第五讲　对称特征值问题

对称特征值问题的常用算法介绍

- **Jacobi 迭代方法**: 最古老, 收敛速度较慢, 但精度较高, 适合并行计算
- **Rayleigh 商迭代方法**: 一般具有三次收敛性, 但需要解方程组
- **对称 QR 迭代方法**: 对于对称三对角矩阵, 若只计算特征值, 速度最快
- **分而治之法**: 基本思想是将大矩阵分解成小矩阵, 然后利用递归思想, 是目前计算对称三对角矩阵的**特征值和特征向量**的首选方法.
- **对分法 + 反迭代**: 主要用于计算对称三对角矩阵在某个区间中的特征值, 运算量约为 $\mathcal{O}(kn)$, 其中 k 为所需计算的特征值的个数.

✍ **注**: 除了 Jacobi 和 Rayleigh 商迭代外, 其余算法都需要先三对角化, 大约需 $4n^3/3$ 运算量, 若需计算特征向量, 则为 $8n^3/3$.

1 Jacobi 迭代

基本思想

通过一系列 **Jacobi 旋转**, 将 A 转化为对角矩阵:

$$A^{(0)} = A, \quad A^{(k+1)} = J_k^{\mathsf{T}} A^{(k)} J_k, \quad k = 0, 1, \ldots,$$

其中 J_k 为 Jacobi 旋转: $J_k = G(i_k, j_k, \theta_k) =$

$$\begin{array}{c} \\ \\ i_k \\ \\ j_k \\ \\ \end{array}
\begin{array}{c}
\begin{array}{cccccc} & & i_k & & j_k & \end{array} \\
\left[\begin{array}{c|ccc|c}
I & & & & \\
\hline
 & \cos\theta_k & \cdots & -\sin\theta_k & \\
 & \vdots & \ddots & \vdots & \\
 & \sin\theta_k & \cdots & \cos\theta_k & \\
\hline
 & & & & I
\end{array}\right]
\end{array}$$

目标: $A^{(k)}$ 收敛到一个对角矩阵

理论基础

引理 设 $A \in \mathbb{R}^{2\times 2}$ 是对称矩阵, 则存在 Givens 变换 $G \in \mathbb{R}^{2\times 2}$ 使得 $G^{\mathsf{T}}AG$ 为对角阵.

证明概要.

$$
\begin{aligned}
G^{\mathsf{T}}AG &= \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} a & b \\ b & c \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \\
&= \begin{bmatrix} a\cos^2\theta + c\sin^2\theta + b\sin 2\theta & \frac{1}{2}(c-a)\sin 2\theta + b\cos 2\theta \\ \frac{1}{2}(c-a)\sin 2\theta + b\cos 2\theta & a\sin^2\theta + c\cos^2\theta - b\sin 2\theta \end{bmatrix}
\end{aligned}
$$

令 $\frac{1}{2}(c-a)\sin 2\theta + b\cos 2\theta = 0$, 可得

$$
\frac{a-c}{2b} = \cot 2\theta = \frac{1-\tan^2\theta}{2\tan\theta} \quad \Longrightarrow \quad \tan\theta = \frac{\operatorname{sign}(\tau)}{|\tau| + \sqrt{1+\tau^2}}, \quad \tau = \frac{a-c}{2b}.
$$

记 $\mathrm{off}(A)$ 为所有非对角元素的平方和, 即

$$\mathrm{off}(A) = \sum_{i \neq j} a_{ij}^2 = \|A\|_F^2 - \sum_{i=1}^{n} a_{ii}^2 \, ,$$

我们的目标就是使得 $\mathrm{off}(A)$ 尽快趋向于 0.

引理 设 $A = [a_{ij}]_{n\times n} \in \mathbb{R}^{n\times n}$ 是对称矩阵, $\hat{A} = [\hat{a}_{ij}]_{n\times n} = J^{\mathsf{T}} A J$, $J = G(i, j, \theta)$, 其中 θ 的选取使得 $\hat{a}_{ij} = \hat{a}_{ji} = 0$, 则

$$\mathrm{off}(\hat{A}) = \mathrm{off}(A) - 2a_{ij}^2.$$

(板书)

算法 1 Jacobi 迭代算法

Given a symmetric matrix $A \in \mathbb{R}^{n\times n}$
if eigenvectors are desired **then**
    set $J = I$ and $shift = 1$
end if
while not converge **do**
    choose an index pair (i, j) such that $a_{ij} \neq 0$
    $\tau = (a_{ii} - a_{jj})/(2a_{ij})$
    $t = \text{sign}(\tau)/(|\tau| + \sqrt{1+\tau^2})$ % 计算 $\tan\theta$
    $c = 1/\sqrt{1+t^2}$, $s = c \cdot t$ % 计算 Givens 变换
    $A = G(i, j, \theta)^{\mathsf{T}} A G(i, j, \theta)$ % 实际计算时不需要做矩阵乘积
    if $shift = 1$ **then**
        $J = J \cdot G(i, j, \theta)$
    end if
end while

经典 Jacobi 迭代算法

a_{ij} 的选取：一种直观的选取方法就是使得 a_{ij} 为所有非对角元素中绝对值最大的一个，这就是经典 Jacobi 算法.

算法 2 经典 Jacobi 迭代算法

1: ... ...
2: **while** off$(A)>$ tol **do**
3: choose (i,j) such that $|a_{ij}| = \max_{k\neq l}|a_{kl}|$
4: ... ...
5: **end while**

可以证明, 经典 Jacobi 算法至少是线性收敛的.

定理 对于经典 Jacobi 算法 2, 有

$$\operatorname{off}(A^{(k+1)}) \leq \left(1 - \frac{1}{N}\right) \operatorname{off}(A^{(k)}), \quad N = \frac{n(n-1)}{2}.$$

故 k 步迭代后, 有

$$\operatorname{off}(A^{(k)}) \leq \left(1 - \frac{1}{N}\right)^k \operatorname{off}(A^{(0)}) = \left(1 - \frac{1}{N}\right)^k \operatorname{off}(A).$$

(留作课外自习)

事实上, 经典 Jacobi 算法最终是二次局部收敛的.

定理 经典 Jacobi 算法 2 是 N 步局部二次收敛的, 即对足够大的 k, 有

$$\mathrm{off}\big(A^{(k+N)}\big) = O\Big(\mathrm{off}^2\big(A^{(k)}\big)\Big).$$

循环 Jacobi 迭代方法

每一步都寻找绝对值最大的非对角元, 比较费时. $\longrightarrow$ 改进: **逐行扫描**

算法 3 循环 Jacobi 迭代算法 (逐行扫描)

```
... ...
while off(A)> tol do
    for i = 1 to n − 1 do
        for j = i + 1 to n do
            if a_ij ≠ 0 then
                ... ...
            end if
        end for
    end for
end while
```

✍ 循环 Jacobi 迭代方法也具有局部二次收敛性.

2 Rayleigh 商迭代方法

在反迭代方法中, 以 Rayleigh 商作为位移, 就得到 Rayleigh 商迭代方法.

算法 4 Rayleigh 商迭代算法 (RQI, Rayleigh Quotient Iterations)

1: Given an initial guess $x^{(0)}$ with $\|x^{(0)}\|_2 = 1$
2: compute the Rayleigh quotient $\rho_0 = (x^{(0)}, Ax^{(0)})$
3: set $k = 1$
4: **while** not converge **do**
5: $\quad \sigma = \rho_{k-1}$
6: $\quad y^{(k)} = (A - \sigma I)^{-1} x^{(k-1)}, \quad x^{(k)} = y^{(k)} / \|y^{(k)}\|_2$
7: $\quad \rho_k = (x^{(k)}, Ax^{(k)})$
8: $\quad k = k + 1$
9: **end while**

Rayleigh 商迭代法的收敛性

定理 设 $A \in \mathbb{R}^{n\times n}$ 对称, 且特征值都是单重的. 则当误差足够小时, Rayleigh 商迭代中每步迭代所得的正确数字的位数增至三倍, 即 Rayleigh 商迭代是局部三次收敛的. (留作课外自习)

设 $x^{(k)}$ 收敛到 A 的某个 (单位) 特性向量 v,

- 记 $\varepsilon_k = \|x^{(k)} - v\|_2$, 则当 ε_k 充分小时有 $\varepsilon_{k+1} = \mathcal{O}(\varepsilon_k^3)$.
- 记 $\phi_k = \angle(x^{(k)}, v)$, 则当 $k \to \infty$ 时有 $\left|\dfrac{\phi_{k+1}}{\phi_k^3}\right| < 1$.

RQI 算法具有局部三次收敛性, 但无法确定收敛到哪个特征向量 (特征值), 因此可以作为其他算法的 加速手段, 即先使用其他算法 (比如幂迭代) 计算出所需特征向量 (特征值) 的近似值, 然后再使用 RQI 算法加速.

实际计算中判断 $(\rho_k, x^{(k)})$ 是否收敛可以观察残量 $r_k = (A - \rho_k I)x^{(k)}$ 是否趋于零.

