Math 444: Project 1

Levent Batakci

February 21, 2021

The work in this project focuses on data visualization and basic clustering. All of the mentioned implementations are coded in MATLAB and can be found in my public GitHub Repository.

Problem 1 - Model Reduction Data

The first data set consisted of data vectors $x^{(j)} \in \mathbb{R}^6$, with $1 \leq j \leq 4000$. The data is encoded as a matrix $X \in \mathbb{R}^{6\times 4000}$ and is loaded from the file *ModelReductionData.mat*. As an assessment of the raw data, we produced 2D scatter plots (Figure 1) corresponding to each pairing of components.

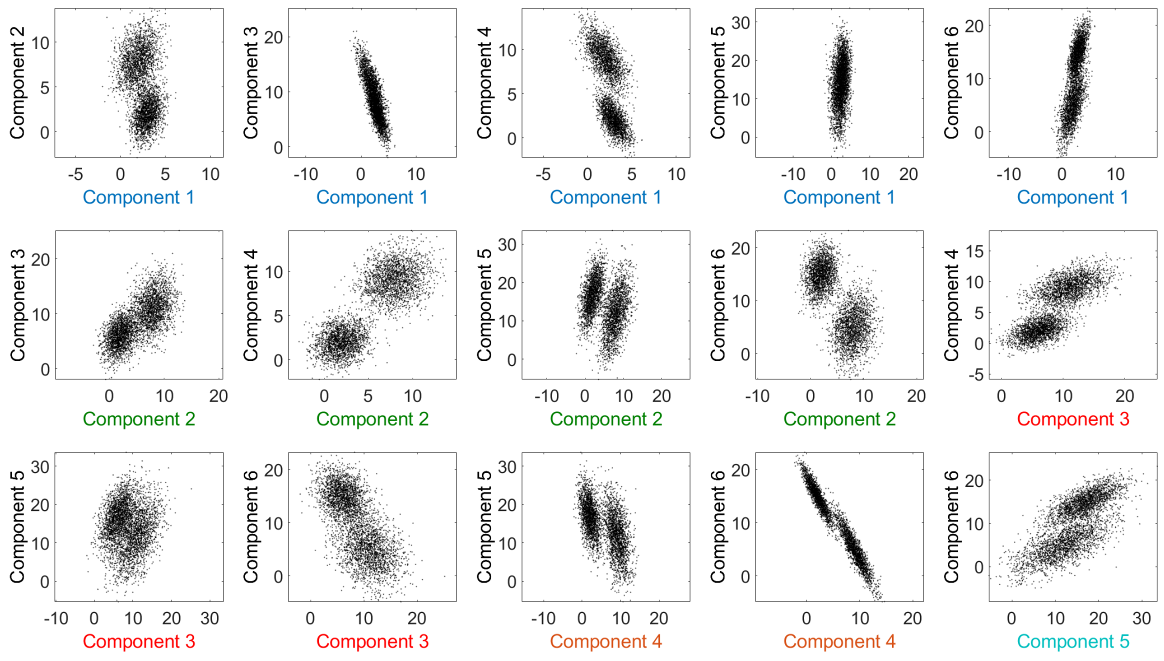

Figure 1: Raw Data Components Pair-wise Comparisons

No more than two apparent clusters are visible in any of the pair-wise plots. To simplify the problem of identifying clusters, we investigated whether the data's dimensionality could be reduced.

We decided to center the data so that performing a Principal Component Analysis would yield interpretations more accurate to how the data is distributed about its own center. When detecting clusters, this is generally preferable to working with interpretations about how the data distributes about the center of space (uncentered data). To compute the centered data X_c, we subtract the center of mass

$\bar{x} = \frac{1}{4000}\sum_{j=1}^{4000} x^{(j)}$ from each column of X. This essentially produces a shifted version of the data whose center is the origin.

After that, we compute the Singular Value Decomposition (SVD) of X_c. The SVD of X_c rewrites it as a product of a orthogonal matrix U, diagonal matrix D, and orthogonal matrix V^T (in that order). We computed the SVD of the centered data by using MATLAB's built-in algorithm 'svd'.

The columns of U can be viewed as feature vectors - they form a basis and reflect how the data distribute. The diagonal entries of D are the singular values of X_c and form a non-increasing, non-negative sequence. In a rough sense, the singular values reflect the relative importance of the associated feature vector in representing the data.

To be precise, larger singular values indicate a larger spread of the data when projected onto the associated feature vector. This is the key idea that drives Principal Component Analysis - comparing the scalar projections (principal components) of data onto feature vectors. The matrix of the first k principal components can be computed as $Z_k = [u_1 \cdots u_k]^T X$.

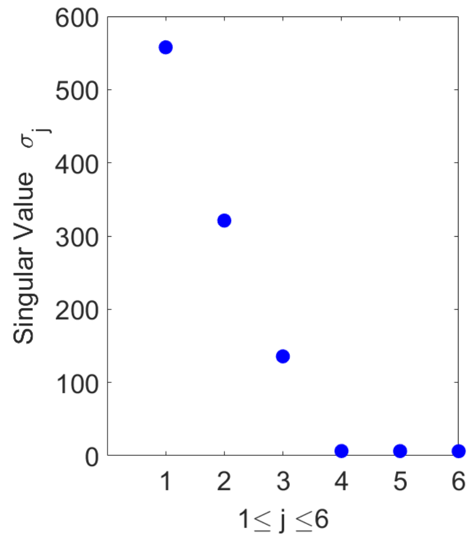

Figure 2: Centered Data (X_C) Singular Values

Once the SVD was computed, the singular values were extracted and plotted. The resulting plot (Figure 2) shows that the first three singular values are significantly larger than the rest. This leads to the conclusion that the data X effectively has a 3 dimensions.

