Introduction to Singular Value Decomposition with Image Compression Application

Kevin C. Cheng , Sabrina I. Thompson, John P. Watson

***Abstract*—The Singular Value Decomposition (SVD) is based on the eigenvalue and eigenvector analysis of matrices. This paper details the theory and method of computation for SVD and demonstrates its application to image compression. The motivation behind image compression is to decrease the amount of space needed to store an image while maintaining visual integrity.**

I. INTRODUCTION

Eigenvalues and eigenvectors are concepts in linear algebra that are heavily applied in a wide range of real-world contexts, including image compression. We will begin with a discussion of the mathematics behind eigenvalues and eigenvectors at a fundamental level. We will then detail the use of these constructs in the singular value decomposition of real-valued square matrices. Finally, we will explain its application to image compression.

The goal of image compression is to reduce the number of bytes associated with an image to make it easier to save or transmit. Using singular value decomposition we can break down the image and compress its data by preserving only the most important parts of the image.

II. EIGENVALUES, EIGENVECTORS

Given a matrix, A, an eigenvalue is any λ that satisfies the following property, where $\mathbf{x}$ is a non 0 vector:

$$A\boldsymbol{x} = \lambda\boldsymbol{x} \quad (1)$$

For each eigenvalue, there are one or more corresponding eigenvectors, $\boldsymbol{x}$. Though the trivial solution of $\lambda = 0$ with $\boldsymbol{x} = \boldsymbol{0}$ is not technically an eigenvalue and vector pair; it is possible to have an eigenvalue $\lambda = 0$ with a nontrivial accompanying eigenvector such that the eigenvectors are in the nullspace of A.

The simple implication of this statement is that there can exist vector(s) for a matrix such that the transformation of that vector by A only modifies the vector's size or inverts its direction. Thinking of eigenvectors and eigenvalues in terms of transformations, eigenvectors determine the directions along which the transformations will occur, whereas the eigenvalues will determine how the object is scaled along those directions. Based on eigenvector and eigenvalue analysis, the transformation that the matrix performs can be analyzed. Eigenvalues and vectors can be complex, but for the purposes of this paper we will be examining cases with real parts only.

III. SINGULAR VALUE DECOMPOSITION

Singular value decomposition is a technique used to factorize a matrix into 3 component matrices. This factorization is of the following form for a real-valued matrix M:

$$M = U\Sigma V^T \quad (2)$$

In this form, U is a matrix whose columns are orthonormal eigenvectors of MM^T (these mutually orthogonal vectors of unit length are called the *left singular vectors*), and V^T is the transpose of a matrix whose columns are orthonormal eigenvectors of M^TM (called the *right singular vectors*). Σ is a diagonal matrix whose non-zero values are the square roots of the eigenvalues of M^TM or MM^T (their eigenvalues are the same). These values, known as the non-zero singular values of M, are placed into the columns of Σ in descending order, such that the square root (absolute value) of the largest eigenvalue of M^TM or MM^T is placed in the first column of Σ, while the smallest eigenvalue is placed in the last column. This ordering is done by convention in order to keep the singular values in Σ in consistent columns with the corresponding singular vectors in U and V, and is also useful for applications, as will be discussed in the next section.

If we were to look at a geometric representation of SVD, the resulting matrices from this decomposition would represent different transformations. Specifically, U and V^T are rotational transformations, while Σ is a scaling transformation. The net effect of rotating by V^T, scaling by Σ, and then rotating by U is equivalent to that of a matrix transformation by M.

Consider a 2x2 matrix M:

$$M = \begin{bmatrix} 1 & 2 \\ -2 & -1 \end{bmatrix}$$

To construct the U matrix, find the eigenvalues and corresponding eigenvectors for MM^T.

$$MM^T = \begin{bmatrix} 5 & -4 \\ -4 & 5 \end{bmatrix}$$

The defining equation (1) can be manipulated as follows and used to solve for eigenvalues by finding all scalars λ that satisfy the equation.