下面是关于 RQI 算法的全局收敛性.

定理 在 RQI 算法中, 设 $r_k = (A - \rho_k I)x^{(k)}$, 则有

$$\|r_{k+1}\| \leq \|r_k\|,$$

其中等号成立当且仅当 $\rho_{k+1} = \rho_k$ 且 $x^{(k)}$ 是 $(A - \rho_k I)^2$ 的特征向量.

3 对称 QR 迭代

将带位移的隐式 QR 方法运用到对称矩阵上, 就得到**对称 QR 迭代方法**.

基本步骤

1. 对称三对角化: 利用 Householder 变换, 将 A 化为对称三对角矩阵, 即计算正交矩阵 Q 使得 $T = QAQ^{\mathsf{T}}$ 为对称三对角矩阵;
2. 使用带 (单) 位移的隐式 QR 迭代算法计算 T 的特征值与特征值向量;
3. 计算 A 的特征向量.

对称三对角化

任何对称矩阵 $A \in \mathbb{R}^{n\times n}$ 都可以通过正交相似变换转化成一个对称三对角矩阵 T. (可利用 Householder 变换或 Givens 变换)

对称 QR 迭代算法的运算量

- 三对角化 $4n^3/3 + \mathcal{O}(n^2)$, 若需计算特征向量, 则为 $8n^3/3 + \mathcal{O}(n^2)$;
- 对 T 做带位移的隐式 QR 迭代, 每次迭代的运算量为 $6n$;
- 计算特征值, 假定每个平均迭代 2 步, 则总运算量约为 $12n^2$ (或 $6n^2$);
- 若要计算 T 的所有特征值和特征向量, 则运算量为 $6n^3 + \mathcal{O}(n^2)$;
- 若只要计算 A 的所有特征值, 运算量为 $4n^3/3 + \mathcal{O}(n^2)$;
- 若计算 A 的所有特征值和特征向量, 则运算量为 $26n^3/3 + \mathcal{O}(n^2)$;

位移的选取

位移的好坏直接影响到算法的收敛速度. 设

$$A_k = \begin{bmatrix} a_1^{(k)} & b_1^{(k)} & & \\ b_1^{(k)} & \ddots & \ddots & \\ & \ddots & \ddots & b_{n-1}^{(k)} \\ & & b_{n-1}^{(k)} & a_n^{(k)} \end{bmatrix},$$

一种简单的位移选取策略就是令 $\sigma_k = a_n^{(k)}$.

事实上, $a_n^{(k)}$ 就是收敛到特征向量的迭代向量的 Rayleigh 商. 这种位移选取方法几乎对所有的矩阵都有三次渐进收敛速度, 但也存在不收敛的例子.

Wilkinson 位移

取 $\begin{bmatrix} a_{n-1}^{(k)} & b_{n-1}^{(k)} \\ b_{n-1}^{(k)} & a_n^{(k)} \end{bmatrix}$ 的最接近 $a_n^{(k)}$ 的特征值作为位移.

通过计算可得 Wilkinson 位移为

$$\sigma = a_n^{(k)} + \delta - \operatorname{sign}(\delta)\sqrt{\delta^2 + \left(b_{n-1}^{(k)}\right)^2}, \quad \text{其中} \quad \delta = \frac{1}{2}(a_{n-1}^{(k)} - a_n^{(k)}).$$

定理 采用 Wilkinson 位移的 QR 迭代是整体收敛的, 且至少是线性收敛. 事实上, 几乎对所有的矩阵都是渐进三次收敛的.

例 带 Wilkinson 位移的 QR 迭代算法收敛性演示. (Eig_TriQR.m)

```
n = 5; tol = 1e-8;
A = randn(n); T = hess(A+A') % Hessenberg 化
for iter = 1 : 4
    % Compute the shift
    sub_matrix = T(n-1:n,n-1:n);
    Eig_sub_matrix = eig(sub_matrix);
    if ( abs(T(n,n) - Eig_sub_matrix(1)) < abs(T(n,n) - Eig_sub_matrix(2)) )
        shift = Eig_sub_matrix(1);
    else
        shift = Eig_sub_matrix(2);
    end
    % Perform QR iteration
    [Q,R] = qr(T-shift*eye(n));
    T = R*Q + shift*eye(n);
    % Enforce symmetry explicitly
    fprintf('iter = %d \n', iter)
    T
end
```

✍ 注：这里只显示计算一个特征值的演示代码, 完整代码请大家自行完成.

```
T =
  -1.1495e+00   1.9345e-01            0            0            0
   1.9345e-01  -5.7144e-01  -3.5163e+00            0            0
            0  -3.5163e+00   1.4138e+00  -1.2639e+00            0
            0            0  -1.2639e+00  -2.0125e-01   4.3216e+00
            0            0            0   4.3216e+00   1.9285e+00
iter = 1
T =
  -1.1606e+00   2.0488e-01            0            0            0
   2.0488e-01  -3.1370e+00  -1.0240e+00            0            0
            0  -1.0240e+00   1.2439e+00  -3.5560e+00            0
            0            0  -3.5560e+00  -1.1053e+00   3.1938e-01
            0            0            0   3.1938e-01   5.5790e+00
iter = 2
T =
  -1.1748e+00   2.6621e-01            0            0            0
   2.6621e-01  -3.3005e+00  -6.4187e-01            0            0
            0  -6.4187e-01  -3.1162e+00  -1.8413e+00            0
            0            0  -1.8413e+00   3.4052e+00   1.3658e-03
            0            0            0   1.3658e-03   5.6064e+00
```

```
iter = 3
T =
  -1.1990e+00   3.4962e-01            0            0            0
   3.4962e-01  -3.3676e+00  -6.3718e-01            0            0
            0  -6.3718e-01  -3.4941e+00  -3.6853e-01            0
            0            0  -3.6853e-01   3.8743e+00   1.4108e-10
            0            0            0   1.4108e-10   5.6064e+00
iter = 4
T =
  -1.2569e+00   4.9647e-01            0            0            0
   4.9647e-01  -3.4224e+00  -6.4282e-01            0            0
            0  -6.4282e-01  -3.3999e+00  -7.0835e-06            0
            0            0  -7.0835e-06   3.8929e+00            0
            0            0            0            0   5.6064e+00
iter = 5
T =
  -1.3717e+00   6.9691e-01            0            0            0
   6.9691e-01  -3.4212e+00  -6.3796e-01            0            0
            0  -6.3796e-01  -3.2863e+00  -5.2850e-20            0
            0            0  -5.2850e-20   3.8929e+00            0
            0            0            0            0   5.6064e+00
```

```
iter = 6
T =
  -1.2373e+00  -5.2803e-01            0            0            0
  -5.2803e-01  -3.9958e+00   8.9840e-02            0            0
            0   8.9840e-02  -2.8461e+00            0            0
            0            0            0   3.8929e+00            0
            0            0            0            0   5.6064e+00
iter = 7
T =
  -1.1937e+00   3.9668e-01            0            0            0
   3.9668e-01  -4.0455e+00  -6.4555e-05            0            0
            0  -6.4555e-05  -2.8400e+00            0            0
            0            0            0   3.8929e+00            0
            0            0            0            0   5.6064e+00
iter = 8
T =
  -1.1695e+00  -2.9629e-01            0            0            0
  -2.9629e-01  -4.0697e+00   1.2962e-14            0            0
            0   1.2962e-14  -2.8400e+00            0            0
            0            0            0   3.8929e+00            0
            0            0            0            0   5.6064e+00
```

```
iter = 9
T =
  -1.1396e+00  -2.2223e-17            0            0            0
  -2.2223e-17  -4.0996e+00            0            0            0
            0            0  -2.8400e+00            0            0
            0            0            0   3.8929e+00            0
            0            0            0            0   5.6064e+00
```

4 分而治之法

分而治之法由 Cuppen 于 1981 年首次提出, 主要是利用的递归思想, 将大矩阵的特征值问题转化为小矩阵的特征值问题, 通过矩阵分块不断递归.

分而治之法是同时计算特征值和特征向量.

考虑不可约对称三对角矩阵

$$T=\left[\begin{array}{cccc|cccc}
a_1 & b_1 & & & & & & \\
b_1 & \ddots & \ddots & & & & & \\
 & \ddots & a_{m-1} & b_{m-1} & & & & \\
 & & b_{m-1} & a_m & b_m & & & \\
\hline
 & & & b_m & a_{m+1} & b_{m+1} & & \\
 & & & & b_{m+1} & \ddots & \ddots & \\
 & & & & & \ddots & \ddots & b_{n-1} \\
 & & & & & & b_{n-1} & a_n
\end{array}\right]$$

$$
= \left[\begin{array}{cccc|cccc}
a_1 & b_1 & & & & & & \\
b_1 & \ddots & \ddots & & & & & \\
 & \ddots & a_{m-1} & b_{m-1} & & & & \\
 & & b_{m-1} & a_m - b_m & & & & \\
\hline
 & & & & a_{m+1} - b_m & b_{m+1} & & \\
 & & & & b_{m+1} & \ddots & \ddots & \\
 & & & & & \ddots & \ddots & b_{n-1} \\
 & & & & & & b_{n-1} & a_n
\end{array}\right] + \left[\begin{array}{cc|cc}
 & & b_m & b_m \\
 & & b_m & b_m \\
\hline
 & & & \\
 & & &
\end{array}\right]
$$

$$
= \left[\begin{array}{c|c} T_1 & 0 \\ \hline 0 & T_2 \end{array}\right] + b_m v v^{\mathsf{T}},
$$

其中 $v = [0, \ldots, 0, 1, 1, 0, \ldots, 0]^{\mathsf{T}}$.

