CS/NEUR125 Brains, Minds, and Machines

Lab 3: Eigenfaces Approach to Face Recognition

Due: Wednesday, February 15

This lab explores the "Eigenfaces" approach to building an automated face recognition system (Turk & Pentland, 1991), which is based on a statistical method known as Principal Components Analysis. You will work with MATLAB two programs that explore aspects of the Eigenface representation of a set of face images, and the performance of this method on a small dataset that includes face images with varying lighting, expression, and orientation. For one exercise, you will create a MATLAB *script* to display the results of model simulations, reinforcing programming concepts that you learned in the first two labs.

To begin, create a copy of this Google document, modify the title of the copy to include your partner names, and share the copy between partners, as you did in previous labs (for review, see the Working with Google Docs for Lab Handouts webpage). Questions that you should try to answer during lab are shown in blue, and those that you could answer later are shown in purple.

Start MATLAB on the lab Mac that you are sharing. Use **Fetch** to download a folder of code and image files named **EigenfacesLab** from the CS file server to the Desktop on your Mac. You will find this folder inside the download folder in your individual account on the CS server (for review, see details of this process for both Macs and PCs on this webpage). As in Lab 2, set the **Current Folder** in MATLAB to be the Desktop of your Mac. In the Command Window, enter the following command to enable MATLAB to access code and data files in this folder:

```
>> addpath(genpath('EigenfacesLab'))
```

In the contents of the Current Folder listed on the left of the MATLAB window, you should see the EigenfacesLab folder, and its name should no longer be grayed out.

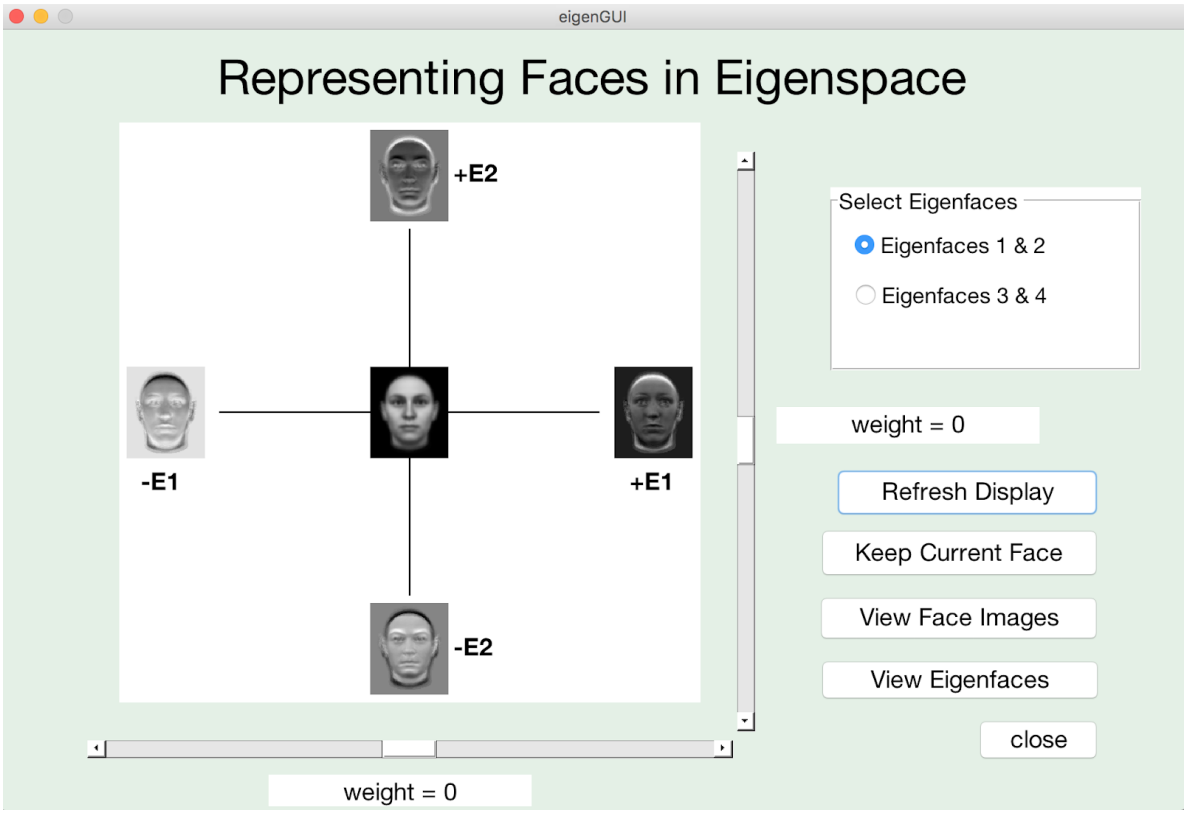
I. Using "Eigenfaces" to Represent Face Images

