AMATH 582: Homework 4

Hyun Ah Lim
March 5, 2020

Abstract

Two different types of data sets (images and audio) were analyzed using Singular Value Decomposition (SVD) for Principal Component Analysis (PCA). The first group, images of various individuals faces ('Yale Faces'), contained two sets of images. The first were 'cleaned' images where faces were cropped and aligned; the second set had no such cleaning performed. As expected, the 'cleaned' images resulted in the dominant modes containing variation of the facial features, rather things like head alignment, hairstyle, etc. Image reconstruction required a much lower rank approximation to construct an image that sufficiently identified the subject's face in the first set. The second part of this analysis involved the same SVD method, using the frequency content of audio signals for Classification (supervised learning). Three classification methods were used, Linear Discrimination Analysis (LDA), Naïve Bayes and K Nearest Neighbor. All three methods provided positive results in terms of accuracy, with one performing significantly worse than the others. Audio data were collected online with varying quality. Increasing both the quality and quantity of data would have likely increased overall accuracy.

In both analyses, the output elements of SVD (U, Σ and V matrices) were used to determine the important feature spaces and allowed for efficient analysis of the variation within the datasets. Therefore, SVD was a reliable method for PCA that could be utilized for different objectives (efficient image reconstruction and music classification). But this exercise also made it clear that the quality of data collection has a significant impact on efficiency and accuracy.

1 Introduction and Overview

This analysis involves the application of Singular Value Decomposition (SVD) for image analysis/reconstruction and music classification. The datasets to be evaluated are outlined below.

- Yale Faces (two data sets)
  - ✓ Cropped and Aligned
  - ✓ Original (uncropped and not aligned)

The pixel content of these images will be transformed using SVD and the results used to generate eigenfaces representing the feature dimensions and their dominance. In addition, a single subject image from each dataset will be reconstructed and the lowest rank approximation needed to reconstruct a recognizable image will be determined.

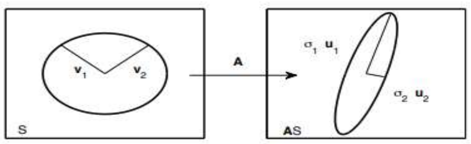
- Five Second Music Clips (three data sets comprised of 50 five second music clips for each Classification type)
  - ✓ Band Classification from different genres: Frank Sinatra, The Smiths and Public Enemy
  - ✓ Band Classification from the same genre: Oasis, Blur and The Verve
  - ✓ Genre Classification – British Pop/Rock, Classical and Classic Jazz

The frequency content of the music clips will be transformed using SVD and the results used to classify 'new' clips of music within each dataset. The three classification methods applied in this exercise will be Linear Discrimination Analysis, Naïve Bayes and K Nearest Neighbor.

2 Theoretical Background

This analysis employs Singular Value Decomposition (SVD) for Principal Component Analysis. This data transformation method reduces dimensionality, while capturing the maximum variance given that selected dimension/feature space. .

Singular Value Decomposition (for Principal Component Analysis) has been discussed in previous papers in this class series, but as a quick review, SVD decomposes any matrix into its basic actions, rotation and scaling. An illustration of what happens to vectors v1, and v2 are shown below in figure 1.



$$A\,V = U\,\Sigma$$

FIGURE 1. EXAMPLE WHERE VECTORS, V1 AND V2, ARE HIT WITH MATRIX A, AND THE RESULTING IMPACT THE EQUATION THAT REPRESENTS THIS IS SHOWN IN EQUATION FORM AS WELL AS MATRIX FORM (TEXTBOOK[1]).

You can then rewrite this to outline the SVD components of any matrix A: U, Σ and V.

$$A = U\,\Sigma\,V^T \qquad \text{(eq 1)}$$

If we think of these output matrices, when the matrix A is comprised of face images, then we can think of them as follows:

- U: The columns of this matrix are the orthonormal bases, ordered by the level of variation of the dataset in that basis. They are the eigenvectors of the covariance matrix of A (AA^T). They represent the feature dimensions that we can use to define matrix A, ordered by dominance (i.e. energy, variance, etc.). In this dataset they are referred to as eigenfaces.
- Σ: These are the singular values that tell us the level of dominance in each of the corresponding bases (eigenfaces).
- V: This matrix contains the weight of an image in each eigenface. Each row of V represents a single image (column) from matrix A and each element in that row tells us how to project that specific image onto that basis.

Based on the above elements a low rank image approximation of the image where the subject is recognizable can be generated by taking the dot product of U, Σ and V^T, using only the minimum modes (minimum number of columns of U and elements of Σ) and the specific row of V^T that corresponds to our image.

For music classification, SVD is performed on the Fourier Transformed frequency content of each audio clip (background also discussed in a previous paper in this series). The V output matrix from SVD contains the dominant discriminating elements between song clips and will be used as the data for classification purposes.

Three supervised machine learning classification methods are used, and their descriptions are below. Supervised learning indicates that the dataset includes labels and the model is trained using this known information. The number of Principal Components used (based on SVD) is also evaluated for the best prediction accuracy.

- *Linear Discrimination Analysis* works to find a suitable projection that maximizes the distance between the inter-class data while minimizing the intra-class data. In other words, construct a projection such that (ref [1]). This discriminate line for projection is determined using the training data.

$$\mathrm{w} = \mathrm{argmax}_{\mathrm{W}} \frac{w^T\, \mathrm{s}_\beta w}{w^T\, \mathrm{s}_\mathrm{w} w} \qquad \text{(eq 2)}$$

- *Naïve Bayes* method takes a statistical approach by using the probability of an event occurring in the prediction. If the prediction is binary, then this can be written as:

$$\frac{P(1|x)}{P(0|x)} = \frac{P(1|x) * P(1)}{P(0|x) * P(0)} \qquad \text{(eq 3)}$$

  If you know the ratio ($\frac{P(1|x)}{P(0|x)}$) then you can make a prediction (the training data is used to generate this ratio)

- *K Nearest Neighbor* predicts the classification by searching for the n number of nearest neighbors (in the training set and their classifications) to predict the test data point. 'Nearest' is in terms of distance to the nearest neighbor(s) within the features space (dimensions) selected.

All methods require a labelled training dataset and a corresponding unlabeled set to test the model against. The total dataset is randomly shuffled to produce the train vs. test subsets. This is done multiple times, reshuffling and resampling the training vs test data set for the essential practice of Cross Validation.

3 Algorithm Implementation and Development

The algorithm for all four scenarios were performed in a similar fashion and are outlined below:

3.1 Yale Faces

3.1.1 Import Images and Generate Matrix 'A_cropped' using a for loop, where each iteration imports a single file, reshapes it to a column vector and then appends it to the matrix A_cropped.

3.1.2 Generate mean face by subtracting from each element, its row's mean value.

3.1.3 Execute Singular Value Decomposition (SVD) to generate matrices, U, Σ and V.

3.1.4 Extract Singular Values from the diagonal matrix, Σ, to determine dominance of each mode, and plot.

3.1.5 Use a for loop to establish the minimum mode to reach cumulative energy increment totals of 10 (i.e. 10%, 20%, 30%, etc.). Set this up as a table for reference later.

3.1.6 Use imagesc() to generate mean face image along with the first through fourteenth eigenface images.

3.1.7 Select a single subject from A_cropped and use the mode numbers in the table generated in step 3.1.5 to reconstruct the image using 10% cumulative energy, 20% of cumulative energy, etc. The reconstruction is performed by taking the dot product of SVD output matrices ($U*\Sigma*V^T$), using only the modes specified for reconstruction in U and Σ and then for V^T include only the row that corresponds to the subject image. For example if in the original image matrix A_cropped, we want to recreate the image in column 13 (image 13) then we use row 13 the V^T matrix (sample code: face13 = u(:, 1:mode)*s(1:mode, :)* v(13, :)).

3.1.8 Generate these images and evaluate at which % of cumulative energy does the reconstructed image sufficiently identify the subject.

3.1.9 Repeat steps 3.1.1 – 3.1.9 for the 'Original Images' (uncropped and not aligned) and compare results.

3.2 Music Classification

3.2.1 Compile 50 five second music clips for each of the following artists:
Test1 (Band Classification, different genres): Frank Sinatra, The Smiths, Public Enemy

3.2.2 Compile matrix of song clips for each artist.
Use dir() function to generate a list of the filenames and locations for one of the artists. Use this list to execute a for loop, where each iteration imports a single file, reshapes it to a column vector, takes the discrete fast Fourier transform, shifts the values so zero is in the center, fftshift(), and then appends it to a single matrix. Repeat this step for each artist.

3.2.3 Concatenate all three matrices horizontally, using horzcat(), and normalize by subtracting from each element the mean value of its row.

