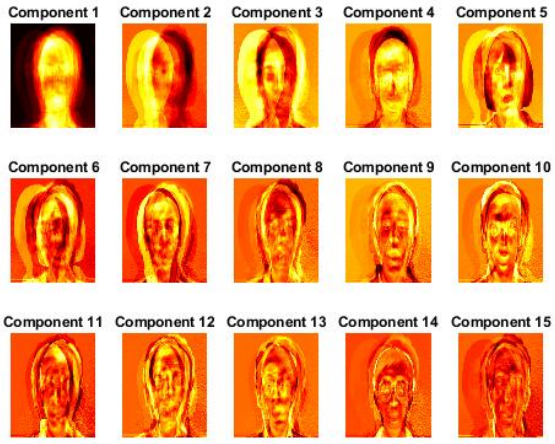


Figure 7: Projection of onto all 15 components

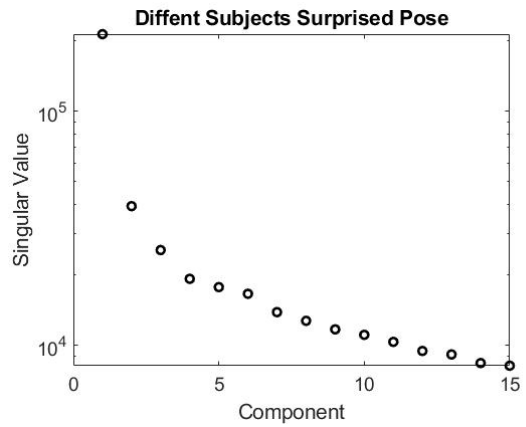


Figure 8: Singular values of faces (log-scale)

## 4.2 Music Classification

In our first test, we classified artists from three different genres. Two songs from each artist were used to create the training set and a third song was used to test the classification. We sampled each song 10 times for 5 seconds, with sample start times at 10 second increments. This yields 60 total samples in our training set and 30 total samples in our test set.

After performing SVD on our training data, we plot the projection of the data onto the second, third, and fourth principle component to get an idea of the clustering. We can see that there is overlap among the sets in the middle of the plot. Figure 4 shows the result of the classification. In this test, we achieve 63.3% accuracy which is significantly better than random chance (33.3%). This could likely be improved by expanding the training set to more than 2 songs. It is also worth noting that all of these artists are modern and relatively mainstream; therefore they may all have elements of pop music that make classification more difficult.

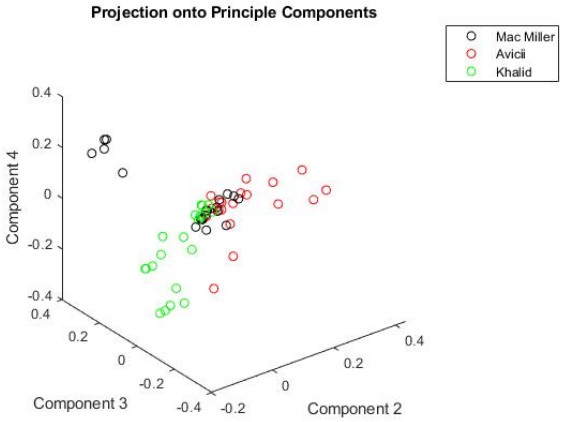


Figure 9: Test data projected onto principle components

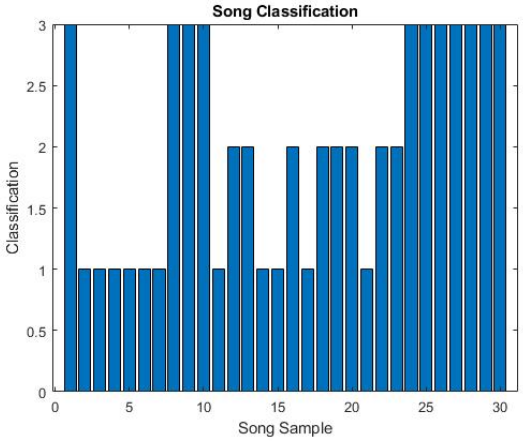


Figure 10: Test 1 classification results. Truth: songs 1-10, class 1; songs 11-20, class 2; songs 21-30, class 3

