PES University, Bangalore

(Established under Karnataka Act No. 16 of 2013)

MAY 2020: IN SEMESTER ASSESSMENT (ISA) B.TECH. IV SEMESTER

UE18MA251- LINEAR ALGEBRA

MINI PROJECT REPORT

ON

May 17, 2020

Submitted by

1. Name :Mahima. M.Rao SRN:PES2201800257
2. Name:Muskaan. A SRN: PES2201800672
3. Name:Adithi. K SRN: PES2201800643

Branch & Section : ECE 4B

PROJECT EVALUATION

(For Official Use Only)

Sl.No.	Parameter	Max Marks	Marks Awarded
1	Background & Framing of the problem	4	
2	Approach and Solution	4	
3	References	4	
4	Clarity of the concepts & Creativity	4	
5	Choice of examples and understanding of the topic	4	
6	Presentation of the work	5	
	Total	25	

Name of the Course Instructor :

Signature of the Course Instructor :

DATA AND IMAGE COMPRESSION AND MANIPULATION USING SVD TRANSFORMATION

Content

1| 1.1 Introduction

1.2 Problem Statement

2| Singular Value Decomposition

2.1 Intro

2.2 Intuation

2.3 Computeing the SVD

3| Image Compression

3.1 Intro

3.2 Mathematical Overview

3.3 Saving Memory using SVD

3.4 Python code to demonstrate image compression

3.4.1 Code

3.4.2 Results

(Approximations , graphs ,outputs)

4| Conclution

5| References

1.1 Introduction

Data represented in terms of matrix or Image is a 2 Dimensional signal represented by Digital system,however forprocessing, transmitting and storage, images are converted in to digital form. A Digital Image is basically 2- Dimensional array of pixels .

[1].Different types of images are used in remote sensing, bio medical and video processing techniques which require compression for transmission and storage.

[2].Compression is achieved by removing redundant or extra bits from the image.

Here we use SVD factorization of Linear algebra in order to perform this compression and few intuations are described for svd used in various data manupulation   And will visualizing image compression using python programming language

1.2 Problem statement

Need of Compression : Uncompressed images can occupy a large amount of memory in RAM and in storage media,and they can take more time to transfer from one device to another.

So here we are performing   image compression using SVD , as well also show how effectively it reduces storage size

2 Singular Value Decomposition

2.1

When coming to singular value decomposition, or SVD, is,

Given the matrix A, where the size of a is $m \times n$ where m represents the number of rows in the matrix, and n represents the number of columns, A can be broken down into three sub matrices

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

where U is of size $m \times$ n, Σ is of size $m \times n$ and is diagonal, and $V\ T$ is of size $n \times n$. It is required for matrix multiplication that the size of the columns of the first matrix must match up with the size of the rows of the second matrix.

$$m \times n = [(m \times m)(m \times n)](\mathrm{n} \times n)$$

$$m \times n = (m \times n)(n \times n)$$

$$m \times n = (m \times n)$$

2.2 **Intuation [** To provide various manupulations done usin svd on data matrix**]**

$$\begin{array}{c}\hat{X}\\ \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n}\\ x_{21} & x_{22} & \cdots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn}\end{pmatrix}\\ m \times n\end{array} \approx \begin{array}{c}U\\ \begin{pmatrix} u_{11} & \cdots & u_{1r}\\ \vdots & \ddots & \\ u_{m1} & & u_{mr}\end{pmatrix}\\ m \times r\end{array} \begin{array}{c}S\\ \begin{pmatrix} s_{11} & 0 & \cdots\\ 0 & \ddots & \\ \vdots & & s_{rr}\end{pmatrix}\\ r \times r\end{array} \begin{array}{c}V^{\mathsf{T}}\\ \begin{pmatrix} v_{11} & \cdots & v_{1n}\\ \vdots & \ddots & \\ v_{r1} & & v_{rn}\end{pmatrix}\\ r \times n\end{array}$$

In terms of matrix A as Data

- Matrix X is called the data matrix ,for example if each column vector represents images of faces of a person or the images of the results obtained then their are n columns of data
- *U* is a matrix that holds important information about the columns of the matrix X , each column vector in here represents eigen faces or eigen results of corresponding faces or results of column vector of X .

