QR 分解

张晓平

2018 年 10 月 22 日

设 $A \in \mathbb{R}^{m\times n}$，$b \in \mathbb{R}^m$。由于 2 范数具有正交不变性，故 $\forall$ 正交矩阵 $Q \in \mathbb{R}^{m\times m}$，有

$$\|Ax-b\|_2 = \|Q^T(Ax-b)\|_2.$$

于是，原最小二乘问题 $\min\|Ax-b\|_2$ 等价于

$$\min\|Q^TAx - Q^Tb\|_2 \tag{1}$$

因此，可通过适当选取正交矩阵 Q，使原问题转化为较为容易求解的最小二乘问题 (1)。

1 QR 分解

定理 1 (QR 分解定理). 设 $A \in \mathbb{R}^{m\times n}(m \geq n)$，则 A 有 QR 分解：

$$A = Q\begin{bmatrix} R \\ 0 \end{bmatrix} \tag{2}$$

其中 $Q \in \mathbb{R}^{m\times m}$ 为正交矩阵，$R \in \mathbb{R}^{n\times n}$ 是具有非负对角元的上三角阵；而且当 $m = n$ 时且 A 非奇异时，上述分解唯一。

证明. 先证明 QR 分解的存在性。对 n 用数学归纳法。

1^o 当 $n = 1$ 时，A 退化为列向量，由 Householder 变换可知存在正交矩阵 Q

$$QA = \|A\|_2 e_1.$$

2^o 假定已经证明定理对所有 $p\times(n-1)$ 矩阵成立，其中 $p \geq n-1$。设 $A \in \mathbb{R}^{m\times n}$ 的第一列为 a_1，则由 Householder 变换可知，存在正交阵 $Q_1 \in \mathbb{R}^{m\times m}$ 使得

$$Q_1^T a_1 = \|a_1\|_2 e_1.$$

于是，

$$Q_1^T A = \begin{bmatrix} \|a_1\|_2 & v^T \\ 0 & A_1 \end{bmatrix}.$$

由归纳假设，存在 $(m-1)\times(m-1)$ 正交矩阵 Q_2 使得

$$A_1 = Q_2\begin{bmatrix} R_2 \\ 0 \end{bmatrix},$$

其中 R_2 为具有非负对角元的 $(n-1)\times(n-1)$ 上三角阵。令

$$Q = Q_1\begin{bmatrix} 1 & 0 \\ 0 & Q_2 \end{bmatrix}, \quad R = \begin{bmatrix} \|a_1\|_2 & v^T \\ 0 & R_2 \\ 0 & 0 \end{bmatrix}$$

则 $A=QR$。于是由归纳法可知存在性得证。

再证唯一性。设 $m=n$ 且 A 非奇异，并假定 $A=QR=\tilde{Q}\tilde{R}$。A 奇异蕴含着 $R,\tilde{R}$ 的主对角元均为正。因此

$$\tilde{Q}^TQ=\tilde{R}R^{-1}.$$

这说明等式两边既是正交阵，也是主对角元恒为正的上三角阵，从而必是单位阵。于是 $Q=\tilde{Q},R=\tilde{R}$，即分解是唯一的。 □

设 $A\in\mathbb{R}^{m\times n}(m\geq n)$ 有线性无关的列，$b\in\mathbb{R}^m$，且假定已知 A 的QR 分解 (2)。现将 Q 分块为

$$Q=\begin{array}{c}\left(\begin{array}{cc} Q_1 & Q_2\end{array}\right)\\ \begin{array}{cc} n & m-n\end{array}\end{array}$$

且令

$$Q^Tb=\left(\begin{array}{c} Q_1^T\\ Q_2^T\end{array}\right)b=\left(\begin{array}{c} c_1\\ c_2\end{array}\right)\begin{array}{l} n\\ m-n\end{array}$$

则

$$\|Ax-b\|_2^2=\|Q^TAx-Q^Tb\|_2^2=\|Rx-c_1\|_2^2+\|c_2\|_2^2.$$

由此可知，x 是最小二乘问题的解当且仅当 x 是 $Rx=c_1$ 的解。

正交化方法的步骤

(1) 计算A的QR分解
(2) 计算$c_1=Q_1^Tb$
(3) 求解上三角方程组$Rx=c_1$

2 QR 分解的实现

设 $m=7,n=5$，假定已计算出 Householder 变换 $H_1\in\mathbb{R}^{7\times 7}$ 使得

$$H_1A=\begin{bmatrix} \times & \times & \times & \times & \times\\ 0 & + & \times & \times & \times\\ 0 & + & \times & \times & \times\\ 0 & + & \times & \times & \times\\ 0 & + & \times & \times & \times\\ 0 & + & \times & \times & \times\\ 0 & + & \times & \times & \times\end{bmatrix},$$

