AMATH 582 Homework 1[1]

Jesse Dumas

September 14, 2018

[1] Let the hand waving begin!

This is a discussion of my journey through singular value decomposition (SVD) analysis for AMATH 582 homework 1[2]. SVD facilitates taking large data sets with high dimensionality and reducing the dimensionality by collapsing the data onto its principle components (directions or basis vectors). The work explores an SVD analysis of the Extended Yale Face Database B[3] and finds the number of modes from the singular value spectrum required for a decent image reconstruction.

[2] This is written for me, so I'm going to write in the first person here and there, rather than clunky academic prose.

[3] A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 23(6):643–660, 2001

Introduction and Overview

Singular value decomposition[4] is a useful method for reducing high dimensional data to low rank data by finding the most important dimensions for use as basis vectors, and image recognition is an archetypal problem for its application. An excellent data set for this purpose is the extended Yale Face Database B, comprised of 2414 grayscale images of 10 human faces under different lighting conditions [5]. As the human face contains multiple distinguishing features (eyes, mouth, nose, and their configuration), the data set has high dimensionality. The purpose of this exercises is to distill the data down to their most important constituents, and that is exactly what singular value decomposition accomplishes.

[4] J.N. Kutz. *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*. OUP Oxford, 2013. ISBN 9780199660339

[5] K.C. Lee, J. Ho, and D. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 27(5):684–698, 2005

This exploration will investigate the data set using SVD, showing that images can be approximated using bases made of subsets of the singular values. However, this paper will stop short of actually developing and implementing a facial recognition algorithm, as it is beyond the scope of the project.

Theoretical Background

One of the most useful steps to take when analyzing large data sets is to check for the possibility that the system behavior described by the data can be explained and reconstructed by a reduced subset of the data. SVD is just the tool for such a task. If $\mathbf{A}$ is our data matrix, SVD will decompose it as

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^* \tag{1}$$

Figure 1: Nine random faces from the extended Yale Face Database B. *Not too creepy.*

which can also be expressed as

$$[\mathbf{A}] = \begin{bmatrix} & & & \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_n \\ & & & \end{bmatrix} \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_n \end{bmatrix}. \tag{2}$$

In equations 1 and 2, Σ is a diagonal matrix of eigenvectors comprising the singular values of $\mathbf{A}$. The columns in $\mathbf{U}$ are orthonormal basis vectors for the "facespace," and $\mathbf{V}$ describes how each face is projected onto the principal components in $\mathbf{U}$.

Algorithm Implementation and Development

The implementation of SVD on the data set was trivial using MATLAB. With just one line of code,

```
[U,S,V] = svd(A);
```

the entire decomposition can be completed. The bulk of work implementing the algorithm was in preparing the images in such a way that the data contained could be manipulated as matrices. The full code for the project is presented in Appendix B.

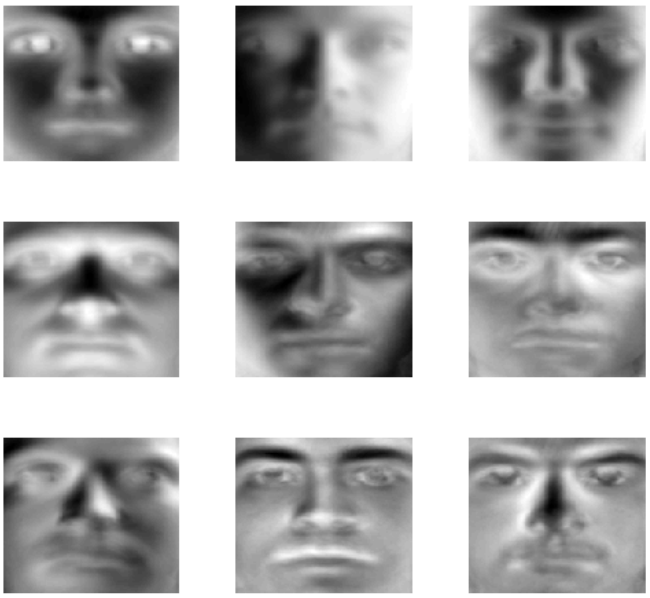

Figure 2: The first nine eigenfaces. *Pretty creepy.*

Computational Results

Eigenfaces

Eigenfaces are the set of eigenvectors derived from the data set that can be used to reconstruct the images. Essentially, they reveal which features of the data most strongly correlate between the original faces. Figure 2 shows the first nine eigenfaces. The first face is essentially the "average" face–there's nothing too special about it: just eyes, a nose, and a mouth, with average shading. The remaining eigenfaces in the figure highlight the next eight most prominent features in the data.

