EEE-5114 Engineering Analysis

Midterm Exam 3: Numerical Linear Algebra

Tom DesRosiers 12/9/2013

File Compression using Singular Value Decomposition

The field of numerical linear algebra primarily involves solving problems involving matrices. An important result of numerical linear algebra is a matrix factorization method called singular value decomposition or SVP. This discussion will cover SVP generation along with important 'real world' applications.

Singular value decomposition is an integral part of many computer vision and image processing applications. SVD provides tools to create algorithms used in: motion planning for robotics, video tracking, facial and optical character recognition. SVD is also used in many image processing algorithms dealing with topics such as: photo enhancement, satellite imagery, image compression and medical imagery.

The purpose of SVD is to break down a given matrix into three simpler matrices, two orthogonal and one diagonal. The three resulting matrices have useful properties.

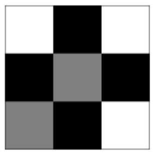
Definition: Any $m\ x\ n$ matrix can be factored as $A = U\Sigma V^T$ where U is an $m\ x\ n$ orthogonal matrix, V is an $n\ x\ n$ orthogonal matrix, and Σ is a $m\ x\ n$ matrix with entries: $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$ down the main diagonal and zeros elsewhere.

$$A_{m\,x\,n} \quad = \quad U_{m\,x\,m} \quad \Sigma_{m\,x\,n} \quad {V^T}_{n\,x\,n}$$

$$A_{m\,x\,n} = \begin{pmatrix} u_1 & u_2 & \dots & u_m \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \end{pmatrix} \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_n \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_n^T \end{bmatrix}$$

Fig. 1

The $U\Sigma V^T$ form of **A** and its properties make all of the applications mentioned earlier possible. This discussion focuses on the properties of $U\Sigma V^T$ used for minimizing data transfer and storage. Computers store images in a matrix where each value represents pixel brightness. Figure two below shows a 3 x 3 matrix with a value of 0 for white, 0.5 for gray and 1 for black [1]. Color images are stored as three separate matrixes containing pixel brightness values for red; blue and green (see Fig. 3). Storage requirements for an m x n matrix is m*n [1]. Memory requirements increase exponentially with the matrix size. The purpose of SVD when applied to image processing is to reduce the storage memory requirements.



$$= \begin{bmatrix} 1 & 0 & 1 \\ 0 & .5 & 0 \\ .5 & 0 & 1 \end{bmatrix}$$

Fig. 2.Cooper, Ian & Lorenc, Craig. Image. SVD_Slideshow.pdf 13. Dec 2006. 8 Dec. 2013 <http://online.redwoods.edu/instruct/darnold/laproj/fall2006/iancraig/SVD_Slideshow.pdf>

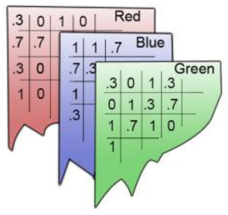


Fig. 3.Cooper, Ian & Lorenc, Craig. Image. SVD_Slideshow.pdf 13. Dec 2006. 8 Dec. 2013 <http://online.redwoods.edu/instruct/darnold/laproj/fall2006/iancraig/

SVD allows for a matrix to be rewritten as a sum of rank one matrices. This can be written as:

$$A = \overrightarrow{\boldsymbol{u_1}}\sigma_1\overrightarrow{\boldsymbol{v_1^T}} + \cdots + \overrightarrow{\boldsymbol{u_n}}\sigma_n\overrightarrow{\boldsymbol{v_n^T}} \quad or \quad A = \sum_{i=1}^{n}\sigma_i\overrightarrow{\boldsymbol{u_i}\ \boldsymbol{v_i^T}}$$