The Eigenfaces approach to face recognition captures the variation that exists in a collection of face images, through a representation in which each image is encoded with a small number of parameters, or *weights*. The Eigenfaces are the *principal components* of the particular set of face images in the dataset. Each real face image can be reconstructed as a combination of the various Eigenfaces. The contribution of a particular Eigenface to a particular face image is the *weight* of that Eigenface (principal component) in that particular face image. The Eigenfaces can be ordered in such a way that the first Eigenface has a pattern of image brightness that captures the largest variance across the set of images, the second Eigenface captures the next largest variance in brightness, and so on. The Eigenfaces, which can be viewed as images, implicitly capture facial characteristics that we intuitively use to distinguish faces, such as the

shape of the head and the brightness and geometric arrangement of facial features. This first part of this lab explores the connection between the Eigenface representation and facial appearance. To begin, enter the following command in the Command Window:

```
>> eigenGUI
```

This opens a window for an interactive program that looks like this:



The large image display shows an *average face* at the center, obtained by calculating the **mean** of a set of 100 face images. The set of images can be viewed in a separate figure window by clicking the **View Face Images** button. The principal components (called *Eigenfaces* in this context) were computed from this set of 100 images. You can view the top 25 Eigenfaces by clicking on the **View Eigenfaces** button. In the initial display shown above, the first Eigenface is portrayed as images on the horizontal axis, with a "positive" version displayed on the far right (labeled **+E1** and associated with a positive weight, as described in class), and a "negative" version displayed on the far left end of the horizontal axis (labeled **-E1** and associated with a negative weight). The second Eigenface is shown on the vertical axis, with the positive version at the top (labeled **+E2**) and negative version at the bottom (labeled **-E2**).

Using the graphical interface, you can view the result of adding these two Eigenfaces to the average face, for different combinations of weights for E1 and E2. You can change the weights using the **horizontal and vertical sliders** below and to the right of the visual display. The position of each slider can be changed either by dragging the white rectangle that initially appears at the center of the slider, or by clicking on one of the triangles at the two ends of the

slider. As you alter the sliders, the new weight is printed in the adjacent textboxes (**weight = 0** in the above picture), and the result of adding the scaled Eigenfaces to the average face appears on the display. After you adjust the weights, you can click on the **Keep Current Face** button to keep a particular composite image on the display. Clicking on the **Refresh Display** button will reset the display back to its original state. You can also view the third and fourth Eigenfaces by selecting **Eigenfaces 3 & 4** in the upper right corner and then refreshing the display. These two Eigenfaces capture more subtle variations between faces in the dataset.

- **Q1.** Consider the effect of adding only the first Eigenface (E1) to the average face, with a large positive or large negative weight. Within each Eigenface image shown at the far ends of the horizontal axis, regions that are very bright or very dark have a larger effect when added to the average face. For the first Eigenface, where do these regions occur? Compare the composite images with weights of 40 vs. -40. How does the appearance of these two faces differ? Describe at least three differences in their appearance.

- **Q2.** Make the same comparison for Eigenfaces 2, 3, and 4 that you did in Q1. Again describe how the appearance of the average face changes when only one of these Eigenfaces is added with a weight of 40 vs. -40. Consider, for example, differences in the location or appearance of the eyes, nose, and mouth, or overall shape of the head (for each Eigenface, describe at least three facial characteristics that are altered).

