Exploring Image compression using Singular Value Decomposition (SVD)

1st Chandrakishor Singh
Department of Information Technology
Indian Institute of Information Technology, Allahabad
mit2021117@iiita.ac.in

2nd Dablu Chauhan
Department of Information Technology
Indian Institute of Information Technology, Allahabad
mit2021115@iiita.ac.in

3rd Madhu Donipati
Department of Information Technology
Indian Institute of Information Technology, Allahabad
mit2021116@iiita.ac.in

***Abstract*—This paper presents a fundamental introduction to the Singular Value Decomposition(SVD) of a matrix. We begin with a short discussion on how digital images are represented and why compression of images are desired in many applications. Then we present some mathematical preliminaries which are required to understand the SVD. This includes the understanding of eigenvalues and eigenvectors, orthogonal matrix and positive definite matrix. After that we derive SVD and give a general algorithm to calculate SVD of a matrix. Finally we discuss the application of SVD in image compression and demonstrate this with the help of a simple python script to compress an image with lower ranks. At last, we give some other important applications of SVD.**

***Keywords*—Singular Value Decomposition, SVD, Image Compression, Python**

I. Introduction

A digital image is essentially a matrix of order $p1$ x $p2$. Here $p1$, $p2$ defines the width and height of the image in terms of pixels. For example, when we capture an image, say of 1920 x 1080 resolution, then it basically means that the width of the image is 1920 pixels and its height is 1080 pixels. Hence, an image is just a matrix of pixels which in turn are a combination of red, green and blue(RGB) colors.

Image compression is a crucial topic of digital image processing that deals with ways for minimising the amount of storage or bandwidth necessary to save or transmit an image. The image compression algorithm's purpose is to represent the image with less data. Redundancy is an approximate repeated pattern that tends to give the image a degree of resolution. The presence of redundancies in the image file can be used to lower the number of bits used. However, the image quality should not be degraded to the point where the user is unable to understand it. Linear algebra is used extensively in data compression. In today's world, the necessity to reduce the amount of digital data stored and transmitted is becoming extremely important. A typical photographic image can be compressed to roughly 80% of its original size without noticeably deteriorating in quality.

Various picture compression techniques have been developed to date. Predictive Coding, Fractal Compression, Wavelet Compression are some of these techniques. This paper focuses on Singular Value Decomposition (SVD), a method of factoring matrices into a sequence of linear approximations that reveal the matrix's underlying structure. Object detection, face recognition, field matching techniques, and the analysis of meteorological and oceanographic data all use SVD. [4]

Specifying the rate and distortion settings for the target image, and separating the image data into several classes depending on their relevance are the typical procedures that are involved in compressing an image. [4]

There are several methods for compressing images, but broadly they can be divided into two categories: Lossless and Lossy.

Lossless image compression involves a compression technique which is numerically equivalent to the original image. Hence, it is possible to construct the original image pixel-by-pixel from the compressed image.

On the other hand, Lossy compression results in loss of some information related to the original image and hence it is not possible to uniquely reconstruct the original image from the compressed image.

II. Mathematical Preliminaries

Before trying to understand SVD and its application of compressing an image, one must first understand a few related concepts that are used to obtain the singular value decomposition formula.

A. Eigenvalue & Eigenvector

Eigenvalue and eigenvector are some of the most important properties of a matrix. They contain a lot of information regarding the nature of the matrix. Mathematically, for a matrix A, the vector x is said to be its eigenvector if for some scalar λ the following holds true. [1]

$$Ax = \lambda x$$

λ is called the corresponding eigenvalue of A associated with the eigenvector x.

There may be multiple eigenvector and eigenvalue of a matrix A. If the order of matrix A is n then, at maximum, it will have n different eigenvalues and eigenvectors.

An eigenvector, which corresponds to a real nonzero eigenvalue, points in the direction that the transformation(the matrix multiplication Ax) stretches it, and the eigenvalue is the factor by which it stretches it. The orientation is reversed if the eigenvalue is negative. [1]

The eigenvalues of a matrix A can be found out from the solution of its characteristic equation $|A - \lambda I| = 0$

Below are some properties of eigenvalues and eigenvectors which will be essential in understanding further content of this paper.

- If $Ax = \lambda x$ then $Ax = \lambda x$ i.e., the eigenvectors of the powers of the same matrix remain unchanged while the eigenvalues change in an exponential manner.
- If $Ax = \lambda x$ then $(A + cI)x = (\lambda + c)x$

B. Orthogonal Matrix

An orthogonal matrix is a real square matrix whose columns and rows are orthonormal vectors i.e., any two column or row vectors are orthogonal to each other. One example of such a matrix is as follows. [1]

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

Properties of Orthogonal Matrix:

- The dot product of any two such column or row vectors is always zero and their length is 1 i.e. they are unit vectors.
- All of the vectors are orthogonal and independent to each other and hence they also form a basis for the respective vector space.
- The transpose of the orthogonal matrix is its inverse. Hence, $QQ^T = Q^TQ = I$ and the inverse Q^T is also an orthogonal matrix.
- Determinant of an orthogonal matrix is always either 1 or -1. [3]

C. Positive Definite Matrix

A positive definite matrix is a matrix in which all its eigenvalues are positive and it is symmetric. One example of such a matrix is as follows. [1]

$$\begin{bmatrix} 4 & -2 & 0 \\ -2 & 4 & -2 \\ 0 & -2 & 4 \end{bmatrix}$$

Properties of positive definite matrix:

- **Scaling:** If M is a positive definite and $r > 0$ is a real number then rM is a positive definite.
- **Addition:** If M and N are positive definite, then the sum $M + N$ is also positive definite.
- **Multiplication:** If M and N are positive definite, then the products MNM and NMN are also positive definite. If $MN = NM$, then MN is also positive definite.

