SINGULAR VALUE DECOMPOSITION

Preetham Ganesh Kamisetty, Raviteja Tiruvoipaty

Department of Electrical Engineering

University of South Florida

Tampa, Florida

Abstract:

Now a days Data generation is increasing day by day and is becoming a challenge for everyone. Storing such a huge amount of data and analyzing it afterwards is more complex problem in this digital decade. Not every data stored have necessary information or exact information but is a mixture of redundant data along with it. One such type of data storage is in the form of Images. Here in this paper we are trying to tackle the ways to reduce the image footprint's data by a process of data compression using Singular Value Decomposition. In order to demonstrate the image compression technique, an example image is used and singular value decomposition is applied to it. The simulation results have been attached to the report.

Keywords: *Singular Value Decomposition (SVD), Image Compression, Diagonalization.*

I. Introduction

In general, a matrix is a rectangular array of numbers. It has elemental rows and columns. It is denoted as MxN where M is the number of rows whereas N gives the number of Columns. Coming to Linear algebra, which is one of the branches of Mathematics dealing with Linear equations in the form of Matrices and spaces. Matrices are a great way to analyze the data and are simple to understand. They have a lot of applications such as in mathematics they are used to plot different graphs, solving linear equations, extracting statistical information etc. and real-world applications of matrices include data manipulation, data compression, data formatting etc. Image compression is a technique used in almost every area to reduce the redundant bits of data present in the actual or original image without degrading the important information. Compression allows us to store more data which is an advantage as we can use a greater number of images in the available storage One of the techniques used in Image compression is derived from linear algebra, which is Singular Value Decomposition (SVD). So, we use an image and try to convert it into a matrix for manipulating the data and in the end, we try to compress the image size by applying Singular Value Decomposition on that matrix.

To clearly understand the oncoming topics, first we need to have an idea about the following small topics in linear algebra-

Inverse of a matrix:

A matrix X is invertible if and only if its concerned determinant is not equal to zero (i.e.; matrix X should be non-singular). The inverse of the matrix is denoted as X^{-1}. A matrix when multiplied by its inverse gives us an identity matrix.

Eigenvalues of a matrix:

The characteristic roots of a given matrix are the eigenvalues of that matrix. These values are the scalars of the system of linear equations written in the form of the matrix.

Eigenvectors of a matrix:

An eigenvector is a non-zero vector which gives us the direction of a matrix corresponding to its eigenvalue. The characteristic roots are basically the positional factors which are used to find the eigenvector which gives us the direction. These vectors further help us diagonalizing the matrix.

Diagonal Matrix:

A matrix is said to be a diagonal matrix, if the values other than the diagonal elements are zero.

II. Diagonalization

A matrix 'X' is diagonalized if and only if we can write its product as follows,

$$X=SDS^{-1}$$

Here, S is said to be a matrix which is formed by the eigenvectors of 'X' and it is also an invertible matrix. S^{-1} is simply the inverse of the vector matrix. On the other hand, 'D' is the Diagonal Matrix with eigen values as its entries.

The important thing about the diagonalization is that we can notice that, as 'S' is invertible, it must be a square matrix. So basically, if the given matrix is not a square matrix, we cannot diagonalize it. The major limitation is that there is no chance for the rectangular matrices.

From the above formula,

$$X=SDS^{-1}$$

That implies,

$$XS=DS$$

Where, S= (x_1, x_2, x_3… x_n) and

$$D = \begin{pmatrix} \lambda 1 & 0 & \dots\dots\dots\dots\dots & 0 \\ 0 & \lambda 2 & 0 & 0 \\ \dots\dots\dots & \dots\dots\dots\dots & \dots\dots\dots & \dots\dots \\ 0 & \dots\dots\dots & 0 & \lambda n \end{pmatrix}$$

λ corresponds eigenvalues of 'X'

Where, DS= ($\lambda_1 x_1$, $\lambda_2 x_2$, $\lambda_3 x_3$… $\lambda_n x_n$)

III. Singular Value Decomposition

SVD is the acronym for Singular Value Decomposition. SVD is widely used as a decomposing tool. Its application greatly lies in the areas of image compression. As we know that Diagonalization is only capable of decomposing the square matrices. SVD is designed in such a way to overcome this limitation. So Singular value decomposition can be used for matrices of any shape.