3.2.4 Execute SVD to generate matrices, U, Σ and V.

3.2.5 Extract Singular Values from the diagonal matrix, Σ, to determine dominance of each mode, and plot.

3.2.6 Use a for loop to establish the minimum mode to reach cumulative energy increment totals of 10 (i.e. 10%, 20%, 30%, etc.). Set this up as a table for reference later.

3.2.7 Use for loop to perform the following Classification algorithm for 1:n modes that comprise 40, 50, 60, 70 and 80% of the cumulative energy as determined from step 3.2.6

3.2.7.1 Use inner for loop to execute 100 iterations (for cross validation) for Linear Discrimination Classification. This involves randomly selecting 40 song clips from each artist to train, and the balance of 10 songs to test, using the MATLAB classify() function.

3.2.7.2 Use inner for loop to execute 100 iterations (for cross validation) for Linear Discrimination Classification. This involves randomly selecting 40 song clips from each artist to train, and the balance of 10 songs to test, using the MATLAB classify() function.

3.2.7.3 Use inner for loop to execute 100 iterations (for cross validation) for Linear Discrimination Classification. This involves randomly selecting 40 song clips from each artist to train, and the balance of 10 songs to test, using the MATLAB classify() function. Each artist is provided a class value of 1, 2 or 3. Prediction of 30 (10 per artist) song clips are then compared to the actual class. In this case, the V matrix from the SVD output is used as the input representing the song.

3.2.7.4 Plot resulting accuracy for all the selected cumulative energy % and plot for comparison.

3.2.7.5 Repeat steps 3.2.7.1-3.2.7.4 using classification methods Naïve Bayes and K Nearest Neighbor.

3.2.8 Repeat steps 3.2.1 – 3.2.7 for the two additional test cases as follows:

3.2.8.1 Band Classification (Same Genre): Oasis, Blur, The Verve

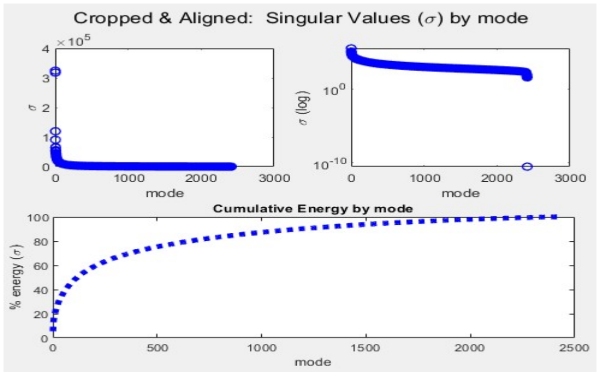
3.2.8.2 Genre Classification (Various Artists): British Pop/Rock, Classical, Classic Jazz

4 Computation Results

4.1 Part 1: Yale Faces

4.1.1 Singular Values: The Σ output element of SVD

The resulting Singular values from the SVD analysis are displayed below for both the cropped/aligned and uncropped/not aligned datasets(figure1). Note that the charts at the bottom show that the variation for the uncropped/not aligned images shows that the variation (energy) is more distributed across all the later modes. This leads us to expect that the cropped images will require a lower rank approximation to reconstruction the original subject images.



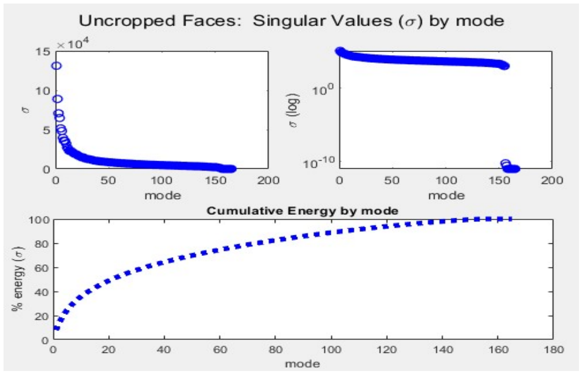
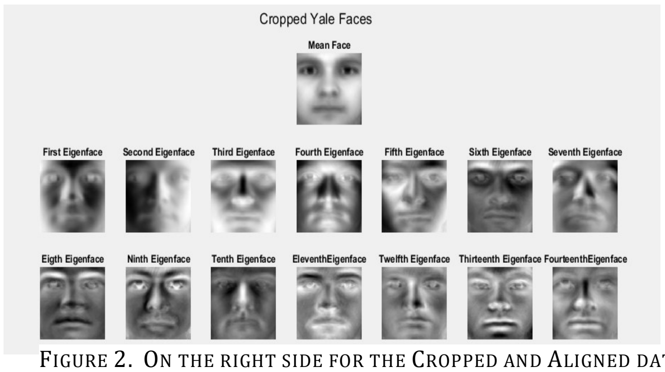


FIGURE 1. ON THE LEFT WE HAVE THE SINGULAR VALUES FROM THE Σ MATRIX FROM SVD (CUMULATIVE SIGMA ENERGY IS PLOTTED IN THE SECOND ROW). AND THEN ON THE RIGHT, WE HAVE THE SAME FOR THE UNCROPPED DATASET.

4.1.2 Eigenface Visualizations: The U element of SVD

In figure 2, the mean face (the pixel average of all the images) is shown on the top row for both the cropped and uncropped images. It is apparent that in the cropped dataset, the eigenfaces primarily variations in facial features, where the uncropped, shows the variation in head placement, angle and hair and much less specificity to facial features.



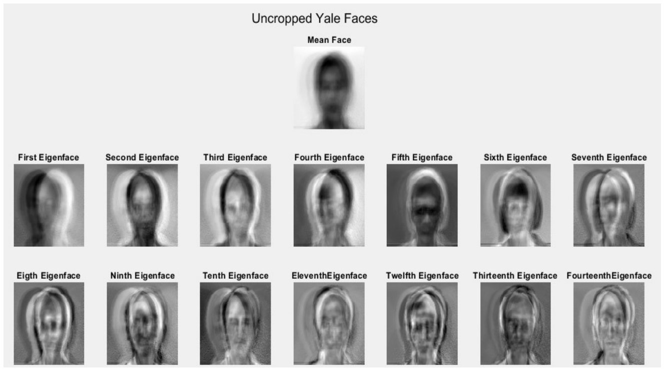
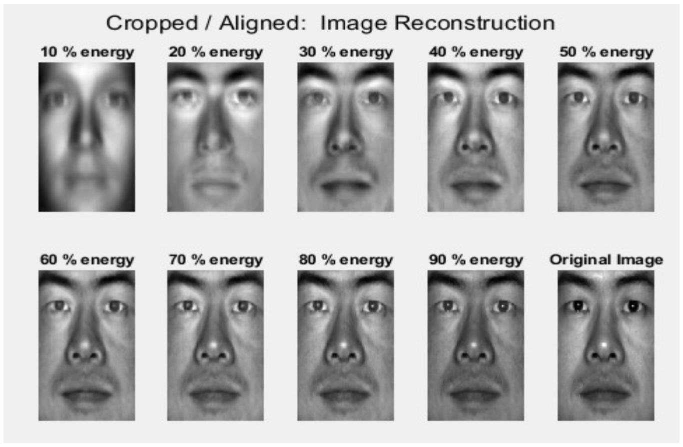


FIGURE 2. ON THE RIGHT SIDE FOR THE CROPPED AND ALIGNED DATASET, WE HAVE THE MEAN FACE (COMPOSED OF THE AVERAGE OF ALL IMAGES) AND THE FIRST 14 COMPONENT EIGENFACES. AT THE RIGHT SIDE, WE HAVE THE SAME FOR THE UNCROPPED AND UNALIGNED DATASET.

4.1.3 Image Reconstruction: Utilizing the V element of SVD.

In figure 3, we can see that the cropped and aligned images require a lower rank approximation (less mode data) to reconstruct the subject, in contrast to the unaligned and uncropped dataset. This is expected given the singular values generated by each SVD analysis. In the figure below you can see that the photo from the cropped data set only requires ~ 30-40% energy for the face to be well reconstructed; for the uncropped/unaligned subject, ~ 70-80% is required.



