12 How to Compute the SVD

We saw earlier that the nonzero singular values of A are given by the square roots of the nonzero eigenvalues of either A^*A or AA^*. However, computing the singular values in this way is usually *not stable* (cf. solution of the normal equations).

Recall the strategy for finding the eigenvalues of a real symmetric matrix A:

1. Transform A to tridiagonal form with a unitary matrix Q_1, i.e., $A = Q_1 T Q_1^T$. This is what we called Hessenberg reduction.

2. Transform T to diagonal form using a sequence of unitary matrices and deflation (i.e., QR iteration). Thus $T^{(k)} = \left[\underline{Q}^{(k)}\right]^T T \underline{Q}^{(k)}$. Note that this is equivalent to $T = \underline{Q}^{(k)} T^{(k)} \left[\underline{Q}^{(k)}\right]^T$.

We can now combine steps 1 and 2 to get

3. $$A = \underbrace{Q_1 \underline{Q}^{(k)}}_{=Q} \underbrace{T^{(k)}}_{=\Lambda} \underbrace{\left[\underline{Q}^{(k)}\right]^T Q_1^T}_{=Q^T},$$

   where Q and Λ contain accurate approximations to the eigenvectors and eigenvalues of A, respectively.

Now we will employ a similar idea to find the SVD of an arbitrary (albeit square) matrix A (note that it will later be possible to reduce rectangular SVD problems to square ones):

1. Transform A to *bidiagonal* form B using two unitary matrices U_1 and V_1:
   $$A = U_1 B V_1^*.$$

2. Transform B to diagonal form Σ using two sequences of unitary matrices:
   $$B = \underline{U}^{(k)} \Sigma \left[\underline{V}^{(k)}\right]^*.$$

3. Combine 1. and 2. to get
   $$A = \underbrace{U_1 \underline{U}^{(k)}}_{=U} \Sigma \underbrace{\left[\underline{V}^{(k)}\right]^* V_1^*}_{=V^*},$$
   where U, Σ and V contain good approximations to the left singular vectors, singular values, and right singular vectors, respectively.

Step 2. (the computation of the approximate SVD of B) can be viewed as an eigenvalue problem of a larger matrix H in the following way. We define

$$H = \begin{bmatrix} O & B^* \\ B & O \end{bmatrix},$$

and use the SVD of B in the form $B = U\Sigma V^*$ to arrive at

$$\begin{bmatrix} O & B^* \\ B & O \end{bmatrix} \begin{bmatrix} V & V \\ U & -U \end{bmatrix} = \begin{bmatrix} V & V \\ U & -U \end{bmatrix} \begin{bmatrix} \Sigma & O \\ O & -\Sigma \end{bmatrix}, \tag{37}$$

or individually

$$\begin{aligned} B^*U &= V\Sigma, \\ BV &= U\Sigma, \\ -B^*U &= -V\Sigma, \\ BV &= U\Sigma \end{aligned}$$

(which shows how (37) follows from the SVD of B). Clearly, (37) describes the eigenvalue problem for the matrix H, and the singular values of B are given by the eigenvalues of H.

Remark If the columns of H are permuted appropriately then $P^T HP$ is symmetric and tridiagonal.

Example Take

$$B = \begin{bmatrix} a & b & 0 \\ 0 & c & d \\ 0 & 0 & e \end{bmatrix},$$

so that

$$H = \begin{bmatrix} O & B^* \\ B & O \end{bmatrix} = \begin{bmatrix} & & & a & & \\ & & & b & c & \\ & & & & d & e \\ a & b & & & & \\ & c & d & & & \\ & & e & & & \end{bmatrix}.$$

To get a tridiagonal matrix we pick the permutation matrix $P = [\boldsymbol{e}_1, \boldsymbol{e}_4, \boldsymbol{e}_2, \boldsymbol{e}_5, \boldsymbol{e}_3, \boldsymbol{e}_6]$. Then

$$HP = \begin{bmatrix} & a & & & & \\ & b & & c & & \\ & & & d & & e \\ a & & b & & & \\ & & c & & d & \\ & & & & e & \end{bmatrix}$$

and

$$P^T HP = \begin{bmatrix} & a & & & & \\ a & & b & & & \\ & b & & c & & \\ & & c & & d & \\ & & & d & & e \\ & & & & e & \end{bmatrix},$$

which is both symmetric and tridiagonal as desired.

The preceding remark shows us that we can use a modification of the QR algorithm for the second phase of the procedure. However, the matrix H is never explicitly formed. This part of the algorithm can also be replaced one of the newer eigenvalue algorithms mentioned earlier such as divide-and-conquer or RRR.

The operations count (for the general setting of rectangular matrices) is $\mathcal{O}(mn^2)$ for phase 1, and $\mathcal{O}(n^2)$ for the second phase. As before, the cost for the first phase is the dominant one.

We will now close with the discussion of an algorithm for the first phase.

