Applications of Singular Value Decomposition in Image Processing

Zijian Liu Chunbo Luo Qiang Li

January 4, 2019

Abstract

This paper discusses and analyzes the meaning and related properties of matrix singular value decomposition from the perspective of image processing, and applies it to image compression and reconstruction. According to the characteristics of singular value decomposition - dimensionality reduction, principal component projection is carried out to realize face image recognition and classification.

Contents

1 Introduction

With the development of computer technology and network technology, various kinds of multimedia information services have been increasingly integrated into people's production and life. Digital media, especially unstructured data (pictures, video, audio), has become the main form of media in the information society. As two-dimensional data, the large size of images has become a major obstacle to information transmission. In the process of communication and transmission, it is bound to consume a lot of resources if the complete transmission of images without distortion is to be realized. Therefore, image compression and reconstruction become the key to solve this problem.

The singular value decomposition of the matrix realizes the decomposition and dimensionality reduction of the matrix on each feature. By analyzing the principal components of the data, the lossy compression and reconstruction of the image are realized. At the same time, according to the characteristics of matrix singular value decomposition, principal component analysis is carried out on the data to realize the recognition and classification of face images.

The arrangement of this article is given as follow

1. This paper introduces the development trend of multimedia information, analyzes the existing problems of image transmission, and puts forward a method of image compression and reconstruction based on singular value decomposition. In addition, according to the feature of matrix singular value decomposition with characteristic dimensionality reduction, an image recognition and classification method based on singular value decomposition is proposed.

2. Introduce the meaning and related properties of singular value decomposition;

3. Introduce the method and implementation process of image compression and reconstruction based on singular value decomposition;

4. The method and implementation process of image recognition and classification based on singular value decomposition are introduced.

2 Singular Value Decomposition

In linear algebra, singular value decomposition is the decomposition of real (complex) matrices. Compared with eigenvalue decomposition, singular value decomposition extends the decomposed matrices from square matrices to more general non-square matrices. For any normal Matrix A of positive semidefinite (for example, Hermitian Matrix $A^H = A$), it is unitary similar diagonal Matrix, the eigenvalue decomposition is defined

$$A = U\Lambda U^H \tag{1}$$

where U denotes for the unitary matrix, Λ denotes for diagonal matrix, diagonal element for the eigenvalues of the matrix A. For a general m by n non-square matrix A, singular value decomposition is defined as

$$A = U\Sigma V \tag{2}$$

where the U is an m order unitary matrix, Σ denotes for $m \times n$ long diagonal matrix, V is an n order unitary matrix. And Σ satisfies

$$\Sigma = \begin{bmatrix} \Lambda & 0 \\ 0 & 0 \end{bmatrix} \tag{3}$$

Λ is an r order diagonal matrix, diagonal elements for matrix A are non zero eigenvalues. To non square matrix A right by A^H

$$AA^H = U\Sigma V(U\Sigma V)^H = U\Sigma VV^H\Sigma^H U^H = U(\Sigma\Sigma^H)U^H \tag{4}$$

Therefore, for AA^H matrix eigenvalue decomposition, we get the relationship between the eigenvalue and singular value $\Lambda = \Sigma\Sigma^H$, namely the $\lambda = \|\sigma\|^2$.

The singular value vector of the matrix is unique, which characterizes the distribution characteristics of the data of the matrix. The singular value decomposition formula of A can be written as

$$A = U\Sigma V = U \begin{bmatrix} \Lambda & 0 \\ 0 & 0 \end{bmatrix} V = \sum_{i=1}^{r} A_i = \sum_{i=1}^{r} \sigma_i u_i v_i \tag{5}$$

where u_i and v_i are column vectors of unitary matrices U and V, respectively. And σ_i is the non-zero singular value of A. A digital image is a matrix with time frequency information, and each A_i represents the time frequency information. If the energy of the image is represented by the square of the Frobenius norm of the matrix, it can be decomposed by the singular value of the matrix

$$\|A\|_F^2 = \mathrm{Tr}(AA^H) = \sum_{i=1}^{r} \|\sigma_i\|^2 \tag{6}$$

The energy of the image is reflected in the magnitude of the modulus of the singular value. The energy of each decomposed component of the matrix can be obtained by descending ordering according to the magnitude of different singular values.

