A Review of a Singularly Valuable Decomposition: The SVD of a Matrix

Reed Tillotson

June 6, 2013

Introduction

Dan Kalman begins his article by claiming that "Every teacher of linear algebra should be familiar with the matrix *singular value decomposition.*" He emphasizes that the singular value decomposition (which we will refer to as the SVD from here on) is not far beyond the scope of a first course in linear algebra, and that it also has significance in theoretical and practical applications. Kalman's primary goal in his paper is to make more people aware of the SVD, particularly linear algebra teachers.

The main body of this paper is split into two parts: the first part is the theory related to the SVD, and the second part covers some of its applications. In the theory part we will cover some important definitions related to the SVD, and compare the SVD of a matrix A with the eigenvalue decomposition of A^TA, which results in a degree of uniqueness for the SVD. Following that we define show how to analytically calculate SVD. The application part of this paper covers how the SVD is used to calculate linear least squares, and how to compress data using reduced rank approximations.

The SVD

Some definitions:

Let A be an m by n matrix. Then the **SVD** of A is $A = U\Sigma V^T$ where U is m by m, V is n by n, and Σ is an m by n diagonal matrix where the diagonal entries $\Sigma_{ii} = \sigma_i$ are nonnegative, and are arranged in non-increasing order. The positive diagonal entries are called the **singular values** of A. The columns of U are called the **left singular vectors** for A, and the columns of V are called the **right singular values** for A.

The **outer product** of the vectors x and y is xy^T. Note that x and y do not have to be of the same length, and that the outer product is a matrix. In particular, it is a matrix of rank one as each column is linearly dependent on x.

Let X be an m by k matrix, and Y a k by n matrix. The **outer product expansion** of two matrices X and Y is

$$XY = \sum_{i=1}^{k} x_i y_i^T$$

where x_i are the columns of X and y_i^T are the rows of Y.

The **outer product expansion of the SVD** is given by

$$A = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

where u_i and v_i are the columns of U and V respectively, and k is the number of non-trivial singular values of A.

The **Frobenius norm** $|X|$ of a matrix X is the square root of the sum of the squares of its entries. Note that this coincides with the 2-norm of a column vector.

Define the **inner product** of two matrices to be $X \cdot Y = \sum_{ij} x_{ij} y_{ij}$, so $|X|^2 = X \cdot X$ as usual.

Some Results:

The SVD and the eigenvalue decomposition are similar and closely related. The SVD of A can be used to determine the eigenvalue decomposition (which will hence be referred to as the EVD) of A^TA. In particular, the right singular vectors of A are the eigenvectors of A^TA, and the singular values of A are equal to the square roots of the eigenvalues of A^TA. It also turns out that the left singular vectors of A are the eigenvectors of AA^T. This shows that if the SVD of A is $A = U\Sigma V^T$, U and V are uniquely determined except for orthogonal basis transformations in the eigenspaces of A^TA and AA^T. Kalman also shows that the EVD and the SVD are the same for a square and symmetric matrix, except that the singular values are actually the absolute values of the eigenvalues.

How to analytically calculate a SVD of a matrix:

If A is an m by n matrix, then it can be expressed as $U\Sigma V^T$. V is the matrix such that $A^TA = VDV^T$ with the diagonal entries of D arranged in non-increasing order. The nonzero entries of Σ are equal to the square roots of the corresponding entries of D. U is made up of the non-vanishing normalized columns of AV extended to an orthonormal basis in $\mathbf{R}^m$. This results in V and U being orthogonal matrices.

Proof: First off, we will find an orthonormal basis for $\mathbf{R}^n$ such that its image under A is orthogonal. Let A be of rank k such that k is less than n. Let $A^TA = VDV^T$ be the EVD of A^TA with the diagonal entries λ_i of D be arranged in non-increasing order, and the columns of V be the orthonormal basis $\{v_1, v_2, \ldots, v_n\}$ since symmetric matrices have orthogonal eigenvectors. Then

$$Av_i \cdot Av_j = (Av_i)^T(Av_j) = v_i^T A^T A v_j = v_i^T(\lambda_i v_j) = \lambda_j v_i \cdot v_j$$

so the we clearly have an orthonormal basis for $\mathbf{R}^n$ with an orthogonal image. Also, the nonzero vectors in this image set form a basis for the range of A.