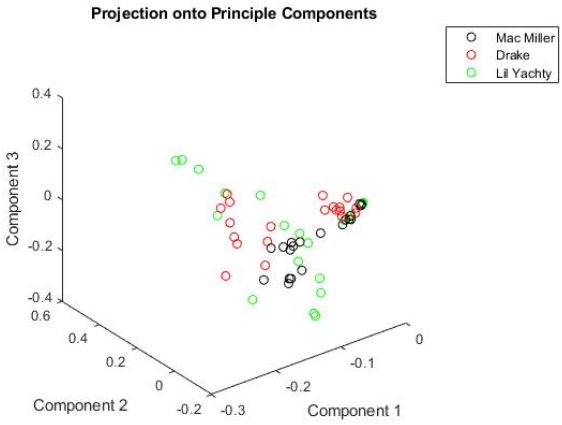


Figure 11: Test data projected onto principle components

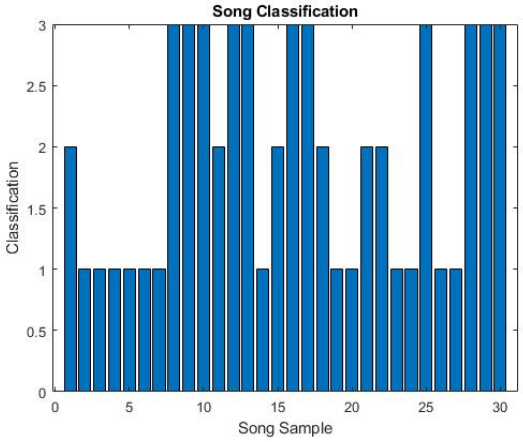


Figure 12: Test 2 classification results. Truth: songs 1-10, class 1; songs 11-20, class 2; songs 21-30, class 3

In test 2 (figures 9 and 10), we used the same procedure as in test 1, but with artists in the same genre. This makes classification far more difficult as the music shares many common features. Figure 5 shows how

the songs project onto their principal components, and more interclass overlap versus test 1. We achieved 43.3% accuracy which is expectedly lower than the accuracy in test 1, but nonetheless better than average. This also could have been increased by using more training data or picking artists that sound even less alike (ie from different time periods).

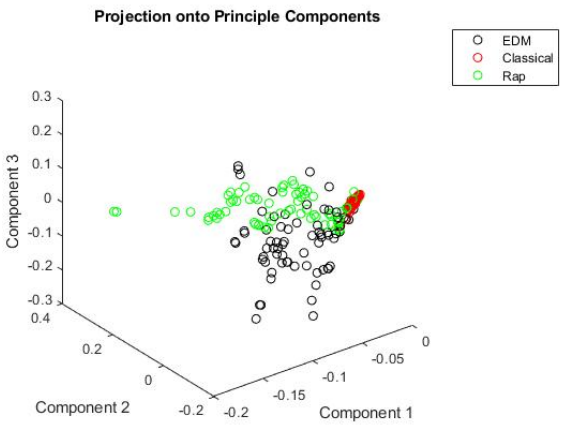


Figure 13: Test data projected onto principle components

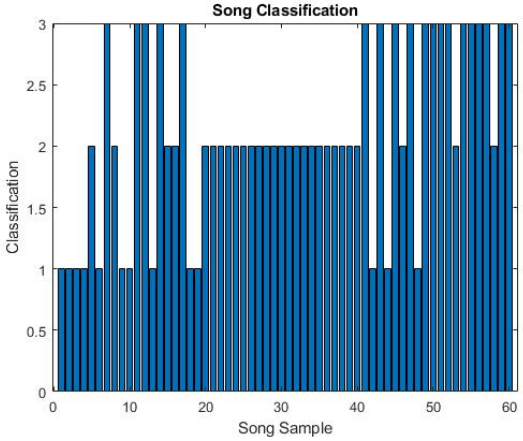


Figure 14: Test 3 classification results. Truth: songs 1-20, class 1; songs 21-40, class 2; songs 41-60, class 3