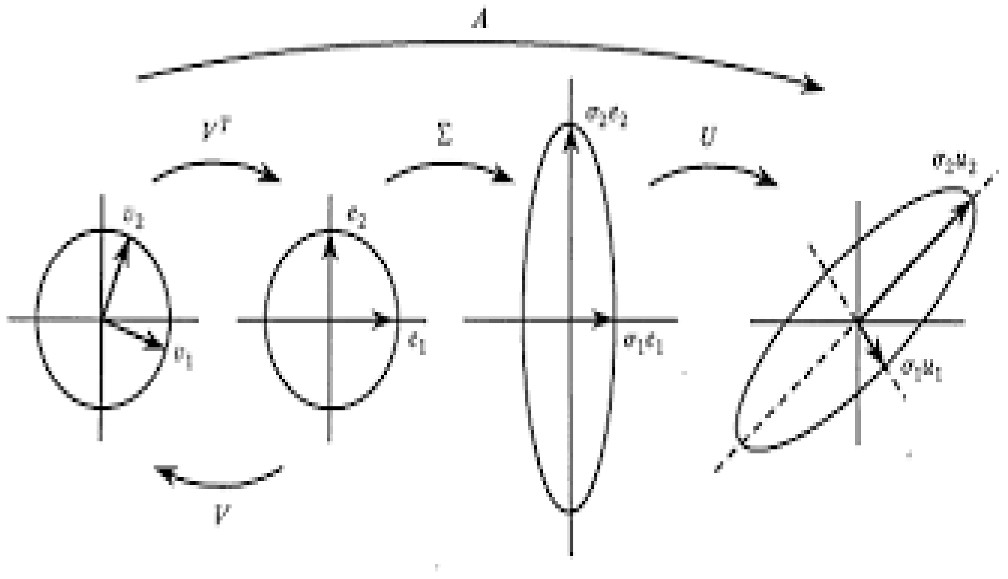
- V^T is a matrix that holds the most important information about the   of the rows of matrix X,each row vector represents the eigen mixture of columns of U to add them to make A1 M
- Σ is a diagonal matrix which has only diagonal values with rest all the values being zero.This is called singular value which is arranged in decending order , it represents the relative imporatance and as well a scaling factor.
- Application : dominant correlation analysis , eigengaces , Netflix prize etc.

In terms of matrix A as image

- Matrix X is the image in pixels . size = m*n as shown in image .
- U is column vector matrix which contains column information   of X. in here first few columns are dominant columns which discribes most of the columns of X
- Vt is the row vector   matrix which contains row information   of X. in here first few rows   are dominant columns which discribes most of the rows of X, as eigen mixture

- Σ is a diagonal matrix which has only diagonal values with rest all the values being zero.This is called singular value which is arranged in decending order , it represents the relative imporatance and as well a scaling factor.
- Applications : image compression

In terms of A as geometery



- If matrix A for example as shown above is circle
- V^T does the unitary transformation preserving length , and origin to its Right eigen basis

- S is a scaling factor interms major and minor axis .
- U is does the unitary transformation preserving length , and origin to its left eigen basis basically changes orientations
- Application : cross product , and other algorithms

2.3 COMPUTING THE SVD :

Getting into the mathematical description of SVD,

Given A = [2 −1 2; 1 0 0] find the SVD:

Step 1: Finding the matrix A^TA.

$$\begin{bmatrix} 2 & -1 \\ 2 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 2 & 2 & 0 \\ -1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 5 & 3 & 0 \\ 3 & 5 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Step 2: Find its eigenvalue and eigenvectors.

First compute the determinate of (A

$TA - \lambda I)$

And solving for lambda where I is identity matrix.

$$\begin{bmatrix} 5 & 3 & 0 \\ 3 & 5 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 5-\lambda & 3 & 0 \\ 3 & 5-\lambda & 0 \\ 0 & 0 & -\lambda \end{bmatrix}$$

Finding the determinate of the resultant :

$$\begin{vmatrix} 5-\lambda & 3 & 0 \\ 3 & 5-\lambda & 0 \\ 0 & 0 & -\lambda \end{vmatrix} = 0 \begin{vmatrix} 3 & 5-\lambda \\ 0 & 0 \end{vmatrix} - 0 \begin{vmatrix} 5-\lambda & 3 \\ 0 & 0 \end{vmatrix} + (-\lambda) \begin{vmatrix} 5-\lambda & 3 \\ 3 & 5-\lambda \end{vmatrix}$$

$$\begin{vmatrix} 5-\lambda & 3 & 0 \\ 3 & 5-\lambda & 0 \\ 0 & 0 & -\lambda \end{vmatrix} = -\lambda \begin{vmatrix} 5-\lambda & 3 \\ 3 & 5-\lambda \end{vmatrix}$$

