A Singularly Valuable Decomposition: The SVD of a Matrix

Dan Kalman

Dan Kalman is an assistant professor at American University in Washington, DC. From 1985 to 1993 he worked as an applied mathematician in the aerospace industry. It was during that period that he first learned about the SVD and its applications. He is very happy to be teaching again and is avidly following all the discussions and presentations about the teaching of linear algebra.

Every teacher of linear algebra should be familiar with the matrix *singular value decomposition* (or SVD). It has interesting and attractive algebraic properties, and conveys important geometrical and theoretical insights about linear transformations. The close connection between the SVD and the well-known theory of diagonalization for symmetric matrices makes the topic immediately accessible to linear algebra teachers and, indeed, a natural extension of what these teachers already know. At the same time, the SVD has fundamental importance in several different applications of linear algebra.

Gilbert Strang was aware of these facts when he introduced the SVD in his now classical text [**22**, p. 142], observing that "it is not nearly as famous as it should be." Golub and Van Loan ascribe a central significance to the SVD in their definitive explication of numerical matrix methods [**8**, p. xiv], stating that "perhaps the most recurring theme in the book is the practical and theoretical value" of the SVD. Additional evidence of the SVD's significance is its central role in a number of recent papers in *Mathematics Magazine* and the *American Mathematical Monthly*; for example, [**2**, **3**, **17**, **23**].

Although it is probably not feasible to include the SVD in the first linear algebra course, it definitely deserves a place in more advanced undergraduate courses, particularly those with a numerical or applied emphasis. My primary goals in this article are to bring the topic to the attention of a broad audience, and to reveal some of the facets that give it both practical and theoretical significance.

Theory

The SVD is intimately related to the familiar theory of diagonalizing a symmetric matrix. Recall that if A is a symmetric real $n \times n$ matrix, there is an orthogonal matrix V and a diagonal D such that $A = VDV^T$. Here the columns of V are eigenvectors for A and form an orthonormal basis for $\mathbb{R}^n$; the diagonal entries of D are the eigenvalues of A. To emphasize the connection with the SVD, we will refer to VDV^T as the *eigenvalue decomposition* (EVD) for A.

For the SVD we begin with an arbitrary real $m \times n$ matrix A. As we shall see, there are orthogonal matrices U and V and a diagonal matrix, this time denoted Σ, such that $A = U\Sigma V^T$. In this case, U is $m \times m$ and V is $n \times n$, so that Σ is rectangular with the same dimensions as A. The diagonal entries of Σ, that is the $\Sigma_{ii} = \sigma_i$, can be arranged to be nonnegative and in order of decreasing magnitude; the positive ones are called the *singular values* of A. The columns of U and V are called left and right *singular vectors* for A.

The analogy between the EVD for a symmetric matrix and the SVD for an arbitrary matrix can be extended a little by thinking of matrices as linear transformations. For a symmetric matrix A, the transformation takes $\mathbb{R}^n$ to itself, and the columns of V define an especially nice basis. When vectors are expressed relative to this basis, we see that the transformation simply dilates some components and contracts others, according to the magnitudes of the eigenvalues (with a reflection through the origin tossed in for negative eigenvalues). Moreover, the basis is orthonormal, which is the best kind of basis to have.

Now let's look at the SVD for an $m \times n$ matrix A. Here the transformation takes $\mathbb{R}^n$ to a different space, $\mathbb{R}^m$, so it is reasonable to ask for a natural basis for each of domain and range. The columns of V and U provide these bases. When they are used to represent vectors in the domain and range of the transformation, the nature of the transformation again becomes transparent: It simply dilates some components and contracts others, according to the magnitudes of the singular values, and possibly discards components or appends zeros as needed to account for a change in dimension. From this perspective, the SVD tells us how to choose orthonormal bases so that the transformation is represented by a matrix with the simplest possible form, that is, diagonal.

How do we choose the bases $\{v_1, v_2, \ldots, v_n\}$ and $\{u_1, u_2, \ldots, u_m\}$ for the domain and range? There is no difficulty in obtaining a diagonal representation. For that, we need only $Av_i = \sigma_i u_i$, which is easily arranged. Select an orthonormal basis $\{v_1, v_2, \ldots, v_n\}$ for $\mathbb{R}^n$ so that the first k elements span the row space of A and the remaining $n - k$ elements span the null space of A, where k is the rank of A. Then for $1 \leq i \leq k$ define u_i to be a unit vector parallel to Av_i, and extend this to a basis for $\mathbb{R}^m$. Relative to these bases, A will have a diagonal representation. But in general, although the v's are orthogonal, there is no reason to expect the u's to be. The possibility of choosing the v-basis so that its orthogonality is preserved under A is the key point. We show next that the EVD of the $n \times n$ symmetric matrix A^TA provides just such a basis, namely, the eigenvectors of A^TA.

Let $A^TA = VDV^T$, with the diagonal entries λ_i of D arranged in non-increasing order, and let the columns of V (which are eigenvectors of A^TA) be the orthonormal basis $\{v_1, v_2, \ldots, v_n\}$. Then

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

so the image set $\{Av_1, Av_2, \ldots, Av_n\}$ is orthogonal, and the nonzero vectors in this set form a basis for the range of A. Thus, the eigenvectors of A^TA and their images under A provide orthogonal bases allowing A to be expressed in a diagonal form.

To complete the construction, we normalize the vectors Av_i. The eigenvalues of A^TA again appear in this step. Taking $i = j$ in the calculation above gives $|Av_i|^2 = \lambda_i$, which means $\lambda_i \geq 0$. Since these eigenvalues were assumed to be arranged in non-increasing order, we conclude that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k \geq 0$ and, since the rank of A is k, $\lambda_i = 0$ for $i > k$. The orthonormal basis for the range is therefore defined by

$$u_i = \frac{Av_i}{|Av_i|} = \frac{1}{\sqrt{\lambda_i}} Av_i, \qquad 1 \leq i \leq k.$$

If $k < m$, we extend this to an orthonormal basis for $\mathbb{R}^m$.

This completes the construction of the desired orthonormal bases for $\mathbb{R}^n$ and $\mathbb{R}^m$. Setting $\sigma_i = \sqrt{\lambda_i}$ we have $Av_i = \sigma_i u_i$ for all $i \leq k$. Assembling the v_i as the columns of a matrix V and the u_i to form U, this shows that $AV = U\Sigma$, where Σ has the same dimensions as A, has the entries σ_i along the main diagonal, and has all other entries equal to zero. Hence, $A = U\Sigma V^T$, which is the singular value decomposition of A.

In summary, an $m \times n$ real matrix A can be expressed as the product $U\Sigma V^T$, where V and U are orthogonal matrices and Σ is a diagonal matrix, as follows. The matrix V is obtained from the diagonal factorization $A^TA = VDV^T$, in which the diagonal entries of D appear in non-increasing order; the columns of U come from normalizing the nonvanishing images under A of the columns of V, and extending if necessary to an orthonormal basis for $\mathbb{R}^m$; the nonzero entries of Σ are the square roots of corresponding diagonal entries of D.

The preceding construction demonstrates that the SVD exists, and gives some idea of what it tells about a matrix. There are a number of additional algebraic and geometric insights about the SVD that will be derived with equal ease. Before proceeding to them, two remarks should be made. First, the SVD encapsulates the most appropriate bases for the domain and range of the linear transformation defined by the matrix A. A beautiful relationship exists between these bases and the four fundamental subspaces associated with A: the range and nullspace, and their orthogonal complements. It is the full picture provided by the SVD and these subspaces that Strang has termed the *fundamental theorem of linear algebra*. He also invented a diagram schematically illustrating the relationship of the bases and the four subspaces; see Figure 1. Strang's article [**23**] is recommended for a detailed discussion of this topic.

