Extended Yale Faces B Database – Eigenfaces & Music Genre Identification

Song Genre and Band Classification using Machine Learning and Feature Selection

Dino de Raad

April 11, 2019

Abstract

Faces and music are analyzed via Singular Value Decomposition techniques. Supervised learning is used to classify musical genre or band from unknown clips.

I Introduction and Overview

In this paper the Singular Value Decomposition and supervised learning techniques are applied to two datasets. Specifically, a general view of the Eigenface space is given as an overview of what the SVD can bring to learning approaches. Then, the SVD is applied to supervised learning in a music classification study.

II Theoretical Background

The Singular Value Decomposition

The Singular Value Decomposition is a matrix decomposition technique which takes the intuitive notion that a matrix rotates, scales, and projects a vector by the action of multiplication and converts these three operations into matrices; the matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ can be written as

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

where $\mathbf{U} \in \mathbb{R}^{m\times m}$ is an orthonormal set of basis vectors, $\mathbf{\Sigma} \in \mathbb{R}^{m\times n}$ are the r singular values along the diagonal, and $\mathbf{V}^* \in \mathbb{R}^{n\times n}$ is an orthonormal set of basis vectors. For a more detailed explanation of the Singular Value Decomposition, refer to paper 3.

The Eigenfaces

We consider a set of images of faces. Each image is reshaped into a column and put into the matrix $\mathbf{X}$. We take the SVD of $\mathbf{X}$:

$$\mathbf{X} = \mathbf{U_X \Sigma_X V_X^*}$$

and consider what these matrices and their products mean in context.

1. $\mathbf{U}_X$ are the basis vectors for the Eigenface space.

2. $\mathbf{\Sigma}_X$ are a measure of how well each of the basis vectors account for the variance in the data.

3. The column vectors of $\mathbf{V_X}$ give the linear combination of the weighted basis vectors ($\mathbf{U_X\Sigma_X}$) that return the original faces in $\mathbf{X}$.

In theory if these faces are closely related, and with a large enough completely noise-free data, then the magnitude of the singular values will decrease significantly and a rank-r truncation will occur naturally. However, since the data-set is likely to be noisy and the available data is meaningful but small, it is unlikely that $\mathbf{\Sigma}_X$ will be of rank-r where r is much smaller than m or n. However, this does not mean that in application there will be a point after which additional singular values do not change the result of the reconstructed faces using a truncation of order r', where r' is chosen at some reasonable threshold of variance captured.

The Eigenspectra

We consider the task of classifying music from 5 second audio clips. In a single second clip of `mp3` data, 44100 single precision numbers are used for each channel to determine the audio. We Fourier transform the signal and then take each sample to be a column of $\mathbf{X}$. The principle components of $\mathbf{X}$ give coordinates with which we can differentiate each song, considering each song as point in principle component space. Presumably, songs by the same artist or in the same genre will cluster in some way, allowing machine learning techniques to easily classify the data and label new songs.

Supervised Learning

Three machine learning algorithms are used to try to classify the data. Linear Discriminant, Quadratic Discriminant, and the Naive Bayes. Linear Discriminant models choose a line that best separates the data along some optimal projection. The quadratic model does the same but instead separates the data with a quadratic boundary. The Naive Bayes approach assumes that every category of the data is independent and attempts to classify it taking into account that independence.

III Algorithm Implementation and Development

Algorithm 1: Faces

This algorithm finds the SVD of the face space, giving the eigenfaces.

1. For each of your faces (images):
   - Reshape them into a column vector.
   - Make these the columns of `X`.
2. Take the `svd` of `X`.
3. The reshaped columns of `u` are the eigenfaces.
4. `u*X'` are the principle components.
5. Truncating `s` and constructing `X=u*strunc*v.'` gives the r-truncated `X`.

Algorithm 2: Music Classification

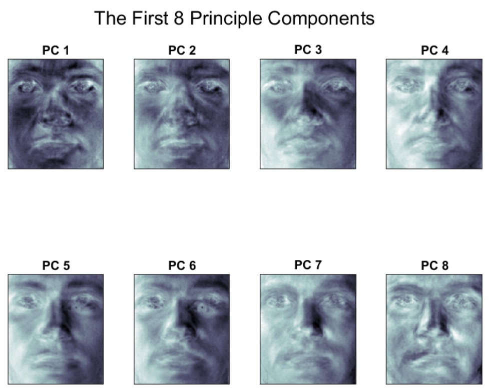
This algorithm takes music clips and preprocesses them for machine learning.

1.