Since σ_i are ordered from biggest to smallest a sufficient approximation to the oringal matrix A can generally be derived by dropping some of the terms at the end of the $\sum_{i=1}^{n}\sigma_i\overrightarrow{\boldsymbol{u_i}\ \boldsymbol{v_i^T}}$ expansion [2]. Let k be the number of terms needed to create a sufficient representation of **A**. The matrix $\mathbf{A}_k$ can now approximate **A** by the partial sum: $A \approx A_k = \sum_{i=1}^{k}\sigma_i\overrightarrow{\boldsymbol{u_i}\ \boldsymbol{v_i^T}}$. The SVD partial sum requires the storage of just the values $(\overrightarrow{u_1,u_2,}\ldots..\overrightarrow{u_k}),(\overrightarrow{v_1,v_2,}\ldots..\overrightarrow{v_k})\ and\ (\overrightarrow{\sigma_1\sigma_2}\ldots..\overrightarrow{\sigma_k})$ to represent A[2].

The amount of memory required by an SVD approximation of value k for an m x n matrix A is described by the relationship $A_{memory} = k\,(m+n+1)$. The linear SVD relationship is much more desirable versus the exponential relationship of the orginal matrix [2]. When bandwitch and /or memory limitiations are an issue it's easy to see why singular value decomposition is used.

The following example illustrates how $A = U\Sigma V^T$ is generated. The SVD algorithm is applied to the matrix $A = \begin{bmatrix} 2 & 2 \\ -1 & 1 \end{bmatrix}$.

(1) Calculate AA^T *and* A^TA.

$$AA^T = \begin{bmatrix} 2 & 2 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} 2 & -1 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 8 & 0 \\ 2 & 0 \end{bmatrix} \qquad A^TA = \begin{bmatrix} 2 & -1 \\ 2 & 1 \end{bmatrix}\begin{bmatrix} 2 & 2 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix}$$

(2) Find eigenvalues for AA^T.

$$|AA^T - \lambda I| = 0$$

$$\left|\begin{bmatrix} 8 & 0 \\ 0 & 2 \end{bmatrix} - \lambda\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right| = 0$$

$$\begin{vmatrix} 8-\lambda & 0 \\ 0 & 2-\lambda \end{vmatrix} = 0$$

$$(8-\lambda)(2-\lambda) = 0$$

$$\lambda_1 = 8 \; and \; \lambda_2 = 2$$

(3) Find values of matrix Σ.

The singular values of AA^T represented by σ_i are related to the eigenvalues by the relationship: $\sigma_i = \sqrt{\lambda_i}$. The singular values for AA^T are:

$$\sigma_1 = \sqrt{\lambda_1} = \sqrt{8} \quad \text{and} \quad \sigma_2 = \sqrt{\lambda_2} = \sqrt{2} \quad \text{therefore: } \Sigma = \begin{bmatrix} \sqrt{8} & 0 \\ 0 & \sqrt{2} \end{bmatrix}$$

(4) The columns of U are the eigenvectors of AA^T.

$$(AA^T - \lambda I)\vec{\boldsymbol{x}} = 0 \qquad \text{For} \quad \lambda_1 = 8 \qquad \text{For} \quad \lambda_2 = 2$$

$$\left(\begin{bmatrix} 8 & 0 \\ 0 & 2 \end{bmatrix} - \lambda_1\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right)\vec{\boldsymbol{x}}_1 = 0 \qquad \begin{bmatrix} 8-8 & 0 \\ 0 & 2-8 \end{bmatrix}\vec{\boldsymbol{x}}_1 = 0 \qquad \begin{bmatrix} 8-2 & 0 \\ 0 & 2-2 \end{bmatrix}\vec{\boldsymbol{x}}_1 = 0$$

$$\begin{bmatrix} 8-\lambda_1 & 0 \\ 0 & 2-\lambda_1 \end{bmatrix}\vec{\boldsymbol{x}}_1 = 0 \quad \nearrow \quad \begin{bmatrix} 0 & 0 \\ 0 & -6 \end{bmatrix}\vec{\boldsymbol{x}}_1 = 0 \qquad \begin{bmatrix} 6 & 0 \\ 0 & 0 \end{bmatrix}\vec{\boldsymbol{x}}_2 = 0$$

$\overrightarrow{\boldsymbol{x_1}} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$ $\qquad$ $\overrightarrow{\boldsymbol{x_2}} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

$\boldsymbol{U} = [\overrightarrow{\boldsymbol{x_1}} \quad \overrightarrow{\boldsymbol{x_2}}] = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$

(5) The columns of V are the eigenvectors of A^TA.