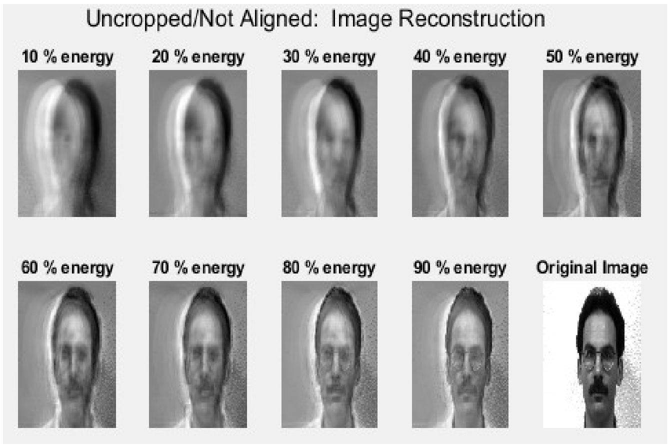


FIGURE 3. IMAGE RECONSTRUCTION. ON THE LEFT, THE SUBJECT IS FROM THE CROPPED AND ALIGNED DATASET; AND ON THE RIGHT, THE UNCROPPED AND NOT ALIGNED.

4.2 Part 2: Music Classification

4.2.1 Singular Values: The Σ output element of SVD

Singular values are plotted below in figure 4, resulting from the SVD analyses for the Band Classification (Frank Sinatra, The Smiths, Public Enemy), Same Genre Classification (British Pop/Rock: Oasis, Blur, The Verve) and the Genre Classification (British Pop/Rock, Classical, Classic Jazz).

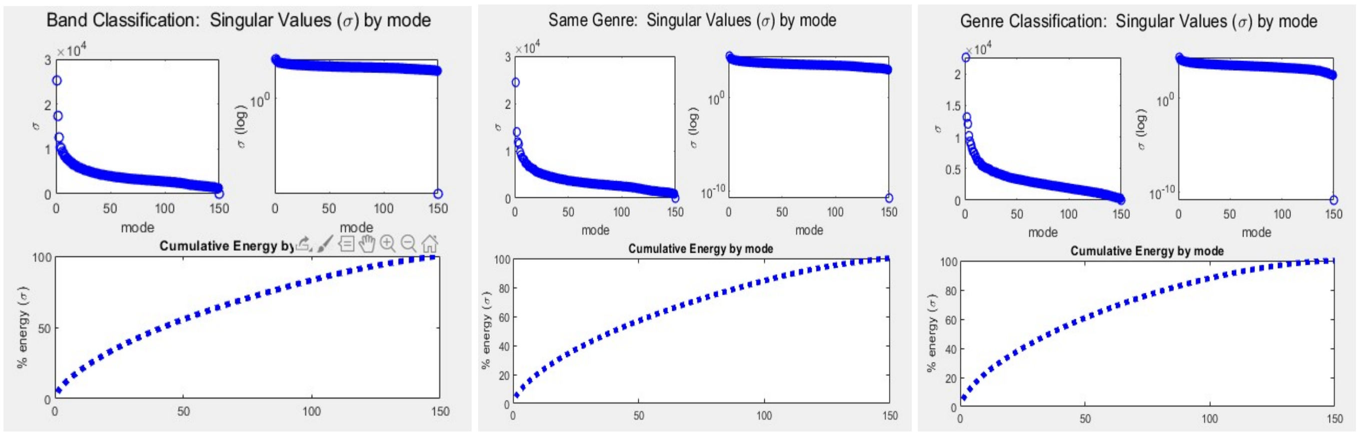


FIGURE 4: THE SIGMA VALUES ARE PLOTTED IN THE TOP ROW OF EACH CASE SECTION. THE SECOND ROW IN EACH SECTION DISPLAYS THE CUMULATIVE ENERGY BY MODE.

4.2.2 Classification Methods: Linear Discrimination, Naïve Bayes and K Nearest Neighbor

Samples of the results for one iteration of each of the classification methods are shown below in figure 5 (as well as for each test case). Figure 6 summarizes these results (average accuracy) for each of the classification methods given 100 iterations through rank approximations correlating to 40, 50, 60 , 70 and 80 % of the cumulative energy (variation).

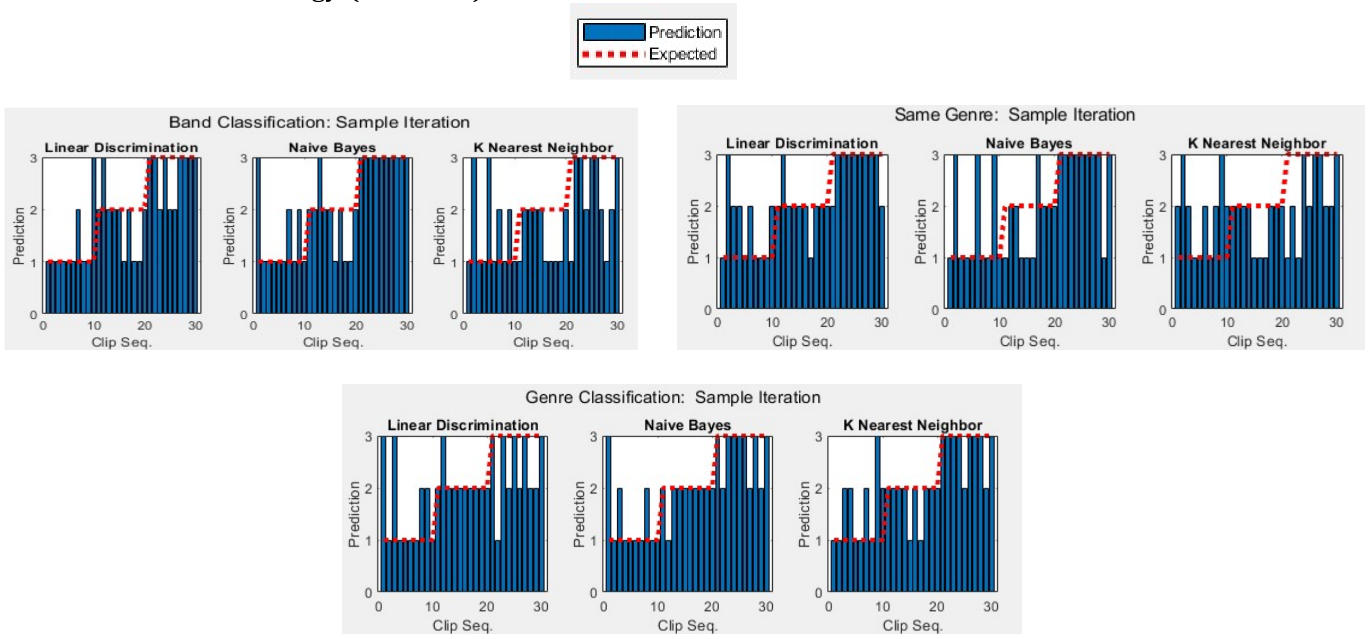


FIGURE 6: SAMPLE RESULTS FROM EACH OF THE TEST CASES AND CLASSIFICATION METHOD COMBINATIONS IS SHOWN BELOW. SONGS 1-10 BELONG TO THE CLASSIFICATION CODE '1', SONGS 11-20, BELONG TO CODE '2' AND SONGS 21-30, TO CODE '3'. THE RED LINE DISPLAYS THE CORRECT RESULTS, WHILE THE BLUE BARS DISPLAY THE PREDICTED VALUE FROM MODEL.