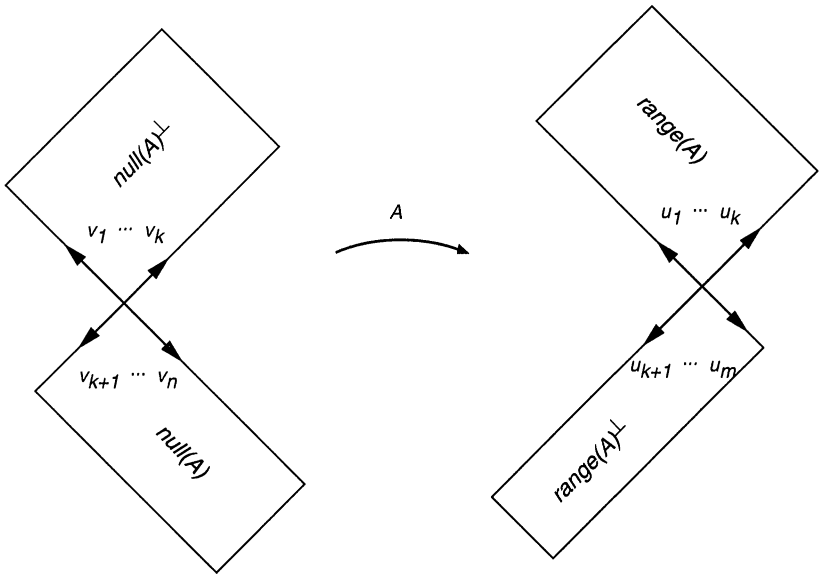

Figure 1. Strang's diagram.

The second remark concerns computation. There is often a gap between mathematical theory and computational practice. In theory, we envision arithmetic operations being carried out on real numbers in infinite precision. But when we carry out arithmetic on a digital computer we compute not with reals, but with a finite set of rationals, and the results can only approximate with limited precision the real computations. Procedures that seem elegant and direct in theory can sometimes be dramatically flawed as prescriptions for computational algorithms. The SVD illustrates this point. Our construction appears to offer a straightforward algorithm for the SVD: Form A^TA, compute its eigenvalues and eigenvectors, and then find the SVD as described above. Here practice and theory go their separate ways. As we shall see later, the computation using A^TA can be subject to a serious loss of precision. It turns out that direct methods exist for finding the SVD of A without forming A^TA, and indeed in many applications the practical importance of the SVD is that it allows one to find the EVD of A^TA without ever forming this numerically treacherous product.

Let us now explore several additional aspects of the SVD.

The SVD and the EVD. Our construction showed how to determine the SVD of A from the EVD of the symmetric matrix A^TA. Conversely, it is easy to recover the EVD of A^TA from the SVD of A. For suppose the singular value decomposition $A = U\Sigma V^T$ is given. Clearly, $A^TA = V\Sigma^T\Sigma V^T$ and $AA^T = U\Sigma\Sigma^T U^T$. Now in either order the product of Σ and Σ^T is a square diagonal matrix whose first k diagonal entries are the σ_i^2, with any remaining diagonal entries equal to 0. Thus, $A^TA = V\Sigma^T\Sigma V^T$ is the EVD of A^TA and $AA^T = U\Sigma\Sigma^T U^T$ is the EVD of AA^T. Our argument also yields a uniqueness result for the singular value decomposition. In any SVD of A, the right singular vectors (columns of V) must be the eigenvectors of A^TA, the left singular vectors (columns of U) must be the eigenvectors of AA^T, and the singular values must be the square roots of the nonzero eigenvalues common to these two symmetric matrices. Thus, up to possible orthogonal transformations in multidimensional eigenspaces of A^TA and AA^T, the matrices V and U in the SVD are uniquely determined. Finally, note that if A itself is square and symmetric, each eigenvector for A with eigenvalue λ is an eigenvector for $A^2 = A^TA = AA^T$ with eigenvalue λ^2. Hence the left and right singular vectors for A are simply the eigenvectors for A, and the singular values for A are the absolute values of its eigenvalues. That is, the EVD and SVD essentially coincide for symmetric A and are actually identical if A has no negative eigenvalues. In particular, for any A, the SVD and EVD of A^TA are the same.

A geometric interpretation of the SVD. One way to understand how A deforms space is to consider its action on the unit sphere in $\mathbb{R}^n$. An arbitrary element x of this unit sphere can be represented by $x = x_1v_1 + x_2v_2 + \cdots + x_nv_n$ with $\sum_1^n x_i^2 = 1$. The image is $Ax = \sigma_1x_1u_1 + \cdots + \sigma_kx_ku_k$. Letting $y_i = \sigma_ix_i$, we see that the image of the unit sphere consists of the vectors $y_1u_1 + y_2u_2 + \cdots + y_ku_k$, where

$$\frac{y_1^2}{\sigma_1^2} + \frac{y_2^2}{\sigma_2^2} + \cdots + \frac{y_k^2}{\sigma_k^2} = \sum_1^k x_i^2 \le 1.$$

If A has full column rank, so that $k = n$, the inequality is actually a strict equality. Otherwise, some of the x_i are missing on the right, and the sum can be anything from 0 to 1. This shows that A maps the unit sphere of $\mathbb{R}^n$ to a k-dimensional ellipsoid with semi-axes in the directions u_i and with the magnitudes σ_i. If $k = n$ the image is just the surface of the ellipsoid, otherwise it is the solid ellipsoid. In summary, we can visualize the effect A as follows: It first collapses $n - k$ dimensions of the domain, then distorts the remaining dimensions, stretching and squeezing the unit k-sphere into an ellipsoid, and finally it embeds the ellipsoid in $\mathbb{R}^m$. This is illustrated for $n = m = 3$ and $k = 2$ in Figure 2.

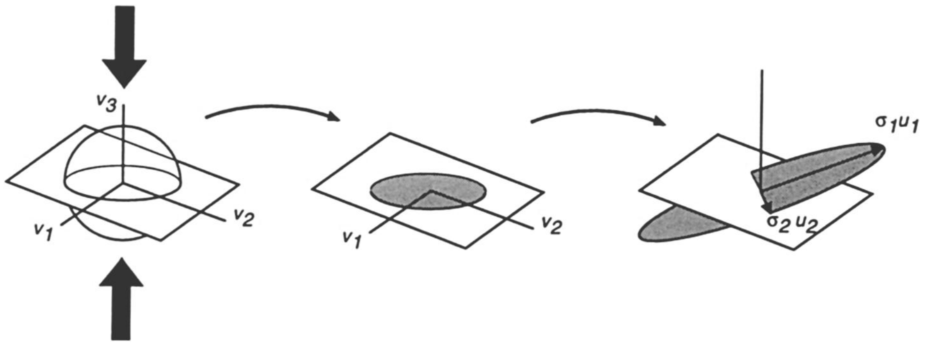

Figure 2. How A deforms $\mathbb{R}^n$.

As an immediate consequence, we see that $\|A\|$, the *operator norm* of A, defined as the maximum value of $|Av|$ for v on the unit sphere, is simply σ_1, the largest singular value of A. Put another way, we have the inequality $|Ax| \leq \sigma_1|x|$ for all $x \in\mathbb{R}^n$, with equality only when x is a multiple of v_1.

Partitioned matrices and the outer product form of the SVD. When viewed in a purely algebraic sense, any zero rows and columns of the matrix Σ are superfluous. They can be eliminated if the matrix product $A = U\Sigma V^T$ is expressed using partitioned matrices as follows:

$$A = [u_1 \ \dots \ u_k \mid u_{k+1} \ \dots \ u_m] \left[\begin{array}{ccc|c} \sigma_1 & & & \\ & \ddots & & 0 \\ & & \sigma_k & \\ \hline & 0 & & 0 \end{array}\right] \left[\begin{array}{c} v_1^T \\ \vdots \\ v_k^T \\ \hline v_{k+1}^T \\ \vdots \\ v_n^T \end{array}\right]$$

Although these partitions assume that k is strictly less than m and n, it should be clear how to modify the arguments if k is equal to m or n. When the partitioned matrices are multiplied, the result is

$$A = [u_1 \ \dots \ u_k] \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix} \begin{bmatrix} v_1^T \\ \vdots \\ v_k^T \end{bmatrix} + [u_{k+1} \ \dots \ u_m]\,[0] \begin{bmatrix} v_{k+1}^T \\ \vdots \\ v_n^T \end{bmatrix}$$

From this last equation it is clear that only the first k of the u's and v's make any contribution to A. Indeed, we may as well shorten the equation to

$$A = [u_1 \ \dots \ u_k] \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix} \begin{bmatrix} v_1^T \\ \vdots \\ v_k^T \end{bmatrix}.$$

Notice that in this form the matrices of u's and v's are now rectangular ($m \times k$ and $k \times n$ respectively), and the diagonal matrix is square. This is an alternative version of the SVD that is taken as the definition in some expositions: Any $m \times n$ matrix A of rank k can be expressed in the form $A = U\Sigma V^T$ where U is an $m \times k$ matrix such that $U^TU = I$, Σ is a $k \times k$ diagonal matrix with positive entries in decreasing order on the diagonal, and V is an $n \times k$ matrix such that $V^TV = I$.