In test 3, we classified based on genre instead of band. We chose the genres EDM, classical, and rap. How long mashups of each type of music were downloaded to create test and training data, which allowed to create a larger training set than that used in parts 1 and 2. We took ninety 5 second samples that were spaced 30 seconds apart for each genre. We used the first samples for our training and the last 20 for our testing. Therefore our overall training set is 210 samples while our overall test set is 60 samples. Figure 7 shows how the first three principal components are projected onto. It is worth noting that classical clusters exceptionally well, likely because of the limited instruments used in classical musics.

Figure 8 shows the results of our classification. We achieved a 73.3% accuracy, the highest of any of the tests. It is somewhat deceiving to compare this to the other tests, however, as the genres were deliberately chosen to have good clustering (classical sounds nothing like modern rap). We also increased the size of the training set from 60 samples to 210 samples which also increased accuracy. Table 1 below summarizes the accuracy across all test.

| Test Case | Percentage Accuracy |
|---|---|
| 1 | 63.3% |
| 2 | 43.3% |
| 3 | 73.3% |

Table 1: Accuracy of classification tests

# 5 Summary and Conclusions

In part 1, we showed that the SVD can be used to project face data into dominant modes and reduce dimension. This was particularly successful when we took a large sample of the same subject in different conditions, where we achieved 86% reduction in rank. In cases where we used small data sets or faces from many different people, we did not achieve any reduction of dimension. This illustrates the complexity of the human face and that it requires many modes to be described. In part 2, we successfully classified music with various success rates all better than chance. We leveraged the SVD to project our music onto its dominant modes thus increasing our ability to classsify the music.

# Appendix A MATLAB Functions

Below are some of the most critical functions in our MATLAB implementation. Included are details pertinent to how we used them.

- `[U, S, V] = svd(A, 'econ')` returns the singular value decomposition of A, with economy returning only the relevant rows/columns to save storage and computation time
- `class = classify(test,training,group)` classifies each row of test using one of the labels in group.

