MATH 5330: Computational Methods of Linear Algebra

Lecture Note 15: The Singular Value Decomposition

Xianyi Zeng
Department of Mathematical Sciences, UTEP

1 Mathematical Background

Let $A\in\mathbb{R}^{n\times n}$ be arbitrary but non-singular, then because AA^t is symmetric positive definite, we can find its diagonalization:

$$AA^t = UDU^t\,,$$

where U is orthogonal and D is diagonal with positive diagonal entries. Let the diagonal elements of D be $d_1\geq d_2\geq\cdots\geq d_n>0$, we can define $\sigma_i=\sqrt{d_i}$ and $\Sigma=\mathrm{diag}(\sigma_1,\cdots,\sigma_n)$, then $D=\Sigma^2$.

Define $V=(\Sigma^{-1}U^tA)^t$, then we have:

$$V^tV=\Sigma^{-1}U^tAA^tU\Sigma^{-1}=\Sigma^{-1}U^tU\Sigma^2U^tU\Sigma^{-1}=I\,,$$

hence V is orthogonal and:

$$A=U\Sigma V^t\,, \tag{1.1}$$

known as the *singular value decomposition (SVD)* of the matrix A.

We can also start with the diagonalization of A^tA and derive a similar form. Both paths prove the existence of the SVD for non-singular matrices.

If A is singular, the diagonal matrix D and Σ can still be computed but there is some $r<n$ such that:

$$\sigma_1\geq\sigma_2\geq\cdots\sigma_r>\sigma_{r+1}=\cdots=\sigma_n=0\,.$$

It is thusly written:

$$U=[U_1\ U_2]\,,\quad \Sigma=\begin{bmatrix}\Sigma_1 & \\ & 0\end{bmatrix}\Rightarrow AA^t=U\Sigma^2U^t=U_1\Sigma_1^2U_1^t\,,$$

where $U_1\in\mathbb{R}^{n\times r}$, $U_2\in\mathbb{R}^{n\times(n-r)}$, and $\Sigma_1\in\mathbb{R}^{r\times r}$ is diagonal and non-singular. We similarly define $V_1=(\Sigma_1^{-1}U_1^tA)^t\in\mathbb{R}^{n\times r}$, and find that:

$$V_1^tV_1=\Sigma_1^{-1}U_1^tAA^tU_1\Sigma_1^{-1}=\Sigma_1^{-1}U_1^tU_1\Sigma_1^2U_1^tU_1\Sigma_1^{-1}=I_r\,,$$

hence the column vectors of V_1 are orthonormal, and we can extend them to a full set of orthonormal basis of $\mathbb{R}^n$ – and define $V=[V_1\ V_2]$. Then it is not difficult to check that $A=U\Sigma V^t$ and establish the existence of the SVD for any general matrix.

The singular value decomposition is useful in many senses, for example one can compute the inverse of a non-singular matrix A by $A^{-1}=V\Sigma^{-1}U^t$. Furthermore, the condition number w.r.t. the L^2-norm is computed as σ_1/σ_n. Indeed, $\kappa(A)=||A||_2\left|\left|A^{-1}\right|\right|_2$ and the right hand sides are computed from:

$$\begin{aligned}||A||_2&=\max_{\boldsymbol{x}:\,||\boldsymbol{x}||_2=1}||A\boldsymbol{x}||_2=\max_{\boldsymbol{x}:\,||\boldsymbol{x}||_2=1}\sqrt{\boldsymbol{x}A^tA\boldsymbol{x}}=\max_{\boldsymbol{x}:\,||\boldsymbol{x}||_2=1}\sqrt{\boldsymbol{x}V\Sigma^2V^t\boldsymbol{x}}\\&=\max_{\boldsymbol{y}=V^t\boldsymbol{x}:\,||\boldsymbol{x}||_2=1}\sqrt{\boldsymbol{y}^t\Sigma^2\boldsymbol{y}}=\max_{\boldsymbol{y}:\,||\boldsymbol{y}||_2=1}\sqrt{\boldsymbol{y}^t\Sigma^2\boldsymbol{y}}=\sigma_1^2\,;\end{aligned}$$

and similarly $||A^{-1}||_2 = 1/\sigma_n^2$.