递推方法: 假定 T_1 和 T_2 的特征值分解已经计算得到

设 $T_1 = Q_1\Lambda_1 Q_1^{\mathsf{T}}, T_2 = Q_2\Lambda_2 Q_2^{\mathsf{T}}$, 则

$$T = \begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix} + b_m vv^{\mathsf{T}} = \begin{bmatrix} Q_1\Lambda_1 Q_1^{\mathsf{T}} & 0 \\ 0 & Q_2\Lambda_2 Q_2^{\mathsf{T}} \end{bmatrix} + b_m vv^{\mathsf{T}}$$

$$= \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} \left(\begin{bmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{bmatrix} + b_m uu^{\mathsf{T}} \right) \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix}^{\mathsf{T}},$$

其中

$$u = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix}^{\mathsf{T}} v = \begin{bmatrix} Q_1^{\mathsf{T}} \text{ 的最后一列} \\ Q_2^{\mathsf{T}} \text{ 的第一列} \end{bmatrix}.$$

令 $\alpha = b_m$, $D = \mathrm{diag}(\Lambda_1, \Lambda_2) = \mathrm{diag}(d_1, d_2, \ldots, d_n)$, 并假定 $d_1 \geq \cdots \geq d_n$. 则 T 的特征值与 $D + \alpha uu^{\mathsf{T}}$ 的特征值相同

考虑 $D+\alpha uu^{\mathsf{T}}$ 的特征值

设 λ 是 $D+\alpha uu^{\mathsf{T}}$ 的一个特征值, 若 $D-\lambda I$ 非奇异, 则

$$\det(D+\alpha uu^{\mathsf{T}}-\lambda I)=\det(D-\lambda I)\cdot\det(I+\alpha(D-\lambda I)^{-1}uu^{\mathsf{T}}).$$

故 $\det(I+\alpha(D-\lambda I)^{-1}uu^{\mathsf{T}})=0$.

引理 设 $x,y\in\mathbb{R}^n$, 则 $\det(I+xy^{\mathsf{T}})=1+y^{\mathsf{T}}x$.

于是

$$\det(I+\alpha(D-\lambda I)^{-1}uu^{\mathsf{T}})=1+\alpha u^{\mathsf{T}}(D-\lambda I)^{-1}u=1+\alpha\sum_{i=1}^{n}\frac{u_i^2}{d_i-\lambda}\triangleq f(\lambda)$$

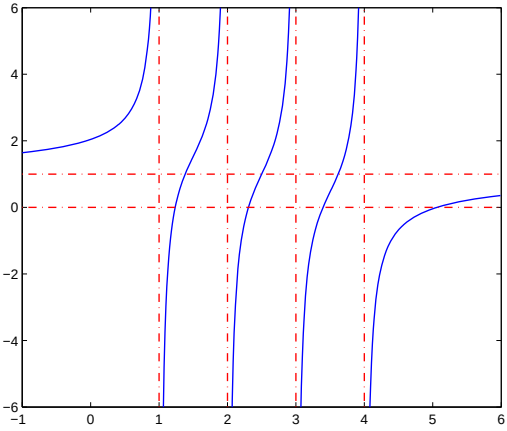
故求 A 的特征值等价于求特征方程 $f(\lambda)=0$ 的根.

由于

$$f'(\lambda) = \alpha \sum_{i=1}^{n} \frac{u_i^2}{(d_i - \lambda)^2}.$$

若 d_i 都互不相同, 且 u_i 都不为零, 则 $f(\lambda)$ 在 $\lambda \neq d_i$ 处都严格单调.

$f(\lambda)$ 在每个区间 (d_{i+1}, d_i) 内都有一个根, 共 $n-1$ 个, 另一个根在 (d_1, ∞) (若 $\alpha > 0$) 或 $(-\infty, d_n)$ (若 $\alpha < 0$) 中.



(**例**: $\alpha = 0.5, d_i = 4, 3, 2, 1, u_i = 1$)

特征值

- 由于 $f(\lambda)$ 在每个区间 (d_{i+1}, d_i) 内光滑且严格单调递增 $(\alpha > 0)$ 或递减 $(\alpha < 0)$, 所以在实际计算中, 可以使用对分法, 牛顿法及其变形, 或有理逼近等算法来求解. 通常都能很快收敛, 一般只需迭代几步即可.

- 计算一个特征值的运算量约为 $\mathcal{O}(n)$, 计算所有特征值约为 $\mathcal{O}(n^2)$.

特征向量

引理 设 $D \in \mathbb{R}^{n\times n}$ 为对角矩阵, $u \in \mathbb{R}^n$, $\alpha \in \mathbb{R}$, 若 λ 是 $D + \alpha uu^{\mathsf{T}}$ 的特征值, 且 $\lambda \neq d_i$, $i = 1, 2, \ldots, n$, 则 $(D - \lambda I)^{-1}u$ 是其对应的特征向量. (板书)

算法 5 计算对称三对角矩阵的特征值和特征向量的分而治之法

1: function $[Q, \Lambda]$ = dc_eig(T) % $T = Q\Lambda Q^{\mathsf{T}}$
2: **if** T is of 1×1 **then**
3: $Q = 1, \Lambda = T$, return
4: **end if**
5: form $T = \begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix} + b_m v v^{\mathsf{T}}$
6: $[Q_1, \Lambda_1]$ = dc_eig(T_1), $[Q_2, \Lambda_2]$ = dc_eig(T_2)
7: form $D + \alpha u u^{\mathsf{T}}$ from $\Lambda_1, \Lambda_2, Q_1, Q_2$
8: compute the eigenvalues Λ and eigenvectors $\hat{Q}$ of $D + \alpha u u^{\mathsf{T}}$
9: compute the eigenvectors of T with $Q = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} \hat{Q}$
10: end

✍ **注: 在分而治之法中, 特征值和特征向量是同时计算的.**

分而治之法的实施细节

下面我们详细讨论分而治之算法的几个细节问题:

(1) 如何减小运算量;

(2) 如何求解特征方程 $f(\lambda) = 0$;

(3) 如何稳定地计算特征向量.

(1) 如何减小运算量 — 收缩技巧 (deflation)

分而治之算法的计算复杂性分析如下: 用 $t(n)$ 表示对 n 阶矩阵调用函数 `dc_eig` 的运算量, 则

$$
\begin{aligned}
t(n) = \; & 2\,t(n/2) && \text{递归调用 } \texttt{dc\_eig} \text{ 两次} \\
& + \mathcal{O}(n^2) && \text{计算 } D + \alpha u u^{\mathsf{T}} \text{ 的特征值和特征向量} \\
& + c \cdot n^3 && \text{计算 } Q.
\end{aligned}
$$

如果计算 Q 时使用的是稠密矩阵乘法, 则 $c = 2$; 若不计 $\mathcal{O}(n^2)$ 项, 则由递归公式 $t(n) = 2\,t(n/2) + c \cdot n^3$ 可得 $t(n) \approx c \cdot 4n^3/3$.

✍ 事实上, 由于**收缩 (deflation)** 现象的存在, 常数 c 通常比 1 小得多.

收缩现象

在算法描述过程中, 假定 d_i 互不相等且 u_i 不能为零. 事实上, 若 $d_i = d_{i+1}$ 或 $u_i = 0$, 则 d_i 即为特征值, 这种现象我们称为**收缩** (deflation)

在实际计算时, 当 $d_i - d_{i+1}$ 或 $|u_i|$ 小于一个给定的阈值时, 我们就近似认为 d_i 为特征值 (即出现收缩现象). **收缩现象会经常发生, 而且非常频繁**.

简化计算

计算量集中在 Q 的计算 (矩阵乘积), 若 $u_i = 0$, 则 d_i 为特征值, 对应特征向量为 e_i, 即 $\hat{Q}$ 的第 i 列为 e_i, 故 Q 的第 i 列不需要做任何的计算.