The partitioned matrix formulation of the SVD is a little unusual in one respect. Usually in a matrix product XY, we focus on the rows in X and on the columns in Y. Here, the factors are expressed in just the opposite way. This is the ideal situation to apply what is called an *outer product expansion* for the product of two matrices. In general, if X is an $m \times k$ matrix with columns x_i and Y is a $k \times n$ matrix with rows y_i^T, the matrix product XY can be expressed as

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

Each of the terms $x_i y_i^T$ is an outer product of vectors x_i and y_j. It is simply the standard matrix product of a column matrix and a row matrix. The result can be visualized in terms of a multiplication table, with the entries of x_i listed along the left margin and those of y_j across the top. The body of the table is the outer product of x_i and y_j. This idea is illustrated in Figure 3, showing the outer product of (a, b, c) and (p, q, r). Observe that in the figure, each column is a multiple of $[\ a \quad b \quad c\ ]^T$ and each row is a multiple of $[\ p \quad q \quad r\ ]$, so that the outer product is clearly of rank 1. In just the same way, the outer product $x_i y_i^T$ is a rank 1 matrix with columns that are multiples of x_i and rows that are multiples of y_i^T.

	p	q	r
a	ap	aq	ar
b	bp	bq	br
c	cp	cq	cr

Figure 3. Outer product as multiplication table.

We shall return to outer product expansions in one of the applications of the SVD. Here, we simply apply the notion to express the SVD of A in a different form. Let

$$X = [u_1 \ \dots \ u_k] \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix} = [\sigma_1 u_1 \ \dots \ \sigma_k u_k] \quad \text{and}$$

$$Y = \begin{bmatrix} v_1^T \\ \vdots \\ v_k^T \end{bmatrix}.$$

Then $A = XY$ can be expressed as an outer product expansion,

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

This is yet another form of the SVD, and it provides an alternative way of expressing how A transforms an arbitrary vector x. Clearly,

$$Ax = \sum_{i=1}^{k} \sigma_i u_i v_i^T x.$$

Since $v_i^T x$ is a scalar, we can rearrange the order of the factors to

$$Ax = \sum_{i=1}^{k} v_i^T x \sigma_i u_i.$$

Now in this sum Ax is expressed as a linear combination of the vectors u_i. Each coefficient is a product of two factors, $v_i^T x$ and σ_i. Of course, $v_i^T x = v_i \cdot x$ is just the ith component of x relative to the orthonormal basis $\{v_1, \ldots, v_n\}$. Viewed in this light, the outer product expansion reconfirms what we already know: Under the action of A each v component of x becomes a u component after scaling by the appropriate σ.

Applications

Generally, the SVD finds application in problems involving large matrices, with dimensions that can reach into the thousands. It is the existence of efficient and accurate computer algorithms for its computation that makes the SVD so useful in these applications. There is beautiful mathematics involved in the derivation of the algorithms, and the subject is worth looking into. However, for this discussion I will treat the computation of the SVD as if performed by a black box. By way of analogy, consider any application of trigonometry. When we require the value of a sine or cosine, we simply push the buttons on a calculator with never a thought to the internal workings. We are confident that the calculator returns a close enough approximation to serve our purposes and think no more about it. So, too, we can be confident that computers will quickly and accurately approximate the SVD of arbitrary matrices, letting us concentrate on when and why to push the SVD button, and how to use the results.

The SVD is an important tool in several different applications. I will briefly mention a few, then discuss in detail two samples that are apparently quite unrelated: linear least squares optimization and data compression with reduced rank approximations.

Applications in brief. One of the most direct applications of the SVD is to the problem of computing the EVD of a matrix product A^TA. This type of problem is encountered frequently under the name of *principal component analysis*[1] and in connection with the statistical analysis of covariance matrices [**9**]. As the discussion of least squares optimization will make clear, the computation of A^TA can lead to a significant degradation of accuracy in the results. In contrast, the SVD can be computed by operating directly on the original matrix A. This gives the desired eigenvectors of A^TA (the right singular vectors of A) and eigenvalues of A^TA (the squares of the singular values of A) without ever explicitly computing A^TA.

[1] An application to digital image processing is described in an introductory section of [**15**]. Succeeding sections discuss the SVD and its other applications. A much more detailed presentation concerning image processing appears in [**21**].

A second application of the SVD is as a numerically reliable estimate of the *effective rank* of a matrix. Often linear dependencies in data are masked by measurement error. Thus, although computationally speaking the columns of a data matrix appear to be linearly independent, with perfect measurement the dependencies would have been detected. Or, put another way, it may be possible to make the columns of the data matrix dependent by perturbing the entries by small amounts, on the same order as the measurement errors already present in the data. The way to find these dependencies is to focus on the singular values that are of a larger magnitude than the measurement error. If there are r such singular values, the effective rank of the matrix is found to be r. This topic is closely related to the idea of selecting the closest rank r approximation to a matrix, which is considered below in the discussion of data compression.

Another application of the SVD is to computing the *generalized inverse* of a matrix. This is very closely related to the linear least squares problem and will be mentioned again in the discussion of that topic.

Linear least squares. The general context of a linear least squares problem is this: We have a set of vectors which we wish to combine linearly to provide the best possible approximation to a given vector. If the set of vectors is $\{a_1, a_2, \ldots, a_n\}$ and the given vector is b, we seek coefficients $x_1, x_2, \ldots, x_n$ that produce a minimal error

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

The problem can arise naturally in any vector space, with elements that are sequences, functions, solutions to differential equations, and so on. As long as we are interested only in linear combinations of a finite set $\{a_1, a_2, \ldots, a_n\}$, it is possible to transform the problem into one involving finite columns of numbers. In that case, define a matrix A with columns given by the a_i, and a vector x whose entries are the unknown coefficients x_i. Our problem is then to choose x minimizing $|b - Ax|$. As before, we denote the dimensions of A by m and n, meaning that the a_i are vectors of length m.

The general least squares problem has a geometric interpretation. We are seeking an element of the subspace S spanned by the a_i that is closest to b. The solution is the projection of b on S, and it is characterized by the condition that the error vector (that is, the vector difference between b and its projection) should be orthogonal to S. Orthogonality to S is equivalent to orthogonality to each of the a_i. Thus, the optimal solution vector x must satisfy $a_i \cdot (Ax - b) = 0$ for all i. Equivalently, in matrix form, $A^T(Ax - b) = 0$.

Rewrite the equation as $A^TAx = A^Tb$, a set of equations for the x_i generally referred to as the *normal equations* for the linear squares problem. Observe that the independence of the columns of A implies the invertibility of A^TA. Therefore, we have $x = (A^TA)^{-1}A^Tb$.

What a beautiful analysis! It is neat, it is elegant, it is clear. Unfortunately, it is also poorly behaved when implemented in approximate computer arithmetic. Indeed, this is the classic example of the gap between theory and practice mentioned earlier. Numerically, the formation of A^TA can dramatically degrade the accuracy of a computation; it is a step to be avoided. A detailed discussion of the reasons for this poor performance is provided in [**8**]. Here, I will be content to give some insight into the problem, using both a heuristic analysis and a numerical example. First, though, let us see how the SVD solves the least squares problem.

We are to choose the vector x so as to minimize $|Ax - b|$. Let the SVD of A be $U\Sigma V^T$ (where U and V are square orthogonal matrices, and Σ is rectangular with the same dimensions as A). Then we have

$$\begin{aligned} Ax - b &= U\Sigma V^T x - b = U(\Sigma V^T x) - U(U^T b) \\ &= U(\Sigma y - c) \end{aligned}$$

where $y = V^T x$ and $c = U^T b$. Now U is an orthogonal matrix, so preserves lengths. That is, $|U(\Sigma y - c)| = |\Sigma y - c|$, hence $|Ax - b| = |\Sigma y - c|$. This suggests a method for solving the least squares problem. First, determine the SVD of A and calculate c as the product of U^T and b. Next, solve the least squares problem for Σ and c. That is, find a vector y so that $|\Sigma y - c|$ is minimal. (We shall see in a moment that the diagonal nature of Σ makes this step trivial.) Now $y = V^T x$, so we can determine x as Vy. That gives the solution vector x as well as the magnitude of the error, $|\Sigma y - c|$.