# Appendix B MATLAB Code

```
%% AMATH 582 HW4

clc; clear all; close all;
fig = 1;

%% Create 4-D uint8 array with all cropped images for easy access

listing_cropped = dir('yalefaces_cropped/CroppedYale');

uncropped_subjects = extractfield(listing_cropped, 'name');
uncropped_subjects = uncropped_subjects(3:40);
size_cropped = size(uncropped_subjects);
format = 'yalefaces_cropped/CroppedYale/%s';

for i = 1:size_cropped(2)
    str = sprintf(format,char(uncropped_subjects(i)));
    subject_listing(:, :, i) = extractfield(dir(str), 'name');
end

subject_listing = subject_listing(:, 3:66, :);
size_subjects = size(subject_listing);
format = 'yalefaces_cropped/CroppedYale/%s/%s';

for i = 1:size_subjects(3)
    for j = 1:size_subjects(2)
        str = sprintf(format, char(uncropped_subjects(i)), char(subject_listing(:, j, i)));
        cropped_images(:, :, j, i) = imread(str);
    end
end

%% SVD on first image of all subjects

figure(fig); fig = fig+1;
im_size = size(cropped_images);
im_length = im_size(1)*im_size(2);
for i = 1:size_subjects(3)
    subplot(5,8,i)
    imshow(cropped_images(:, :, 1, i))
    image_reshaped(i, :) = reshape(cropped_images(:, :, 1, i), [1, im_length]);
end
sgtitle("All Subjects in Baseline Lighting")
```

```
[U, S, V] = svd(double(image_reshaped), 'econ');

for j = 30
    figure(fig); fig = fig+1;
    im_rank1 = U(:, 1:j)*S(1:j, 1:j)*V(:, 1:j)';
    im_rank1s = reshape(im_rank1, [38, 192, 168]);
    for i = 1:size_subjects(3)
        subplot(5,8,i)
        imshow(squeeze(uint8(im_rank1s(i, :, :))))
    end
end

figure(fig); fig = fig+1;
for i = 1:6
    subplot(2, 3, i)
    Vt1 = reshape(V(:, i), 192, 168);
    Vt2=Vt1(192:-1:1,:);
    pcolor(Vt2), shading interp, colormap hot
    title(sprintf("Component %i", i))
    set(gca,'Xtick',[],'Ytick',[])
end

figure(fig); fig = fig+1;
semilogy(diag(S),'ko','Linewidth',[2])
set(gca,'Fontsize',[14],'Xlim',[0 80])
title("Singular Values of Different People")
ylabel("Singular Value")
xlabel("Component")

%% SVD on first subject

figure(fig); fig = fig+1;
for i = 1:size_subjects(2)
    subplot(8,8,i)
    imshow(cropped_images(:, :, i, 1))
    image_reshaped(i, :) = reshape(cropped_images(:, :, i, 1), [1, im_length]);
end
sgtitle("First Subject in Different Poses")
[U, S, V] = svd(double(image_reshaped), 'econ');

for j = 9
    figure(fig); fig = fig+1;
    im_rank1 = U(:, 1:j)*S(1:j, 1:j)*V(:, 1:j)';
    im_rank1s = reshape(im_rank1, [64, 192, 168]);
    for i = 1:size_subjects(2)
        subplot(8,8,i)
        imshow(uint8(squeeze(im_rank1s(i, :, :))))
    end
end

figure(fig); fig = fig+1;
for i = 1:6
    subplot(2, 3, i)
    Vt1 = reshape(V(:, i), 192, 168);
```

```
    Vt2=Vt1(192:-1:1,:);
    pcolor(Vt2), shading interp, colormap hot
    set(gca,'Xtick',[],'Ytick',[])
    title(sprintf("Component %i", i))
end

figure(fig); fig = fig+1;
semilogy(diag(S),'ko','Linewidth',[2])
set(gca,'Fontsize',[14],'Xlim',[0 80])

title("Singular Values of Same Person")
ylabel("Singular Value")
xlabel("Component")

%% Uncropped faces

clc; clear all; close all;

