Faces and Genres Identification

Han Song
University of Washington
Department of Applied Mathematics
Seattle, WA
GitHub repository for article:

Abstract: In this paper, we are going to apply SVD method on 2 different kinds of art forms which are images and songs. For images, we reconstructed some image in lower modes than original images; result yielded image can be reconstructed with 10% number of total modes in. For songs, we intend to classify a random 5 sec music audio and see whether the algorithm can sort it to the right band or genre. KNN tends to predict less accurate than SVM but we can achieve around 60% accuracy for all tests.

GitHub Repo: Id.git

I. Introduction and Overview:

Our brain is amazing, and it continues to evolve beyond what we thought to be the limit. When you look at a person, you register their dominant features such as eyes, hairline, lips, glasses… long term memory does not commit to memorize how exactly people look like, but you can differentiate between two individuals based on unique characteristic. Same thing can be said about songs we heard. Talented society has invented multiple genres. After we heard the rhythm once or twice, our brain could response with "What a good country song!" when we heard Keith Urban. In this paper, we are going to use MATLAB to mimic how our brains remember people's faces at a school gathering event and to classify genres of songs we heard at the event using the powerful Singular Values Decomposition (SVD), K nearest neighbor (KNN), support vector machine (SVM) methods. In part 1, when we analyze images, SVD is used to determine the least amount of modes to reconstruct images without losing the characteristic of these faces. In part 2, SVD is used to statistically correlate all songs to determine characteristic of every sets of songs and to predict to which genre a random song belongs.

II. Theoretical Background:

We are going to approach with a theorem which states that every matrix **A** has a singular value decomposition and it takes form $\boldsymbol{A} = \boldsymbol{U\Sigma V}^*$. Each of these matrix components has specific meaning to matrix $\boldsymbol{A}$ (m x n matrix)**.** **U** matrix is a unitary matrix consists of n orthonormalized eigenvectors of AA^T which will rotate matrix **A** so that its axis the principle coordinates,$\boldsymbol{\Sigma}$ consists of non-negative square roots of the eigenvalues of $\boldsymbol{A^T A}$ in a descending order which will "modify" the rotated axes to scale with the principle coordinates and**,** and lastly $\boldsymbol{V}$ matrix is a unitary matrix consists of n orthonormalized eigenvectors of $\boldsymbol{A^T A}$ which will rotate the data point from the rotated principle axis.

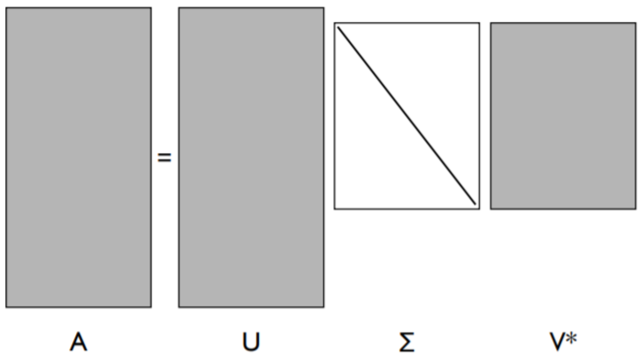


Figure 1: Graphical description of SVD (Credit: Professor Kutz)

III. Algorithm Implementation and Development:

1. Images

First and foremost, we must obtain the right data type which MATLAB can understand and analyze. Because all pictures are in black and white and aligned, I went through each folder and reshaped each image into a column and place it into a big matrix **all_img**. I then took the SVD of matrix **A** to obtain $\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}^*$and plot the singular value spectrum of this data set. I then tried to reconstruct images by using multiple modes to verify my understanding about SVD. The process is repeated for original image data set.

2. Songs

For preprocessing process, I created a function to form a song matrix and various information that I need in the later process. I randomly grabbed 5 songs to test and the rest of the songs are my training set. Because I don't have large library of music, my code is scalar to any size library where songs are organized into 1 folder for each artist or genre. I chose to work with spectrogram data because every genre has different voice frequency. I use V matrices to classify either band or genre in KNN and SVM method.

