Machine Learning
for Signal Processing
Homework 2

07.12.2020

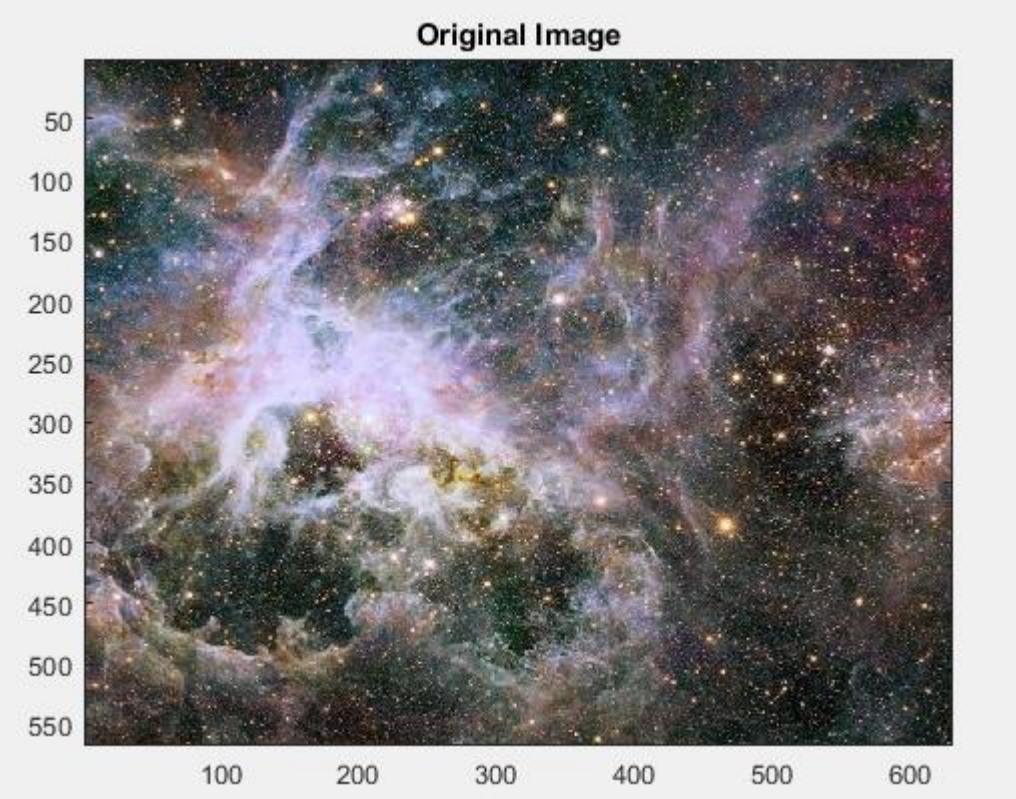
Mehmet Şerbetçioğlu -- 040160056

Veli Bulur -- 040150051

SVD Exercise 1:

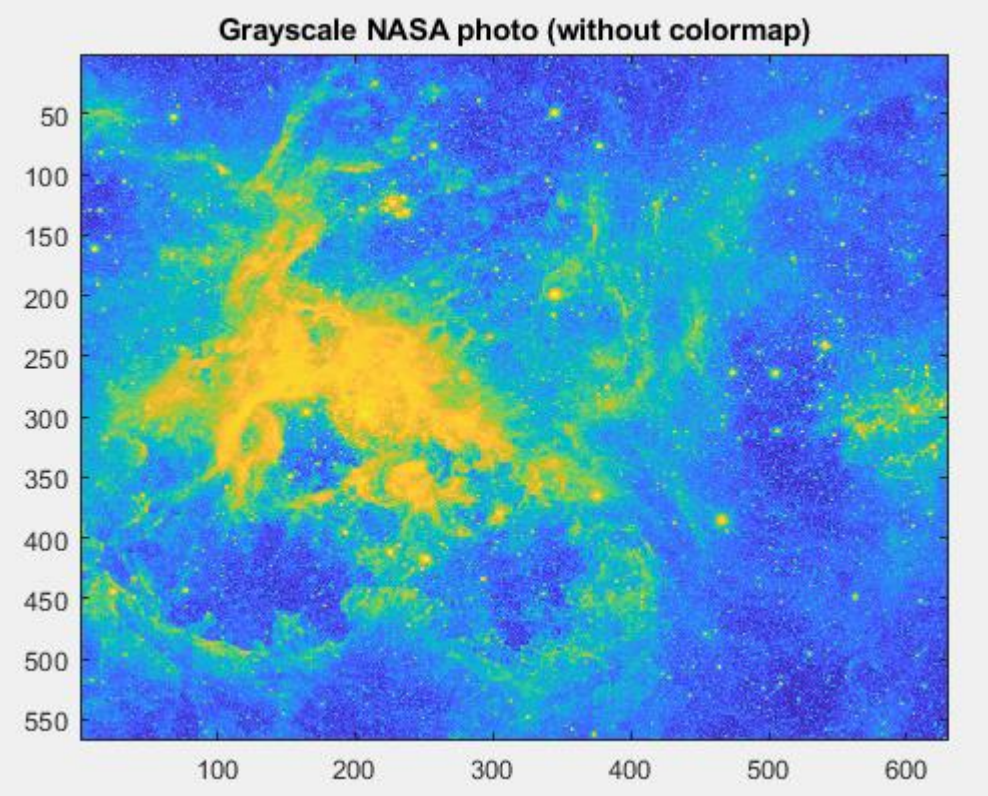
For this exercise, an image of Tarantula Nebula is used. This image is loaded and can be shown as such.

```
clear, clc, close all;

nasacolor = imread('TarantulaNebula.jpg');

figure
image(nasacolor)
title('Original Image')
```



