Table of Contents

Introduction

```
%Written by Ian Good on Dec 7th, 2020. For questions, you can reach me
 at
%    iangood@uw.edu- Matlab 2020b
%{
The goal of this file is to explore data reduction methods using
 randomized
    algorithms. We analyze the Yale Faces Dataset through singular
 value
    decomposition (SVD). Power iteration methods are used to compute
 the
    largest eigenvalue and its corresponding eigenvector, and the
 faces
    are randomly sampled to determine the most optimal reconstruction.
    Knowledge was applied from previous analysis on the Yale Face
 Dataset
    which can be found here:
    https://github.com/liquidmercury8/PCA-and-SVD-Exploration


    Data from the The Yale Faces Project was used. A link to the raw
 data
    can be found below:
    https://drive.google.com/drive/
folders/1SQ77P5t5RUWCSucmk4jPFbufFMX8VrJG

%}
```

BEGIN CODE

```
%Clear Previous Workspace and Figures

clear all
close all
clc
```

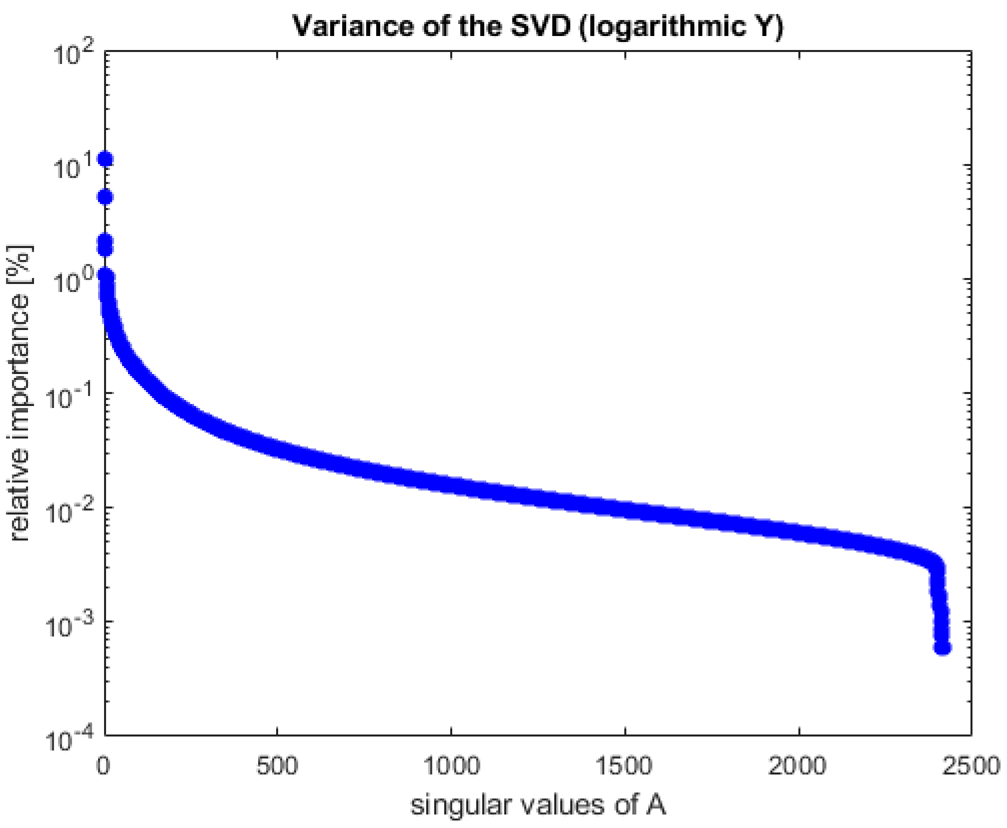
Load in the Cropped face database

```
tic
% % Specify the folder where the files live.
% myFolder = 'C:\Users\PhD\Documents\MATLAB\Power Iteration and
 Randomized Matrices\CroppedYale\';   %CroppedYale\yaleB31   or
 yalefaces\
% % Check to make sure that folder actually exists.  Warn user if it
 doesn't.
% if ~isfolder(myFolder)
%     errorMessage = sprintf('Error: The following folder does not
 exist:\n%s\nPlease specify a new folder.', myFolder);
%     uiwait(wfaceSetdlg(errorMessage));
%     myFolder = uigetdir(); % Ask for a new one.
%     if myFolder == 0
%          % User clicked Cancel
%          return;
%     end
% end
% % Get a list of all files in the folder with the desired file name
 pattern.
% filePattern = fullfile(myFolder, '**/*.pgm'); %search for whatever
 you need. The **/ searches subfolders as well
% theFiles = dir(filePattern);
% N = length(theFiles);
%
% m = 192;
% n = 168;
% A = [];
% count = 0;
%
%
% for k = 1 : length(theFiles)
%     baseFileName = theFiles(k).name;
%     fullFileName = fullfile(theFiles(k).folder, baseFileName);
%     imageArray = imread(fullFileName);
%     %figure
%     %imshow(imageArray);  % Display image.
%     %drawnow; % Force display to update immediately.
%
%
%     if(size(imageArray,3)==1)
%         M=double(imageArray);
%     else
%         M=double(rgb2gray(imageArray));
%     end
%     %pause(0.01);
%     R = reshape(M,m*n,1);
%     A = [A,R];
%    count = count + 1
% end
% Size of each picture in the cropped dataset
```

```
load('faceMatrix.mat','A') %precomputed for your convinience
```