IV. Computational Results and Analysis:

1. Images:

a. Cropped faces

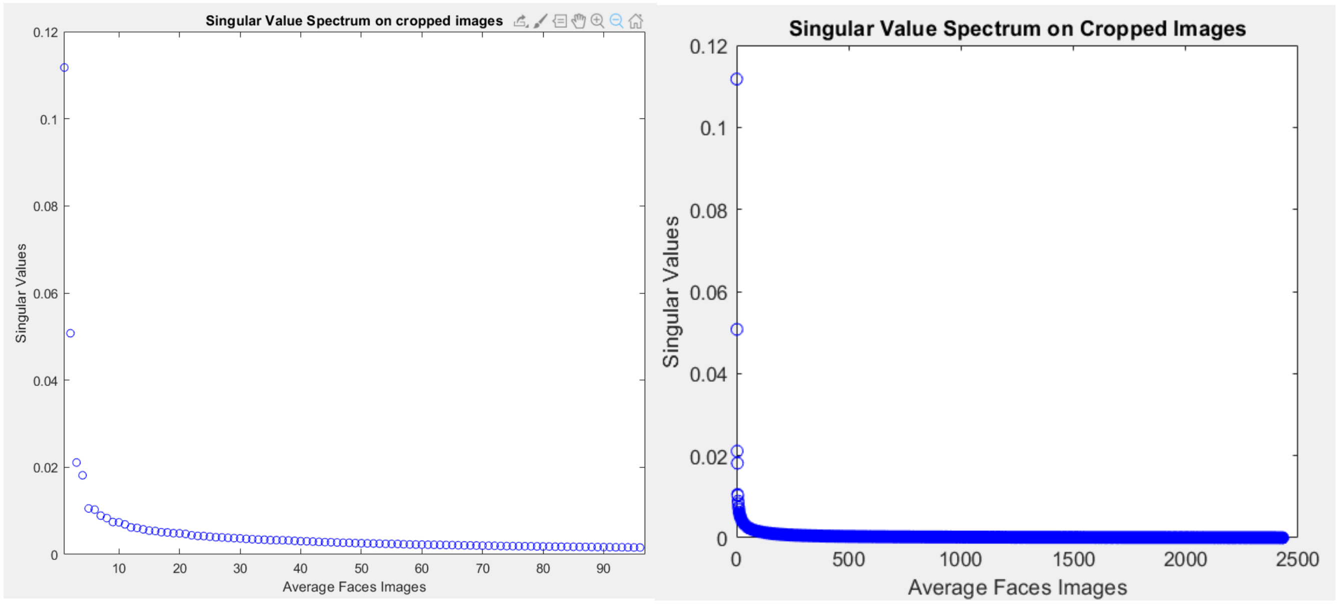


Figure 2: Modes of Cropped Images domimant modes (left) and all modes(right)

The very first figure I extracted is the Singular Value Spectrum. In figure 2, we can see majority of modes are 0 and when we zoom in like figure on the left, we can clearly see that the mode converges to zero after 90 modes. Figure 3 below is a result of reconstructing a random image. We can clearly see that rank 4 picture looks nothing like the original image. In other words, every picture in this data set starts outs with eyes, nose,and lip. Rank 10 gave us more nose definition and eyebrows; more and more characters were added as I reconstructed the image in a higher rank of U. At rank 150, it looks more like our original image and we can stop there if our purpose is to have a smaller data size image.