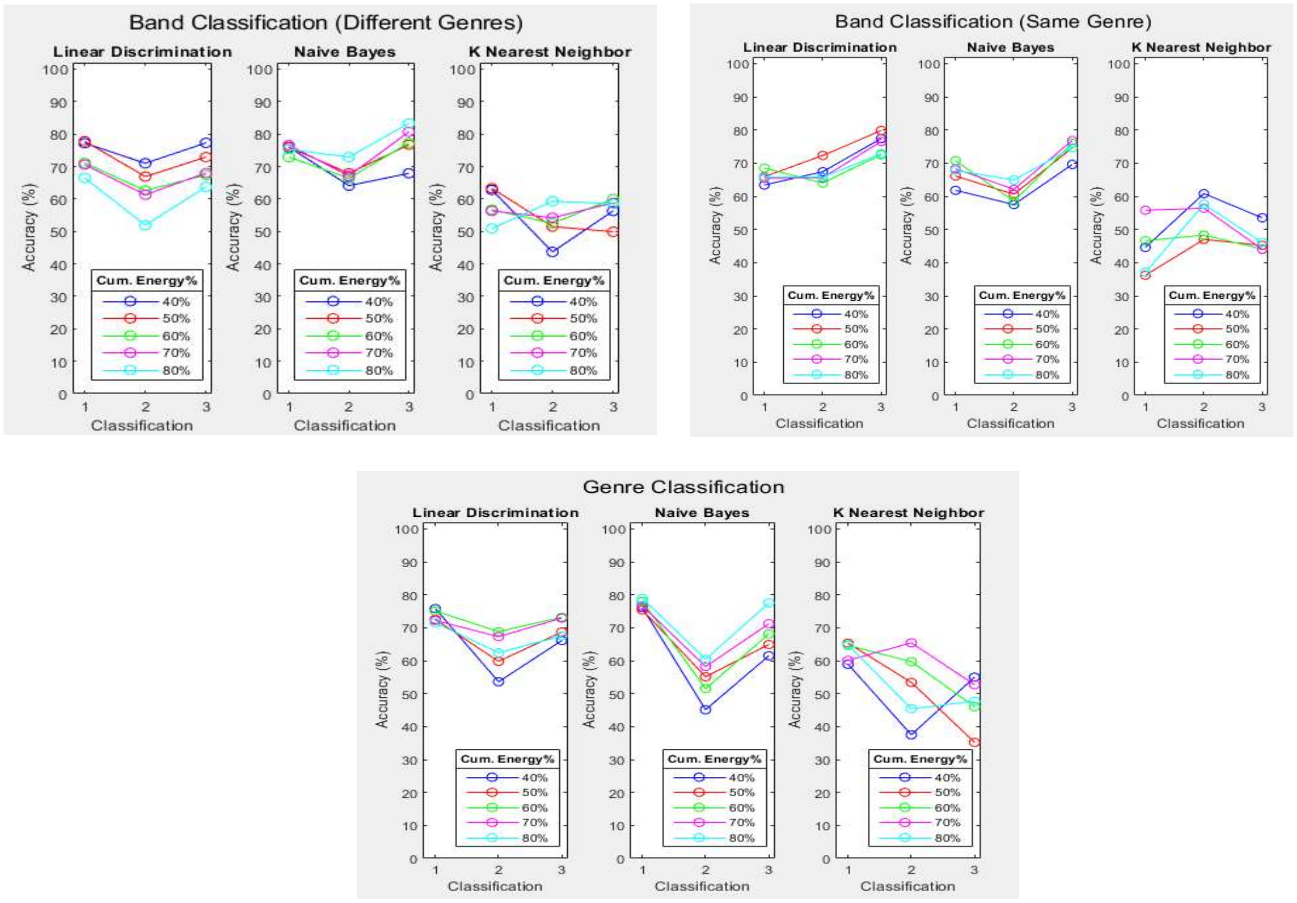


FIGURE 6: THE AVERAGE ACCURACY OVER 100 ITERATIONS FOR EACH CLASSIFICATION METHOD/TEST CASE ARE SUMMARIZED ABOVE.

5 Summary and Conclusions

In this analysis SVD provided a means to transform each of our datasets into an efficient ordered set of dimensions. This allowed us to capture maximum information in the minimum set of dimensions.

For the two Yale Face datasets, we could see that when the data is 'cleaned' (i.e. the cropped and aligned) that we could reconstruct a subject's face in a much low number of dimensions than with the Original Image dataset (i.e. the low rank approximation in the cropped data for the same quality in the original image dataset was much lower). This is because the variation in the images due to alignment, hairstyle and distance to the camera were minimized.

The results of the music classification were not as expected. It did not show significantly higher singular values or classification accuracy percentages for mands in different genres versus bands in the same genre. Nor did we see that increasing the rank of the approximation always lead to better classification results. I believe this is likely due to the varying quality of the audio clips that were collected from different sources. There is also the question of bands often having wildly different styles among and even within the same song. Those clips that might've been a snippet of audio without the band's lead singer could so have confounded the results. The positive aspect of the classification is that all three classification methods were able to provide, measured with 99 cross validation iterations, above 33% (accuracy if randomly selected), and Linear Discrimination Analysis and Naïve Bayes returned significantly better results than K Nearest Neighbor. Likely increasing size of the dataset and equalizing the quality of the sound sources would've helped accuracy in all regards.

Overall, we can see that transforming data sets using SVD, gives us insight into the variations within the feature space of a dataset in very efficient and easily accessible format.

References

[1] J. Nathan Kutz 2013, Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data, Oxford University Press

Appendix A: Relevant MATLAB functions

- dir(): Provides a list of the filenames and locations in a given path. This was used to import each image/audio clip in the respective datasets.
- imread(): Reads image files (used import Yale Faces)
- double(): converts value to double precision to allow for calculations
- svd(): Performs SVD on a given matrix, outputting matrices [U,S, V].
- diag(): converts a diagonal matrix (extracting the non-zero values) to a column vector
- find(): find the index values of a specific non-zero value in an array
- imagesc(): displays image with scaled colors
- reshape(): reshapes and array into the selected dimensions
- audioread(): reads audio files as two outputs, Y (sampled data) and FS (sample rate in Hertz)
- uint64(): converts value to unsigned 64-bit integer
- randperm(): generates an N length vector containing a random permutation of the integers 1:N
- classify(): function used to perform the Linear Discrimination Analysis (supervised classification)
- nb.predict(): function used to perform Naïve Bayes classification (supervised)
- knnsearch(): function used to perform K Nearest Neighbor classification (supervised)
- repmat(): replicate and tile and array. This was used in normalization calculations for a matrix
- 'for loop' (honorable mention): This isn't a function but the 'for loop' allowed us to perform iterations across all 20 measurements with minimal coding.