Computing the SVD

```
[U,S,V] = svd(A,0);
figure()
semilogy(100*diag(S)/sum(diag(S)),'b*','linewidth',[2]);
title('Variance of the SVD (logarithmic Y)')
hold on; xlabel('singular values of A'); ylabel('relative importance
 [%]');
%Note, we do not have a singular dominant mode, which matches what we
 saw
%in the previous homework that explored this dataset.
```



Power iteration method for finding the largest eigenvalue

```
ACor = A.'*A; %build a correlation matrix
vt = zeros(length(ACor),1);vt(1,1) = 1; %initial guess for
 eigenvectors
%(unit length)
error = 1;
tol = 1e-9;
iter = 1;
```

```
while(error > tol)
    w = ACor*vt(:,iter);
    vt(:,iter+1) = w/norm(w,2);
    evalue = abs(vt(:,iter+1)'*ACor*vt(:,iter+1));
    error = norm(ACor*vt(iter)-evalue*vt(iter),2);
    out(iter,:) = [ iter+1 error];
    iter = iter+1;

end
eVecDiff = norm(abs(vt(:,end))-abs(V(:,1)),2); disp(strcat('The
 difference between power iteration eigen vectors and ground truth
 is:',num2str(eVecDiff)));
eValDiff = norm(abs(evalue)-abs(S(1,1)^2),2); disp(strcat('The
 difference between power iteration eigen values and ground truth
 is:',num2str(eValDiff)));
%so we get an error that seems large but is actually many orders of
%magnitude smaller than the values we are working with. We can see a
%small difference in the eigenvectors leads to a rather large error in
%the eigenvalue.
```

```
The difference between power iteration eigen vectors and ground truth
 is:0.0011468
The difference between power iteration eigen values and ground truth
 is:542974.7556
```

Using randomized sampling to reduce the size of the SVD

Stage A

```
[M,N] = size(A);
K = 75; %number of random projections (usually rank+10)
Omega = randn(N,K); %sampling matrix with values chosen based on
 suggested number of eigenfaces to keep from the previous homework
Y = A*Omega;
[Q,R] = qr(Y,0);

% Stage B
B = Q'*A;
[uLowDim,sLowDim,vLowDim] = svd(B,'econ');
uApprox = Q*uLowDim;

ReconstructionError = norm(A-uApprox*sLowDim*vLowDim',2);
disp(strcat('The Reconstruction error for A
 is:',num2str(ReconstructionError)));
```