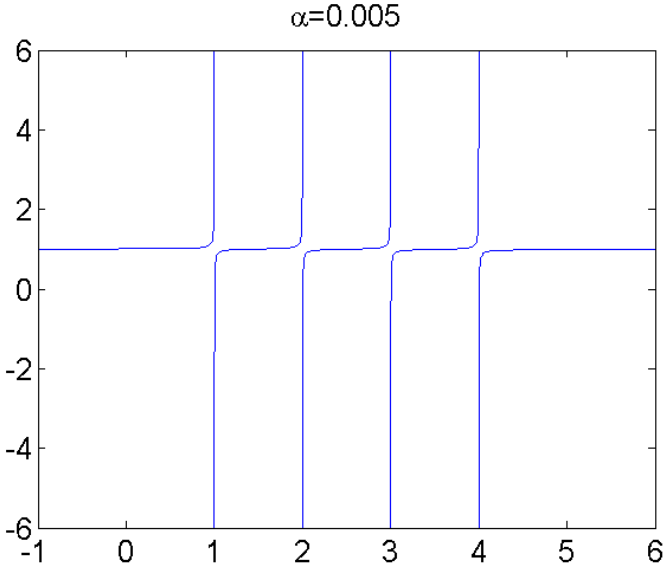
当 $d_i = d_{i+1}$ 时, 也存在一个类似的简化.

(2) 特征方程求解

当 $d_i \neq d_{i+1}$ 且 $u_i \neq 0$ 时, 用牛顿法计算 $f(\lambda)$ 在 (d_{i+1}, d_i) 中的零点 λ_i.

如果 $|u_i|$ 小于给定的阈值时, 我们可直接将 d_i 作为特征值 λ_i 的一个近似.

但当 u_i 很小, 却大于给定的阈值时, 此时 $f(\lambda)$ 在区间 $[d_{i+1}, d_i]$ 中的大部分处的斜率几乎为 0 (见下图). 这时, 如果任取 $[d_{i+1}, d_i]$ 中的一个点作为迭代初始点, 经过一次牛顿迭代后, 迭代解可能会跑到区间 $[d_{i+1}, d_i]$ 的外面, 造成不收敛.



$f(\lambda) = 1 + 0.005\left(\frac{1}{4-\lambda} + \frac{1}{3-\lambda} + \frac{1}{2-\lambda} + \frac{1}{1-\lambda}\right)$ 的图像

迭代法计算近似解

记当前的近似值为 $\tilde{\lambda}$.

牛顿法: 用 $f(\lambda)$ 在 $\tilde{\lambda}$ 处的切线来近似 $f(\lambda)$, 切线的零点为下一个近似值

修正牛顿法: 寻找其它**简单函数** $h(\lambda)$ 来近似 $f(\lambda)$, 然后用 $h(\lambda)$ 的零点作为 $f(\lambda)$ 零点的近似, 并不断迭代下去, 直至收敛.

近似函数的要求

$h(\lambda)$ 需要满足一定的要求 (定义在区间 $[d_{i+1}, d_i]$ 上):

(1) 必须容易构造;

(2) 其零点容易计算;

(3) 尽可能与 $f(\lambda)$ 相近.

修正的牛顿法

因为 d_i 和 d_{i+1} 是 $f(\lambda)$ 的奇点, 所以我们令

$$h(\lambda) = \frac{c_1}{d_i - \lambda} + \frac{c_2}{d_{i+1} - \lambda} + c_3,$$

其中 c_1, c_2, c_3 为参数.

- 显然, $h(\lambda)$ 的零点很容易计算, 与牛顿法相差无几.
- 在选取这些参数时, 要使得 $h(\lambda)$ 在 $\tilde{\lambda}$ 附近尽可能地接近 $f(\lambda)$. 记

$$f(\lambda) = 1 + \alpha \sum_{k=1}^{n} \frac{u_k^2}{d_k - \lambda} = 1 + \alpha \left(\sum_{k=1}^{i} \frac{u_k^2}{d_k - \lambda} + \sum_{k=i+1}^{n} \frac{u_k^2}{d_k - \lambda} \right)$$
$$\triangleq 1 + \alpha\big(\Psi_1(\lambda) + \Psi_2(\lambda)\big).$$

当 $\lambda \in (d_{i+1}, d_i)$ 时, $\Psi_1(\lambda)$ 为正项的和, $\Psi_2(\lambda)$ 为负项的和, 因此它们都可以较精确地计算. 但如果把它们加在一起时可能会引起对消, 从而失去相对精度. 因此我们也将 $h(\lambda)$ 写成

$$h(\lambda) = 1 + \alpha\big(h_1(\lambda) + h_2(\lambda)\big),$$

其中

$$h_1(\lambda) = \frac{c_1}{d_i - \lambda} + \hat{c}_1, \quad h_2(\lambda) = \frac{c_2}{d_{i+1} - \lambda} + \hat{c}_2$$

满足

$$h_1(\tilde{\lambda}) = \Psi_1(\tilde{\lambda}), \quad h_1'(\tilde{\lambda}) = \Psi_1'(\tilde{\lambda}),$$
$$h_2(\tilde{\lambda}) = \Psi_2(\tilde{\lambda}), \quad h_2'(\tilde{\lambda}) = \Psi_2'(\tilde{\lambda}).$$

即 $h_1(\lambda)$ 和 $h_2(\lambda)$ 分别在点 $\tilde{\lambda}$ 与 $\Psi_1(\lambda)$ 和 $\Psi_2(\lambda)$ 相切 $\quad \rightarrow \quad$ 数值插值

直接计算可得

$$\begin{cases} c_1 = \Psi_1'(\tilde{\lambda})(d_i - \tilde{\lambda})^2, \quad \hat{c}_1 = \Psi_1(\tilde{\lambda}) - \Psi_1'(\tilde{\lambda})(d_i - \tilde{\lambda}), \\ c_2 = \Psi_2'(\tilde{\lambda})(d_{i+1} - \tilde{\lambda})^2, \quad \hat{c}_2 = \Psi_2(\tilde{\lambda}) - \Psi_2'(\tilde{\lambda})(d_{i+1} - \tilde{\lambda}). \end{cases} \tag{5.1}$$

所以, 最后取

$$h(\lambda) = 1 + \alpha(\hat{c}_1 + \hat{c}_2) + \alpha\left(\frac{c_1}{d_i - \lambda} + \frac{c_2}{d_{i+1} - \lambda}\right). \tag{5.2}$$

算法 6 修正的 Newton 算法

1: set $k = 0$
2: choose an initial guess $\lambda_0 \in [d_{i+1}, d_i]$
3: **while** not convergence **do**
4: let $\tilde{\lambda} = \lambda_k$ and compute $c_1, c_2, \hat{c}_1, \hat{c}_2$ from (5.1)
5: set $k = k + 1$
6: compute the solution λ_k of $h(\lambda)$ defined by (5.2)
7: **end while**

(3) 计算特征向量的稳定算法

- 设 λ_i 是 $D+\alpha uu^{\mathsf{T}}$ 的特征值, 可利用公式 $(D-\lambda_i I)^{-1}u$ 计算特征向量
- 但当相邻的两个特征值非常接近时, 这个公式可能不稳定.

若 λ_i 与 λ_{i+1} 非常接近, 则它们都非常靠近 d_{i+1} (这里假定 $\lambda_i \in (d_{i+1}, d_i)$, $\lambda_{i+1} \in (d_{i+2}, d_{i+1})$), 在计算 $d_{i+1}-\lambda_i$ 和 $d_{i+1}-\lambda_{i+1}$ 时会存在对消, 这就可能损失有效数字, 产生较大的相对误差, 从而导致 $(D-\lambda_i I)^{-1}u$ 与 $(D-\lambda_{i+1}I)^{-1}u$ 的计算是不准确的, 正交性也会失去.

定理 (Löwner) 设对角阵 $D = \mathrm{diag}(d_1, d_2, \ldots, d_n)$ 满足 $d_1 > d_2 > \cdots > d_n$. 若矩阵 $\hat{D} = D + \hat{u}\hat{u}^{\mathsf{T}}$ 的特征值 $\lambda_1, \lambda_2, \ldots, \lambda_n$ 满足交错性质

$$\lambda_1 > d_1 > \lambda_2 > d_2 > \cdots > \lambda_n > d_n,$$

则向量 $\hat{u}$ 的分量满足

$$|\hat{u}_i| = \left(\frac{\prod_{k=1}^{n} (\lambda_k - d_i)}{\prod_{k=1,\, k\neq i}^{n} (d_k - d_i)} \right)^{1/2}.$$

(留作课外自习)

箭型分而治之法

- 1995 年才由 Gu 和 Eisenstat 给出了一种快速稳定的实现方式, 称为**箭型分而治之法** (**Arrowhead Divide-and-Conquer** , ADC).
- 他们做了大量的数值试验, 采用修正的有理逼近法求解特征方程.
- 数值结果表明, ADC 算法的计算精度可以与其他算法媲美, 而计算速度通常比对称 QR 迭代快 5 至 10 倍, 比 Cuppen 的分而治之法快 2 倍.

Gu and Eisenstat, A Divide-and-Conquer algorithm for the symmetric tridiagonal eigenproblem, SIMAX, 1995.

