

Image Compression Using SVD

Ian Cooper and Craig Lorenc

December 13, 2006

Table of Contents

- 1 Introduction
- 2 SVD Overview
- 3 SVD Example
 - Step 1: Calculate AA^T and $A^T A$
 - Step 2: Eigenvalues and Σ
 - Step 3: Finding U
 - Step 4: Finding V
 - Step 5: The complete SVD
- 4 Image Compression
 - Digitizing Images
 - SVD and Compression
 - Saving Memory
 - MATLAB

Introduction

Data Compression

Data compression is an important application of linear algebra. The need to minimize the amount of digital information stored and transmitted is an ever growing concern in the modern world. **Singular Value Decomposition** is an effective tool for minimizing data storage and data transfer. This presentation explores image compression through the use of singular value decomposition on image matrices.

SVD Overview

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

$$U = [\mathbf{u}_1 \quad \cdots \quad \mathbf{u}_n], \quad \Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_n \end{bmatrix}, \quad V^T = \begin{bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_n \end{bmatrix}$$



SVD Example

A 2×2 matrix of rank 2 is a natural choice for an example of the SVD approach as most images will undoubtedly have full rank. It also provides the proper stage for an efficient illustration of the process at hand. Consider matrix A ,

$$A = \begin{bmatrix} 2 & 2 \\ -1 & 1 \end{bmatrix}$$

it follows that,

$$A^T = \begin{bmatrix} 2 & -1 \\ 2 & 1 \end{bmatrix}$$

Step 1: Calculate AA^T and $A^T A$

The first step in the SVD algorithm is calculating AA^T and $A^T A$

$$AA^T = \begin{bmatrix} 2 & 2 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 2 & -1 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 8 & 0 \\ 2 & 0 \end{bmatrix}$$

$$A^T A = \begin{bmatrix} 2 & -1 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 2 & 2 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix}$$

Step 2: Eigenvalues and Σ

The second step of the process is to find eigenvalues for AA^T and $A^T A$. Once we have those we can easily compute the singular values by taking the square roots of the eigenvalues, λ_1 and λ_2 . With the acquisition of σ_1 and σ_2 we can form Σ ,

$$|AA^T - \lambda I| = 0 \quad (1)$$

$$|A^T A - \lambda I| = 0 \quad (2)$$

Step 2: Eigenvalues and Σ

Solving equation (1) yields λ_1 and λ_2 :

$$\begin{aligned} |AA^T - \lambda I| &= 0 \\ \left| \begin{bmatrix} 8 & 0 \\ 0 & 2 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right| &= 0 \\ \begin{vmatrix} 8 - \lambda & 0 \\ 0 & 2 - \lambda \end{vmatrix} &= 0 \\ (8 - \lambda)(2 - \lambda) &= 0 \\ \lambda_1 &= 8 \\ \lambda_2 &= 2 \end{aligned}$$

Step 2: Eigenvalues and Σ

Now that we have λ_1 and λ_2 we can compute the singular values of AA^T . With σ_1 and σ_2 we can form Σ .

$$\sigma_1 = \sqrt{\lambda_1} = \sqrt{8}$$

$$\sigma_2 = \sqrt{\lambda_2} = \sqrt{2}$$

Step 2: Finding Σ

$$\Sigma = \begin{bmatrix} \sqrt{8} & 0 \\ 0 & \sqrt{2} \end{bmatrix}$$

Step 3: Finding U

The columns of U are the unit eigenvectors of AA^T . We will need to solve equation (3) using both eigenvalues to find two eigenvectors that we can use for the columns of U .

$$(AA^T - \lambda I)\mathbf{x} = 0 \quad (3)$$

$$(A^T A - \lambda I)\mathbf{x} = 0. \quad (4)$$

Step 3: Finding U

Solve for the first eigenvector of AA^T . This will be used to generate the first column of U .

$$\begin{aligned}
 (AA^T - \lambda I)\mathbf{x} &= 0 \\
 \left(\begin{bmatrix} 8 & 0 \\ 0 & 2 \end{bmatrix} - \lambda_1 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \mathbf{x}_1 &= 0 \\
 \begin{bmatrix} 8 - \lambda_1 & 0 \\ 0 & 2 - \lambda_1 \end{bmatrix} \mathbf{x}_1 &= 0 \\
 \begin{bmatrix} 8 - 8 & 0 \\ 0 & 2 - 8 \end{bmatrix} \mathbf{x}_1 &= 0 \\
 \begin{bmatrix} 0 & 0 \\ 0 & -6 \end{bmatrix} \mathbf{x}_1 &= 0 \\
 \mathbf{x}_1 &= \begin{bmatrix} -1 \\ 0 \end{bmatrix}
 \end{aligned}$$

Step 3: Finding U

Solve for the second eigenvector of AA^T . This will be used to generate the second column of U .

$$(AA^T - \lambda I)\mathbf{x} = 0$$

$$\left(\begin{bmatrix} 8 & 0 \\ 0 & 2 \end{bmatrix} - \lambda_2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \mathbf{x}_2 = 0$$

$$\begin{bmatrix} 8 - \lambda_2 & 0 \\ 0 & 2 - \lambda_2 \end{bmatrix} \mathbf{x}_2 = 0$$

$$\begin{bmatrix} 8 - 8 & 0 \\ 0 & 2 - 8 \end{bmatrix} \mathbf{x}_2 = 0$$

$$\begin{bmatrix} 0 & 0 \\ 0 & -6 \end{bmatrix} \mathbf{x}_2 = 0$$

$$\mathbf{x}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Step 3: Finding U

$$\mathbf{x}_1 = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \quad \mathbf{x}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Generate the columns of U by turning x_1 and x_2 into unit vectors (U and V must be orthonormal).