To find the rank of the facespace, let's plot the singular values in Figure 3. In the top plot, we can see that the first several singular values dominate, but a look at the bottom plot (on a semi-log scale) shows that it is more likely that maybe the first 100 modes might be necessary for a good image reconstruction.

To test that, let's look at Figure 4. The top row is the original image of our friend, Bob, and the bottom row shows a rank 9, 50, and 100 approximation, respectively, of Bob. The rank 9 approximation captures the same shading as the original image, but that's about it. Bob's facial features are mostly lost. The rank 50 reconstruction does

a better job at capturing Bob's facial structures, but some of the shading detail is not right on the left of the image. It also looks a little like a police sketch. The rank 100 approximation looks very close to the original, however. The reconstruction includes most of Bob's defining features, and the shading is also close to the original. Figure [6] shows the modes for the singular value spectrum, but it isn't very helpful.

[6]

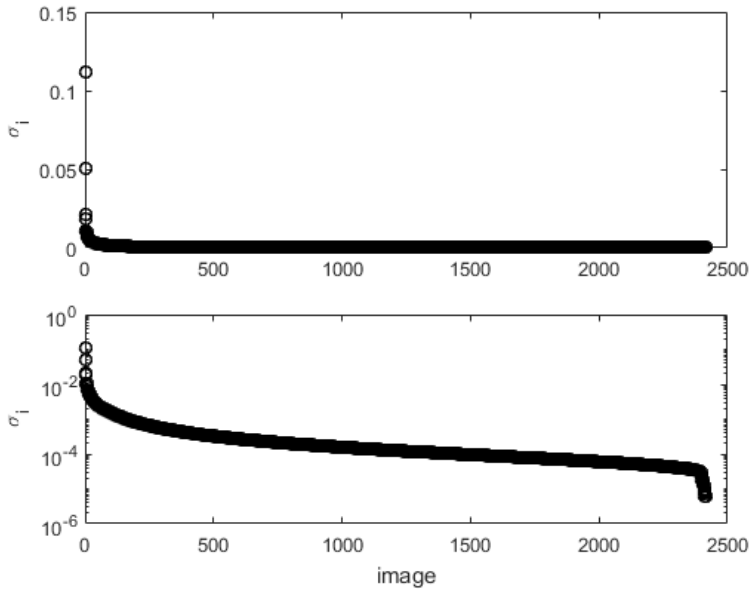

Figure 3: The singular values σ_i from the data set. The top plot is a standard scale, and the bottom plot is a semi-log plot.

Summary and Conclusions

Clearly SVD is an effective technique for reducing data with high dimensionality into low rank data that can both capture important features and save computational time. Performing an SVD analysis of the extended Yale Face Database B was an enlightening exercise, allowing me to gain an understanding of the technique. In the future, it would be interesting to take the low rank approximation of the data set and use it with a clustering algorithm to identify individual faces in the database.

References

A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman. From few to many: Illumination cone models for face recognition under variable

Figure 4: Top row: The original image of our friend, Bob. Bottom row: Approximate Bobs using rank 9, 10, and 100 reconstructions, respectively.