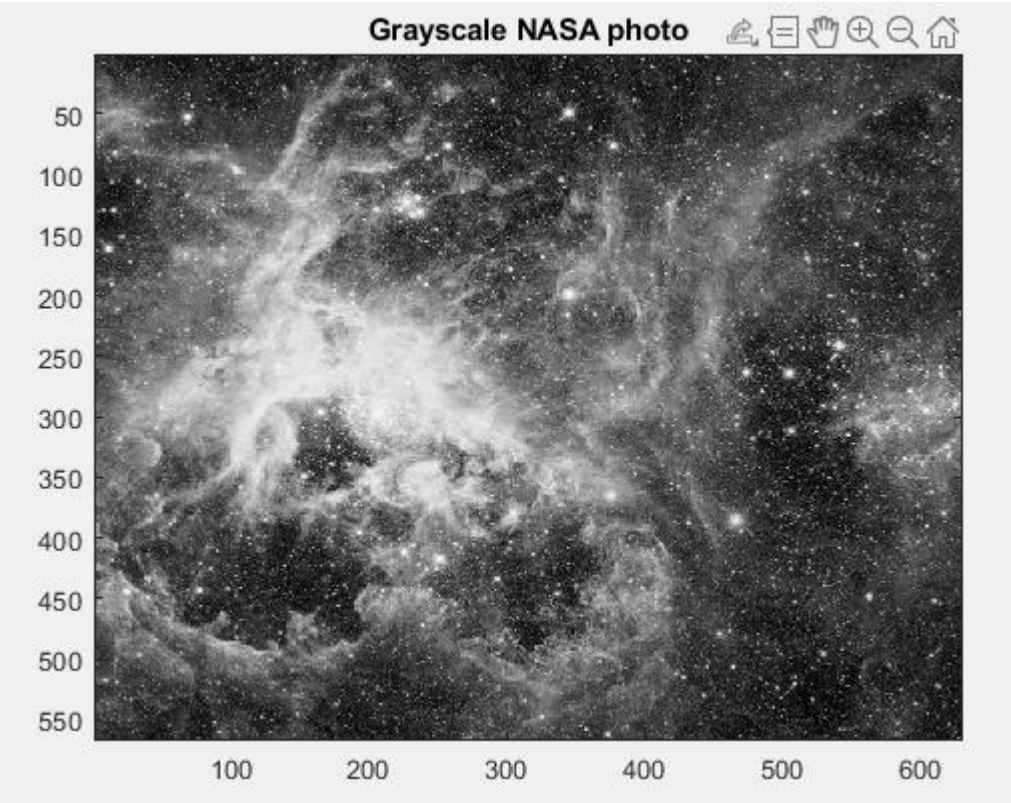
The original image contains three dimensions. Third dimension has three elements, which correspond to rgb scale of each pixel. For convenience, third dimension will be reduced to one element with grayscaling.

```
nasa = sum(nasacolor, 3, 'double');
m = max(max(nasa));
nasa = nasa*255/m;

figure
image(nasa)
title('Grayscale NASA photo (without colormap)')
```