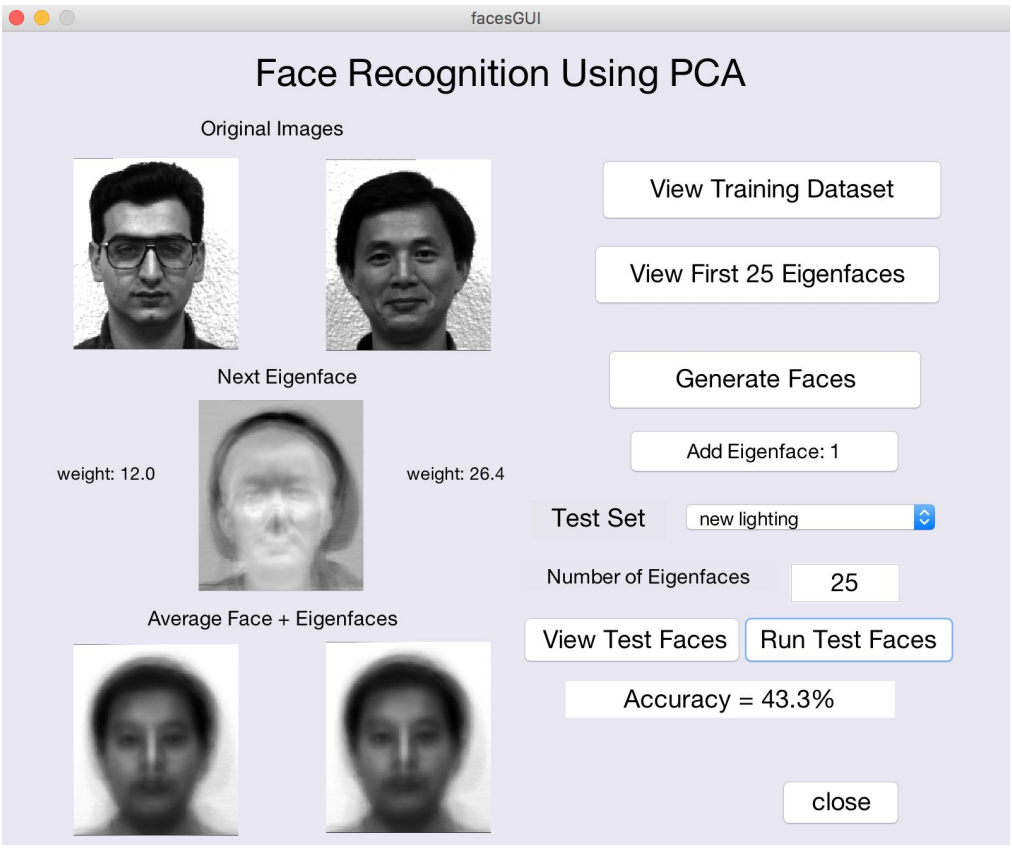
- **Q3.** Examine the two face composite images below. Each one is a composite of two Eigenfaces (either E1 & E2, or E3 & E4), with extreme values for each of the two weights (40 or -40). For each example, what are the component Eigenfaces and weights?

II. Assessing the Performance of the Eigenfaces Approach to Face Recognition

This part of the lab explores the performance of the Eigenfaces approach for recognition, using the Yale Face Database, which includes multiple images of 15 individuals, with variations in facial expression and direction of the light source. This enables us to explore how well this approach can generalize to the recognition of faces whose appearance differs from those used to construct the Eigenfaces. To begin, enter the following command in the Command Window:

```
>> facesGUI
```

An interactive window will appear that looks something like that shown on the next page (some regions will initially be blank).



The set of 105 images used to compute the Eigenfaces can be viewed in a separate figure window by clicking the **View Training Dataset** button. You can view the top 25 Eigenfaces by clicking on the **View First 25 Eigenfaces** button.

- **Q4.** You'll see that the Eigenfaces here look quite different from those generated from the face images used in the eigenGUI program. Why are the Eigenfaces different? Select one Eigenface in the set of 25 and describe in general terms, what features of an average face may be altered if a very large weight (positive or negative) were used to create a composite image?