In effect, the SVD has allowed us to make a change of variables so that the least squares problem is reduced to a diagonal form. In this special form, the solution is easily obtained. We seek y to minimize the norm of the vector $\Sigma y - c$. Let the components of y, which is the unknown, be y_i for $1 \le i \le n$. Then

$$\Sigma y = \begin{bmatrix} \sigma_1 y_1 \\ \vdots \\ \sigma_k y_k \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \text{so} \quad \Sigma y - c = \begin{bmatrix} \sigma_1 y_1 - c_1 \\ \vdots \\ \sigma_k y_k - c_k \\ -c_{k+1} \\ -c_{k+2} \\ \vdots \\ -c_m \end{bmatrix}$$

By inspection, when $y_i = c_i/\sigma_i$ for $1 \le i \le k$, $\Sigma y - c$ assumes its minimal length, which is given by

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

Recall that k is the rank of A, and note that when $k = m$, the sum (1) is vacuous. In this case, the columns of A span $\mathbb{R}^m$ so the least squares problem can be solved with zero error. Also observe that when k is less than n, there is no constraint on the values of y_{k+1} through y_n. These components can be assigned arbitrary values with no effect on the length of $\Sigma y - c$.

As the preceding analysis shows, the SVD allows us to transform a general least squares problem into one that can be solved by inspection, with no restriction on the rank of the data matrix A. Indeed, we can combine the transformation steps and the solution of the simplified problem as follows. First, we know that $c = U^T b$. The calculation of y from c amounts to multiplying by the matrix Σ^+ defined by transposing Σ and inverting the nonzero diagonal entries. Then $y = \Sigma^+ c$ will have its first k entries equal to c_i/σ_i, as required. Any remaining entries (which were previously unconstrained) all vanish. Finally, we compute $x = Vy$. This gives the compact solution

$$x = V\Sigma^{+}U^{T}b. \tag{2}$$

At this point, the subject of generalized inverses surfaces. Given $b \in \mathbb{R}^m$, there is a unique element $x \in \mathbb{R}^n$ of minimal norm for which Ax is nearest b. The mapping from b to that x defines what is called the *Moore-Penrose generalized inverse,* A^+, of A. It is clear that Σ^+, as defined in the solution of the least squares problem for Σ and c, is the Moore-Penrose inverse of Σ. Moreover, equation (2) shows that when $A = U\Sigma V^T$ is the SVD of A, the generalized inverse is given by $A^+ = V\Sigma^+U^T$. A nice treatment of the generalized inverse that is both compact and self-contained is given in [**5**], although it does not mention the SVD. Generalized inverses and their connection with the SVD are also discussed in [**8**, **14**, **22**, **23**].

Now we are in a position to compare the solutions obtained using the SVD and the normal equations. Since the solutions are theoretically equivalent, carrying them out in exact arithmetic must yield the same results. But when executed in limited precision on a computer, they can differ significantly. At the heart of this difference lie the effects of roundoff error. Let's pause, therefore, to consider one important aspect of roundoff error.

Suppose $x = 1.23456789012345 \cdot 10^8$ and $y = 1.23456789012345 \cdot 10^{-1}$. Computing with 15 correct digits, the sum of x and y is given by

$$\begin{array}{rl} & 12345678.9012345 \\ + & .123456789012345 \\ \hline & 12345679.0246912 \end{array}$$

which corresponds to introducing an error in about the eighth decimal place of y. Observe that while each number is represented in the computer with a precision of 15 decimal places, the less significant digits of the smaller term are essentially meaningless. If the two terms differ in magnitude by a sufficient amount (16 orders of magnitude here), the contribution of the lesser term is completely lost when the sum is computed.

This problem shows up in the inversion of linear systems when the system matrix has singular values that vary widely in magnitude. As an example, consider the inversion of A^TA that occurs as part of the normal equations. The normal equations formulation requires that A^TA be nonsingular. If nearly singular, A^TA will have a singular value very near 0, and in this case the normal equations become unreliable. Let us write the singular values of A^TA in decreasing order as $\lambda_1, \lambda_2, \ldots, \lambda_n$. These are actually the eigenvalues of A^TA (as observed earlier) and they are the squares of the singular values of A. The eigenvectors of A^TA are $v_1, v_2, \ldots, v_n$, the right singular vectors of A. With λ_n very near 0, the λ_i will typically cover a wide range of magnitudes.

To dramatize this situation, we will assume that λ_n is several orders of magnitude smaller than λ_1, and that the other λ_i are roughly comparable to λ_1 in magnitude. Geometrically, A^TA maps the unit sphere of $\mathbb{R}^n$ onto an ellipsoid with axes in the direction of its eigenvectors v_i. The length of the axis in the direction of v_n is so much smaller than all the other axes that the ellipsoid appears to flatten out completely and lie in an $(n-1)$-dimensional hyperplane (Figure 4).

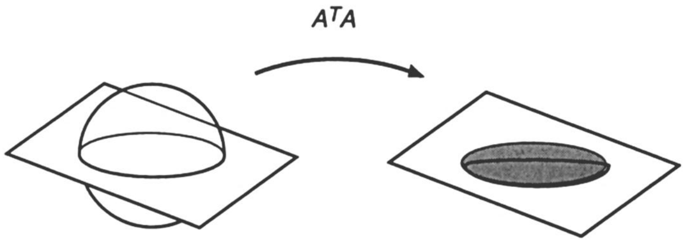

Figure 4. Unit Sphere Deformed by A^TA.

Therefore, in the image under A^TA of a random vector of unit length, the expected contribution of the v_n component is essentially negligible. In this case, the effect of finite precision is to introduce a significant error in the contribution of v_n. However, in solving the normal equations we are interested in the inverse mapping, $(A^TA)^{-1}$, which has the same eigenvectors as A^TA, but whose eigenvalues are the reciprocals $1/\lambda_i$. Now it is λ_n, alone, that is significant, and the ellipsoid is essentially one dimensional; see Figure 5. In every direction except v_n, the image of a random unit vector will have no noticable contribution. Arguing as before, significant errors are introduced in the contributions of all of the v_i except v_n.

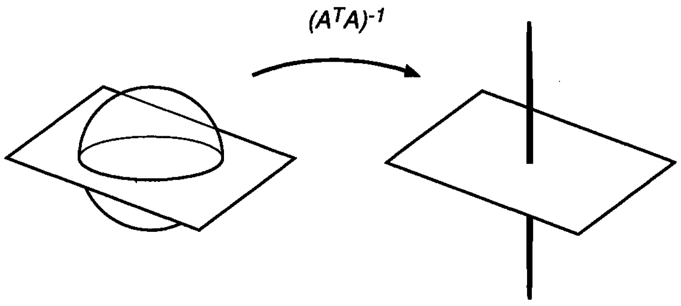

Figure 5. Unit Sphere Deformed by $(A^TA)^{-1}$.

Let us make this analysis more concrete. Select a particular vector x, with $x = x_1v_1 + x_2v_2 + \cdots + x_nv_n$. Thinking of x as a column vector, consider a single entry, c of (x), and write c_i for the corresponding entry of x_iv_i. Then $c = c_1 + c_2 + \cdots + c_n$. In the image $(A^TA)^{-1}x$ the corresponding entry will be $c_1/\lambda_1 + c_2/\lambda_2 + \cdots + c_n/\lambda_n$. Now if the c_i are all of roughly comparable magnitude, then the final term in this sum will dwarf all the rest. The situation illustrated above will occur. The c_n/λ_n term will play the role of 12345678.9012345; every other term will be in the position of .123456789012345. In effect, several decimal digits of each term, save the last, are lost. Of course, the number of digits lost depends on the difference in magnitude between λ_1 and λ_n. If λ_1/λ_n is on the order of 10^τ, so λ_1 and λ_n differ by τ orders of magnitude, then the inferior terms of the sum lose τ digits of accuracy. If τ is large enough, the contributions can be completely lost from every term except the last.