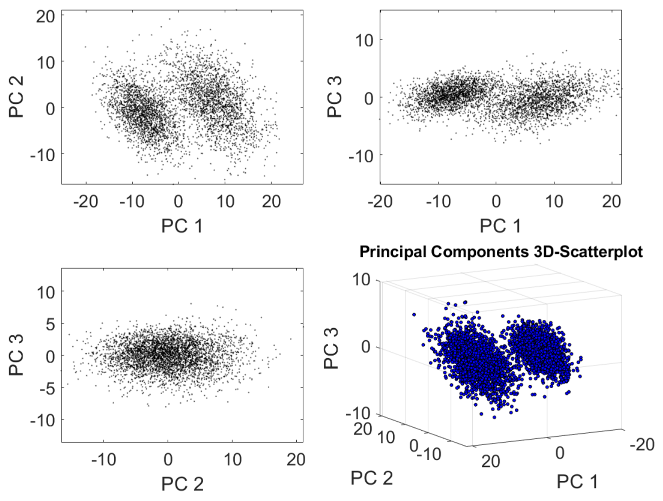

Figure 3: Centered Data (X_c) Principal Components (PC) Compared

From there, we computed the matrix Z_3 of the first 3 principal components. After that, we compared the first three principal components with 2D scatter plots and a 3D scatter plot (Figure 3). From these graphics, we were able to confirm that there are two apparent clusters in the data.

These results show practically the power of Principal Component Analysis. By reducing the dimensions of this data from 6 to 3, we were able to visually analyze the data and how its distributed. This enabled us to confidently conclude that the data consists of two main clusters.

Problem 2 - Biopsy Data

The next data we analyzed was the Breast Cancer Wisconsin data set. Particularly, we are interested to see if the data lends itself to a clustering which can distinguish benign and malignant tumors. The data is of the form

$$x^{(j)} = \begin{pmatrix} \text{Clump Thickness} \\ \text{Uniformity of Cell Size} \\ \text{Uniformity of Cell Shape} \\ \text{Marginal Adhesion} \\ \text{Simple Epithelial Cell Size} \\ \text{Bare Nuclei} \\ \text{Bland Chromatin} \\ \text{Normal Nucleoli} \\ \text{Mitoses} \end{pmatrix},$$

where $1 \leq j \leq 699$, and each of the components of data components is a integer between 1 and 10. Furthermore, missing data components are represent with NaN, which denotes ”Not a Number” in MATLAB. The data is encoded as a matrix $X \in \mathbb{R}^{9\times 699}$ and is loaded from the file *BiopsyData.mat*.

As a preliminary step, we cleaned to the data by removing all data points with missing components. It turned out that 16 data points had missing components. That is, our cleaned data matrix contains 683 data points. The matrix was easily cleaned in MATLAB with the following command.

```
    X(:, any(isnan(X))) = [];
```

After that, we performed a Principal Component Analysis. Again, we centered the data. After doing so, we computed the SVD and analyzed the singular values. Figure 4 shows a plot of the singular values of the centered, cleaned data matrix.

Immediately, it is apparent that the first singular value is significantly larger than all of the others. For this reason, we chose to isolate and analyze the first principal components of the data.

We chose to plot the first PC's of the data as semi-transparent ($\alpha = 0.2$) markers. This leads to a plot where darker colors indicate a higher density of points. This result can be seen in Figure 5. Note that the y-axis in this figure has no significance since we are only analyzing a distribution along the first principal component.

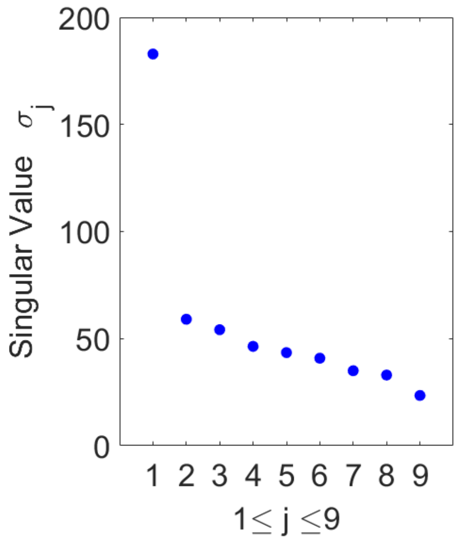

Figure 4: Centered Data Singular Values

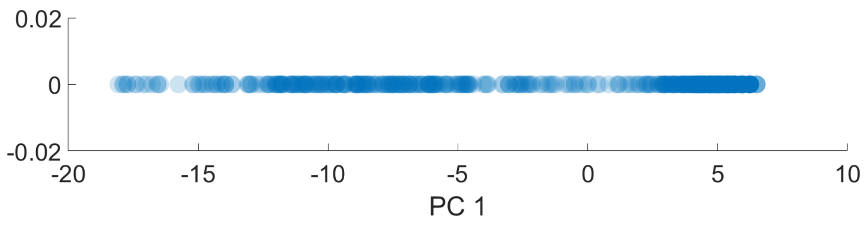

Figure 5: Distribution of First Principal Component

Immediately, it is clear that there two separated, dark regions on this plot. So we conclude that the first principal component does suggest a natural clustering of the data into two groups. Note that we do not know whether these two clusters have anything to do with the malignant/benign nature of the tumors. Without annotations of the data, it is impossible to make concrete conclusions about what real-life categories the clusters correspond to.

Note: The unshown principal components do not offer any insight as to the clustering of the data. For more details, see the appendix.

Problem 3 - Iris Data

The next data we analyzed was the Iris data set. This data sets consists of data from three different subspecies of Iris (flowers) - *Iris setosa*, *Iris virginica*, and *Iris versicolor*. We are interested to see if the data lends itself to a natural clustering which can distinguish between these three subspecies. The data is of the form

$$x^{(j)} = \begin{pmatrix} \text{petal length in cm} \\ \text{sepal width in cm} \\ \text{petal length in cm} \\ \text{petal width in cm} \end{pmatrix},$$

where $1 \leq j \leq 150$. The data is encoded as a matrix $X \in \mathbb{R}^{9\times 699}$ and is loaded from the file *IrisData.mat*.