$$(A - \lambda I)\boldsymbol{x} = 0 \quad (3)$$

According to the Invertible Matrix Theorem[i], this can be done by setting the determinant of $(A - \lambda I)$ equal to zero and solving for the roots

$$det(A - \lambda I) = 0$$

$$det\begin{bmatrix} 5-\lambda & -4 \\ -4 & 5-\lambda \end{bmatrix} = (5-\lambda)(5-\lambda) - (-4)(-4) = 0$$

$$\lambda_1 = 9, \qquad \lambda_2 = 1$$

Next, finding the corresponding eigenvectors to the eigenvalues can be done by plugging the eigenvalues back in to Eqn. (3) and solving for **x**

$$\begin{bmatrix} 5-\lambda & -4 \\ -4 & 5-\lambda \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

$$\overrightarrow{e_1} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \qquad \overrightarrow{e_2} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

These vectors will make up the columns of the U matrix, but they must be orthonormal. In this example, they are already orthogonal. However, if this was not the case, the Gram-Schmidt Process for orthonormalization could be applied to the vectors. Normalizing the orthogonal vectors yields:

$$\overrightarrow{u_1} = \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}, \qquad \overrightarrow{u_2} = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

Thus, the U matrix is:

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

To find the *V* matrix, repeat this process by finding the eigenvectors of M^TM (the eigenvalues are the same as those of MM^T). This yields:

$$\lambda_1 = 9, \qquad \lambda_2 = 1$$

$$\overrightarrow{e_1} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \qquad \overrightarrow{e_2} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Once again, these vectors are already orthogonal, so normalizing gives:

$$\overrightarrow{u_1} = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}, \qquad \overrightarrow{u_2} = \begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

Now place these vectors in the V matrix, and take the transpose:

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

Finally, construct the Σ matrix by placing the square roots of the eigenvalues of MM^T (and M^TM) in descending order along the diagonal.

$$\Sigma = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}$$

This completes the singular value decomposition, and the following equality holds.

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

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

IV. Application

A. Theory

The goal of image compression is to decrease the total amount of bytes associated with an image to save disk space or perhaps transmission duration. Image compression by SVD behaves slightly differently to other image compression algorithms such as JPEG or PNG who rely more on the Fourier transform of images.

Given a black and white image, I, we can create a matrix M where the entries of M are the gray scale values of corresponding pixels of the image I. Each pixel (assuming 256 grayscale values) is saved in the computer as a single byte. We can apply SVD on the matrix to understand how this image compression operates. Examining the matrices in Eqn. (2) by columns we can see that:

$$U = \begin{bmatrix} \vdots & \vdots & & \vdots \\ u_1 & u_2 & \cdots & u_r \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_r \end{bmatrix}$$

$$V = \begin{bmatrix} \vdots & \vdots & & \vdots \\ v_1 & v_2 & \cdots & v_r \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

Breaking down the matrix multiplication of the SVD: [ii]

$$M = \underbrace{u_1\sigma_1 v_1^T + u_2\sigma_2 v_2^T + \cdots + u_r\sigma_r v_r^T}_{\text{‘r’ terms}} \quad (4)$$

Note that each $u_i\sigma_i v_i^T$ unit has the same dimensions of the original matrix M, and r is the number of columns.

To perfectly recreate the original matrix M, we would need to use all terms in the decomposition. However, if we include only select terms, we can recreate a similar image. As you may imagine, the larger the value σ_i is, the more significant that element is to the image. In SVD, the singular values σ are ordered from greatest to smallest magnitude along the diagonal. Therefore, to approximate the image, we can include the first k terms where $0 < k < r$ in this decomposition shown in Eqn. (4).

B. Data Compression

To understand how much space we can save by image compression through SVD, imagine each matrix entry as a byte (or packet) of data that we must save. Therefore if we have a 512 x 512 pixel image that corresponds to a 512 x 512 matrix of grayscale values, in order to save this image, we must use a total of 512 x 512 = 262,144 bytes. However, we can take advantage of the decomposition that SVD provides and instead of saving each byte of the 512 x 512 square array, we can save the first k terms of the breakdown in Eqn. (4). Each element consists of a $u_i, \sigma_i,$ and v_i^T. Recall that $u_i$is a n x 1 vector, v_i is a m x 1 vector, and σ_i is a singular value, which is a scalar. Therefore, for each term from the decomposition, there are only $n + m + 1$ bytes that we need to save.