Does this loss of accuracy really matter? Isn't the value of $y = (A^TA)^{-1}x$ still correct to the accuracy of the computation? If y is really all we care about, yes. But y is supposed to be an approximate solution of $A^TAy = x$, and we should judge the adequacy of this approximation by looking at the difference between A^TAy and x. In exact arithmetic, the first $n - 1$ terms c_i/λ_i, which were nearly negligible in the computation of y, regain their significance and make an essential contribution to the restoration of x. But the digits that were lost in the limited precision arithmetic cannot be recovered, so every component, except the last, is either corrupted or lost entirely. Thus, while $y = (A^TA)^{-1}x$ might be correct to the limits of computational precision, the computed value of A^TAy can be incorrect in *every* decimal place.

In the context of the normal equations, the computation of $x = (A^TA)^{-1}A^Tb$ is supposed to result in a product Ax close to b. This calculation is subject to the same errors described above. Small errors in the computation of $(A^TA)^{-1}$, inflated by the final multiplication by A, can severely degrade the accuracy of the final result.

Based on the foregoing, it is the range of the magnitudes of the eigenvalues of A^TA that determines the effects of limited precision in the computation of the least squares vector x, and these effects will be most severe when the ratio λ_1/λ_n is large. As many readers will have recognized, this ratio is the *condition number* of the matrix A^TA. More generally, for any matrix, the condition number is the ratio of greatest to least singular values. For a square matrix A, the condition number can be interpreted as the reciprocal of the distance to the nearest singular matrix [**8**, p. 26]. A large condition number for a matrix is a sign that numerical instability may plague many types of calculations, particularly the solution of linear systems.

Now all of this discussion applies equally to the SVD solution of the least squares problem. There too we have to invert a matrix, multiplying U^Tb by Σ^+. Indeed, the singular values are explicitly inverted, and we can see that a component in the direction of the smallest positive singular value gets inflated, in the nonsingular case by the factor $1/\sigma_n$. Arguing as before, the effects of truncation are felt when the condition number σ_1/σ_n is large. So the benefit of the SVD solution is not that it avoids the effects of near dependence of the columns of A. But let us compare the two condition numbers. We know that the eigenvalues of A^TA are the squares of the singular values of A. Thus $\lambda_1/\lambda_n = (\sigma_1/\sigma_n)^2$, that is, *the condition number of A^TA is the square of the condition number of A.* In view of our heuristic error analysis, this relation between the condition numbers explains a rule of thumb for numerical computation: When computing with A^TA you need roughly twice as many digits to be as accurate as when you compute with the SVD of A.

The heuristic analysis is plausible, but by itself not completely convincing. As further support, I offer the following numerical example. Of course, there is no substitute for a careful mathematical analysis, and the interested reader is encouraged to consult [**8**] for a discussion of condition number.

A numerical example. We will simulate a least squares analysis where experimental data have been gathered for four variables and we are attempting to estimate the *dependent* variable by a linear combination of the other three *independent* variables. For simplicity, suppose each variable is specified by just four data values, so the columns a_1, a_2, a_3 of A, as well as the dependent column b, lie in $\mathbb{R}^4$. We would like the columns of A to be nearly dependent, so that we can observe the situation described in the heuristic analysis. Thus we will make one column, say a_3, differ from a linear combination of the other two by a very small random vector. Similarly, we want the dependent vector b to be near, but not in, the range of A, so we will define it, too, as a linear combination of a_1 and a_2 plus a small random vector. We then compute the least squares coefficients $x = (x_1, x_2, x_3)$ by the two formulas $x = V\Sigma^+ U^T b$ and $x = (A^T A)^{-1} A^T b$, and we compare the errors by calculating the magnitude of the residual $|b - Ax|$ in each case.

In choosing the random vectors in a_3 and b, we have different goals in mind. The random component of b will determine how far b is from the range of A, hence how large a residual the least squares solution should have. All our computations will be performed with 16 decimal place accuracy, so we will take the random component of b to be fairly small, on the order of 10^{-4}. In contrast, the goal in choosing the random component of a_3 is to make the condition number of A on the order of 10^8. Then the condition number of $A^T A$ will be on the order of 10^{16}, large enough to corrupt all 16 decimal digits of the solution produced by the normal equations. As will be shown in our later discussion of reduced rank approximations, the smallest singular value of A will be of about the same magnitude as the norm of the random vector used to define a_3. If the norms of a_1 and a_2 are on the order of 10, then the largest singular value will be of this order as well. Thus, choosing the random component of a_3 with norm on the order of 10^{-7} should produce a condition number of about 10^8, as desired.

It is worth noting that the near dependence between the columns of A should not degrade the quality of the least squares solution. Using just the first two columns of A, we can approximate b to within an error of about 10^{-4}. The third column of A can only improve the approximation. So, the true least squares solution should be expected to produce a residual no greater than 10^{-4}. The large error obtained using the normal equations is therefore attributable entirely to the numerical instability of the algorithm, not to any intrinsic limitations of the least squares problem.

All of the computations in this example were performed using the computer software package *MATLAB* [**18**]. Its simple procedures for defining and operating on matrices make this kind of exploration almost effortless. To help readers explore this topic further by using *MATLAB* to recreate and modify this example, I include the *MATLAB* commands. The actual program displays the results of each computation on the computer screen, but, to save space, I will show only the final results.

First, define two data vectors:

```
c1 = [1 2 4 8]'
c2 = [3 6 9 12]'
```

(The primes indicate the transpose operator, so that **c1** and **c2** are defined as column vectors.) The third data vector is a combination of these, plus a very small random vector:

```
c3 = c1 - 4*c2 + .0000001*(rand(4,1) - .5*[1 1 1 1]')
```

and the matrix A is defined to have these three vectors as its columns:

```
A = [c1 c2 c3]
```

In defining **c3** the command **rand(4,1)** returns a four-entry column vector with entries randomly chosen between 0 and 1. Subtracting 0.5 from each entry shifts them to between $-\frac{1}{2}$ and $\frac{1}{2}$. Thus, the length of the random component of **c3** will be at most 10^{-7}, as stipulated above.

Next, we define the b vector in a similar way, by adding a small random vector to a specified linear combination of columns of A:

```
b = 2*c1 - 7*c2 + .0001*(rand(4,1) - .5*[1 1 1 1]')
```

This time the random vector will be no longer than 10^{-4}, again as desired.

The SVD of A is quickly determined by *MATLAB*:

```
[U,S,V] = svd(A)
```

The three matrices **U**, **S** (which represents Σ), and **V** are displayed on the screen and kept in the computer memory. When we ran the program, the singular values turned out to be 59.810, 2.5976, and 1.0578×10^{-8}. Thus the condition number of A was about $6 \cdot 10^9$, the desired magnitude.

To compute the matrix Σ^+ we need to transpose the diagonal matrix **S** and invert the nonzero diagonal entries. This matrix, denoted by **G**, is defined by the following *MATLAB* commands:

```
G = S'
G(1,1) = 1/S(1,1)
G(2,2) = 1/S(2,2)
G(3,3) = 1/S(3,3)
```

Now let's see how well the SVD solves the least squares problem. We multiply $x = V\Sigma^+U^Tb$ by the matrix A and see how far the result is from b. Upon receiving the commands

```
r1 = b - A*V*G*U'*b
e1 = sqrt(r1'*r1)
```

MATLAB responds

```
e1 = 4.2681e-05
```

As desired, the computed magnitude of the residual $|b - Ax|$ is a bit smaller than 10^{-4}. So the SVD provided a satisfactory solution of our least squares problem.

The solution provided by the normal equations is $x = (A^TA)^{-1}A^Tb$. We enter

```
r2 = b - A*inv(A'*A)*A'*b
e2 = sqrt(r2'*r2)
```

and *MATLAB* responds

```
e2 = 51.9255
```

which is of the same order of magnitude as $|b|$, the distance from b to the origin! As we anticipated from our heuristic analysis, the solution to the least squares problem computed using the normal equations does a poor job of approximating b as a linear combination of the columns of A. What is more, the computed residual for this method is of no value as an estimate of the distance of b from the range of A.