Now we relax the requirement of A being square, and consider $A \in \mathbb{R}^{m\times n}$. Following a similar procedure to the previous case of singular square matrices, we can show the existence of the singular value decomposition of A in the following form:

$$A = U\Sigma V : \quad U \in \mathbb{R}^{m\times m} \text{ and } V \in \mathbb{R}^{n\times n} \text{ are orthogonal , } \Sigma \in \mathbb{R}^{m\times n} \text{ is diagonal.} \tag{1.2}$$

Here by the last statement we mean only the (i,i)-th component of Σ is possibly non-zero, where $1 \le i \le \min(m,n)$. If we denote these diagonal elements by σ_i, then we further require:

$$\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_{\min(m,n)} \ge 0 \,.$$

The SVD of the general matrices will allow us to compute the pseudo inverse of any matrix A as $A^+ = V\Sigma^+U^t$ (see Section 3 of Lecture 8). Furthermore, it is very useful in mathematical proofs. For example, if $n \le m$ and $\text{rank}(A) = n$, then $A^tA = V\Sigma^t\Sigma V^t$ is non-singular since $\Sigma^t\Sigma = \text{diag}(\sigma_1^2,\sigma_2^2,\cdots,\sigma_n^2)$ is non-singular.

2 The SVD Algorithm

Computing the SVD of $A \in \mathbb{R}^{m\times n}$ is actually easier than the eigenvalue problem for a general matrix $A \in \mathbb{R}^{n\times n}$ (nor should they be compared since the SVD and the eigenvalue decomposition or the real Schur decomposition are very different animals). The existence proof before actually sheds some light on a numerical method.

Suppose $m \ge n$, then the method involves the following steps:

Algorithm 2.1 The Jacobi-SVD Algorithm for $A \in \mathbb{R}^{m\times n}$, $m \ge n$.

1: Compute $C = A^tA$.
2: Use the Jacobi method to compute $V^tCV = D = \text{diag}(\sigma_1^2,\cdots,\sigma_n^2)$.
3: Compute the QR decomposition of AV: $AV = U_1R_1$.
4: R_1 must be diagonal, and compute $\Sigma = |R_1|$ and adjust U_1 correspondingly to obtain U.

If every step is done exactly, we have after the third step:

$$R^tU^tUR = V^tA^tAV \quad \text{ or equivalently } \quad R^tR = D \,.$$

Hence R must be diagonal. This can be shown, for example, if D is non-singular we have $R^t = DR^{-1}$ with a lower-triangular matrix on the left hand side and an upper-triangular matrix on the right hand side; and the case of singular D is similar.

In practice, every step is only performed approximately, with the error determined from either accumulated round-off errors or the user-specified convergence tolerance. A full study of the validity of Algorithm 2.1 requires deeper perturbation theory; here we adopt the belief that the SVD is stable against small perturbations. If $m < n$, of course the method needs modification and we start with $C = AA^t$.

A more preferred SVD method is due to Golub and Kahan in 1965, for which the first author also earned the popular title "Prof SVD" (this is also his plate number). The technique finds U

and V simultaneously by implicitly applying the QR method to A^tA.

The bidiagonalization. Still assuming $m \geq n$, the first step of the SVD algorithm performs the bidiagonalization of A:

$$U_B^t A V_B = \begin{bmatrix} B \\ 0 \end{bmatrix}, \quad B = \begin{bmatrix} d_1 & f_1 & & \cdots & 0 \\ 0 & d_2 & \ddots & & \vdots \\ & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & f_{n-1} \\ 0 & \cdots & & 0 & d_n \end{bmatrix}. \tag{2.1}$$

We already know the tool to achieve this. Recall that for any two numbers $1 \leq i \leq m$ and $1 \leq j \leq n$, we can left-multiply a properly chosen Householder matrix to A, so that (1) the first i rows are unchanged, (2) the rest $m-i$ elements of the j-th column become a multiple of $\boldsymbol{e}_1 \in \mathbb{R}^{m-i}$, and (3) all the last $m-i$ rows are changed, but each new row is a linear combination of the old $m-i$ rows. Similarly, we can choose an appropriate Householder matrix and right-multiply it to A, so that: (1) the first j columns are unchanged, (2) the rest $n-j$ elements of the i-th row become a multiple of $\boldsymbol{e}_1^t$ of $\mathbb{R}^{n-j}$, and (3) all the last $n-j$ columns are changed, but each new column is a linear combination of the old $n-j$ rows.