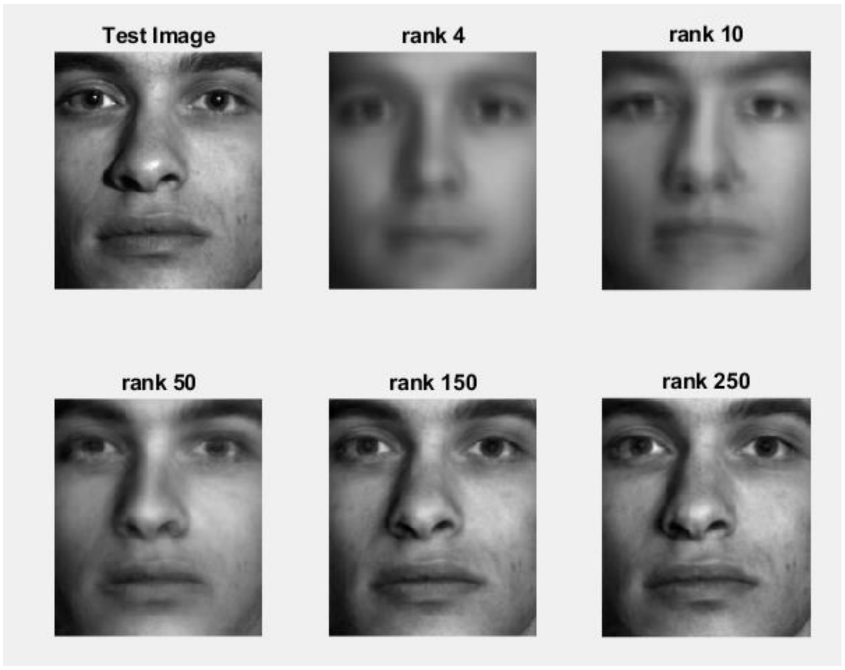


Figure 3: random image reconstructed in multiple rank

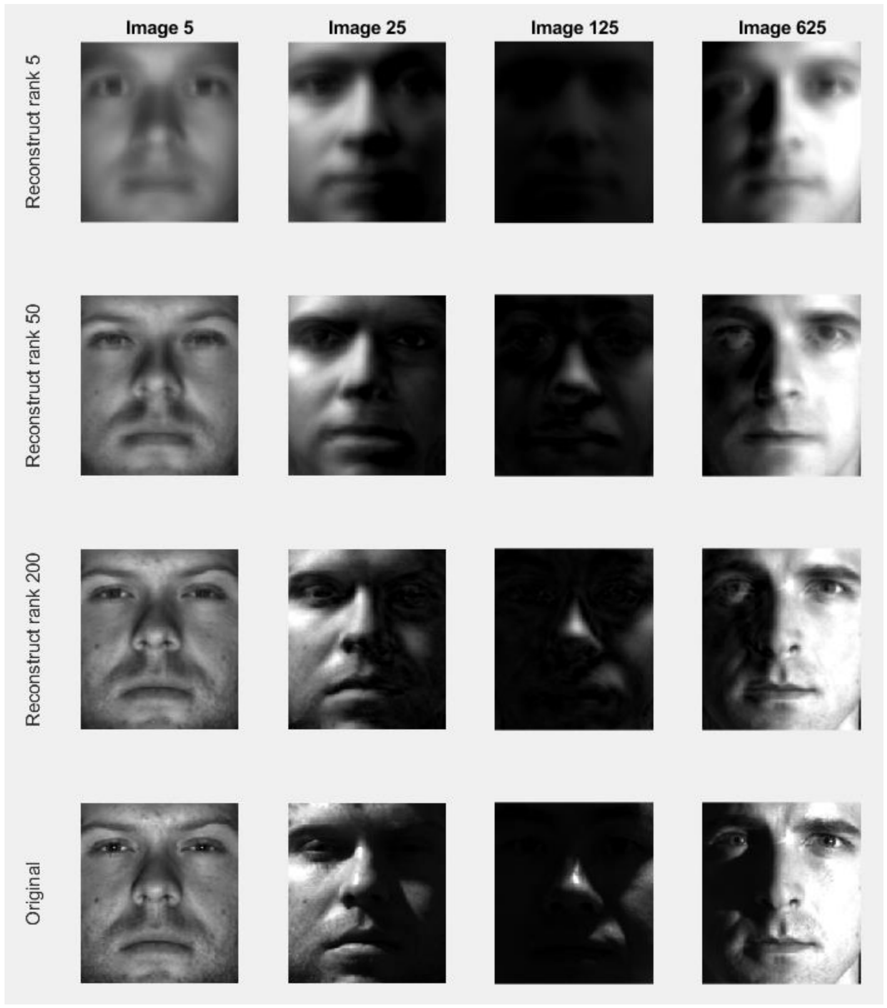
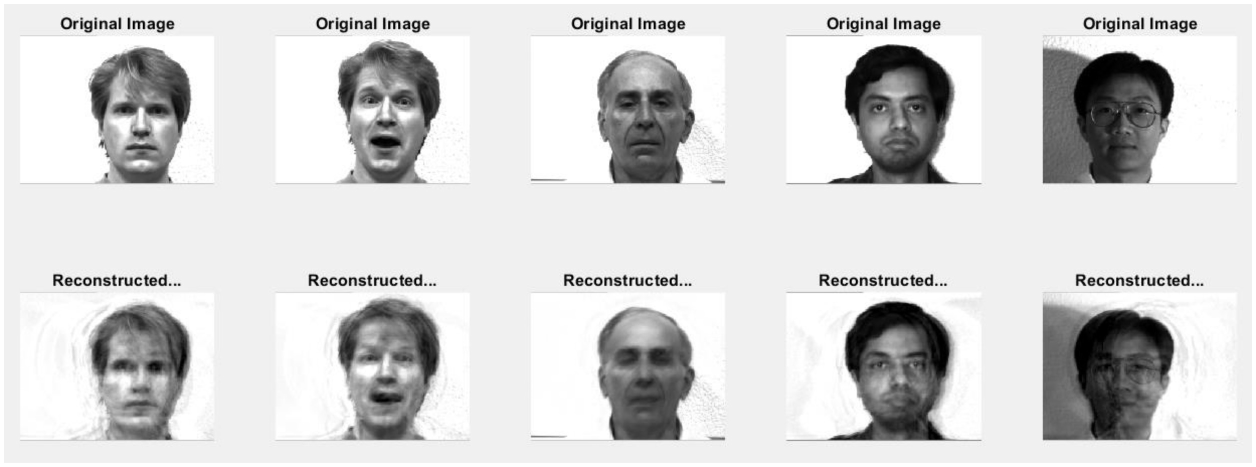