To understand the singular value decomposition of matrices geometrically, we can first understand the linear transformation.

Set any nonzero vector $X = (x_1, x_2) \in R^2$, after Ω get linear rotation vector Y, as shown in Figure 1

$$Y = \Omega X = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{7}$$

Similarly, if the unit element coordinates $X = (x_1, x_2) \in C = (x_1, x_2)|x_1^2 + x_2^2 = 1$linear transform, after transformation of vector. See Figure 2

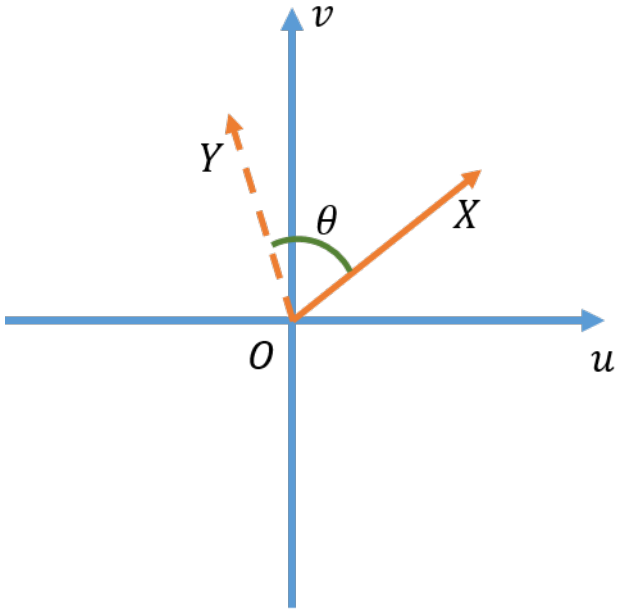

Figure 1: Linear rotation of a vector

$$Y = \Omega X = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{8}$$

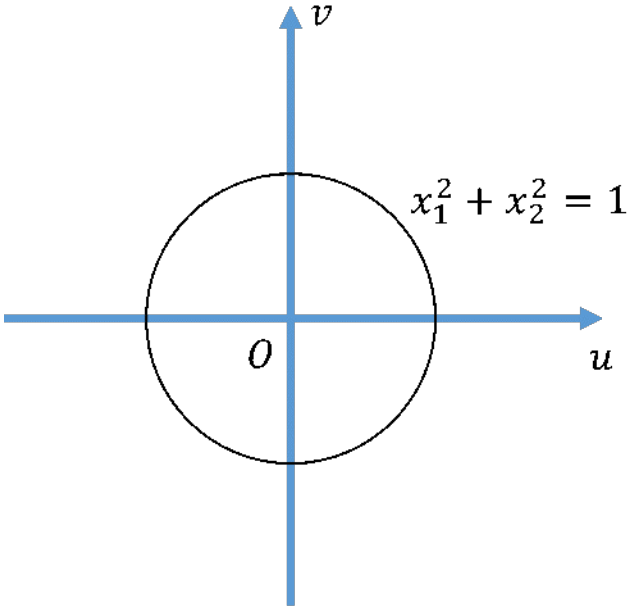

Figure 2: Unit Circle

Get the transformed graph (see Figure 3)

