Image Compression Using SVD Methods

SX1916115

December 7, 2019

Abstract

SVD (Singular Value Decomposition) is one of the most important decomposition methods for matrix and is wildly used in signal processing and statistics. In this project, I'd like to analyze an application of SVD in image compression in computer science aspect.

1 Background

1.1 Images Represented in Computers

Inside the computer, the images are represented by pixels. For a two-dimension image, the pixels are organized as an two-dimension array. For a basic greyscale image, each pixel's value is an 8-bit integer ranging from 0-255, which describes how black the pixel is.

For a colored image, each pixel needs more room to be represented. Generally, each pixel contains three color dimensions: red (R), green (G) and blue (B), each dimension is represented in an 8-bit integer. So, a colored pixel needs a storage of 24-bit. For an image with resolution of 1920 * 1080, it need a storage of 5.93MB.

1.2 Image Compression

Internet is becoming a necessary part of our daily life, within which multi-media traffic makes up the main part of the data traffic, through the sharing of photos and videos in social-network. Since the bandwidth of network is limited, users hope the transferring process to be faster, so that they will not be waiting for so long. However, the service provider cannot guarantee the stability of user's network speed, a common way is to reduce the amount of data transferred through compressing the images. By compressing the image, sending it through the network, and decompressing the image at client side, the service providers can reduce the data transferred at the cost of a little bit more CPU computing resource, which is acceptable.

In another case, for a photo which is taken by a professional photographer with his advanced camera, the resolution can be up to 3840×2160 pixels. However, for a mobile phone user with a 720p screen, say, with resolution of 1280×720 pixels, it is not necessary to display such a clear image. As a result, a compression is also needed.

Generally, there are two categories of methods of image compression, based on whether there will be a loss of data after compression. But the goal of compression is the same: to represent an image with less data. SVD is one of the methods for compressing an image.

1.3 SVD

In linear algebra, the singular value decomposition (SVD) is a factorization of a real or complex matrix. It is the generalization of the eigen decomposition of

a positive semidefinite normal matrix (for example, a symmetric matrix with non-negative eigenvalues) to any $m \times n$ matrix via an extension of the polar decomposition. It has many useful applications in signal processing and statistics.

In detail, Suppose M is an $m \times n$ matrix whose entries come from the field K, which is either the field of real numbers or the field of complex numbers. Then the singular value decomposition of M exists, and is a factorization of the form: $M = U\Sigma V^{\mathrm{H}}$, where U is an $m \times m$ unitary matrix over k. Σ is a diagonal $m \times n$ matrix with non-negative real numbers on the diagonal, V is an $n \times n$ unitary matrix over K, and V^{H} is the conjugate transpose of V.

Specifically, the column vectors of U and V are two standard orthogonal basis, respectively. And each diagonal element σ in Σ represents the stretch relationship. Bigger the diagonal element, more important the dimension is.

2 Motivation

Consider a real case: suppose we have an HD image with a resolution of 1920×1080 pixels. The image is represented in the memory in a two-dimension array (a matrix). If I want to transfer this image through network without compression, we need to send $1920 \times 1020 \times 8$ bytes of data, supposing the image is represented in the standard RGB-form. We decompose the matrix of the image M of 1920×1080 with SVD method, where U is a 1920×1920 matrix, V^{H} is a 1080×1080 matrix, and Σ is a 1920×1080 diagonal matrix. Specifically, most elements in Σ are zero.

Suppose the rank of Σ is k. In order to recover M, only k lines of U and V is needed, with k elements of Σ. Thus, a total number of bytes to be transferred is

$$(1920 \times k + 1080 \times k + k) \times 8$$

The ratio of compression is

$$\frac{1920 \times 1080}{(1920 + 1080 + 1) \times k}$$

Moreover, since the value of some non-zero diagonal elements are not so important compared with others, they can be ignored by regarding them as 0. Thus we get the most important k' dimension, which is smaller than k. As a result, a larger compression ratio, at the cost of losing $k - k'$ dimension of data while recovering. Smaller the k', more dimension will be lost. However, in most cases, a loss of unimportant dimension is acceptable, considering the speed-up of transferring benefit from compression.