Appendix B: MATLAB Code

```
clear all; close all; clc;
pgmfiles=dir('CroppedYale/yale*/*.pgm');
file_count = length(pgmfiles);
A_cropped = zeros(192*168, file_count);

for jj = 1:file_count
    pic = imread(fullfile(pgmfiles(jj).folder, pgmfiles(jj).name));
    p_vector =pic(:);
    A_cropped(:, jj) = double(p_vector);
end
%
[m,n] = size(A_cropped);
mn=mean(A_cropped,2); %compute mean for each row
A_mean=A_cropped-repmat(mn, 1, n); % subtract mean
[u,s,v] = svd(A_mean, 'econ');
 sig = diag(s);
 %
energy_all = zeros(file_count, 2);

for ee = 1:file_count
    energy_s = sum(sig(1:ee))/sum(sig) ;
    energy_all(ee, 2) = energy_s;
    energy_all(ee, 1) = ee;
end
%
 figure(1)
 sgtitle('Cropped & Aligned:  Singular Values (\sigma) by mode')
 subplot(2,2,1)
plot(sig, 'bo', 'Linewidth', [1]);
xlabel('mode')
ylabel('\sigma')
subplot(2,2,2)
semilogy(sig,'bo','Linewidth',[1])
xlabel('mode')
ylabel('\sigma (log)')
subplot(2,1, 2)
plot(energy_all(:, 2)*100, 'b:', 'Linewidth', [4])
title('Cumulative Energy by mode')
ylabel('% energy (\sigma)')
xlabel('mode')

energy_summ = zeros(10, 2);

for ii = 0.1:0.1: 1
thres = find(energy_all(:,2) >= ii);
min_loc = uint64(min(thres));
energy_summ(uint64(ii*10), :) = energy_all(min_loc, :);
end
%
s_table = table(energy_summ(:, 1), energy_summ(:, 2), 'VariableNames', {'Mode'; 'Energy %'});
% plot first four modes
figure(2)
sgtitle('Cropped Yale Faces');
subplot(3,7,4)
imagesc(reshape(mn, 192, 168)), colormap(gray);
title('Mean Face');
set(gca,'XColor', 'none','YColor','none')
```

```
subplot(3, 7, 8); imagesc(reshape(u(:, 1), 192, 168)); title('First Eigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')
subplot(3, 7, 9); imagesc(reshape(u(:, 2), 192, 168));  title('Second Eigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')
subplot(3, 7, 10); imagesc(reshape(u(:, 3), 192, 168)); title('Third Eigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')
subplot(3, 7, 11); imagesc(reshape(u(:, 4), 192, 168)); title('Fourth Eigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')
subplot(3, 7, 12); imagesc(reshape(u(:, 5), 192, 168));  title('Fifth Eigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')
subplot(3, 7, 13); imagesc(reshape(u(:, 6), 192, 168)); title('Sixth Eigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')
subplot(3, 7, 14); imagesc(reshape(u(:, 7), 192, 168)); title('Seventh Eigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')
subplot(3, 7, 15); imagesc(reshape(u(:, 8), 192, 168)); title('Eigth Eigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')
subplot(3, 7, 16); imagesc(reshape(u(:, 9), 192, 168)); title('Ninth Eigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')
subplot(3, 7, 17); imagesc(reshape(u(:, 10), 192, 168)); title('Tenth Eigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')
subplot(3, 7, 18); imagesc(reshape(u(:, 11), 192, 168)); title('EleventhEigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')
subplot(3, 7, 19); imagesc(reshape(u(:, 12), 192, 168)); title('Twelfth Eigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')
subplot(3, 7, 20); imagesc(reshape(u(:, 13), 192, 168)); title('Thirteenth Eigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')
subplot(3, 7, 21); imagesc(reshape(u(:, 14), 192, 168)); title('FourteenthEigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')

% reconstructing subject 1 and 2 at each 10% threshold of energy
s_tablem = table2array(s_table);s_vector = s_tablem(:, 1);

figure(4)
sgtitle('Cropped / Aligned:  Image Reconstruction');
for mm = 1: length(s_vector)-1;
    face_65 = u(:, 1:s_vector(mm, 1))*s(1:s_vector(mm, 1) ,:)*v(65, :)';
subplot(2,5, mm)
imagesc(reshape(face_65, 192, 168)), colormap(gray);;set(gca,'XColor', 'none','YColor','none')
title(strcat(num2str(mm*10), ' % energy'));
end
%
subplot(2,5,10)
imagesc(reshape(A_cropped(:, 65), 192, 168)), colormap(gray);set(gca,'XColor', 'none','YColor','none')
title('Original Image')
%% uncropped Yalefaces
clear all; close all; clc;
pm = 243; pn = 320;
pgmfiles=dir('yalefaces/subject*.*');
file_count = length(pgmfiles);
A= zeros(pm*pn, file_count);

for jj = 1:file_count
   pic = imread(fullfile(pgmfiles(jj).folder, pgmfiles(jj).name));
   p_vector =pic(:);
   A(:, jj) = double(p_vector);
end

[m,n] = size(A);
```

```
mn=mean(A,2); %compute mean for each row
A_mean=A -repmat(mn, 1, n); % subtract mean
[u,s,v] = svd(A_mean, 'econ');
 sig = diag(s);
energy_all = zeros(file_count, 2);

for ee = 1:file_count
    energy_s = sum(sig(1:ee))/sum(sig) ;
    energy_all(ee, 2) = energy_s;
    energy_all(ee, 1) = ee;
end
%
 figure(1)
 sgtitle('Uncropped Faces:  Singular Values (\sigma) by mode')
 subplot(2,2,1)
plot(sig, 'bo', 'Linewidth', [1]);
xlabel('mode')
ylabel('\sigma')
subplot(2,2,2)
semilogy(sig,'bo','Linewidth',[1])
xlabel('mode')
ylabel('\sigma (log)')
subplot(2,1, 2)
plot(energy_all(:, 2)*100, 'b:', 'Linewidth', [4])
title('Cumulative Energy by mode')
ylabel('% energy (\sigma)')
xlabel('mode')

energy_summ = zeros(10, 2);

for ii = 0.1:0.1: 1
thres = find(energy_all(:,2) >= ii);
min_loc = uint64(min(thres));
energy_summ(uint64(ii*10), :) = energy_all(min_loc, :);
end
%
s_table = table(energy_summ(:, 1), energy_summ(:, 2), 'VariableNames', {'Mode'; 'Energy %'});
% plot first four modes
figure(2)
sgtitle('Uncropped Yale Faces');
subplot(3,7,4)
imagesc(reshape(mn, pm, pn)) , colormap(gray);
title('Mean Face');
set(gca,'XColor', 'none','YColor','none')
subplot(3, 7, 8); imagesc(reshape(u(:, 1), pm,pn)); title('First Eigenface'),colormap(gray);set(gca,'XColor', 'none','YColor','none')
subplot(3, 7, 9); imagesc(reshape(u(:, 2), pm, pn));   title('Second Eigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')
subplot(3, 7, 10); imagesc(reshape(u(:, 3), pm, pn));  title('Third Eigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')
subplot(3, 7, 11); imagesc(reshape(u(:, 4), pm, pn)); title('Fourth Eigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')
subplot(3, 7, 12); imagesc(reshape(u(:, 5),pm, pn));  title('Fifth Eigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')
subplot(3, 7, 13); imagesc(reshape(u(:, 6), pm, pn));  title('Sixth Eigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')
subplot(3, 7, 14); imagesc(reshape(u(:, 7), pm, pn));  title('Seventh Eigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')
```

```
subplot(3, 7, 15); imagesc(reshape(u(:, 8), pm, pn));  title('Eigth Eigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')
subplot(3, 7, 16); imagesc(reshape(u(:, 9), pm, pn));  title('Ninth Eigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')
subplot(3, 7, 17); imagesc(reshape(u(:, 10),pm, pn));  title('Tenth Eigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')
subplot(3, 7, 18); imagesc(reshape(u(:, 11), pm, pn));  title('EleventhEigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')
subplot(3, 7, 19); imagesc(reshape(u(:, 12),pm, pn));  title('Twelfth Eigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')
subplot(3, 7, 20); imagesc(reshape(u(:, 13), pm, pn)); title('Thirteenth Eigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')
subplot(3, 7, 21); imagesc(reshape(u(:, 14), pm, pn));  title('FourteenthEigenface'),colormap(gray);set(gca,'XColor',
'none','YColor','none')

% reconstruct faces for subjects 1 and 2 at each 10% threshold increment.

s_tablem = table2array(s_table);s_vector = s_tablem(:, 1);

figure(3)
sgtitle('Uncropped/Not Aligned:  Image Reconstruction');
for mm = 1: length(s_vector)-1;
    face_13 = u(:, 1:s_vector(mm, 1))*s(1:s_vector(mm, 1) ,:)*v(13, :)';
subplot(2,5, mm)
imagesc(reshape(face_13, pm, pn)), colormap(gray);;set(gca,'XColor', 'none','YColor','none')
title(strcat(num2str(mm*10), ' % energy'));
end
subplot(2,5,10)
imagesc(reshape(A(:, 13), pm, pn)), colormap(gray);set(gca,'XColor', 'none','YColor','none')
title('Original Image')
```

