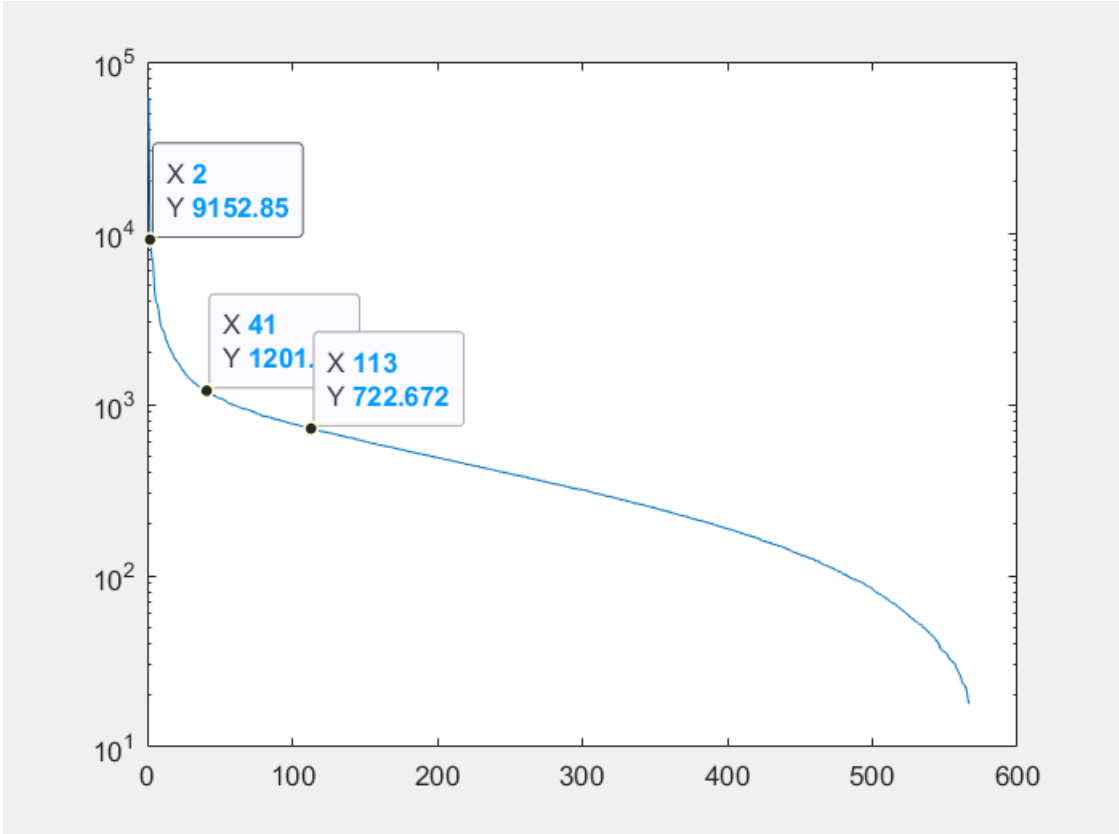
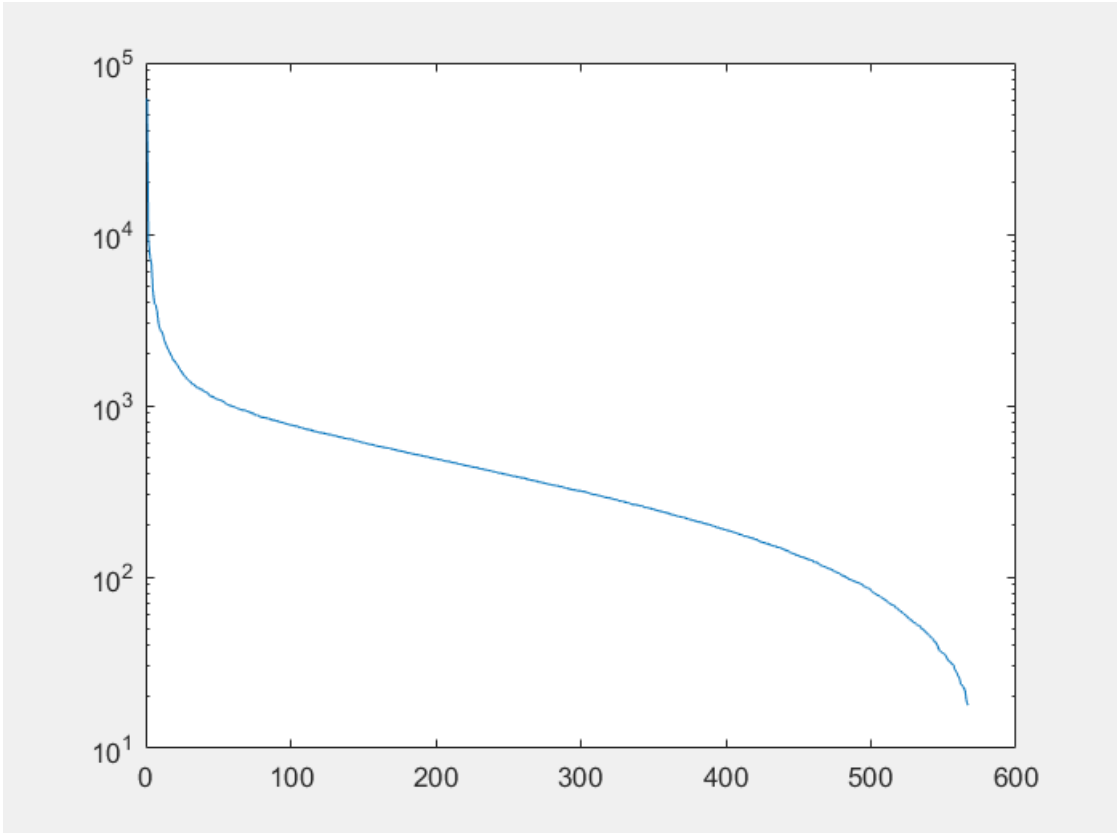
HALİT ERDOĞAN 040160260

SVD Exercise 1 (Tarantula Nebula)

```
%read the picture
nasacolor=imread('TarantulaNebula.jpg');

%display the picture first as it is
figure; image(nasacolor); title('colored nasa');

nasa_summed=sum(nasacolor,3,'double'); %sum up
red+green+blue
m=max(max(nasa_summed)); %find the max value
nasa_normalized=nasa*255/m; %make this be bright white

colormap(gray(256));

%display grayscale photo
figure; imshow(nasa_normalized); title('Grayscale NASA
photo');

%apply singular value decomposition
[U, S, V]=svd(nasa_normalized);

%display eigenvalues in a log scale
semilogy(diag(S))

%define different picture matrices with different numbers
of eigenvectors
%and eigenvalues
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)';

%display each of the pictures defined above
figure; image(nasa100); title('NASA100 photo');

image(nasa50); title('NASA50 photo');

image(nasa25); title('NASA25 photo');
```





Eigenvalues decrease rapidly after the 50th corresponding eigenvector.

SVD Exercise 2.1 (Smile vs Neutral)

```
%define empty arrays for later use
neutral = [];
smile = [];

%Nomalize each vector to unit
[nSmp,nFea] = size(fea);
for i = 1:nSmp
     fea(i,:) = fea(i,:) ./ max(1e-12,norm(fea(i,:)));
end

%when fea is displayed with display_faces.m, it is seen
that 3rd picture of
%each individual is smiling. We label every (3+11k)th face
as smile.