Figure 4: multiple images with multiple ranks reconstruction

By reconstructing multiple images with multiple ranks, it demonstrated that different images can go through different ranks to reserve majority of the energy.

b. Uncropped image:

The result of uncropped images is like cropped image data set, except for the U matrix will now contain information of position of the person within the frame.



2. Songs:

   Accuracy of 3 tests is

Test	KNN	SVM
Test 1	0.7	0.8
Test 2	0.4	0.4
Test 3	0.4	0.6

   Test 1 has higher accuracy of all because the data in this set is very different from one another while the other two tests have a mix of similar bands or genres.

V. Summary:

This paper demonstrates that SVD can reduce significant size of data and still represent data with correct unique characteristic. KNN and SVM are somewhat efficient to classify data that are highly distinguishable such as classical music vs pop music.

Appendix A: MATLAB functions

KNN=fitcknn(X,Y) is an alternative syntax that accepts X as an

N-by-P matrix of predictors with one row per observation and one column

per predictor. Y is the response and is an array of N class labels.

MODEL=fitcsvm(X,Y) is an alternative syntax that accepts X as an

N-by-P matrix of predictors with one row per observation and one column

per predictor. Y is the response and is an array of N class labels.

[U,S,V] = svd(X,'econ') also produces the "economy size"

decomposition. If X is m-by-n with m >= n, then it is

equivalent to svd(X,0). For m < n, only the first m columns

of V are computed and S is m-by-m.