IV Computational Results

Part 1: Faces

Figures 1, 2, 3, and 4 refer to the Yale Faces datasets. The photoIDs given assume 64 faces per person (some of these are missing) and are just the index of the face if all listed, ordered first by the Yale face id and next



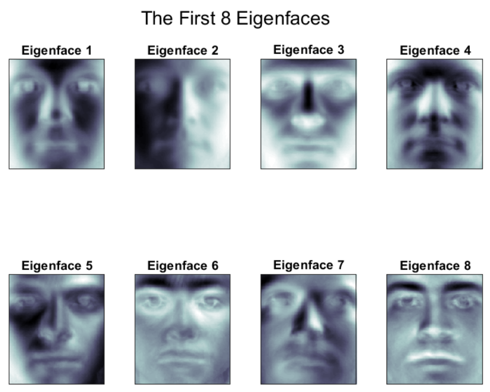
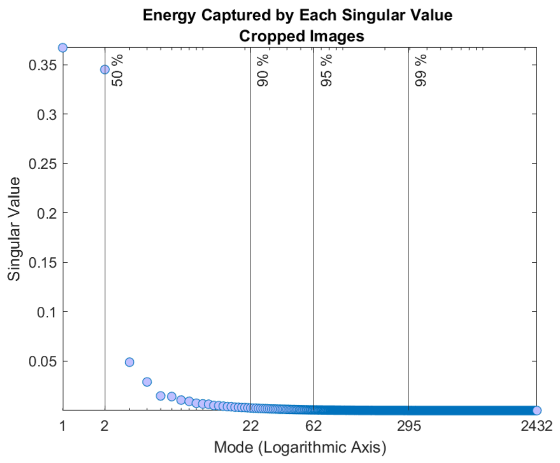


Figure 1: Cropped Dataset. On the left, the first 8 Principle Component Projections of the data matrix **X** onto the Eigenface Basis on the right are shown. Clear features are evident in the Eigenface basis; several eye, nose, and mouth shapes as well as other distinctive features in the facial structure (cheekbone height, eye height, etc) can be seen in various amounts in each vector. The two dominant modes appear to create a general facial structure (Eigenface 1) and left/right lighting (Eigenface 2).



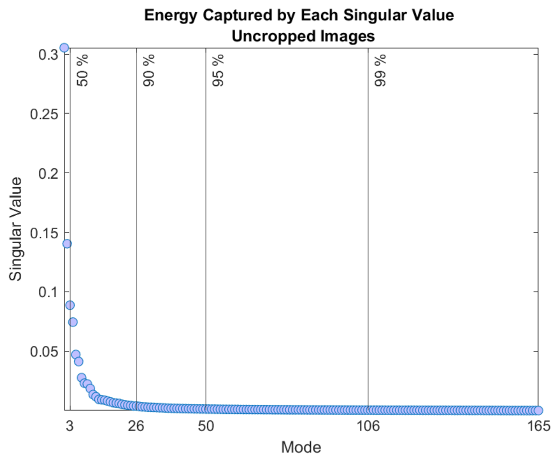
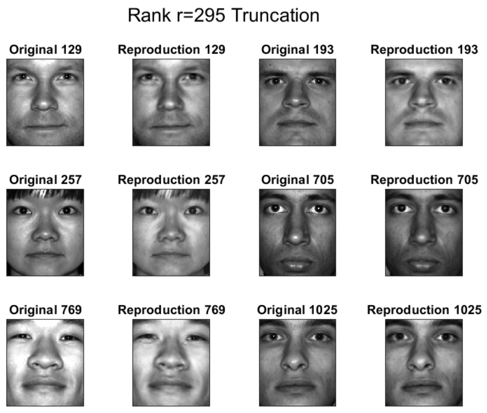


Figure 2: Cropped and Uncropped Datasets. Singular Value Spectra of each of the datasets, Cropped plotted with a logarithmic x axis for readability. While it may initially seem that it takes many more modes to plot the cropped images, consider that to capture 99% of the variance in the cropped images, we need only 12% of the modes while in the case of the uncropped we need 64%. This is a dramatic difference in compression. Not only that, but for most reasonable applications 95% accuracy may be desirable and this is achievable with as little as 2.5% of the available modes in the cropped set, while we would still need 30% for the uncropped set.

by the natural lexicographical ordering of a Windows machine. Overall, the cropped data set gives a much nicer basis to work with (the eigenfaces of the uncropped set are difficult to distinguish or even discern as faces, and look like a blurry mess). While it is still possible to compress these images and retain reasonable accuracy (Figure 4), the degree of compression is nowhere near as much as for the first dataset (Figure 2).

Part 2: Music Classification

For samples of size 50 and with 10 test clips, the algorithm had the following accuracies in each of the tests. Principle component projections (Figures 5, 6) were chosen based on how well the linear discriminant model



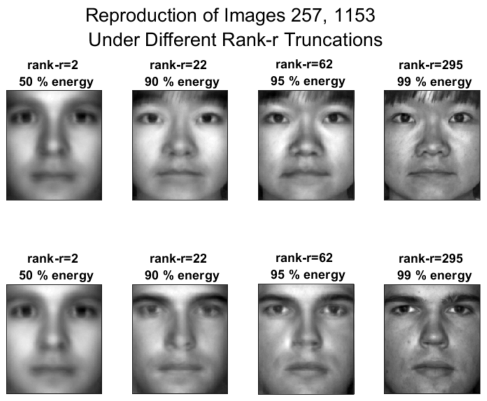
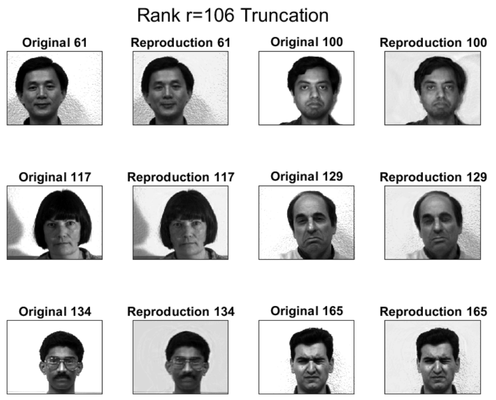


Figure 3: Cropped Dataset. Reconstructions of 7 of the original faces (ID:257 repeated in the left and right subfigure). On the left, 6 faces are reconstructed using a rank-r truncation of 295 out of a potential 2432 modes, capturing 99% of the variance and creating near-perfect reconstruction. The main differences are in the brightest and darkest parts of the image. On the right, we see that with just 22 modes, faces can easily be differentiated, and with 62 modes the faces are distinct enough to tell who the original model is. 295 modes are good enough to reconstruct details like acne, lighting, strands of hair, and other very fine details.