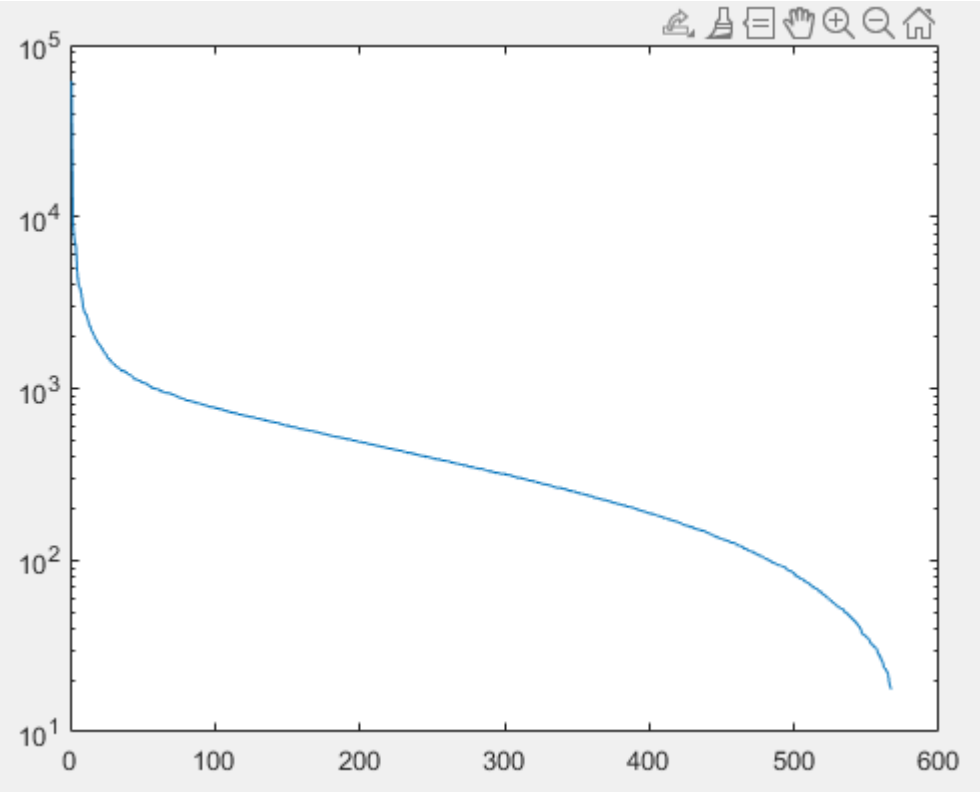
New image makes more sense when shown with a black and white filter.

```
figure
colormap(gray(256));
image(nasa)
title('Grayscale NASA photo');
```



Now, singular value decomposition of the image matrix is taken. Columns of U matrix will be eigen vectors and corresponding diagonal elements of S matrix will be eigen values. Eigen values determine each eigen vectors energy.

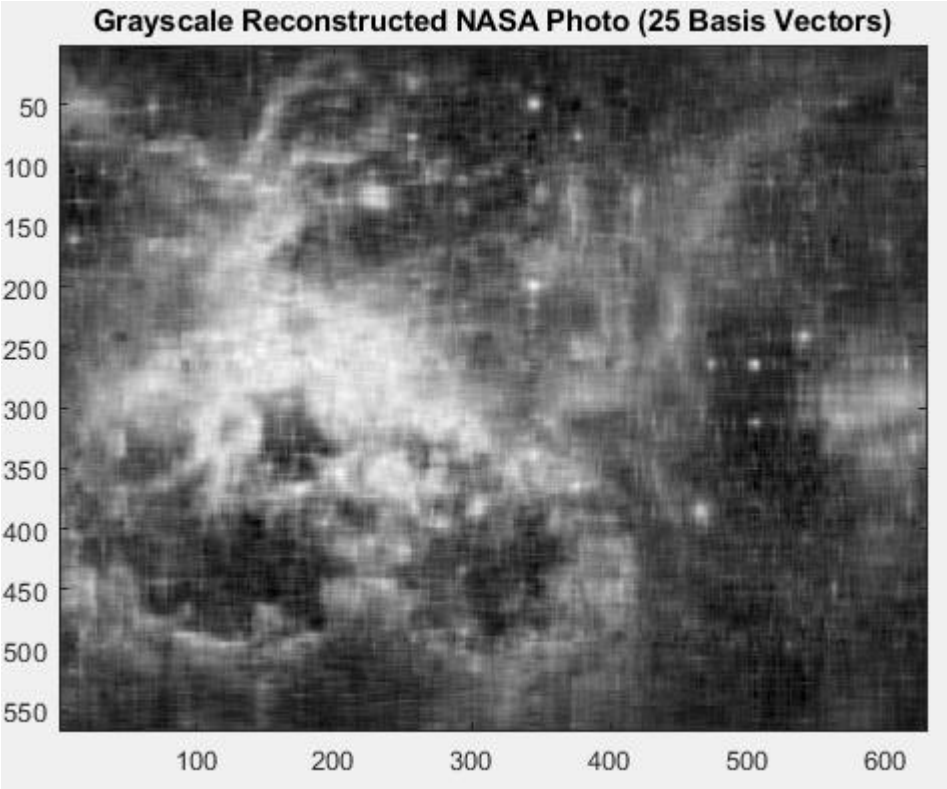
```
[U S V] = svd(nasa);

%Max value is 61930, values fall to less than %2 in after 38 points. This
%shows that among 567 basis vectors, only 38 of them contribute more than
%two percent of what biggest vector contributes to the picture.
figure
semilogy(diag(S))
```