$$y_1 = ax_1, \quad y_2 = bx_2 \tag{9}$$

$$\frac{y_1^2}{a^2} + \frac{y_2^2}{b^2} = 1 \tag{10}$$

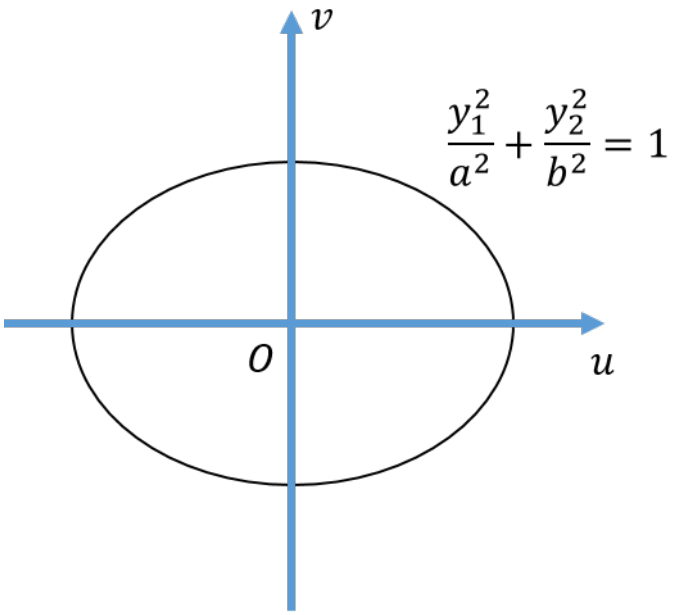

Figure 3: Ellipse obtained by linear transformation

For this transformation matrix, it is easy to know that the parameters a and b are the singular values of the 2rd-order linear transformation matrix ($\sigma_1 = a, \sigma_2 = b$). By linear transformation, the data distribution is better presented on the long axis of the ellipse. Therefore, it is usually possible to select the long axis of the ellipse as the main component of data analysis to realize the simple dimensionality reduction of data. The same is true for high-dimensional data. After transformation, high-dimensional data features of dimensionality reduction are obtained. Several main components of the data are used to replace the research of high-dimensional data itself, which makes the features more obvious and more convenient for analysis.

3 Image Compression and Reconstruction

Image information has become more and more the main form of information in production and life. The image itself contains a lot of information. However, due to the huge size of the image, a large amount of communication resources are needed for the transmission of the image. In order to solve this problem, we need to compress the image lossy and reduce the image size as much as possible on the premise of high definition.

For an $m \times m$ gray image, the size of m^2 bytes

$$Volume(A) = m * m * 8/8 = m^2 \quad \text{(byte)} \tag{11}$$

If it is decomposed by singular value, arranged in descending order of singular value module, and the first r components are taken, then

$$A = \sum_{i=1}^{m} u_i \sigma_i v_i \approx \sum_{i=1}^{r} u_i \sigma_i v_i = \hat{A} \tag{12}$$

In the process of transmission, we only need to transport the decomposition of principal components on the unitary matrix singular value $\hat{U}, \hat{V}$ and diagonal matrix diagonal element of $\sigma_i (i = 1, ..., r)$, then

$$Volume(\bar{A}) = mr + r + rm = (2m + 1)r \quad \text{(byte)} \tag{13}$$

And the image of $\bar{A}$ retained for the energy of the original image size

$$\eta = \frac{\sum_{i=1}^{r} \|\sigma_i\|^2}{\sum_{i=1}^{m} \|\sigma_i\|^2} \tag{14}$$

In order to ensure the compression of the image (the volume is smaller than the original image), it needs to be satisfied

$$(2m + 1)r < m^2 \tag{15}$$

Then

$$r < \frac{m^2}{2m + 1} \approx \frac{m}{2} \tag{16}$$

Carry out singular value decomposition for the image of $m = 500$, and take 25, 50, 100, 150 and 200 principal components, respectively, for image reconstruction. The reconstructed image and the original image are shown in Figure 4

As can be seen from the figure, when the principal components are less, the image volume is smaller, the energy ratio is smaller, and the reconstructed image is more fuzzy and distorted. With the increase of principal components, the image volume and energy ratio become larger, and the reconstructed image is gradually close to the original image with better results. See Table 1

Meanwhile, PSNR is used to represent the clarity of the image (to be exact, the closeness to the original image). Therefore, it can be considered to seek the image compression as much as possible under the allowable PSNR threshold.

4 Face Recognition

Face recognition, especially face recognition based on deep learning, has gradually become an essential function in production and life. According to the geometric characteristics of singular value decomposition, this paper decomposes the data into several independent

Figure 4: SVD based reconstruction image and original image

Table 1: The volume, PSNR and energy radio of reconstruction image

Selected SV	Volume (KB)	PSNR	Energy Radio
25	24	26.82	69%
50	49	31.11	80%
100	98	37.75	90%
150	147	43.73	95%
200	195	49.44	98%
500 (Original)	244	–	100%

orthogonal components, and takes the main components as the research object to realize face recognition and classification.

This paper takes ORL face database as an example to study. The database contains 40 subjects, each with 10 pictures of different expressions from different angles. And partial images are shown in Figure 5.