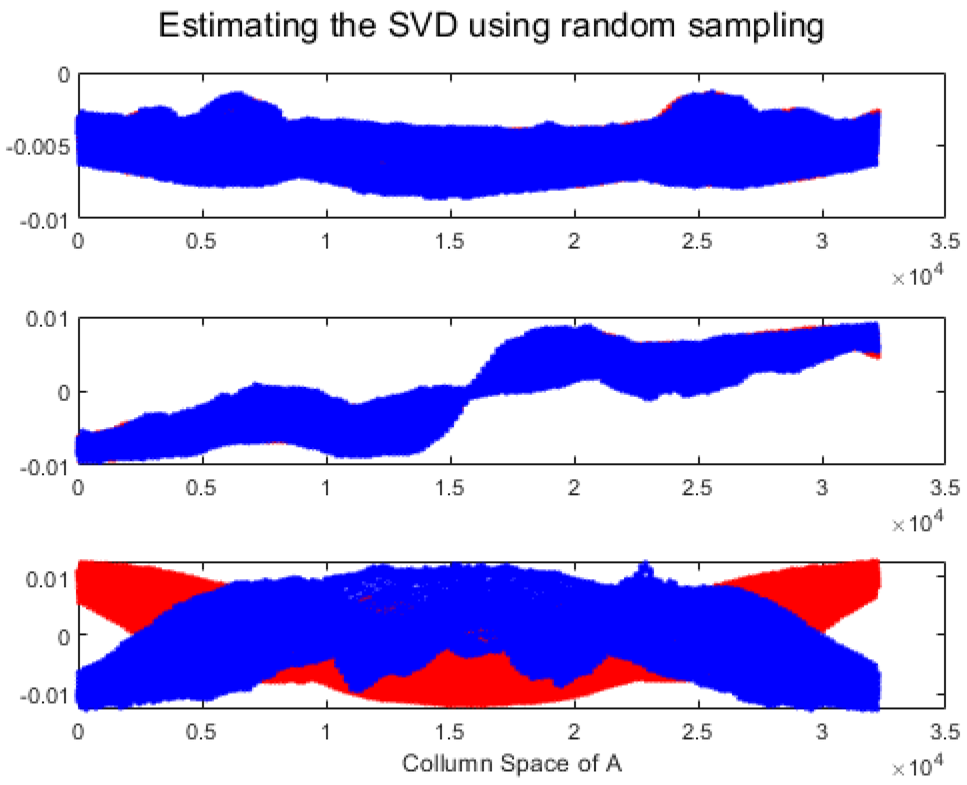
```
The Reconstruction error for A is:30953.8165
```

comparing Results from the Randomized Algorithms

```
figure()
```

```
x = 1:length(U(:,1));
subplot(3,1,1)
plot(x,U(:,1),'r.',x,uApprox(:,1),'b.','linewidth',[2]);
subplot(3,1,2)
plot(x,U(:,2),'r.',x,uApprox(:,2),'b.','linewidth',[2]);
subplot(3,1,3)
plot(x,U(:,3),'r.',x,uApprox(:,3),'b.','linewidth',[2]);
sgtitle('Estimating the SVD using random sampling')
hold on; xlabel('Collumn Space of A');

%{
    We can see from the plots of the first three vectors defining the
    collumn space of A, that K=75 does a very good job matching the
    full rank dataset. Remember the values are comparable up to a sign
    value. This represents a massive reduction in the compute time
 with
    a dimention of the matrix shrinking by two orders of magnitude!
%}
```



Exploring error as a function of K (number of random projections)

```
tolReconst = 1e-4;
iter=1;
errorModes = tolReconst+1;
```

