第9章 特征值与特征向量

陈路 CHENLU.SCIEN@GMAIL.COM
201800301206

本文首先简要总结特征值与特征向量的基础知识，给出相关的定义和定理。进而对求解特征值的幂方法和QR方法进行代码实现，并结合具体题目运行程序，展示不同算法的结果。

1. 特征值与特征向量基础知识

1.1 特征值与特征向量定义

定义 1
如果A是一个$n \times n$实矩阵，则它的n个**特征值**$\lambda_1, \lambda_2, \cdots, \lambda_n$是下面$n$阶特征多项式的实根或复根：

$$p(\lambda) = det(\mathbf{A} - \lambda \mathbf{I}).$$

定义 2
如果λ是$\mathbf{A}$的特征值并且非零向量$\mathbf{V}$具有如下特性：

$$\mathbf{AV} = \lambda \mathbf{V}.$$

1.2 特征值范围估计

定义 3
设$\|X\|$是向量范数，则对应的**自然矩阵范数**为

$$\|A\| = \max_{\|X\|=1} \left\{ \frac{\|AX\|}{\|X\|} \right\}$$

范数$\|A\|_\infty$可表示为

$$\|A\|_\infty = \max_{1 \le i \le n} \left\{ \sum_{j=1}^{n} |a_{ij}| \right\}$$

定理 1
如果λ是矩阵$\mathbf{A}$的任意特征值，则对于所有自然矩阵范数$\|A\|$，有

$$|\lambda| \leq \|A\|$$

定理 2
Gerschgorin圆盘定理:
设A是一个$n \times n$矩阵，C_j表示位于复平面$z = x + iy$上，以a_{jj}为圆心，以

$$r_j = \sum_{k=1, k\neq j}^{n} |a_{jk}|, \quad j = 1, 2, \cdots, n$$

为半径的圆盘，即C_j包含所有满足条件

$$C_j = \{z : |z - a_{jj}| \leq r_j\}$$

的复数$z = x + iy$.
如果$S = \cup_{i=1}^{n} C_i$，则A的所有特征值包含在集合S中.
进一步可得，以上k个圆盘的并如果与其余的$n-k$个圆盘不相交，则它们一定包含k个特征值（含重复）.

定理 3
谱半径定理:
设A是一个对称阵，则A的谱半径定义为$\|A\|_2$，并且满足如下关系

$$\|A\|_2 = \max |\lambda_1|, |\lambda_2|, \cdots, |\lambda_n|.$$

2. 上机实验题

问题 1
已知矩阵

$$\mathbf{A} = \begin{bmatrix} 4 & -1 & 1 \\ -1 & 3 & -2 \\ 1 & -2 & 3 \end{bmatrix}$$

是一个对称矩阵，且其特征值为$\lambda_1 = 6$, $\lambda_2 = 3$, $\lambda_3 = 1$.
分别利用幂法、对称幂法、反幂法求其最大特征值和特征向量.
注意:可取初始向量$\mathbf{x}^{(0)} = (1 \quad 1 \quad 1)^T$.

解:

(1)幂法

代码实现如下:

```
function [lambda, V] = power1 (A, X, epsilon , max1)
% Input  - A   is an nxn matrix
%        - X   is the nx1 starting vector
%        - epsilon   is the tolerance
%        - max1 is the maximum number of iterations
% Output - lambda is the dominant eigenvalue
%        - V is the dominant eigenvector

% Initialization
lambda = 0 ;
cnt = 0 ;
err = 1;
state = 1 ;

while ((cnt<=max1) && (state==1))
  Y = A*X;
  %Normalize Y
  [m j] = max(abs(Y));
  c1 = m;
  dc = abs(lambda - c1) ;
  Y = (1/c1)*Y;
  %Update X and lambda and check for convergence
  dv = norm(X- Y) ;
  err = max(dc,dv) ;
  X = Y;
  lambda = c1;
  state = 0;
  if (err > epsilon)
    state = 1;
  end
  cnt = cnt + 1;
end
V = X;
end
```

运行程序

```
syms A X epsilon max1;
A = [4 -1 1;-1 3 -2;1 -2 3];
X = [1 1 1]';
epsilon = 0.0000000001;
max1 = 100;
[lambda, V] = power1 (A, X, epsilon , max1)
```

求得:

最大特征值为5.999999999912689，对应的特征向量为

$$V = (1.000000000000000 \quad -0.999999999978172 \quad 0.999999999978172)^T$$

(2)对称幂法

代码实现如下:

```
function [lambda,V] = sympower(A, X, epsilon, max1)
% Input  - A   is an nxn matrix
%        - X   is the nx1 starting vector
%        - epsilon   is the tolerance
%        - max1 is the maximum number of iterations
% Output - lambda is the dominant eigenvalue
%        - V is the dominant eigenvector

X = X/norm(X,2);
for k = 2:max1
  Y = A*X;
  u = X'*Y;
  if norm(Y,2) == 0
    break, end
  err = norm((X-Y/norm(Y,2)),2);
  X = Y/norm(Y,2);
  if(err < epsilon)
    break, end
end
lambda = u;
V = X;
end
```

求得:

最大特征值为5.999999999999999，对应的特征向量为

$$V = (0.577350269256838 \quad -0.577350269156020 \quad 0.577350269156020)^T$$

(3)反幂法

代码实现如下:

```
function [lambda,V] = invpower(A, X, alpha, epsilon, max1)
% Input  - A   is an nxn matrix
%        - X   is the nx1 starting vector
%        - alpha    is the given shift
%        - epsilon  is the tolerance
%        - max1 is the maximum number of iterations
% Output - lambda is the dominant eigenvalue
%        - V is the dominant eigenvector

% Initilization
[n n] = size(A);
A = A - alpha * eye(n);
lambda = 0;
cnt = 0;
err = 1;
state = 1;

while ((cnt <= max1) && (state == 1))
```

```
  % solve system AY=X
  Y = A\X;
  % normalizae Y
  [m j] = max(abs(Y));
  c1 = m;
  dc = abs(lambda - c1);
  Y = (1/c1)*Y;
  % update X and lambda and check for convergence
  dv = norm(X - Y);
  err = max(dc, dv);
  X = Y;
  lambda = c1;
  state = 0;
  if (err > epsilon)
    state = 1;
  end
  cnt = cnt+1;
end
lambda = alpha + 1/c1;
V = X;
end
```

运行程序

```
syms A X alpha epsilon max1;
A = [4 -1 1;-1 3 -2;1 -2 3];
X = [1 1 1]';
alpha = 5.8;
epsilon = 0.0000000001;
max1 = 100;
[lambda, V] = power1 (A, X, alpha, epsilon, max1)
```

求得:
最大特征值为6.000000000000043，对应的特征向量为

$$V = (1.000000000000000 \quad -0.999999999999947 \quad 0.999999999999947)^T$$

问题 2
分别利用Householder变换和Givens旋转变化方法求**A**的**QR**分解

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

写出每一步具体求解过程，及最终分解结果.

解:

1. 使用Householder变换求矩阵A的QR分解：

$$\mathbf{A}^{(1)} = \mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Step1: 对于A的第一列列向量

$$\mathbf{x}_1 = \mathbf{a}^{(1)}_{1:4,1} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \quad \|\mathbf{x}_1\|_2 = 2$$

对应的Householder向量为

$$\tilde{\mathbf{v}}_1 = \mathbf{x}_1 + \|\mathbf{x}_1\|_2 \mathbf{e}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + 2 \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

借助此向量，构造Householder反射

$$c = \frac{2}{\tilde{\mathbf{v}}_1^T \tilde{\mathbf{v}}_1} = 1/6, \quad \tilde{H}_1 = I - c\tilde{\mathbf{v}}_1\tilde{\mathbf{v}}_1^T$$

将此反射作用于$A^{(1)}$，得

$$A^{(2)} = \tilde{H}_1 A^{(1)} = \begin{bmatrix} 2 & 1.5 & 1 \\ 0 & -0.866025403784438 & -0.577350269189625 \\ 0 & 1.11022302462516e-16 & -0.816496580927726 \\ 0 & 1.11022302462516e-16 & 1.11022302462516e-16 \end{bmatrix}$$

Step2: 对于$A^{(2)}$的第二列列向量2至4行的分量

$$\mathbf{x}_2 = \mathbf{a}^{(2)}_{2:4,2} = \begin{bmatrix} -0.866025403784438 \\ 1.11022302462516e-16 \\ 1.11022302462516e-16 \end{bmatrix}, \quad \|\mathbf{x}_2\|_2 = 0.866025403784438$$