Gu and Eisenstat, A Divide-and-Conquer algorithm for the bidiagonal SVD, SIMAX, 1995.

5 对分法和反迭代

对分法的基本思想是利用惯性定理来计算所需的部分特征值.

定义 设 A 为对称矩阵, 则其惯性定义为

$$\text{Inertia}(A) = (\nu, \zeta, \pi)$$

其中 ν, ζ, π 分别表示 A 的负特征值, 零特征值和正特征值的个数.

定理 (Sylvester 惯性定理, 合同变换) 设 $A \in \mathbb{R}^{n\times n}$ 是对称矩阵, $X \in \mathbb{R}^{n\times n}$ 非奇异, 则 $X^{\mathsf{T}}AX$ 与 A 有相同的惯性.

$$A - zI = LDL^{\mathsf{T}} \quad \Longrightarrow \quad \text{Inertia}(A - zI) = \text{Inertia}(D)$$

记 $\text{Negcount}(A, \alpha)$ 为小于 α 的 A 的特征值的个数, 即

$$\text{Negcount}(A, \alpha) = A - \alpha I \quad \text{的负惯性指数.}$$

则

$$\text{Negcount}(A, \alpha_2) - \text{Negcount}(A, \alpha_1)$$

为 A 在区间 $[\alpha_1, \alpha_2)$ 中的特征值个数.

如果 $\alpha_2 - \alpha_1 < tol$ (事先给定的阈值), 且 A 在 $[\alpha_1, \alpha_2)$ 中有特征值, 则将 $[\alpha_1, \alpha_2)$ 中的任意一个值作为 A 在该区间中的特征值的近似.

算法 7 对分法: 计算 A 在 $[a,b)$ 中的所有特征值

1: Let tol be a given threshold
2: compute $n_a = \text{Negcount}(A, a)$
3: compute $n_b = \text{Negcount}(A, b)$
4: **if** $n_a = n_b$ **then**
5: return % 此时 $[a,b)$ 中没有 A 的特征值
6: **end if**
7: put (a, n_a, b, n_b) onto $worklist$
8: % $worklist$ 中的元素是“四元素对”, 即由四个数组成的数对
9: **while** $worklist$ not empty **do**
10: remove $(low, n_{low}, up, n_{up})$ from the $worklist$
11: % $(low, n_{low}, up, n_{up})$ 是 $worklist$ 中的任意一个元素
12: **if** $(up - low) < tol$ **then**

print "There are $n_{up} - n_{low}$ eigenvalues in [low, up)"
else
compute $mid = (low + up)/2$
compute $n_{mid} =$ Negcount(A, mid)
if $(n_{mid} > n_{low})$ **then**
put $(low, n_{low}, mid, n_{mid})$ onto $worklist$
end if
if $(n_{up} > n_{mid})$ **then**
put $(mid, n_{mid}, up, n_{up})$ onto $worklist$
end if
end if
end while

对分法的主要运算量集中在计算 Negcount(A, α). 通常是事先将 A 转化成对称三对角矩阵, 这样计算 $A - zI$ 的 $\mathrm{LDL}^{\mathrm{T}}$ 分解就非常简单.

注记

- 单独调用一次 Negcount 的运算量为 $4n$, 故计算 k 个特征值的总运算量约为 $\mathcal{O}(kn)$;
- 当特征值计算出来后, 我们可以使用带位移的**反迭代方法**来计算对应的特征向量. 通常只需迭代 1 至 2 次即可, 由于 A 是三对角矩阵, 故计算每个特征向量的运算量为 $\mathcal{O}(n)$;
- 当特征值紧靠在一起时, 计算出来的特征向量可能会失去正交性, 此时需要进行**再正交化**, 可通过 MGS 的 QR 分解来实现.

6 奇异值分解

稠密矩阵 SVD 算法

大致可分为两类: 基于 Jacobi 旋转和基于二对角化.

基本思想

- 转化为对称特征值问题: $A^{\mathsf{T}}A$, AA^{T} 或 $\begin{bmatrix} 0 & A^{\mathsf{T}} \\ A & 0 \end{bmatrix}$
- 实际计算中需要利用特殊结构或技术使得算法更加有效和准确.

基于二对角化的 SVD 算法基本步骤

(1) **二对角化**: $B = U_1^{\mathsf{T}} A V_1$, 其中 B 为上二对角矩阵, U_1, V_1 为正交阵;

(2) **计算 SVD**: $B = U_2 \Sigma V_2^{\mathsf{T}}$, 其中 Σ 为对角阵, U_2, V_2 为正交阵;

(3) 合并得到 A 的 SVD: $A = U_1 B V_1^{\mathsf{T}} = (U_1 U_2)\Sigma(V_1 V_2)^{\mathsf{T}}$.

6.1 二对角化

通过 Householder 变换, 将 A 转化为二对角矩阵:

$$U_1^{\mathsf{T}} A V_1 = B, \tag{5.3}$$

其中 B 是一个实 (上) 二对角矩阵, U_1 和 V_1 是正交矩阵.

注记

与上 Hessenberg 化和对称三对角化不同, A 与 B 是不相似的.

二对角化过程

- 确定 Householder 矩阵 H_1 和 $\tilde{H}_1$, 处理 A 的第一列和第一行:

$$H_1 A = \begin{bmatrix} * & * & * & \cdots & * \\ 0 & * & * & \cdots & * \\ 0 & * & * & \cdots & * \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & * & * & \cdots & * \end{bmatrix}, \quad H_1 A \begin{bmatrix} 1 & 0 \\ 0 & \tilde{H}_1 \end{bmatrix} = \begin{bmatrix} * & * & 0 & \cdots & 0 \\ 0 & * & * & \cdots & * \\ 0 & * & * & \cdots & * \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & * & * & \cdots & * \end{bmatrix}.$$

- 重复上面的过程, 直到把 A 最终化为二对角矩阵.

  整个二对角化过程的运算量约为 $4mn^2 + 4m^2n - 4n^3/3$, 若不需要计算 U_1 和 V_1, 则运算量约为 $4mn^2 - 4n^3/3$.

二对角矩阵的奇异值分解

设 $B=\begin{bmatrix} a_1 & b_1 & & \\ & \ddots & \ddots & \\ & & \ddots & b_{n-1} \\ & & & a_n \end{bmatrix}$

- 令 $T=\begin{bmatrix} 0 & B^{\mathsf{T}} \\ B & 0 \end{bmatrix}$, 存在置换阵 P, 使得 $P^{\mathsf{T}}AP$ 对称三对角

- $T_{BB^{\mathsf{T}}}=\begin{bmatrix} a_1^2+b_1^2 & a_2b_1 & & \\ a_2b_1 & \ddots & \ddots & \\ & \ddots & a_{n-1}^2+b_{n-1}^2 & a_nb_{n-1} \\ & & a_nb_{n-1} & a_n^2 \end{bmatrix}$, $T_{B^{\mathsf{T}}B}=\begin{bmatrix} a_1^2 & a_1b_1 & & \\ a_1b_1 & a_2^2+b_1^2 & \ddots & \\ & \ddots & \ddots & a_{n-1}b_{n-1} \\ & & a_{n-1}b_{n-1} & a_n^2+b_{n-1}^2 \end{bmatrix}$

注记

理论上, 我们可以直接使用 QR 迭代方法、分而治之法等, 计算 $P^{\mathsf{T}}AP$, $T_{BB^{\mathsf{T}}}$ 或 $T_{B^{\mathsf{T}}B}$ 的特征值和特征向量. 但一般来说, 这种做法并不是最佳的:

(1) 对 $P^{\mathsf{T}}AP$ 做 QR 迭代并不划算, 因为 QR 迭代计算所有的特征值和特征向量, 而事实上只要计算正的特征值即可;

(2) 直接构成 $T_{BB^{\mathsf{T}}}$ 或 $T_{B^{\mathsf{T}}B}$ 是数值不稳定的. 事实上, 这样做可能会使得 B 的小奇异值的精度丢失一半.

常见 SVD 算法

- Jacobi **迭代法**: 隐式地对 AA^{T} 或 $A^{\mathsf{T}}A$ 实施对称 Jacobi 迭代
- Golub-Kahan SVD **算法**: 将带位移隐式对称 QR 迭代算法用到 $B^{\mathsf{T}}B$ 上
- **分而治之法**: 将 ADC 用在二对角矩阵上
- **对分法和反迭代**: 用于计算某个区间内的奇异值及对应的奇异向量.
- dqds **算法**: LR 算法的变形, 与 QR 算法类似, 是 LAPACK 中计算二对角矩阵奇异值的标准算法.
- MRRR **算法**: Multiple Relatively Robust Representations

J. Dongarra, et al., The Singular Value Decomposition: Anatomy of Optimizing an Algorithm for Extreme Scale, SIAM Review, 60 (2018), 808–865.

6.2 Golub-Kahan SVD 算法

Golub-Kahan SVD 算法有时也简称 SVD 算法.