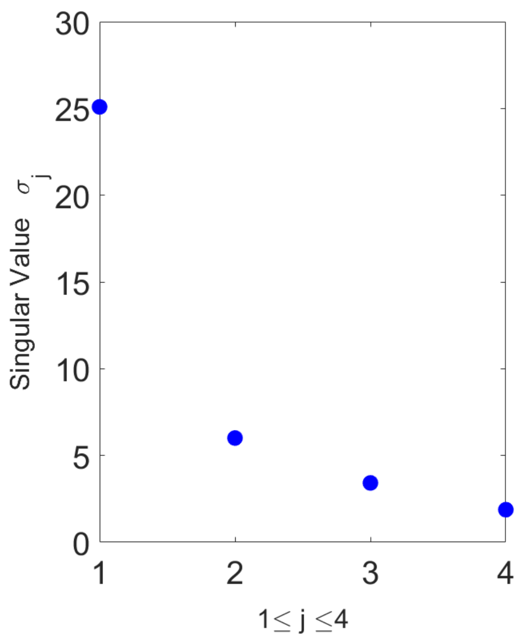

Figure 6: Centered Data Singular Values

Again, we performed a Principal Component Analysis to identify natural clustering in the data As with both data sets before, we centered the data and computed the SVD. We plotted the singular values, which can be seen in Figure 6.

As with the Biopsy data, it is immediately clear that the first singular value is significantly greater than the rest. For this reason, our first choice was to isolate and analyze the distribution of the first principal components.

The results indicated clustering - but only indicated the presence of two clusters. This indicates that the first principal component can likely be used to tell apart one of the species from the others. The two first-principal component clusters can be seen clearly below in Figure 7.

Since the first principal component does not provide evidence of 3 clusters in the data, we decided to look at pair-wise comparisons of all four principal components.

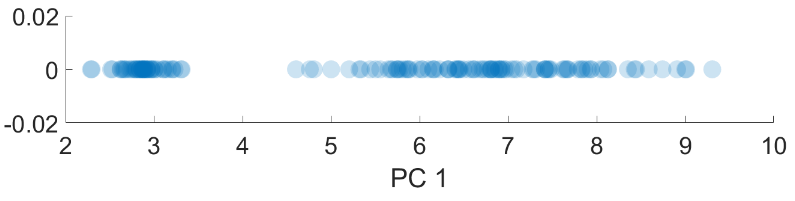

Figure 7: Distribution of First Principal Component

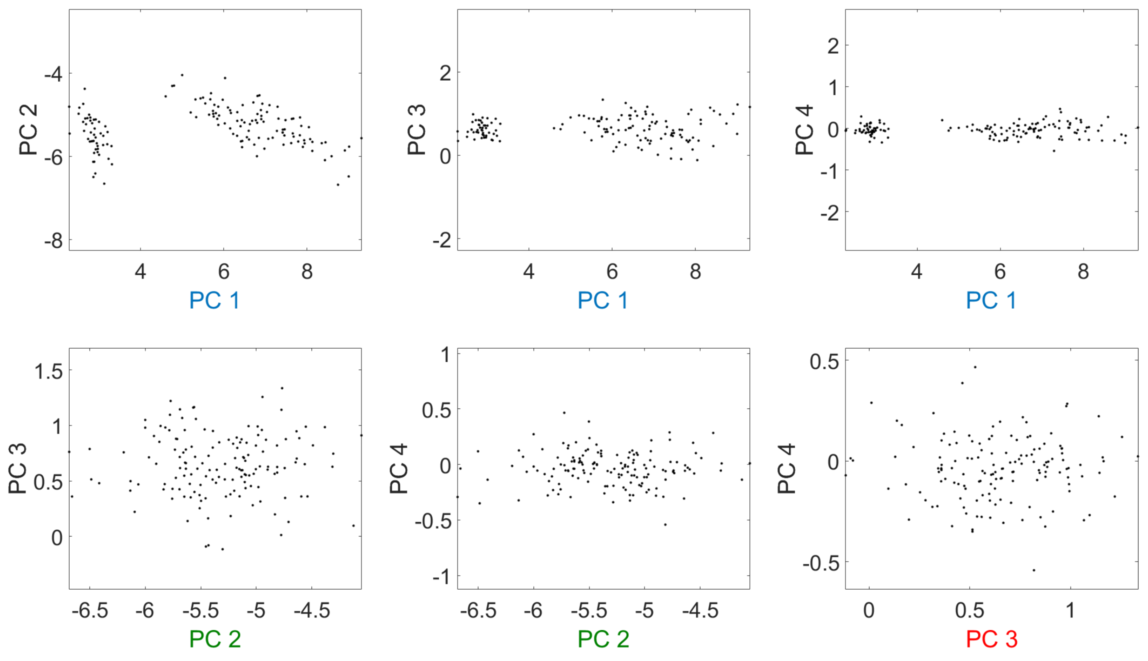

Figure 8: Centered Data Principal Components (PC) Pairwise Comparisons

All of the 6 pairwise comparison of the 4 principal components can be seen above in Figure 8. Each figure show at most two clear clusters. It is unfortunately clear that there are no obvious 3 clusters. That being said, it is very possible that a computer would be able to 'see' three clusters here. The only thing more impressive than how capable the human brain is how limited it is.

Problem 4 - Aligned Face Data

The final data we analyzed was the Yale Aligned Faces data set. This data sets consists of images of 15 individuals. Specifically, there are 11 images of each individual, with each image corresponding to a specific category. Each data point is a 28341 component vector encoding a 201×141 pixel grayscale images. Each component is between 0 and 255. The data is stored in a matrix $X \in \mathbb{R}^{28341\times165}$ and is loaded from the file *AlignedYaleFaces.mat*. More than clustering, we are interested in how well low-rank approximations approximate the data.

For the sake of this analysis, we've focused on the 5 categories corresponding to the facial expressions 'sad', 'surprised', 'happy', 'sleepy', and 'wink'. In Figure 9, 5 representative images of these categories are shown. Below these images, the rank-50 approximation is shown. One thing worth noting is that the rank-50 approximation is an approximation of the overall data, not just the data confined to the 5 facial-expression categories.

The rank 50 approximation does a mediocre job of representing the data. It's clear from Figure 9 that the surprised and wink approximations are the only well-approximated images (for this specific individual). The happy approximation is the next best, but even that one has major flaws when viewed in contrast with the original image. Given that there are 165 data points, it's reasonable for a rank-50 approximation to produce subpar results.