This completes our in-depth look at least squares problems. The next section examines another area of application: reduced rank approximation.

Data compression using reduced rank approximations. The expression of the SVD in terms of the outer product expansion was introduced earlier. This representation emerges in a natural way in another application of the SVD: *data compression,* where we begin with an $m \times n$ matrix A of numerical data and try to describe a close approximation to A using far fewer numbers than the mn original entries. The matrix is not considered as a linear transformation, or indeed as an algebraic object at all. It is simply a table of mn numbers, and we want to find an approximation that captures the most significant features of the data.

Because the rank of a matrix specifies the number of linearly independent columns (or rows), it is a measure of redundancy. A matrix of low rank has a large amount of redundancy, so can be expressed much more efficiently than simply by listing all the entries. As a graphic example, suppose a scanner is used to digitize a photograph, replacing the image by an $m \times n$ matrix of pixels, each assigned a gray level on a scale of 0 to 1. Any large-scale features in the image are reflected by redundancy in the columns or rows of pixels, thus we may hope to recapture these features in an approximation by a matrix of lower rank than $\min(m, n)$.

The extreme case is a matrix of rank one. If B is such a matrix, then the columns are all multiples of one another—the column space is one-dimensional. If u is the single element of a basis, then each column is a multiple of u. We represent the coefficients as v_i, meaning that the ith column of B is given by $v_i u$, so that $B = [v_1 u \;\; v_2 u \;\; \ldots \;\; v_n u] = uv^T$. Thus, any rank one matrix can be expressed as an outer product, that is, as the product of a column and row. The mn entries of the matrix are determined by the m entries of the column and the n entries of the row. For this reason, we can achieve a large compression of the data matrix A if it can be approximated by a rank one matrix. Instead of the mn entries of A, we need only $m + n$ numbers to represent this rank one approximation to A. It is natural to seek the *best* rank one approximation.

We will call B the best rank one approximation to A if the error matrix $B - A$ has minimal norm, where now we define the (*Frobenius*) norm $|X|$ of a matrix X to be simply the square root of the sum of the squares of its entries. This norm is just the Euclidean norm of the matrix considered as a vector in $\mathbb{R}^{mn}$. Thinking of matrices this way, we define the inner product of two matrices by $X \cdot Y = \sum_{ij} x_{ij} y_{ij}$ and, as usual, $|X|^2 = X \cdot X$. Evidently this inner product of matrices can be thought of in three ways: as the sum of the inner products of corresponding rows, as the sum of the inner products of corresponding columns, or as the sum of the mn products of corresponding entries.

There is a simple expression for the norm of a matrix product XY that is easily derived using the outer product expansion. First, note that for rank one matrices xy^T and uv^T,

$$\begin{aligned} xy^T \cdot uv^T &= [xy_1 \;\; \ldots \;\; xy_n] \cdot [uv_1 \;\; \ldots \;\; uv_n] \\ &= \sum_i xy_i \cdot uv_i = \sum_i (x \cdot u) y_i v_i \\ &= (x \cdot u)(y \cdot v) \end{aligned} \tag{3}$$

where we have computed the matrix inner product as the sum of vector inner products of corresponding columns. In particular, xy^T and uv^T will be orthogonal with respect to the matrix inner product provided that either x and u or y and v are orthogonal as vectors. Using equation (3) and the outer product expansion $XY = \sum_i x_i y_i^T$

(with the x_i being the columns of X and the y_i^T the rows of Y) gives

$$XY \cdot XY = \left(\sum_i x_i y_i^T\right) \cdot \left(\sum_j x_j y_j^T\right) = \sum_{ij} (x_i \cdot x_j)(y_i \cdot y_j)$$

or

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

As a special case, if the x_i are orthogonal, then

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

Furthermore, if the x_i are both orthogonal and of unit length,

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

Similar conclusions apply if the y_i are orthogonal, or orthonormal. This shows that the norm of a matrix is unaffected by multiplying on either the left or right by an orthogonal matrix. Applying these results to the outer product expansion for the SVD, observe that in $A = \sum \sigma_i u_i v_i^T$, the terms are orthogonal with respect to the matrix inner product. This shows that $|A|^2 = \sum |\sigma_i u_i v_i^T|^2$, so that the SVD can be viewed as an orthogonal decomposition of A into rank one matrices.[2] More generally, if we partition the sum, taking S_r as the sum of the first r terms and E_r as the sum of the remaining terms, then $|A|^2 = |S_r|^2 + |E_r|^2$.

Now we are in a position to show a connection between rank one approximation and the SVD. Specifically, we will show that the best rank one approximation to A must be of the form $\sigma_1 u_1 v_1^T$. Clearly, $\sigma_1 u_1 v_1^T$ is one possible rank one approximation, and it gives rise to an error of $\sigma_2^2 + \cdots + \sigma_k^2$. The idea is to show that this is the smallest the error can be. For any rank one A_1, because the Frobenius norm is preserved under multiplication by orthogonal matrices, $|A - A_1| = |U\Sigma V^T - A_1| = |\Sigma - U^T A_1 V|$, where $A = U\Sigma V^T$ expresses the SVD of A. Write $U^T A_1 V$ as $\alpha x y^T$ with positive α and unit vectors $x \in \mathbb{R}^m$ and $y \in \mathbb{R}^n$. Next use the properties of the matrix inner product:

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

Focusing on the final term, we find that the matrix inner product of the diagonal matrix Σ with the outer product xy^T is given by

$$\Sigma \cdot xy^T = \sum_{i=1}^k \sigma_i x_i y_i \leq \sum_{i=1}^k \sigma_i |x_i| \, |y_i| \leq \sigma_1 \sum_{i=1}^k |x_i| \, |y_i| = \sigma_1 x^* \cdot y^*,$$

where $x^* = (|x_1|, \ldots, |x_k|)$ and $y^* = (|y_1|, \ldots, |y_k|)$. By the Cauchy–Schwarz inequality, $x^* \cdot y^* \leq |x^*| \, |y^*| \leq |x| \, |y| = 1$. Combined with the last equation above, this gives $\Sigma \cdot xy^T \leq \sigma_1$.

[2] This also shows that $|A|^2 = \sum \sigma_i^2$. Interestingly, both the operator norm $\|A\| = \sigma_1$ and the Frobenius norm $|A|$ are simply expressed in terms of singular values.

Returning to the main thread of the argument, we now have $|\Sigma - \alpha xy^T|^2 \geq |\Sigma|^2 + \alpha^2 - 2\alpha\sigma_1 = |\Sigma|^2 + (\alpha - \sigma_1)^2 - \sigma_1^2$. We immediately see that the right-hand side is minimized when α is taken to be σ_1, so that $|\Sigma - \alpha xy^T|^2 \geq |\Sigma|^2 - \sigma_1^2$. Moreover, if the σ_i are distinct, this minimum is actually obtained only when $\alpha = \sigma_1$ and x and y are of the form $e_1 = (1, 0, \ldots, 0)$ in $\mathbb{R}^m$ and $\mathbb{R}^n$, respectively. Finally, $A_1 = \alpha Uxy^T V^T = \sigma_1(Ue_1)(Ve_1)^T = \sigma_1 u_1 v_1^T$. That is what we wished to show. (If σ_1 is a repeated singular value, a slight modification of the argument is needed. It amounts to making an orthogonal change of basis in the subspace of right singular vectors corresponding to σ_1.)

This result shows that the SVD can be used to find the best rank one approximation to a matrix. But in many cases, the approximation will be too crude for practical use in data compression. The obvious next step is to look for good rank r approximations for $r > 1$. In this regard, there is an attractive *greedy* algorithm. To begin, choose a rank one A_1 for which the error $E_1 = A - A_1$ has minimal norm. Next choose a rank one matrix A_2 for which the norm of $E_2 = E_1 - A_2$ is minimal. Then $A_1 + A_2$ is a rank two approximation to A with error E_2. Continue in this fashion, each time choosing a best possible rank one approximation to the error remaining from the previous step. The procedure is a greedy algorithm because at each step we attempt to capture as large a piece of A as possible. After r steps, the sum $A_1 + \cdots + A_r$ is a rank r approximation to A. The process can be repeated for k steps (where k is the rank of A), at which point the error is reduced to zero. This results in the decomposition

$$A = \sum_i A_i,$$

which is none other than the SVD of A. To be more precise, each A_i can be expressed as the product of a positive scalar σ_i with an outer product $u_i v_i^T$ of unit vectors. Then A is given by

$$A = \sum_i \sigma_i u_i v_i^T,$$

which, true to the notation, is the SVD of A expressed in the form of the outer product expansion.

