Lab 3: Data Compression in Image Processing using SVD

(Due on October 20, 2023)

Goals

The goal of this lab is to demonstrate how the SVD can be used to remove redundancies in data; in this example, we will be compressing image data. We will see that for a matrix of rank r, the SVD can give a rank $p < r$ approximation to a matrix and that this approximation is the one that minimizes the Frobenius norm of the error.

Introduction

Any image from a digital camera, scanner or in a computer is a digital image. The "real world" color image is digitized by converting the images to numerical data. A pixel is the smallest element of the digital image. For example, a 3-megapixel camera has a grid of 2048×1536 = 3,145,828 pixels. Since the size of a digitized image is dimensioned in pixels of say m rows and n columns, it is easy for us to think of the image as an $m \times n$ matrix. However, each pixel of a color image has an RGB values (red, green, blue) which is represented by three numbers. The composite of the three RGB values creates the final color for the single pixel. Therefore, we can think of each entry in the $m \times n$ matrix as having three numerical values stored in that location, i.e., an $m \times n \times 3$ matrix.

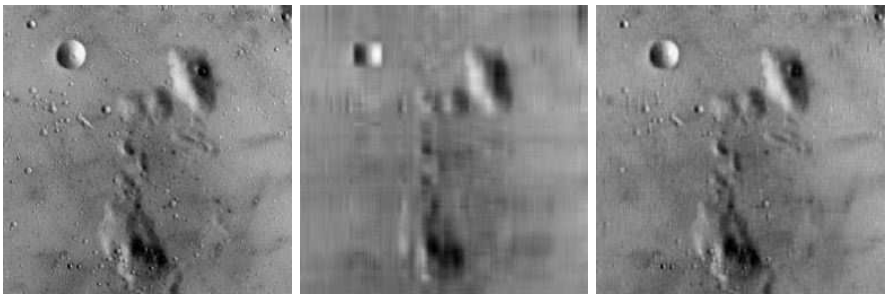

Figure 1: The image on the left is the original image while the other two images represent the results of data compression.

Now suppose we have a digital image taken with a 3-megapixel camera and each color pixel is determined by a 24-bit number (8 bits for intensity of red, blue and green). Then the information we have is roughly 3×106×24. However, when we print the picture suppose we only use 8-bit colors giving 28=256 colors. We are still using 3 million pixels but the information used to describe the image has been reduced to 3×106×8, i.e., a reduction of one-third. This is an example of image compression. In the figure below, we give a grayscale image of the moon's surface (the figure on the left) along with two different compressed images. Clearly, the center image is not acceptable but the compressed image on the right has most of the critical information. We want to investigate using the SVD for doing data compression in image processing.

Understanding the SVD

Recall from the notes that the SVD is related to the familiar result that any $n \times n$ real symmetric matrix can be made orthogonally similar to a diagonal matrix which gives us the decomposition $A = Q\Lambda Q^T$ where Q is orthogonal and Λ is a diagonal matrix containing the eigenvalues of A.

The SVD provides an analogous result for a general non-symmetric, rectangular $m \times n$ matrix A. For the general result, two different orthogonal matrices are needed for the decomposition. We have that A can be written as

$$A = U\Sigma V^T$$

where Σ is an $m \times n$ diagonal matrix and U, V are orthogonal.

Recall from the notes that the SVD of an $m \times n$ matrix A can also be viewed as writing A as the sum of rank one matrices. Recall that for any nonzero vectors x, y the outer product xy^T is a rank one matrix since each row of the resulting matrix is a multiple of the other rows. The product of two matrices can be written in terms of outer products. In particular, the product BC^T can be written as the sum of the outer products $\sum_i b_i c_i^T$ where b_i and c_i denote the i-th columns of B and C respectively. Hence, if we let u_i denote the i-th column of U and v_i the i-th column of V then writing the decomposition $A = U\Sigma V^T$ as the sum of outer products we get

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^T$$

where r denotes the rank of A. Thus our expression for A is the sum of rank one matrices. If we truncate this series after p terms, then we have an approximation to A, which has rank p. What is amazing, is that it can be shown that this rank p matrix is the best rank p approximation to A measured in the Frobenius norm. Frobenius norm is just the matrix analogue of the standard Euclidean length, i.e.,

$$\|A\|_F = \left(\sum_{ij} A_{ij}^2 \right)^{\frac{1}{2}}$$

where A_{ij} denotes the (i, j) entry of the matrix A. Note it is NOT an induced matrix norm. We can get a result for the error that is made by approximating a rank r matrix by a rank p approximation. Clearly, we have that the error E_p is given by

$$E_p = A - \sum_{i=1}^{p} \sigma_i u_i v_i^T = \sum_{i=p+1}^{r} \sigma_i u_i v_i^T.$$