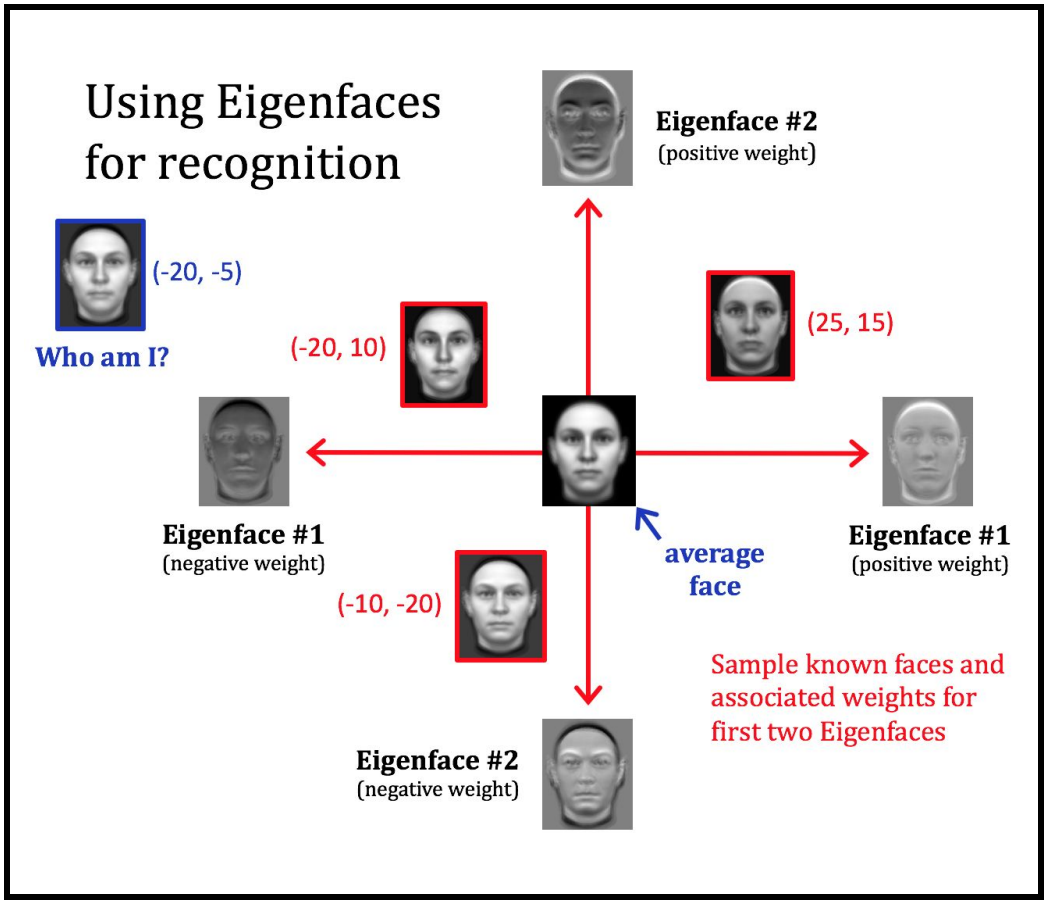
Each face image can be encoded by a set of numerical weights, one for each Eigenface that is used to construct the representation. We noted earlier that the Eigenfaces (and the associated weights) can be ordered in such a way that the first Eigenfaces capture large variations across the set of face images, while later Eigenfaces capture more subtle variations. To see this, click on the **Generate Faces** button. On the left side of the window, two images will appear at the top, randomly selected from the dataset. The first Eigenface will appear in the middle and the **Add Eigenface** button will become active. At the bottom, two copies of the *average face* computed for the entire dataset will appear. You can now click repeatedly on the **Add Eigenface** button, and on each button press, an additional Eigenface will be added to the bottom images to create composite faces that eventually look like the two individual faces at the

top. For each person, the specific weight associated with the next Eigenface is printed on the left or right side of the Eigenface image, and this weight is used to construct the composite face image at the bottom. Click on the **Generate Faces** button again to view the construction of a new pair of faces.

- **Q5.** How many Eigenfaces are typically needed to start distinguishing the two faces? How many Eigenfaces are needed to create an identifiable version of each face?

To see how the Eigenfaces representation can be used to recognize a new face, we'll first consider the problem posed in the picture shown below (from the last slide posted for Class 5). Imagine that there are only three people that the system can recognize, which are highlighted with a red outline in the figure below. The weights associated with the first two Eigenfaces for these three faces are shown in parentheses (in red) next to each face image. Given a new face such as the one shown in the upper left corner (highlighted in blue), which is assumed to be one of the known individuals, the PCA method can compute weights for the two Eigenfaces that can be used to encode this new face (weights shown in blue).

- **Q6.** Describe a strategy for determining the identity of this new face, and use your strategy to determine which of the three known people is the right identity for this face.



In general, we have a choice about how many Eigenfaces are used to encode each face image, which also means that there's a choice about how many numbers we use to encode each face.

Suppose we decide to keep 6 Eigenfaces, and therefore encode each face with 6 weights, one for each Eigenface. You can generalize your recognition strategy to handle additional weights. To see how this can be done, imagine again that there are three people who can be recognized by the system. In the table below, there are 6 weights listed for each of these three people, associated with the first 6 Eigenfaces.

