From *SIAM News*, Volume 42, Number 4, May 2009

Blockbuster Algorithm

By Barry A. Cipra

Nineteen sixty-five was an extraordinary year for three-letter acronyms in numerical analysis. James Cooley and John Tukey published what would come to be called the FFT. That same year, Gene Golub and William Kahan introduced an algorithm for computing something known as the SVD. Decades later, both algorithms are mainstays of large-scale scientific—and commercial—computation. Indeed, without them, computers might only now be finishing calculations started when Lyndon Johnson was launching the Great Society.

The FFT, of course, is the famously versatile fast Fourier transform, beloved of physicists, computational engineers, and signal processors of all stripes. Its name describes what it does: It computes Fourier transforms (more precisely, discrete Fourier transforms), and it does so quickly. (To be sure, the FFT was later discovered to be a *re*-discovery of an algorithm first found by Gauss.) Roughly speaking, it replaced a straightforward but lethargic $O(N^2)$ algorithm with a subtle but sleek $O(N \log N)$ approach, thereby saving orders of magnitude in computational effort with increasing N.

The SVD is similarly beloved—and even more versatile. SVD stands for singular value decomposition and is also often taken to refer to the algorithm outlined by Golub and Kahan. In essence, the SVD is an all-purpose, blue-collar, nuts-and-bolts method for revealing information trapped inside large matrices of arbitrary size. As such, it is the key step in solving a vast array of problems in linear algebra. Dianne O'Leary, an SVD expert at the University of Maryland, calls it "the Swiss Army knife of matrix computation."

Cigars and Pancakes

Linear algebra fairly bristles with matrix decompositions, also known as factorizations. Much like factorizations of composite numbers into primes, or of polynomials into linear and quadratic terms, matrix decompositions express general matrices as products of matrices that are somehow "atomic." The higher dimensionality of matrix theory, however, makes it possible to do this in many ways: There are LU decompositions, QR decompositions, eigendecompositions, Cholesky, Schur, and Jordan decompositions, and endless others, including the SVD. Each has its own requirements and adherents. Some are highly specialized—to be eligible for, say, Cholesky decomposition, a matrix must be square, symmetric, and positive definite. Others, including the SVD, are far more general.

The singular value decomposition takes any $m \times n$ matrix A and rewrites it as the product of an $m \times m$ unitary matrix U, an $n \times n$ unitary matrix V, and an $m \times n$ diagonal matrix S:

$$A = USV^*.$$

Most important, the entries down the diagonal of S are non-negative real numbers, traditionally arranged in decreasing order. These are the so-called singular values.

Geometrically, the singular value decomposition describes the unit sphere in R^n taking on an ellipsoidal appearance in R^m under the transformation A. A single large singular value and a slew of smaller ones give the ellipsoid a cigar-like look, two comparable large values and you've got a pancake, and so forth. The unitary matrix U essentially identifies the orientation of the ellipsoid in R^m, while V^* picks out the axes in R^n along which the singular-value dilations and contractions of the sphere take place.

The geometric interpretation makes it more or less obvious that the singular value decomposition is well defined, with a unique diagonal matrix S. Singular values, in general, are *not* eigenvalues. For one thing, eigenvalues don't exist for non-square matrices. But even when A is square, the connection is tenuous at best; only when A is Hermitian and positive semi-definite are the singular values and eigenvalues the same.

The squares of the singular values, on the other hand, *are* eigenvalues—not of A itself, but of A^*A. Indeed, statisticians have long made use of this for singular value decomposition by another name: principal component analysis. In PCA, each row of the matrix A corresponds to data from a single trial in an experiment, such as the age, weight, height, IQ, testosterone level, and batting average of a bunch of baseball players. More precisely, the entries in a column are the departures from the average for that column (so that each column of A sums to 0). The aim is to see what correlates most strongly with what. The columns of the $n \times n$ matrix V are the "principal components." The SVD–PCA connection stems from the straightforward linear algebra calculation

$$A^*A = (VS^*U^*)(USV^*) = VS^2V^*$$

(S being a diagonal matrix with real entries, and $U^*U = I$).

The number of nonzero singular values reveals the underlying dimensionality of the data set, and their relative sizes hint at the spread of the data cloud in the directions of the various principal components. In particular, the principal component corresponding to the largest singular value gives the direction for a linear regression.

Using A^*A to get at the principal components makes tremendous sense, especially when n is fairly small. But by squaring the singular values, the eigen-approach exaggerates the ratio of largest to smallest, to the point that important information can be lost in the numerical clutter: A poorly designed principal component algorithm could mistake a pancake for a crepe.