算法基本框架

- 将矩阵 A 二对角化, 得到上二对角矩阵 B;
- 用隐式 QR 迭代计算 $B^{\mathsf{T}}B$ 的特征值分解, 即

$$B^{\mathsf{T}}B = Q\Lambda Q^{\mathsf{T}}, \quad \Lambda = \mathrm{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2).$$

- 计算 BQ 的列主元 QR 分解, 即

$$(BQ)P = UR,$$

其中 P 是置换矩阵, U 是正交矩阵, R 是上三角矩阵.

又

$$(BQ)^{\mathsf{T}}BQ = \Lambda,$$

因此 BQ 是列正交矩阵 (但不是单位列正交).

于是 $R = U^{\mathsf{T}}(BQ)P$ 也是列正交矩阵. 又 R 是上三角矩阵, 所以 R 必定是对角矩阵.

令 $V = QP$, 则可得

$$U^{\mathsf{T}}BV = R$$

这就是二对角矩阵 B 的奇异值分解.

☞ 算法的具体实现可参见相关文献.

QR 算法与 DC 算法比较 (SIAM Review, 60 (2018), 808–865)

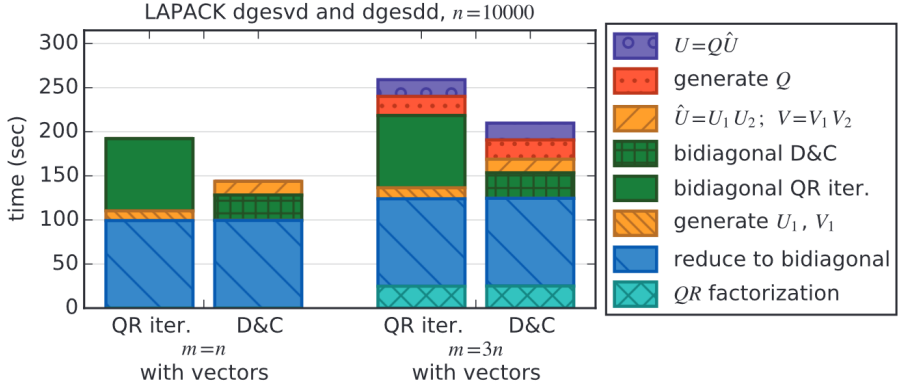


Fig. 11 *Profile comparing LAPACK QR iteration (`dgesvd`) and D&C (`dgesdd`) algorithms. For QR iteration, it generates U_1 and V_1, then updates them with U_2 and V_2 during QR iteration. D&C generates U_2 and V_2, then multiplies $\hat{U} = U_1U_2$ and $V = V_1V_2$ afterwards.*

6.3 dqds 算法

dqds 算法基于下面的 LR 算法.

算法 8 带位移的 LR 算法

Let T_0 be a given real symmetric positive definite matrix
set $i = 0$
while not converge **do**
    choose a shift τ_i^2 satisfying $\tau_i^2 < \min\{\lambda(T_i)\}$
    compute B_i such that $T_i - \tau_i^2 I = B_i^{\mathsf{T}} B_i$ % Cholesky factorization
    $T_{i+1} = B_i B_i^{\mathsf{T}} + \tau_i^2 I$
    $i = i + 1$
end while

LR 迭代算法与 QR 迭代算法

LR 迭代算法在形式上与 QR 迭代算法非常类似.

事实上, 对于不带位移的 LR 迭代算法, 两步 LR 迭代等价于一步 QR 迭代.

引理 设 $\tilde{T}$ 是不带位移的 LR 算法迭代两步后生成的矩阵, $\hat{T}$ 是不带位移的 QR 算法迭代一步后生成的矩阵, 则 $\tilde{T} = \hat{T}$.

(1) LR 算法中要求 T_0 对称正定, 但并不一定是三对角矩阵;

(2) 由该引理可知, QR 算法与 LR 算法有相同的收敛性.

dqds 算法

在数学上, dqds 算法与 LR 算法是等价的, 但在该算法中, 我们是直接通过 B_i 来计算 B_{i+1}, 从而避免计算中间矩阵 T_{i+1}, 这样也就尽可能地避免了由于计算 $B_iB_i^{\mathsf{T}}$ 而可能带来的数值不稳定性.

下面推导如何从 B_i 直接计算 B_{i+1}. 设

$$B_i = \begin{bmatrix} a_1 & b_1 & & \\ & a_2 & \ddots & \\ & & \ddots & b_{n-1} \\ & & & a_n \end{bmatrix}, \quad B_{i+1} = \begin{bmatrix} \tilde{a}_1 & \tilde{b}_1 & & \\ & \tilde{a}_2 & \ddots & \\ & & \ddots & \tilde{b}_{n-1} \\ & & & \tilde{a}_n \end{bmatrix}.$$

为了书写方便, 我们记 $b_0 = b_n = \tilde{b}_0 = \tilde{b}_n = 0$. 由 LR 算法 8 可知

$$B_{i+1}^{\mathsf{T}}B_{i+1} + \tau_{i+1}^2 I = B_iB_i^{\mathsf{T}} + \tau_i^2 I.$$

比较等式两边矩阵的对角线和上对角线元素, 可得

$$\tilde{a}_k^2 + \tilde{b}_{k-1}^2 + \tau_{i+1}^2 = a_k^2 + b_k^2 + \tau_i^2, \quad k = 1, 2, \ldots, n$$

$$\tilde{a}_k \tilde{b}_k = a_{k+1} b_k \quad \text{或} \quad \tilde{a}_k^2 \tilde{b}_k^2 = a_{k+1}^2 b_k^2, \quad k = 1, 2, \ldots, n-1.$$

记 $\delta = \tau_{i+1}^2 - \tau_i^2$, $p_k = a_k^2$, $q_k = b_k^2$, $\tilde{p}_k = \tilde{a}_k^2$, $\tilde{q}_k = \tilde{b}_k^2$, 则可得 **qds 算法**:

算法 9 qds 算法的单步 ($B_i \to B_{i+1}$)

1: $\delta = \tau_{i+1}^2 - \tau_i^2$

2: **for** $k = 1$ to $n-1$ **do**

3: $\quad \tilde{p}_k = p_k + q_k - \tilde{q}_{k-1} - \delta$

4: $\quad \tilde{q}_k = q_k (p_{k+1} / \tilde{p}_k)$

5: **end for**

6: $\tilde{p}_n = p_n - \tilde{q}_{n-1} - \delta$

注记

- qds 算法中的每个循环仅需 5 个浮点运算, 所以运算量较少.
- 为了提高算法精度, 我们引入一个辅助变量 $d_k \triangleq p_k - \tilde{q}_{k-1} - \delta$, 则

$$\begin{aligned} d_k &= p_k - \tilde{q}_{k-1} - \delta \\ &= p_k - \frac{q_{k-1}p_k}{\tilde{p}_{k-1}} - \delta \\ &= p_k \cdot \frac{\tilde{p}_{k-1} - q_{k-1}}{\tilde{p}_{k-1}} - \delta \\ &= p_k \cdot \frac{p_{k-1} - \tilde{q}_{k-2} - \delta}{\tilde{p}_{k-1}} - \delta \\ &= \frac{p_k}{\tilde{p}_{k-1}} \cdot d_{k-1} - \delta. \end{aligned}$$

算法 10 dqds 算法的单步 ($B_i \to B_{i+1}$)

1: $\delta = \tau_{i+1}^2 - \tau_i^2$

2: $d_1 = p_1 - \delta$

3: **for** $k = 1$ to $n-1$ **do**

4: $\quad \tilde{p}_k = d_k + q_k$

5: $\quad t = p_{k+1}/\tilde{p}_k$

6: $\quad \tilde{q}_j = q_k \cdot t$

7: $\quad d_{k+1} = d_k \cdot t - \delta$

8: **end for**

9: $\tilde{p}_n = d_n$

dqds 算法的运算量与 dqs 差不多, 但更精确 (详情请参见相关文献).

6.4 Jacobi 算法

本节讨论对矩阵 $M = A^{\mathsf{T}}A$ 实施隐式的 Jacobi 算法来计算 A 的奇异值.

我们知道, Jaboci 算法的每一步就是对矩阵作 Jacobi 旋转, 即 $A^{\mathsf{T}}A \to J^{\mathsf{T}}A^{\mathsf{T}}AJ$, 其中 J 的选取将某两个非对角元化为 0.

在实际计算中, 我们只需计算 AJ, 故该算法称为**单边 Jacobi 旋转**.