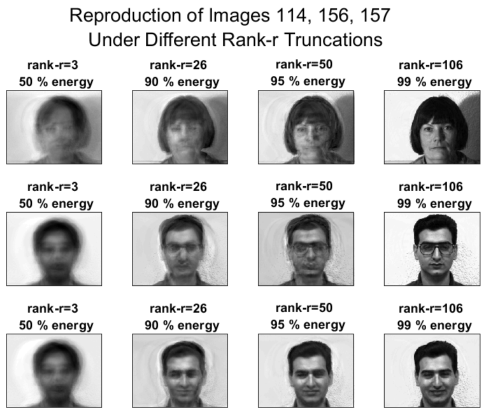


Figure 4: Uncropped Dataset. Reconstructions of 6 of the original models, but 9 of the original images (the models on the right appear several times). On the left, we see that with our 99% variance truncation of rank-$r = 106$ works decently well in some pictures. Pictures without glasses and with neutral or close-to-neutral facial expressions, like 61, 100, 117, and 165 are fairly well recreated. However, in 134 the eye region of the model is completely muddled due to his glasses, and 129's frowning expression is muted to look more neutral, and again the eyes are poorly defined. This is illustrated well by the subfigure on the right, which shows the faces under different rank-r truncations. Image 114 is a fairly typical image for this set, and it is clear that not even a rank-50 truncation will be good beyond guessing gender, even though some more neutral images like 157 can be well approximated by 50 modes. In most cases, the 99% rank truncation will be good enough in application, but again this is not even good enough for the entire dataset as seen in the first subfigure. Finally, the presence of glasses as the only variation between 156 and 157 vastly changes the accuracy of the rank-50 truncation.

was able to classify the test data.

```
test 1
```

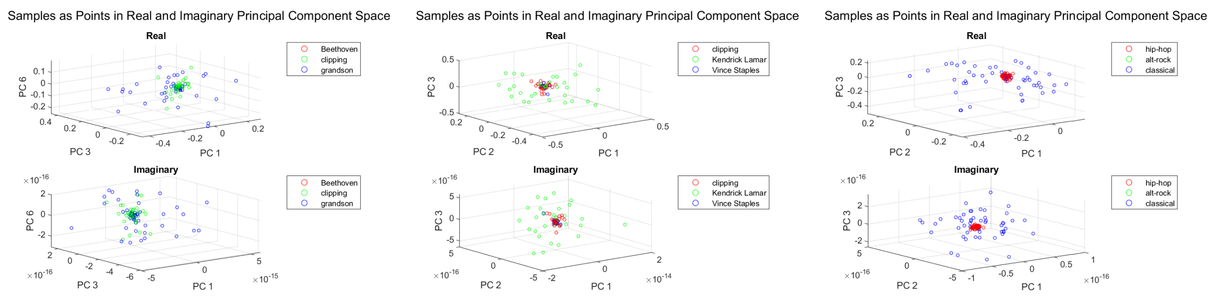


Figure 5: Music Tests 1-3. The imaginary components of these principle components are of incredibly low order so as to be ignored. All of the datasets have the musical types clustered concentrically about the origin as shells, almost irrespective of principal component direction.

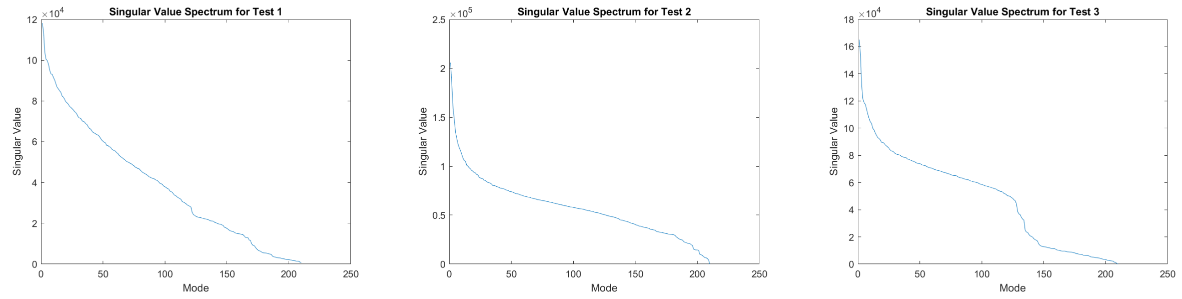


Figure 6: Music Tests 1-3. The spectra have distinctive characteristics depending on the types of songs mixed. When the songs are of very different genres, more of the singular values are of high magnitude (Leftmost, Rightmost). When the songs are similar, the curve is much smoother and the tail has no abrupt fall-off.

```
 LDM: 80.00
 QDM: 91.11
 NB: 91.11
test 2
 LDM: 77.78
 QDM: 78.89
 NB: 82.22
test 3
 LDM: 81.11
 QDM: 87.78
 NB: 88.89
```

Overall Naive Bayes tends to do better than the Linear Discriminant, and on par with the Quadratic model. In this particular instance (all of the graphs shown in Figures are for the tests with this outcome), the Naive Bayes outperformed the Quadratic, but this is not always the case. Finally we show the consistency of the Naive Bayes against itself in test over a number of trials (Figure **??**).