III. Singular Value Decomposition(SVD)

The factorization of a matrix A into the product of three matrices $A = USV^T$ where the columns of U and V are orthonormal and the matrix S is diagonal with positive real entries, is known as singular value decomposition. The SVD is useful for a variety of applications. In many cases, the data matrix A is a good approximation to a low rank matrix, hence finding a low rank matrix that is a good approximation to the data matrix A is useful. SVD has the advantage of being able to be applied to any real m x n matrix unlike other decomposition like LU, QR which does not exists for every matrix. [4]

In the equation $A = USV^T$, the diagonal elements s_i of the S matrix are called singular values, while the columns of U and V are represented as u_i and v_i respectively. The orthonormal basis is formed by the singular vectors. One the main diagonal, the singular values are arranged in the following order,

$$\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \cdots \sigma_r \geq \sigma_{r+1} = \cdots = \sigma_p = 0$$

here r, is the rank of the matrix A, and p is the smaller dimension among m and n.

If the image has a low rank when represented as a matrix, or can be approximated sufficiently well by a matrix of low rank, SVD can be used to determine this approximation, and this low rank approximation can be written in a much more compatible manner than the original image.

To do the SVD, we select the matrix A which has m rows and n columns. Then this matrix is factorized into 3 matrices U, S and V^T. To get the matrix V, we need to follow below steps.

Multiply both sides of equation $A = USV^T$ by A^T yields,

$$A^TA = (USV^T)^T(USV^T) = VS^TU^TUSV^T$$

Since, U and U^T are inverse of each other, UU^T results in I and S is a digonal matrix, hence $SS^T = S^2$. So finally we get,

$$A^TA = VS^2V^T$$

The Eigen values are the squares of the elements of S(the singular values) in the context of SVD compression, and the Eigen vector is made up of columns of matrix V(the right singular vectors).

The next step would be to find U, which is done as follows.

Multiply both sides of equation $A = USV^T$ by A^T yields,

$$A^TA = (USV^T)(USV^T)^T = USV^TVS^TU^T$$

Simplifying further, we get,

$$A^TA = US^2U^T$$

This time, eigenvector for the matrix AA^T are calculated. These vectors corresponds to the columns of U(the left singular vector).

Once U, S and V matrices are obtained, matrix A can be generated which is represented as the product of matrices U, S and V^T.

IV. SVD in Image Compression

The matrix A (m x n) is approximated using a much smaller number of elements than the original matrix. The redundant information is deleted when the rank $r < m$ or $r < n$. The total number of non-zero diagonal elements of the S-matrix is represented by rank. These are called singular values which are arranged in decreasing order along the main diagonal. The values which are after the rank r are represented by 0. [4]

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_r u_r v_r^T + \cdots + 0 u_r v_r^T$$

Adding on the dependent terms when the singular values are equal to zero has no effect on the image quality because the singular values are always bigger than zero. As a result, the terms at the end of the equation are zeroed out, resulting in,

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_r u_r v_r^T$$

We can further approximate the matrix by omitting some of the matrix A's singular terms. This minimises the amount of space necessary to store the image on a computer, allowing for better disc space use.

A. Demonstration

The Singular Value Decomposition approach, which was mentioned in the preceding section, is used to further compress JPEG images. We have used the following approach to demonstrate the usage of SVD in image compression.

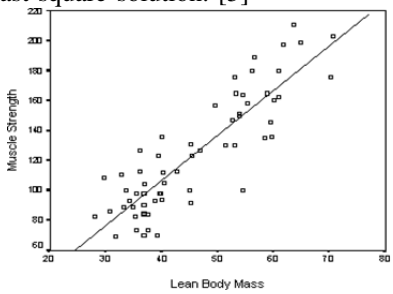
We are using python and its related library like matplotlib for plotting and reading image files, numpy for matrix manipulation and to calculate SVD.

After that we calculate the SVD of the image matrix using the built-in linalg.svd method of the numpy package. Then we try to approximate the image matrix to rank of 5, 20, 100 respectively to see how much compression is achieved and what is the quality of the corresponding image.

