Theoretical Project in Data Mining: SVD Computation

[redacted]

Abstract

SVD conveys important geometrical insights about the matrix. It is a step in many algorithms and play a central role in data analysis and scientific computing. It has applications in many areas such as least squares problem , computing pseudoinverse , Jordan canonical form , solving integral equations , digital image processing , optimization, ets.. Golub and Van Loan [4] ascribe a central significance to the SVD in their definitive explication of numerical matrix methods stating:

....perhaps the most recurring theme in the book is the practical and theoretical value of the SVD ..

Many applications involve computing SVD of large matrices using limited computer resources, such as smartphones. Therefore, more than ever before, it is important to make the computaton as efficient as possible. Combination of theoretical chalenge and applicativity is my personal reason to choose this subgect for the project.

Definition

$A \in C^{m\times n}$ *a singular value decomposition* SVD of A is a factorization

$$A = U\Sigma V^*$$

where

- $U \in C^{m\times m}$ is unitary ,
- $V \in C^{n\times n}$ is unitary ,
- $\Sigma \in R^{m\times n}$ is unitary .

Σ is rectangular diagonal matrix and has the same shape as A. Its diagonal entries σ_i, called *singular values*, are real and nonnegative. We will assume $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0$, where $p = min\,(m,n)$. U consist of m *left singular vectors* $\{u_1, \ldots, u_m\}$ and V consist of n *right singular vectors* $\{v_1, \ldots, v_n\}$.

Matrix Properties Via the SVD

Let $r \leq p$ denote the number of nonzero singular values of A.

1. Every matrix has a singular value decomposition
2. The nonzero singular values of A are square roots of the nonzero eigenvalues of AA^* or A^*A
3. The rank of A is equal to the number of its nonzero singular values
$$rank(A) = r$$
4. $range(A) = span\{u_1, \ldots, u_r\}$ and $null(A) = span\{v_{r+1}, \ldots, v_n\}$
5. $\|A\|_2 = \sigma_1$
6. $\|A\|_F = \sqrt{\sigma_1^2 + \sigma_2^2 + \ldots + \sigma_r^2}$
7. If A is Hermitian then
$$\sigma_i(A) = |\lambda_i(A)|$$
8. For $A \in C^{m \times m}$
$$|det(A)| = \prod_{i=1}^{m} \sigma_i$$
9. A can be represented as a sum of rank one matrices
$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^*$$
10. For any k with $0 \leq k \leq r$ define *truncates SVD*
$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^*$$

If $k = p$, define $\sigma_{k+1} = 0$. Then

$$\|A - A_k\|_2 = \inf_{\substack{B \in C^{m \times n} \\ rank(B) \leq k}} \|A - B\|_2 = \sigma_{k+1}$$

Most of this properties can be easily proved and have computational consequences. The best method for determining the rank of a matrix is to count the number of singular values greater than a judiciously chosen tolerance. The most accurate method for finding an orthonormal basis of a range or nulspace is via SVD. Althouh QR factorization is faster, it is not always as accurate. Property 5 gives us a way to compute$\|.\|_2$ and property 9 provides us with a standard way for computing low rank approximation with respect to $\|.\|_2$.

SVD of A is Eigenvalues of A^*A

SVD is related to EVD of Hermitiam matrix: If $A = U\Sigma V^*$SVD decomposition of A, then $A^*A = V\Sigma^*\Sigma V^*$ EVD of A^*A. The eigenvalues of Hermitiam matrix are real and non-negative, thus it is diagonalizable. The matrix A^*A is known as *covariance matrix* of A, and has interpretations in statistics and other fields. Therefore, computation of SVD of arbitrary matrix can be reduced to the computation of the eigenvalue decomposition of Hermitian square matrix. Mathematically speaking, we might calculate the SVD of A as follows:

1. Form A^*A
2. Compute the eigenvalue decomposition of $A^*A = V\Lambda V^*$
3. Define Σ to be $m \times n$ nonnegative diagonal square root of Λ
4. Solve the system $U\Sigma = AV$ for unitary U , via QR factorization

Unfortunately, it is numerically unstable. This algorithm is not practical, because it reduces the SVD problem to eigenvalue problem, that may be much more sensitive to perturbations. From analyzing stability of the algorithms as appears in [1], we get that this algorithm performs well on dominant singular values of A, but for small singular values $\sigma \ll \|A\|_2$ we expect loss of accuracy.

Two Phase Algorithm

Since 1960s two phase approach has been standard for the SVD.The matrix is brought into bidiagonal form , and then the bidiagonal matrix is diagonalized:

$$\underset{A}{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}} \overrightarrow{Phase1} \underset{B}{\begin{bmatrix} \times & \times & & \\ & \times & \times & \\ & & \times & \times \\ & & & \times \\ & & & \end{bmatrix}} \overrightarrow{Phase2} \underset{\Sigma}{\begin{bmatrix} \times & & & \\ & \times & & \\ & & \times & \\ & & & \times \\ & & & \end{bmatrix}}$$

In standard algorithms for SVD no matrices of dimension as $m + n$ are formed explicitly. The key step is to make the process fast is an initial unitary reduction to bidiagonal form.