To this end, we arrange Householder transformations carefully and try to interlace the left ones and right ones to reduce A to a bidiagonal form as shown in (2.1). This process is illustrated next for a 6×4 matrix:

$$A = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \xrightarrow{U_1^t A} \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{bmatrix} \xrightarrow{U_1^t A V_1} \begin{bmatrix} * & * & 0 & 0 \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{bmatrix} \xrightarrow{U_2^t U_1^t A V_1} \begin{bmatrix} * & * & 0 & 0 \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \end{bmatrix} \xrightarrow{U_2^t U_1^t A V_1 V_2}$$

$$\begin{bmatrix} * & * & 0 & 0 \\ 0 & * & * & 0 \\ 0 & 0 & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \end{bmatrix} \xrightarrow{U_3^t U_2^t U_1^t A V_1 V_2} \begin{bmatrix} * & * & 0 & 0 \\ 0 & * & * & 0 \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & * \end{bmatrix} \xrightarrow{U_4^t U_3^t U_2^t U_1^t A V_1 V_2} \begin{bmatrix} * & * & 0 & 0 \\ 0 & * & * & 0 \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

The Golub-Kahan SVD Step. This step is essentially applying the QR method to the symmetric tridiagonal matrix $T = B^t B$:

$$T = B^t B = \begin{bmatrix} d_1^2 & d_1 f_1 & 0 & \cdots & & & 0 \\ d_1 f_1 & d_2^2 + f_1^2 & d_2 f_2 & & & & \\ 0 & d_2 f_2 & \ddots & \ddots & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & & \\ & & & \ddots & d_{n-1}^2 + f_{n-2}^2 & d_{n-1} f_{n-1} \\ 0 & & \cdots & & d_{n-1} f_{n-1} & d_n^2 + f_{n-1}^2 \end{bmatrix}$$

Because T is symmetric, we do not actually need to worry about complex eigenvalues, but nevertheless in the effort in reducing the $(n,n-1)$-component of T to nearly zero the Golub-Kahan SVD step still start working with the trailing 2×2 matrix of T.

Particularly, the shift is chosen as the eigenvalue μ of:

$$T((n-1):n,(n-1):n) = \begin{bmatrix} d_{n-1}^2+f_{n-2}^2 & d_{n-1}f_{n-1} \\ d_{n-1}f_{n-1} & d_n^2+f_{n-1}^2 \end{bmatrix}$$

that is closer to $d_n^2+f_{n-1}^2$; then the QR decomposition $Q^t(T-\mu I)=R$ is computed by using a sequence of Givens rotations to eliminate the sub-diagonals of T: $Q^t = G_{n-1}G_{n-2}\cdots G_1$. The next iterates is given by $Q^t(T-\mu I)Q$, which remains to be tridiagonal.

An working example of one step is given by Golub and van Loan as below:

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 0 & 1 & 3 & .01 \\ 0 & 0 & .01 & 4 \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} .5 & .5916 & 0 & 0 \\ .5916 & 1.785 & .1808 & 0 \\ 0 & .1808 & 3.7140 & .0000044 \\ 0 & 0 & .0000044 & 4.002497 \end{bmatrix}.$$

The fast convergence of the last sub-diagonal element to zero is one major reason why the QR method is preferred to the Jacobi method for tridiagonal matrices – the convergence can be shown to be quadratic for the former whereas it is only linear for the latter.