We have V, so next we will normalize the vectors Av_i to form U. If $i = j$ in the equation above, we get $|Av_i|^2 = \lambda_i$. Then $\lambda_i \geq 0$ so, since they were assumed to be in non-increasing order, $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_k \geq 0$ and λ_i is zero for all i greater than k because k is the rank of A. With this we have an orthonormal basis for the range of A defined as

$$u_i = \frac{Av_i}{|Av_i|} = \frac{1}{\sqrt{\lambda_i}} Av_i \text{ for } 1 \leq i \leq k$$

Extend this to an orthonormal basis in $\mathbf{R}^m$ if $k < m$.

Now all that is left is to determine Σ, and then show that we have the SVD of A. Set $\sigma_i = \sqrt{\lambda_i}$. Then by the equation above, we have $Av_i = \sigma_i u_i$ for all $i \leq k$. Putting the v_i together as the columns of a matrix V and the u_i together as the columns of a matrix U we get $AV = U\Sigma$, where Σ has the same size as A, its diagonal entries are σ_i and the rest are zero. Multiplying both sides of $AV = U\Sigma$ by V^T gives us $A = U\Sigma V^T$, which is the SVD of A. ■

Applications

It should be noted that part of what makes the SVD so useful are the efficient and accurate algorithms for computing it. However, this paper is not about that so, like Kalman, we will assume that we have a good algorithm for computing the SVD and just focus on its applications. However, before going over the main applications of the SVD in his paper, Kalman briefly describes these three other applications.

Computing the EVD of the matrix product A^TA is sometimes of interest, but can be prone to a loss of accuracy. However, the SVD can be computed reliably directly from A, and the right singular values of A are the eigenvectors of A^TA and the squares of the singular values of A are the eigenvalues of A^TA. Thus the SVD can be used to accurately determine the EVD of the matrix product A^TA.

Effective rank estimation is another application of the SVD. The SVD of a matrix can be used to numerically estimate the effective rank of a matrix. In the case that the columns of a matrix A represent data the measurement error from obtaining the data can make a matrix seem to be of greater rank than it really should be. By only looking at the singular values of A greater than the measurement error we can determine the effective rank of A.

The third application is just briefly mentioned, and is that the SVD can be used in computing what is called the generalized inverse of a matrix. Now we move to the first main application.

Linear Least Squares

This kind of problem comes up for a variety of reasons, but can be quickly summarized as: we want to approximate a vector b as closely as we can with a linear combination of the vectors in the finite set $\{a_1, a_2, \ldots, a_n,\}$. Written as an expression, we want $x^T = [x_1, x_2, \ldots, x_n,]$ such that

$$\left| b - \sum_{i=1}^{n} x_i a_i \right|$$

is minimized. Let A be the m by n matrix with columns a_i.

There is a nice way of doing this problem analytically without the SVD, but it behaves poorly when implemented on a computer, and this is a paper about the SVD. Let the SVD of A be $U\Sigma V^T$. Then we have

$$\begin{aligned} b - \textstyle\sum_{i=1}^{n} x_i a_i = Ax - b &= U\Sigma V^T x - b \\ &= U(\Sigma V^T x) - U(U^T b) \\ &= U(\Sigma y - c) \end{aligned}$$

where $y = V^T x$ and $c = U^T b$. U is an orthogonal matrix so $|U(\Sigma y - c)| = |\Sigma y - c|$, and thus $|Ax - b| = |\Sigma y - c|$. This shows that solving for y will be equivalent to solving for x once we know the SVD of A. But we already know the SVD of A is $U\Sigma V^T$ so this will be easy.

We want y to minimize $|\Sigma y - c|$. Let the nonzero diagonal entries of Σ be σ_i for $1 \le i \le k$ for k equal to the rank of A. Then

$$(\Sigma \mathrm{y})^T = [\sigma_1 y_1, \sigma_2 y_2, \dots, \sigma_k y_k]$$

So

$$(\Sigma \mathrm{y} - \mathrm{c})^T = [\sigma_1 y_1 - c_1, \dots, \sigma_k y_k - c_k, -c_{k+1}, -c_{k+2}, \dots, -c_m]$$

With a glance we can determine that $y_i = c_i/\sigma_i$ for $1 \le i \le k$ and vanishes for other i minimizes $|\Sigma y - c|$, which then equals

$$\left[\sum_{i=k+1}^{m} c_i^2\right]^{1/2}$$

This is the error left of the least squares approximation, which goes to zero when k equals m. To determine the actual error we need to know c_i, but we already do as we defined c as equal to $U^T b$. In the interest of writing x nicely in terms of b and the SVD of A, define Σ^+ to be the transpose of Σ with its nonzero entries inverted. Then $y = \Sigma^+ c$. Thus we get the solution

$$x = V\Sigma^+ U^T b$$

Although Kalman provides a detailed but fairly simple example of implementing this method of least squares approximation using MATLAB, we will skip it and move on to the other main application of the SVD that Kalman covers.