Let 'X' be a matrix of size mxn, then SVD of 'X' is given by a formula,

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

U and V are the orthogonal matrices and Σ is a diagonal matrix.

Here, U has a size of mxm and lies in R^m and it forms the fundamental subspaces such as-

u1, ..., Ur forms an orthonormal basis for the column space

Ur+ 1, ..., Um is an orthonormal basis for the left nullspace N (X^T) [1]

The V matrix has the size of nxn and lies in R^n. It also forms some subspaces of matrix 'X' -

v1, ..., Vr is an orthonormal basis for the row space

Vr+l, ..., Vn is an orthonormal basis for the null space N(X). [1]

And 'Σ' is matrix containing singular values derived from the eigen values of the given matrix 'X'. Its size is mxn.

Now we can see that,

mxn = (mxm)(mxn)(nxn) gives us the same size matrix.

Steps to Find the SVD of Matrix:

1. Calculate X^TX and XX^T
2. Find the characteristic roots by formula (X^TX- λI) = 0. The roots will be our eigen values.
3. Find the singular values with the obtained eigen values (i.e., $\alpha_n^2=\lambda_n$). It is our Σ.
4. Use the roots to find the corresponding eigen vectors.
5. Multiply the vectors by its length, as the vectors need to be orthonormal vectors.
6. Transposing the orthonormal vectors of X^TX gives us the V^T.
7. By using AV=U Σ (Since $V^T=V^{-1}$), we can find the required U.
8. Verify by multiplying $U\Sigma V^T$.

By following the above steps, we can find the SVD of any size matrix.

IV. Computing SVD

Let's take an example matrix and prove $X=U\sum V^T$

$$\text{Let } X=\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

$$X^T=\begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}$$

$$XX^T=\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}=\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

$| XX^T-\lambda I|=0$

$$\begin{vmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{vmatrix}=0$$

$\lambda^2-4\lambda+3=0$

$\lambda=3,1$

Therefore, the eigenvalues for XX^T are 3,1.

So, in the case of $XX^{T,}$ the matrix which is formed by using the eigenvectors is nothing but the U matrix.

Therefore, the eigenvectors which are obtained are

For λ=3

$$u1=\begin{pmatrix}1/\sqrt{2}\\1/\sqrt{2}\end{pmatrix}$$

where u1 is the eigenvector and to make it orthonormal, we should divide it by its own length.

for λ=1

$$u2=\begin{pmatrix}1/\sqrt{2}\\-1/\sqrt{2}\end{pmatrix}$$

where u2 is the eigenvector and to make it orthonormal, we should divide it by its own length.

U= (u1 u2)

$$U=\begin{pmatrix}1/\sqrt{2} & 1/\sqrt{2}\\1/\sqrt{2} & -1/\sqrt{2}\end{pmatrix}$$

Now we have to find

$$X^TX=\begin{pmatrix}1 & 0\\1 & 1\\0 & 1\end{pmatrix}\begin{pmatrix}1 & 1 & 0\\0 & 1 & 1\end{pmatrix}=\begin{pmatrix}1 & 1 & 0\\1 & 2 & 1\\0 & 1 & 1\end{pmatrix}$$

Here,

$$|X^TX-\lambda I|=0 \quad \begin{vmatrix}1-\lambda & 1 & 0\\1 & 2-\lambda & 1\\0 & 1 & 1-\lambda\end{vmatrix}=0$$

$\lambda^3-4\lambda^2+3\lambda=0$

Therefore, the eigenvalues are 3,1,0.