The *Golub-Kahan SVD step* assumes that T is unreduced (i.e., f_i and d_i are all non-zero except for d_n), and it describes how to incorporate the construction of the matrix Q with its right-multiplication to $Q^t(T-\mu I)$, so that the matrix B^tB will never be explicitly formed. Note that the first Givens rotation has the effect of computing $G_1T = G_1B^tB$ (we omit the shift part for now). Without forming T explicitly and reusing the storage for B (two vectors storing the diagonal and the superdiagonal elements), the effect is demonstrated below for the case $n=6$:

$$B \leftarrow BG_1^t = \begin{bmatrix} * & * & 0 & 0 & 0 & 0 \\ + & * & * & 0 & 0 & 0 \\ 0 & 0 & * & * & 0 & 0 \\ 0 & 0 & 0 & * & * & 0 \\ 0 & 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & 0 & * \end{bmatrix},$$

where + denotes a newly added non-zero entry that needs to be stored separately. Next a sequence of Givens rotations $U_1, V_2, U_2, \cdots, V_{n-1}$, and U_{n-1} are used to chase the unwanted nonzero element down the bidiagonal.

$$B \leftarrow U_1B = \begin{bmatrix} * & * & + & 0 & 0 & 0 \\ 0 & * & * & 0 & 0 & 0 \\ 0 & 0 & * & * & 0 & 0 \\ 0 & 0 & 0 & * & * & 0 \\ 0 & 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & 0 & * \end{bmatrix}, \quad B \leftarrow BV_2^t = \begin{bmatrix} * & * & 0 & 0 & 0 & 0 \\ 0 & * & * & 0 & 0 & 0 \\ 0 & + & * & * & 0 & 0 \\ 0 & 0 & 0 & * & * & 0 \\ 0 & 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & 0 & * \end{bmatrix}, \quad B \leftarrow U_2B = \begin{bmatrix} * & * & 0 & 0 & 0 & 0 \\ 0 & * & * & + & 0 & 0 \\ 0 & 0 & * & * & 0 & 0 \\ 0 & 0 & 0 & * & * & 0 \\ 0 & 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & 0 & * \end{bmatrix}, \cdots$$

In the end, we compute:

$$\overline{B} = (U_{n-1}\cdots U_1)B(G_1^tV_2^t\cdots V_{n-1}^t) = \overline{U}^tB\overline{V}.$$

It is not difficult to see that the first column of $\overline{V}$ is exactly the same as Q, which is what we want: Then by the implicit Q theorem (which states the nearly uniqueness of the reduction of a matrix to upper-Hessenberg form once the first column of the transformation matrix is given) we can assert that $\overline{V}$ and Q are essentially the same.

One last issue in the Golub-Kahan SVD step is in order to have an unreduced T, all the diagonal elements of B except the last one also need to be non-zero. If this happens, say $d_k = 0$ for some $k < n$, we again use a sequence of Givens rotations to left multiply B (row operations) so that the latter is put in block bidiagonal form. The following example illustrates how it is done for the case $n = 6$ and $k = 3$:

$$B = \begin{bmatrix} * & * & 0 & 0 & 0 & 0 \\ 0 & * & * & 0 & 0 & 0 \\ 0 & 0 & 0 & * & 0 & 0 \\ 0 & 0 & 0 & * & * & 0 \\ 0 & 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & 0 & * \end{bmatrix} \xrightarrow{G_{3,4}} \begin{bmatrix} * & * & 0 & 0 & 0 & 0 \\ 0 & * & * & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & + & 0 \\ 0 & 0 & 0 & * & * & 0 \\ 0 & 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & 0 & * \end{bmatrix} \xrightarrow{G_{3,5}} \begin{bmatrix} * & * & 0 & 0 & 0 & 0 \\ 0 & * & * & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & + \\ 0 & 0 & 0 & * & * & 0 \\ 0 & 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & 0 & * \end{bmatrix} \xrightarrow{G_{3,6}} \begin{bmatrix} * & * & 0 & 0 & 0 & 0 \\ 0 & * & * & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & * & * & 0 \\ 0 & 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & 0 & * \end{bmatrix}. \quad (2.2)$$