$(A^TA - \lambda I)\vec{\boldsymbol{x}} = 0$ $\qquad$ For $\quad \lambda_1 = 8$ $\qquad$ For $\quad \lambda_2 = 2$

$\left(\begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix} - \lambda_1 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right)\vec{\boldsymbol{x}}_1 = 0$ $\qquad$ $\begin{bmatrix} 5-8 & 3 \\ 3 & 5-8 \end{bmatrix}\vec{\boldsymbol{x}}_1 = 0$ $\qquad$ $\begin{bmatrix} 5-2 & 3 \\ 3 & 5-2 \end{bmatrix}\vec{\boldsymbol{x}}_1 = 0$

$\begin{bmatrix} 5-\lambda_1 & 3 \\ 3 & 5-\lambda_1 \end{bmatrix}\vec{\boldsymbol{x}}_1 = 0$ $\quad \nearrow \quad$ $\begin{bmatrix} -3 & 3 \\ 3 & -3 \end{bmatrix}\vec{\boldsymbol{x}}_1 = 0$ $\qquad$ $\begin{bmatrix} 3 & 3 \\ 3 & 3 \end{bmatrix}\vec{\boldsymbol{x}}_2 = 0$

$\overrightarrow{\boldsymbol{x_1}} = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$ $\qquad$ $\overrightarrow{\boldsymbol{x_2}} = \begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$

$\boldsymbol{V} = [\overrightarrow{\boldsymbol{x_1}} \quad \overrightarrow{\boldsymbol{x_2}}] = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$

We now have all of the values to complete the SVD factorication for $A = \begin{bmatrix} 2 & 2 \\ -1 & 1 \end{bmatrix}$. The final SVD factorization is:

$$A \quad = \quad U_{m\,x\,m} \quad \Sigma_{m\,x\,n} \quad {V^T}_{n\,x\,n}$$

$$A \quad = \quad \begin{pmatrix} u_1 & u_2 & \dots & u_m \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \end{pmatrix} \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_n \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_n^T \end{bmatrix}$$

$$A \quad = \quad \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{8} & 0 \\ 0 & \sqrt{2} \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$

A can now be represented as a sum of rank one matrices:

$$A = \sum_{i=1}^{n} \sigma_i \overrightarrow{\boldsymbol{u_i}} \overrightarrow{\boldsymbol{v_i}}^T = \sum_{i=1}^{2} \sigma_i \overrightarrow{\boldsymbol{u_i}} \overrightarrow{\boldsymbol{v_i}}^T = \sqrt{8}\begin{bmatrix} -1 \\ 0 \end{bmatrix}[1/\sqrt{2} \quad 1/\sqrt{2}] + \sqrt{2}\begin{bmatrix} 0 \\ 1 \end{bmatrix}[-1/\sqrt{2} \quad 1/\sqrt{2}]$$

MATLAB makes it easy to generate the SVD of a matrix. The following code generates the SVD for the example matrix above:

MATLAB CODE:

```
A = [2 2;-1 1]    %Input A
[U,S,V] = svd(A)  %Generate SVD decomposition for A

B = U * S * V'    %Optional: Show that SVD Equals
                  %the original Matrix A
```

OUTPUT:

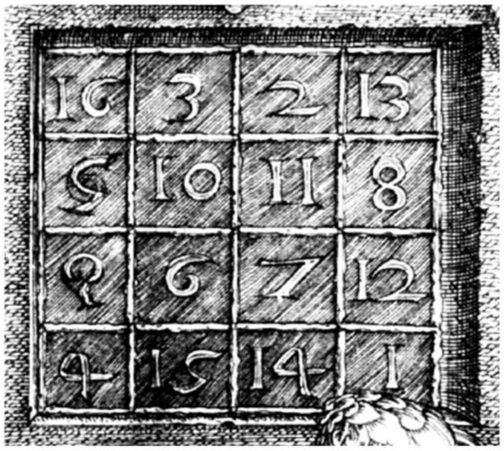
```
A =

   2    2

  -1    1

U =

  -1.0000    0.0000

   0.0000    1.0000

S =

   2.8284         0

        0    1.4142

V =

  -0.7071   -0.7071

  -0.7071    0.7071

B =

   2.0000    2.0000

  -1.0000    1.0000
```

The example above is a good example of how the SVD of a matrix is calculated, however, because of it's small size it is not very useful for showing how SVD can be used in image compression.

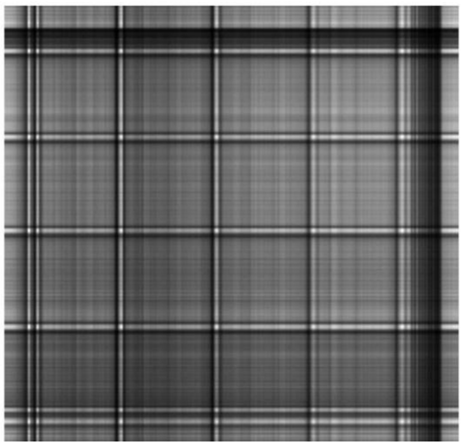
The following MATLAB example calculates the SVD of an image file and then displays the images created from the terms of the summation: $A = \sum_{i=1}^{n} \sigma_i \overrightarrow{\boldsymbol{u_i}} \overrightarrow{\boldsymbol{v}_i^T}$. A listing of the code can be found at the end of this report [3].

ORIGINAL IMAGE

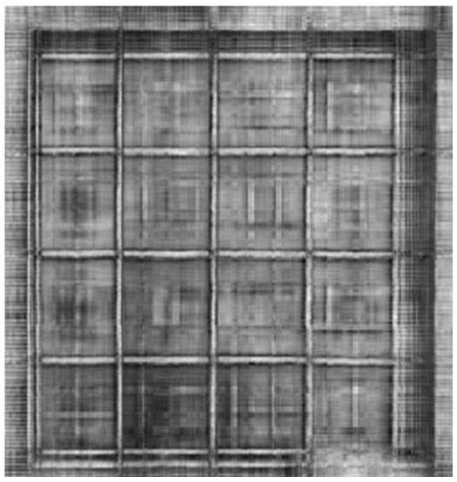
Detail from Durer's Melancolia, dated 1514., 359x371 image