From these eigenvalues we have to find the singular values

s1= $\sqrt{3}$

s2= $\sqrt{1}$=1

$$\sum=(s1\ s2)=\begin{pmatrix}\sqrt{3} & 0 & 0\\0 & 1 & 0\end{pmatrix}$$

Now by finding the eigenvectors, we get the matrix V.

For λ=3

$$v1=\begin{pmatrix}1/\sqrt{6}\\2/\sqrt{6}\\1/\sqrt{6}\end{pmatrix}$$

for λ=0

$$v2=\begin{pmatrix}1/\sqrt{2}\\0\\-1/\sqrt{2}\end{pmatrix}$$

for λ=0

$$v3=\begin{pmatrix}1/\sqrt{3}\\-1/\sqrt{3}\\1/\sqrt{3}\end{pmatrix}$$

So, V= (v1 v2 v3)

$$V=\begin{pmatrix}1/\sqrt{6} & 1/\sqrt{2} & 1/\sqrt{3}\\2/\sqrt{6} & 0 & -1/\sqrt{3}\\1/\sqrt{6} & -1/\sqrt{2} & 1/\sqrt{3}\end{pmatrix};$$

$$V^T=\begin{pmatrix}1/\sqrt{6} & 2/\sqrt{6} & 1/\sqrt{6}\\1/\sqrt{2} & 0 & -1/\sqrt{2}\\1/\sqrt{3} & -1/\sqrt{3} & 1/\sqrt{3}\end{pmatrix}$$

As we know $X=U\sum V^T$

X=

$$\begin{pmatrix}1/\sqrt{2} & 1/\sqrt{2}\\1/\sqrt{2} & -1/\sqrt{2}\end{pmatrix}\begin{pmatrix}\sqrt{3} & 0 & 0\\0 & 1 & 0\end{pmatrix}\begin{pmatrix}1/\sqrt{6} & 2/\sqrt{6} & 1/\sqrt{6}\\1/\sqrt{2} & 0 & -1/\sqrt{2}\\1/\sqrt{3} & -1/\sqrt{3} & 1/\sqrt{3}\end{pmatrix}$$

V. SVD in Image Compression

In general, an Image means a group of pixels together. Pixels are extremely small and depending on their value they display that color. So, a pixel has a range of 0 to 255 where 0 value displays properly no color and 255 gives the dark shade. They are only three colors (i.e., Red, Green and Blue) with which any color can be formed. So, depending on the image different pixels are allotted different values within the range of 0 to 255. These pixels form a grid like structure. The best thing with this is that it can be represented in a matrix format. So any number of pixels present in an image can be formed into a matrix. We can use that matrix to perform image compression tasks of our requirement. Using singular value decomposition on Image gives a compressed image with the most important parts of the image taken from the original image. By this we

can compress the image. Basically, what SVD does is that it will store the Important values in its matrices and provide the best possible outcome. The matrix U store the required important information on its top rows, where as V stores the important data in its top columns.

Let's make it clear by taking an example:

Example:

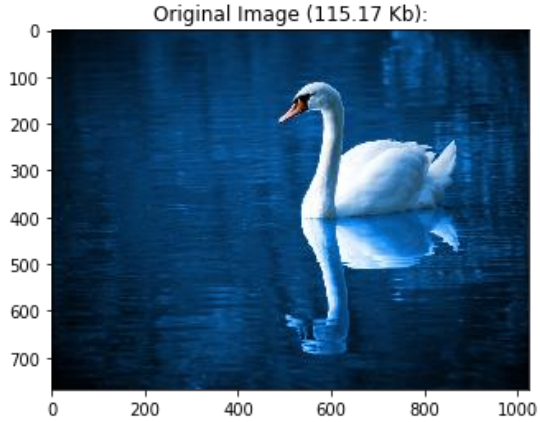
```
#Importing necessary libraries

import matplotlib.pyplot as plt

from PIL import Image

import numpy as np

import matplotlib

import os

%matplotlib inline

image = Image. Open('Swan_large.jpg') #opening the image

s = float(os.path.getsize('Swan_large.jpg'))/1000 #Storing the size of the image

print("dimensions of the image: ",image.size) #Printing the dimensions of the image

print("size of original image in terms of KB",os.path.getsize('Swan_large.jpg')) #size in terms of KB

plt.title("Original Image (%0.2f Kb):" %s) #plotting the original

plt.imshow(image)

gray_img = image.convert('LA') #Converting the image to gray scale

matrix_img = np.array( list(gray_img.getdata(band = 0)), float) #Converting the image to matrix

matrix_img.shape = (gray_img.size[1], gray_img.size[0]) #setting the image size

matrix_img = np.matrix(matrix_img)

plt.figure() #setting dimensions of the figure

plt.imshow(matrix_img, cmap = 'gray') #initializing the gray scaled image for plot

plt.title("Image after converting it into the Grayscale pattern") #Title for gray scale

plt.show()#plotting the image

print("After compression: ")

U, S, V = np.linalg.svd(matrix_img) #Applying SVD on the image

for i in range(5, 51, 20): #Looping through to print different compressions

    compressed_img= np.matrix(U[:, :i]) * np.diag(S[:i]) * np.matrix(V[:i,:]) #Reforming the image with differnt eigen columns

    print("dimensions of image ",compressed_img.shape) #Dimensions of the reformatted imate

    result = Image.fromarray((compressed_img).astype(np.uint8)) #Converting the matrix to image

    result.save('compressed.jpg') #Saving the image

    print("Size of compressed image (in KB)",os.path.getsize('compressed.jpg')) #the size of the new image

    plt.imshow(compressed_img, cmap = 'gray') #Showing the gray scale in the terminal

    title = (" Rank =  %s" %i,"image")

    plt.title(title)

    plt.show()

#Part 2:

eigvals = S/((matrix_img.shape[1]**2)/10) #converting the eigen values to probabilities

num_vars = eigvals.shape[0] #the dimensions

fig = plt.figure(figsize=(8,5)) #setting the figure dimensions

sing_vals = np.arange(num_vars) + 1

plt.plot(sing_vals, eigvals, 'ro-', linewidth=2) #plotting the probabilities

leg = plt.legend(['Eigenvalues from SVD'], loc='best', borderpad=0.3,  #adding the legend


prop=matplotlib.font_manager.FontProperties(size='small'),

            markerscale=0.4)

plt.xlim([0,30]) #setting the x axis limit

plt.title('Contribution of each eigen vector')

plt.xlabel('Eigen vectors')
```

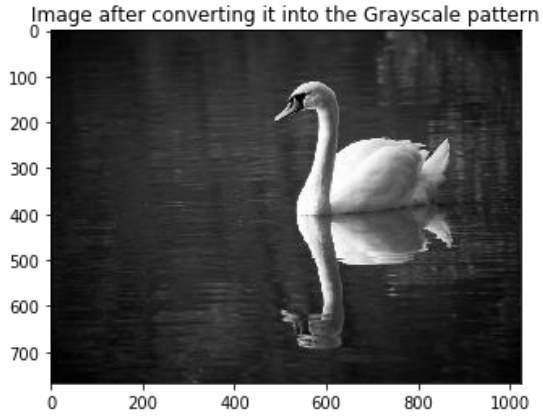
plt.ylabel('Percentage contribution')

plt.show()

Simulation Results:

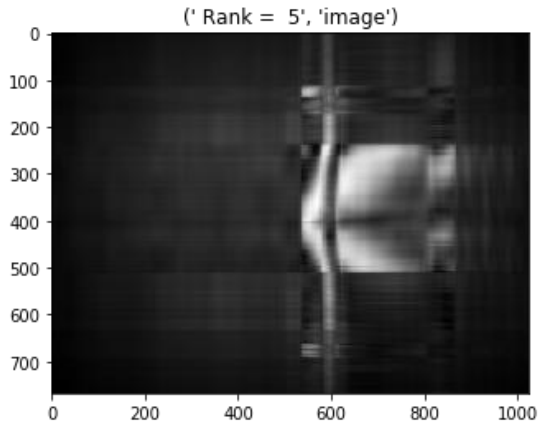
Dimensions of the image: (1024, 768)
Size of original image in terms of KB 395428