In our 512 x 512 picture, if we were to save the first 100 terms of equation (4), we would need to save $100 * (512 + 512 + 1) = 102{,}500$ bytes. In other words, we only need save the bytes from the first 100 columns of U, the first 100 singular values of Σ and the first 100 columns of V. In the end this means that instead of saving 262,144 bytes, we only need to save 102,500, which is 39.1% of the original method. As we can see from the following section, 100 terms from the SVD creates an image that is almost indistinguishable to the human eye from the original image.

C. Results

Using the image processing standard picture of Lena (Figure 1), we can demonstrate the effects of this type of compression. The image used is a 512 x 512 image and therefore the matrices *M, U,Σ,* and V are all 512 x 512 matrices.

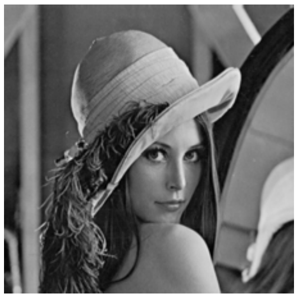

Figure 1: Original Image

As we increase the number of singular values used, we can see the form of the image start to develop.

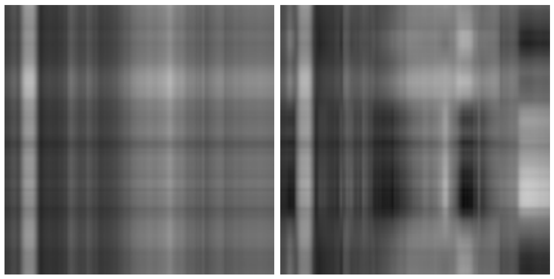

Figure 2: Image Compression by SVD using one singular value, (left), and two singular values (right),

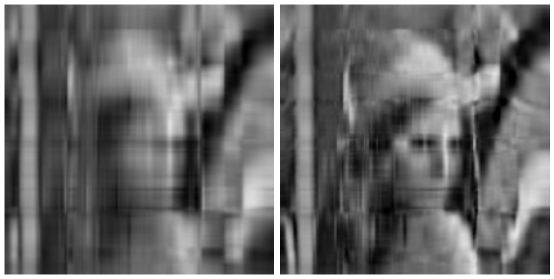

Figure 3: Image Compression by SVD using five singular values (left), and ten singular values (right).

As we can see in Figure 2 and 3, as we increase the number of singular values used, the details of the image start to take form. After using the most 10 significant singular values, we can see the general form of the image. This image takes 96% less storage space.

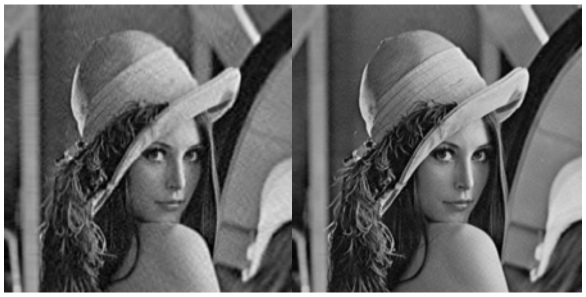

Figure 4: First 60 (left), and 100 (right) entries in decomposition used.

By the time that we get to the first 100 of the 512 terms of the decomposition (Figure 4), the image is almost indistinguishable from the original image, yet the compression algorithm saves about 60.9% storage space between the original image and the compressed image.

These images were produced by the MATLAB code located in the appendix. We implemented the algorithm described in this paper where we utilize the SVD to compress an image. By changing the *k* value, we can alter how many terms of Eqn. (4) to use in the construction of the image.

V. Conclusion

Based on our results, using SVD as an image compression technique, we can save at least 60.9% of memory storage while maintaining the visual integrity of the image.

SVD allows for the 'factorization' of a matrix into a diagonal form and can be used to breakdown a transformation into isolated rotation and scalar modifications. It is a powerful method for breaking apart a complex system into accessible pieces to better understand the matrix for more efficient usage. This form makes it a useful analytical tool and has widespread applications beyond image compression including solving homogenous linear equations, 3D animations and eigenfaces.

Acknowledgment

We would like to acknowledge Yifei Feng and Maia Bittner for constructive criticism of our project. Also, we acknowledge Dr. Sarah Spence Adams for her guidance on this paper.