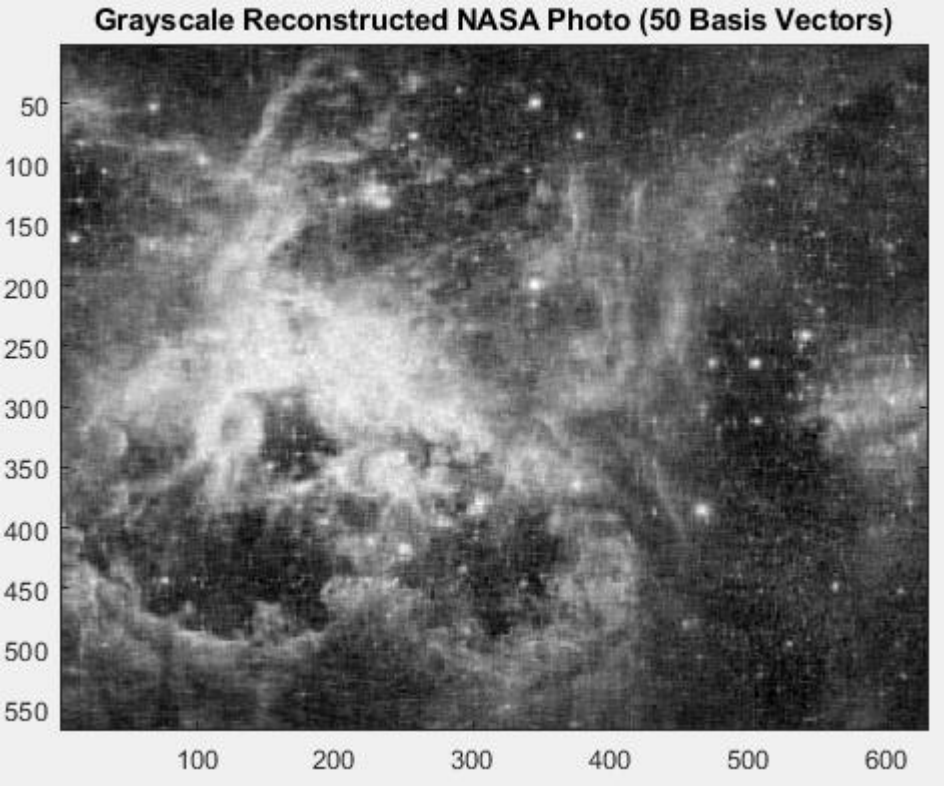
These eigen vectors are basis vectors of the image. By using high energy basis vectors, image can be recreated somewhat accurately. With more basis vectors, recreated image comes closer to the original image. This allows a big reduction in data size and is a standardized method in image compression. Recreations with 25, 50 and 100 basis vectors can be shown as such.

```
nasa100=U(:,1:100)*S(1:100,1:100)*V(:,1:100)';
nasa50=U(:,1:50)*S(1:50,1:50)*V(:,1:50)';
nasa25=U(:,1:25)*S(1:25,1:25)*V(:,1:25)';
```

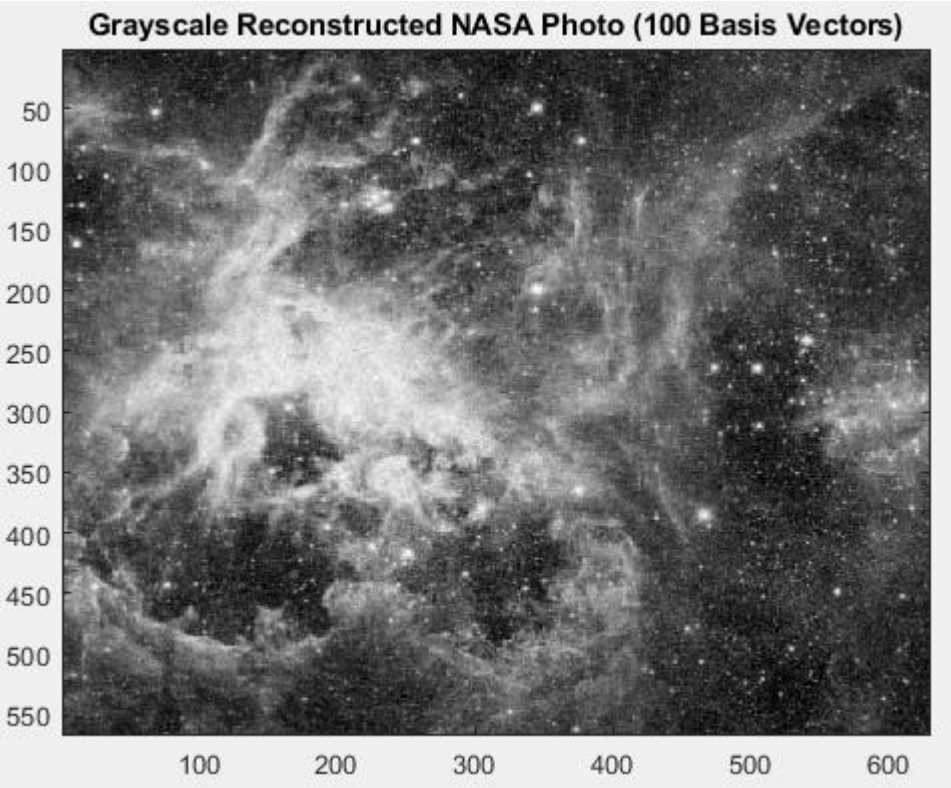
```
%Data is much smaller than before with 25 basis vectors.
%The details are missing, image is blurry and grainy. Although the image
%isn't very clear, it is still recognizable.
figure
colormap(gray(256));
image(nasa25)
title('Grayscale Reconstructed NASA Photo (25 Basis Vectors)')
```



```
%With 25 more basis vectors, image becomes much clearer. Most of the
%details are in and image is very recognizable. Data site doubled compared
%to the one before but it is still very small compared to the original.
figure
colormap(gray(256));
image(nasa50)
title('Grayscale Reconstructed NASA Photo (50 Basis Vectors)')
```



```
%With 100 basis vectors, the image is almost the same as the original.
%There are still tiny errors but those are unrecognizable without through
%examination. Almost all details are in, even very small stars.
%This reconstructed image is still more than 5 times smaller
%than the original.
figure
colormap(gray(256));
image(nasa100)
title('Grayscale Reconstructed NASA Photo (100 Basis Vectors)')
```



