MSSC 6040 - Homework 3

Instructor: Greg Ongie

Fall 2022

Problem 1 (5 pts)**.** T-B 4.1 (Hint: You do *not* need to compute any eigenvalues/vectors so find these SVDs. Instead, try and find the U, Σ, V matrices directly by expressing A as a product of elementary matrices, or as an outer product.)

Problem 2 (5 pts, MATLAB)**.** T-B 4.3. You may use MATLAB's `svd` command to find the left and right singular vectors. Submit a printout/screenshot of your code as well as the plots showing the singular vectors for the indicated examples.

Problem 3 (5 pts)**.** T-B 5.3 parts (c)-(g) only. You may use the following SVD of A:

$$A = U\Sigma V^* = \begin{bmatrix} -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \sqrt{200} & 0 \\ 0 & \frac{\sqrt{200}}{2} \end{bmatrix} \begin{bmatrix} \frac{3}{5} & -\frac{4}{5} \\ -\frac{4}{5} & -\frac{3}{5} \end{bmatrix}$$

For part (e), find the eigenvalues using the characteristic equation $\det(A - \lambda I) = 0$.

Problem 4 (10 pts, MATLAB)**.** For this problem, you will modify the provided MATLAB script `problem4.m` to create a face detection system using Eigenfaces.

(a) Your first task is to define a MATLAB function in the file `facedist.m` that takes in a 32×32 pixel image and measures its distance from "face space"; the steps needed to do this are detailed below. The function has three inputs and one output: `dist = facedist(x,mu,U)`. Here `x` is an image patch given as a 32×32 array, `mu` is the mean face image given as a 1024×1 column vector, and `U` is the 1024×400 matrix whose columns are the ordered Eigenfaces. The output `dist` is the Euclidean distance of the image to "face space" (i.e., the function returns a small value if the image is a face, and a large value if the image is not a face). Your function should consist of the following steps:

1. Vectorize the input image.
2. De-mean the input by subtracting the mean face image from the vectorized input image.
3. Project the de-meaned input onto the "face space" described by the span of the top 16 Eigenfaces.

4. Finally, compute the 2-norm of the difference between the de-meaned input image and its projection onto face space as computed in step 3.

(b) Test out your `facedist` function by applying it to the face images saved in the files `greg.mat` and `nic_cage.mat`, and a random image `x = rand(32,32)`. Is the output consistent with your intuition?

(c) Now run the code block in the `problem4.m` script labelled `%%Part (c), step 1`. This will compute a "distance image" for the Friends image, which computes the distance to face space of every 32×32 pixel patch in the image. (Warning: This code block may take a long time to run if you coded "facedist" in an inefficient way, but an efficient implementation should only take a few seconds). In the next code block labelled `%%Part (c), step 2`, find a threshold `thresh` to apply to the distance image such that at least five of the six faces are recognized as faces, with no false positives. Can you do better than five out of six?

In your write-up, include a print-out/screenshot of your `facedist.m` function from part (a), the results of the tests in part (b), and a picture showing where the faces were detected on the Friends image in part (c). Finally, based on your observations of how well this system performed on the Friends image, discuss the potential limitations of this system for face detection in general, and how one might improve it.