References

[1] Lay, D. (1997). *Linear Algebra and Its Applications*. New York: Addison-Wesley.
[2] Baker, K. *Singular Value Decomposition Tutorial*. http://web.ics.purdue.edu/~park283/wp-content/uploads/2010/10/Singular_Value_Decomposition_Tutorial.pdf
[3] Abrahamsen, Adam and Richards David: "Image Compression using Singular Value Decomposition." December 14, 2001
[4] Weisstein, Eric W. "Singular Value Decomposition." From Mathworld—A Wolfram Web Resources. http://mathworld.wolfram.com/SingularValueDecomposition.html

Appendix

```
function imageCompressJPG2(imagePath, k)
% Precondition:
%   imagePath: the name of the jpg image
%  k is the number of singular values to use
%  in the image construction
% Postcondition:
%   displays the original image and image
%   generated by SVD compression

JPG=imread(imagePath);
grayscale = (JPG(:,:,1)*.3+JPG(:,:,2) ...
     *.59+JPG(:,:,3)*.11);
figure(1)
imshow(grayscale)  % original image
B=im2double(grayscale,'indexed');
[u,s,v]=svd(B);
C=zeros(size(B));
for j=1:k
    C=C+s(j,j)*u(:,j)*v(:,j).';
end
C=uint8(round(C-1));
figure(2)
imshow(C); % Resulting image
end
```

Practice Problems

1. Find SVD of:

$$A = \begin{bmatrix} 3 & 2 \\ 2 & 3 \\ 2 & -2 \end{bmatrix}$$

a) Find eigenvalues and vectors of A^TA and AA^T
b) Construct V^T and show the columns are orthonormal
c) Construct Σ
d) Construct U and show the columns are orthonormal
e) Check $A = U\Sigma V^T$

2. Exploration of image compression. This problem requires the use of and some experience with MATLAB.

a) Download the code in the appendix and save your favorite picture in jpg format in the same directory.
b) Try running the program using the image name (in single quotes) and a *k* value as input arguments. Play around with several different *k* values and observe the resulting image.
c) At what *k* value does the resulting image indistinguishable to the original image.
d) Hypothetically, how much space would you save if you used this *k* value.
e) Observe the matrix 's' that contains all the singular values (you can do this by setting a breakpoint in the middle of the program and double clicking the matrix in the workspace).
f) What is the largest singular value in the matrix? What is the smallest? What is the value of the singular value where you deemed the image "good enough." How does this value compare to the other singular values.
g) For further exploration, try the algorithm on different pictures. Try it on a single vertical white line. Then try it on a set of horizontal and vertical lines. How many singular values are need to recreate those images? What insight does this shine on how the number of terms needed to approximate the image.

[i] The Invertible Matrix Theorem states that if a matrix, B, is invertible, the equation $B\boldsymbol{x} = \boldsymbol{0}$ has only the trivial solution. By definition, to have eigenvalue and eigenvector pairs, you must have a non-trivial solution to $(A - I\lambda)\boldsymbol{x} = \boldsymbol{0}$. Therefore, solving for λ such that there is a non-trivial solution is the same as solving for where the matrix, B, is non-invertible, and a matrix is non-invertible if its determinant is equal to 0.

[ii] See D. Carlson , D. Lay, and A. D. Porter, “Modern Views of Matrix Multiplication” Linear Algebra Gems

SVD Practice Problem Soln

$$A = \begin{bmatrix} 3 & 2 \\ 2 & 3 \\ 2 & -2 \end{bmatrix} \quad A^T = \begin{bmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{bmatrix}$$

① Find eigenvalues & eigenvectors of A^TA & AA^T

$$A^TA = \begin{bmatrix} 17 & 8 \\ 8 & 17 \end{bmatrix}$$

Find eigenvalues (A^TA)

$$(A^TA - \lambda I)\bar{x} = \bar{0}$$

determinant $(17-\lambda)(17-\lambda)-64=0$

$$\lambda^2 - 34\lambda + 289 - 64 = 0$$

$$\lambda_1 = 9 \quad \lambda_2 = 25$$

Find eigenvectors (A^TA)