%% Same person, different  poses

fig = 1;

listing_uncropped = dir('yalefaces_uncropped/yalefaces');
uncropped_subjects = extractfield(listing_uncropped, 'name');
uncropped_subjects = uncropped_subjects(3:167);
size_cropped = size(uncropped_subjects);
format = 'yalefaces_uncropped/yalefaces/%s';


for i = 1:165
        str = sprintf(format, char(uncropped_subjects(i)));
        uncropped_images(:, :, i) = imread(str);
        uncropped_images_shape(i, :) = reshape(imread(str), [1, 243*320]);
end

figure(fig); fig = fig+1;

for i = 1:11
    subplot(3,4,i)
    imshow(uncropped_images(:, :, i))
end

[U,S,V] = svd(double(uncropped_images_shape(1:11, :)), 'econ');

figure(fig); fig = fig+1;
for i = 1:11
    subplot(3, 4, i)
    Vt1 = reshape(V(:, i), 243, 320);
    Vt2=Vt1(243:-1:1,:);
    pcolor(Vt2), shading interp, colormap hot
    title(sprintf("Component %i", i))
    set(gca,'Xtick',[],'Ytick',[])
end
```

```
figure(fig); fig = fig+1;
semilogy(diag(S),'ko','Linewidth',[2])
set(gca,'Fontsize',[14],'Xlim',[0 11])
title("Subject in Different Poses")
ylabel("Singular Value")
xlabel("Component")

for j = 9
    figure(fig); fig = fig+1;
    im_rank1 = U(:, 1:j)*S(1:j, 1:j)*V(:, 1:j)';
    im_rank1s = reshape(im_rank1, [11, 243, 320]);
    for i = 1:11
        subplot(3,4,i)
        imshow(uint8(squeeze(im_rank1s(i, :, :))))
    end
end

%% Different people, same  poses

figure(fig); fig = fig+1;
for i = 1:15
    subplot(3,5,i)
    faces(:, :, i) = uncropped_images(:, :, 10+ 11*(i-1));
    imshow(faces(:, :, i))
end

faces_shape = reshape(faces, [243*320, 15]);
[U,S,V] = svd(double(faces_shape'), 'econ');

figure(fig); fig = fig+1;
for i = 1:15
    subplot(3, 5, i)
    Vt1 = reshape(V(:, i), 243, 320);
    Vt2=Vt1(243:-1:1,:);
    pcolor(Vt2), shading interp, colormap hot
    title(sprintf("Component %i", i))
    set(gca,'Xtick',[],'Ytick',[])
end

figure(fig); fig = fig+1;
semilogy(diag(S),'ko','Linewidth',[2])
set(gca,'Fontsize',[14],'Xlim',[0 15])
title("Subject in Different Poses")
ylabel("Singular Value")
xlabel("Component")

for j = 12
    figure(fig); fig = fig+1;
    im_rank1 = U(:, 1:j)*S(1:j, 1:j)*V(:, 1:j)';
    im_rank1s = reshape(im_rank1, [15, 243, 320]);
    for i = 1:15
        subplot(3,5,i)
        imshow(uint8(squeeze(im_rank1s(i, :, :))))
    end
```

```
end

%% Music Classificaitio ------------------------------------------------
clc; clear all; close all;
fig = 1;

%% Load Music

% Mac Miller (rap)-----------------------------------------------------

[mac1, Fs_mac1] = audioread("MacMiller_WhatsTheUse.mp3");
[mac2, Fs_mac2] = audioread("MacMiller_Ladders.mp3");
[mac3, Fs_mac3] = audioread("MacMiller_GoodNews.mp3");

% Avicii (EDM)---------------------------------------------------------

[avicii1, Fs_avicii1] = audioread("Avicii_TheNights.mp3");
[avicii2, Fs_avicii2] = audioread("Avicii _SOS.mp3");
[avicii3, Fs_avicii3] = audioread("Avicii_HeyBrother.mp3");

% Khalid (R&B)---------------------------------------------------------

[khalid1, Fs_khalid1] = audioread("Khalid_UpAllNight.mp3");
[khalid2, Fs_khalid2] = audioread("Khalid _Eleven.mp3");