V Summary and Conclusions

The eigenface space is easy to intuitvely understand and forms a useful basis for techniques such as image compressiona and facial recognition. Additionally, these techniques can be applied to 1 dimensional data like the fft or 2 dimensional data like the spectrogram and can serve to classify pieces of audio with the help of machine learning techniques.

Appendix A MATLAB functions

```
function mn = mono(stereo)
mask = ~sum(stereo == 0, 2);
mn = sum(stereo, 2);
mn(mask) = mn(mask)/2;
end

function in = isin(name,namelist)
in = false;

for n = 1:length(namelist)
    nm = cell2mat(namelist(n));
    in = in | strcmp(name, nm);
    if in
        break
    end
end

end

function [Xtrain, ltrain, Xtest, ltest] = generatesamples(D,fldnames,train,test,samplelength,samplerate
sl = samplelength;
Xtrain = zeros([sl*samplerate train*length(fldnames)]);
Xtest = zeros([sl*samplerate test*length(fldnames)]);
lr = ones([train 1]);
ltrain = [1*lr; 2*lr; 3*lr];
ls = ones([test 1]);
ltest = [1*ls; 2*ls; 3*ls];

bnd = -1;

for i = 1:length(D)
    in = isin(D(i).name, fldnames);
    if in
        bnd = bnd + 1;
        SD = dir(sprintf('%s\\%s',D(i).folder,D(i).name));
        SD = SD(3:end);

        trainsamples = train;
        testsamples = test;

        for j = 1:length(SD)
            % We will sample from each of the songs available at random.
            if trainsamples == 0 || j-length(SD)==0
                trainthissong = trainsamples;
            else
                trainthissong = randi(trainsamples);
            end

            if testsamples == 0 || j-length(SD)==0
                testthissong = testsamples;
            else
                testthissong = randi(testsamples);
            end
            trainsamples = trainsamples - trainthissong;
            testsamples = testsamples - testthissong;
            filename = sprintf('%s\\%s',SD(j).folder,SD(j).name);
            disp(filename)
            info = audioinfo(filename);
            %[y, Fs] = audioread(filename);
            %t = linspace(0, length(y)/Fs, length(y));
            fiveseconds = samplelength*samplerate;
            samplestarts = randperm(info.TotalSamples-fiveseconds-1, trainthissong + testthissong);
            trainstarts = samplestarts(1:trainthissong);
            teststarts = samplestarts(trainthissong+1:end);

            traintaken = train-trainsamples-trainthissong;
            testtaken = test-testsamples-testthissong;

            for k1 = 1:length(trainstarts)
                stereo = audioread(filename, [trainstarts(k1) trainstarts(k1)+fiveseconds-1]);
                Xindex = k1+bnd*train+traintaken;
                Xtrain(:,Xindex) = mono(stereo);
            end
            for k2 = 1:length(teststarts)
                stereo = audioread(filename, [teststarts(k2) teststarts(k2)+fiveseconds-1]);
                Xindex = k2+bnd*test+testtaken;
                Xtest(:,Xindex) = mono(stereo);
            end
        end
    end
end

end
```