SVD Exercise 2:

Part 1:

In this exercise, 32x32 grayscaled faces are loaded from ORL_32x32.mat file. There are 400 faces in the file. File also contains a gnd vector which represents classes of each face. For exercise's purpose, each face in the file has been manipulated and given classes of smiling and neutral.

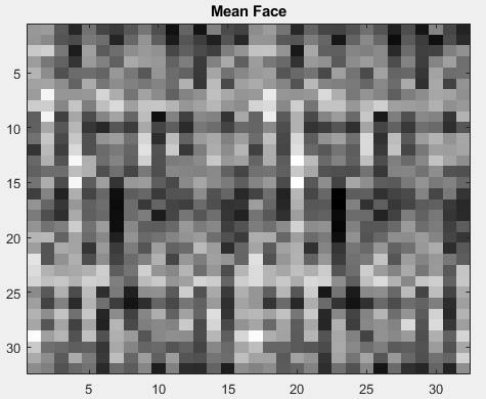
```
clear, clc, close all;

% fea -> Each row represents a face
% gnd -> Each row represents the classification of a face.
% gnd(i) = -1 => fea(i,:) is neutral
% gnd(i) = -2 => fea(i,:) is smiling
fileDir = fullfile(pwd, 'face_databases/ORL_32x32.mat');
load(fileDir)

w = 32;
h = 32;
numFaces = 400;
```

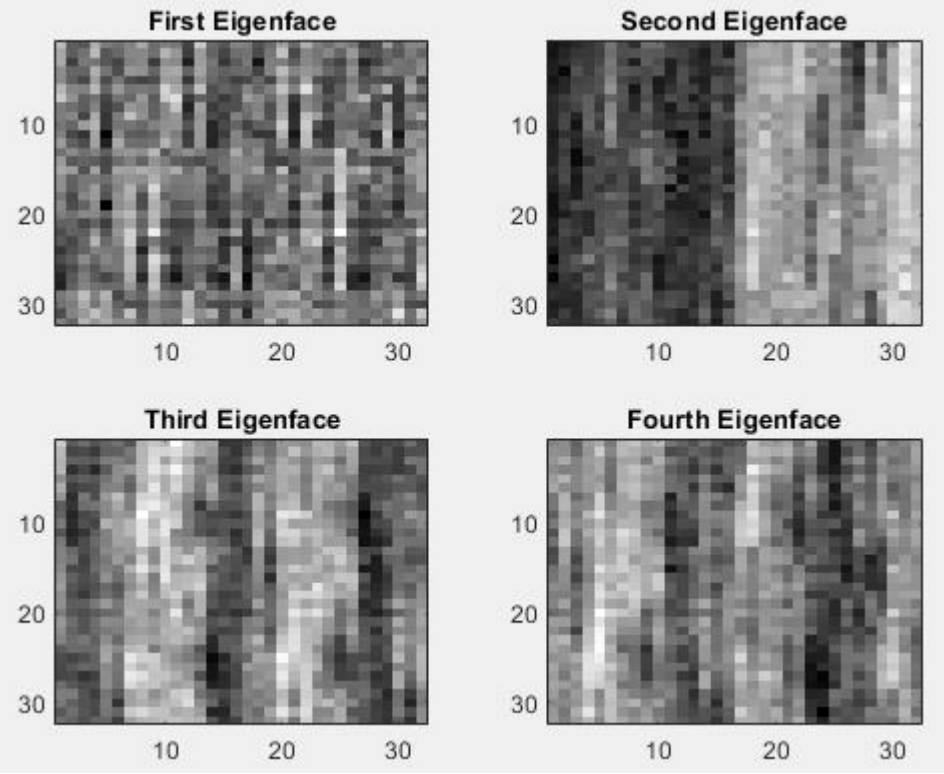
Next, training sets of smiling and neutral faces are initiated. These are selected from the first 40 faces. Also, faces are stored in a matrix called faces.

```
% Determining the size of neutral and smile classes.
neutralSize = 0;
smileSize = 0;
for i=1:40
    if gnd(i) == -1
        neutralSize = neutralSize + 1;
    elseif gnd(i) == -2
        smileSize = smileSize + 1;
    end
end
neutral = zeros(1024, neutralSize);
smile = zeros(1024, smileSize);

% Each column is a face instead of each row.
faces = reshape(fea,1024,400);

% neutral and smile classes are populated.
neutralIndex = 1;
smileIndex = 1;
for i=1:40
    if gnd(i) == -1
        neutral(:,neutralIndex) = faces(:,i);
        neutralIndex = neutralIndex + 1;
    elseif gnd(i) == -2
        smile(:,smileIndex) = faces(:,i);
        smileIndex = smileIndex + 1;
    end
end
```

