EAS 596, Fall 2019, Homework 6

Due Friday 11/15, **3:30 PM**, Box outside Jarvis 326

Work all problems. Show all work, including any M-files you have written or adapted. Make sure your work is clear and readable - if the TA cannot read what you've written, they will not grade it. All electronic work (m-files, etc.) **must** be submitted through UBlearns and obey the following naming convention: `ubitname_hw6_pN.m`, replacing ubitname with your ubitname and `N` with the problem number. Any handwritten work may be submitted in class.

All two point problems will be graded according to the following scheme:

- 2 Points: Solution is complete and correct.
- 1 Points: Solution is incomplete or incorrect, but was using correct ideas and concepts.
- 0 Points: Using incorrect ideas and concepts.

All four point problems will be graded according to the following scheme:

- 4 Points: Solutions are complete and correct. Code runs with no need for modification.
- 3 Points: One mistake in the code and it is easily found. Code runs after the modification.
- 2 Points: Two to three minor mistakes in the code, which are easily found. Code runs after the modification.
- 1 Points: Many mistakes in the code. No attempt will be made to modify it to run.
- 0 Points: Code has major conceptual issues.

1. (2 pts) It was shown previously that solving the Normal Equations, $\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b}$, results in the least-squares solutions for $\boldsymbol{x}$. Show that solving the systems using the QR-Decomposition and SVD-Decomposition both result in the least squares solution to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. This problem must be submitted via hardcopy.

2. (2 pts) Determine the full SVD of the following matrix:

$$A = \begin{bmatrix} 0 & 1 & 1 \\ \sqrt{2} & 2 & 0 \\ 0 & 1 & 1 \end{bmatrix}.$$

Use the SVD to determine the rank of the matrix. You must compute the SVD by hand but you can verify it using the `svd` command in MATLAB. This problem must be submitted via hardcopy.

3. (4 pts) Write a MATLAB function `B = ubitname_hw6_p3(A, p)` that accepts a matrix `A` and returns the rank-`p` matrix `B` that is the best approximation to `A` in the 2-norm. If the requested rank `p` is larger than Rank(`A`) simply return matrix `A`. You may use the MATLAB functions `svd`, `diag`, `rank`, `zeros`, and `size` to produce the result. You may not use any other built-in MATLAB function. Note that commands like `if`, `for`, etc are not considered MATLAB functions. NOTE: You may consider that any singular value below 10^{-14} as zero. Please upload your code to UBlearns. This problem does not require a hardcopy submission.

4. Bitmap images, such as jpg or png files, are nothing but matrices with a single number at each pixel location indicating the intensity of a particular color. If the image is an 8-bit gray-scale image then one single matrix will suffice, with a value of 0 indicating black and 255 indicating white. Values between 0 and 255 indicate how gray the color at that pixel is to be. Color images are typically composed of three such matrices: one for red, one for green, and one for blue. Using the intensity of each of these three colors allows for other colors, such as purple, to be shown at a pixel. As with the gray-scale images the scalar values for each of these colors ranges from 0 to 255 for 8-bit images.

   Recall from class that any matrix can be decomposed into the summation of rank-1 matrices:

   $$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \boldsymbol{u}_1\sigma_1\boldsymbol{v}_1^T + \boldsymbol{u}_2\sigma_2\boldsymbol{v}_2^T + \cdots + \boldsymbol{u}_r\sigma_r\boldsymbol{v}_r^T,$$

   where σ is a singular value, $\boldsymbol{u}$ and $\boldsymbol{v}$ are singular vectors, and r is the rank of matrix $\boldsymbol{A}$.

   To compress an image you simply truncate the sum at the p^{th} singular value:

   $$\boldsymbol{A} \approx \boldsymbol{u}_1\sigma_1\boldsymbol{v}_1^T + \boldsymbol{u}_2\sigma_2\boldsymbol{v}_2^T + \cdots + \boldsymbol{u}_p\sigma_p\boldsymbol{v}_p^T,$$

   where $p < r$. What this means is that instead of storing the entire image matrix, we only need to store a small number of vectors and their associated singular values to create an approximation of an image.

In this problem, we will use MATLAB to study image compression using the SVD.

(a) (4 pts) Write a MATLAB function `img = ubitname_hw6_p3(f, p)` that takes in a file name `f` and approximation rank `p` and returns the rank-p approximate image `img`. You will need to use MATLAB's `imread` function and then convert the image array from integers to doubles using the `double` function. Also note you need to do each of the three color arrays separately. The function should return the compressed image as an $m \times n \times 3$ array (the same dimensions as the array generated by the `imread` function. Please upload your code to UBlearns. This problem does not require a hardcopy submission.

(b) (4 pts) Write a MATLAB function `ubitname_hw6_p3b(f, pArray)` that takes in a file name `f` and an array of `p` values. For each of the values in `pArrary` produce the rank-`p` approximate image using your prior function. Show the original image along with the rank-`p` approximate images on a single figure using the `imshow` and `subplot` MATLAB commands. Properly title each image using the `title` MATLAB command. An example of what is expected is shown below. Please upload your code to UBlearns. This problem does not require a hardcopy submission.

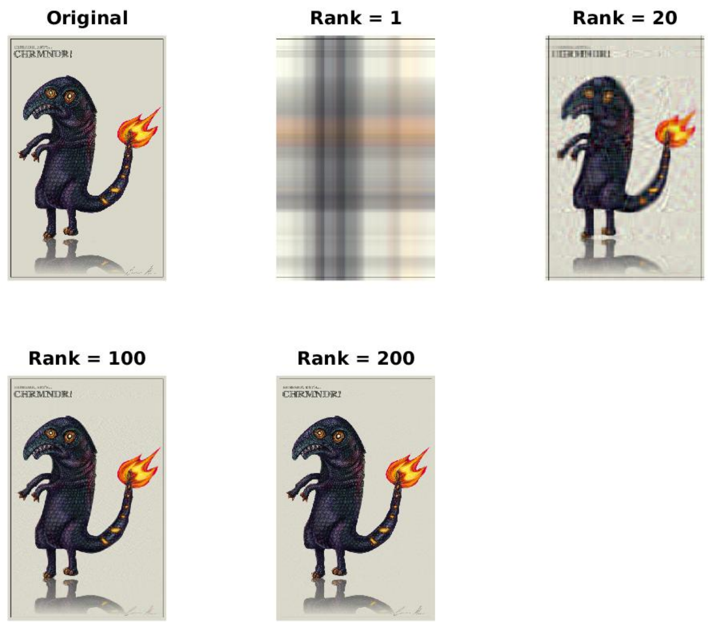

Figure 1: Example output of image compression comparison.

(c) (2 pts) Write a MATLAB script that uses the previous function to perform image compression on each of the 3 images included with the assignment: `square.png`, `UB.png`, and `futurama.png`. What is the minimum rank you feel gives a good quality image for each one? What is your reasoning? Show at least five different low-rank approximations for each image, including the minimum rank approximation you feel gives a good quality image and a higher rank approximation than the minimum one. Please upload your code to UBlearns. On hardcopy indicate the minimum rank approximation which gives a good quality image and the compression ratio for each of the images.