$$\mathbf{u}_1 = \frac{\mathbf{x}_1}{\|\mathbf{x}_1\|} \quad \mathbf{u}_2 = \frac{\mathbf{x}_2}{\|\mathbf{x}_2\|}$$

$$\mathbf{u}_1 = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \quad \mathbf{u}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Step 3: Finding U

Now that we have unit eigenvectors \mathbf{u}_1 and \mathbf{u}_2 we can form U .

$$U = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

Step 4: Finding V

We use a similar process using equation (4) to find unit eigenvectors of V ...

$$\mathbf{v}_1 = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

$$\mathbf{v}_2 = \begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

Step 4: Finding V

Now that we have unit eigenvectors \mathbf{v}_1 and \mathbf{v}_2 we can form V .

$$V = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$

Step 5: The complete SVD

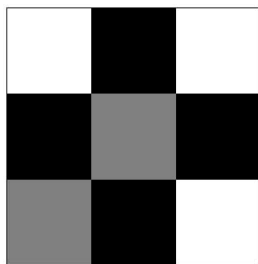
By securing U , Σ , and V the factorization of A is realized and the SVD is complete.

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

$$\begin{bmatrix} 2 & 2 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{8} & 0 \\ 0 & \sqrt{2} \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$

Digitizing Images

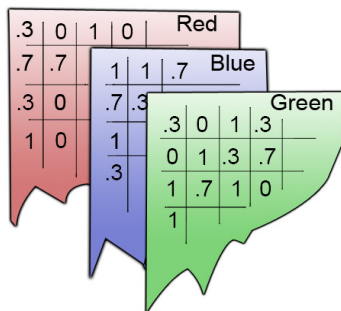
A computer stores an image as a matrix. Each value represents a corresponding pixel brightness. For example, this is how the computer would record a 3×3 pixel image (enlarged), where 0 is black, and 1 is white.



$$= \begin{bmatrix} 1 & 0 & 1 \\ 0 & .5 & 0 \\ .5 & 0 & 1 \end{bmatrix}$$

Color vs. Grayscale

For color images, a computer stores three brightness values for each pixel: one for red, green, and blue. It saves each of these color spectrums in a different “layer”,



Each layer by itself can be treated as a grayscale image matrix.

SVD as a Sum

Important :

SVD allows us to rewrite a matrix as a sum of rank one matrices.

$$A = U\Sigma V'$$

$$A = [\mathbf{u}_1 \quad \cdots \quad \mathbf{u}_n] \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^T \\ \vdots \\ \mathbf{v}_n^T \end{bmatrix}$$

$$A = \mathbf{u}_1\sigma_1\mathbf{v}_1^T + \cdots + \mathbf{u}_n\sigma_n\mathbf{v}_n^T$$

$$A = \sum_{i=1}^n \sigma_i \mathbf{u}_i \mathbf{v}_i^T$$

SVD as a Sum

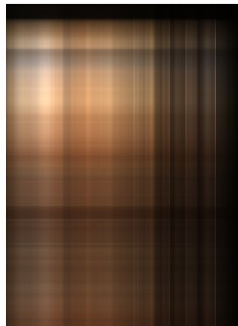
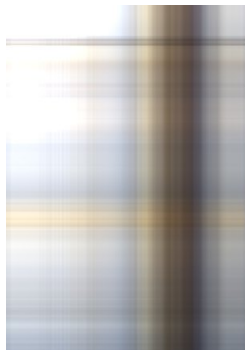
Recall that $\sigma_1 \geq \dots \geq \sigma_n$ which means the first term will have the largest impact on the total sum, followed by the the second term, then the third term, etc.

We can approximate a matrix by adding only the first few terms of the series!

The resulting sum is rank k , where k is the number of rank one matrices added together for the approximation.

Rank 1 SVD Approximations

You can actually see how the compression breaks down the matrix for a rank one approximation.



How much Memory does SVD Save?

Memory required to store an uncompressed image of size $m \times n$:

$$I_M = mn$$

Notice, the amount of memory required increases exponentially as the dimensions get larger.

How much Memory does SVD Save?

Memory required by an SVD approximation of rank k :

$$A_M = k(m + n + 1)$$

The amount of memory required increases *linearly* as the dimensions get larger, as opposed to *exponentially*. Thus, as the image gets larger, more memory is saved by using SVD.

How much Memory does SVD Save?

A problem arises:

If SVD is used to replicate an image *exactly*, the “compressed” image takes up more space than the uncompressed image.

This implies there are important limits on k for which $A_M < I_M$

$$A_M < I_M$$

$$k(m + n + 1) < mn$$

$$k < \frac{mn}{m + n + 1}$$

Now we can try it out.

The Code

Example Program:

```
I = imread('pig.jpg');
I = im2double(I);
k=5;
for i = 1:3
    [U,S,V] = svd(I(:, :, i));
    A(:, :, i) = U(:, 1:k)*S(1:k, 1:k)*V(:, 1:k)';
end
A(A>1)=1; A(A<0)=0;
imshow(A)
axis off
imwrite(A, 'pig_compressed.jpg')
```

The GUI

Conclusion: quote from *Strang*, pg. 357

“I give you my opinion directly. The SVD is the climax of this linear algebra course. I think of it as the final step in the Fundamental Theorem. First come the *dimensions* of the four subspaces. Then their *orthogonality*. Then the *orthonormal bases which diagonalize A*. It is all in the formula $A = U\Sigma V^T$. More applications are coming-they are certainly important-but you have made it to the top.”