Due to the orthogonality of U and V, we can write

$$\|E_p\|_F^2 = \sum_{i=p+1}^{r} \sigma_i^2$$

and so a relative error measure can be computed from

$$\left[\frac{\sum_{i=p+1}^{r} \sigma_i^2}{\sum_{i=1}^{r} \sigma_i^2}\right]^{\frac{1}{2}}$$

Data compression using the SVD

How can the SVD help us with our data compression problem? Remember that we can view our digitized image as an array of $m \times n$ values and we want to find an approximation that captures the most significant features of the data. Recall that the rank of an $m \times n$ matrix A tells us the number of linearly independent columns of A; this is essentially a measure of its non-redundancy. If the rank of A is small compared with n, then there is a lot of redundancy in the information. We would expect that an image with large-scale features would possess redundancy in the columns or rows of pixels and so we would expect to be able to represent it with less information.

For example, suppose we have the simple case where the rank of A is one; i.e., every column of A is a multiple of a single basis vector, say u. Then if the i-th column of A is denoted a_i and $a_i = k_i u$ where k_i is a scalar number, then $A = uk^T$ with $k = (k_1, k_2, \ldots, k_n)$. If we can represent A by a rank-one matrix then all we need to specify are the vectors u and k; that is $m + n$ entries as opposed to mn entries for the full matrix A. If we choose the Frobenius norm to measure the error, then the SVD decomposition of A will give us the desired result.

Of course, in most applications the original matrix A is of higher rank than one, so that a rank-one approximation would be very crude. In general, we seek a rank p approximation to A such that $\left\|A - \sum_{i=1}^{p} \sigma_i u_i v_i^T\right\|_F$ is minimized. It is important to remember that the singular values given in Σ are ordered so that they are non-increasing. Consequently, if the singular values decrease rapidly, then we would expect that fewer terms in the expansion of A in terms of rank-one matrices would be needed.

Computational Algorithms

In this lab, we will treat the software for the SVD as a "black box" and assume that the results are accurate. We will do image compression using Python. We will use Python commands numpy.linalg.svd() to do SVD.

A test image library for use in image compression is maintained by the electrical engineering department at the University of Southern California and the images for this lab were obtained from there. In addition to the SVD algorithm, we will need routines to generate the image chart (i.e., our matrix) from an image and to generate an image from our approximation. There are various ways to do this. One of the simplest approaches is to use the commands in *matplotlib*:

- matplotlib.pyplot.imread() - reads an image from a graphics file

- matplotlib.pyplot.imshow() - displays an image. To show grayscale image, add cmap='gray' parameter as imshow(my_fig, cmap='gray')

Task 1. The purpose of this problem is to make sure that you are using the SVD algorithm correctly. First, make sure that you get the SVD of

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix}$$

as

$$A = \begin{pmatrix} .1409 & -.8247 & -.5456 & .0486 \\ .3439 & -.4263 & .6912 & -.4714 \\ .5470 & -.0278 & .2543 & .7971 \\ .7501 & .3706 & -.3999 & -.3743 \end{pmatrix} \begin{pmatrix} 25.46 & 0 & 0 \\ 0 & 1.291 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} .5045 & .7608 & 4082 \\ .5745 & -.05714 & .8165 \\ .6445 & -.6465 & .4082 \end{pmatrix}^T$$

Next, compute a rank-1 approximate to *A* and determine the error in the Frobenius norm by (i) calculating the difference of *A* and its approximation and then computing its Frobenius norm and (ii) using the singular values. For this problem just turn in your rank-1 approximation and your error estimates.

Task 2. In this problem, we will be using the grayscale image *boat.tiff*, which is on Canvas. Create an integer matrix chart representing this image. Your image chart should be a 512 × 512 matrix with entries between 0 and 255. View your image with imshow() to make sure you have read it in correctly.

a) Write code to create the matrix chart for an image and to do a SVD decomposition; you should read in the rank of the approximation you are using as well as the image file name. Use your code to determine and plot the first 150 singular values for the SVD of this image. What do the singular values imply about the number of terms we need to approximate the image?

b) Modify your code to determine approximations to the image using rank 8, 16, 32, 64 and 128 approximations. Display your results as images along with the original image and discuss the various quality of the images.

c) Now suppose we want to determine a reduced rank approximation of our image so that the relative error (measured in the Frobenius norm) is no more than 0.5%. Determine the rank of such an approximation and display your approximation. Compare the storage required for this data compression with the full image storage.

Task 3. In this problem, we will use the color image *mandrill.tiff*. Now each pixel in the image is represented by three RGB values and so the output of imread() is a three dimensional array. You will compress each R, G, and B matrix separately, and then combine them together to obtain the compressed image.

a) Plot the first 150 singular values of the R, G, and B matrices, and discuss the implications. For example, do the singular values decay fast?

b) Obtain rank 8, 16, 32, 64 and 128 approximations to your image. Display and compare your results.