Appendix B MATLAB codes

hw4faces.m

```
close all; clear variables; clc

%% Load Cropped
% Load all cropped photos to put in X

n = 192;
m = 168;

crpdir = 'yalefaces_cropped\CroppedYale';
D = dir(crpdir);
D = D(3:end);

% The following chooses %numfolders% random folders and generates the faces
% from those only.
% numfolders = 4; % number of folders to grab pictures from
%
% faces = D(randperm(length(D), numfolders));
```

```
numfolders = length(D);
faces = D;

X = zeros([n*m 64*numfolders]); % assuming 64 pictures in each folder

for i = 1:numfolders % for each folder of faces
    FD = dir(sprintf('%s\\%s',crpdir,faces(i).name));
    FD = FD(3:end);

    fld = (i-1)*length(FD);

    for j = 1:length(FD) % read faces into a column of X
        im = imread(sprintf('%s\\%s',FD(j).folder,FD(j).name));
        X(:,j+fld) = reshape(im, [n*m 1]);
    end
end

%% SVD on Cropped


[M,N]=size(X); % compute data size
mn=mean(X,2); % compute mean for each row
Xmn = repmat(mn,1,N);
X=X-Xmn; % subtract mean
[u,s,v] = svd(X/sqrt(N-1),'econ');

sv = diag(s);
var = sv.^2;
energy = var./sum(var);
energyperPC = cumsum(energy);
plotenergies = [.50 .90 .95 .99];

[rw,cl] = find(energyperPC>plotenergies);

x = [rw(find(cl==1, 1))
     rw(find(cl==2, 1))
     rw(find(cl==3, 1))
     rw(find(cl==4, 1))];


%% Plotting Cropped

fig1 = figure(1);
semilogx(energy, 'o', 'MarkerFaceColor', [0.75 0.75 1])
title(sprintf('Energy Captured by Each Singular Value \n Cropped Images'))
ylabel('Singular Value')
xlabel('Mode (Logarithmic Axis)')
hold on
for i = 1:length(plotenergies)
    xline(x(i), '-', {sprintf('%.0f %%', 100*plotenergies(i))});
end
xticks([1 x' length(energy)])
axis tight
```

```
Y = u*X';

fig2 = figure(2);
colormap(bone)
a = 2; b = 4;
sgtitle(sprintf('The First %d Principle Components', a*b))
for k = 1:a*b
    subplot(a,b,k)
    imagesc(reshape(Y(k,:), [n m])), axis image
    title(sprintf('PC %d', k))
    xticks([])
    yticks([])
end

fig8 = figure(8);

colormap(bone)
a = 2; b = 4;
sgtitle(sprintf('The First %d Eigenfaces', a*b))
for k = 1:a*b
    subplot(a,b,k)
    imagesc(reshape(u(:,k), [n m])), axis image
    title(sprintf('Eigenface %d', k))
    xticks([])
    yticks([])
end

fig3 = figure(3);

r = x(plotenergies==.99);
sgtitle(sprintf('Rank r=%d Truncation', r))
sv(r+1:end) = 0;
srtrunc = diag(sv);

Xrtrunc = u*srtrunc*v.' * sqrt(N-1);

colormap(gray)
a = 3; b = 4;
%b = numfolders; % randomized

% randomize which faces are
% printed
%faces2recon = sort(randperm(numfolders, a*b/2)); % randomized

faces2recon = [129 193 257 705 769 1025]; % specific faces for figure
for k = 1:2:a*b
    % conversion for randomized
    %imgi = ((faces2recon((k+1)/2)+1)/2-1)*64+1;
    imgi = faces2recon((k+1)/2); % specific faces for figure\

    subplot(a,b,k)
    imagesc(reshape(X(:,imgi) + Xmn(:,imgi), [n m])), axis image
    title(sprintf('Original %d', imgi))
    xticks([])
    yticks([])

    subplot(a,b,k+1)
    imagesc(reshape(Xrtrunc(:,imgi) + Xmn(:,imgi), [n m])), axis image
    title(sprintf('Reproduction %d', imgi))

    xticks([])
    yticks([])
end



fig4 = figure(4);
colormap(gray)

imgids = [257 1153];
sgtitle(sprintf('Reproduction of Images %d, %d \n Under Different Rank-r Truncations', imgids(1), imgids

for p = 1:length(plotenergies)
    sv = diag(s);
    r = x(p);
    sv(r+1:end) = 0;
    srtrunc = diag(sv);

    Xrtrunc = u*srtrunc*v.' * sqrt(N-1);

    for id = 1:length(imgids)
        subplot(length(imgids),length(plotenergies),p+(id-1)*length(plotenergies))
        imagesc(reshape(Xrtrunc(:,imgids(id)) + Xmn(:,imgids(id)), [n m])), axis image
        title(sprintf('rank-r=%d \n %.0f %% energy', r, 100*plotenergies(p)))


        xticks([])
        yticks([])
    end
end


%% Load Uncropped

n = 243;
m = 320;

uncrpdir = 'yalefaces_uncropped\yalefaces';
D = dir(uncrpdir);
D = D(3:end);

% numfaces = 20; % random
% faces = D(randperm(length(D), numfaces)); % random

faces = D; % all faces

X = zeros([n*m length(faces)]);
```

```
for j = 1:length(faces) % read faces into a column of X
    im = imread(sprintf('%s\\%s',faces(j).folder,faces(j).name), 'gif');
    X(:,j) = reshape(im, [n*m 1]);
end


%% SVD on Uncropped

[M,N]=size(X); % compute data size
mn=mean(X,2); % compute mean for each row
Xmn = repmat(mn,1,N);
X=X-Xmn; % subtract mean

[u,s,v] = svd(X/sqrt(N-1),'econ');


sv = diag(s);
var = sv.^2;
energy = var./sum(var);
energyperPC = cumsum(energy);
plotenergies = [.50 .90 .95 .99];

[rw,cl] = find(energyperPC>plotenergies);

x = [rw(find(cl==1, 1))
     rw(find(cl==2, 1))
     rw(find(cl==3, 1))
     rw(find(cl==4, 1))];

%% Plotting Uncropped

fig5 = figure(5);
plot(energy, 'o', 'MarkerFaceColor', [0.75 0.75 1])
title(sprintf('Energy Captured by Each Singular Value \n Uncropped Images'))
ylabel('Singular Value')
xlabel('Mode')
hold on
for i = 1:length(plotenergies)
    xline(x(i), '-', {sprintf('%.0f %%', 100*plotenergies(i))});
end
xticks([0 x' length(energy)])
axis tight

fig6 = figure(6);

r = x(plotenergies==.99);
sgtitle(sprintf('Rank r=%d Truncation', r))
sv(r+1:end) = 0;
srtrunc = diag(sv);

Xrtrunc = u*srtrunc*v.' * sqrt(N-1);
```

```
colormap(gray)
a = 3; b = 4;
%b = numfolders; % randomized

% randomize which faces are
% printed
%faces2recon = sort(randperm(length(D), a*b/2)); % randomized

faces2recon = [61 100 117 129 134 165]; % specific faces for figure
for k = 1:2:a*b
    % conversion for randomized
    imgi = faces2recon((k+1)/2);
    %imgi = faces2recon((k+1)/2); % specific faces for figure\

    subplot(a,b,k)
    imagesc(reshape(X(:,imgi) + Xmn(:,imgi), [n m])), axis image
    title(sprintf('Original %d', imgi))

    xticks([])
    yticks([])

    subplot(a,b,k+1)
    imagesc(reshape(Xrtrunc(:,imgi) + Xmn(:,imgi), [n m])), axis image
    title(sprintf('Reproduction %d', imgi))

    xticks([])
    yticks([])
end


fig7 = figure(7);
colormap(gray)

imgids = [114 156 157];
sgtitle(sprintf(...
'Reproduction of Images %d, %d, %d \n Under Different Rank-r Truncations',...
imgids(1), imgids(2), imgids(3)))

for p = 1:length(plotenergies)
    sv = diag(s);
    r = x(p);
    sv(r+1:end) = 0;
    srtrunc = diag(sv);

    Xrtrunc = u*srtrunc*v.' * sqrt(N-1);

    for id = 1:length(imgids)
        subplot(length(imgids),length(plotenergies),p+(id-1)*length(plotenergies))
        imagesc(reshape(Xrtrunc(:,imgids(id)) + Xmn(:,imgids(id)), [n m])), axis image
        title(sprintf('rank-r=%d \n %.0f %% energy', r, 100*plotenergies(p)))


        xticks([])
        yticks([])
    end
end

%% Saving

saveas(fig1,'svface1.png')
saveas(fig2,'PC1.png')
saveas(fig8,'PC2.png')
saveas(fig3,'recon1.png')
saveas(fig4,'recon2.png')
saveas(fig5,'svface2.png')
saveas(fig6,'recon3.png')
saveas(fig7,'recon4.png')
```