V. Other Applications

A. Least Square problems

In real life, there might be times when we want to find a "best-fitting" curve to a set of some given points. For example, people might want to find the relationship between the lean body mass of human bodies and their muscle strength or analyze how the housing price changes by years to obtain an estimation of the price in the future. Most of the time, we might not be able to find such a linear equation that can be satisfied by all of those points, so finding the "closest" solution is the best we can do. We call this solution to be the least-square solution. [5]



B. Recommendation System

A recommender system is an intelligent system that predicts the rating and preferences of users on products. The primary application of recommender systems is finding a relationship between users and products in order to maximize user-product engagement. The major application of recommender systems is in suggesting relatedvideo or music for generating a playlist for the user when they are engaged with a related item. Companies such as Amazon and Netflix collect tons of data, for example, users' browsing or purchasing records, and provide recommendations to them according to some analyses on the data. How to know the user better and recommend them the products they truly like then becomes a crucial problem that needs an elective solution. This is done by using singular value decomposition. [6]

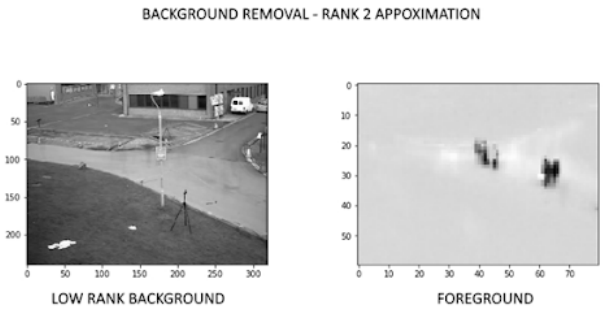
User\movie	Matrix	Inspection	Martin	Titanic	War	Bahubali
Madhu	1	1	1	0	0	2
Dablu	3	3	3	1	0	1
Chandrakisor	4	4	4	0	0	2
Rashmi	5	5	5	1	2	0
Rohit	0	0	0	4	3	5
Jyoti	5	4	2	1	5	6
Aarohi	0	0	0	2	2	1

C. Removing Background from Videos

Think of how you would distinguish the background of a video from its foreground. The background of a video is essentially static – it does not see a lot of movement. All the movement is seen in the foreground. This is the property that we exploit to separate the background from the foreground.

The following approach is used for removing Background from the video: [7]

- Create matrix M from video – This is done by sampling image snapshots from the video at regular intervals, flattening these image matrices to arrays, and storing them as the columns of matrix M
- Matrix M is the sum of two matrices – one representing the background and the other the foreground
- The background matrix does not see a variation in pixels and is thus redundant i.e. it does not have a lot of unique information. So, it is a low-rank matrix, a low-rank approximation of M is the background matrix. We use SVD in this step
- We can obtain the foreground matrix by simply subtracting the background matrix from the matrix M



VI. Conclusion

This report draws attention towards singular value decomposition(SVD). We have explained the major prerequisites required to understand SVD and how it is used in image compression. We were able to achive compression of about

60-80%. Finally, we have demonstrated the concept by writing a python script which compress the image by approximating the matrix to have a rank of k.

VII. Acknowledgment

We would like to express our special thanks of gratitude to our teacher Dr. Mohammed Javed, who gave us the assignment which helped me to enhance our understanding in the field of digital image processing and application of linear transformation. We also want to thank Mr. Bulla Rajesh, who mentored us to complete this assignment Successfully.

References

[1] Introduction to Linear Algebra, 5th Edition - MIT Mathematics
[2] Wikimedia Foundation. (2021, November 19). Definite matrix. Wikipedia. Retrieved November 20, 2021, from https://en.wikipedia.org/wiki/Definite_matrix.
[3] Banerjee, Sudipto; Roy, Anindya (2014), Linear Algebra and Matrix Analysis for Statistics.
[4] H. S. Prasantha, H. L. Shashidhara and K. N. Balasubramanya Murthy, "Image Compression Using SVD," International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007), 2007, pp. 143-145, doi: 10.1109/ICCIMA.2007.386.
[5] https://www.analyticsvidhya.com/blog/2019/08/5-applications-singular-value-decomposition-svd- data-science/
[6] J. Leskovec, A. Rajaraman, J. Ullman, Mining of Massive Datasets, Cambridge University Press, 2011, 418-427.
[7] G. Dallal, Introduction to Simple Linear Regression, available at http://www.jerrydallal.com/lhsp/slr.htm.

VIII. Appendix

The python code used is as follows.

```
from matplotlib.image import imread
import matplotlib.pyplot as plt
import numpy as np
import os
plt.rcParams['figure.figsize'] = [16, 8]


A = imread(os.path.join('..','DATA','dog.jpg'))
X = np.mean(A, -1);

img = plt.imshow(X)
img.set_cmap('gray')
plt.axis('off')
plt.show()

U, S, VT = np.linalg.svd(X,full_matrices=False)
S = np.diag(S)

j = 0
for r in (5, 20, 100):
    # Construct approximate image
    Xapprox = U[:,:r] @ S[0:r,:r] @ VT[:r,:]
    plt.figure(j+1)
    j += 1
    img = plt.imshow(Xapprox)
    img.set_cmap('gray')
    plt.axis('off')
    plt.title('r = ' + str(r))
    plt.show()
```