smile = [];
for k = 1:11
    l = (11 .* (k-1)) + 3;
    smile = [smile, l];
end

%The rest of the faces are neutral
neutral = [];
for k = 1:121
    if rem(k,11) == 3
        k = k + 1;
    else
        neutral = [neutral, k];
    end
end

%The transpose is taken to make each face a column
faces = fea';
%sizes of the pictures are defined for plotting
h = 32; w = 32;
%mean face is calculated
meanFace = mean(faces, 2);
%mean face is subtracked from faces
faces = faces - repmat(meanFace, 1, nSmp);

%Singular value decomposition is applied
[u,d,v] = svd(faces, 0);

% Eigenvalues and eigenvectors are calculated
eigVals = diag(d);
eigVecs = u;
```

```
%figures are plotted for the mean face and the first 3
eigenvectors/eigenfaces
figure; imshow(reshape(meanFace, h, w)); title('Mean
Face');
figure;
subplot(1, 3, 1); imagesc(reshape(u(:, 1), h, w));
colormap(gray); title('First Eigenface');
subplot(1, 3, 2); imagesc(reshape(u(:, 2), h, w));
colormap(gray); title('Second Eigenface');
subplot(1, 3, 3); imagesc(reshape(u(:, 3), h, w));
colormap(gray); title('Third Eigenface');

% The cumulative energy content for the m'th eigenvector
is the sum of the energy content across eigenvalues 1:m
energy = [];
for i = 1:nSmp
    energy(i) = sum(eigVals(1:i));
end
propEnergy = energy./energy(end);

% Determine the number of principal components required to
model 90% of data variance
percentMark = min(find(propEnergy > 0.9));

% Pick those principal components
eigenVecs = u(:, 1:percentMark);

%project each of the neutral and smiling faces onto the
corresponding eigenfaces
neutralFaces = faces(:, neutral);
smileFaces = faces(:, smile);
neutralWeights = eigenVecs' * neutralFaces;
smileWeights = eigenVecs' * smileFaces;

%the means of each weight matrices are taken, giving us a
mean neutral and
%smiling weight which we can finally use to classify
remaining pictures/vectors
mean_neutralWeights = mean(neutralWeights, 2);
mean_smileWeights = mean(smileWeights, 2);


%we do the classification
%122 to 165 is the remaining indices and are our test
data, since the first 121 is labeled and therefore has
become our training data
```

```
%we multiply each column by eigenvectors/eigenfaces, one
by one, and calculate its difference with mean neutral
smiling faces
%we then compare the two differences and label that