On computing the 2*2 matrix, we get :

$$\begin{vmatrix} 5-\lambda & 3 & 0 \\ 3 & 5-\lambda & 0 \\ 0 & 0 & -\lambda \end{vmatrix} = -\lambda((5-\lambda)(5-\lambda) - (3)(3))$$

$$\begin{vmatrix} 5-\lambda & 3 & 0 \\ 3 & 5-\lambda & 0 \\ 0 & 0 & -\lambda \end{vmatrix} = -\lambda(\lambda^2 - 10\lambda + 16)$$

Now, we can solve to find λ = 0 to find eigenvalues :

$$-\lambda(\lambda^2 - 10\lambda + 16) = -\lambda(\lambda - 2)(\lambda - 8)$$

Therefore the eigenvalues are 8,2,0.

Using this, we can find the important value of "σ" which is the square root of the eigenvalues.

$$\sigma_1 = \sqrt{8} = 2\sqrt{2} \qquad \textbf{and} \qquad \sigma_2 = 2$$

These values are the important values along the diagonal of matrix "Σ".

Step 3: Find the normalized version of corresponding eigenvectors to each eigenvalue.

To find an eigenvalue, replace λ with the corresponding eigenvalue in the equation $(A\ TA - \lambda I)$. Then find the nullspace of that resulting matrix.

When λ = 8, the resulting matrix yields:

$$\begin{bmatrix} -3 & 3 & 0 \\ 3 & -3 & 0 \\ 0 & 0 & -8 \end{bmatrix}$$

To find the nullspace of the matrix, we need to find some vector "$\vec{}$" that when multiplied by the matrix, yields a zero vector.

$$\vec{v}_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

The above vectoe when multipied with the matrix, yields a zero vector.

The normalized version of above vector is obtained by multiplying the reciprocal of the square root of the sum of squared rows. i.e,

$$\vec{v}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

Similarly for λ = 2 and when λ = 0.

When λ = 2, the resulting matrix yields :

$$\begin{bmatrix} 3 & 3 & 0 \\ 3 & 3 & 0 \\ 0 & 0 & -2 \end{bmatrix} \longrightarrow \vec{v}_2 = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix} \longrightarrow normalized \longrightarrow \vec{v}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}$$

When λ = 0, the resulting matrix yields:

$$\begin{bmatrix} 5 & 3 & 0 \\ 3 & 5 & 0 \\ 0 & 0 & 0 \end{bmatrix} \rightarrow \vec{v}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \rightarrow normalized \rightarrow \vec{v}_2 = \frac{1}{\sqrt{1}} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \rightarrow or\ just \rightarrow \vec{v}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Step 4: Find the term U

This can be found by the equation

$$\rightharpoonup = \sigma u \rightharpoonup\rightharpoonup \text{ or } 1\ \sigma \rightharpoonup = u \rightharpoonup\rightharpoonup .$$

$$\frac{1}{\sigma}A\overrightarrow{v_1} = \overrightarrow{u_1} : \frac{1}{2\sqrt{2}}\begin{bmatrix}2 & 2 & 0\\ -1 & 1 & 0\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\\0\end{bmatrix} = \overrightarrow{u_1}$$

$$\frac{1}{4}\begin{bmatrix}4\\0\end{bmatrix} = \overrightarrow{u_1} \longrightarrow \begin{bmatrix}1\\0\end{bmatrix} = \overrightarrow{u_1}$$

Calculating the next term:

$$\frac{1}{\sigma}A\overrightarrow{v_2} = \overrightarrow{u_2} : \frac{1}{\sqrt{2}}\begin{bmatrix}2 & 2 & 0\\ -1 & 1 & 0\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}-1\\1\\0\end{bmatrix} = \overrightarrow{u_2}$$

$$\frac{1}{2}\begin{bmatrix}0\\2\end{bmatrix} = \overrightarrow{u_1} \longrightarrow \begin{bmatrix}0\\1\end{bmatrix} = \overrightarrow{u_2}$$

Hence ,

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

Σ term can be found by placing   terms along the diagonal of the matrix, ant then filling in the rest of the matrix as 0.