[khalid3, Fs_khalid3] = audioread("Khalid_Better.mp3");

%% Create music 5 sec music vectors

Fs = Fs_mac1; % sample rate is same for all songs

t = 0:1/Fs:5;

% Truncated (5 sec) music samples

for i = 0:9
    mac1_truncated(:, i+1) = mac1((1+10*i)*Fs:length(t)+(1+10*i)*Fs, 1);
    mac2_truncated(:, i+1) = mac2((1+10*i)*Fs:length(t)+(1+10*i)*Fs, 1);
    mac3_truncated(:, i+1) = mac3((1+10*i)*Fs:length(t)+(1+10*i)*Fs, 1);

    avicii1_truncated(:, i+1) = avicii1((1+10*i)*Fs:length(t)+(1+10*i)*Fs, 1);
    avicii2_truncated(:, i+1) = avicii2((1+10*i)*Fs:length(t)+(1+10*i)*Fs, 1);
    avicii3_truncated(:, i+1) = avicii3((1+10*i)*Fs:length(t)+(1+10*i)*Fs, 1);

    khalid1_truncated(:, i+1) = khalid1((1+10*i)*Fs:length(t)+(1+10*i)*Fs, 1);
    khalid2_truncated(:, i+1) = khalid2((1+10*i)*Fs:length(t)+(1+10*i)*Fs, 1);
    khalid3_truncated(:, i+1) = khalid3((1+10*i)*Fs:length(t)+(1+10*i)*Fs, 1);
end



%% Spectrograms

for i = 1:10
    Spec_mac1(:, i) = reshape(spectrogram(mac1_truncated(:, i), 1.1025e+04), [8193*38 1]);
```

```
    Spec_mac2(:, i) = reshape(spectrogram(mac2_truncated(:, i), 1.1025e+04), [8193*38 1]);
    Spec_mac3(:, i) = reshape(spectrogram(mac3_truncated(:, i), 1.1025e+04), [8193*38 1]);

    Spec_avicii1(:, i) = reshape(spectrogram(avicii1_truncated(:, i), 1.1025e+04), [8193*38 1]);
    Spec_avicii2(:, i) = reshape(spectrogram(avicii2_truncated(:, i), 1.1025e+04), [8193*38 1]);
    Spec_avicii3(:, i) = reshape(spectrogram(avicii3_truncated(:, i), 1.1025e+04), [8193*38 1]);

    Spec_khalid1(:, i) = reshape(spectrogram(khalid1_truncated(:, i), 1.1025e+04), [8193*38 1]);
    Spec_khalid2(:, i) = reshape(spectrogram(khalid2_truncated(:, i), 1.1025e+04), [8193*38 1]);
    Spec_khalid3(:, i) = reshape(spectrogram(khalid3_truncated(:, i), 1.1025e+04), [8193*38 1]);
end

Spec_mac = [Spec_mac1 Spec_mac2 Spec_mac3];
Spec_avicii = [Spec_avicii1 Spec_avicii2 Spec_avicii3];
Spec_khalid = [Spec_khalid1 Spec_khalid2 Spec_khalid3];



%% Assembling matrix with all samples and taking SVD

Spec_mac_normal = abs(Spec_mac) - abs(mean(Spec_mac));
Spec_avicii_normal = abs(Spec_avicii) - abs(mean(Spec_avicii));
Spec_khalid_normal = abs(Spec_khalid) - abs(mean(Spec_khalid));
Spec_train = [Spec_mac_normal(:, 1:20) Spec_avicii_normal(:, 1:20) Spec_khalid_normal(:, 1:20)];
Spec_test = [Spec_mac_normal(:, 21:30) Spec_avicii_normal(:, 21:30) Spec_khalid_normal(:, 21:30)];


[U,S,V] = svd(Spec_train, 'econ');

figure(fig); fig = fig+1;
subplot(2,1,1)
plot(diag(S),'ko','Linewidth',[2])
set(gca,'Fontsize',[14],'Xlim',[0 80])
subplot(2,1,2)
semilogy(diag(S),'ko','Linewidth',[2])
set(gca,'Fontsize',[14],'Xlim',[0 80])

%% Classification

figure(fig); fig=fig+1;

plot3(V(1:20, 2), V(1:20, 3), V(1:20, 4), 'ko')
hold on
plot3(V(21:40, 2), V(21:40, 3), V(21:40, 4), 'ro')
hold on
plot3(V(41:60, 2), V(41:60, 3), V(41:60, 4), 'go')
legend("Mac Miller", "Avicii", "Khalid")
title("Projection onto Principle Components")
xlabel("Component 2"); ylabel("Component 3"); zlabel("Component 4");


figure(fig); fig=fig+1;
train1 = ((U(:, 2:6))'*Spec_train)';