particular vector as whichever has the smaller difference
for i = 122:165

    weightdiff_smile = abs((eigenVecs' * faces(:,i)) -
mean_smileWeights);
    weightdiff_neutral = abs((eigenVecs' * faces(:,i)) -
mean_neutralWeights);
    if sum(weightdiff_smile) > sum(weightdiff_neutral)
        neutral = [neutral, i];
    end
    if sum(weightdiff_smile) < sum(weightdiff_neutral)
        smile = [smile, i];
    end
end
```

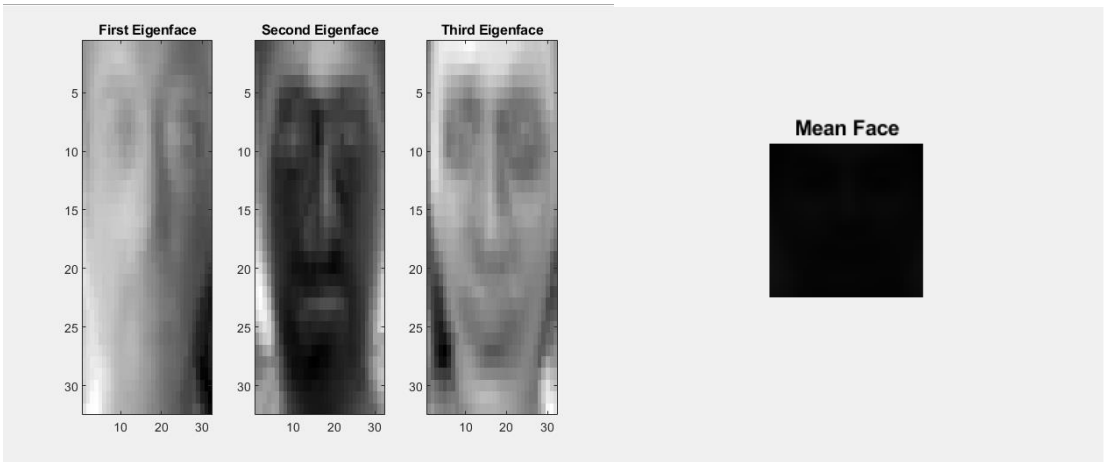
Since we knew that 3rd picture of each individual is smiling, we expected the smile matrix to be,

```
smile = [ 3   14   25   36   47   58   69   80   91   102
113  124  135  146  157]
```

However, the code returned smile as,

```
smile = [ 3   14   25   36   47   58   69   80   91   102
113  127  135  146  157]
```

With the only mistake being 127 instead of 124. For further improvement, we can apply SVD only to second half of the features (from 512 to 1024) which corresponds to the bottom half of each picture. The bottom of the picture includes themouth,which is the main indicator of a smile.



SVD Exercise 2.2 (Gender Classification)

```
%read all pictures in the training/men folder to struct
men_training_folder = 'training/men/';
men_training_struct = dir([men_training_folder '*.jpg']);

%read all pictures in the training/women folder to struct
women_training_folder = 'training/women/';
women_training_struct = dir([women_training_folder
'*.jpg']);

%convert struct to array and array to matrix for men
training pictures
men_training_pics_array = [];
for k = 1:length(men_training_struct)
    men_pic_index = imread([men_training_folder
men_training_struct(k).name]);
    men_pics_index = reshape(men_pic_index, 1, []);
    men_training_pics_array = [men_training_pics_array,
men_pics_index];
end

%define number of samples(nSmp) and number of
features(nFea) which is the
%same for both men and women
nSmp = 2500;
nFea = 1296;

%2500 is the number of pictures, 1296 is the number of
pixels
men_training_pics = reshape(men_training_pics_array,
[1296, 2500]);

%scale the features of men training pictures
men_training_pics = double(men_training_pics);
maxValue_m = max(max(men_training_pics));
men_training_pics = men_training_pics ./ maxValue_m;

%convert struct to array and array to matrix for women
training pictures
women_training_pics_array = [];
for k = 1:length(women_training_struct)