对应的Householder向量为

$$\tilde{\mathbf{v}}_2 = \mathbf{x}_2 + \|\mathbf{x}_2\|_2 \mathbf{e}_1 = \begin{bmatrix} -0.866025403784438 \\ 1.11022302462516e-16 \\ 1.11022302462516e-16 \end{bmatrix} + 0.866025403784438 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 1.11022302462516e-16 \\ 1.11022302462516e-16 \end{bmatrix}$$

借助此向量，构造Householder反射

$$c = \frac{2}{\tilde{\mathbf{v}}_2^T \tilde{\mathbf{v}}_2} = 8.112963841460618e+31, \quad \tilde{H}_2 = I - c\tilde{\mathbf{v}}_2\tilde{\mathbf{v}}_2^T$$

将此反射作用于$A^{(2)}$，得

$$\tilde{H}_2 A^{(2)} = \begin{bmatrix} 0.866025403784438 & 0.577350269189625 \\ 0 & -0.816496580927726 \\ 0 & 3.70074341541719e-17 \end{bmatrix}$$

$$A^{(3)} = \begin{bmatrix} 2.00000000000000 & 1.50000000000000 & 1.00000000000000 \\ 0 & 0.866025403784438 & 0.577350269189625 \\ 0 & 0 & -0.816496580927726 \\ 0 & 0 & 3.70074341541719e-17 \end{bmatrix}.$$

Step3: 对于$A^{(3)}$的第三列列向量3至4行的分量

$$\mathbf{x}_3 = \mathbf{a}_{3:4,3}^{(3)} = \begin{bmatrix} -0.816496580927726 \\ 3.70074341541719e-17 \end{bmatrix}, \quad \|\mathbf{x}_3\|_2 = 0.816496580927726$$

对应的Householder向量为

$$\tilde{\mathbf{v}}_3 = \mathbf{x}_3 + \|\mathbf{x}_3\|_2 \mathbf{e}_1 = \begin{bmatrix} -0.816496580927726 \\ 3.70074341541719e-17 \end{bmatrix} + 0.816496580927726 \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

借助此向量，构造Householder反射

$$c = \frac{2}{\tilde{\mathbf{v}}_3^T \tilde{\mathbf{v}}_3} = 1/6, \quad \tilde{H}_3 = I - c\tilde{\mathbf{v}}_3\tilde{\mathbf{v}}_3^T$$

将此反射作用于$A^{(3)}$，得

$$\tilde{H}_3 A^{(3)} = \begin{bmatrix} 0.816496580927726 \\ 0 \end{bmatrix},$$

$$R = A^{(4)} = \begin{bmatrix} 2.00000000000000 & 1.50000000000000 & 1.00000000000000 \\ 0 & 0.866025403784438 & 0.577350269189625 \\ 0 & 0 & 0.816496580927726 \\ 0 & 0 & 0 \end{bmatrix}.$$

将以上的Householder变换矩阵顺序相乘，得到正交矩阵Q

$$Q = H_1 H_2 H_3 = \begin{bmatrix} -0.500000000000000 & 0.866025403784439 & 0 & 0.000000000000000 \\ -0.500000000000000 & -0.288675134594813 & 0.816496580927726 & 0.000000000000000 \\ -0.500000000000000 & -0.288675134594813 & -0.408248290463863 & -0.707106781186548 \\ -0.500000000000000 & -0.288675134594813 & -0.408248290463863 & 0.707106781186547 \end{bmatrix}.$$

Householder QR分解的代码实现如下：

```
function [W,A] = house(A)
% Householder QR Factorization
% Input  - A  a symmetric matrix can be factorized as Q*R
% Output - W  NOT the factorized orthogonal matrix Q!
%        - A  the upper triangle matrix

[m,n] = size(A);
W = zeros(m, n);
for k = 1:n
  x = A(k:m, k);
  v = x;
  v(1) = sign(x(1))*norm(x) + v(1);
  v = v/norm(v);
  A(k:m,k:n) = A(k:m,k:n) - 2*v* (v'*A(k:m,k:n));
  W(k:m, k) = v;
end

end
```

```
function Q = formQ (W)
[m,n] = size(W);
Q = eye(m);
for k = n:-1:1
  v = W(k:m,k);
  Q(k:m, :) = Q(k:m, :) - 2*v* (v'*Q(k:m, :));
end
```