Each face is centralized by subtracting the mean value. With singular value decomposition of the faces, eigen vectors and eigen values are gathered. Mean face and eigen vectors with largest eigen values are plotted.

```
% The mean of each faces is subtracted.
mean_1 = mean(faces, 2);
mean_2 = repmat(mean_1, 1, size(faces,2));
faces = faces - mean_2;

% Singular value decomposition, which gives eigenvectors and eigenvalues.
[u,s,v] = svd(faces, 0);
eigVals = diag(s);
eigVecs = u;

figure; imagesc(reshape(mean_1, h, w)); title('Mean Face'); colormap(gray);
figure;
subplot(2, 2, 1); imagesc(reshape(u(:, 1), h, w)); colormap(gray); title('First Eigenface');
subplot(2, 2, 2); imagesc(reshape(u(:, 2), h, w)); colormap(gray); title('Second Eigenface');
subplot(2, 2, 3); imagesc(reshape(u(:, 3), h, w)); colormap(gray); title('Third Eigenface');
subplot(2, 2, 4); imagesc(reshape(u(:, 4), h, w)); colormap(gray); title('Fourth Eigenface');
```





SVD analysis is done on natural and smiling matrices. These basis vectors will be used to predict whether a face is smiling or not. Dominant vectors will have a higher chance of being from the right class.

```
function meanMat = getMean(Mat)
mean_1 = mean(Mat, 2);
mean_2 = repmat(mean_1, 1, size(Mat,2));
meanMat = Mat - mean_2;
end


% The mean of smiling and neutral faces are subtracted.
neutral = getMean(neutral);
smile = getMean(smile);

% Svd of neutral faces.
[uN,sN,vN] = svd(neutral, 0);
eigValsN = diag(sN);
eigVecsN = uN;

% Svd of smiling faces.
[uS,sS,vS] = svd(smile, 0);
eigValsS = diag(sS);
eigVecsS = uS;
```

Energies of basis vectors are calculated in order to select high energy vectors. For this exercise, energy threshold is selected as %90 of the highest energy vector.

```
  % Energy of each neutral face eigenvector is calculated to select high
  % energy eigenvectors.
  energyN = zeros(size(uN,2),1);
for i = 1:size(uN,2)
  energyN(i) = sum(eigValsN(1:i));
end
  propEnergyN = energyN./energyN(end);
  percentMarkN = min(find(propEnergyN > 0.9));
  eigenVecsN = uN(:, 1:percentMarkN);

  % Energy of each smiling face eigenvector is calculated to select high
  % energy eigenvectors.
  energyS = zeros(size(uS,2),1);
for i = 1:size(uS,2)
  energyS(i) = sum(eigValsS(1:i));
end
  propEnergyS = energyS./energyS(end);
  percentMarkS = min(find(propEnergyS > 0.9));
  eigenVecsS = uS(:, 1:percentMarkS);
```

Each eigen vector is normalized and projection of faces with vectors from smiling and neutral classes are calculated. These projections, or weights, will determine which class will be predicted for a face.

```
% Each eigen vector is normalized.
for i = 1:size(eigenVecsN,2)
eigenVecsN(:,i) = eigenVecsN(:,i)./sqrt(sum(eigenVecsN(:,i).^2));
end
for i = 1:size(eigenVecsS,2)
eigenVecsS(:,i) = eigenVecsS(:,i)./sqrt(sum(eigenVecsS(:,i).^2));
end

% Projection of all faces onto neutral and smiling eigenvectors stored as
% weights.
nWeights = pinv(eigenVecsN)*faces;
sWeights = pinv(eigenVecsS)*faces;
```