label1 = [1*ones(20, 1); 2*ones(20,1); 3*ones(20,1)];
```

```
test1 = ((U(:, 2:6))'*Spec_test)';
class1 = classify(test1, train1, label1);
bar(class1);
title("Song Classification")
xlabel("Song Sample")
ylabel("Classification")

truth = [1*ones(10, 1); 2*ones(10,1); 3*ones(10,1)];

num_correct_case1 = 0;

for i = 1:length(class1)
    if class1(i) == truth(i)
        num_correct_case1 = num_correct_case1 + 1;
    end
end

percent_correct_1 = (num_correct_case1/length(class1))*100;




%% Bands of same genre (rap)

% Reusing Mac Miller

% Drake ------------------------------------------------------------

drake1 = audioread("Drake_Nonstop.mp3");
drake2 = audioread("Drake_InMyFeelings.mp3");
drake3 = audioread("Drake_GodsPlan.mp3");

% Lil Yachty-------------------------------------------------------------

yachty1 = audioread("Lil Yachty_ImTheMac.mp3");
yachty2 = audioread("Lil Yachty_Broccoli.mp3");
yachty3 = audioread("Lil Yachty_1Night.mp3");

%% Create music 5 sec music vectors

% Truncated (5 sec) music samples

for i = 0:9

    drake1_truncated(:, i+1) = drake1((1+10*i)*Fs:length(t)+(1+10*i)*Fs, 1);
    drake2_truncated(:, i+1) = drake2((1+10*i)*Fs:length(t)+(1+10*i)*Fs, 1);
    drake3_truncated(:, i+1) = drake3((1+10*i)*Fs:length(t)+(1+10*i)*Fs, 1);

    yachty1_truncated(:, i+1) = yachty1((1+10*i)*Fs:length(t)+(1+10*i)*Fs, 1);
    yachty2_truncated(:, i+1) = yachty2((1+10*i)*Fs:length(t)+(1+10*i)*Fs, 1);
    yachty3_truncated(:, i+1) = yachty3((1+10*i)*Fs:length(t)+(1+10*i)*Fs, 1);

end
```

```
%% Spectrograms

for i = 1:10

    Spec_drake1(:, i) = reshape(spectrogram(drake1_truncated(:, i), 1.1025e+04), [8193*38 1]);
    Spec_drake2(:, i) = reshape(spectrogram(drake2_truncated(:, i), 1.1025e+04), [8193*38 1]);
    Spec_drake3(:, i) = reshape(spectrogram(drake3_truncated(:, i), 1.1025e+04), [8193*38 1]);

    Spec_yachty1(:, i) = reshape(spectrogram(yachty1_truncated(:, i), 1.1025e+04), [8193*38 1]);
    Spec_yachty2(:, i) = reshape(spectrogram(yachty2_truncated(:, i), 1.1025e+04), [8193*38 1]);
    Spec_yachty3(:, i) = reshape(spectrogram(yachty3_truncated(:, i), 1.1025e+04), [8193*38 1]);
end

Spec_drake = [Spec_drake1 Spec_drake2 Spec_drake3];
Spec_yachty = [Spec_yachty1 Spec_yachty2 Spec_yachty3];



%% Assembling matrix with all samples and taking SVD

Spec_drake_normal = abs(Spec_drake) - abs(mean(Spec_drake));
Spec_yachty_normal = abs(Spec_yachty) - abs(mean(Spec_yachty));

Spec_train = [Spec_mac_normal(:, 1:20) Spec_drake_normal(:, 1:20) Spec_yachty_normal(:, 1:20)];
Spec_test = [Spec_mac_normal(:, 21:30) Spec_drake_normal(:, 21:30) Spec_yachty_normal(:, 21:30)];


[U,S,V] = svd(Spec_train, 'econ');

figure(fig); fig = fig+1;
subplot(2,1,1)
plot(diag(S),'ko','Linewidth',[2])
set(gca,'Fontsize',[14],'Xlim',[0 80])
subplot(2,1,2)
semilogy(diag(S),'ko','Linewidth',[2])
set(gca,'Fontsize',[14],'Xlim',[0 80])

%% Classification

figure(fig); fig=fig+1;

plot3(V(1:20, 1), V(1:20, 2), V(1:20, 3), 'ko')
hold on
plot3(V(21:40, 1), V(21:40, 2), V(21:40, 3), 'ro')
hold on
plot3(V(41:60, 1), V(41:60, 2), V(41:60, 3), 'go')