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

Step 5: Finding V^T

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

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

So now we have the finished equation $= U\Sigma V^{T}$, yields:

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

Now the computaion is done , this showed how U N and sigma are calculated. Moving on to one of the applications image compression.

3. Image compression

3.1.Introduction:

When you look at the color white on your screen, you’re not actually looking at white, and the same thing for the color yellow.

There is actually no white or yellow pigment in your screen. What you are looking at is a mixture of the colors red, green, and blue displayed by extremely small pixels on your screen.These pixels are displayed in a grid like pattern, and the saturation of each pixel tricks your braininto thinking it’s a different color entirely when looked at from a distance.

These red, green, and blue pixels range in saturation on a scale of 0 to 255; 0 being completelyoff, and 255 being completely on. They can also be written in hexadecimal format like #F5C78A for example. In hexadecimal, A is the value 10, and F is the value 15, therefore 0F = 15 and A0 =16. The first two numbers in this string of numbers represents the red value, the next tworepresenting the green value, and the final two representing the blue value. To put reference into what these are doing, here are some easy color examples:

#000000 = Black

#FFFFFF = White

#A0A0A0 = Gray

#FF0000 = Red

#00FF00 = Green

#0000FF = Blue

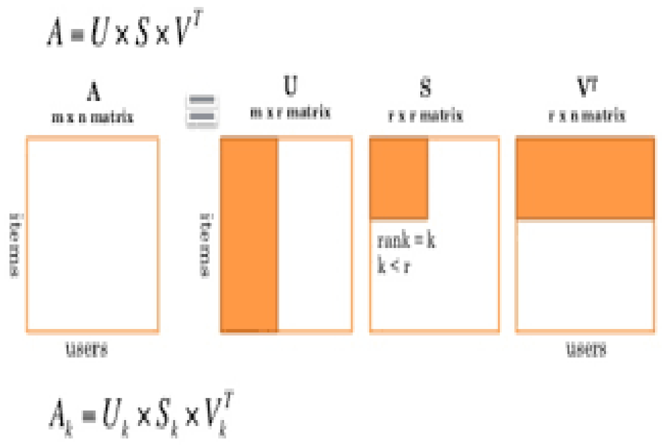
Because of a pixel’s grid like nature on your monitor, a picture can actually be represented asdata in a matrix. Let’s stick with a grayscale image for right now. To make an image gray, the values for red, green, and blue need to be the same. Therefore you can represent a pixel as having a value of 0 through 255 (in hexadecimal 00 through FF), and then repeating that valueacross the red, green, and blue saturation to get the corresponding shade of gray.

if image size=1280 × 1024 in which you would have to store 1,310,720 different pixel values! And that's if it was a grayscale image, if it was colored, it would be triple that, having to keep track of 3,932,160 different numbers, which if you think about one of those numbers equating to a byte on your computer, that equals 1.25MB for a grayscale image or 3.75MB for a colored image. Just imagine how quickly a movie would increase in size if it was updating at the rate of 30-60 frames per second.

What we can actually do to save memory on our image is to compute the SVD and then calculatesome level of precision.

3.2 Mathematical overview:

Consider this figure,



Expression: $\boldsymbol{A = USV^T}$ **[considering m>n]**

Matrix expantion: $A=u_1s_1v_1^T + u_2s_2v_2^T + \ldots \quad u_ns_nv_n^T + 0$

$$0 \quad + \quad ..m$$

$$A = U^*S^*V^T$$

this is called economy SVD , it contains only n values , which is only required .

Furthure it can be reduced to precision of rank r to compress image called trucate at r (best approximation of A) . given by Eckard young theorm .

$$A = \quad U'S'V^{T'}$$

i.e

$= [(m \times r)(r \times r)](r \times n)$

$= (m \times r)(r \times n)$

$= (m \times n)$

3.3. Saving memory using SVD

The matrix A requires m x n size where as in svd

$U' = m \times r$

$V^{T}{}' = r \times n$

$S' = r \times r$

total $= [(m \times r) + (r \times n) + (r)]$

Example: if we have a matrix 100×100, and we use a level of precision of 10 modes, we will find that our matrices are:

$U' = (100 \times 10)$,

$S' = (10)$,(But as S' is diognal element can be stored in 1D)

$V^{T}{}' = (10 \times 100)$

So now we are only keeping track of 2,010 different numbers instead of 10,000 which greatly decreases the storage of memory.