This statement follows from the earlier result linking the best rank one approximation of a matrix to its largest singular value and corresponding singular vectors. Assuming the singular values are distinct, write $A = \sum_{i=1}^k \sigma_i u_i v_i^T$. It is clear that A_1 must equal $\sigma_1 u_1 v_1^T$ and $E_1 = \sum_{i=2}^k \sigma_i u_i v_i^T$. But that gives the SVD of E_1, from which we can instantly obtain the best rank one approximation $A_2 = \sigma_2 u_2 v_2^T$. Clearly, this argument can be repeated until the complete decomposition of A is obtained. (As before, if the singular values of A are not distinct, the argument must be modified slightly to take into account orthogonal changes of basis in the subspace of right singular vectors corresponding to a particular singular value.)

The understanding we now have of the connection between successive best rank one estimates and the SVD can be summarized as follows. The outer product expansion form for the SVD, $A = \sum \sigma_i u_i v_i^T$, expresses A as a sum of rank one matrices that are orthogonal with respect to the matrix inner product. Truncating the sum at r terms defines a rank r matrix $S_r = \sum_{i=1}^r \sigma_i u_i v_i^T$. Approximating A with S_r leaves an error of $E_r = A - S_r = \sum_{i=r+1}^k \sigma_i u_i v_i^T$ with $|E_r|^2 = \sum_{i=r+1}^k \sigma_i^2$ and $|A|^2 = |S_r|^2 + |E_r|^2$. The sum S_r is the result of making successive best rank one approximations. But there is a stronger result: S_r is actually the best rank r approximation possible. Proofs of this assertion can be found in Lawson and Hanson [**14**, pp. 23–26] and Leon [**16**].

The properties of reduced rank approximations can now be used to clarify a remark made earlier. In the example showing the advantage of the SVD in least squares problems, the matrix A had three independent columns, but the final column differed from a linear combination of the other two by a small random vector. Subtracting that small random vector from the third column produces a rank two matrix B that closely approximates A. In fact, $|A - B|$ is equal to the norm of the random vector, expected to be about 5×10^{-8}. Of course, the minimal error for a rank two approximation to A must be precisely σ_3, so we can conclude that $\sigma_3 \le |A - B| \approx 5 \times 10^{-8}$. This shows that the magnitude of the random vector provides a bound on the least singular value of A. In particular, the order of magnitude of the least singular value can be controlled by choosing the random vector appropriately. This is what allowed us to construct A with a prescribed condition number in the least squares example.

In practice, the SVD can be used to select the rank r and find the best rank r approximation to A. Note that with r terms, the SVD outer product expansion results in a relative error

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

Typically, the value of r is chosen to reduce this relative error to some specified threshold. There is a nice visual example in [**17**] of this idea, used to approximate an image of a surface in $\mathbb{R}^3$. Image processing is also discussed in [**2**] and [**12**], the latter including an interesting illustration involving fingerprints. A related discussion of the use of the SVD in cryptographic analysis appears in [**19**]. For a completely different application of reduced rank approximations, see [**6**] which employs the best rank 2 approximation of a matrix in connection with a data visualization technique.

We conclude this section with a detailed example of the SVD and reduced rank approximations. This example was developed by K. Kirby [**13**], based on a discussion in [**20**].

The image shown in Figure 6 (page 20) represents a 24×24 matrix A whose entries are all either 0 or 1. The image is created using a rectangular grid of the same dimensions, with each cell in the grid colored black (if the corresponding matrix entry is 0) or white (if the entry is 1). Here are the first 16 singular values for this matrix shown to four decimal places:

9.5403	6.6288	5.6369	3.4756	2.7385	2.2023	1.5835	1.5566
1.4207	1.2006	0.9905	0.9258	0.7479	0.6744	0.6122	0.4698

The remaining singular values were computed as zero to four decimal places. Now suppose we adopt an accuracy threshold of 90%. That would mean we wish to choose a reduced rank approximation with error no more than 10% of $|A|$. Define

$$e(r) = 1 - \sqrt{\frac{\sum_1^r \sigma_i^2}{\sum_1^{16} \sigma_i^2}}.$$

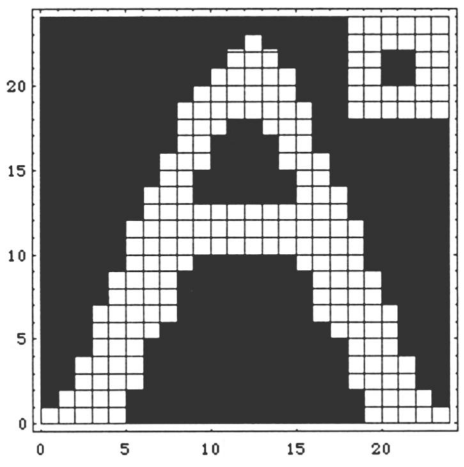

Figure 6. A 24 × 24 image.

This gives the relative error for a sum of the first r terms of the SVD outer product expansion. Computing $e(2) = 0.18$ and $e(3) = 0.09$, we see that three terms of the expansion are needed to achieve an error of 10% or less. The result of using just three terms of the series is displayed as an image in Figure 7. Here the numerical entries of the matrix are displayed as gray levels. Similarly, setting a threshold of 95% would lead us to use a rank 5 approximation to the original matrix. That produces the image shown in Figure 8, in which one can recognize the main features of the original image.

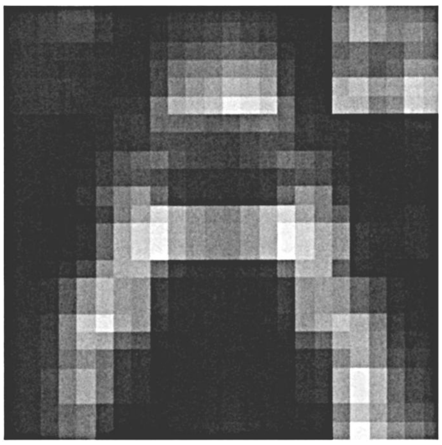

Figure 7. Rank 3 approximation.

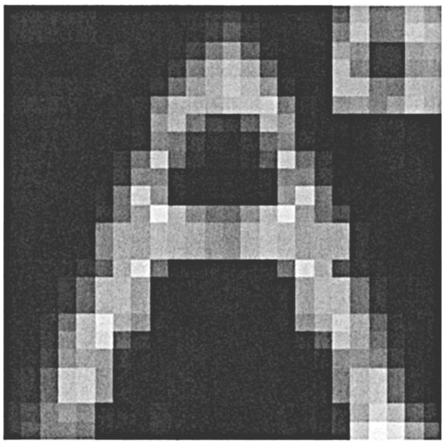

Figure 8. Rank 5 approximation.

In fact, simply rounding the entries of the rank 5 approximation to the nearest integer restores the original image almost perfectly, as shown in Figure 9. Observe in this regard that the error matrix E_5 has $|E_5|^2 = \sum_{i>5} \sigma_i^2 = 16.7$. Thus, the average value of the squares of the entries of E_5 is $16.7/24^2 = 0.029$ and we might estimate the entries themselves to be around 0.17. Since this value is well below 0.5, it is not surprising that Figure 9 is nearly identical to the original image.

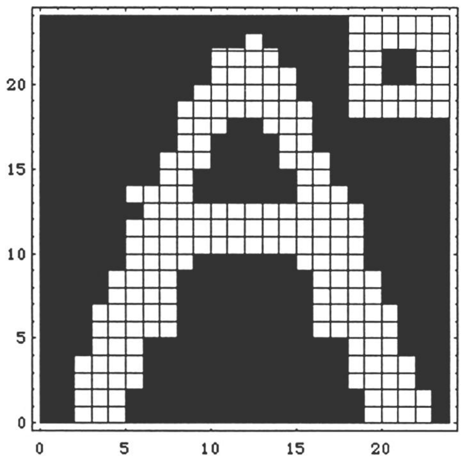

Figure 9. Rank 5 approximation, rounded to integers.

