

Your name: _____

Project 3: SVD Image Compression

In class, we used the SVD to compress an image. We treated the image as a matrix of grayscale values; call it A . Then the SVD gives that $A = USV^T$; we construct a compressed version by using the first k columns of U and V , in the matrices U_k and V_k respectively, and the first k singular values only in S_k ; the compressed image is then $A_k = U_k S_k V_k^T$. This is the “best” approximation to A in the sense that there $\|A_k - A\|$ has the smallest possible matrices of rank k or less. This optimality is true in the 2-norm, and in another norm that serves us better here, the Frobenius norm:

$$\|X\|_F = \left(\sum_{i=1}^m \sum_{j=1}^n |x_{ij}|^2 \right)^{1/2}. \quad (1)$$

Unlike our other matrix norms, the Frobenius norm is not induced by a vector norm. However, in the image context there is a relationship between the 2-norm and the Frobenius norm: if Z is an image in matrix form and \mathbf{z} is its vector form, then $\|Z\|_F = \|\mathbf{z}\|_2$. The Frobenius norm is also closely related to the SVD of A and the formulation of A_k , via

$$\|A\|_F^2 = \sigma_1^2 + \sigma_2^2 + \cdots + \sigma_n^2, \quad (2)$$

$$\|A_k\|_F^2 = \sigma_1^2 + \sigma_2^2 + \cdots + \sigma_k^2. \quad (3)$$

As we saw in class and the homework, the low-rank approximation to A_k is not necessarily a good way to compress an image. In this project, we investigate the benefits of breaking the image into smaller blocks, compressing each of the blocks separately via the SVD, and then recombining them back into a single image.

1. Write a function `k = cutoff(sigma,q)`. Here `sigma` is a vector of n singular values and `q` ≥ 0 is a number that is used to control the quality of the compression of the block. The value `k` that is returned should be the smallest possible `k` such that

$$\frac{\sigma_1^2 + \cdots + \sigma_k^2}{\sigma_1^2 + \cdots + \sigma_n^2} \geq 1 - 2^{-q}. \quad (4)$$

Typical values for `q` will be 4,8,12,16 in later parts of this project. The builtin function `cumsum`, which computes the cumulative sum of a vector, is handy in this context.

2. Write a function `[Z,ratio] = svdcompress(X,b,q)` that performs SVD compression. The grayscale image `X` is divided into nonoverlapping $b \times b$ blocks. For example, the first block is `X(1:b,1:b)`. A block A should have its SVD computed. The singular values and `q` are passed to the `cutoff` function to determine the appropriate value of `k` for the block. The corresponding approximation A_k will occupy the same position in `Z` as A does in `X`.

The number of scalars needed to define A_k is $k(2b + 1)$. Therefore, if $k \geq b/2$, you should use A rather than A_k in the output `Z`, as it will be more efficient. In addition, along the bottom

and right edges of the image there may be blocks whose width or height is smaller than b . Those can be copied directly into the output Z .

You should also keep a running total of the number of values needed to reconstruct Z (that is, either $k(2b + 1)$ or the number of elements in the block). At the end of the function, divide this total by the total number of elements in X to get the output value `ratio`, which represents the compression ratio achieved. It should be no greater than unity (one) in all cases.

3. Apply your function `svdcompress` to the eye image from your second project. Try block sizes of 5, 10, 20, and 40, with `q=2`. Display the resulting images and report the compression ratio in each case. If you require a better image, try your own photo of an eye or image with similar level of complexity.
4. Apply the function `svdcompress` to your image, with the block sizes 10 and 20, and use cutoff thresholds `q` of 2, 4, 8 and 12. Again, display the images and report the compression ratios.
5. Write a *brief* report summarizing your results, and attach your images, appropriately labeled, and attach hardcopy of your Matlab functions. Attach this sheet as cover for your report.

Due Date: Wednesday November 26, 2008, 3pm.