hw4music.m

```
close all; clear variables; clc

%% Directory
% https://ytmp3.cc/

mp3Fs = 44100; % Sample Rate, Hz

mdir = 'music';
D = dir(mdir);
D = D(3:end);

%% (test 1) Different Genre Bands
% Can we guess whether a clip is from one of the following different bands:
% Beethoven - classical
% clipping - experimental hip-hop
% grandson - alt-rock

bandnames = {'Beethoven','clipping','grandson'};
train = 60; % samples to train
test = 10; % samples to confirm
sl = 5; % sample length in seconds

tic
[Xtrain, ltrain, Xtest, ltest] = generatesamples(D,bandnames,train,test,sl,mp3Fs);
toc

Xtrain = fft(Xtrain, [], 1);
Xtest = fft(Xtest, [], 1);

X = [Xtrain Xtest];

%% SVD
[u, s, v] = svd(X - mean(X(:)), 'econ');

%% Classification
fig1 = figure(1);
plot(diag(s))
title('Singular Value Spectrum for Test 1')
ylabel('Singular Value')
```

```
xlabel('Mode')

% Y = u*X'; this creates a matrix that is big and useless
% We will form the first PCs columns of this matrix
% C(i,j) = A(i,:)*B(:,j)

PCs =[1 3 6]; % PC 3 for feature discrimination
%Xp = X';
%Y(:, PCs) = u(:,:)*Xp(:,PCs);

xtrain = v(1:size(Xtrain,2), PCs)'; %v(PCs, 1:size(Xtrain,2));
xtest = v(size(Xtrain,2)+1:end, PCs)'; %v(PCs, size(Xtrain,2)+1:end);

fig2 = figure(2);
sgtitle('Samples as Points in Real and Imaginary Principal Component Space')
subplot(2,1,1)
scatter3(real(xtrain(1,ltrain==1)), real(xtrain(2,ltrain==1)), real(xtrain(3,ltrain==1)), 10, [1,0,0])
hold on
scatter3(real(xtrain(1,ltrain==2)), real(xtrain(2,ltrain==2)), real(xtrain(3,ltrain==2)), 10, [0,1,0])
scatter3(real(xtrain(1,ltrain==3)), real(xtrain(2,ltrain==3)), real(xtrain(3,ltrain==3)), 10, [0,0,1])
title('Real')
xlabel(sprintf('PC %d', PCs(1)))
ylabel(sprintf('PC %d', PCs(2)))
zlabel(sprintf('PC %d', PCs(3)))
legend(bandnames)
subplot(2,1,2)
scatter3(imag(xtrain(1,ltrain==1)), imag(xtrain(2,ltrain==1)), imag(xtrain(3,ltrain==1)), 10, [1,0,0])
hold on
scatter3(imag(xtrain(1,ltrain==2)), imag(xtrain(2,ltrain==2)), imag(xtrain(3,ltrain==2)), 10, [0,1,0])
scatter3(imag(xtrain(1,ltrain==3)), imag(xtrain(2,ltrain==3)), imag(xtrain(3,ltrain==3)), 10, [0,0,1])
title('Imaginary')
xlabel(sprintf('PC %d', PCs(1)))
ylabel(sprintf('PC %d', PCs(2)))
zlabel(sprintf('PC %d', PCs(3)))
legend(bandnames)

lclass = classify(real(xtest'),real(xtrain'),ltrain);
qclass = classify(real(xtest'),real(xtrain'),ltrain,'quadratic');
truth = ltest;
t1lE = 100-sum((1/length(bandnames))*abs(lclass-truth))/(length(bandnames)*test)*100;
t1qE = 100-sum((1/length(bandnames))*abs(qclass-truth))/(length(bandnames)*test)*100;

Model = fitcnb(real(xtrain'),ltrain);
test_labels = predict(Model,real(xtest'));
t1nbE = 100-sum((1/length(bandnames))*abs(test_labels-truth))/(length(bandnames)*test)*100;


%% (test 2) Same Genre Bands
% Can we guess whether a clip is from one of the following different bands
% of the same genre:
% clipping - experimental hip-hop
% Kendrick Lamar - hip-hop
% Vince Staples - hip-hop
```

```
bandnames = {'clipping','Kendrick Lamar','Vince Staples'};
train = 60; % samples to train
test = 10; % samples to confirm
sl = 5; % sample length in seconds
[Xtrain, ltrain, Xtest, ltest] = generatesamples(D,bandnames,train,test,sl,mp3Fs);

Xtrain = fft(Xtrain, [], 1);
Xtest = fft(Xtest, [], 1);

X = [Xtrain Xtest];

%% SVD
[u, s, v] = svd(X - mean(X(:)), 'econ');

%% Classification
fig3 = figure(3);
plot(diag(s))
title('Singular Value Spectrum for Test 2')
ylabel('Singular Value')
xlabel('Mode')

% Y = u*X'; this creates a matrix that is big and useless
% We will form the first PCs columns of this matrix
% C(i,j) = A(i,:)*B(:,j)

PCs = [1 2 3 5]; % PC 3 for feature discrimination
Xp = X';
%Y(:, PCs) = u(:,:)*Xp(:,PCs);

xtrain = v(1:size(Xtrain,2), PCs)'; %v(PCs, 1:size(Xtrain,2));
xtest = v(size(Xtrain,2)+1:end, PCs)'; %v(PCs, size(Xtrain,2)+1:end);

fig4 = figure(4);
sgtitle('Samples as Points in Real and Imaginary Principal Component Space')
subplot(2,1,1)
scatter3(real(xtrain(1,ltrain==1)), real(xtrain(2,ltrain==1)), real(xtrain(3,ltrain==1)), 10, [1,0,0])
hold on
scatter3(real(xtrain(1,ltrain==2)), real(xtrain(2,ltrain==2)), real(xtrain(3,ltrain==2)), 10, [0,1,0])