    women_pic_index = imread([women_training_folder
women_training_struct(k).name]);
    women_pics_index = reshape(women_pic_index, 1, []);
    women_training_pics_array =
[women_training_pics_array, women_pics_index];
end
```

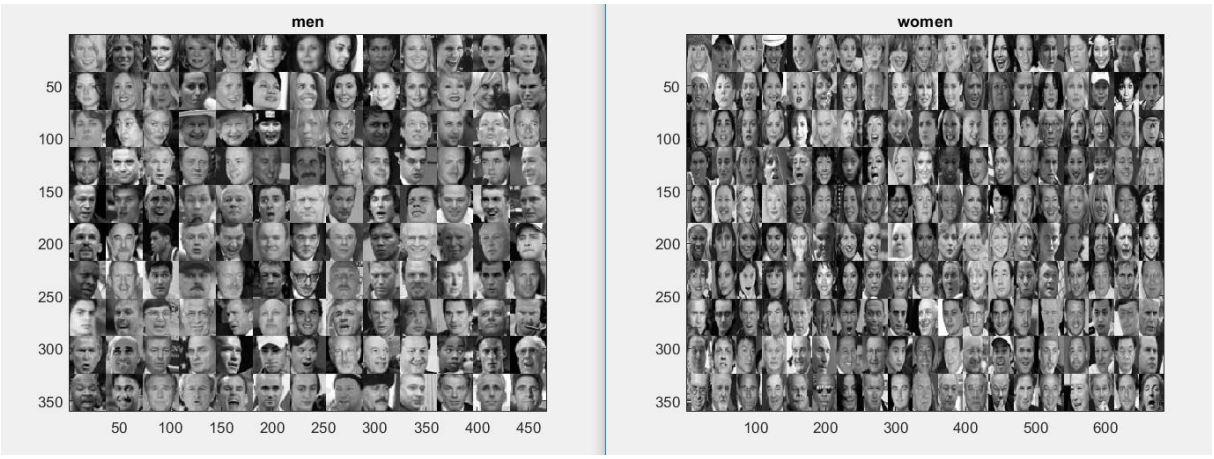
```
%2500 is the number of pictures, 1296 is the number of
pixels
women_training_pics = reshape(women_training_pics_array,
[1296, 2500]);

%scale the features of women training pictures
women_training_pics = double(women_training_pics);
maxValue_w = max(max(women_training_pics));
women_training_pics = women_training_pics ./ maxValue_w;

%subtrack mean men face
meanMen = mean(men_training_pics, 2);
men_training_pics = men_training_pics - repmat(meanMen, 1,
nSmp);

%subtrack mean women face
meanWomen = mean(women_training_pics, 2);
women_training_pics = women_training_pics -
repmat(meanWomen, 1, nSmp);

%SVD of both men and women pictures
[um, dm, vm] = svd(men_training_pics, 'econ');
[uw, dw, vw] = svd(women_training_pics, 'econ');

%eigenvalues and eigenvectors of SVD of men pictures
eigenVals_men = diag(dm);
eigenVecs_men = um;

%eigenvalues and eigenvectors of SVD of women pictures
eigenVals_women = diag(dw);
eigenVecs_women = uw;

% The cumulative energy content for the m'th eigenvector
is the sum of the energy content across eigenvalues 1:m
energy_m = [];
for i = 1:1296
    energy_m(i) = sum(eigenVals_men(1:i));
end
propEnergy_m = energy_m./energy_m(end);

% Determine the number of principal components required to
model 90% of data variance
percentMark_m = min(find(propEnergy_m > 0.9));
percentMark_m = percentMark_m +1;

% Pick those principal components
```

```
eigVecs_men = um(:, 1:percentMark_m);

%apply the same process to women
energy_w = [];
for k = 1:1296
    energy_w(k) = sum(eigenVals_women(1:k));
end
propEnergy_w = energy_w ./ energy_w(end);

percentMark_w = min(find(propEnergy_w > 0.9));

eigVecs_women = uw(:, 1:percentMark_w);

%To calculate weights of each gender reduced eigenvectors
are multiplied by
%the training pictures of each gender
men_weights = eigVecs_men' * men_training_pics;
women_weights = eigVecs_women' * women_training_pics;

%The means of weights of each gender is taken
mean_men_weights = mean(men_weights, 2);
mean_women_weights = mean(women_weights, 2);
%Training process is complete!