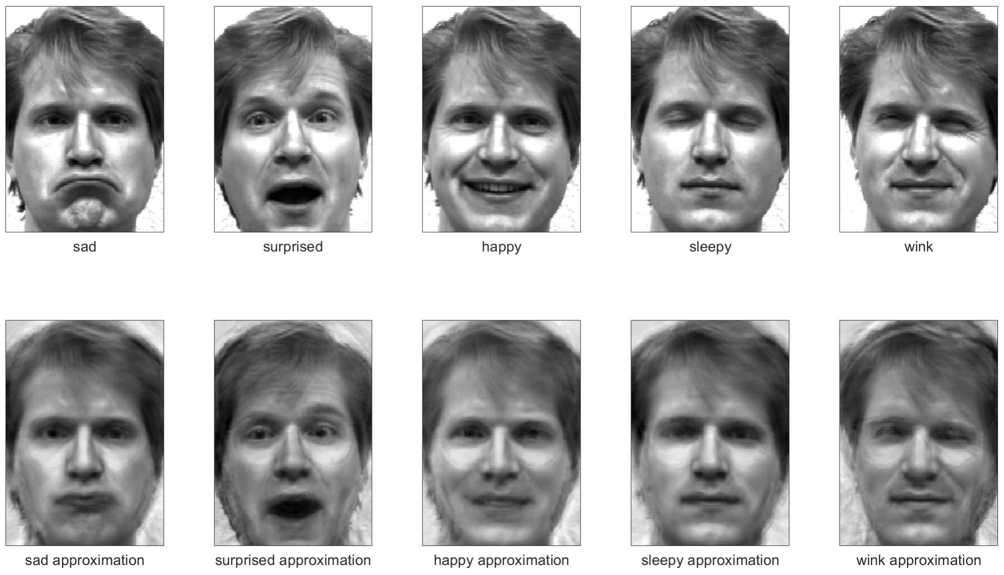

Figure 9: Original images vs. Those Produced by a Rank-50 Approximation

We also computed the full SVD data and analyzed the singular values. It's clear from Figure 10, where the the logarithm of the singular values is plotted, that the singular values decrease incredibly slowly.

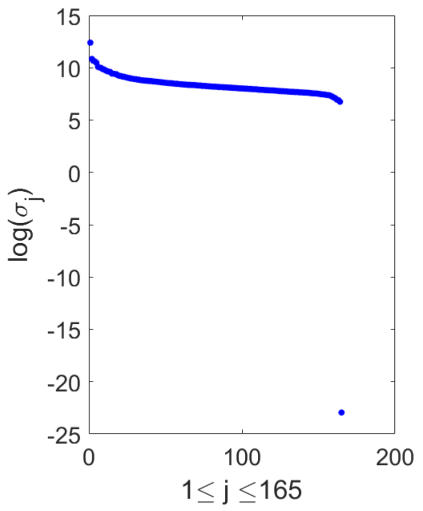

Figure 10: Raw Data Singular Values

This behavior of the singular values explains why even a rank-50 approximation has glaring flaws. The slow decrease of the singular values indicates that a high rank is required to accurately represent the data. It then makes sense that a rank-50 approximation does a mediocre job of representing data which has roughly 160 significant singular values.

Furthermore, we can reason why the singular values decrease so slowly. Not only are there 11 different categories, but there are also 15 separate individuals. These subjects have different hairstyles, facial features, and often fundamentally different takes on any given expression. This leads to an incredibly diverse range of visual data.

It makes sense that such data cannot be described as combinations of a small amount of visual layers - feature vectors. Consequently, the data cannot have only a few significant singular values.

In Figure 11 on the next page, we see the first 10 feature vectors of the rank-50 approximation of the data. Note that feature vectors are mutually orthogonal by definition. For this to be possible, the feature vectors need to allow for negative components, despite the data itself not doing so. This explains why the feature vectors look ”negative” or ”inverted” in some cases.

The first feature vector seems like the most generic representation of a face. We were unable to identify any clear facial expression in any of the feature vectors. That being said, the lighting clearly varies between the feature vectors and glasses are clearly visible in numbers 5, 7, 9, and 10. Also, the feature vectors pretty clearly include different individuals.

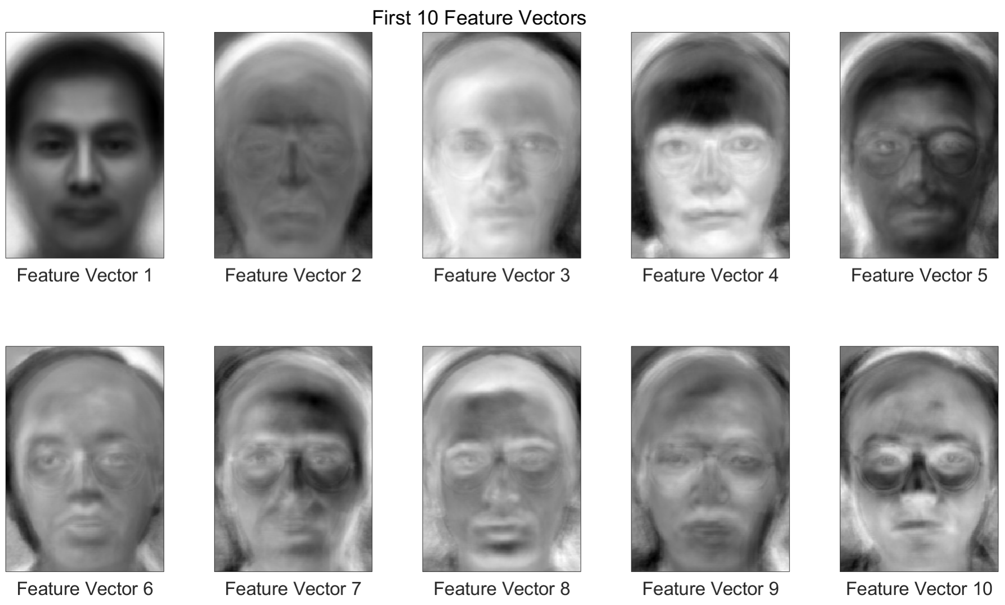

Figure 11: First 10 Feature Vectors of Approximated Data

Furthermore, we can use the first k feature vectors of a rank-r approximation of the data to approximate the images. This essentially means projecting the image onto the space spanned by the first k feature vectors. In MATLAB, this can be done with

```
Uk = U(:, 1:k);
Xk = U(:, 1:k) * (Uk' * Xr);
```

In Figure 12, Figure 13, and Figure 14 below, the approximations with $k = 4, 8, 15$ are shown in comparison with the original data.