12.1 Golub-Kahan Bidiagonalization

The procedure is similar to the Householder reduction for the eigenvalue case. However, now we use two different sets of Householder reflectors to get a bidiagonal (instead of an upper Hessenberg) matrix. Note that we are allowed to do that since we no longer need to perform a similarity transformation.

Example Consider

$$A = \begin{bmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{bmatrix}.$$

Householder reflectors applied alternatingly from the left and the right will be used to zero parts of the matrix as follows:

$$U_1^* A = \begin{bmatrix} \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ \mathbf{0} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ \mathbf{0} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ \mathbf{0} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ \mathbf{0} & \mathbf{x} & \mathbf{x} & \mathbf{x} \end{bmatrix} \longrightarrow U_1^* A V_1 = \begin{bmatrix} x & \mathbf{x} & \mathbf{0} & \mathbf{0} \\ 0 & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{x} & \mathbf{x} & \mathbf{x} \end{bmatrix}$$

$$\longrightarrow U_2^* U_1^* A V_1 = \begin{bmatrix} x & x & 0 & 0 \\ 0 & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{0} & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{0} & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{0} & \mathbf{x} & \mathbf{x} \end{bmatrix} \longrightarrow U_2^* U_1^* A V_1 V_2 = \begin{bmatrix} x & x & 0 & 0 \\ 0 & x & \mathbf{x} & \mathbf{0} \\ 0 & 0 & \mathbf{x} & \mathbf{x} \\ 0 & 0 & \mathbf{x} & \mathbf{x} \\ 0 & 0 & \mathbf{x} & \mathbf{x} \end{bmatrix}$$

$$\longrightarrow U_3^* U_2^* U_1^* A V_1 V_2 = \begin{bmatrix} x & x & 0 & 0 \\ 0 & x & x & 0 \\ 0 & 0 & \mathbf{x} & \mathbf{x} \\ 0 & 0 & \mathbf{0} & \mathbf{x} \\ 0 & 0 & \mathbf{0} & \mathbf{x} \end{bmatrix} \longrightarrow U_4^* U_3^* U_2^* U_1^* A V_1 V_2 = \begin{bmatrix} x & x & 0 & 0 \\ 0 & x & x & 0 \\ 0 & 0 & x & x \\ 0 & 0 & 0 & \mathbf{x} \\ 0 & 0 & 0 & \mathbf{0} \end{bmatrix} = B.$$

Note that no more right-multiplications were needed after the second step, so the corresponding identity matrices are omitted. The final matrix B is bidiagonal.

The procedure just illustrated is just like applying two separate QR factorizations, alternatingly applied to A and A^*. The resulting algorithm for $m \times n$ matrices with $m \geq n$ dates back to 1965 and is given by

Algorithm (Golub-Kahan Bidiagonalization)

for $k = 1 : n$

$\boldsymbol{x} = A(k : m, k)$

$\boldsymbol{u}_k = \boldsymbol{x} + \text{sign}(\boldsymbol{x}(1))\|\boldsymbol{x}\|_2\boldsymbol{e}_1$

$\boldsymbol{u}_k = \boldsymbol{u}_k/\|\boldsymbol{u}_k\|_2$

$A(k : m, k : n) = A(k : m, k : n) - 2\boldsymbol{u}_k\,(\boldsymbol{u}_k^* A(k : m, k : n))$

if $k \leq n - 2$

$\boldsymbol{x} = A(k, k+1 : n)$

$\boldsymbol{v}_k = \boldsymbol{x} + \text{sign}(\boldsymbol{x}(1))\|\boldsymbol{x}\|_2\boldsymbol{e}_1$

$\boldsymbol{v}_k = \boldsymbol{v}_k/\|\boldsymbol{v}_k\|_2$

$A(k : m, k+1 : n) = A(k : m, k+1 : n) - 2\,(A(k : m, k+1 : n)\boldsymbol{v}_k)\,\boldsymbol{v}_k^*$

end

end

The operations count is that for two QR factorizations, i.e., approximately $4mn^2 - \frac{4}{3}n^3$ floating point operations.

An improvement over the Golub-Kahan algorithm is given by the *Lawson-Hanson-Chan* algorithm. Its operations count is approximately $2mn^2 + 2n^3$ which is more efficient if $m > \frac{5}{3}n$.

The main idea for the Lawson-Hanson-Chan algorithm is to first compute a QR factorization of A, i.e., $A = QR$. Then one applies the Golub-Kahan algorithm to R, i.e., $R = UBV^*$. Together this results in

$$A = QUBV^*.$$

The advantage of this approach is that the bidiagonalization algorithm has to be applied only to a small $n \times n$ matrix, namely the nonzero part of R.

Remark

1. In the book [Trefethen/Bau] a hybrid method is discussed which is more efficient for any $m \geq n$.

2. In practice other implementations also exist. Some emphasize speed (such as divide-and-conquer methods), while others focus on the accuracy of small singular values (such as some QR implementations).