则对于第二列标为“+”的 6 个元素，可确定一个 Householder 变换 $\tilde{H}_2\in\mathbb{R}^{6\times 6}$ 使得

$$\tilde{H}_2\begin{bmatrix} +\\ +\\ +\\ +\\ +\\ +\end{bmatrix}=\begin{bmatrix} \times\\ 0\\ 0\\ 0\\ 0\\ 0\end{bmatrix},$$

令 $H_2 = \mathrm{diag}(I_1, \tilde{H}_2)$，则

$$H_2 H_1 A = \begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & + & \times & \times \\ 0 & 0 & + & \times & \times \\ 0 & 0 & + & \times & \times \\ 0 & 0 & + & \times & \times \\ 0 & 0 & + & \times & \times \end{bmatrix},$$

则对于第三列标为“+”的 5 个元素，可确定一个 Householder 变换 $\tilde{H}_3 \in \mathbb{R}^{5\times 5}$ 使得

$$\tilde{H}_3 \begin{bmatrix} + \\ + \\ + \\ + \\ + \end{bmatrix} = \begin{bmatrix} \times \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

令 $H_3 = \mathrm{diag}(I_2, \tilde{H}_3)$，则

$$H_3 H_2 H_1 A = \begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}.$$

对于一般的矩阵 $A \in \mathbb{R}^{m\times n}$，假设已进行了 $k-1$ 步，得到了 Householder 矩阵 $H_1, H_2, \cdots, H_{k-1}$，使得

$$H_{k-1}\cdots H_2 H_1 = \begin{array}{cc} \begin{pmatrix} A_{11}^{(k)} & A_{12}^{(k)} \\ 0 & A_{22}^{(k)} \end{pmatrix} & \begin{array}{l} k-1 \\ m-k+1 \end{array} \\ \begin{array}{cc} k-1 & \quad n-k+1 \end{array} & \end{array}$$

其中 $A_{11}^{(k)}$ 是上三角阵。假定

$$A_{22}^{(k)} = [u_k, \cdots, u_n],$$

则第 k 步是确定 Householder 变换

$$\tilde{H}_k = I_{m-k+1} - \beta_k v_k v_k^T \in \mathbb{R}^{(m-k+1)\times(m-k+1)}$$

使得

$$\tilde{H}_k u_k = r_{kk} e_1$$

其中 $r_{kk} \geq 0$, $e_1 = (1, 0, \cdots, 0)^T \in \mathbb{R}^{m-k+1}$。令 $H_k = \mathrm{diag}(I_{k-1}, \tilde{H}_k)$，则

$$\begin{aligned} A_{k+1} \quad &= \quad H_k A_k = \begin{pmatrix} A_{11}^{(k)} & A_{12}^{(k)} \\ 0 & \tilde{H}_k A_{22}^{(k)} \end{pmatrix} \\ &= \begin{array}{cc} \begin{pmatrix} A_{11}^{(k+1)} & A_{12}^{(k+1)} \\ 0 & A_{22}^{(k+1)} \end{pmatrix} & \begin{array}{l} k \\ m-k \end{array} \\ \begin{array}{cc} k & \quad n-k \end{array} & \end{array} \end{aligned}$$

其中 $A_{11}^{(k+1)}$ 是上三角阵。这样，从 $k=1$ 出发，依次进行 n 次，就可将 A 约化为上三角阵。记

$$R = A_{11}^{(n)}, \quad Q = H_1 \cdots H_n,$$

则

$$A = Q \begin{bmatrix} R \\ 0 \end{bmatrix}.$$

以下考虑 QR 分解的存储问题。当分解完成后，A 不再需要，可用来存放 Q 与 R。实际计算中，并不会将 Q 算出，而只存放 n 个 Householder 矩阵，而对每个 H_k，只需保存 v_k 和 β_k。注意到 v_k 有如下形式

$$v_k = \begin{bmatrix} 1 \\ v_{k+1}^{(k)} \\ \vdots \\ v_n^{(k)} \end{bmatrix}$$