Finally, if $d_n = 0$ is the only zero element of the diagonal and superdiagonal ones in B, the Golub-Kahan SVD step still works because T is unreduced. However, this essentially means we're hoping that the QR iterations will give rise to an eigenvalue that is close to zero; and we expect to do better than using an iterative procedure just to find a zero eigenvalue! This happens, for example, in (2.2) where we eventually obtain a 3×3 bidiagonal matrix with a zero trailing element. The idea is to right-multiplying with Givens matrices (column operations) to put the entire column to zero; in the next example we only display the upper-left 3×3 submatrix from the previous example:

$$\begin{bmatrix} * & * & 0 \\ 0 & * & * \\ 0 & 0 & 0 \end{bmatrix} \xrightarrow{G_{2,3}} \begin{bmatrix} * & * & + \\ 0 & * & 0 \\ 0 & 0 & 0 \end{bmatrix} \xrightarrow{G_{1,3}} \begin{bmatrix} * & * & 0 \\ 0 & * & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

The whole step is described in Algorithm 2.2. The net outcome of this algorithm is to overwrite B with the bidiagonal matrix $\overline{B} = \overline{U}^t B \overline{V}$ where both $\overline{U}$ and $\overline{V}$ are orthogonal and $\overline{V}$ is essentially the one that would be obtained by applying the QR iteration to $T = B^t B$.

Combining everything, we have the followng algorithm to compute the SVD of $A \in \mathbb{R}^{m \times n}$ in the case $m \geq n$. The outcome of this algorithm is overwriting A with $U^t A V = \Sigma + E$, where $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ are orthogonal, $\Sigma \in \mathbb{R}^{m \times n}$ is diagonal; and E satisfies $||E||_2 \approx \varepsilon ||A||_2$, where ε is a small positive number that only depends on the sizes m and n, the machine accuracy, and the parameter ε_0.

Algorithm 2.2 The Golub-Kahan SVD Step

Compute μ as described before.
Set $y = t_{11} - \mu$.
Set $z = t_{12}$.
for $k = 1, 2, \cdots, n-1$ **do**
    Determine $c_1 = \cos(\theta_1)$ and $s_1 = \sin(\theta_1)$ such that

$$[y \;\; z]\begin{bmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{bmatrix} = [* \;\; 0]\,.$$

    Compute $B = BG_{k,k+1}(\theta_1)^t$.
    Set $y = b_{kk}$ and $z = b_{k+1,k}$.
    Determine $c_2 = \cos(\theta_2)$ and $s_2 = \sin(\theta_2)$ such that

$$\begin{bmatrix} c_2 & -s_2 \\ s_2 & c_2 \end{bmatrix}\begin{bmatrix} y \\ z \end{bmatrix} = \begin{bmatrix} * \\ 0 \end{bmatrix}.$$

    Compute $B = G_{k,k+1}(\theta_2)B$.
    if $k < n-1$ **then**
        Set $y = b_{k,k+1}$ and $z = b_{k,k+2}$
    end if
end for

Algorithm 2.3 The SVD Algorithm ($m \geq n$)

Compute the bidiagonalization:

$$\begin{bmatrix} B \\ 0 \end{bmatrix} \leftarrow U^t A V\,.$$

Set $q = 0$ and pick a small positive number ε_0.
while $q < n$ **do**
    Set $b_{i,i+1}$ to zero if $|b_{i,i+1}| \leq \varepsilon_0(|b_{ii}| + |b_{i+1,i+1}|)$ for any $1 \leq i \leq n-1$.
    Find the largest q and the smallest p such that:

$$B = \begin{bmatrix} B_{11} & 0 & 0 \\ 0 & B_{22} & 0 \\ 0 & 0 & B_{33} \end{bmatrix}$$

    where B_{33} is diagonal and B_{22} has nonzero superdiagonal.
    if $q < n$ **then**
        if Any diagonal entry of B_{22} is zero **then**
            Bidiagonalize B_{22} as described before.
        else
            Apply Algorithm 2.2 to B_{22}.
            Copmute $B = \text{diag}(I_p, U, I_{q+m-n})^t B \text{diag}(I_p, V, I_q)$.
        end if
    end if
end while