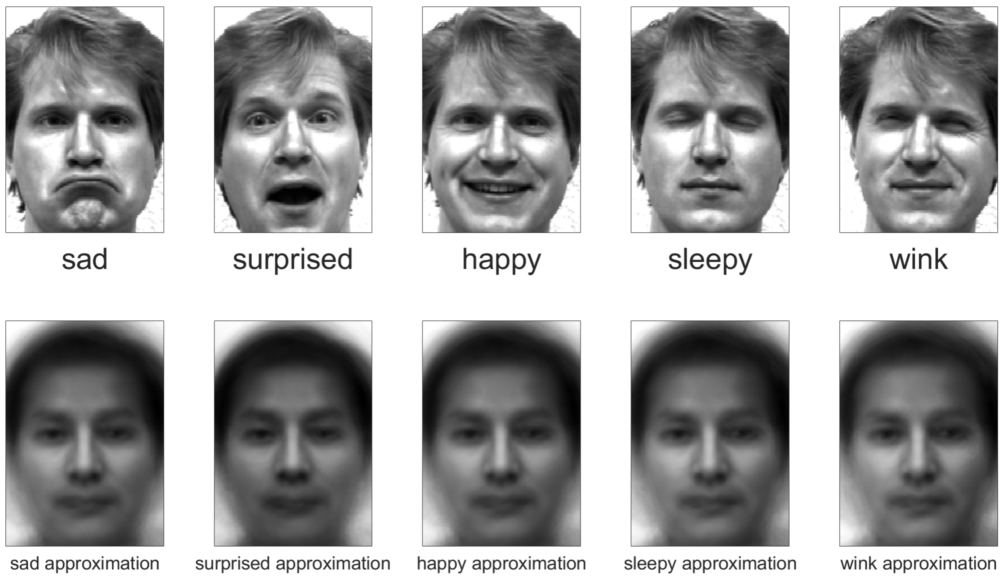

Figure 12: Approximations with $k = 4$

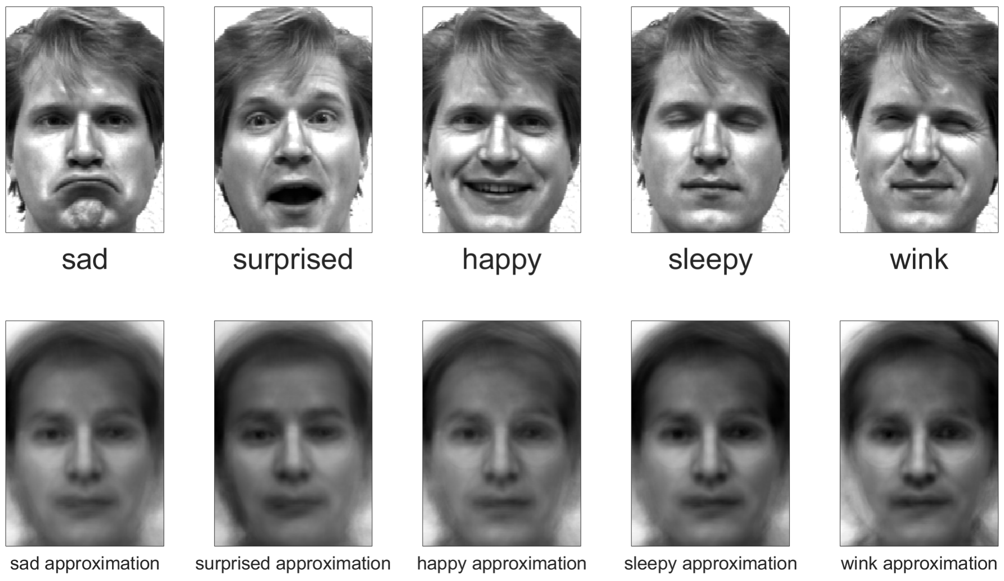

Figure 13: Approximations with $k = 8$

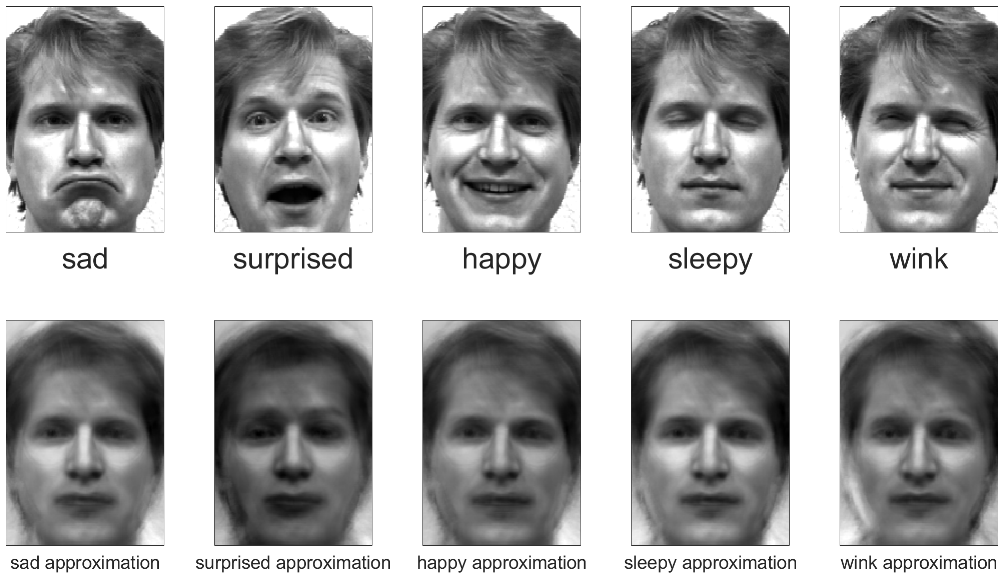

Figure 14: Approximations with $k = 15$

Immediately, it's apparent that these approximations aren't great in terms of favial expression. However, a noticeable improvement that occurs as k increases is that the individual becomes more identifiable. This makes sense since the first 10 feature vectors include a variety of lighting conditions and individuals.