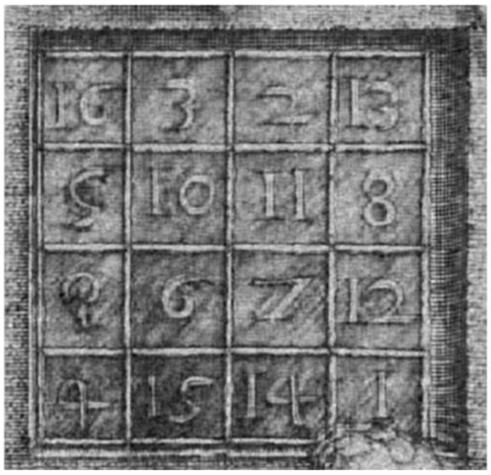
EOF reconstruction with 1 modes



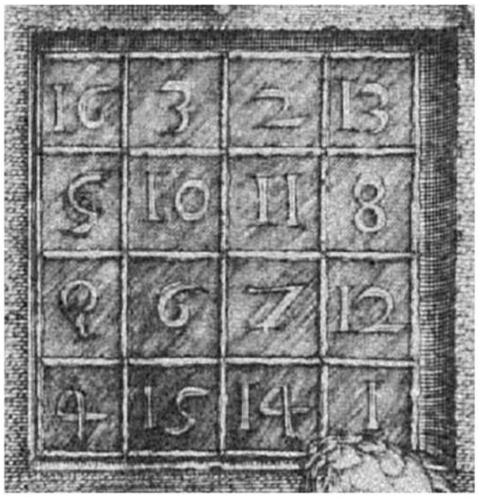
EOF reconstruction with 10 modes



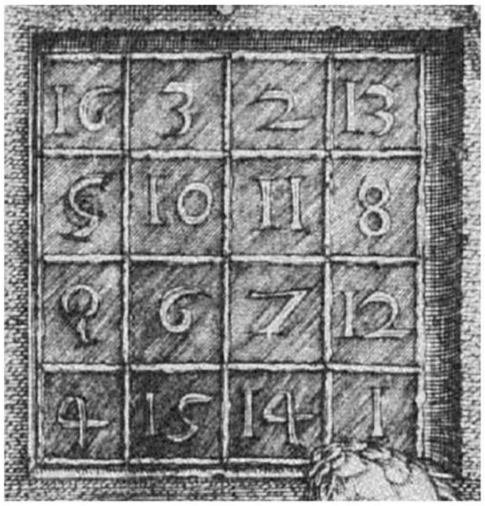
EOF reconstruction with 40 modes

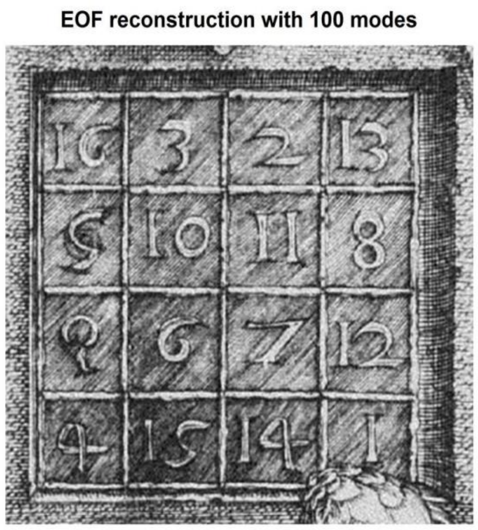


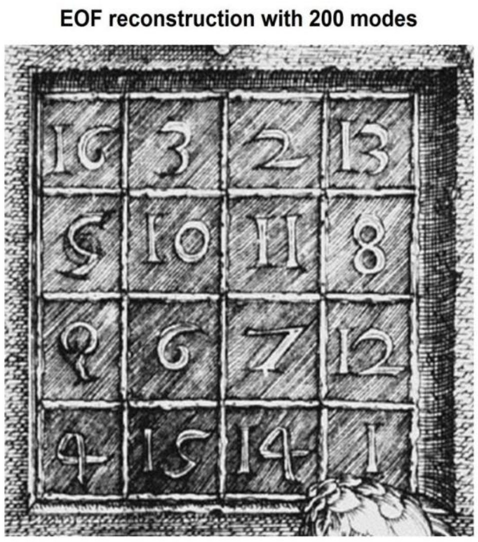
EOF reconstruction with 60 modes



EOF reconstruction with 80 modes







The following chart shows the data usage for increasing values (modes).

Mode	Required Pixels	% Original
10	7,310	5%
40	29,240	22%
60	43,860	33%
80	58,480	44%
100	73,100	55%
Original	133,189	100%

Fig. 4

From the chart it's easy to see that a significant amount of storage space can be saved if you are willing to sacrifice a level of image resolution.