%% music classification

```
clear all; close all; clc;
% sinatra clips
pgmfiles=dir('music/sinatra/*.mp3');
file_count = length(pgmfiles);

for ii = 1:file_count
   [y, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
   y = y(1:2:220601);
   v = y'/2; yf = fft(v); yfs = abs(fftshift(yf));
   yfs1(:, ii) = yfs;
end
L=5; n=110301; % set up time domain and fourier modes
t2=linspace(0,L,n+1); t= t2(1:n); % setup time vector based on signal data, periodic
k=(2*pi/L)*[0:n/2 -n/2:-1];  % setup freq vector for odd fourier mode (symmetrical)
ks=fftshift(k); % flip so ks has ordered modes with 0 at center
%
figure(1)
subplot(3,1,1);
sgtitle('Song Samples')
plot((1:length(v))/FS*2,v);
xlim([0 5.25]);
set(gca,'Fontsize',[9]) ;
ylabel('Amplitude');
title('Frank Sinatra')
```

```
% smiths clips
pgmfiles=dir('music/smiths/*.mp3');
file_count = length(pgmfiles);

for ii = 1:file_count
    [y, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
    y = y(1:2:220601);
    v = y'/2; yf = fft(v); yfs = abs(fftshift(yf));
    yfs2(:, ii) = yfs;
end
subplot(3,1,2);
plot((1:length(v))/FS*2,v);
xlim([0 5.25]);
set(gca,'Fontsize',[9]) ;
ylabel('Amplitude');
title('The Smiths')

% public enemy clips
pgmfiles=dir('music/pe/*.mp3');
file_count = length(pgmfiles);

for ii = 1:file_count
    [y, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
    y = y(1:2:220601);
     v = y'/2; yf = fft(v);, yfs = abs(fftshift(yf));
    yfs3(:, ii) = yfs;
end
subplot(3,1,3);
plot((1:length(v))/FS*2,v);
xlim([0 5.25]);
set(gca,'Fontsize',[9]) ;
xlabel('Time [sec]');
ylabel('Amplitude');
title('Public Enemy')

%
figure(2)
sgtitle('Spectral Mosaic by Artist')
subplot(1,3,1)
pcolor(1:file_count,ks,yfs1);
shading interp
set(gca,'Fontsize',[9])
colormap('hot')
colorbar(); title('Frank Sinatra')
xlabel('Song Clip No.');
ylabel('frequency (\omega)');
caxis([0 1600])
%

subplot(1,3,2)
pcolor(1:file_count,ks,yfs2);
shading interp
set(gca,'Fontsize',[9])
colormap('hot')
colorbar(); title('The Smiths')
xlabel('Song Clip No.');
ylabel('frequency (\omega)');
caxis([0 1600])
```

```
subplot(1,3,3)
pcolor(1:file_count,ks,yfs3);
shading interp
set(gca,'Fontsize',[9])
colormap('hot')
colorbar(); title('Public Enemy')
xlabel('Song Clip No.');
ylabel('frequency (\omega)');
caxis([0 1600])

%  Combine all matrices and perform SVD and find singular values
songs_ideal = horzcat(yfs1, yfs2, yfs3);
[m,n] = size(songs_ideal);
mn=mean(songs_ideal,2); %compute mean for each row
ideal_mean=songs_ideal -repmat(mn, 1, n); % subtract mean
%
[u,s,v] = svd(ideal_mean', 'econ');
 sig = diag(s);
energy_all = zeros(n, 2);

for ee = 1:n
    energy_s = sum(sig(1:ee))/sum(sig) ;
    energy_all(ee, 2) = energy_s;
    energy_all(ee, 1) = ee;
end
figure(3)
 sgtitle('Band Classification:  Singular Values (\sigma) by mode')
 subplot(2,2,1)
plot(sig, 'bo', 'Linewidth', [1]);
xlabel('mode')
ylabel('\sigma')
subplot(2,2,2)
semilogy(sig,'bo','Linewidth',[1])
xlabel('mode')
ylabel('\sigma (log)')
subplot(2,1, 2)
plot(energy_all(:, 2)*100, 'b:', 'Linewidth', [4])
title('Cumulative Energy by mode')
ylabel('% energy (\sigma)')

%% Supervised Learning
%% Cross valudation using 100 iter of sampled train vs test data.
m_vector = [40:10:80];
for mm = m_vector
modes = min(find(energy_all(:, 2) >mm/100));

for jj = 1:100
q1 = randperm(file_count);
q2 = randperm(file_count);
q3 = randperm(file_count);
m_vector =  [40: 10: 80];

x1= v(1:file_count, 1:modes);
x2 = v(file_count+1: file_count*2, 1:modes);
x3 = v((2*file_count)+ 1: file_count*3, 1:modes);

xtrain = [x1(q1(1:40), :); x2(q2(1:40), :); x3(q3(1:40), :)];
```

```
xtest = [x1(q1(41:50), :); x2(q2(41:50), :); x3(q3(41:50), :)];
ctrain = [ones(40, 1); 2*ones(40, 1); 3*ones(40, 1)];
pre1 = classify(xtest, xtrain, ctrain);
ans = [ones(10, 1); 2*ones(10, 1); 3*ones(10, 1)];
error1 = nnz(pre1(1:10) - 1);
accur1 = (10-error1)/10;
error2 = nnz(pre1(11:20) - 2);
accur2 = (10-error2)/10;
error3 = nnz(pre1(21:30) - 3);
accur3 = (10-error3)/10 ;
ld_accur(jj, :)  = horzcat(accur1, accur2, accur3);
ld_mean_accuracy = mean(ld_accur, 1);

nb = fitcnb(xtrain, ctrain);
pre2 = nb.predict(xtest);

error1 = nnz(pre2(1:10) - 1);
accur1 = (10-error1)/10;
error2 = nnz(pre2(11:20) - 2);
accur2 = (10-error2)/10;
error3 = nnz(pre2(21:30) - 3);
accur3 = (10-error3)/10 ;
nb_accur(jj, :)  = horzcat(accur1, accur2, accur3);
nb_mean_accuracy = mean(nb_accur, 1);

 [ind, D] = knnsearch(xtrain, xtest);
pre3 = ceil(ind/(file_count - 10));
error1 = nnz(pre3(1:10) - 1);
error2 = nnz(pre3(11:20) - 2);
error3 = nnz(pre3(21:30) - 3);
accur1 = (10-error1)/10;
accur2 = (10-error2)/10;
accur3 = (10-error3)/10 ;

knn_accur(jj, :)  = horzcat(accur1, accur2, accur3);
knn_mean_accuracy = mean(knn_accur, 1);
end

results((mm/10)-3, :) = [ld_mean_accuracy, nb_mean_accuracy,knn_mean_accuracy];

end

figure (4)
sgtitle('Band Classification: Sample Iteration')
subplot(1, 3, 1);
bar(pre1); hold on;
title('Linear Discrimination')
plot(ans, 'r:', 'Linewidth', [3]);
ylabel('Prediction')
xlabel('Clip Seq.')
%legend('Prediction', 'Expected', 'Location',  'NorthEastOutside')
hold off;
subplot(1, 3, 2);
bar(pre2); hold on;
title('Naive Bayes')
plot(ans, 'r:', 'Linewidth', [3]);
ylabel('Prediction')
xlabel('Clip Seq.')
```

```
%legend('Prediction', 'Expected', 'Location',  'NorthEastOutside')
hold off;

subplot(1, 3, 3);

bar(pre3); hold on;
title('K Nearest Neighbor')
plot(ans, 'r:', 'Linewidth', [3]);
ylabel('Prediction')
xlabel('Clip Seq.')
%legend('Prediction', 'Expected', 'Location',  'NorthEastOutside')
hold off;


%%
figure(5)
sgtitle('Band Classification (Different Genres)')
x = 1:3;
subplot(1,3, 1)
plot(x, results(1, 1:3)*100, 'b-o'); hold on;
plot(x, results(2, 1:3)*100, 'r-o');
plot(x, results(3, 1:3)*100, 'g-o');
plot(x, results(4, 1:3)*100, 'm-o');
plot(x, results(5, 1:3)*100, 'c-o');
xlim([.8 3.1]); xlabel('Classification')
ylim([0 102])
title('Linear Discrimination')
legend('40%', '50%', '60%','70%', '80%', 'Location', 'SouthEast')
title(legend,'Cum. Energy%')
ylabel('Accuracy (%)')
%
subplot(1,3,2)
plot(x, results(1,4:6)*100, 'b-o'); hold on;
plot(x, results(2, 4:6)*100, 'r-o');
plot(x, results(3, 4:6)*100, 'g-o');
plot(x, results(4, 4:6)*100, 'm-o');
plot(x, results(5, 4:6)*100, 'c-o');
xlim([.8 3.1]); xlabel('Classification')
ylim([0 102])
title('Naive Bayes')
legend('40%', '50%', '60%','70%', '80%', 'Location', 'SouthEast')
title(legend,'Cum. Energy%')
ylabel('Accuracy (%)')
%
subplot(1,3, 3)
plot(x, results(1, 7:9)*100, 'b-o'); hold on;
plot(x, results(2, 7:9)*100, 'r-o');
plot(x, results(3,7:9)*100, 'g-o');
plot(x, results(4, 7:9)*100, 'm-o');
plot(x, results(5, 7:9)*100, 'c-o');
xlim([.8 3.1]); xlabel('Classification')
ylim([0 102])
title('K Nearest Neighbor')
legend('40%', '50%', '60%','70%', '80%', 'Location', 'SouthEast')
title(legend,'Cum. Energy%')