Overall, this data isn't approximated well by a low-rank approximation. This stems from the fact that the variety of data is simply too large. If one were to confine the data to one individual and only a few categories, we are confident that a low rank approximation would do a *better* job.

A GIF showing how the approximation improves with rank can be downloaded.

Appendix

1 Biopsy Data Revisited

It was stated that the unshown principal components do not offer any extra insight on the clustering of the data. Indeed, this is the case and we believe that involving more principal components only muddies the water for this specific data set. Figure 15 below clearly shows this effect.

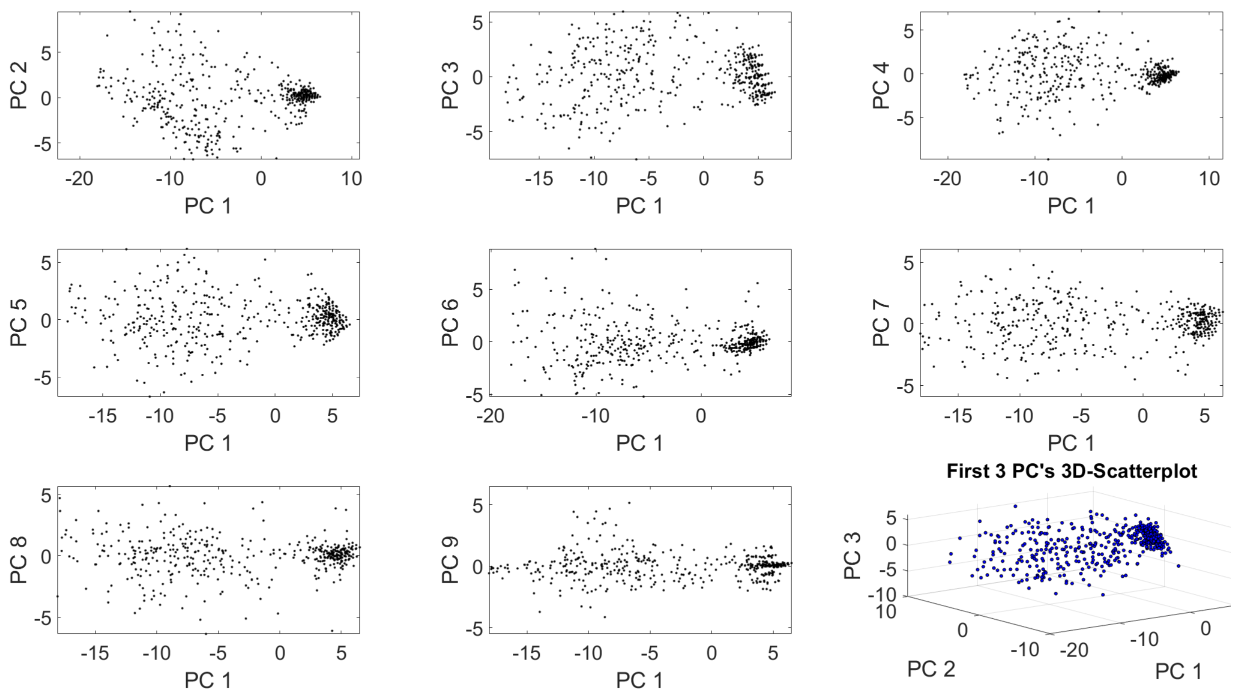

Figure 15: Some Biopsy Data Pairwise Principal Component Comparisons

2 Model Reduction Code

```
    %Levent Batakci - LAB192@case.edu
%MATH444 HW#1
%Model Reduction Data portion

%Clear memory
clc
clear all

%Load the model reduction data
%The data is stored in 6x4000 a matrix X
%The entries are all of type 'double'
load ModelReductionData
```

```
%PART a
%In this part, the data will be visualized by plotting the components
%of the data (one pair at a time) against each other
%Since there are 6 components, there will be (6 choose 2)=15 plots
colors = [[0, 0.4470, 0.7410]; [0, 0.5, 0]; [1, 0, 0]; [0.8500, 0.3250, 0.0980]; [0, 0.75, 0.75]; [0.

c = size(X,1); %Number of components
sbplt_index=1;
figure(1);
for i = 1:c
   for j= i+1:c
       subplot(3, 5, sbplt_index)
       sbplt_index = sbplt_index + 1;
       plot(X(i,:),X(j,:),'k.','MarkerSize',3)
       axis('equal')
       set(gca,'FontSize',23)

       xlabel(append("Component ", string(i)), 'Color', colors(i,:))
       ylabel(append("Component ", string(j)), 'Color', 'k')
   end
end
sgtitle("Figure 1: Raw Data Components Compared", 'FontSize', 30) %MAKE THIS PLOT PRETTIER !!!!


%PART b
%In this part, I calculate the SVD of the data (after centering it)
%Then, the singular values are plotted to determine the effective
%dimensionality of the data

%Center the data
N = size(X,2);
xc = sum(X, 2)/N; %Get the average
Xc = X -  xc*ones(1,N); %Subtract out the average

[U,S,V] = svd(Xc, 'eco'); %Compute the SVD

%Plot the singular values
singular_values = diag(S); %Extract the singular values
figure(2);
subplot(1, 1, 1); %Make the plot into 1 large one
plot(1:size(Xc,1), singular_values, 'k.','MarkerSize', 45, 'Color', 'b')
set(gca,'FontSize',25)

%Set up the x-axis
xlbl = append("\fontsize{25}1", "\leq j \leq", string(size(Xc,1)));
xlabel(xlbl, 'interpreter','tex');
xticks(1:size(Xc,1));

%Set up the y-axis
ylbl = "\fontsize{25}Singular Value   \sigma_j";%Label y
ylabel(ylbl, 'interpreter','tex');
sgtitle("Figure 2: Centered Data Singular Values", 'FontSize', 20);
```

```
%THE RESULTING PLOT MAKES IT CLEAR THAT THE DATA HAS EFFECTIVELY
% % % % % 3 DIMENSIONS % % % % %
%From here, it remains to check how many distinct data clumps there are.
%To do this, we will project the data onto u1, u2, and u3 and plot the
%data in 3-space
U3 = U(:, 1:3)'; %Get the first 3 feature vectors
Z3 = U3 * Xc; %Project to get the 3 principal components

%Look at the principal components in pairs
figure(3);
c = size(Z3,1); %Number of components
sbplt_index=1;
for i = 1:c
   for j= i+1:c
       subplot(2, 2, sbplt_index)
       sbplt_index = sbplt_index + 1;
       plot(Z3(i,:),Z3(j,:),'k.','MarkerSize',3)
       axis('equal')
       set(gca,'FontSize',22)

       xlabel(append("PC ", string(i)))
       ylabel(append("PC ", string(j)))
   end
end

%Look at the 3D Scatterplot
s = 15; %Marker size
subplot(2, 2, 4)
scatter3(Z3(1,:),Z3(2,:),Z3(3,:), s, 'o', 'MarkerEdgeColor','k', 'MarkerFaceColor','b')
xlabel("PC 1")
ylabel("PC 2")
zlabel("PC 3")
set(gca,'FontSize',22)
title("Principal Components 3D-Scatterplot", 'FontSize', 20)

sgtitle("Figure 3: Centered Data Principal Components (PC) Compared");

%Looking at the comparisons, there appear to be
% % % % % 2 CLUSTERS % % % % %
```