Euclidian norm of each weight vector is calculated and subtracted. If the difference is in favor of neutral, face will be predicted to be natural. Otherwise, prediction will be smiling.

```
% For faces other than in training set, a weight difference is calculated
% by subtracting the euclidian norm of each weight. If the
% difference is greater than zero, picture is predicted to be of a
% neutral face. Else, picture is predicted to be of a smiling face.
smileCount = 0;
neutralCount = 0;
gndPred = string.empty;
for i = 1:numFaces
    Wdif = sqrt(sum(nWeights(:,i).^2)) - sqrt(sum(sWeights(:,i).^2));
    if Wdif > 0
        gndPred(i) = 'neutral';
        neutralCount = neutralCount + 1;
    else
        gndPred(i) = 'smile';
        smileCount = smileCount + 1;
    end
end
gndPred = gndPred';
```

While above %50, results are still not that great. While roughly half of the total faces were smiling, program predicted only 42 smiling faces and 13 of them were wrong predictions. Reason for this may be low count of smiling faces in the training set, which was 11, or low count of total faces in training set altogether. Not many basis vectors for smiling faces can be gathered from such a low amount of data. Another reason may be the resolution of faces, which was 32x32. With that resolution and not-that-clear difference between smiling and neutral faces, basis vectors may be less than healthy. Even with all the disadvantages, program was able to predict around %75 of faces accurately.

```
Predicted Neutral right: 209          Predicted Neutral but were wrong: 149
Predicted Smiling right: 29           Predicted Smiling but were wrong: 13
```

SVD Exercise 2:

Part 2:

With the previously established system, genders of different 36x36 faces will be predicted. For this part of the exercise, training set for men and women will consist of 2500 faces each and testing set will consist of 200 faces each. These faces are loaded into corresponding matrices.

```
clear, clc, close all;

w = 36;
h = 36;
trainFaces = 2500;
testFaces = 200;

training_fileDir = fullfile(pwd, 'gender_classification/training');
testing_fileDir = fullfile(pwd, 'gender_classification/testing');

for k = 1:trainFaces
  trainMenFilename = strcat(training_fileDir,'\men\',num2str(k), '.jpg');
  imageData = imread(trainMenFilename);
  trainMen_int(:,k) = reshape(imageData,[],1);
end
trainMen = im2double(trainMen_int);

for k = 1:trainFaces
  trainWomenFilename = strcat(training_fileDir,'\women\',num2str(k), '.jpg');
  imageData = imread(trainWomenFilename);
  trainWomen_int(:,k) = reshape(imageData,[],1);
end
trainWomen = im2double(trainWomen_int);
for k = 1:testFaces
  testMenFilename = strcat(testing_fileDir,'\men\',num2str(k), '.jpg');
  imageData = imread(testMenFilename);
  testMen_int(:,k) = reshape(imageData,[],1);
end
testMen = im2double(testMen_int);


for k = 1:testFaces
  testWomenFilename = strcat(testing_fileDir,'\women\',num2str(k), '.jpg');
  imageData = imread(testWomenFilename);
  testWomen_int(:,k) = reshape(imageData,[],1);
end
testWomen = im2double(testWomen_int);
```

Training sets are centralized and eigen vectors and values are gathered with SVD analysis.

```
trainMen = getMean(trainMen);
trainWomen = getMean(trainWomen);

% Svd of male faces.
[uM,sM,vM] = svd(trainMen, 0);
eigValsM = diag(sM);
eigVecsM = uM;

% Svd of female faces.
[uF,sF,vF] = svd(trainWomen, 0);
eigValsF = diag(sF);
eigVecsF = uF;
```

Later, high energy vectors are selected. Energy threshold is selected as %99.9 of the highest energy vector. This corresponds to 102 female vectors and 105 male vectors.

```
% Energy of each neutral face eigenvector is calculated to select high
% energy eigenvectors.
energyM = zeros(size(uM,2),1);
energyThM = 0.999;
for i = 1:size(uM,2)
energyM(i) = sum(eigValsM(1:i));
end
propEnergyM = energyM./energyM(end);
percentMarkM = size(eigVecsM,1) - min(find(propEnergyM > energyThM));
eigenVecsM = uM(:, 1:percentMarkM);
eigenValsM = sM(:, 1:percentMarkM);

energyF = zeros(size(uF,2),1);
energyThF = 0.999;
for i = 1:size(uF,2)
energyF(i) = sum(eigValsF(1:i));
end
propEnergyF = energyF./energyF(end);
percentMarkF = size(eigVecsF,1) - min(find(propEnergyF > energyThF));
eigenVecsF = uF(:, 1:percentMarkF);
eigenValsF = sF(:, 1:percentMarkF);

for i = 1:size(eigenVecsM,2)
eigenVecsM(:,i) = eigenVecsM(:,i)./sqrt(sum(eigenVecsM(:,i).^2));
end
```