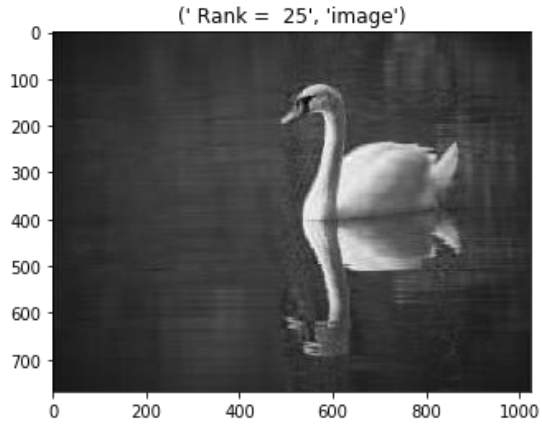


After compression:

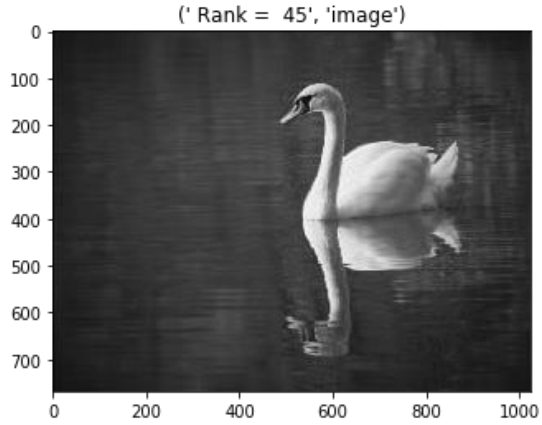
Dimensions of image (768, 1024)
Size of compressed image (in KB) 31755

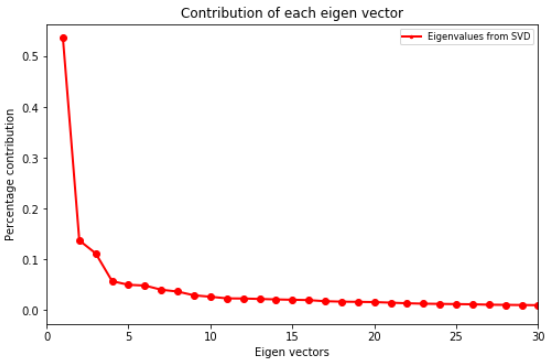


Dimensions of image (768, 1024)
Size of compressed image (in KB) 46117



Dimensions of image (768, 1024)
Size of compressed image (in KB) 52459





VI. Conclusions

We can see that how a simple image turned out to be a compressed image using Singular value decomposition. We can deduce that by using SVD, we can try to decrease the increasing computational cost, we can try to decrease the space occupied by the image and many more. The major problem that SVD solves is that the dimensionality Curse, when

the image has large dimensions, it might have lot of unnecessary information stored in the grid of pixels. SVD takes out the unnecessary information and points at the meticulous information that could deploy the exact copy of the original image. By this SVD has wide range of applications. Other applications of SVD are used in Principal Component Analysis (PCA), in finding Least squares and in the analysis of latent semantic.

VII. Reference

1. Gilbert Strang-Introduction to Linear Algebra (ED-52016, Wellesley-Cambridge Press).
2. Golub, G.H., and Van Loan, C.F. (1989) Matrix Computations, 2nd ed. (Baltimore: Johns Hopkins University Press).
3. Alter O, Brown PO, Botstein D. (2000) Singular value decomposition for genome-wide expression data processing and modeling. *Proc Natl Acad Sci U S A*, **97**, 10101-6.