

# Applications of Linear Algebra

## Image Compression using Singular Value Decomposition

David Richards and Adam Abrahamsen



1/41



# Introduction

- The Singular Value Decomposition is a very important process. In this presentation we will demonstrate how to use Singular Value Decomposition (SVD) to factorize and approximate large matrices, specifically images.





# The method of Singular Value Decomposition (SVD)

- We begin by selecting a matrix ( $A$ ) that can be any  $m \times n$  matrix. The value of this is sort of vague but extremely important, matrix ( $A$ ) can be any matrix without any provisos or delimiters like most other linear algebra techniques. In other words there are no “provided that’s” for (SVD).
- The first step of (SVD) is factoring the matrix ( $A$ ) into three parts  $USV^T$ . We need to find two orthonormal bases that diagonalize ( $A$ ).
- $U$  and  $V$  are orthogonal matrices, and  $S$  is a diagonal matrix.
- The matrix  $U$  contains one orthonormal basis.  $U$  is also known as the left singular vectors.







## Factoring $V$ and $S$

- First we will find  $V$ . To eliminate  $U$  from the equation  $A = USV^T$  you simply multiply on the left by  $A^T$ :

$$A^T A = (USV^T)^T (USV^T) = VS^T U^T USV^T.$$

Since  $U$  is an orthogonal matrix,  $U^T U = I$  which gives

$$A^T A = VS^2 V^T$$

Notice that this is similar to the diagonalization of a matrix  $A$ , where  $A = Q\Lambda Q^T$ . But now the symmetric matrix is not  $A$  it is  $A^T A$ .

- To find  $V$  and  $S$  we need to diagonalize  $A^T A$  by finding the eigenvalues and eigenvectors. The eigenvalues are the square of the elements of  $S$  (the singular values) and the eigenvectors are the columns of  $V$  (the right singular vectors).





## A $2 \times 2$ Example

- Let :

$$A = \begin{pmatrix} 2 & -2 \\ 1 & 1 \end{pmatrix}$$

$$\begin{aligned} A^T A &= \begin{pmatrix} 2 & 1 \\ -2 & 1 \end{pmatrix} \begin{pmatrix} 2 & -2 \\ 1 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 5 & -3 \\ -3 & 5 \end{pmatrix} \end{aligned}$$

- The eigenvalues for this matrix are  $\lambda = 8, 2$ .





- We construct the matrix  $S^2$  by placing the eigenvalues along the main diagonal in decreasing order.

$$S^2 = \begin{pmatrix} 8 & 0 \\ 0 & 2 \end{pmatrix}$$

Therefore, taking the square root of matrix  $S^2$  gives,

$$S = \begin{pmatrix} 2\sqrt{2} & 0 \\ 0 & \sqrt{2} \end{pmatrix}$$

- Now we will find the matrix  $V$ .
- The eigenvector for  $\lambda = 8$  is

$$v_1 = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

and for  $\lambda = 2$  the eigenvector is

$$v_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$



- But we need these to be of length one. So dividing by the magnitude we get:

$$v_1 = \begin{pmatrix} \frac{-\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{pmatrix}$$

and

$$v_2 = \begin{pmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{pmatrix}$$

- Constructing the augmented matrix

$$V = (v_1 \ v_2) = \begin{pmatrix} \frac{-\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$$





## Factoring $U$

- Eliminating  $V$  from the equation is very similar to eliminating  $U$ . Instead of multiplying on the left by  $A^T$  we will multiply on the right by  $A^T$ . This gives:

$$AA^T = (USV^T)(USV^T)^T = USV^T V S^T U^T.$$

- Since  $V^T V = I$ , this gives

$$AA^T = US^2 U^T$$

- Again we will find the eigenvectors, but this time for  $AA^T$ . These are the columns of  $U$  (the left singular vectors).





