Exercise 10.1 (June 5, 2020)

-B9TB1707

Question:

- Calculate the first 20 eigenvalues of the covariance matrix of X and plot them
- Calculate also the eigenvectors and then display them as images of 92x112 pixels

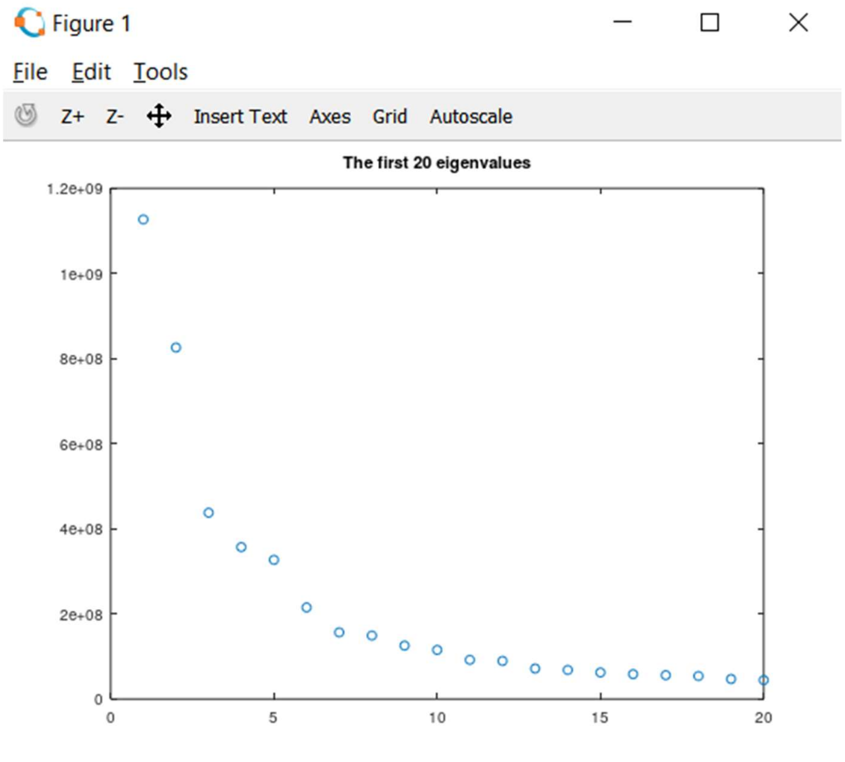
Solution:

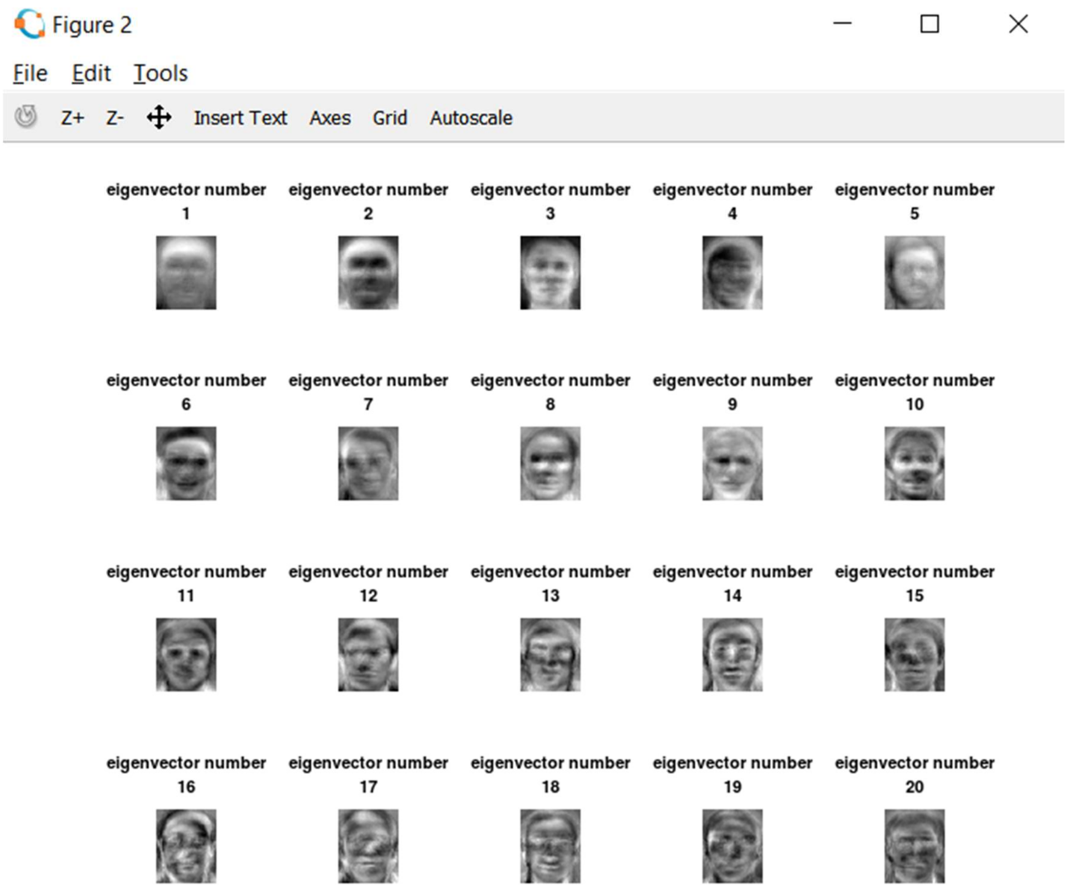
My code for the solution is as follows:

CAPS_11_B9TB1707_11.1.m

```
load_faces
[U,W,V]=svds(X-ones(400,1)*mean(X),20);
W=W^2;
figure(1);
plot(diag(W),'o')
title('The first 20 eigenvalues')
figure(2);
for i=1:20,
 subplot(4,5,i)
 svec=V(:,i);
 imshow(reshape(svec,[112,92]),[min(svec),max(svec)])
 title({'eigenvector number';i})
 hold on;
 end
```

And the output is as follows:





How it works:

1. Line 1 runs the scripts given in class to load the images and to initialize matrix X.
2. Line 2 performs singular value decomposition on the covariance matrix of X. the covariance matrix of X is calculated by subtracting the mean of X from every term of X. The svds() function calculates a specified number of the largest singular values for a given matrix. That is why we give it an extra parameter, 20.
3. From the definition of singular value decomposition we get the values of the eigenvalues by squaring the W in Line 3 (The values stored inside the diagonal elements of W after SVD is the square root of the eigenvalues.)
4. Line 4 creates a new plot window.
5. Line 5 plots the eigenvalues.
6. Line 6 titles the graph.
7. Line 7 creates a new plot for the images created using the eigenvectors.
8. Line 8 creates a for loop construct.
9. Line 9 divides the plot into twenty spaces using the subplot function.
10. Line 10 retrieves the eigenvector stored in the i^{th} column and stores it in variable svec.
11. Line 11 reconstructs the image from the eigenvector along a normalized scale for brightness and specifies the dimensions using the reshape() and imshow() functions.
12. Line 12 titles the image.
13. Line 13 is a hold on command to hold on to the data in the plot.

14. Line 14 ends the for block.

Conclusion:

And thus rather than to find the eigenvalues and eigenvectors directly using the command eig(), I used singular value decomposition to more efficiently solve the problem. Though eig() is a perfectly valid option for problems involving a small amount of data, it becomes too intensive on the system when a large set of data is involved (especially for images, where each pixel is a data point). Using SVD we decompose the matrix into 3 matrices, out which 1 is the eigenvectors and one contains the square root of the eigenvalues in its diagonals. It is then very easy to extract this data and reconstruct the images from the data.