3.4. Python code to demonstrate image compression

3.4.1 Code :

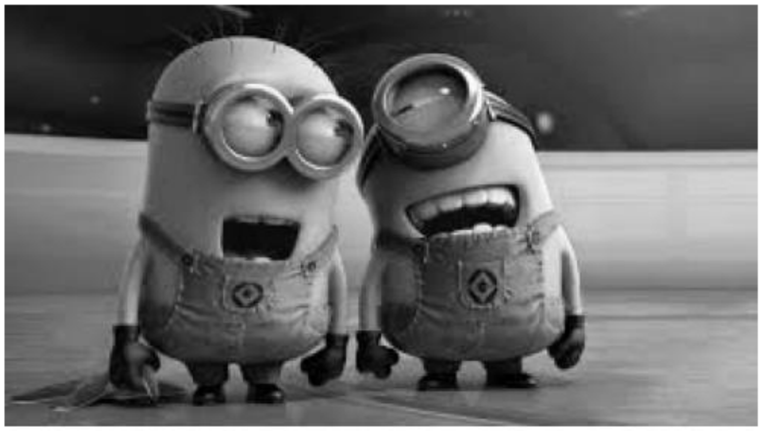
```
##############################################################
import os

a=imread('minion2.jpg')

x=np.mean(a,-1)
```

```
print('Shape of image = '+ str(x.shape))

print('Size of image =' + str(x.size))

img=plt.imshow(x)

img.set_cmap('gray')

plt.axis('off')

plt.show()


# performing SVD


u,s,vt=np.linalg.svd(x,full_matrices=False)

#economy svd extracting only first m columns of u


s=np.diag(s)


j=0

for r in (5,20,50,60,75):

        # approximation,

        xapprox=u[:,:r] @ s[0:r,:r] @ vt[:r,:]

        print('Rank ='+ str(r))
        print('Size'+ str((u[:,:r].size + s[0:r,:r].size +
vt[:r,:].size)) )

        plt.figure(j+1)

        j+=1

        img=plt.imshow(xapprox)

        img.set_cmap('gray')

        plt.axis('off')

        plt.title('r ='+ str(r))

        plt.show()


plt.figure(1)

plt.semilogy(np.diag(s))

plt.title('singular value')

plt.show()


plt.figure(2)

plt.plot(np.cumsum(np.diag(s))/np.sum(np.diag(s)))

plt.title('singular value:cumulative sum')

plt.show()
```

##

3.4.2 Result:

A)Original:

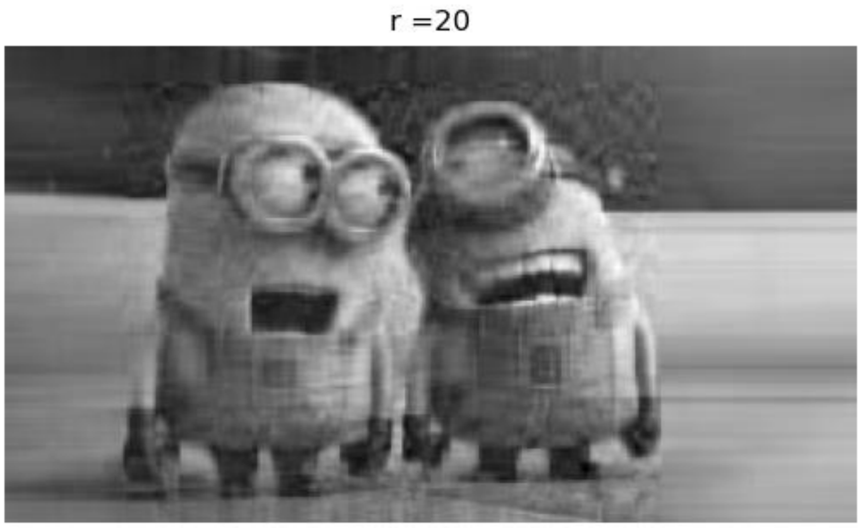


B)Approximation:

rank = 5

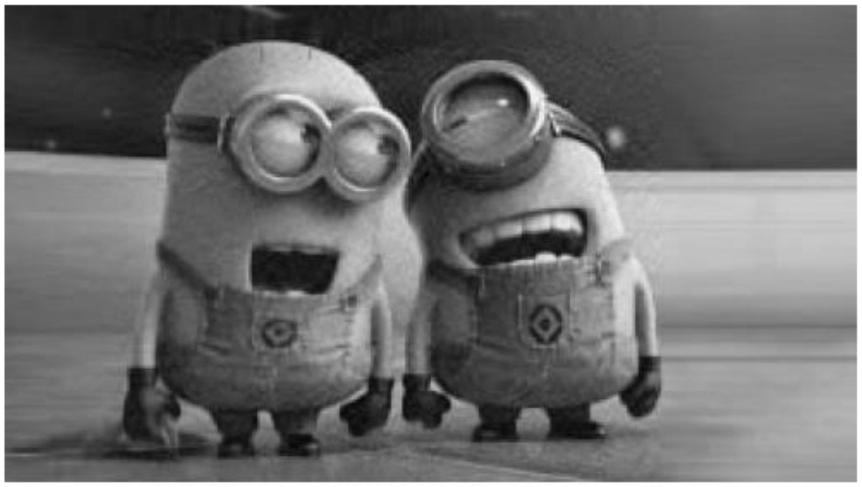
r =5

rank=20

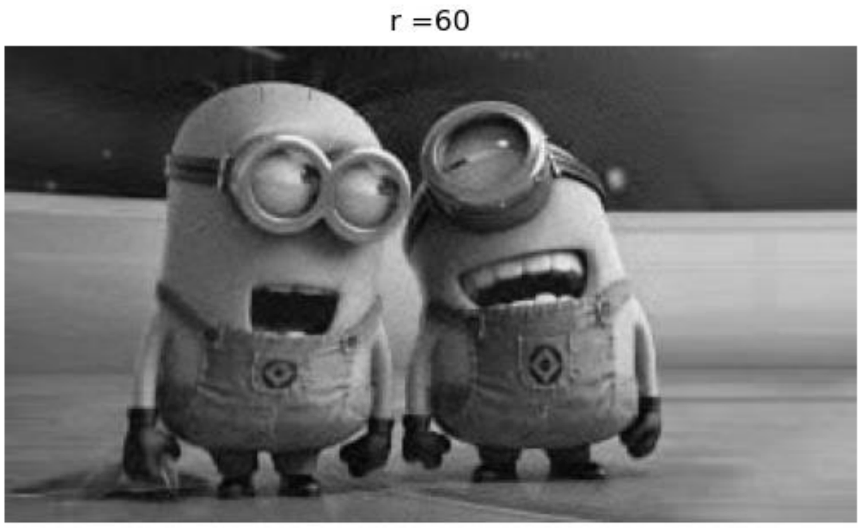


rank = 50

r =50



rank =60

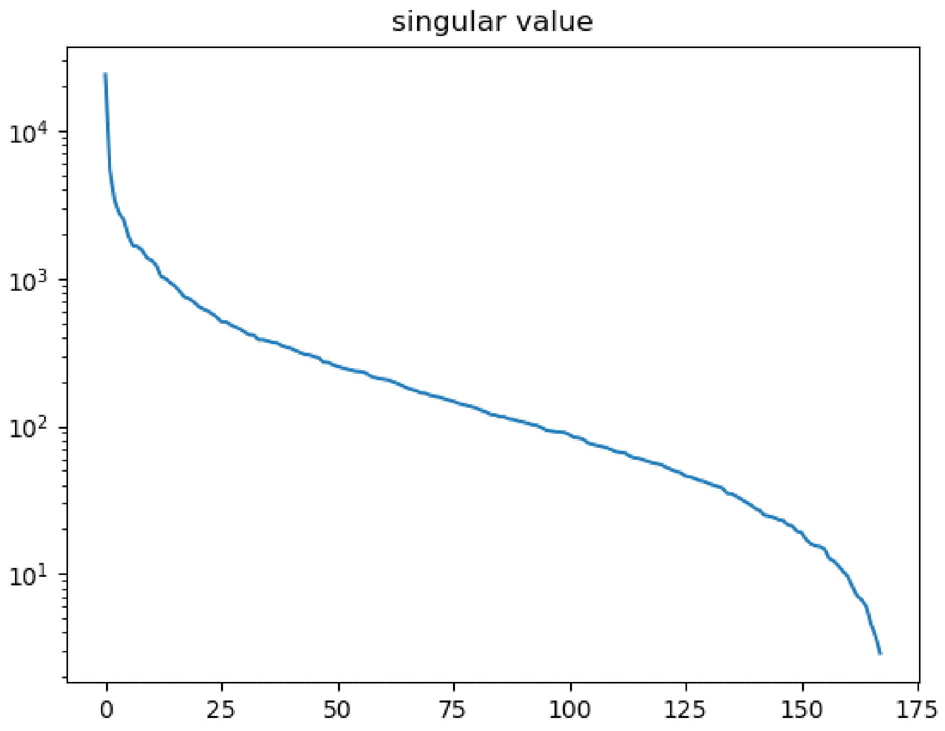


rank = 70

r =75

C)Graphs:

Singular Value(S) : equation = semilog(S_r)/r graph

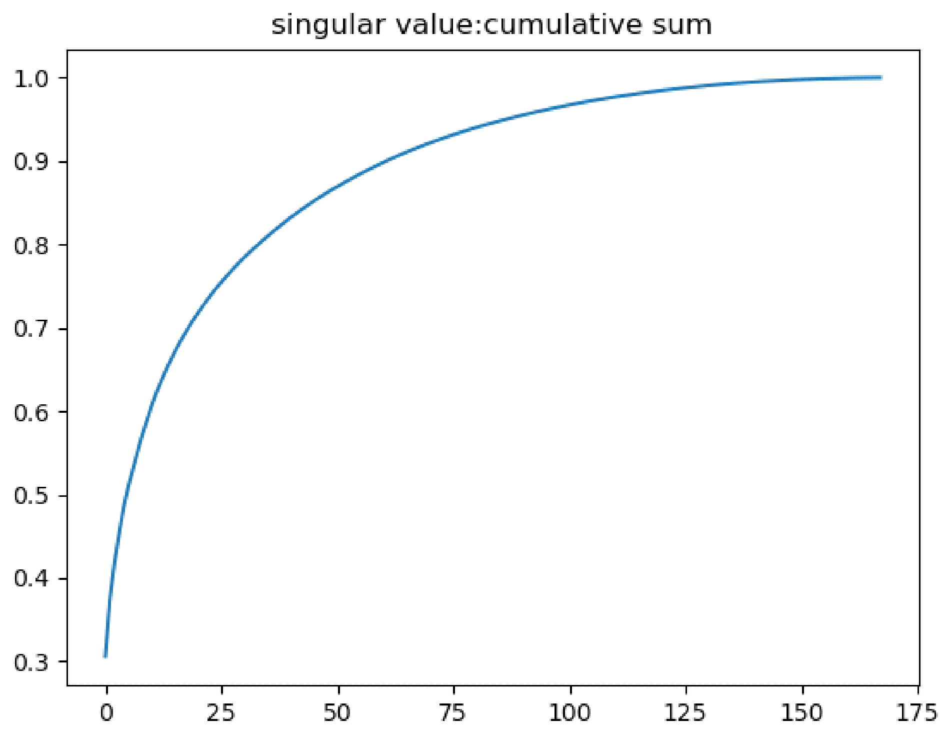


What this graph says ?

The narrow gap at first few ranks (r) like till 25 the precision is high i.e it represents major part of image ,the best approximation we get arrond 75

cummulative sum graph:

eq = cumulative sum(S_r)/sum(S)



What this graph says ?

This graph shows th error of approximated graph wrt original , arround 75 we get quite less error .

D) Output:

Shape of image = (168, 300)

Size of image =50400

Rank =5

Size=2365

Rank =20

Size=9760

Rank =50

Size=25900

Rank =60

Size=31680

Rank =75

Size=40725

Inference: storage optimization , reduced by 11KB

4. Conclusion

We saw what is svd and its varios application and also computed the 3 decomposed matrices . then as per our problem statement we explained and performed image compression , which inferred the following

All an image is, is data represented on a matrix being visually displayed to youthrough pixels. This data can be manipulated through the use of the SVD theorem to calculate a level of precision close to the original without storing as much data. The SVD allows us to store (#r)$(m + n)$ information instead of $(m \times n)$ information where the size of the image is $m \times n$, or

in case of coloured image its $3(\#modes)(m + n)$ when the image is in color instead of $3(m \times n)$.

5.References

1) https://youtu.be/H7qMMudo3e8

2) https://blog.statsbot.co/singular-value-decomposition-tutorial-52c695315254

3) http://www.math.utah.edu/~goller/F15_M2270/BradyMathews_SVDImage.pdf

4) https://www.cmi.ac.in/~ksutar/NLA2013/imagecompression.pdf

5) http://databookuw.com/databook.pdf by Steve brunton.