Phase 1 : Golub-Kahan Bidiagonalization

Also known as Golub-Reinsch algorithm GR-SVD. In *Phase1* we bring A into bidiagonal form by applying distinct unitary operations on the left and right. Finite Householder reflectors are applied alternately: n are applied on the left and $n - 2$ on the right. Define:

$$P^{(k)}, k = 1, 2, ..., n$$

and

$$Q^{(k)}, k = 1, 2, ..., n-2$$

such that

$$P^{(n)}...P^{(1)}AQ^{(1)}...Q^{(n-2)} = B$$

where B is upper bidiagonal matrix. $P^{(i)}$ zeros out the subdiagonal elements in column i and $Q^{(j)}$ zeros out the appropriate elements in row j, without destroying the zeros introduced before. Because all the transformations are unitary, the singular values of B are the same as those of A. Thus if:

$$B = G\Sigma H^*$$

then

$$A = PG\Sigma H^* Q^*$$

so that

$$U = PG, V = QH \tag{1}$$

with

$$P = P^{(1)}...P^{(n)}, Q = Q^{(1)}...Q^{(n-2)}$$

Total amount of work for Golub-Kahan bidiagonalization : $\sim 4mn^2 - \frac{4}{3}n^3$ flops.

Phase 1: Lawson-Hanson-Chan Bidiagonalization

For $m \gg n$ this operation count is unnecessary large.Alternative method for bidiagonalization first proposed by Lawson and Hanson and later developed by Chan [2]. The idea is illustrated as follows:

$$\underset{A}{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}} \rightarrow \underset{Q^*A}{\begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & & \times \\ & & & \end{bmatrix}} \rightarrow \underset{U^*Q^*AV}{\begin{bmatrix} \times & \times & & \\ & \times & \times & \\ & & \times & \times \\ & & & \times \\ & & & \end{bmatrix}}$$

We first triangulate A before bidiagonalizing it. We begin by computing QR factorization $A = QR$. Then we compute the Golub-Kahan bidiagonalization of R, $B = U^*RV$. The QR factorization requires $2mn^2 - \frac{2}{3}n^3$ flops and Golub-Kahan procedure now operate on the upper $n \times n$ submatrix requres $\frac{8}{3}n^3$ flops. Total amount of work for LHC bidiagonalization: $\sim 2mn^2 + 2n^3$ flops.

Phase 1:Tree Step Bidiagonalization

LHS is cheaper than Golub-Kahan for $m > \frac{5}{3}n$, but this idea can be generalized so as to realize a saving for any $m > n$. The Trick is to apply the QR factorization not at the beginning of the computation, but at the a suitable point in the middle. This is advantageouse because in Golub-Kahan process, a matrix with $m > n$ becomes skinnier as bediagonalization proceeds. After step k, aspect ratio of the remaining matrix is $(m-k)/(n-k)$, and when this figure gets sufficiently large, it makes sense to perform a QR factorization to reduce the problem to a square matrix. If the goal to minimize the operation count, QR factorization should be performed when the aspect ratio reaches $(m-k)/(n-k) = 2$.

Total amount of work for Three Step Bidiagonalization:$\sim$ $4mn^2 - \frac{4}{3}n^3 - \frac{2}{3}(m-n)^3$ flops.

This is a modest improvement over the two methods for $n < m < 2n$.

Phase 2

In *Phase2,* B is iteratively diagonalized by QR method

$$B^{(0)} \to B^{(1)} \to ... \to \Sigma$$

where

$$B^{(i+1)} = S^{(i)*}B^{(i)}T^{(i)}$$

wher $S^{(i)}$ and $T^{(i)}$ are products of Givens transformations and therefore unitary.

The matrices $T^{(i)}$are chosen so that the sequence $M^{(i)} = B^{(i)*}B^{(i)}$ converges to a diagonal matrix, while $S^{(i)}$ are chosen so that all $B^{(i)}$ are of bidiagonal form. The product of the $S^{(i)}$ and the $T^{(i)}$ are exactly the matrices H^* and G^* as defined above in eq. (1). The average number of iterations on $B^{(i)}$ is usually less than $2n$. In other words $B^{(2n)}$ is usually a good approximation to a diagonal matrix.

Implementation Details

Assume that we can destroy A and return U in the storage for A. In *Phase1* the $P^{(k)}$ are stored in the lower part of A, and $Q^{(j)}$ are stored in the upper triangular part of A. After bidiagonalization the $Q^{(j)}$ are accumulated in the storage provided for V, the two diagonals of $B^{(0)}$are copied to two other linear arrays, and $P^{(k)}$ are accumulated in A.

In *Phase2* for each i :

- $S^{(i)}$ is applied to P from the right
- $T^{(i)*}$ is applied to Q^* from the left

in order to accumulate the transformations.

References

[1] Trefethen N.L , Bau D ,*Numerical Linear Algebra. Lectures 4,5,31*

[2] Chan , *An Improved Algorithm for Computing the Singular Value Decomposition , ACM Transactions on Mathematical Software (TOMS) Volume 8 Issue 1, March 1982 Pages 72-83*

[3] Kalman D , *The SVD: A Singularly Valuable Decomposition. College Mathematics Journal. Vol 27, No 1, 1996, pp 2 - 23.*

[4] 4.Golub G.H , Van Loan C.F ,*Matrix Compitations , John Hopkins Univ Press,Baltimore 1983.*

[5] Data Mining , Lesson 6 .