
Lecture Notes 13, Math/Comp 128, Math 250

Misha Kilmer
Tufts University

November 18, 2007

Computing the SVD

We've noted already that in theory, one could compute A^*A and/or AA^* and determine the SVD once the eigenvalue decomposition of either of those matrices is known. However, this is **not recommended** from the point of view of numerical stability.

Instead, we consider the matrix:

$$H = \begin{bmatrix} 0 & A^* \\ A & 0 \end{bmatrix}$$

Note that this matrix is Hermitian.

Let $A = U\Sigma V^*$ be the full SVD of A . This means

$$AV = U\Sigma, \quad A^*U = V\Sigma^*$$

Therefore

$$\begin{bmatrix} 0 & A^* \\ A & 0 \end{bmatrix} \underbrace{\begin{bmatrix} V & V \\ U & -U \end{bmatrix}}_Q = \begin{bmatrix} V & V \\ U & -U \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & -\Sigma \end{bmatrix}$$

or

$$H = Q\Lambda Q^*$$

So the singular values of A are the absolute values of the eigenvalues of H , and the singular vectors of A can be extracted from Q .

SVD Computation

- This procedure is stable
- Standard algorithms do not form Q explicitly! It is all done implicitly (in fact, you will first wonder why I included the first slide at all)
- Key is a **Phase I** reduction to **bidiagonal form**

The Two Phase Process

1. Reduce A to bidiagonal (usually upper bidiagonal) matrix B
2. Reduce B to a diagonal matrix Σ .

As we saw in the eigenvalue studies, phase I is the “direct” phase, which we can do in $O(mn^2)$ flops.

Phase 2 is the iterative phase. It would in theory require an infinite number of operations, but converge superlinearly, implying $O(n \log(|\log(\epsilon_{mach})|))$ **iterations** needed to converge to order of ϵ_{mach} . But since the matrix is bidiagonal, this requires only $O(n)$ flops, so $O(n^2)$ for this phase.

Golub-Kahan Bidiagonalization

Alternate unitary multiplications (Householder reflectors, in the general case), to introduce zeros below main diagonal and to the right of the first super diagonal.

Example for a 6 by 4 matrix. See board for details.

$$A \rightarrow U_1^* A V_1 \rightarrow U_2^* U_1^* A V_1 V_2 \rightarrow U_3^* U_2^* U_1^* A V_1 V_2$$

Work is about $4mn^2 - 4/3n^3$ flops

Faster Phase I

If $m \gg n$, this is too much. I will only mention one (popular) alternative.

First, do a QR factorization of A : $A = QR$. Then reduce R to upper bidiagonal form by applying Golub-Kahan to R , so that $U^*RV = B$. Then we have

$$U^*Q^*AV = B$$

This is known as Lawson-Hanson-Chan (LHC) bidiagonalization.

There's a plot of the trade-off of using one method vs. the other depending on sizes of m, n on page 239.

Phase 2

A word on Phase 2.

From the 60's to the 90's, the standard looked like a variant of the QR iteration (note can be applied to bidiagonal form as “augmented” Hermitian system). Now, Divide-and-conquer is gaining in popularity.

Partial Factorization

There are plenty of applications (we've seen some) where we only need to compute a few of the “singular triplets”. For compression of data, for example, we want the singular triplets corresponding to the largest singular values only. Plenty of current work on computing some of these accurately (Baglama and Reichel) for very large A .

Alternatively: use approximations through Krylov subspace methods (our next topic for after break).