Because the face image itself is too complex, we take the image variance matrix as the research object (shown as Figure 6).

$$A = (X - \bar{X})^T (X - \bar{X}) \tag{17}$$

Perform singular value decomposition

$$(X - \bar{X}) = U \Sigma V \tag{18}$$

The feature basis vector was constructed by selecting 90% principal components

Figure 5: ORL face data (partial images)

Figure 6: Feature face (partial images)

$$Base = (X - \bar{X})^T U\Sigma = (X - \bar{X})^T (X - \bar{X})V^T = AV^T \tag{19}$$

By projecting the training data and the test data to the characteristic direction, the dimensionality reduction matrix is obtained

$$\begin{aligned} LowTrain &= (Train - \bar{Train}) * Base \\ LowTest &= (Test - \bar{Test}) * Base \end{aligned} \tag{20}$$

By comparing the distance of the matrix (Figure 7), the test data are classified into the training data of the nearest distance.

$$Dist = norm(LowTrain, LowTest) \tag{21}$$

In this paper, the data was divided equally, half as training and half as test, and the recognition rate was 94.50%.

5 Conclusion

This paper combines the knowledge of singular value decomposition learned in matrix theory and applies the theory to image compression reconstruction and recognition classification. The main components of the image are extracted by singular value decomposition of the matrix, so as to realize image compression. The corresponding (left, right) eigenvectors are multiplied to reconstruct the image. Moreover, with the increase of the number of selected principal components, the size of the image becomes larger, and the resolution and energy ratio of the image gradually improve. According to the geometric characteristics of singular value decomposition, the image is mapped to the feature vector of the principal component

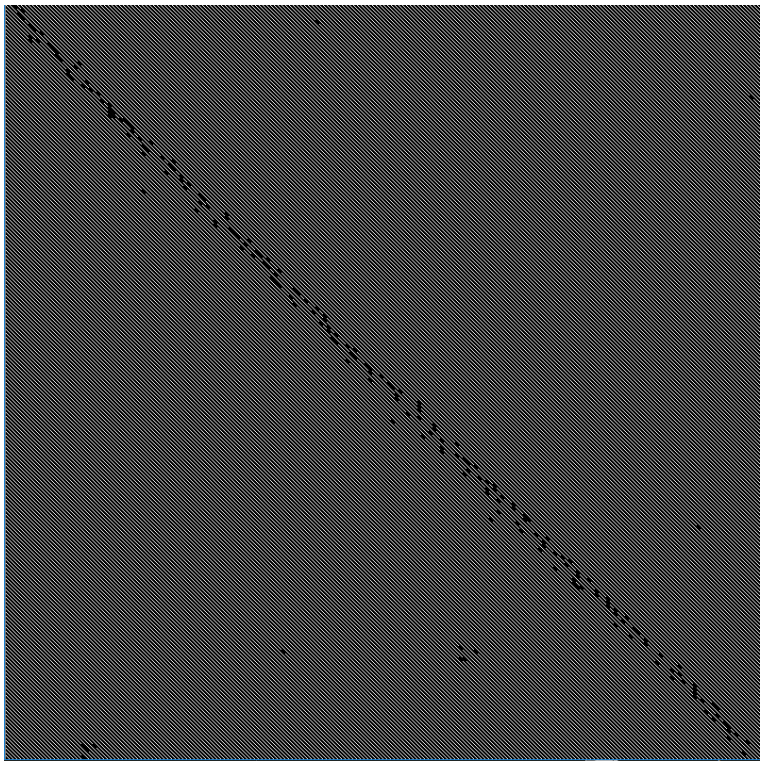

Figure 7: Distance Matrix (200×200)

to realize the dimensionality reduction of the image matrix. The face image is recognized and classified by comparing the distance between the training image and the test image.