legend("Mac Miller", "Drake", "Lil Yachty")
title("Projection onto Principle Components")
xlabel("Component 1"); ylabel("Component 2"); zlabel("Component 3");

figure(fig); fig=fig+1;
train1 = ((U(:, 2:5))'*Spec_train)';
label1 = [1*ones(20, 1); 2*ones(20,1); 3*ones(20,1)];
test1 = ((U(:, 2:5))'*Spec_test)';
```

```
class2 = classify(test1, train1, label1);
bar(class2);
title("Song Classification")
xlabel("Song Sample")
ylabel("Classification")

truth = [1*ones(10, 1); 2*ones(10,1); 3*ones(10,1)];

num_correct_case2 = 0;

for i = 1:length(class2)
    if class2(i) == truth(i)
        num_correct_case2 = num_correct_case2 + 1;
    end
end

percent_correct_2 = (num_correct_case2/length(class2))*100;



%% Different Genres

% Hour long rap mix
rap = audioread("rap_1hr.mp3");

% Hour long EDM mix
edm = audioread("house_1hr.mp3");

% Hour long classical mix
classical = audioread("classical_1hr.mp3");

%% 5 second samples

for i = 0:89

    edm_truncated(:, i+1) = edm((1+30*i)*Fs:length(t)+(1+30*i)*Fs, 1);
    classical_truncated(:, i+1) = classical((1+30*i)*Fs:length(t)+(1+30*i)*Fs, 1);
    rap_truncated(:, i+1) = rap((1+30*i)*Fs:length(t)+(1+30*i)*Fs, 1);

end

%% Spectograms

for i = 1:90

    Spec_edm(:, i) = reshape(spectrogram(edm_truncated(:, i), 1.1025e+04), [8193*38 1]);
    Spec_classical(:, i) = reshape(spectrogram(classical_truncated(:, i), 1.1025e+04), [8193*38 1]);
    Spec_rap(:, i) = reshape(spectrogram(rap_truncated(:, i), 1.1025e+04), [8193*38 1]);

end


%% Assembling matrix with all samples and taking SVD
```

```
Spec_edm_normal = abs(Spec_edm) - abs(mean(Spec_edm));
Spec_classical_normal = abs(Spec_classical) - abs(mean(Spec_classical));
Spec_rap_normal = abs(Spec_rap) - abs(mean(Spec_rap));
Spec_train = [Spec_edm_normal(:, 1:70) Spec_classical_normal(:, 1:70) Spec_rap_normal(:, 1:70)];
Spec_test = [Spec_edm_normal(:, 71:90) Spec_classical_normal(:, 71:90) Spec_rap_normal(:, 71:90)];

[U,S,V] = svd(Spec_train, 'econ');

figure(fig); fig = fig+1;
subplot(2,1,1)
plot(diag(S),'ko','Linewidth',[2])
set(gca,'Fontsize',[14],'Xlim',[0 80])
subplot(2,1,2)
semilogy(diag(S),'ko','Linewidth',[2])
set(gca,'Fontsize',[14],'Xlim',[0 80])

%% Classification

figure(fig); fig=fig+1;

plot3(V(1:70, 1), V(1:70, 2), V(1:70, 3), 'ko')
hold on
plot3(V(71:140, 1), V(71:140, 2), V(71:140, 3), 'ro')
hold on
plot3(V(141:210, 1), V(141:210, 2), V(141:210, 3), 'go')
legend("EDM", "Classical", "Rap")
title("Projection onto Principle Components")
xlabel("Component 1"); ylabel("Component 2"); zlabel("Component 3");

figure(fig); fig=fig+1;
train1 = ((U(:, 1:3))'*Spec_train)';
label1 = [1*ones(70, 1); 2*ones(70,1); 3*ones(70,1)];
test1 = ((U(:, 1:3))'*Spec_test)';
class3 = classify(test1, train1, label1);
bar(class3);
title("Song Classification")
xlabel("Song Sample")
ylabel("Classification")

truth = [1*ones(20, 1); 2*ones(20,1); 3*ones(20,1)];

num_correct_case3 = 0;

for i = 1:length(class3)
    if class3(i) == truth(i)
        num_correct_case3 = num_correct_case3 + 1;
    end
end

percent_correct_3 = (num_correct_case3/length(class3))*100;
```