Applying Singular Value Decompositions for Image Compression

Introduction

Originally, computer graphics were kept in TIFF (Tagged Image File Format) files. These files did not have any compression and were originally for black and white screens, or CGA. As monitors advanced, the need to have an RGB coded file came in, meaning instead of one or two bits per pixel, image files were dealing with 3x8 bit or at present day 3x24 bits per pixel. On a 100x100 picture that is equivalent to 90 Megs of storage uncompressed. Clearly, image compression was needed. CompuServe came up with the GIF file format to compress graphics files, and JPEG has gained wide spread use, with JPEG-2000 as a recent extension.

Although humans do not need more than 8 bits per pixel to view gray scale images, or 15 bits for color (The human eye can ascertain about 30K colors), computer vision can analyze data of much higher intensity resolutions. Therefore, higher resolution applications require higher intensity resolution (more bits per pixel), meaning the need for image compression is increasing. Medical imaging is a prime example of images increasing in both spatial resolution and intensity resolution.

Compression rate is simply the size of the original data divided by the size of the compressed data. A technique that compresses a 1 Megabyte image to 100 kilobytes has achieved a compression rate of 0.1 (which is average for something like JPEG). For a given image, the smaller the compression rate, the smaller the final image file will be.

compression rate = compressed data size / original data size = 100 k B / 1 M B = 0.1

There are two basic types of image compression: lossless compression and lossy compression. A lossless scheme (LZW or Lempel-Ziv-Welsh, think pkzip) encodes and decodes the data perfectly, and the resulting image matches the original image exactly. There is no degradation in the process, i.e., no data is lost. Lossy compression schemes are designed to remove redundant information, thus removing to compress that data. A typical natural scene has a great degree of redundancy and almost never uses the full scale of possible intensities. The average contrast of the natural world is about 30% when seen through the photoreceptor array of the human fovea and less when scene through larger apertures. At the cochlea, the visual system is producing several billion bytes of information per second! While only a small fraction makes its way to the visual cortex. One can say that the eye produces a lossy compression, that leads to the counterintuitive statement that lossy schemes can provide a very small compression rate if they work under the assumptions about the human vision system, without information loss.

Typically, with lossy schemes, there is a tradeoff between compression and image quality. One may be able to provide a small compression rate but the image looks so poor that no one will use it, or so much information is lost that no processing algorithm can

operate on it. Lossy compression techniques are typically more complex and computationally intensive. Images with a high degree of detail that can't be lost, such as medical images or CAD drawings, cannot be compressed with lossy algorithms.

Lossy Quality Measures

The best way to judge an images quality after lossy compression is still a human observer to judge the quality. Since that cannot be done with every picture one needs an engineering measure of merit. The most abundant measure of merit in engineering terms is the Signal-to-Noise Ratio (SNR) in power terms. This can have many different names, and change guises such as the Mean-Square-Error (MSE) or variance ratios, but they all mean the same thing. For a grey scale image the Frobenius Norm can represent the pixel by pixel power.

Given a matrix A, which is [M x N]. The Frobenius Norm is:

$$\|A\|_{Fro} = \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N}\left|A_{ij}\right|^2}$$

If B is the compressed image, another [M x N] matrix, then the residual percentage error will be:

$$E = \frac{\|A-B\|_{Fro}}{\|A\|_{Fro}}$$

Now on to the work:

Part I: use SVD for image compression

You will need to download the .bmp and .jpg files. For each of the files,

1. Take the SVD of the input picture.
2. Plot the Singular values on one graph. (provide graph)
3. Recompose the input picture with the SVD rank approximations {1, 2, 4, 16, 32, 64, 128, 220} (provide graphs of them all, use subplot to limit the number of pages)
4. Calculate the residual percentage energy error with the Frobenius norm.
5. Calculate the compression rate of the approximation. Since the input is 220x220 it needs 220^2 units of space. The SVD takes only a subset of the U,S,V matrices, count the elements per rank approximation that are necessary.
6. Repeat the above steps with the Input being a 220x220 matrix whose elements are picked from the uniform distribution between 0 and 100.

Observations:

Compression plays a large part in Optical Character Recognition (OCR). Usually, as much information as possible is thrown away so the OCR program isn't confused by minutia.

1. For the bmp-format and jpg-format McMaster logo,

    a. at which rank approximation can you make out what University?
    b. How quickly does the wording come out as opposed to the crest?
2. For bmp-format and jpg-format square picture, repeat steps 1.a-1.b.
3. For the randomly generated picture, repeat steps 1.a-1.b.

Discussions:

1. The singular values drop much faster in some pictures than in other ones. What features of the picture or image file may contribute to this feature?
2. For each picture, what is the minimum number of ranks needed to recover a reasonably good quality of the picture after SVD approximation? How is this number related to the percentage of energy left over in the approximations?
3. How efficient is the SVD compression for different types of images? Answer the question by comparing jpeg vs. bmp format images, natural vs. random pictures, and simple (e.g., the square picture) vs. more complicated pictures (e.g., the weave picture).

Report:

1. Include all the results, codes, and discussions in your report.
2. Use grayscale only (or black-and-white) to display all images.