Singular Value Decomposition on Image Compression

Jiaqi Zhang
University of Washington
zjq95@uw.edu

Abstract

This project applies “Singular Value Decomposition (SVD)” on image compression. SVD transforms a matrix A into the product of three matrices *U, S, V^T* through rotation and stretching. It allows us to reconstruct the original image, i.e. represent the image with a matrix of smaller dimension. The compressed image should have satisfactory quality but requires less space of storage.

I. Introduction and Overview

Grayscale images can be represented by 2D matrices. Dimension of the matrix corresponds to size of the image, i.e. an image of (m x n) pixels is represented by a matrix with *m* columns and *n* rows. The magnitude of each value carries the intensity information. For example, each grayscale image can be represented as a matrix *M*, where M_{mn} is the gray intensity at column *m* and row *n*[1].

The purpose of this homework is to demonstrate the usage of Singular Value Decomposition (SVD) on Image Compression. SVD factors a matrix *M* into three new matrices *U*, *S*, and *V*, in a way that $M = USV^T$. *U* and *V* are orthonormal matrices and *S* is a diagonal matrix which contains *M*’s singular values. This report will give detailed explanations of the compression process. Various terms of singular values will be tested on the same image to show the effects of the compression.

II. Theoretical Background

i. SVD

In linear algebra, SVD is the factorization of any real or complex matrices. For a matrix *M* of dimension (*m* x *n*), the SVD factorization of matrix *M* is of the form:

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* \tag{1}$$

U is a *m x m* unitary matrix.
Σ is a *m x n* diagonal matrix, with non-negative decreasing numbers on the diagonal.
*V, V** are *n x n* unitary matrices.

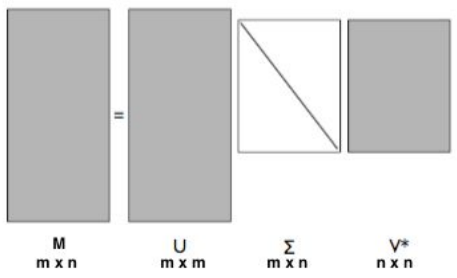


Figure 1. Illustration of the reduced SVD decompostition.

The diagonal entries σ_i of Σ are known as the singular values of M. $\sigma_1, \sigma_2, \ldots, \sigma_n$ are unique and it can be proven that

$$\sigma_1 \geq \sigma_2 \geq \sigma_3 \ \ldots, \geq \sigma_n > 0 \tag{2}$$

The rank of the matrix M is equal to the number of nonzero singular values. By eliminating smaller singular value, or higher ranks, we are able to reduce noise and thus compress the image.

The detailed SVD decompostition can be expressed as:

$$M_r = USV^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T \tag{3}$$

which is equivalent to

$$M_r = \sigma_1 u_1 v_1^T + \ldots + \sigma_r u_r v_r^T \tag{4}$$

The singular values that are small enough are dropped in order to compress the image. Note that singular values are ordered from large to small on the diagonal of *Σ.* Assume that the *(k+1)*th singular value is small enough to be truncated, then the compressed matrix of rank k can be represented by:

$$M_k = \sigma_1 u_1 v_1^T + \ldots + \sigma_k u_k v_k^T \tag{5}$$

In this report, different *K*s are tested and results will be shown in section IV.

ii. Measurement of Image Compression

To measure the quality of the compression, we calculate how much of the original image is restored by the compressed image, in other words, the ratio between the sum of singular values:

$$energy = \frac{\sum_{k}^{n=1} \sigma_n}{\sum_{r}^{n=1} \sigma_n} \tag{6}$$

To measure the compression performance, we can compute the compression ratio:

$$C_k = \frac{\frac{k \cdot (m \cdot n)}{r}}{m \cdot n} = \frac{k}{r} \tag{7}$$

Smaller C_k is, better the performance is.