- Find the eigenvectors for  $AA^T$  using the eigenvalues of  $\lambda = 8, 2$ , that we found while finding  $V$ . This provides us with the augmented matrix  $U$ .

$$U = (u_1 \ u_2) = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$

- Therefore

$$A = USV^T = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 2\sqrt{2} & 0 \\ 0 & \sqrt{2} \end{pmatrix} \begin{pmatrix} \frac{-\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}$$





# Image Compression with (SVD)

- What is the purpose of transforming the matrix ( $A$ ) into  $USV^T$ ?
- We want to approximate the  $m \times n$  matrix ( $A$ ) by using far fewer entries than in the original matrix.
- By using the rank of a matrix we remove the redundant information (the dependant entries) when  $r < m$ , or  $r < n$ .

$$A = \sigma_1 u_1 v_1^T + r \sigma_2 u_2 v_2^T + \cdots + \sigma_r u_r v_r^T + 0 u_{r+1} v_{r+1}^T + \cdots$$

- Since the singular values are always greater than zero. Adding on the dependant terms where the singular values are equal to zero does not effect the image. The terms at the end of the equation zero out leaving us with:

$$A = \sigma_1 u_1 v_1^T + r \sigma_2 u_2 v_2^T + \cdots + \sigma_r u_r v_r^T$$





- We can further approximate the matrix by leaving off more singular terms of the matrix ( $A$ ). Since the singular values are arranged in decreasing order, the last terms will have the least affect on the overall image.
- Doing this reduces the amount of space required to store the image on a computer.





# Using Matlab to perform (SVD)

- Matlab provides us with the ability to perform (SVD) on larger matrices. For an example we will use a  $7 \times 7$  matrix with rank 5.
- Using the command

```
A=randint(7,7,25,5)+25
```

provides us with a  $7 \times 7$  matrix ( $A$ ) with values between 0 and 50, and with a rank of 5.





$$A = \begin{pmatrix} 24 & 36 & 31 & 31 & 39 & 44 & 39 \\ 27 & 39 & 35 & 33 & 33 & 35 & 33 \\ 29 & 37 & 35 & 43 & 35 & 33 & 35 \\ 32 & 29 & 43 & 36 & 27 & 35 & 27 \\ 39 & 35 & 30 & 31 & 25 & 15 & 25 \\ 29 & 44 & 26 & 47 & 40 & 26 & 40 \\ 34 & 21 & 37 & 22 & 27 & 39 & 27 \end{pmatrix}$$

- Now we can use the power of matlab again to perform (SVD) on the matrix ( $A$ ). Using the commands:

$[U,S,V]=\text{svd}(A)$  factors  $A$  into  $USV^T$ .



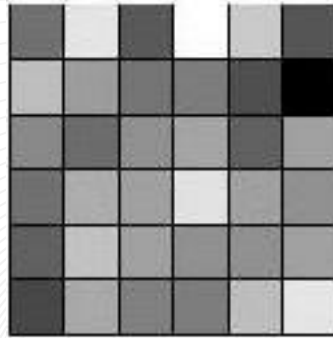


- After factoring ( $A$ ), using the command

```
svdimage(A,U,S,V,1,gray)
```

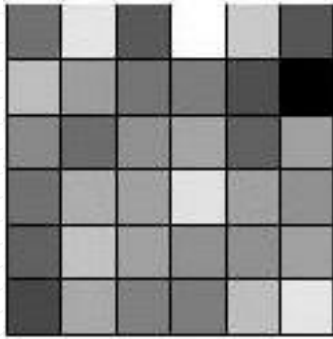
gives us the opportunity to view the matrix  $A$  as an image, and see each individual iteration of ( $A$ ) using the rank approximation. A little explanation of the entries of this function require some explanation. The  $A$ ,  $U$ ,  $S$ , and  $V$  are self explanatory, but the 1 and the gray haven't been explained yet. The "1" starts the program off with the first iteration and the "gray" uses the colormap that matlab defines as gray. Now each number of the matrix corresponds to the color that matlab has defined as the colormap(gray).