scatter3(real(xtrain(1,ltrain==3)), real(xtrain(2,ltrain==3)), real(xtrain(3,ltrain==3)), 10, [0,0,1])
title('Real')
xlabel(sprintf('PC %d', PCs(1)))
ylabel(sprintf('PC %d', PCs(2)))
zlabel(sprintf('PC %d', PCs(3)))
legend(bandnames)
subplot(2,1,2)
scatter3(imag(xtrain(1,ltrain==1)), imag(xtrain(2,ltrain==1)), imag(xtrain(3,ltrain==1)), 10, [1,0,0])
hold on
scatter3(imag(xtrain(1,ltrain==2)), imag(xtrain(2,ltrain==2)), imag(xtrain(3,ltrain==2)), 10, [0,1,0])
scatter3(imag(xtrain(1,ltrain==3)), imag(xtrain(2,ltrain==3)), imag(xtrain(3,ltrain==3)), 10, [0,0,1])
title('Imaginary')
xlabel(sprintf('PC %d', PCs(1)))
ylabel(sprintf('PC %d', PCs(2)))
zlabel(sprintf('PC %d', PCs(3)))
legend(bandnames)
```

```
lclass = classify(real(xtest'),real(xtrain'),ltrain);
qclass = classify(real(xtest'),real(xtrain'),ltrain,'quadratic');
truth = ltest;
t2lE = 100-sum((1/length(bandnames))*abs(lclass-truth))/(length(bandnames)*test)*100;
t2qE = 100-sum((1/length(bandnames))*abs(qclass-truth))/(length(bandnames)*test)*100;

Model = fitcnb(real(xtrain'),ltrain);
test_labels = predict(Model,real(xtest'));
t2nbE = 100-sum((1/length(bandnames))*abs(test_labels-truth))/(length(bandnames)*test)*100;


%% (test 3) Genre Classification
% Can we categorize music as one of the following genres:
% hip-hop
% alt-rock
% classical

bandnames = {'hip-hop','alt-rock','classical'};
train = 60; % samples to train
test = 10; % samples to confirm
sl = 5; % sample length in seconds
[Xtrain, ltrain, Xtest, ltest] = generatesamples(D,bandnames,train,test,sl,mp3Fs);

Xtrain = fft(Xtrain, [], 1);
Xtest = fft(Xtest, [], 1);

X = [Xtrain Xtest];

%% SVD
[u, s, v] = svd(X - mean(X(:)), 'econ');

%% Classification
fig5 = figure(5);
plot(diag(s))
title('Singular Value Spectrum for Test 3')
ylabel('Singular Value')
xlabel('Mode')

% Y = u*X'; this creates a matrix that is big and useless
% We will form the first PCs columns of this matrix
% C(i,j) = A(i,:)*B(:,j)

PCs = [1 2 3 5]; % PC 3 for feature discrimination
Xp = X';
%Y(:, PCs) = u(:,:)*Xp(:,PCs);

xtrain = v(1:size(Xtrain,2), PCs)'; %v(PCs, 1:size(Xtrain,2));
xtest = v(size(Xtrain,2)+1:end, PCs)'; %v(PCs, size(Xtrain,2)+1:end);

fig6 = figure(6);
sgtitle('Samples as Points in Real and Imaginary Principal Component Space')
subplot(2,1,1)
scatter3(real(xtrain(1,ltrain==1)), real(xtrain(2,ltrain==1)), real(xtrain(3,ltrain==1)), 10, [1,0,0])
```

```
hold on
scatter3(real(xtrain(1,ltrain==2)), real(xtrain(2,ltrain==2)), real(xtrain(3,ltrain==2)), 10, [0,1,0])
scatter3(real(xtrain(1,ltrain==3)), real(xtrain(2,ltrain==3)), real(xtrain(3,ltrain==3)), 10, [0,0,1])
title('Real')
xlabel(sprintf('PC %d', PCs(1)))
ylabel(sprintf('PC %d', PCs(2)))
zlabel(sprintf('PC %d', PCs(3)))
legend(bandnames)
subplot(2,1,2)
scatter3(imag(xtrain(1,ltrain==1)), imag(xtrain(2,ltrain==1)), imag(xtrain(3,ltrain==1)), 10, [1,0,0])
hold on
scatter3(imag(xtrain(1,ltrain==2)), imag(xtrain(2,ltrain==2)), imag(xtrain(3,ltrain==2)), 10, [0,1,0])
scatter3(imag(xtrain(1,ltrain==3)), imag(xtrain(2,ltrain==3)), imag(xtrain(3,ltrain==3)), 10, [0,0,1])
title('Imaginary')
xlabel(sprintf('PC %d', PCs(1)))
ylabel(sprintf('PC %d', PCs(2)))
zlabel(sprintf('PC %d', PCs(3)))
legend(bandnames)

lclass = classify(real(xtest'),real(xtrain'),ltrain);
qclass = classify(real(xtest'),real(xtrain'),ltrain,'quadratic');
truth = ltest;
t3lE = 100-sum((1/length(bandnames))*abs(lclass-truth))/(length(bandnames)*test)*100;
t3qE = 100-sum((1/length(bandnames))*abs(qclass-truth))/(length(bandnames)*test)*100;

Model = fitcnb(real(xtrain'),ltrain);
test_labels = predict(Model,real(xtest'));
t3nbE = 100-sum((1/length(bandnames))*abs(test_labels-truth))/(length(bandnames)*test)*100;


%% Results
fmt = 'test %d \n LDM: %.2f \n QDM: %.2f \n NB: %.2f';
result1 = sprintf(fmt, 1, t1lE, t1qE, t1nbE);
result2 = sprintf(fmt, 2, t2lE, t2qE, t2nbE);
result3 = sprintf(fmt, 3, t3lE, t3qE, t3nbE);

disp(result1)
disp(result2)
disp(result3)

%% Saving

saveas(fig1,'svtest1.png')
saveas(fig2,'PCmusic1.png')
saveas(fig3,'svtest2.png')
saveas(fig4,'PCmusic2.png')
saveas(fig5,'svtest3.png')
saveas(fig6,'PCmusic3.png')
```