Singular Value Decomposition is an example of a practical application of numerical linear algebra. SVD applications span a wide variety of topics ranging from social network analysis to geology [4]. The importance of SVD and numerical linear algebra will only increase in the future.

MATLAB Code for SVD program:

This code can be found at: http://www.ldeo.columbia.edu/~mspieg/e3101/Matlab/SVD_fun2002.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   matlab script to calculate the SVD of an image
%   This script first reads in an image (a detail of an etching by Durer of a
%   magic square) and plots it.
%
%      It then takes the SVD of the  image and calculates the
%     spectrum of singular values.  Note that only the first 50 or so of the
%     singular values are large (and really only the first 4 or so).  Because the
%     high singular values are negligible, we can reconstruct much of the image just using
%    the first 50 "Empirical orthogonal Functions" (which are just the first 50 Eigenvectors
%    in V.  This script builds up the reconstruction one EOF at a time.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
close all;
%%%%%%
%  load and display the original image
%%%%%%
load detail
[m,n]=size(X);
imagesc(X); colormap(map); axis image; axis off;
set(gca,'fontweight','bold','fontsize',[14]);
title(sprintf('%s, %dx%d image',caption(1,:),m,n));

%%%%%
%  now calculate the SVD of the image
%     and set VT=V transpose
%%%%%

[U,S,V]=svd(X,0);
VT=V';

%%%%%%
%  plot the spectrum of the Singular values sigma_1-sigma_m
%%%%%%

figure
semilogy(diag(S),'b-o');
set(gca,'fontsize',[16],'fontweight','bold');
title('Singular Values');
grid;
%
% okay now do it as sums of rank 1 matrices
%
figure
i=1;
Xi=S(i,i)*U(:,i)*VT(i,:);
imagesc(Xi); colormap(map); axis image; axis off;
set(gca,'fontsize',[16],'fontweight','bold');
title(sprintf('EOF reconstruction with %d modes',i))
disp('Hit return for more modes')
pause;
for i=2:50
  Xi=U(:,1:i)*S(1:i,1:i)*VT(1:i,:);
  imagesc(Xi); colormap(map); axis image; axis off;
  set(gca,'fontsize',[16],'fontweight','bold')
  title(sprintf('EOF reconstruction with %d modes',i))
  pause;
  drawnow;
end
%
% and show for 100 and 200 modes
%
figure
i=100;
```

```
Xi=U(:,1:i)*S(1:i,1:i)*VT(1:i,:);
imagesc(Xi); colormap(map); axis image; axis off;
set(gca,'fontsize',[16],'fontweight','bold');
title(sprintf('EOF reconstruction with %d modes',i))
figure
i=200;
Xi=U(:,1:i)*S(1:i,1:i)*VT(1:i,:);
imagesc(Xi); colormap(map); axis image; axis off;
set(gca,'fontsize',[16],'fontweight','bold');
title(sprintf('EOF reconstruction with %d modes',i))
```

Works Cited

[1] Cooper, Ian & Lorenc, Craig. "Image Compression Using SVD." SVD_Slideshow.pdf 13. Dec 2006. 8 Dec. 2013 <http://online.redwoods.edu/instruct/darnold/laproj/fall2006/iancraig/SVD_Slideshow.pdf>

[2] LoBue, Janine. "Using the Singular Value Decomposition for Image Compression." svd.pdf No date given. 8 Dec. 2013 <www.math.ucsd.edu/~jlobue/102/**svd**.pdf>

[3] Spiegelman, Marc. "Applications of the SVD." SVD_Applications.pdf No Date Given. 8 Dec. 2013 <http://www.columbia.edu/itc/applied/e3101/SVD_applications.pdf>

[4] Martin, Carla & Porter, Mason. "The Extraordinary SVD." The Mathematical Association of America December 2012: 838-848. <http://people.maths.ox.ac.uk/~porterm/papers/s4.pdf>