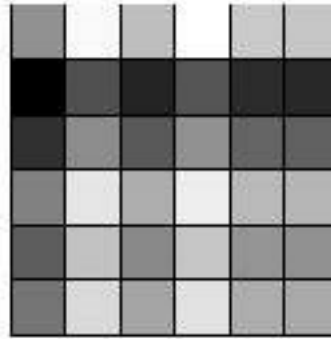


Original Matrix



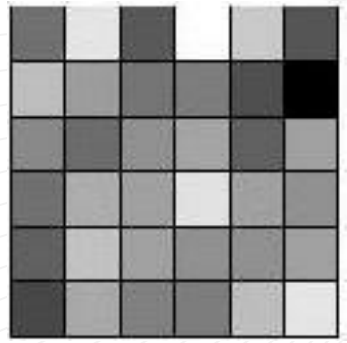


Original Matrix

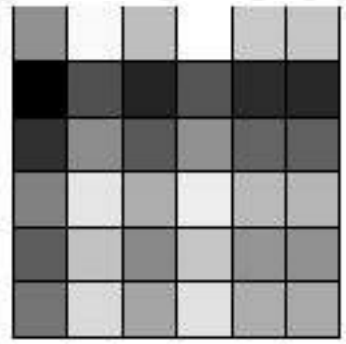


1 Iteration

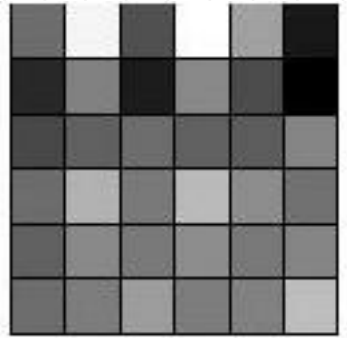




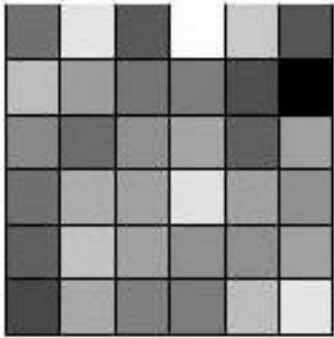
Original Matrix



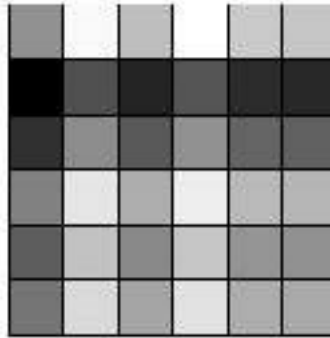
1 Iteration



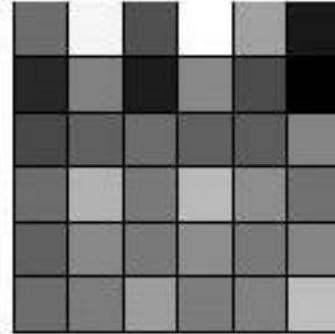
2 Iterations



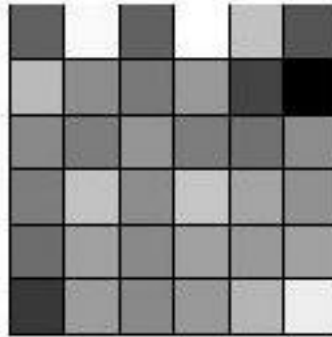
Original Matrix



1 Iteration

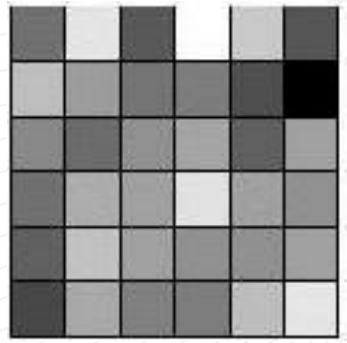


2 Iterations

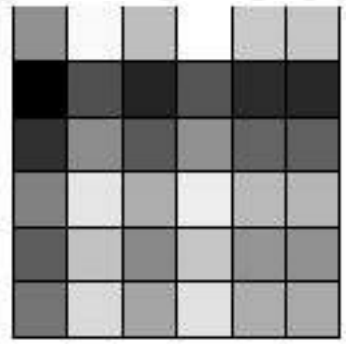


3 Iterations

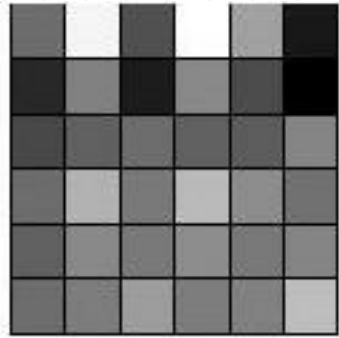