Reduced Rank Approximations and Data Compression Using the SVD

Think of an m by n matrix A as just a matrix of numerical data, not a transformation. A is just an organized collection of mn numbers. In real applications m and n can be very large, so we are interested in finding an approximation that represents just the most significant features of A.

The rank of a matrix is the number of linearly independent columns (or rows) it has, and so rank can be thought of as a measure of redundancy. Particularly that lower rank corresponds to lots of redundancy, and high rank corresponds to lower redundancy. A low rank matrix can be represented more efficiently than a high rank matrix. In the case of a rank one matrix B, the columns are all multiples of each other. If u is a single column of B, then let the coefficients v_i be such that the i^{th} column of B can be written as uv_i. Then $B = [uv_1\, uv_2\, \dots uv_n] = uv^T$. Thus we need only $m + n$ numbers to represent B.

Let $U\Sigma V^T$ be the SVD of A. Then $A = \sum_{i=1}^{k} \sigma_i u_i v_i^T$ is the outer product expansion of the SVD of A. It should be noted that the $\sigma_i u_i v_i^T$ are all rank one m by n matrices. We want to show that $A_1 = \sigma_1 u_1 v_1^T$ is the best rank one approximation of A.

Theorem: $A_1 = \sigma_1 u_1 v_1^T$ is the best rank one approximation of A as long as σ_1 is distinct.

Proof: First, we need to show some things about matrix inner products. For rank one matrices xy^T and uv^T the matrix inner product is

$$xy^T \cdot uv^T = \sum_i xy_i \cdot uv_i = \sum_i (x \cdot u) y_i v_i = (x \cdot u)(y \cdot u)$$

If x_i are the columns of X and y_i^T are the rows of Y, and the outer product expansion $XY = \sum_i x_i y_i^T$, then

$$XY \cdot XY = \sum_{ij} (x_i \cdot x_j)(y_i \cdot y_j) = \sum_i |x_i|^2 |y_i|^2 + \sum_{i \neq j} (x_i \cdot x_j)(y_i \cdot y_j)$$

If X is orthogonal, then

$$|XY|^2 = \sum_i |x_i|^2 |y_i|^2$$

If x_i are also all of unit length, then

$$|XY|^2 = \sum_i |y_i|^2 = |Y|^2$$

Similar conclusions can be drawn if Y is orthogonal. This has shown that multiplying by an orthogonal matrix does not change the norm of a matrix. Thus $|A|^2 = \sum \left|\sigma_i u_i v_i^T\right|^2 = \sum \sigma_i^2$. We can partition this sum as $|A|^2 = |S_r|^2 + |E_r|^2$, where S_r is the sum of the first r terms, and E_r is the sum of the remaining terms.

We now want to show that $\sigma_2^2 + \cdots + \sigma_k^2$ is the smallest possible error. Let $U\Sigma V^T$ be the SVD of A. Since the Frobenius norm is preserved for any rank one matric A_1, we have $|A - A_1| = |U\Sigma V^T - A_1| = |\Sigma - U^T A_1 V|$. Rewrite $U^T A_1 V$ as $\alpha x y^T$ where α is positive, $x \in \boldsymbol{R^m}$ and $y \in \boldsymbol{R^n}$. Next, by the properties of the matrix inner product:

$$|\Sigma - \alpha x y^T|^2 = |\Sigma|^2 + \alpha^2 - 2\alpha\Sigma \cdot xy^T$$

Now for some estimates on the last term:

$$\begin{aligned} \Sigma \cdot xy^T &= \sum_{i=1}^{k} \sigma_i x_i y_i \\ &\leq \sum_{i=1}^{k} \sigma_i |x_i||y_i| \\ &\leq \sigma_1 \sum_{i=1}^{k} |x_i||y_i| \end{aligned}$$

By the Cauchy-Schwarz inequality, $\sum_{i=1}^{k} |x_i||y_i| \leq \sum_{i=1}^{k} |x_i| \cdot \sum_{i=1}^{k} |y_i| \leq |x||y| = 1$. Thus $\Sigma \cdot xy^T \leq \sigma_1$.