算法 11 单边 Jacobi 旋转的单步

% 对 $M = A^{\mathsf{T}}A$ 作 Jacobi 旋转, 将 $M(i,j), M(j,i)$ 化为 0

Compute $m_{ii} = (A^{\mathsf{T}}A)_{ii}$, $m_{ij} = (A^{\mathsf{T}}A)_{ij}$, $m_{jj} = (A^{\mathsf{T}}A)_{jj}$
if m_{ij} is not small enough **then**
    $\tau = (m_{ii} - m_{jj})/(2 \cdot m_{ij})$
    $t = \text{sign}(\tau)/(|\tau| + \sqrt{1+\tau^2})$
    $c = 1/\sqrt{1+t^2}$
    $s = c \cdot t$
    $A = AG(i,j,\theta)$ % $G(i,j,\theta)$ 为 Givens 变换
    if eigenvectors are desired **then**
        $J = J \cdot G(i,j,\theta)$
    end if
end if

在上面算法的基础上, 我们可以给出完整的单边 Jacobi 算法.

算法 12 单边 Jacobi 算法: 计算 $A = U\Sigma V^{\mathsf{T}}$

while $A^{\mathsf{T}}A$ is not diagonal enough **do**
  for $i = 1$ to $n-1$ **do**
    for $j = i+1$ to n **do**
      调用单边 Jacobi 旋转
    end for
  end for
end while
compute $\sigma_i = \|A(:,i)\|_2, i = 1, 2, \ldots n$
$U = [u_1, \ldots, u_n]$ with $u_i = A(:,i)/\sigma_i$
$V = J$

Jacobi 算法的特点

- 不需要双对角化, 这样可以避免双对角化引入的误差;
- 可达到相对较高的计算精度;
- 速度较慢. (目前已有快速的改进算法)

定理 设 $A = DX \in \mathbb{R}^{n\times n}$, 其中 D 为非奇异对角阵, X 非奇异. 设 $\hat{A}$ 是按浮点运算单边 Jacobi 旋转 m 次后所得到的矩阵. 若 A 和 $\hat{A}$ 的奇异值分别为 $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$ 和 $\hat{\sigma}_1 \geq \hat{\sigma}_2 \geq \ldots \geq \hat{\sigma}_n$, 则

$$\frac{\left|\hat{\sigma}_i - \sigma_i\right|}{\sigma_i} \leq \mathcal{O}(m\varepsilon)\kappa(X).$$

故 X 的条件数越小, 计算矩阵 A 的奇异值时相对误差越小.

7 扰动分析

设 $A \in \mathbb{R}^{n\times n}$ 是对称矩阵, 则有下面的谱分解.

定理 设 $A \in \mathbb{R}^{n\times n}$ 是对称矩阵, 则存在一个正交矩阵 Q 使得

$$A = Q\Lambda Q^{\mathsf{T}}$$

其中 $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ 是一个实对角矩阵.

❶ 假定 $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$.

❷ 记 $Q = [q_1, q_2, \ldots, q_n]$, 即 q_i 就是 λ_i 对应的单位正交特征向量.

7.1 特征值与 Rayleigh 商

定义 设 $A \in \mathbb{R}^{n\times n}$ 是对称矩阵, 向量 $x \in \mathbb{R}^n$ 非零, 则 x 关于 A 的 **Rayleigh 商**定义为:

$$\rho(x, A) = \frac{x^{\mathsf{T}} A x}{x^{\mathsf{T}} x}.$$

有时简记为 $\rho(x)$.

Rayleigh 商基本性质

(1) $\rho(\alpha x) = \rho(x), \forall \alpha \in \mathbb{R}, \alpha \neq 0$;

(2) $\rho(q_i) = \lambda_i, i = 1, 2, \ldots, n$;

(3) 设 $x = \alpha_1 q_1 + \alpha_2 q_2 + \cdots + \alpha_n q_n$, 则 $\rho(x) = \dfrac{\alpha_1^2\lambda_1 + \alpha_2^2\lambda_2 + \cdots + \alpha_n^2\lambda_n}{\alpha_1^2 + \alpha_2^2 + \cdots + \alpha_n^2}$;

(4) $\lambda_n \leq \rho(x) \leq \lambda_1, |\rho(x)| \leq \|A\|_2$.

Courant-Fischer 极小极大定理

实对称矩阵的特征值与 Rayleigh 商之间的一个基本性质是 Courant-Fischer 极小极大定理.

定理 (Courant-Fischer) 设 $A \in \mathbb{R}^{n\times n}$ 是对称矩阵, 其特征值为 $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$, 则有

$$\lambda_k = \max_{\mathbb{U}\in\mathbb{S}_k^n} \min_{x\in\mathbb{U},\, x\neq 0} \frac{x^{\mathsf{T}}Ax}{x^{\mathsf{T}}x} = \min_{\mathbb{V}\in\mathbb{S}_{n-k+1}^n} \max_{x\in\mathbb{V},\, x\neq 0} \frac{x^{\mathsf{T}}Ax}{x^{\mathsf{T}}x},$$

其中 $\mathbb{S}_i^n$ 表示 $\mathbb{R}^n$ 中所有 i 维子空间构成的集合. 当

$$\mathbb{U} = \operatorname{span}\{q_1, \ldots, q_k\}, \quad \mathbb{V} = \operatorname{span}\{q_k, \ldots, q_n\}, \quad x = q_k$$

时, 上式中的等号成立. (板书)

Rayleigh-Ritz 定理

当 $k=1$ 和 $k=n$ 时, 就可以得到下面的定理.

定理 (Rayleigh-Ritz) 设 $A \in \mathbb{R}^{n\times n}$ 是对称矩阵, 其特征值为 $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$, 则有

$$\lambda_1 = \max_{x\in\mathbb{R}^n,\, x\neq 0} \frac{x^{\mathsf{T}}Ax}{x^{\mathsf{T}}x}, \quad \lambda_n = \min_{x\in\mathbb{R}^n,\, x\neq 0} \frac{x^{\mathsf{T}}Ax}{x^{\mathsf{T}}x}.$$

特征值分隔定理

由极小极大定理, 我们可以得到下面的特征值分隔定理.

定理 (分隔定理) 设 $A \in \mathbb{R}^{n\times n}$ 是对称矩阵, $B = Q^{\mathsf{T}}AQ$, 其中 $Q \in \mathbb{R}^{n\times(n-1)}$ 满足 $Q^{\mathsf{T}}Q = I_{n-1}$. 再设 A 和 B 的特征值分别为

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \quad \text{和} \quad \tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \cdots \geq \tilde{\lambda}_{n-1},$$

则有

$$\lambda_1 \geq \tilde{\lambda}_1 \geq \lambda_2 \geq \tilde{\lambda}_2 \cdots \geq \tilde{\lambda}_{n-1} \geq \lambda_n.$$

特别地, 在分隔定理中取 $Q = [e_1, \ldots, e_{i-1}, e_{i+1}, \ldots, e_n]$, 则可得下面的结论.

推论 设 $A \in \mathbb{R}^{n\times n}$ 是对称矩阵, $\tilde{A}$ 是 A 的一个 $n-1$ 阶主子矩阵, A 和 $\tilde{A}$ 的特征值分别为

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \quad \text{和} \quad \tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \cdots \geq \tilde{\lambda}_{n-1},$$

则有

$$\lambda_1 \geq \tilde{\lambda}_1 \geq \lambda_2 \geq \tilde{\lambda}_2 \cdots \geq \tilde{\lambda}_{n-1} \geq \lambda_n.$$

反复应用上面的推论, 即可得到下面的结论.

推论 设 $A \in \mathbb{R}^{n\times n}$ 是对称矩阵, $\tilde{A}$ 是 A 的一个 k 阶主子矩阵 $(1 \le k \le n-1)$, A 和 $\tilde{A}$ 的特征值分别为

$$\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_n \quad \text{和} \quad \tilde{\lambda}_1 \ge \tilde{\lambda}_2 \ge \cdots \ge \tilde{\lambda}_k,$$

则有

$$\lambda_i \ge \tilde{\lambda}_i \ge \lambda_{n-k+i}, \quad i = 1, 2, \ldots, k.$$

7.2 对称矩阵特征值的扰动分析

设 $A \in \mathbb{R}^{n\times n}$ 是对称矩阵, 扰动矩阵 $E \in \mathbb{R}^{n\times n}$ 也是对称矩阵, 讨论 $A+E$ 的特征值与 A 的特征值之间的关系.

定理 设 $A \in \mathbb{R}^{n\times n}$ 和 $B = A + E \in \mathbb{R}^{n\times n}$ 都是对称矩阵, 其特征值分别为

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \quad \text{和} \quad \tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \cdots \geq \tilde{\lambda}_n.$$

假定 E 的最大和最小特征值分别为 μ_1 和 μ_n, 则有

$$\lambda_i + \mu_1 \geq \tilde{\lambda}_i \geq \lambda_i + \mu_n, \quad i = 1, 2, \ldots, n.$$

(留作练习)

Weyl 定理

定理 (Weyl) 设 $A \in \mathbb{R}^{n\times n}$ 和 $B = A + E \in \mathbb{R}^{n\times n}$ 都是对称矩阵, 其特征值分别为 $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ 和 $\tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \cdots \geq \tilde{\lambda}_n$, 则

$$|\tilde{\lambda}_j - \lambda_j| \leq \|E\|_2, \quad j = 1, 2, \ldots, n.$$

该定理的结论可以推广到奇异值情形.

