Programming Project for Math 2605

This project is aimed at developing a understanding of low rank approximation using the singular value decomposition. The matrices we will work with come from images.

Consider an image that consists of a rectangle of 200 by 300 pixels. This is a total of $60,000$ pixels. If the image is a standard grayscale image, each pixel will be assigned an integer value between 0 (representing black) and 255 (representing white). The image file represents a large – 200×300 – matrix with integer values in this range.

Images – at least ones we would want to look at – have a lot of structure to them. Because of the structure, the information in them can often be conveyed very accurately by specifying far fewer numbers that the values for each and every pixel. Here is one way to do this: Let $A = VDU^t$ be a singular value decomposition of an $m \times n$ A. As usual, put

$$V = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r] \qquad \text{and} \qquad U = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r] \ .$$

For any integer k with $0 < k < r$, put

$$A^{(k)} = \sum_{j=1}^{k} \sigma_j \mathbf{v}_j \mathbf{u}_j^t \ . \qquad (*)$$

Although $A^{(k)}$ is an $m \times n$ matrix, just like A, the data needed to specify A is just the data needed to specify the k numbers, the k vectors in R^m, and the k vectors in R^n that show up in (*). This would be $k(1 + m + n)$, which is *much* less than $m \times n$ if m and n are large.

As we have seen in class, for every pair of indices i and j,

$$\left| A_{i,j} - A_{i,j}^{(k)} \right| < \sigma_{k+1} \ .$$

Therefore, if we choose k large enough that $\sigma_{k+1} < 1$, the entries of $A^{(k)}$ will be within one grayscale value of the entries of A, and so the images that the two matrices represent will be very close. Even if σ_{k+1} is not *so* small, this will often be the case.

In any case, the fact that the matrix A comes from a *visually interesting image*, and is not just a random matrix, gives it *interesting structure* that results in a rapid drop off in size of the singular values. Most of the information in an *image matrix* is contained in the few largest singular values, and their corresponding left and right singular vectors. This project explores these ideas.

(1) Write a program that read in in an image file, and converts it into a matrix; i.e., an array of arrays. There is information on standard image file formats, and suggestions for how to do this on the main projects page.

Also write a program that takes a given $m \times n$ matrix with floating point entries, and converts it to an image file (using whatever format you chose in the first part). Be sure you round of and truncate the values properly! For example, if your file format requires each pixel to be assigned a integer value in the range from 0 to 255, make sure your program does the something reasonable with entries that are negative or too large!

If you work in color, you will need one matrix per channel. But each will be handled the same way, so this is not much of a complication.

These steps are not so mathematical, so you are free to collaborate and seek expert advice on this part, if you wish. As long as you indicate and credit this, you will not loose any points.

(2) Write a program that takes as input an integer k, and an $m \times n$ matrix A, and returns $A^{(k)}$, σ_1, **and** σ_{k+1}. You may use packages, such as JAMA, for doing the basic linear algebra here.

(3) Put you programs to work: Choose an image file or two in the right format, and convert the image to a matrix. Find the best rank k approximation for $k = 2$, $k = 5$, $k = 10$. Convert these back into images, and print them out, together with the original image. Beside each image, give the values of σ_1 and σ_k. Try this for a few more images. (Once you have coded it up, you may as well use it!).

Question: To get a good easily recognizable image, do you need to have σ_{k+1} small compares to 255, or just small compared to σ_1, or is there some other criterion of smallness that is even more relevant?

Try this compression with portraits of faces. How small can you make k, and still keep the portrait recognizable? In other words (roughly), what is the rank of a human face?

Extra Credit: Organize your programs into a *codec*; that is a compression–decompression package. Invent a file format for storing the k numbers, the k vectors in R^m, and the k vectors in R^n that show up in (*). For example, with whitespace separation, the first three entries could be k, m, and n. The next k could be the singular values, and so forth. Have your program read in an image file and a specified number k, and write out a file in your format. This will be the compressed file.

Then write a program that reads in one of your compressed files, and writes it back to a standard image format. This is the decompression part.

For even more extra credit, after experimenting with how the relative sizes of the various singular values determine a "good" value of k, write a program that chooses k itself, depending on the image.