Getting back to where we were, we now see $|\Sigma - \alpha xy^T|^2 \geq |\Sigma|^2 + \alpha^2 - 2\alpha\sigma_1 = |\Sigma|^2 + (\alpha - \sigma_1)^2 - \sigma_1^2$. Clearly, we can minimize the right-hand side by setting α equal to σ_1, then $|\Sigma - \alpha xy^T|^2 \geq |\Sigma|^2 - \sigma_1^2$. Note that this minimum is only obtained when $|x_1|$ = $|y_1|$ = 1 and α is equal to σ_1. Then $A_1 = \alpha Uxy^TV^T = \sigma_1(Ue_1)(Ve_1)^T = \sigma_1 u_1 v_1^T$. ■

It turns out that $S_r = \sum_{i=1}^{r} \sigma_i u_i v_i^T$ defines a rank *r* matrix that is the best rank *r* approximation to *A*, and its error is $E_r = A - S_r = \sum_{i=1}^{r} \sigma_i u_i v_i^T$ with $|E_r|^2 = \sum_{i=r+1}^{k} {\sigma_i}^2$. However Kalman just gives a couple of references to the proof. When implementing the SVD in practice it can be used to choose the rank *r* with which to approximate a matrix. This typically based on requirements on relative error. The relative error of the SVD outer product expansion is

$$\frac{|E_r|}{|A|} = \sqrt{\frac{\sum_{i=r+1}^{k} {\sigma_i}^2}{\sum_{i=1}^{k} {\sigma_i}^2}}$$

Some Examples:

Although Kalman uses an example in his paper of image compression, the example features only one image being compressed and it is very simple. Here we will go over multiple images to show how adaptable the SVD is, and how visible image qualities can be affected by compression.

Here are three images that will be used as examples:

Let us call these images in order: eye, tree, and nope. These pictures are all black and white to simplify the MATLAB code used to compress these images. The code used will be put at the end of this section.

Here are the sizes in pixels of each image: eye is 168 by 280 pixels, tree is 599 by 900 pixels, and nope is 1500 by 1050 pixels. As you can see right away, the images have much different sizes, but the SVD can be easily implemented even so. Here are the first fifteen singular values for our images followed by the images compressed to within an accuracy threshold of 25%:

Eye: 106.8590, 27.1434, 18.0849, 15.1460, 14.9499, 12.1669, 11.0146, 8.6290, 8.1337, 7.4696, 6.4867, 6.2537, 5.8443, 5.1062, 5.0424
Tree: 399.9686, 75.1662, 42.1528, 39.5619, 22.1923, 20.8705, 18.9550, 17.4170, 14.3010, 13.7155, 12.3462, 10.9162, 10.1470, 9.9213, 9.3591
Nope: 873.8819, 217.7515, 183.9686, 170.7199, 159.5952, 140.5849, 113.6305, 106.1437, 94.3424, 92.9146, 86.6377, 70.3212, 63.1933, 59.7077, 52.8584.

The 25% accuracy threshold means that the error is at most 75%, but even so we can still get the gist of the compressed images. It should also be noted that none of these images used their 14th or 15th singular value to display these compressed images, and that the first singular value is much larger than the rest. The next set of images is for an accuracy threshold of 75%. As we can see, they look pretty spiffy. This time around eye kept 101 singular values, tree kept 276, and nope kept 362. Comparatively, nope is throwing away a higher fraction (roughly two-thirds) of its singular values than tree and especially eye (just over one-third). This likely due to nope having a simpler design compared to its size, while eye has lots of fine detail but a small size.

It should be mentioned here as Kalman did as well, that the reduced rank approximations using the SVD and Fourier analysis work in similar ways. Fourier analysis is used a lot in signal analysis, and is also used in data compression. However, the SVD finds the best possible basis for a given rank, while Fourier analysis always uses the same basis regardless of the rank.

This is the MATLAB code used to do the image compression:

```
%loads the image that will be compressed.
imageRaw = imread('eye.jpg');

%converts the image to grayscale.
image2 = rgb2gray(imageRaw);
%converts the image from uint8 to double.
image3 = im2double(image2);
```

```
%displays image3.
imshow(image3);

%calculates SVD of image3.
[U,S,V] = svd(image3);

%calculate |image3|
absIm = sqrt(sum(diag(S)));
%relative error.
relErr = 1;

%R will be the reduced rank matrix of S.
R = zeros(size(S,1), size(S,2));
%sets the diagonal entries of R to the singular
%values of image3 to get a reduced rank approximation.
i = 1;
while relErr > 0.75     %keeps executing the loop
    R(i,i) = S(i,i);    %until the relative error gets
    i = i + 1;          %low enough.
    %calculates the relative error.
    relErr = sqrt(sum(diag(S - R))) / absIm;
end

%recreates the image from the reduced
%rank SVD.
A = U*R*V';
%display A.
imshow(A);
```

References

1. D Kalman, A Singularly Valuable Decomposition: The SVD of a Matrix, The American University, February 13, 2002.