ylabel('Accuracy (%)')

%% music classification:  test 2
```

```
clear all; close all; clc;
% sinatra clips
pgmfiles=dir('music/oasis/*.mp3');
file_count = length(pgmfiles);

for ii = 1:file_count
    [y, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
    y = y(1:2:220601);
    v = y'/2; yf = fft(v); yfs = abs(fftshift(yf));
    yfs1(:, ii) = yfs;
end
L=5; n=110301; % set up time domain and fourier modes
t2=linspace(0,L,n+1); t= t2(1:n); % setup time vector based on signal data, periodic
k=(2*pi/L)*[0:n/2 -n/2:-1];  % setup freq vector for odd fourier mode (symmetrical)
ks=fftshift(k); % flip so ks has ordered modes with 0 at center
%
figure(1)
subplot(3,1,1);
sgtitle('Song Samples')
plot((1:length(v))/FS*2,v);
xlim([0 5.25]);
set(gca,'Fontsize',[9]) ;
ylabel('Amplitude');
title('Oasis')

% smiths clips
pgmfiles=dir('music/blur/*.mp3');
file_count = length(pgmfiles);

for ii = 1:file_count
    [y, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
    y = y(1:2:220601);
    v = y'/2; yf = fft(v); yfs = abs(fftshift(yf));
    yfs2(:, ii) = yfs;
end
subplot(3,1,2);
plot((1:length(v))/FS*2,v);
xlim([0 5.25]);
set(gca,'Fontsize',[9]) ;
ylabel('Amplitude');
title('Blur')


% public enemy clips
pgmfiles=dir('music/verve/*.mp3');
file_count = length(pgmfiles);

for ii = 1:file_count
    [y, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
    y = y(1:2:220601);
     v = y'/2; yf = fft(v);, yfs = abs(fftshift(yf));
    yfs3(:, ii) = yfs;
end
subplot(3,1,3);
plot((1:length(v))/FS*2,v);
xlim([0 5.25]);
set(gca,'Fontsize',[9]) ;
xlabel('Time [sec]');
```

```
ylabel('Amplitude');
title('The Verve')
%
figure(2)
sgtitle('Spectral Mosaic by Artist')
subplot(1,3,1)
pcolor(1:file_count,ks,yfs1);
shading interp
set(gca,'Fontsize',[9])
colormap('hot')
colorbar(); title('Oasis'); xlabel('Song Clip (5 Sec)');
xlabel('Song Clip No.');
ylabel('frequency (\omega)');
caxis([0 1600])

subplot(1,3,2)
pcolor(1:file_count,ks,yfs2);
shading interp
set(gca,'Fontsize',[9])
colormap('hot')
colorbar(); title('Blur');
xlabel('Song Clip No.');
ylabel('frequency (\omega)');
caxis([0 1600])

subplot(1,3,3)
pcolor(1:file_count,ks,yfs3);
shading interp
set(gca,'Fontsize',[9])
colormap('hot')
colorbar(); title('The Verve');
xlabel('Song Clip No.');
ylabel('frequency (\omega)');
caxis([0 1600])

%  Combine all matrices and perform SVD and find singular values
 songs_ideal = horzcat(yfs1, yfs2, yfs3);
[m,n] = size(songs_ideal);
mn=mean(songs_ideal,2); %compute mean for each row
ideal_mean=songs_ideal -repmat(mn, 1, n); % subtract mean
%
[u,s,v] = svd(ideal_mean', 'econ');
 sig = diag(s);
energy_all = zeros(n, 2);

for ee = 1:n
    energy_s = sum(sig(1:ee))/sum(sig) ;
    energy_all(ee, 2) = energy_s;
    energy_all(ee, 1) = ee;
end
figure(3)
 sgtitle('Same Genre:  Singular Values (\sigma) by mode')
 subplot(2,2,1)
plot(sig, 'bo', 'Linewidth', [1]);
xlabel('mode')
ylabel('\sigma')
subplot(2,2,2)
semilogy(sig,'bo','Linewidth',[1])
```

```
xlabel('mode')
ylabel('\sigma (log)')
subplot(2,1, 2)
plot(energy_all(:, 2)*100, 'b:', 'Linewidth', [4])
title('Cumulative Energy by mode')
ylabel('% energy (\sigma)')

%% Supervised Learning
%% Cross valudation using 10 iter of sampled train vs test data.
m_vector = [40:10:80];
for mm = m_vector
modes = min(find(energy_all(:, 2) >mm/100));

for jj = 1:100
q1 = randperm(file_count);
q2 = randperm(file_count);
q3 = randperm(file_count);
m_vector =  [40: 10: 80];

x1= v(1:file_count, 1:modes);
x2 = v(file_count+1: file_count*2, 1:modes);
x3 = v((2*file_count)+ 1: file_count*3, 1:modes);

xtrain = [x1(q1(1:40), :); x2(q2(1:40), :); x3(q3(1:40), :)];
xtest = [x1(q1(41:50), :); x2(q2(41:50), :); x3(q3(41:50), :)];
ctrain = [ones(40, 1); 2*ones(40, 1); 3*ones(40, 1)];
pre1 = classify(xtest, xtrain, ctrain);
ans = [ones(10, 1); 2*ones(10, 1); 3*ones(10, 1)];
error1 = nnz(pre1(1:10) - 1);
accur1 = (10-error1)/10;
error2 = nnz(pre1(11:20) - 2);
accur2 = (10-error2)/10;
error3 = nnz(pre1(21:30) - 3);
accur3 = (10-error3)/10 ;
ld_accur(jj, :)  = horzcat(accur1, accur2, accur3);
ld_mean_accuracy = mean(ld_accur, 1);

nb = fitcnb(xtrain, ctrain);
pre2 = nb.predict(xtest);

error1 = nnz(pre2(1:10) - 1);
accur1 = (10-error1)/10;
error2 = nnz(pre2(11:20) - 2);
accur2 = (10-error2)/10;
error3 = nnz(pre2(21:30) - 3);
accur3 = (10-error3)/10 ;
nb_accur(jj, :)  = horzcat(accur1, accur2, accur3);
nb_mean_accuracy = mean(nb_accur, 1);
%
[ind, D] = knnsearch(xtrain, xtest);
pre3 = ceil(ind/(file_count - 10));
error1 = nnz(pre3(1:10) - 1);
error2 = nnz(pre3(11:20) - 2);
error3 = nnz(pre3(21:30) - 3);
accur1 = (10-error1)/10;
accur2 = (10-error2)/10;
accur3 = (10-error3)/10 ;
```

```
knn_accur(jj, :)  = horzcat(accur1, accur2, accur3);
knn_mean_accuracy = mean(knn_accur, 1);
end

results((mm/10)-3, :) = [ld_mean_accuracy, nb_mean_accuracy,knn_mean_accuracy];

end

figure (4)
sgtitle('Same Genre:  Sample Iteration')
subplot(1, 3, 1);
bar(pre1); hold on;
title('Linear Discrimination')
plot(ans, 'r:', 'Linewidth', [3]);
ylabel('Prediction')
xlabel('Clip Seq.')
legend('Prediction', 'Expected', 'Location',  'NorthEastOutside')
hold off;
subplot(1, 3, 2);
bar(pre2); hold on;
title('Naive Bayes')
plot(ans, 'r:', 'Linewidth', [3]);
ylabel('Prediction')
xlabel('Clip Seq.')
legend('Prediction', 'Expected', 'Location',  'NorthEastOutside')
hold off;

subplot(1, 3, 3);
bar(pre3); hold on;
title('K Nearest Neighbor')
plot(ans, 'r:', 'Linewidth', [3]);
ylabel('Prediction')
xlabel('Clip Seq.')
legend('Prediction', 'Expected', 'Location',  'NorthEastOutside')
hold off;

%%
figure(5)
sgtitle('Band Classification (Same Genre)')
x = 1:3;
subplot(1,3, 1)
plot(x, results(1, 1:3)*100, 'b-o'); hold on;
plot(x, results(2, 1:3)*100, 'r-o');
plot(x, results(3, 1:3)*100, 'g-o');
plot(x, results(4, 1:3)*100, 'm-o');
plot(x, results(5, 1:3)*100, 'c-o');
xlim([.8 3.1]); xlabel('Classification')
ylim([0 102])
title('Linear Discrimination')
legend('40%', '50%', '60%','70%', '80%', 'Location', 'SouthEast')
title(legend,'Cum. Energy%')
ylabel('Accuracy (%)')
%
subplot(1,3,2)
plot(x, results(1,4:6)*100, 'b-o'); hold on;
plot(x, results(2, 4:6)*100, 'r-o');
plot(x, results(3, 4:6)*100, 'g-o');
plot(x, results(4, 4:6)*100, 'm-o');
plot(x, results(5, 4:6)*100, 'c-o');
xlim([.8 3.1]); xlabel('Classification')
ylim([0 102])
title('Naive Bayes')
legend('40%', '50%', '60%','70%', '80%', 'Location', 'SouthEast')
title(legend,'Cum. Energy%')
ylabel('Accuracy (%)')
%
subplot(1,3, 3)
plot(x, results(1, 7:9)*100, 'b-o'); hold on;
plot(x, results(2, 7:9)*100, 'r-o');
plot(x, results(3,7:9)*100, 'g-o');
plot(x, results(4, 7:9)*100, 'm-o');
plot(x, results(5, 7:9)*100, 'c-o');
xlim([.8 3.1]); xlabel('Classification')
ylim([0 102])
title('K Nearest Neighbor')
legend('40%', '50%', '60%','70%', '80%', 'Location', 'SouthEast')
title(legend,'Cum. Energy%')