3 Biopsy Code

```
    %Levent Batakci - LAB192@case.edu
%MATH444 HW#1
%Biopsy Data portion

%Clear memory
clc
clear all
```

```
%Load the biopsy data
%The data is stored in 9x699 a matrix X
%The entries are all of type 'double' and are between 1 & 10
%If an entry is missing, it will be NaN
load BiopsyData

%Remove columns with missing data
X(:, any(isnan(X))) = [];

n = size(X,1); %Number of attributes
p = size(X,2); %Number of data points

%Center the data and compute the SVD
xc = sum(X,2) / p;
Xc = X - xc * ones(1, p);
[U,S,V] = svd(Xc, 'eco');

%Plot the singular values
singular_values = diag(S); %Extract the singular values
figure(1);
plot(1:n, singular_values, 'k.','MarkerSize', 30, 'Color', 'b')

%Set up the x-axis
xlbl = append("\fontsize{25}1", "\leq j \leq", string(size(Xc,1)));
xlabel(xlbl, 'interpreter','tex');
xticks(1:size(Xc,1));

%Set up the y-axis
ylbl = "\fontsize{25}Singular Value   \sigma_j";%Label y
ylabel(ylbl, 'interpreter','tex');

set(gca,'FontSize', 25)
%%%%

%Compute the principal components
Z = U' * Xc;

%Look at the principal components in pairs
figure(2);
c = size(Z,1); %Number of components
sbplt_index=1;
for i = 1:1
   for j= i+1:n
       subplot(3, 3, sbplt_index)
       sbplt_index = sbplt_index + 1;
       plot(Z(i,:),Z(j,:),'k.','MarkerSize',8)
       axis('equal')
       set(gca,'FontSize',25)

       xlabel(append("PC ", string(i)))
       ylabel(append("PC ", string(j)))
   end
```

```
end

%Look at the 3D Scatterplot
s = 15; %Marker size
subplot(3, 3, 9)
scatter3(Z(1,:), Z(2,:), Z(3,:), s, 'o', 'MarkerEdgeColor','k', 'MarkerFaceColor','b')
xlabel("PC 1")
ylabel("PC 2")
zlabel("PC 3")
set(gca, 'FontSize', 25)
title("First 3 PC's 3D-Scatterplot", 'FontSize', 25)

sgtitle("Figure 2: Centered Data Principal Components (PC) Compared");

%PLOT JUST FIRST PC
figure(3)
scatter(Z(1,:), zeros(size(Z(1,:))),500,'filled')
ylim(gca, [-0.02 0.02])
xlabel("PC 1")
alpha(0.2)
set(gca, 'FontSize', 30)
```

4 Iris Code

```
    %Levent Batakci - LAB192@case.edu
%MATH444 HW#1
%Iris Data portion

%Clear memory
clc
clear all

%Load the iris data
%The data is stored in 4x450 a matrix X
%The entries are lengths measured in centimeters
%In order, the entries are 'sepal length', 'sepal width', 'petal length',
%and 'petal width'
load IrisData

%Center the data and compute the SVD
n = size(X,1);
p = size(X,2);
xc = sum(X,2) / p;
Xc = X - xc * ones(1, p);
[U,S,V] = svd(Xc, 'eco');

%Plot the singular values
singular_values = diag(S);
plot(1:n, singular_values);
figure(1);
plot(1:n, singular_values, 'k.','MarkerSize', 50, 'Color', 'b')
```

```
%Set up the x-axis
xlbl = append("\fontsize{25}1", "\leq j \leq", string(size(Xc,1)));
xlabel(xlbl, 'interpreter','tex');
xticks(1:size(Xc,1));

%Set up the y-axis
ylbl = "\fontsize{25}Singular Value   \sigma_j";%Label y
ylabel(ylbl, 'interpreter','tex');
set(gca, 'FontSize', 30)

%Compute all of the principal components
Z = U' * X;

%Look at the principal components in pairs
colors = [[0, 0.4470, 0.7410]; [0, 0.5, 0]; [1, 0, 0]; [0.8500, 0.3250, 0.0980]; [0, 0.75, 0.75]; [0.
figure(2);
c = size(Z,1); %Number of components
sbplt_index=1;
for i = 1:n-1
   for j= i+1:n
       subplot(2, 3, sbplt_index)
       sbplt_index = sbplt_index + 1;
       plot(Z(i,:),Z(j,:),'k.','MarkerSize',10)
       axis('equal')
       set(gca,'FontSize',30)

       xlabel(append("PC ", string(i)),'Color', colors(i,:))
       ylabel(append("PC ", string(j)))
   end
end
%sgtitle("Figure 2: Centered Data Principal Components (PC) Compared");

%Look at a 3D scatter plot of the first 3 PCs
figure(3);
s=30;
scatter3(Z(1,:), Z(2,:), Z(3,:), s, 'o', 'MarkerEdgeColor','k', 'MarkerFaceColor','b')
xlabel("PC 1")
ylabel("PC 2")
zlabel("PC 3")
sgtitle("Figure 3: First 3 Principal Components 3D-Scatterplot")



%PLOT JUST FIRST PC
figure(4)
scatter(Z(1,:), zeros(size(Z(1,:))),500,'filled')
ylim(gca, [-0.02 0.02])
xlabel("PC 1")
alpha(0.2)
set(gca, 'FontSize', 30)

%Honestly, it looks like there are 2 obvious groupings.
%There might be 3 clusters, but I think that two of the flower types
```

```
% are definitely harder to tell apart.
```