Eigen vectors are normalized and projected on the testing set.

```
for i = 1:size(eigenVecsM,2)
eigenVecsM(:,i) = eigenVecsM(:,i)./sqrt(sum(eigenVecsM(:,i).^2));
end

for i = 1:size(eigenVecsF,2)
eigenVecsF(:,i) = eigenVecsF(:,i)./sqrt(sum(eigenVecsF(:,i).^2));
end

% Projection of eigenvectors on men faces from testing set.
WeightsMonM = pinv(eigenVecsM)*testMen;
WeightsFonM = pinv(eigenVecsF)*testMen;

WeightsMonF = pinv(eigenVecsM)*testWomen;
WeightsFonF = pinv(eigenVecsF)*testWomen;
```

After predicting through testing sets, around %82 of predictions are successful. Results are way more consistent compared to the previous part of the exercise. This is due to having a much bigger training set, having data with sharp, non-ambigious differences and having more data with 36x36 faces instead of 32x32.

More basis vectors may improve the results with the cost of more data space. Depending on application, energy threshold and even difference function may be altered to get better results.

```
% For faces other than in training set, a weight difference is calculated
% by subtracting the euclidian norms of projections. Prediction will depend
% on difference of each norm.
errorOnM = 0;
errorFaces = zeros(testFaces);
for i = 1:testFaces
    Wdif = sqrt(sum(WeightsFonM(:,i).^2)) - sqrt(sum(WeightsMonM(:,i).^2));
    if Wdif > 0
        errorOnM = errorOnM + 1;
        errorFaces(i) = 1;
    end
end
fprintf('Number of error while testing on men faces: %d out of %d\n', errorOnM, testFaces);
fprintf('Success rate: %1.3f\n\n\n', 100*(1 - errorOnM/testFaces));
errorOnF = 0;
errorFaces = zeros(testFaces);
for i = 1:testFaces
    Wdif = sqrt(sum(WeightsMonF(:,i).^2)) - sqrt(sum(WeightsFonF(:,i).^2));
    if Wdif > 0
        errorOnF = errorOnF + 1;
        errorFaces(i) = 1;
    end
end
fprintf('Number of error while testing on women faces: %d out of %d\n', errorOnF, testFaces);
fprintf('Success rate: %1.3f\n\n\n', 100*(1 - errorOnF/testFaces));
```

```
Number of error while testing on men faces: 37 out of 200
Success rate: 81.500


Number of error while testing on women faces: 36 out of 200
Success rate: 82.000
```

As an extra example, energy threshold for basis vectors are changed for female vectors to %99.999 of the highest energy vector and %99.996 for males. With this, only one vector for female will be gathered and six vectors for males. The function for testing for male set is changed to difference between the average of sum of elements in weight vector. The function for female testing set is changed to average of sum of squared elements in weight vector. With all these manipulations, %100 of predictions are correct and with only 7 vectors in total. This solves the problem of data space and accuracy.

```
energyM = zeros(size(uM,2),1);
energyThM = 0.99996;
for i = 1:size(uM,2)
energyM(i) = sum(eigValsM(1:i));
end
propEnergyM = energyM./energyM(end);
percentMarkM = size(eigVecsM,1) - min(find(propEnergyM > energyThM));
eigenVecsM = uM(:, 1:percentMarkM);
eigenValsM = sM(:, 1:percentMarkM);

energyF = zeros(size(uF,2),1);
energyThF = 0.99999;
for i = 1:size(uF,2)
energyF(i) = sum(eigValsF(1:i));
end
propEnergyF = energyF./energyF(end);
percentMarkF = size(eigVecsF,1) - min(find(propEnergyF > energyThF));
eigenVecsF = uF(:, 1:percentMarkF);
eigenValsF = sF(:, 1:percentMarkF);
for i = 1:testFaces
    Wdif = sum(WeightsFonM(:,i))/percentMarkF - sum(WeightsMonM(:,i))/percentMarkM;
    if Wdif > 0
        errorOnM = errorOnM + 1;
        errorFaces(i) = 1;
    end
end
for i = 1:testFaces
    Wdif = sum(WeightsMonF(:,i).^2)/percentMarkM - sum(WeightsFonF(:,i).^2)/percentMarkF;
    if Wdif > 0
        errorOnF = errorOnF + 1;
        errorFaces(i) = 1;
    end
end
```

```
Number of error while testing on men faces: 0 out of 200
Success rate: 100.000


Number of error while testing on women faces: 0 out of 200
Success rate: 100.000
```

Codes used in this report can be found at
https://github.com/meserbetcioglu/ehb328hw2