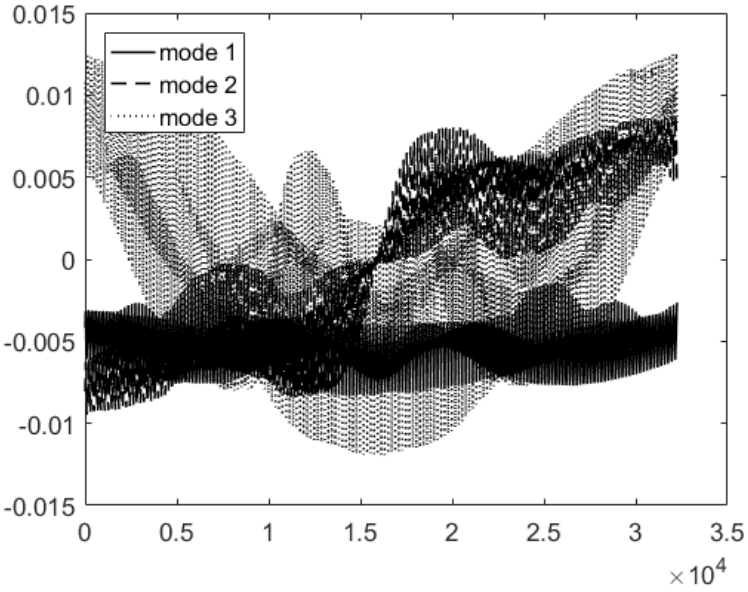

Figure 5: The modes from the singular value spectrum. *Not very helpful here. I probably did something wrong.*

lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 23(6): 643–660, 2001.

J.N. Kutz. *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*. OUP Oxford, 2013. ISBN 9780199660339.

K.C. Lee, J. Ho, and D. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 27(5):684–698, 2005.

Appendix A: MATLAB Functions

```
[U,S,V] = svd(A); -- performs SVD.
reshape() -- rehapes columns back to image dimensions.
imread() -- reads images into matrices.
```

Appendix B: MATLAB Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Homework 1 for AMATH 582
% Jesse Dumas
% SVD Party Time
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Where's ma data?
data_folder = 'C:\Users\jes\Documents\MATLAB\582hw1\just_pgm';
file_extension = fullfile(data_folder, '*.pgm');
pgm_files = dir(file_extension);
image_matrix = [];

% Grab images and add them to image matrix, one per column.
for ii = 1:length(pgm_files)
    baseName = pgm_files(ii).name;
    fullName = fullfile(data_folder, baseName);
    fprintf(1, 'Now reading %s\n', fullName);
    image_matrix2 = imread(fullName);
    image_matrix_reshaped = reshape(image_matrix2, [], 1);

    image_matrix(:,ii) = double(image_matrix_reshaped);
end

% Do SVD
[U, S, V] = svd(image_matrix, 'econ');
```

```
% for image reconstruction
S_rank9 = S;
S_rank9(10:end,10:end) = 0;
A = reshape(image_matrix(:,314),192, 168);
A_9 = U*S_rank9*V';

S_rank50 = S;
S_rank50(51:end,51:end) = 0;
A_50 = U*S_rank100*V';

S_rank100 = S;
S_rank100(101:end,101:end) = 0;
A_100 = U*S_rank100*V';

% Singular values plot from Kutz book
sig = diag(S)/sum(diag(S));

subplot(2,1,1), plot(sig,'ko','Linewidth',[1.1])
subplot(2,1,2), semilogy(sig,'ko','Linewidth',[1.1])

% Mode plot from Kutz book
x=linspace(1,32256,32256);

plot(x,U(:,1),'k',x,U(:,2),'k--',x,U(:,3),'k:','Linewidth',[1.1])
set(gca,'Fontsize',[13])
legend('mode 1','mode 2','mode 3','Location','NorthWest')
text(0.8,0.35,'(c)','Fontsize',[13])

% Eigenfaces plot a la 582 lecture 1/20/2017
for j=1:9
    ef = reshape(U(:,j),192,168);
    subplot(3,3,j)
    imagesc(ef), colormap(gray), axis off
end

% 9 random faces plot
q = randperm(2432);

for j=1:9
    subplot(3,3,j)
    rand_face=reshape(image_matrix(:,q(j)),192,168);
    imagesc(rand_face), colormap(gray), axis off
end
```

```
% Bob plots
subplot(2,1,1), imagesc(A), colormap(gray), axis off
subplot(2,1,2), imagesc(reshape(A_9(:,314),192,168)), colormap(gray), axis off

subplot(2,1,1), imagesc(A), colormap(gray), axis off
subplot(2,1,2), imagesc(reshape(A_50(:,314),192,168)), colormap(gray), axis off

subplot(2,1,1), imagesc(A), colormap(gray), axis off
subplot(2,1,2), imagesc(reshape(A_100(:,314),192,168)), colormap(gray), axis off
```