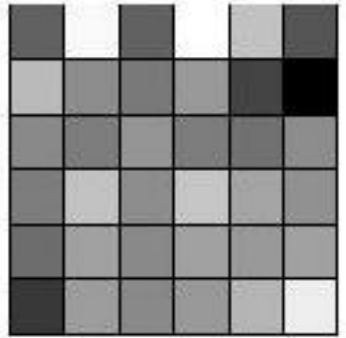
Original Matrix



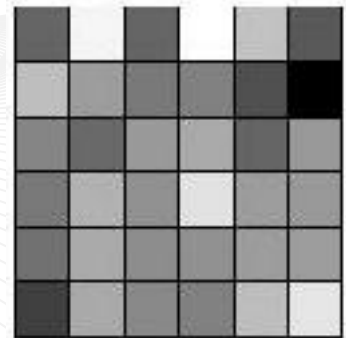
1 Iteration



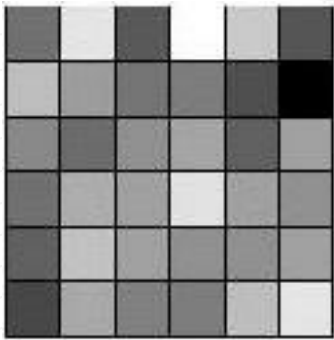
2 Iterations



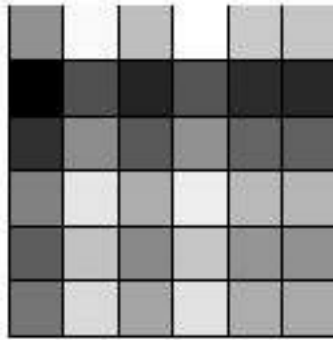
3 Iterations



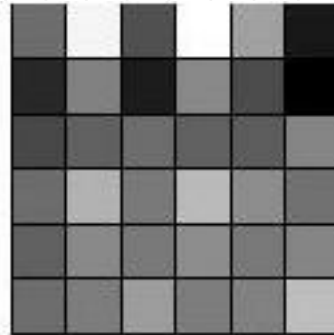
4 Iterations



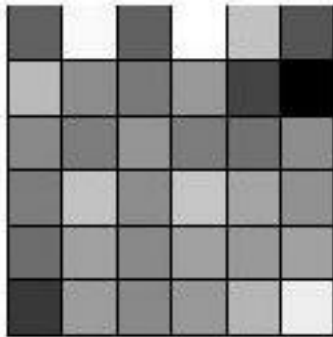
Original Matrix



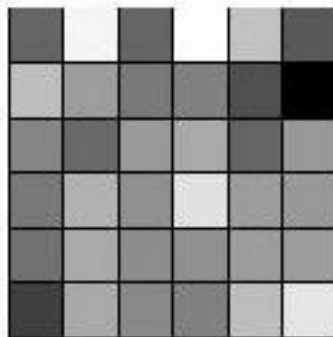
1 Iteration



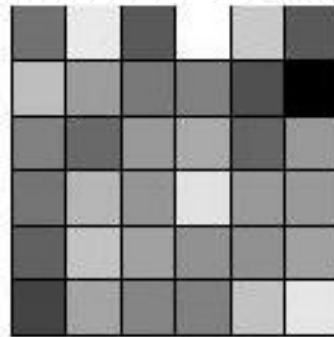
2 Iterations



3 Iterations



4 Iterations



5 Iterations





- As you can see here, five iterations gives us the exact image back.