定理 设 $A, B \in \mathbb{R}^{m\times n}$ $(m \geq n)$, 它们的奇异值分别为 $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$ 和 $\tilde{\sigma}_1 \geq \tilde{\sigma}_2 \geq \cdots \geq \tilde{\sigma}_n$. 则

$$|\tilde{\sigma}_j - \sigma_j| \leq \|B - A\|_2, \quad j = 1, 2, \ldots, n.$$

7.3 对称矩阵特征向量的扰动

定义 设 $A \in \mathbb{R}^{n\times n}$ 的特征值为 $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$, 则 λ_i 与其余特征值之间的间隙 **(gap)** 定义为

$$\mathrm{gap}(\lambda_i, A) = \min_{j\neq i} |\lambda_j - \lambda_i|.$$

有时简记为 $\mathrm{gap}(\lambda_i)$.

特征向量的敏感性依赖于其对应的特征值的 gap, 一般来说, gap 越小, 特征向量越敏感.

定理 设 $A = Q\Lambda Q^{\mathsf{T}}$ 和 $A + E = \tilde{Q}\tilde{\Lambda}\tilde{Q}^{\mathsf{T}}$ 分别为对称矩阵 $A \in \mathbb{R}^{n\times n}$ 和 $A + E \in \mathbb{R}^{n\times n}$ 的特征值分解, 其中 $Q = [q_1, q_2, \ldots, q_n]$ 和 $\tilde{Q} = [\tilde{q}_1, \tilde{q}_2, \ldots, \tilde{q}_n]$ 均为正交矩阵, 且 $\tilde{q}_i$ 为 q_i 对应的扰动特征向量. 用 θ_i 表示 q_i 和 $\tilde{q}_i$ 之间的锐角, 则当 $\mathrm{gap}(\lambda_i, A) > 0$ 时

$$\frac{1}{2}\sin 2\theta_i \leq \frac{\|E\|_2}{\mathrm{gap}(\lambda_i, A)}.$$

类似地, 当 $\mathrm{gap}(\tilde{\lambda}_i, A + E) > 0$ 时

$$\frac{1}{2}\sin 2\theta_i \leq \frac{\|E\|_2}{\mathrm{gap}(\tilde{\lambda}_i, A + E)}.$$

(留作课外自习)

注记

- 当 $\theta_i \ll 1$ 时, $\frac{1}{2}\sin 2\theta_i \approx \theta_i \approx \sin\theta_i$;
- 若 $\|E\|_2 \geq \frac{1}{2}\mathrm{gap}(\lambda_i, A)$, 则定理中给出的上界就失去实际意义;
- 在该定理中, 没有对特征值进行排序;
- 在实际计算中, 我们通常所知道的是 $\mathrm{gap}(\tilde{\lambda}_i, A+E)$.

7.4 Rayleigh 商逼近

定理 设对称矩阵 $A \in \mathbb{R}^{n\times n}$ 的特征值为 $\lambda_1, \lambda_2, \ldots, \lambda_n$.

(1) 若 $x \in \mathbb{R}^n$ 是单位向量, $\beta \in \mathbb{R}$, 则

$$\min_{1\le i\le n} |\lambda_i - \beta| \le \|Ax - \beta x\|_2; \tag{5.4}$$

(2) 给定非零向量 $x \in \mathbb{R}^n$, 当 $\beta = \rho(x)$ 时, $\|Ax - \beta x\|_2$ 达到最小, 即

$$\min_{\beta\in\mathbb{R}} \|Ax - \beta x\|_2 = \|Ax - \rho(x)x\|_2; \tag{5.5}$$

(3) 令 $r = Ax - \rho(x)x$, 设 λ_i 是离 $\rho(x)$ 最近的特征值, $\mathrm{gap}' = \min\limits_{j\ne i} |\lambda_j - \rho(x)|$, θ 是 x 和 q_i 之间的锐角, 其中 q_i 是 λ_i 对应的单位特征向量, 则

$$\sin\theta \le \frac{\|r\|_2}{\mathrm{gap}'} \quad \text{且} \quad |\lambda_i - \rho(x)| \le \frac{\|r\|_2^2}{\mathrm{gap}'}. \tag{5.6}$$

注记

- 由 (5.4) 可知, 在幂迭代和反迭代中可以使用残量 $\|Ax - \tilde{\lambda}x\|_2 < tol$ 作为停机准则, 这里 $\tilde{\lambda}$ 是迭代过程中计算得到的近似特征值.
- 等式 (5.5) 则解释了为什么用 Rayleigh 商来近似特征值.
- 不等式 (5.6) 表明 $|\lambda_i - \rho(x)|$ 的值与残量范数 $\|r\|_2$ 的平方成正比, 这个结论是 Rayleigh 商迭代局部三次收敛的基础.

7.5 相对扰动分析 *

这里主要讨论 A 和 $X^{\mathsf{T}}AX$ 的特征值和特征向量之间的扰动关系, 其中 X 非奇异且满足 $\|X^{\mathsf{T}}X - I\|_2 = \varepsilon$. 这是因为在计算特征向量时, 由于舍入误差的原因, 最后得到的正交矩阵 Q 会带有误差, 从而失去正交性.

定理 (相对 Weyl 定理) 设对称矩阵 A 和 $X^{\mathsf{T}}AX$ 的特征值分别为 $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ 和 $\tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \cdots \geq \tilde{\lambda}_n$, 令 $\varepsilon = \|X^{\mathsf{T}}X - I\|_2$, 则

$$|\tilde{\lambda}_i - \lambda_i| \leq \varepsilon|\lambda_i| \quad \text{或} \quad \frac{|\tilde{\lambda}_i - \lambda_i|}{|\lambda_i|} \leq \varepsilon \quad (\text{if } \lambda_i \neq 0).$$

(留作课外自习)

当 X 正交时, $\varepsilon = 0$, 故 $X^{\mathsf{T}}AX$ 与 A 有相同的特征值. 当 X 几乎正交时, ε 很小, 此时 $X^{\mathsf{T}}AX$ 与 A 的特征值几乎相同.

推论 设 G 和 $Y^{\mathsf{T}}GX$ 的奇异值分别为 $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$ 和 $\tilde{\sigma}_1 \geq \tilde{\sigma}_2 \geq \cdots \geq \tilde{\sigma}_n$, 令 $\varepsilon = \max\left\{\|X^{\mathsf{T}}X - I\|_2, \|Y^{\mathsf{T}}Y - I\|_2\right\}$, 则

$$|\tilde{\sigma}_i - \sigma_i| \leq \varepsilon|\sigma_i| \quad \text{或} \quad \frac{|\tilde{\sigma}_i - \sigma_i|}{|\sigma_i|} \leq \varepsilon \quad (\text{if } \sigma_i \neq 0).$$

下面给出特征向量的相对扰动性质.

定义 设 $A \in \mathbb{R}^{n\times n}$ 的特征值为 $\lambda_1, \lambda_2, \ldots, \lambda_n$, 若 $\lambda_i \neq 0$, 则 λ_i 与其余特征值之间的相对间隙 **(relative gap)** 定义为

$$\operatorname{relgap}(\lambda_i, A) = \min_{j\neq i} \frac{|\lambda_j - \lambda_i|}{|\lambda_i|}.$$

定理 设 $A \in \mathbb{R}^{n\times n}$ 和 $X^{\mathsf{T}}AX \in \mathbb{R}^{n\times n}$ 的特征值分解分别为 $A = Q\Lambda Q^{\mathsf{T}}$ 和 $X^{\mathsf{T}}AX = \tilde{Q}\tilde{\Lambda}\tilde{Q}^{\mathsf{T}}$, 其中 $Q = [q_1, q_2, \ldots, q_n]$ 和 $\tilde{Q} = [\tilde{q}_1, \tilde{q}_2, \ldots, \tilde{q}_n]$ 均为正交矩阵, $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$, $\tilde{\Lambda} = \mathrm{diag}(\tilde{\lambda}_1, \tilde{\lambda}_2, \ldots, \tilde{\lambda}_n)$ 且 $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$, $\tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \cdots \geq \tilde{\lambda}_n$. 设 θ_i 表示 q_i 和 $\tilde{q}_i$ 之间的锐角, 令 $\varepsilon_1 = \|I - X^{-T}X^{-1}\|_2$, $\varepsilon_2 = \|X - I\|_2$, 若 $\varepsilon_1 < 1$ 且 $\mathrm{relgap}(\tilde{\lambda}_i, X^{\mathsf{T}}AX) > 0$, 则

$$\frac{1}{2}\sin 2\theta_i \leq \frac{\varepsilon_1}{1-\varepsilon_1} \cdot \frac{1}{\mathrm{relgap}(\tilde{\lambda}_i, X^{\mathsf{T}}AX)} + \varepsilon_2.$$

(留作课外自习)


谢 谢
THANK YOU