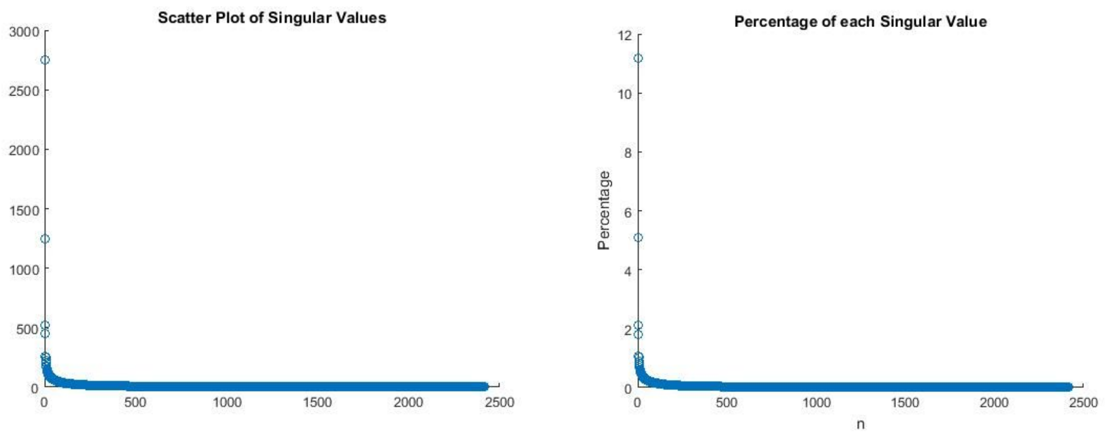
III. Algorithm Implementation and Development

The database we work with is the Cropped Yale Faces Database. This database contains 2414 grayscale images in pgm format. First of all, we load the files into Matlab. Each file is loaded as a matrix of size (192 x 168). Then we reshape the images into column vectors and put them together in a huge matrix *B*. This way, we can treat this set of images as a single large image. Note that we need to convert the values into double in order to visualize the image in Matlab[2]. The complete matrix *B* has dimension (32256 x 2414):

$$M1 = \begin{bmatrix} M_{1,1} & M_{1,2} & \dots & \dots & \dots \\ M_{2,1} & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ M_{192,1} & \dots & \dots & \dots & M_{192,168} \end{bmatrix} \rightarrow B = \begin{bmatrix} M_{1,1} & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ M_{1,168} & \dots & \dots & \dots \\ M_{2,1} & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ M_{192,168} & \dots & \dots & \dots \end{bmatrix} \tag{8}$$

Then, we apply SVD on matrix *B* and obtain matrices *U, S*, and V^T.

The columns of *U* represent an eigenface, which is a generic face collected from all 2414 images. *V* can be reviewed as the rotation matrix, it contains features from different spaces. *S*, singular values of *B*, stands for the weightings of each feature applied to the eigenface. The combination of first column of *U* and the first singular value gives the most basic generic face. As more singular values are used, more features are added to the eigenface and the compressed image would become more and more like the original image.



(a) **(b)**

Figure 2. (a) Regular Plot of σ_n (b) Scatter Plot ofσ_n

Observing the diagonal of *S* (Figure 2), we can see that there is extraordinarily rapid decline and the first 4 singular values are significantly larger than the rest (2-a). From the percentage plot (2-c) we can see that the first 4 singular values produces 20% of the energy. This indicates that the first 4 the rest. However to achieve satisfactory accuracy, we would need potentially hundreds of modes.

Now, we choose multiple *K* terms to test the compression performance adn quality. We set the values that are smaller than the *k*th singular value to zero and reconstruct the matrix:

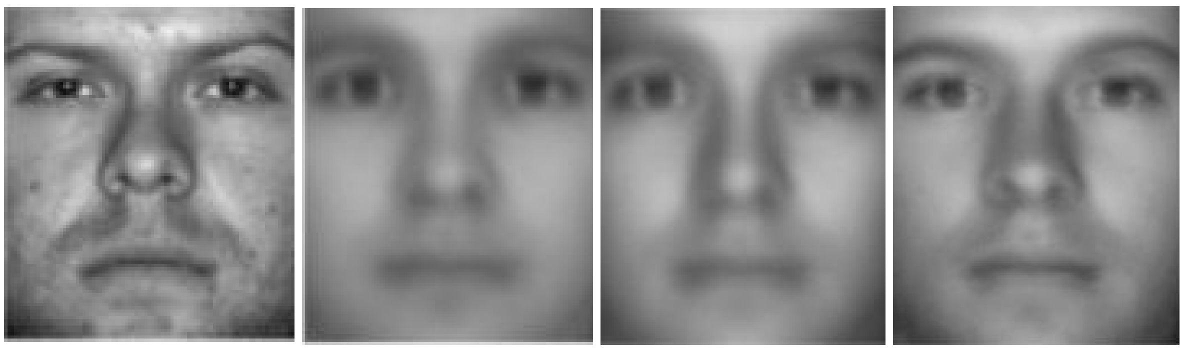
$$S(k+1:end, k+1:end) = 0 \tag{9}$$

$$B_k = US_kV^T \tag{10}$$

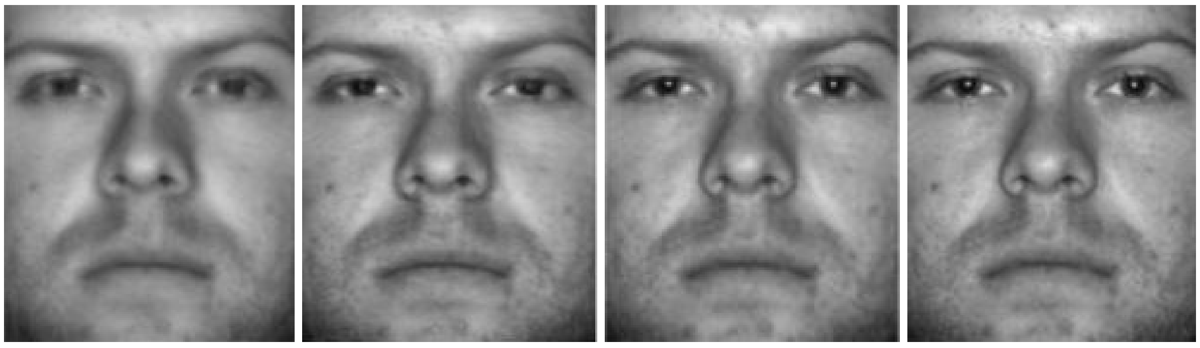
Then we reshape the column vector into a matrix and visualize the image.

IV. Computational Results

Figure 3 shows examples of compressed images under different K. a) is the original image, b) shows the result using 1 singular value, c) shows the result using 20 singular values and so on. Table 1 shows a summary of the compressed storage of the image set, compression performance and the *energy,* which indicates the compression quality.



(a)Original Image **(b) K = 1** **(c) K = 4** **(d) K = 20**



(e) K = 94 **(f) K = 200** **(g) K = 1131** **(h) K = 2165**

Figure 3. Examples of Compreddes Images.

Table I. Summary of Result.

K	Storage Space (Bytes)	Compression Performance (Ratio)	Compression Quality (Energy)
1	13	0.000414	0.1118
4	52	0.001657	0.2018
20	267	0.00828	0.3104
94	1222	0.03894	0.50
170	2210	0.070	0.5934
1131	14703	0.4685	0.90
2165	28145	0.8965	0.99

Observations from results in Table I:

1. Storage space is smaller when fewer singular values (smaller *K*) are used while quality is better when more singular values (larger *K*) are used.
2. When *K* = 1, the image is blurry and hard to recognize. From *K* = 1 to 4, the compressed image gains facial features rapidly.
3. When *K* = 94, the compressed image takes only 3.89% of the original storage space but restores half of the features of the original image.
4. The compressed image starts having high similarity to the original image when *K* = 170. At this point, it is compressed to 7% of the original size and restores 59.34% features.
5. The compressed image restores 90% of the original image at rank 1131 and 99% at rank 2165.

V. Summary and Conclusion

This homework applied Singular Value Decomposition (SVD) to Image Compression. It decomposes the image matrix A into three matrices:

$$A = USV^T \tag{1}$$

U gives a basic eigenface, *V* represents features in different spaces and *S* gives the weighting of each feature.
From the result, we can conclude that rank 170 is the ‘Best Rank’. At rank 170, the image takes 7% of the original storage and restore 59.34% of the features. The compressed image at this rank is close enough to the original image and the face can be easily recognised. Based on the theory and result, we conclude that SVD provides a good compression performance while retaining the image quality.

VI. Bibliorgraphy

[1] S. Johnson. *Digital Photography*. 2006. ISBN 0-596-52370-x.
[2] Math Works Inc. *Conver Image to Double Precision.* Math Works Inc Support. 2016.
[3] J. Chen. *Image Compression with SVD.* ECS 289K Scientific Computation. Dec. 13, 2000.

Appendix A - MATLAB Functions Used

‘dir’
Imports images from file to Matlab.
‘imread’
Reads image from graphic file into a matrix.
‘im2double’
Convert image to double precision.
‘reshape’
Reshapes array. B = reshape (A, [m, n]) reshapes array/matrix A into a matrix of dimension (*m x n*) and stores in B.
‘svd’
[U, S, V] = svd(A) performs a singular value decomposition of matrix A.
[U, S, V] = svd(A, ‘econ’) produces an economy-size decomposition of (*m x n*) matrix A.
‘scatter’
Creates a scatter plot with dots at the locations specified.
‘diag’
Puts the diagonal of matrix and put it in a column vector.
‘imshow’
Displays a grayscale image in a figure.