hw4musictest1acc.m

```
% Test Model Classification LD
clear; close all; clc;

mp3Fs = 44100; % Sample Rate, Hz
```

```
mdir = 'music';
D = dir(mdir);
D = D(3:end);

trials = zeros([100 1]);

for i = 1:1

bandnames = {'Beethoven','clipping','grandson'};
train = 60; % samples to train
test = 10; % samples to confirm
sl = 5; % sample length in seconds

[Xtrain, ltrain, Xtest, ltest] = generatesamples(D,bandnames,train,test,sl,mp3Fs);

Xtrain = fft(Xtrain, [], 1);
Xtest = fft(Xtest, [], 1);

X = [Xtrain Xtest];

%% SVD
[u, s, v] = svd(X - mean(X(:)), 'econ');

PCs =[1 3 6]; % PC 3 for feature discrimination
%Xp = X';
%Y(:, PCs) = u(:,:)*Xp(:,PCs);

xtrain = v(1:size(Xtrain,2), PCs)'; %v(PCs, 1:size(Xtrain,2));
xtest = v(size(Xtrain,2)+1:end, PCs)'; %v(PCs, size(Xtrain,2)+1:end);

Model = fitcnb(real(xtrain'),ltrain);
test_labels = predict(Model,real(xtest'));
t1nbE = 100-sum((1/length(bandnames))*abs(test_labels-truth))/(length(bandnames)*test)*100;

trials(i) = t1bnE;

end

bar(trials)
yline(mean(trials),'--r', {sprintf('average = %.2f', mean(trials))})
```