ylabel('Accuracy (%)')

%% music classification (test3)
clear all; close all; clc;
% sinatra clips
pgmfiles=dir('music/britpop/*.mp3');
file_count = length(pgmfiles);

for ii = 1:file_count
   [y, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
   y = y(1:2:220601);
   v = y'/2; yf = fft(v); yfs = abs(fftshift(yf));
   yfs1(:, ii) = yfs;
end
L=5; n=110301; % set up time domain and fourier modes
t2=linspace(0,L,n+1); t= t2(1:n); % setup time vector based on signal data, periodic
k=(2*pi/L)*[0:n/2 -n/2:-1];  % setup freq vector for odd fourier mode (symmetrical)
ks=fftshift(k); % flip so ks has ordered modes with 0 at center
%
figure(1)
subplot(3,1,1);
sgtitle('Song Samples')
plot((1:length(v))/FS*2,v);
xlim([0 5.25]);
set(gca,'Fontsize',[9]) ;
ylabel('Amplitude');
title('British Pop/Rock')

% smiths clips
pgmfiles=dir('music/classical/*.mp3');
file_count = length(pgmfiles);

for ii = 1:file_count
   [y, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
   y = y(1:2:220601);
   v = y'/2; yf = fft(v); yfs = abs(fftshift(yf));
   yfs2(:, ii) = yfs;
end
subplot(3,1,2);
```

```
plot((1:length(v))/FS*2,v);
xlim([0 5.25]);
set(gca,'Fontsize',[9]) ;
ylabel('Amplitude');
title('Classical')

% public enemy clips
pgmfiles=dir('music/jazz/*.mp3');
file_count = length(pgmfiles);

for ii = 1:file_count
    [y, FS] = audioread(fullfile(pgmfiles(ii).folder, pgmfiles(ii).name));
    y = y(1:2:220601);
     v = y'/2; yf = fft(v);, yfs = abs(fftshift(yf));
    yfs3(:, ii) = yfs;
end
subplot(3,1,3);
plot((1:length(v))/FS*2,v);
xlim([0 5.25]);
set(gca,'Fontsize',[9]) ;
xlabel('Time [sec]');
ylabel('Amplitude');
title('Classic Jazz')
%
figure(2)
sgtitle('Spectral Mosaic by Artist')
subplot(1,3,1)
pcolor(1:file_count,ks,yfs1);
shading interp
set(gca,'Fontsize',[9])
colormap('hot')
colorbar(); title('Brit Pop/Rock'); xlabel('Song Clip No.');
ylabel('frequency (\omega)');

subplot(1,3,2)
pcolor(1:file_count,ks,yfs2);
shading interp
set(gca,'Fontsize',[9])
colormap('hot')
colorbar(); title('Classical');
xlabel('Song Clip No.');
ylabel('frequency (\omega)');

subplot(1,3,3)
pcolor(1:file_count,ks,yfs3);
shading interp
set(gca,'Fontsize',[9])
colormap('hot')
colorbar(); title('Classic Jazz');xlabel('Song Clip No.');
ylabel('frequency (\omega)');


%  Combine all matrices and perform SVD and find singular values

songs_ideal = horzcat(yfs1, yfs2, yfs3);
[m,n] = size(songs_ideal);
mn=mean(songs_ideal,2); %compute mean for each row
ideal_mean=songs_ideal -repmat(mn, 1, n); % subtract mean
```

```
%
[u,s,v] = svd(ideal_mean', 'econ');
 sig = diag(s);
energy_all = zeros(n, 2);

for ee = 1:n
    energy_s = sum(sig(1:ee))/sum(sig) ;
    energy_all(ee, 2) = energy_s;
    energy_all(ee, 1) = ee;
end
figure(3)
 sgtitle('Genre Classification:  Singular Values (\sigma) by mode')
 subplot(2,2,1)
plot(sig, 'bo', 'Linewidth', [1]);
xlabel('mode')
ylabel('\sigma')
subplot(2,2,2)
semilogy(sig,'bo','Linewidth',[1])
xlabel('mode')
ylabel('\sigma (log)')
subplot(2,1, 2)
plot(energy_all(:, 2)*100, 'b:', 'Linewidth', [4])
title('Cumulative Energy by mode')
ylabel('% energy (\sigma)')


%% Supervised Learning
%% Cross valudation using 10 iter of sampled train vs test data.
m_vector = [40:10:80];
for mm = m_vector
modes = min(find(energy_all(:, 2) >mm/100));

for jj = 1:100
q1 = randperm(file_count);
q2 = randperm(file_count);
q3 = randperm(file_count);
m_vector =  [40: 10: 80];

x1= v(1:file_count, 1:modes);
x2 = v(file_count+1: file_count*2, 1:modes);
x3 = v((2*file_count)+ 1: file_count*3, 1:modes);

xtrain = [x1(q1(1:40), :); x2(q2(1:40), :); x3(q3(1:40), :)];
xtest = [x1(q1(41:50), :); x2(q2(41:50), :); x3(q3(41:50), :)];
ctrain = [ones(40, 1); 2*ones(40, 1); 3*ones(40, 1)];
pre1 = classify(xtest, xtrain, ctrain);
ans = [ones(10, 1); 2*ones(10, 1); 3*ones(10, 1)];
error1 = nnz(pre1(1:10) - 1);
accur1 = (10-error1)/10;
error2 = nnz(pre1(11:20) - 2);
accur2 = (10-error2)/10;
error3 = nnz(pre1(21:30) - 3);
accur3 = (10-error3)/10 ;
ld_accur(jj, :)  = horzcat(accur1, accur2, accur3);
ld_mean_accuracy = mean(ld_accur, 1);

nb = fitcnb(xtrain, ctrain);
pre2 = nb.predict(xtest);
```

```
error1 = nnz(pre2(1:10) - 1);
accur1 = (10-error1)/10;
error2 = nnz(pre2(11:20) - 2);
accur2 = (10-error2)/10;
error3 = nnz(pre2(21:30) - 3);
accur3 = (10-error3)/10 ;
nb_accur(jj, :)  = horzcat(accur1, accur2, accur3);
nb_mean_accuracy = mean(nb_accur, 1);

%
[ind, D] = knnsearch(xtrain, xtest);
pre3 = ceil(ind/(file_count - 10));
error1 = nnz(pre3(1:10) - 1);
error2 = nnz(pre3(11:20) - 2);
error3 = nnz(pre3(21:30) - 3);
accur1 = (10-error1)/10;
accur2 = (10-error2)/10;
accur3 = (10-error3)/10 ;

knn_accur(jj, :)  = horzcat(accur1, accur2, accur3);
knn_mean_accuracy = mean(knn_accur, 1);
end
results((mm/10)-3, :) = [ld_mean_accuracy, nb_mean_accuracy,knn_mean_accuracy];

end
%%
figure (4)
sgtitle('Genre Classification:  Sample Iteration')
subplot(1, 3, 1);
bar(pre1); hold on;
title('Linear Discrimination')
plot(ans, 'r:', 'Linewidth', [3]);
ylabel('Prediction')
xlabel('Clip Seq.')
%legend('Prediction', 'Expected', 'Location',  'NorthEastOutside')
hold off;
subplot(1, 3, 2);
bar(pre2); hold on;
title('Naive Bayes')
plot(ans, 'r:', 'Linewidth', [3]);
ylabel('Prediction')
xlabel('Clip Seq.')
%legend('Prediction', 'Expected', 'Location',  'NorthEastOutside')
hold off;

subplot(1, 3, 3);
bar(pre3); hold on;
title('K Nearest Neighbor')
plot(ans, 'r:', 'Linewidth', [3]);
ylabel('Prediction')
xlabel('Clip Seq.')
%legend('Prediction', 'Expected', 'Location',  'NorthEastOutside')
hold off;



%%
```

```
figure(5)
sgtitle('Genre Classification')
x = 1:3;
subplot(1,3, 1)
plot(x, results(1, 1:3)*100, 'b-o'); hold on;
plot(x, results(2, 1:3)*100, 'r-o');
plot(x, results(3, 1:3)*100, 'g-o');
plot(x, results(4, 1:3)*100, 'm-o');
plot(x, results(5, 1:3)*100, 'c-o');
xlim([.8 3.1]); xlabel('Classification')
ylim([0 102])
title('Linear Discrimination')
legend('40%', '50%', '60%','70%', '80%', 'Location', 'SouthEast')
title(legend,'Cum. Energy%')
ylabel('Accuracy (%)')
%
subplot(1,3,2)
plot(x, results(1,4:6)*100, 'b-o'); hold on;
plot(x, results(2, 4:6)*100, 'r-o');
plot(x, results(3, 4:6)*100, 'g-o');
plot(x, results(4, 4:6)*100, 'm-o');
plot(x, results(5, 4:6)*100, 'c-o');
xlim([.8 3.1]); xlabel('Classification')
ylim([0 102])
title('Naive Bayes')
legend('40%', '50%', '60%','70%', '80%', 'Location', 'SouthEast')
title(legend,'Cum. Energy%')
ylabel('Accuracy (%)')
%
subplot(1,3, 3)
plot(x, results(1, 7:9)*100, 'b-o'); hold on;
plot(x, results(2, 7:9)*100, 'r-o');
plot(x, results(3,7:9)*100, 'g-o');
plot(x, results(4, 7:9)*100, 'm-o');
plot(x, results(5, 7:9)*100, 'c-o');
xlim([.8 3.1]); xlabel('Classification')
ylim([0 102])
title('K Nearest Neighbor')
legend('40%', '50%', '60%','70%', '80%', 'Location', 'SouthEast')
title(legend,'Cum. Energy%')
ylabel('Accuracy (%)')
```