- As you can see here, five iterations gives us the exact image back.
- So as we mentioned earlier the dependent singular values are zero after we exceed the rank of the matrix. That is why after five iterations (the rank of this matrix) we get an exact approximation.
- For the pictures in this presentation we used a program that we wrote with the help of David Arnold. We had to write this program in order to fit these images into the document. Using the commands,





```
close all
[A,map]=imread('lena.gif');
B=im2double(A,'indexed');
imshow(B,map)
[u,s,v]=svd(B);
C=zeros(size(B));
for j=1:k
    C=C+s(j,j)*u(:,j)*v(:,j).';
end
C=floor(C);
imshow(C,map)
k=find(C<1);
C(k)=1;
set(gcf,'Unit','inches','Paperposition',[0,0,2,1])
print -djpeg 'lenak.jpg'
```





- By changing the k values in the forloop we were able to construct the different iterations. We also used this for the previous block images but with a few changes. You may notice the similarities between these two equations:

$$C=C+s(j,j)*u(:,j)*v(:,j).'$$

and

$$A = \sigma_1 u_1 v_1^T + r \sigma_2 u_2 v_2^T + \dots + \sigma_r u_r v_r^T$$

- Now we will show the use of (SVD) on Lena which is one of the standardized images.





Original Image





Original Image



5 Iterations





Original Image



5 Iterations



10 Iterations





Original Image



5 Iterations



10 Iterations



20 Iterations





Original Image



5 Iterations



10 Iterations



20 Iterations



60 Iterations





Original Image



5 Iterations



10 Iterations



20 Iterations



60 Iterations



100 Iterations





- The first five iterations actually give the shape of the image, which is a decent approximation. This takes up 99.9% less storage space than the original image.





Original Image



5 Iterations



10 Iterations



20 Iterations



60 Iterations



100 Iterations





- The first five iterations actually give the shape of the image, which is a decent approximation. This takes up 99.9% less storage space than the original image.
- Using the first sixty iterations we get a good approximation, we can identify the person with a substantial degree of detail. This image requires 84% less storage space than the original image. This is good.





Original Image



5 Iterations



10 Iterations



20 Iterations



60 Iterations



100 Iterations





- The first five iterations actually give the shape of the image, which is a decent approximation. This takes up 99.9% less storage space than the original image.
- Using the first sixty iterations we get a good approximation, we can identify the person with a substantial degree of detail. This image requires 84% less storage space than the original image. This is good.
- Finally, the first one-hundred iterations give a near perfect image, and yet requires 55% less storage space. This is very good!





Original Image



5 Iterations



10 Iterations



20 Iterations



60 Iterations



100 Iterations





- The first five iterations actually give the shape of the image, which is a decent approximation. This takes up 99.9% less storage space than the original image.
- Using the first sixty iterations we get a good approximation, we can identify the person with a substantial degree of detail. This image requires 84% less storage space than the original image. This is good.
- Finally, the first one-hundred iterations give a near perfect image, and yet requires 55% less storage space. This is very good!
- Using (SVD) for image compression can be a very useful tool to save storage space. We were able to get an image that is indistinguishable from the original image, but only using 45% of the original storage space.





# Conclusion

As you have seen Singular Value Decomposition is an extremely powerful tool. Which is not only used for image compression, but has many other applications.



# References

- [1] Arnold, David. His matlab and  $\text{\LaTeX}$  expertise.
- [2] Kalman, Dan. **“A Singularly Valuable Decomposition.”** **The College Mathematics Journal. Vol.27\_No.1\_Jan 1998, 2-23.**
- [3] Lay, David C. **Linear Algebra and Its Applications.** Addison Wesley Longman, Inc., 1997
- [4] Strang, Gilbert. **Introduction to Linear Algebra.** Wellesley-Cambridge Press, 1998.
- [5] The Sauceman. His  $\text{\LaTeX}$  expertise.