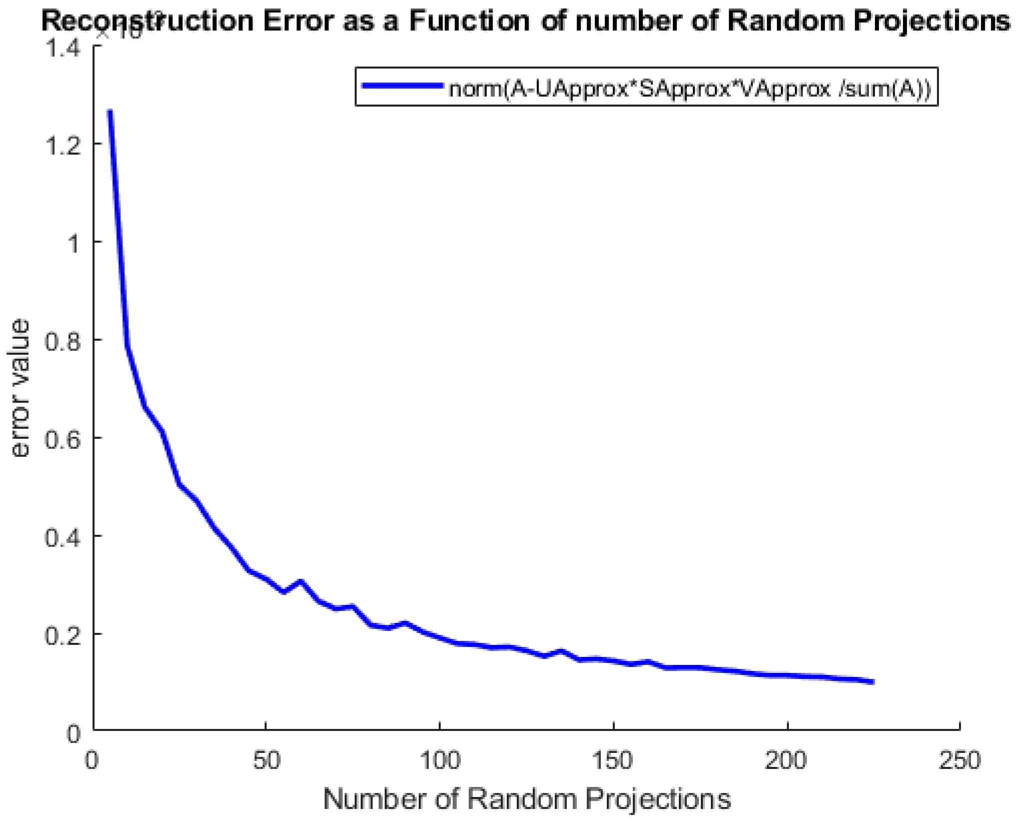
```
while errorModes > tolReconst
    Kfun(iter) = iter*5;
    %Part A
    OmegaFun = randn(N,Kfun(iter));
    Yfun = A*OmegaFun;
    [Qfun,Rfun] = qr(Yfun,0);
    % Stage B
    Bfun = Qfun'*A;
    [uLowDimfun,sLowDimfun,vLowDimfun] = svd(Bfun,'econ');
    uApproxfun = Qfun*uLowDimfun;
    errorModes(iter) = norm(A-uApproxfun*sLowDimfun*vLowDimfun',2)/
norm(sum(A),2);
    iter = iter+1;
end


%Plot the results
figure();
hold on; xlabel('Number of Random Projections'); ylabel('error
 value');
plot(Kfun,errorModes,'b-','linewidth',[2]);
legend('norm(A-UApprox*SApprox*VApprox /sum(A))')
title('Reconstruction Error as a Function of number of Random
 Projections');

%{
We can see that there is a steep drop off in efficiency in the random
sampling relative to the additional computation around the K=25 mark.

The normalization here was chosen as it provides a clear picture of
 the
number of random samples as a function of accuracy of reconstruction
 for
the given matrices. Additional increases in the precision of the solve
dramatically increased the compute time as not only did the matrices
 being
computed grow larger with each increase of K, but the relative value
 gain
became much less important.
For example when the Reconstruction Tolerance was set to 1e-3 (a
 single
order of magnitude increase) K ended at 25 and did not give a clear
picture.

Care was also taken to ensure the values covered in the homework on
 SVD and
PCA were also mapped by this reconstruction.
%}
```



Published with MATLAB® R2020b