$$\lambda_1 : 9 \Rightarrow \begin{bmatrix} 17-9 & 8 \\ 8 & 17-9 \end{bmatrix} \Rightarrow \begin{bmatrix} 8 & 8 \\ 8 & 8 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

$$8x_1 + 8x_2 = 0$$

$$x_1 = -x_2$$

$$V_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$\lambda_2 : 25 \Rightarrow \begin{bmatrix} 17-25 & 8 \\ 8 & 17-25 \end{bmatrix} \Rightarrow \begin{bmatrix} -8 & 8 \\ 8 & -8 \end{bmatrix} \bar{x} = 0$$

$$-8x_1 + 8x_2 = 0$$

$$x_1 = x_2$$

$$V_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

② Find V

$$V = [V_2 \ V_1] = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$$

orthonomalize $V = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$

$\sqrt{\left(\frac{1}{\sqrt{2}}\right)^2 + \left(\frac{1}{\sqrt{2}}\right)^2} = 1$ ✓

$V_1 \cdot V_2 = \emptyset$ ✓

③ Find Σ $\quad \sigma_{1,2} = \sqrt{\lambda_{1,2}} = \sqrt{25}, \sqrt{9} = 5, 3$

$$\Sigma = \begin{bmatrix} 5 & 0 \\ 0 & 3 \\ 0 & 0 \end{bmatrix} \longrightarrow$$

σ's on diagonal
must have same dimensions as A

④ Find U

Find eigenvalues (AA^T)

$$AA^T = \begin{bmatrix} 13 & 12 & 2 \\ 12 & 13 & -2 \\ 2 & -2 & 8 \end{bmatrix}$$

$$(AA^T - \lambda I)x = 0$$

$$\det \begin{bmatrix} 13-\lambda & 12 & 2 \\ 12 & 13-\lambda & -2 \\ 2 & -2 & 8-\lambda \end{bmatrix} = -\lambda(\lambda^2 - 34\lambda + 225)$$

$$\lambda_1 = 0 \quad \lambda_2 = 9 \quad \lambda_3 = 25$$

Find eigenvectors (AA^T)

$\lambda_1 = 0$ $\begin{bmatrix} 13 & 12 & 2 \\ 12 & 13 & -2 \\ 2 & -2 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \bar{0}$ Solve

$$-x_1 = x_2 = -2x_3$$

$$u_1 = \begin{bmatrix} -1 \\ 1 \\ 1/2 \end{bmatrix}$$

$\lambda_2 = 9$ $\begin{bmatrix} 4 & 12 & 2 \\ 12 & 4 & -2 \\ 2 & -2 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \bar{0}$

$$4x_1 = -4x_2 = x_3$$

$$u_2 = \begin{bmatrix} 1 \\ -1 \\ 4 \end{bmatrix}$$

$\lambda_3 = 25$ $\begin{bmatrix} -12 & 12 & 2 \\ 12 & -12 & -2 \\ 2 & -2 & -17 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \bar{0}$

$$x_1 = x_2$$

$$x_3 = 0$$

$$u_3 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

$$\mathcal{U} = [\, u_3 \;\; u_2 \;\; u_1 \,]$$

$$= \begin{bmatrix} 1 & 1 & -1 \\ 1 & -1 & 1 \\ 0 & 4 & 1/2 \end{bmatrix} \xrightarrow{\text{Normalize}} \begin{bmatrix} 1/\sqrt{2} & 1/3\sqrt{2} & -2/3 \\ 1/\sqrt{2} & 1/3\sqrt{2} & 2/3 \\ 0 & 4/3\sqrt{2} & 1/3 \end{bmatrix}$$

5

$$\text{SVD}(A) = U \Sigma V^T$$

$$\underset{A}{\begin{bmatrix} 3 & 2 \\ 2 & 3 \\ 2 & -2 \end{bmatrix}} = \underset{U}{\begin{bmatrix} 1/\sqrt{2} & 1/3\sqrt{2} & -2/3 \\ 1/\sqrt{2} & -1/3\sqrt{2} & 2/3 \\ 0 & 4/3\sqrt{2} & 1/3 \end{bmatrix}} \underset{\Sigma}{\begin{bmatrix} 5 & 0 \\ 0 & 3 \\ 0 & 0 \end{bmatrix}} \underset{V^T}{\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}}$$