正好可以把 $v_k(2:m-k+1)$ 存储在 A 的主对角元以下的位置，另外需要一个数组来存放 $\beta_i, i=1,\cdots,n$。例如，对 $m=4, n=3$ 的问题，存储方式为

$$A = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ v_2^{(1)} & r_{22} & r_{23} \\ v_3^{(1)} & v_3^{(2)} & r_{33} \\ v_4^{(1)} & v_4^{(2)} & v_4^{(3)} \end{bmatrix}, \quad d = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}$$

3 QR 分解：Matlab 代码

通过实验获得数据如下：

x_i	1	2	3	4	6	7	8
y_i	2	3	6	7	5	3	2

试用最小二乘法求多项式曲线，使与此数据相拟合。

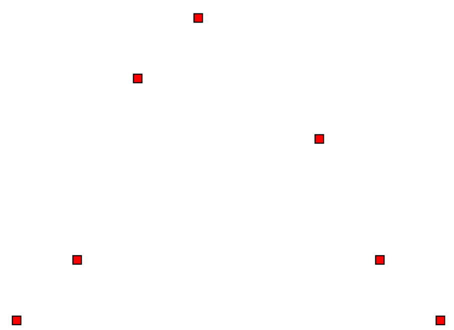

图 1: 数据点散列图

3.1 QR 分解：利用 Householder 变换

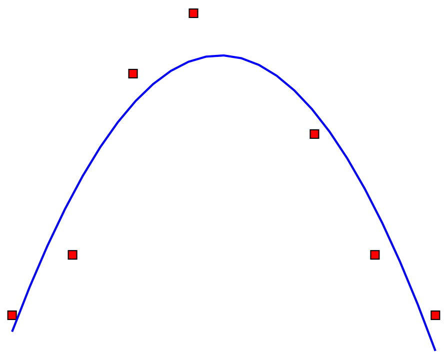

图 2: 曲线拟合：二次曲线

```
function [v, beta] = householder(x)
n = length(x);
eta = max(abs(x));
x = x / eta;
sigma = x(2:n)' * x(2:n);
v = zeros(n, 1);
v(2:n) = x(2:n);
if sigma == 0
    beta = 0;
else
    alpha = sqrt(x(1)^2 + sigma);
    if x(1) <= 0
        v(1) = x(1) - alpha;
    else
        v(1) = -sigma / (x(1) + alpha);
    end
    beta = 2 * v(1)^2 / (sigma + v(1)^2);
    v = v / v(1);
end
end
```

```
function [A, d] = qr_householder(A)
[m, n] = size(A);
d = zeros(n, 1);
for j = 1:n
    if j < m
       [v, beta] = householder(A(j:m, j));
       I = eye(m-j+1);
       %A(j:end,j:end) = (I - beta*v*v') * A(j:end, j:end);
       A(j:end, j:end) = A(j:end, j:end) - (beta * v) * (v'* A(j:end, j:end
       )) ;
       d(j) = beta;
       A(j+1:end, j) = v(2:m-j+1);
```

```
    end
end
end
```

```
function y = bs(U, y)
%  find solution of upper triangular equation
%  by using back substitution
n = size(U, 1);
for j = n:-1:2
    y(j) = y(j) / U(j,j);
    y(1:j-1) = y(1:j-1) - y(j) * U(1:j-1,j);
end
y(1) = y(1) / U(1,1);
end
```

```
m = 3;
n = 2;
%x = [1 2 3 4 6 7 8]';
%A = [ones(m,1) x x.^2];
%b = [2 3 6 7 5 3 2]';
A=[2 3; -2 -6; 1 0];
b=[3 -3 6]';
%% find solution of contradictionary equations

%% Step 1: processing QR factorization
[A d] = qr_householder(A);

%%          recover Q and R
Q = eye(m);
for j = 1:n
    I = eye(j-1);
    Z = zeros(j-1,m-j+1);
    H = [ I  Z; ...
          Z' eye(m-j+1)-d(j)*[1; A(j+1:m, j)]*[1; A(j+1:m, j)]'];
    Q = Q*H;
end
R = triu(A);


%% Step 2: comput c1 = Q1'*b
c1 = Q(:,1:n)'*b;
c1 = bs(R(1:n,1:n), c1)
```

3.2 QR 分解：利用 Givens 变换

```
function [c, s] = givens_rotation(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
```

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

  for j = 1:n
    for i = m:-1:(j+1)
      G = eye(m);
      [c, s] = givens_rotation( 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
```