The singular value decomposition is a more robust way to compute principal components. It also lends itself to a host of other applications, ranging from finding the vector **x** in R^n that minimizes $\|A\mathbf{x} - \mathbf{b}\|$ for a given vector **b** in R^m to finding basis vectors for the range and null space of A. "In many problems, once you've got the SVD, it makes the solution obvious," O'Leary notes. "If you google SVD, you come up with millions of hits." (More precisely, 6,100,000 on March 12, 2009. Some of those hits, however, are for a Russian sniper rifle, while others lead to the Societas Verbi Divini, or Society of the Divine Word. Specifying "singular value decomposition" takes the count down to 430,000—but misses a lot of sites that don't spell out the acronym.)

"Numerically Stable and Fairly Fast"

Until Golub and Kahan showed how, numerical analysts had no accurate way to compute SVDs. Numerical instability seemed unavoidable. MathWorks founder and SIAM past-president Cleve Moler, who received his PhD in 1965, recalls that when he was a grad student, "we knew about the SVD, but only as a theoretical tool."

The Golub–Kahan breakthrough, described as "a numerically stable and fairly fast method" for computing the singular value decomposition, was published in 1965 in *SIAM Journal on Numerical Analysis* (Volume 2, Number 2, pages 205–224) under the title "Calculating the Singular Values and Pseudo-Inverse of a Matrix." The authors modestly pointed out in their abstract that the method "is complicated but does not suffer from the computational difficulties which occasionally afflict some previously known methods." The method has three key steps for computing the singular values. The first produces an $n \times n$ bidiagonal matrix J with the same singular values as A. The second takes the $2n \times 2n$ matrix

$$\begin{bmatrix} 0 & J \\ J^* & 0 \end{bmatrix}$$

and converts it to symmetric tridiagonal form (with zeroes down the main diagonal and the bidiagonal entries of J alternating on the sub- and super-diagonals). The eigenvalues of this matrix are the singular values of A and their negatives. The third step is simply to find the eigenvalues of this tridiagonal matrix—a step for which efficient algorithms were already known.

Overall, the algorithm that Golub and Kahan outlined runs on $m \times n$ matrices in $O(mn^2)$ time, with the convention $m \geq n$. The n^2 may appear worrisome, but it's the linearity in m that's crucial: In many applications m is a huge number, while n is merely large. The algorithm's main virtue, however, is accuracy. The classic A^*A approach of PCA also runs in $O(mn^2)$ time, but the results are all too often worthless, or at least suspect.

Coming Attractions

The Golub–Kahan algorithm, and a more fleshed-out version published in 1970 by Golub and Christian Reinsch, opened the floodgates for applications of the SVD. It is a commonly invoked subroutine in, for example, MATLAB, where it is simply written as svd(A). There are, of course, endless variants, specialized for different purposes.

One current application for which the SVD has, so far, been computational king of the hill is in a million-dollar challenge known as the Netflix Competition. Netflix is the DVD rental behemoth based in Los Gatos, California. With more than 10 million subscribers (who could range from a loner mathematician dropout living in a cabin in Montana to an extended family crammed into a one-bedroom apartment in the Bronx) picking from more than 100,000 titles, Netflix has a database to die for: Each subscriber is invited to rate movies that he or she or they have seen, using a "star" system, from 1 star (terrible) to 5 stars (terrific).

Netflix uses this database primarily to make recommendations to its subscribers, calculating movies they are likely to enjoy. Just about all e-tailers do much the same, sometimes to the annoyance of their customers. Amazon, of course, makes a profit every time you buy something it recommends. For Netflix, the goal is customer satisfaction and loyalty, leading to subscription-plan upgrades and word-of-mouth advertising. (How often have you heard friends *complain* about the service they get from Netflix?) One other goal is to increase circulation of movies in the "long tail" of filmdom: If Netflix can divert customers who would otherwise queue up for the latest releases, which also typically have higher distribution fees, to little-watched cinematic gems, everyone wins.

The general subject of making recommendations based on observed or self-reported behavior goes under the rubric "collaborative filtering." The process can be thought of as looking for doppelgängers. Suppose, for example, that Netflix spotted in its database someone who has seen many of the same movies you have and that within this overlap the two of you have exactly the same opinions. It's reasonable to assume that your tastes will coincide for other movies as well—if your counterpart gave 5 stars to, say, *The Matrix*, it's likely that you'd enjoy it too. (An anti-doppelgänger, who hates the movies you love, and vice versa, would be equally valuable.)

Such exact matches, of course, are rare to the point of being non-existent, except for people who have seen (or ranked) so few movies that they might have multiple doppelgängers who disagree among themselves on other movies. A far more sensible approach is to look for correlations in cinematic tastes—exactly the sort of thing the SVD is designed to do.