Appendix B: MATLAB Codes

```
function [training_set, info_matrix, song_list, artist]=
getTrainingSet(folder_name)
    trainingSet_dir = dir(folder_name);
    trainingSet_dir = trainingSet_dir(3:end);
    sbf = length(trainingSet_dir); % sbf = size_big_folder
    info_matrix = strings(sbf, 4);
    num_song = 0;
    for i = 1: sbf
        sub_folder = dir([folder_name '/'
trainingSet_dir(i).name]);
        num_song = num_song + length(sub_folder(3:end));
        info_matrix(i,1)= trainingSet_dir(i).name;
        info_matrix(i,2) =
num2str(length(sub_folder(3:end)));
        info_matrix(i,4) = num_song;
    end

    sample = 240000; %sampling each song at specific time
5s
    info_matrix(:,3) = sample;
    training_set = zeros(sample, num_song);
    song_list = strings(num_song,1);
    artist = song_list;
    index = 0;
    for i = 1:sbf
        sub_folder = dir([folder_name '/'
trainingSet_dir(i).name]);
        for j = 1: length(sub_folder(3:end))
            index = index + 1;
            [x,~] = audioread([folder_name '/'
trainingSet_dir(i).name '/' sub_folder(j+2).name]);
            new_song = zeros(sample,1);
            for k = 1:sample
                new_song(k,1) = x(2*k-1,1);
            end
            training_set(:,index) = new_song;
            song_list(index,1) = sub_folder(j+2).name;
            artist(index,1) = trainingSet_dir(i).name;
        end
    end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%% AMATH582 HW4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: Han Song
% Class: AMATH 582
% Due Date: 3/6/2020
clear all;clc; close all
%% Part 1: Yale Faces B

% load files to establish number of iterations. Please
extract cropped
% images folder into working folder with name =
"CroppedYale" and uncropped
% images folder into working folder with name = "yalefaces"
for code to
% work smoothly
Cropped_folder = dir('CroppedYale');
Cropped_folder = Cropped_folder(3:end);
Uncropped_folder = dir('yalefaces');
Uncropped_folder = Uncropped_folder(3:end);
%
% Consolidate all images into 1 matrix, each column is 1
image
index = 1; %dummy index
image_size = 192*168;
all_img = zeros(image_size,2432); %2432 is total number of
images in all folders
ave_face = zeros(image_size,length(Cropped_folder(3:end)));
for i=1:length(Cropped_folder)
    subfolder = dir(['CroppedYale/'
Cropped_folder(i).name]);
    for j=3:length(subfolder)
        data = imread(['CroppedYale/'
Cropped_folder(i).name '/' subfolder(j).name]);
        data = reshape(data,image_size,1);
        all_img(:,index) = data;
        index = index+1;
    end
    ave_face(:,i) = sum(all_img,2)/length(all_img);
end

% SVD and plot spectrum
all_img = double(all_img);
[U,S,V] = svd(all_img,'econ');
figure(1)
plot(diag(S)/sum(diag(S)),'bo')
```

```
ylabel('Singular Values')
xlabel('Average Faces Images')
title('Singular Value Spectrum on Cropped Images')
%
% Testing 1 random image and multiple images
% 1 image
random_img = imread('CroppedYale/yaleB18/yaleB18_P00A-
020E+10.pgm');
figure(3)
subplot(2,3,1)
imshow(random_img)
title('Test Image')
random_img = double(reshape(random_img,192*168,1));
rank = [4,10,50,150,250];
for i = 1:length(rank)
    U_rec = U(:,1:rank(i));
    recon = U_rec*U_rec'*random_img;
    recon = reshape(recon,192,168);
    subplot(2,3,i+1)
    imshow(uint8(recon))
    title(['rank ',num2str(rank(i))])
end
%
% multiple images
img_no = [5,25,125,625]; % random images order
rank = [5, 50, 200];
figure(2)
for i = 1: length(img_no)
    for j = 1:length(rank)
        reconstruct = U*S(:,1:rank(j))*V(:,1:rank(j))';
        subplot(length(rank)+1,length(img_no),(j-
1)*length(img_no)+i)

imshow(uint8(reshape(reconstruct(:,img_no(i)),192,168)));
        if j == 1
            title(['Image ' num2str(img_no(i))])
        end
        if i == 1
            ylabel(['Reconstruct rank ' num2str(rank(j))])
        end
    end
    subplot(length(rank)+
1,length(img_no),j*length(img_no)+ i)
    imshow(uint8(reshape(all_img(:,img_no(i)),192,168)));