	E1	**E2**	**E3**	**E4**	**E5**	**E6**
Mike	20	14	-10	5	-6	-4
Ellen	-10	18	-3	10	-1	5
Isabel	14	-10	7	-3	2	5

Imagine that we are given a new face image to recognize and determine that the weights that the PCA method computes for this message are:

	E1	**E2**	**E3**	**E4**	**E5**	**E6**
Mystery	15	-8	10	1	-3	6

We can still use the general strategy of finding which of the three known individuals has a weight code that is *closest* to the set of weights for the mystery face, but now distance is defined in 6 dimensions! Fortunately MATLAB provides a function that we can use to determine the distance between two points in an n-dimensional space, **norm**. In the simple example below, the norm function is used to calculate the distance between two points in a three-dimensional space:

```
>> point1 = [20  -5  12];
>> point2 = [10  2  -3];
>> distance = norm(point1 - point2)
distance =
        19.3391
```

The expression (point1 - point2) defines a line, or vector, between the two points in 3D space, and norm then determines the length of this line.

- **Q7.** Use MATLAB, the norm function, and the weights provided in the above tables, to recognize the identity of the mystery person. Copy/paste your MATLAB code here (you can either create a script or just enter commands into the Command Window) and your final answer.

A key test for a face recognition method is how well it can generalize beyond the particular face images used to construct the system. What happens, for example, if a face is viewed under different lighting, or with a different expression, or appears at a different orientation in the image, relative to previously viewed faces? Return to the interactive facesGUI program in MATLAB.

From the pull-down menu labeled **Test Set**, select a particular set of new test images to be recognized. Each test set consists of 30 images that can be viewed by clicking on the **View Test Faces** button. If you click on the **Run Test Faces** button, the performance of the system on this test set will be printed in the textbox below the button. The statement **Accuracy = 43.3%**, for example, means that the system only recognized 43.3% of the test faces correctly (13 of the 30 test images in this case). The performance of the system also depends on the total number of Eigenfaces, and hence the total number of weights, used to encode each face. This number is initialized to 25, but you can change this number in the textbox labeled **Number of Eigenfaces**.

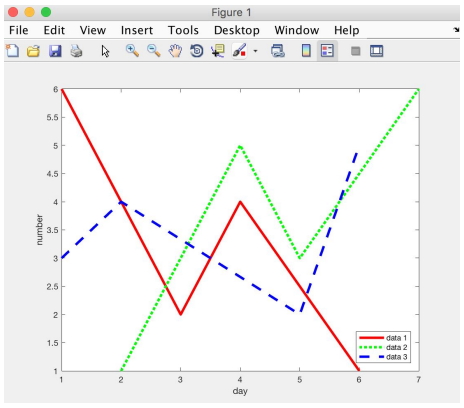
- **Q8.** In the table below, record the performance (Accuracy) of the system for each of the three test conditions, and for 5, 10, 15, 20, and 25 Eigenfaces.

	5	**10**	**15**	**20**	**25**
new lighting					
new expression					
new orientation					

- **Q9.** Write a script in the MATLAB Editor that creates a figure with three plots showing the performance for new lighting, expression, and orientation, as a function of the number of Eigenfaces used. The sample code on the next page below shows how to use the **hold on** command to combine multiple plots, and the **legend** command to add a key to the figure. **Copy and paste your code and final figure into this file.**

```
% create a simple figure with three plots

plot([1 3 4 6], [6 2 4 1], 'r', 'LineWidth', 3)
hold on
plot([2 4 5 7], [1 5 3 6], 'g:', 'LineWidth', 3)
plot([1 2 5 6], [3 4 2 5], 'b--', 'LineWidth', 3)
hold off
legend({'data 1', 'data 2', 'data 3'}, 'Location', 'SouthEast')
xlabel('day')
ylabel('number')
```



- **Q10.** How well does this face recognition method generalize to these new conditions that are not contained in the original dataset used to construct the Eigenfaces representation? How do the results depend on the number of Eigenfaces used? **Challenge question:** *why* do you think this method performs so poorly for one of these general conditions?

References

Turk, M & Pentland, A. (1991) Eigenfaces for Recognition, Journal of Cognitive Neuroscience, 3(1), 71-86.