Source Code

Code URL: https://github.com/liu6zijian/SVD

```
# python code: SVD for image compression and construction
# -*- coding: utf-8 -*-
"""
Created on Wed Nov 21 11:35:02 2018

@author: Lzj
"""

import cv2
import numpy as np
#import matplotlib.pyplot as plt

Img = cv2.imread('lena.jpg')
#cv2.imshow('Lena',Img)
```

```
#cv2.waitKey(0)
row, col, dim = Img.shape
u = list()
sigma = list()
v = list()

for i in range(dim):
   u0, sigma0, v0 = np.linalg.svd(Img[:,:,i])
   u.append(u0)
   sigma.append(sigma0)
   v.append(v0)

u = np.array(u)
v = np.array(v)
sigma = np.array(sigma)
ImgNew = np.zeros(Img.shape)

def eigValPct(eigVals,percentage):
   sortArray=np.sort(eigVals)
   sortArray=sortArray[::-1]
   arraySum=np.sum(sortArray)
   tempSum=0
   num=0
   for i in sortArray:
      tempSum+=i
      num+=1
      if tempSum>=arraySum*percentage:
         return num

def psnr(target, ref, scale):
   # target:            ref:            scale:
   # assume RGB image
   target_data = np.array(target)
   #    target_data = target_data[scale:-scale,scale:-scale]

   ref_data = np.array(ref)
   #    ref_data = ref_data[scale:-scale,scale:-scale]

   diff = ref_data - target_data
   diff = diff.flatten('C')
   rmse = np.sqrt( np.mean(diff ** 2.) )
   return 20*np.log10(255.0/rmse)


def SVDpro(ImgNew,dim,N,u,v,sigma):
```

```
    for i in range(dim):
        Diag = np.diag(sigma[i,:N])
        tmp = np.dot(u[i,:,:N], Diag).dot(v[i,:N,:])
        ImgNew[:,:,i] = tmp
    ImgNew[ImgNew<0] = 0
    ImgNew[ImgNew>255] = 255
    Psnr = psnr(ImgNew[:,:,0],np.float32(Img[:,:,0]),8)
    ImgNew = np.uint8(ImgNew)
    print(Psnr)
    cv2.imshow('result',ImgNew)
    cv2.waitKey(0)
    cv2.imwrite('./construction/'+str(N)+'.png',ImgNew)

    for j in range(10):
        N = int(col * (j+1) * 1.0/20)
        SVDpro(ImgNew,dim,N,u,v,sigma)
    cv2.destroyAllWindows()


# python code: SVD for face recognition
# -*- coding: utf-8 -*-
"""
Created on Wed Nov 21 11:35:02 2018

@author: Lzj
"""
import numpy as np
import cv2

def readImg(path):
    Img = list()
    for i in range(40):
        for j in range(10):
            subPath = path + 's'+str(i+1) +'/'+ str(j+1)+'.pgm'
            tmp = cv2.imread(subPath)
            tmp = cv2.cvtColor(tmp,cv2.COLOR_BGR2GRAY)
    #           cv2.imshow('1',tmp)
    #           cv2.cv2.waitKey(0)
            tmp = tmp.reshape((-1,))
            Img.append(tmp)
    return np.array(Img)

path = './att_faces/'

Img = readImg(path)
```

```
Train = Img[0::2]
Test = Img[1::2]
ImgFloat = Train.astype(np.float32)
ImgM = ImgFloat
ImgMean = Train.mean(axis=1)
for i in range(200):
   ImgM[i,:] = ImgFloat[i,:] - ImgMean[i]

sigma = np.dot(ImgM, ImgM.T)
v, d = np.linalg.eig(sigma)


def eigValPct(eigVals,percentage):
   sortArray=np.sort(eigVals) #
   sortArray=sortArray[::-1] #
   arraySum=np.sum(sortArray) #
   tempSum=0
   num=0
   for i in sortArray:
      tempSum+=i
      num+=1
      if tempSum>=arraySum*percentage:
         return num

def Acc(TestImg, FeaImg):
   cnt = 0
   for j in range(200):
      mdist = np.zeros((200,))
      for i in range(200):
         mdist[i] = np.linalg.norm(TestImg[j,:] - FeaImg[i,:])
   #    dist = np.sort(mdist)
      Ind = np.argmin(mdist)
      I = np.floor(Ind/5) + 1
      l = np.floor(j/5) + 1
      if I == l:
         cnt += 1
   return cnt/200

k = eigValPct(v,0.9)
eigValInd = np.argsort(v)
eigValInd = eigValInd[:-(k+1):-1]
redEigVec = d[:,eigValInd]

#lowImg = np.dot(sigma, redEigVec)
base = np.dot(ImgM.T,redEigVec).dot(np.diag(np.power(v[:k],-0.5)))
```

```
FeaImg = np.dot(ImgFloat,base)

TestImg = np.dot(Test,base)
acc = Acc(TestImg, FeaImg)

print('The accuracy is %.2f%%'%(acc*100))
```