    if i == 1
```

```
        ylabel('Original')
    end
end

%
% Original
image_size_uncropped = 243*320;
original = zeros(image_size_uncropped,165);
ave_face_original = original;
for i=1:length(Uncropped_folder)
    data = imread(['yalefaces/' Uncropped_folder(i).name]);
    data = reshape(data,image_size_uncropped,1);
    original(:,i) = data;
    ave_face_original(:,i) =
sum(original,2)/length(original);
end

% SVD and plot
original = double(original);
[U_O,S_O,V_O] = svd(original,'econ');
figure(4)
plot(diag(S_O)/sum(diag(S_O)),'o')
ylabel('Singular Values')
xlabel('Images')

% Test multiple image with multiple rank
reconstruct = U_O*S_O(:,1:50)*V_O(:,1:50)';
img_no = [5,10,50,100,150];
for i = 1:length(img_no)
    subplot(2,length(img_no),i)
    imshow(uint8(reshape(original(:,img_no(i)),243,320)));
    title('Original Image')
    subplot(2,length(img_no),length(img_no)+i)

imshow(uint8(reshape(reconstruct(:,img_no(i)),243,320)));
    title('Reconstructed...')
end

% similar to the cropped images, size image exceeds matlab
limit and cant
% test.

% rand_img = imread('yalefaces/subject08.wink');
% figure(6)
% subplot(2,3,1)
% imshow(rand_img)
% title('Test Image')
% rand_img = double(reshape(rand_img,243*320,1));
% rank = [2];
% for i = 1:length(rank)
%     U_rec = U_O(:,1:rank(i));
%     recon = reshape(U_rec*U_rec'*rand_img,243,320);
%     subplot(2,3,i+1)
%     imshow(uint8(recon))
%     title(['rank ',num2str(rank(i))])
% end

%% Part 2: Music Classification

% getting data...
[Songs , info_str, song_list] = getTrainingSet('Test1');
info_int = str2double(info_str(:,2:end));
sample_size = info_int(1,2);
num_song = info_int(end,end);
% set label for each artist n = 1,2,3,...
song_matrix = ones(1,num_song);
for i = 1:length(info_int(:,1))
    if i ~= 1
        song_matrix(:,info_int(i-1,3)+1:info_int(i,3)) = i;
    end
end

% Separate training and testing set
num_test = 5; % test 5 songs to get avg accuracy for KNN
and SVM
random_index = randi([1 num_song],1,num_test);
test_song = Songs(:,random_index);
actual_label = song_matrix(:,random_index);
Songs(:,random_index) = [];
song_matrix(:,random_index) = [];
trainingSet = Songs;
target = song_matrix;

% Obtain spectrogram of training and testing set
spec_train = zeros(sample_size,num_song - num_test);
for i = 1:length(trainingSet(1,:))
    ft = fft(trainingSet(:,i));
    spec = abs(fftshift(ft));
    spec_train(:,i) = spec(:,1);
end
```

```
spec_test = zeros(sample_size, 1);
for i = 1:num_test
    ft = fft(test_song(:,i));
    spec = abs(fftshift(ft));
    spec_test(:,i) = spec(:,1);
end

[a,b]=size(spec_train); % compute data size
ab=mean(spec_train,2); % compute mean for each row
spec_train=spec_train-repmat(ab,1,b); % subtract mean

[c,d]=size(spec_test); % compute data size
cd=mean(spec_test,2); % compute mean for each row
spec_test=spec_test-repmat(cd,1,d); % subtract mean

% SVD
[~,S,V] = svd(spec_train','econ');
figure(2)
plot(diag(S)/sum(diag(S)),'bo')
xlabel('Songs')
ylabel('Singular values')
title('Singular Value Spectrum for Test 1')

%
% KNN
knn.mod = fitcknn(V',target,'NumNeighbors',5);
label = predict(knn.mod,test_song');
accuracy = 0;
for i = 1:length(label)
    if label(i) == actual_label(i)
        accuracy = accuracy + 1;
    end
end
accuracy = accuracy/num_test;
fprintf('Accuracy for test 1 using KNN is %.1f \n',
accuracy);

%
% SVM
svm.mod = fitcecoc(V',target);
label_svm = predict(svm.mod,test_song');
accuracy = 0;
for i = 1:length(label_svm)
    if label_svm(i) == actual_label(i)
        accuracy = accuracy + 1;
    end
end
accuracy = accuracy/num_test;
fprintf('Accuracy for test 1 using SVM is %.1f \n',
accuracy);
%
test1 = strings(length(random_index),5);
for i = 1:length(random_index)
    test1(i,1) = random_index(i);
    test1(i,2) = song_list(random_index(i));
    test1(i,3) = label(i);
    test1(i,4) = label_svm(i);
    test1(i,5) = actual_label(i);
end
fprintf('song   title               KNN       SVM    Actual
\n');
test1
```