Netflix considers recommendations so vital that it has ponied up a million dollars in prize money for a 10% improvement in accuracy over its own proprietary algorithm, Cinematch. The company has made more than 100 million ratings of nearly 18,000 movies by more than 480,000 (anonymized) subscribers publicly available as a "training set," reserving another 3 million ratings as a test set (using digital signatures to vouch for the integrity of the data, which will be revealed when the competition ends in 2011). Contestants are told which subscriber-movie ratings they are to predict; the goal is to come as close as possible to the actual test-set values.

The 10% metric refers to a root mean squared error measurement. When Netflix ran Cinematch on the training set, its predictions for the test set had an RMSE of 0.9525. This turned out to be a 9.6% improvement over one of the most naive approaches possible: Compute each movie's average rating over the training set and use those numbers across the board in the test set, which gives an RMSE of 1.0540. In effect, the million-dollar challenge is to slightly more than double Cinematch's edge over mindless averaging. The official target is an RMSE of 0.8572 on the test set.

The Netflix Competition, announced in 2006, has attracted thousands of entries. Teams have come tantalizingly close to the target 10%, but so far (as of early April 2009) double-digit improvement has proved elusive. To maintain interest, Netflix awards an annual "progress prize": $50,000 for specified incremental advances. The 2007 prize went to the three-member team "BellKor": Bob Bell and Chris Volinsky, both of AT&T Labs, and Yehuda Koren, now at Yahoo! Research in Israel. Last year, BellKor joined forces with Andreas Töscher and Michael Jahrer of Commendo Research in Austria, entering the competition as "BellKor in BigChaos," to win the 2008 progress prize. The current leader is a two-member team from Montreal called PragmaticTheory, whose most recent entry weighed in with an RMSE of 0.8596—which is to say, a 9.65% improvement over Cinematch.

The SVD is only one aspect of the various contestants' algorithms—the progress prize papers describe many other components that have gone into the stew, from "Boltzmann machines" to *k*-nearest-neighbor models, not to mention variations on the SVD itself—but it appears to be indispensable. PragmaticTheory's chart of its progress, for example, shows the SVD to have been responsible for an early quick improvement (see Figure 1).

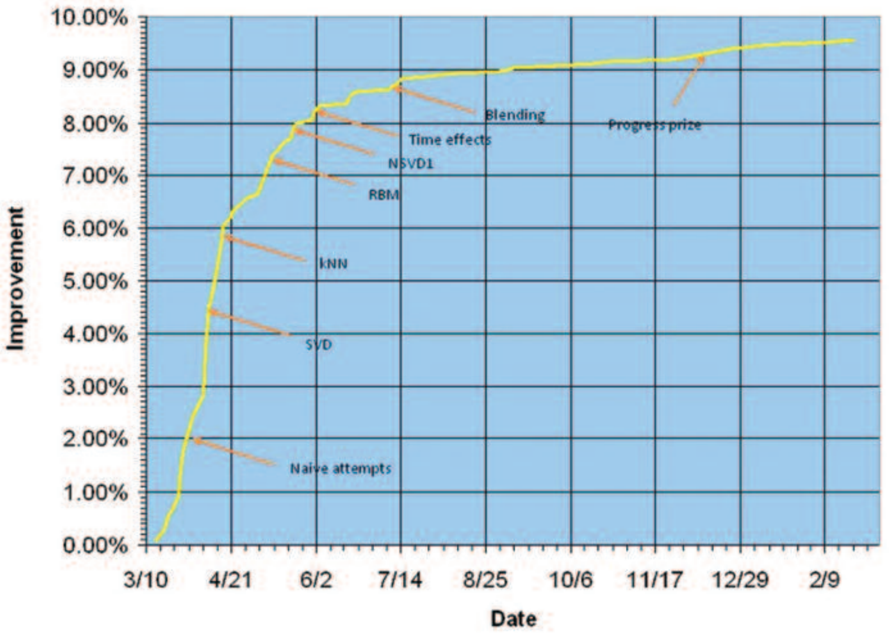

Figure 1. *The PragmaticTheory team, which currently has the lead in the Netflix Competition with a 9.65% improvement over Cinematch, has tracked its performance since entering the race in 2008. As the graph shows, the singular value decomposition was a key element in getting the team off to a fast start.*

"The SVD method was the first 'latent feature model' that we implemented for the Netflix contest," PragmaticTheory's Martin Chabbert says. (His fellow pragmatic theorist is Martin Piotte.) "Because it is pretty simple to understand and implement, and because its application to this problem is very well documented, it is often the first (serious) step that people start out with. The SVD is also interesting because it has a 'psychological' or 'human-level' explanation that can be fun to explore. In the end though, the SVD is more of a building block that helps understand some basic concepts before tackling (and inventing) more complex models and methods."

Barry A. Cipra is a mathematician and writer based in Northfield, Minnesota.