2. 使用Givens旋转变化求矩阵A的QR分解：

Step1: 计算一个旋转变化矩阵，使得当其作用于a_{31}和a_{41}时能够使$a_{41}=0$

$$\begin{bmatrix} 0.707106781186547 & -0.707106781186547 \\ 0.707106781186547 & 0.707106781186547 \end{bmatrix}^T \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1.414213562373095 \\ 0 \end{bmatrix}$$

将此变换作用于第3和第4行，得：

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.707106781186547 & -0.707106781186547 \\ 0 & 0 & 0.707106781186547 & 0.707106781186547 \end{bmatrix}^T \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1.414213562373094 & 1.414213562373094 & 1.414213562373094 \\ 0 & 0 & 0 \end{bmatrix}$$

（由于页面宽度限制，也为了方便阅读，以下步骤中的数据均进行了数值精度的舍弃. 注意只是在此书面报告中舍弃了部分数位，而原程序中未舍弃）

Step2: 计算一个旋转变化矩阵，使得当其作用于a_{21}和a_{31}时能够使$a_{31}=0$

$$\begin{bmatrix} 0.8165 & -0.5774 \\ 0.5774 & 0.8165 \end{bmatrix}^T \begin{bmatrix} 1 \\ 1.4142 \end{bmatrix} = \begin{bmatrix} 1.7321 \\ 0 \end{bmatrix}$$

将此变换作用于第2和第3行，得：

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.5774 & -0.8165 & 0 \\ 0 & 0.8165 & 0.5774 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^T \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1.4142 & 1.4142 & 1.4142 \\ 0 & 0 & 0 \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 0 & 0 \\ 1.7321 & 1.7321 & 1.1547 \\ -1.1102e-16 & -1.1102e-16 & 0.8165 \\ 0 & 0 & 0 \end{bmatrix}$$

Step3: 计算一个旋转变化矩阵，使得当其作用于a_{11}和a_{21}时能够使$a_{21}=0$，将此变换作用于第2和第3行，得：

$$\begin{bmatrix} 0.5000 & -0.8660 & 0 & 0 \\ 0.8660 & 0.50000 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^T \begin{bmatrix} 1 & 0 & 0 \\ 1.7321 & 1.7321 & 1.1547 \\ -1.1102e-16 & -1.1102e-16 & 0.8165 \\ 0 & 0 & 0 \end{bmatrix}$$
$$= \begin{bmatrix} 2 & 1.5 & 1 \\ -1.1102e-16 & 0.8660 & 0.5774 \\ -1.1102e-16 & -1.1102e-16 & 0.8165 \\ 0 & 0 & 0 \end{bmatrix}$$

Step4: $\cdots\cdots$（原理与前面步骤相同，不再赘述）
最终结果：

$$\mathbf{Q} = \begin{bmatrix} 0.5 & -0.866025403784439 & -0.000000000000000 & 0 \\ 0.5 & 0.288675134594813 & -0.816496580927726 & 0 \\ 0.5 & 0.288675134594813 & 0.408248290463863 & -0.707106781186547 \\ 0.5 & 0.288675134594813 & 0.408248290463863 & 0.707106781186547 \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} 2.000000000000000 & 1.500000000000000 & 1.000000000000000 \\ -0.000000000000000 & 0.866025403784439 & 0.577350269189626 \\ -0.000000000000000 & 0 & 0.816496580927726 \\ 0 & 0 & 0 \end{bmatrix}$$

Givens QR分解的代码实现如下：

```matlab
function [Q,R] = qrgivens(A)
% Givens QR Factorization
% Input  - A  a symmetric matrix can be factorized as Q*R
% Output - Q  the orthogonal matrix
%        - R  the upper triangle matrix

[m,n] = size(A);
Q = eye(m);
R = A;

for j = 1:n
for i = m:-1:(j+1)
  G = eye(m);
  [c,s] = givensrotation( R(i-1,j),R(i,j) );
  G([i-1, i],[i-1, i]) = [c -s; s c];
  R = G'*R;
  Q = Q*G;
end
end

end
```

```matlab
function [c,s] = givensrotation(a,b)

if b == 0
  c = 1;
  s = 0;
else
  if abs(b) > abs(a)
    r = a / b;
    s = 1 / sqrt(1 + r^2);
    c = s*r;
  else
    r = b / a;
    c = 1 / sqrt(1 + r^2);
    s = c*r;
  end
end

end
```