5 Yale Code

```
    %Levent Batakci - LAB192@case.edu
%MATH444 HW#1
%Aligned Faces portion

%Clear memory
clc
clear all

%Load the face data
%The data is stored in a 28341x165 matrix X
%The entries are integers ranging from
%0 to 255 (256 possible values)
%Each data vector is an encoded image with size 201x141=28341 pixels
%Each image falls into 1 of 11 categories
%The row-vector I represents the categorization of the data
% 1='No glasses',2='Glasses',3='Sad',4='Surprised',
% 5='Happy', 6='Sleepy',7='Wink',8='Centerlight',
% 9='Leftlight',10='Rightlight',11='Normal'
load AlignedYaleFaces

%Get an easy way to access the images belonging to a given category
noGlasses = find(I==1);
glasses = find(I==2);
sad = find(I==3);
surprised = find(I==4);
happy = find(I==5);
sleepy = find(I==6);
wink = find(I==7);
centerlight = find(I==8);
leftlight = find(I==9);
rightlight = find(I==10);
normal = find(I==11);

%UNUSED, REMOVES OTHER CATEGORIES
%X(:,[noGlasses glasses centerlight leftlight rightlight normal]) = [];

%Plot SVs
N=size(X,2)
[Ux,Sx,Vx] = svd(X, 'eco'); %Compute the SVD
figure(10)
plot(1:165, log(diag(Sx)), 'k.','MarkerSize', 20, 'Color', 'b')
%Set up the x-axis
xlbl = append("\fontsize{25}1", "\leq j \leq", string(size(X,2)));
xlabel(xlbl, 'interpreter','tex');

%Set up the y-axis
ylbl = "\fontsize{25}log(\sigma_j)";%Label y
ylabel(ylbl, 'interpreter','tex');
```

```
set(gca,'FontSize',22)
keyboard()

%Obtain an approximation for the images
r = 50;
[U,D,V] = svds(X, r);
Xr = U * D * V';



samples = [sad(1) surprised(1) happy(1) sleepy(1) wink(1)];
label = ["sad" "surprised" "happy" "sleepy" "wink"];

%USED FOR ANIMATION CREATION, VERY VERY VERY VERY VERY VERY TIME CONSUMING
% for k = 1: 165
%     figure(1);
%     colormap(gray);
%     [U,D,V] = svds(X, k);
%     Xr = U * D * V';
%     for i = 1:5
%         subplot(2,5,i);
%         imagesc(reshape(X(:,samples(1,i)), ImageSize));
%         xlabel(label(i));
%         xticks([])
%         yticks([])
%         set(gca,'FontSize', 20)
%
%         subplot(2,5,i+5);
%         imagesc(reshape(Xr(:,samples(1,i)), ImageSize));
%         xlabel(label(i) + " approximation");
%         xticks([])
%         yticks([])
%         set(gca,'FontSize', 20)
%     end
%     sgtitle("Original Images vs. those Achieved by an Approximation of the Data", 'FontSize', 30)
%     saveas(gcf, "animation\frame " + string(k) + ".png");
% end

% Write to the GIF File
%WILL BE AVAILABLE THROUGH LINK ON README
% for k = 1:165
%     filename = "animation\frame " + string(k) + ".png";
%     im = imread(filename);
%     im = rgb2gray(im);
%     disp(k)
%     if k == 1
%         imwrite(im,"animation\FacesGif.gif",'gif');
%     else
%         imwrite(im,"animation\FacesGif.gif",'gif','WriteMode','append');
%     end
% end

%Compare originals w/ approximations
```

```
figure(1)
colormap(gray)
for i = 1:5
    subplot(2,5,i);
    imagesc(reshape(X(:,samples(1,i)), ImageSize));
    xlabel(label(i));
    xticks([])
    yticks([])
    set(gca,'FontSize', 40)

    subplot(2,5,i+5);
    imagesc(reshape(Xr(:,samples(1,i)), ImageSize));
    xlabel(label(i) + " approximation");
    xticks([])
    yticks([])
    set(gca,'FontSize', 25)
end
sgtitle("Original Images vs. those Achieved by a rank " + string(r) + " Approximation of the Data", '

%Plot the SVs
figure(2)
singular_values = diag(D);
plot(1:r, log(singular_values), 'k.','MarkerSize', 20, 'Color', 'b')
%MAKE THIS PART BETTER

%10 feature vectors
figure(3)
colormap(gray)
for i = 1:5
    subplot(2,5,i);
    imagesc(reshape(U(:,i), ImageSize));
    xlabel("Feature Vector " + string(i));
    xticks([])
    yticks([])
    set(gca,'FontSize', 25)

    subplot(2,5,i+5);
    imagesc(reshape(U(:,i+5), ImageSize));
    xlabel("Feature Vector " + string(i+5));
    xticks([])
    yticks([])
    set(gca,'FontSize', 25)
end
sgtitle("First 10 Feature Vectors", 'FontSize', 30)

K = [4 8 15];
for j = 1:3
    %Compare originals w/ approximations
    figure(3+j)
    colormap(gray)
    k= K(j);
    Uk = U(:, 1:k);
    Xk = U(:, 1:k) * (Uk' * Xr); %Approximation by first k feature vectors
    for i = 1:5
        subplot(2,5,i);
        imagesc(reshape(X(:,samples(1,i)), ImageSize));
        xlabel(label(i));
        xticks([])
        yticks([])
        set(gca,'FontSize', 40)

        subplot(2,5,i+5);
        imagesc(reshape(Xk(:,samples(1,i)), ImageSize));
        xlabel(label(i) + " approximation");
        xticks([])
        yticks([])
        set(gca,'FontSize', 25)
    end
    sgtitle("Original Images vs. those Approximated by the First " + string(k) + " Feature Vectors",
end
```