%read all the pictures in the testing/men folder
men_testing_folder = 'testing/men/';
men_testing_struct = dir([men_testing_folder '*.jpg']);

%read all the pictures in the testing/women folder
women_testing_folder = 'testing/women/';
women_testing_struct = dir([women_testing_folder
'*.jpg']);

%convert men testing pictures struct to array
men_testing_pics_array = [];
for k = 1:length(men_testing_struct)
    men_test_pic_index = imread([men_testing_folder
men_testing_struct(k).name]);
    men_test_pics_index = reshape(men_test_pic_index, 1,
[]);
    men_testing_pics_array = [men_testing_pics_array,
men_test_pics_index];
end

%convert women testing pictures struct to array
women_testing_pics_array = [];
for k = 1:length(women_testing_struct)
```

```
    women_test_pic_index = imread([women_testing_folder
women_testing_struct(k).name]);
    women_test_pics_index = reshape(women_test_pic_index,
1, []);
    women_testing_pics_array = [women_testing_pics_array,
women_test_pics_index];
end

%convert women testing pictures array to 1296x200 matrix
and convert it
%from uint8 to double and divide by 255 for normalization
which is the max
%value
women_testing_pics = reshape(women_testing_pics_array,
[1296, 200]);
women_testing_pics = double(women_testing_pics);
women_testing_pics_weights = women_testing_pics ./ 255;

%convert women testing pictures array to 1296x200 matrix
and convert it
%from uint8 to double and divide by 255 for normalization
which is the max
%value
men_testing_pics = reshape(men_testing_pics_array, [1296,
200]);
men_testing_pics = double(men_testing_pics);
men_testing_pics_weights = men_testing_pics ./ 255;

%create empty arrays for later use
tested_women = [];
tested_men = [];

%we do the classification of women testing pictures
%we multiply each column by eigenvectors/eigenfaces, one
by one, and calculate its difference with mean men and
women faces
%we then compare the two differences and label that
particular vector as whichever has the smaller difference
for i = 1:200
    weightdiff_women = abs((eigVecs_women' *
women_testing_pics_weights(:,i)) - mean_women_weights);
    weightdiff_men = abs((eigVecs_women' *
women_testing_pics_weights(:,i)) - mean_men_weights);
    if sum(weightdiff_women) < sum(weightdiff_men)
        tested_women = [tested_women,
women_testing_pics_weights(:,i)];
    end
```

```
    if sum(weightdiff_women) > sum(weightdiff_men)
        tested_men = [tested_men,
women_testing_pics_weights(:,i)];
    end
end

%we do the classification of men testing pictures
%we multiply each column by eigenvectors/eigenfaces, one
by one, and calculate its difference with mean men and
women faces
%we then compare the two differences and label that
particular vector as whichever has the smaller difference
for i = 1:200
    weightdiff_women = abs((eigVecs_men' *
men_testing_pics_weights(:,i)) - mean_women_weights);
    weightdiff_men = abs((eigVecs_men' *
men_testing_pics_weights(:,i)) - mean_men_weights);
    if sum(weightdiff_women) > sum(weightdiff_men)
        tested_men = [tested_men,
men_testing_pics_weights(:,i)];
    end
    if sum(weightdiff_women) < sum(weightdiff_men)
        tested_women = [tested_women,
men_testing_pics_weights(:,i)];
    end
end

%finally, we display tested_men and tested_women matrices
in different figures
faceW = 36;
faceH = 36;
numPerLine = 13;
ShowLine = 10;
Y_m = zeros(faceH*ShowLine,faceW*numPerLine);
for i=0:ShowLine-1
    for j=0:numPerLine-1
        a =
reshape(tested_men(:,i*numPerLine+j+1),[faceH,faceW]);
        Y_m(i*faceH+1:(i+1)*faceH,j*faceW+1:(j+1)*faceW) =
a;
    end
end

figure; imagesc(Y_m);colormap(gray); title ('men');


faceW = 36;
```

```
faceH = 36;
numPerLine = 19;
ShowLine = 10;
Y_w = zeros(faceH*ShowLine,faceW*numPerLine);
for i=0:ShowLine-1
    for j=0:numPerLine-1
        a =
reshape(tested_women(:,i*numPerLine+j+1),[faceH,faceW]);
        Y_w(i*faceH+1:(i+1)*faceH,j*faceW+1:(j+1)*faceW) =
a;
    end
end

figure; imagesc(Y_w);colormap(gray); title('women');
```



For further improvement, we can apply SVD only to the part of the picture which includes the mouth as we suggested for the previous example. Although not a universal solution this time, for the particular dataset we are working with, by inspection, it is seen that women appear to smile more compared to men.