There is an intriguing analogy between reduced rank approximations and Fourier analysis. Particularly in the discrete case, Fourier analysis can be viewed as representing a data vector relative to a special orthogonal basis. The basis elements are envisioned as pure vibrations, that is, sine and cosine functions, at different frequencies. The Fourier decomposition thus represents the input data as a superposition of pure vibrations with the coefficients specifying the amplitude of each constituent frequency. Often, there are a few principal frequencies that account for most of the variability in the original data. The remaining frequencies can be discarded with little effect. The reduced rank approximations based on the SVD are very similar in intent. However, the SVD captures the best possible basis vectors for the particular data observed, rather than using one standard basis for all cases. For this reason, SVD-based reduced rank approximation can be thought of as an adaptive generalization of Fourier analysis. The most significant *vibrations* are adapted to the particular data that appear.

A Computational Algorithm for the SVD

The applicability of the SVD is a consequence of its theoretical properties. In practical applications, the software that calculates the SVD is treated as a black box: We are satisfied that the results are accurate and content to use them without worrying about how they were derived. However, peek into the box and you will be rewarded with a glimpse of an elegant and powerful idea: *implicit matrix algorithms*. The basic idea behind one of the standard algorithms for computing the SVD of A depends on the close connection to the EVD of A^TA. As the algorithm proceeds, it generates a sequence of approximations $A_i = U_i\Sigma_i V_i^T$ to the correct SVD of A. The validity of the SVD algorithm can be established by showing that after each iteration, the product $A_i^T A_i$ is just what would have been produced by the corresponding iteration of a well-known algorithm for the EVD of A^TA. Thus, the convergence properties for the SVD algorithm are inferred from those of the EVD algorithm, although A^TA is never computed and the EVD algorithm is never performed. From this perspective, the SVD algorithm can be viewed as an implicit algorithm for the EVD of A^TA. It provides all the information needed to construct the EVD by operating directly on

A. Indeed, the operations on A are seen to implicitly perform the EVD algorithm for A^TA, without ever explicitly forming A^TA.

Implicit algorithms are a topic of current research interest [**10**]. References [**1**] and [**4**] describe some implicit algorithms for computations other than the SVD and suggest sources for further reading. A detailed account of the SVD algorithm is found in [**8**, Sec. 8.3], where additional citations and notes about the history of the algorithm are also given. In [**7**] there is a more compact (though less general) development that makes the connection to the QR algorithm more direct. It should also be noted that there are alternatives to the algorithm described above. One alternative that has significance in some kinds of parallel processing is due to Jacobi; see [**8**, Sec. 8.4]. Another interesting alternative uses a rank 1 modification to split an SVD problem into two problems of lower dimension, the results of which can be used to find the SVD of the original problem. This method is described in [**11**].

Conclusion

The primary goal of this paper is to bring the SVD to the attention of a broad audience. The theoretical properties have been described, and close connections were revealed between the SVD and standard topics in the first linear algebra course. Several applications of the SVD were mentioned, with a detailed discussion of two: least squares problems and reduced rank estimation. The computational algorithm for the SVD was also briefly mentioned.

Emphasizing the main themes of the subject has unfortunately meant omitting interesting details and limiting my presentation to general ideas. The reader is encouraged to consult the references for a more thorough treatment of the many aspects of this singularly valuable decomposition.

Acknowledgments. I thank Professor Kevin Kirby for *Mathematica* files used in the example of reduced rank approximation.

References

1. Gary E. Adams, Adam W. Bojanczyk, and Franklin T. Luk, Computing the PSVD of two 2×2 triangular matrices, *SIAM Journal of Matrix Analysis Applications* 15:2 (April 1994) 366–382.
2. Harry C. Andrews and Claude L. Patterson, Outer product expansions and their uses in digital image processing, *American Mathematical Monthly* 82:1 (January 1975) 1–13.
3. S. J. Blank, Nishan Krikorian, and David Spring, A geometrically inspired proof of the singular value decomposition, *American Mathematical Monthly* 96:3 (March 1989) 238–239.
4. Adam Bojanczyk and Paul Van Dooren, On propagating orthogonal transformations in a product of 2×2 triangular matrices, in L. Reichel, A. Ruttan, and R. S. Varga, eds., *Numerical Linear Algebra* (Proceedings of the Conference in Numerical Linear Algebra and Scientific Computation, Kent, OH, March 13–14, 1992), W. de Gruyter, New York, 1993, pp. 1–9.
5. Randall E. Cline, *Elements of the Theory of Generalized Inverses for Matrices,* Education Development Center, Newton, MA, 1979.
6. K. R. Gabriel, The biplot graphic display of matrices with application to principal components analysis, *Biometrika* 58:3 (1971) 453–467.
7. Gene H. Golub and C. Reinsch, Singular value decomposition and least squares solutions, *Numerical Mathematics* 14 (1970) 403–420.
8. Gene H. Golub and Charles F. Van Loan, *Matrix Computations,* Johns Hopkins University Press, Baltimore, MD, 1983.

9. I. J. Good, Some applications of the singular decomposition of a matrix, *Technometrics* 11:4 (Nov. 1969) '823–831.
10. Implicit Matrix Computations. Mini Symposium 36, SIAM 40th anniversary meeting, July 20–24, 1992, Los Angeles.
11. E. R. Jessup and D. C. Sorensen, A parallel algorithm for computing the singular value decomposition of a matrix, *SIAM Journal of Matrix Analysis and Applications* 15 (1994) 530–548.
12. David Kahaner, Cleve Moler, and Stephen Nash, *Numerical Methods and Software,* Prentice Hall, Englewood Cliffs, NJ, 1988, pp. 222–224.
13. Kevin Kirby, private communication, Northern Kentucky University, Highland Heights, KY 41076.
14. Charles L. Lawson and Richard J. Hanson, *Solving Least Squares Problems,* Prentice Hall, Englewood Cliffs, NJ, 1974.
15. David C. Lay, *Linear Algebra and Its Applications,* Addison-Wesley, Reading, MA, 1994, chapter 8.
16. Stephen J. Leon, *Linear Algebra with Applications,* 4th ed., Macmillan, New York, 1994, pp. 422–424.
17. Cliff Long, Visualization of matrix singular value decomposition, *Mathematics Magazine* 56:3 (May 1983) 161–167.
18. *MATLAB,* student ed., Prentice Hall, Englewood Cliffs, NJ, 1992.
19. Cleve Moler and Donald Morrison, Singular value analysis of cryptograms, *American Mathematical Monthly* 90:2 (February 1983) 78–86.
20. Ben Noble and James W. Daniel, *Applied Linear Algebra,* 2nd ed., Prentice Hall, Englewood Cliffs, NJ, 1977, pp. 343–345.
21. John A. Richards, *Remote Sensing Digital Image Analysis,* Springer-Verlag, New York, 1993, chapter 6.
22. Gilbert Strang, *Linear Algebra and Its Applications,* 2nd ed., Academic Press, New York, 1980.
23. ———, The fundamental theorem of linear algebra, *American Mathematical Monthly* 100:9 (November 1993) 848–859.

A Shortcut

To the Editor: In H. Krishnapriyan's article [*CMJ* 26:2 (March 1995) 118–123], the proof of the main theorem uses Eulerian polynomials to prove that

$$S_k(-n-1)=(-1)^{k+1}S_k(n) \qquad \text{(A)}$$

where S_k is the polynomial of degree $k+1$ such that $S_k(n)=1^k+2^k+\cdots+n^k$ for positive integers n. Here is a short proof of (A) without Eulerian polynomials: By replacing n by $-n$ in the equation $S_k(n-1)=S_k(n)-n^k$, we get

$$S_k(-n-1)=S_k(-n)+(-1)^{k+1}n^k. \qquad \text{(B)}$$

Equation (A) easily follows from (B) by induction on n, starting with $n=0$; the details are easy to fill in.

—David M. Bloom, Brooklyn College of CUNY

Editor's note. Krishnapriyan suggests that our readers may find interesting a recent paper by Donald Knuth, "Johann Faulhaber and Sums of Powers" [*Mathematics of Computation* 61:203 (July 1993) 277–294].