QR Decomposition

Wu Yunni

We are going to perform QR Decomposition of n × n Matrix in this report via modified Gram-Schmidt method and Householder reflections respectively and compare these two methods. The following code is written via Python3.

I. modified Gram-Schmidt method

For a n × n Matrix A = $[v_1 \quad v_2 \quad \cdots \quad v_n]$, with columns $v_1, v_2, \cdots, v_n$.

(1) Normalize first column.

$$r_{11} = \|v_1\|_2, q_1 = \frac{v_1}{r_{11}}$$

(2) For other columns v_i, obtain u_i that is normal q_j for $j < i$ via delete orthogonal projection of v_i onto q_j for $j < i$ and normalize it.

$$u_i = v_i - \sum_{j=1}^{i-1} r_{ji}\, q_j, \text{with } r_{ji} = \langle v_i, q_j \rangle, \text{for } j < i \text{ and } r_{jj} = \left\|u_j\right\|_2$$

$$q_i = \frac{u_i}{r_{ii}}$$

(3) QR decomposition

$$\mathrm{R_{ij}} = \begin{cases} r_{ij} = \langle v_j, q_i \rangle & \text{for } i < j \\ r_{ii} = \|u_i\|_2 & \\ r_{ij} = 0 & \text{for } i < j \end{cases}$$

$$\mathrm{Q} = [q_1 \quad q_2 \quad \cdots \quad q_n]$$

2. Code

```
# input: nxn Matrix
def GM(Matrix):
    # col_no=n
    column_no=Matrix.shape[0]
    Q[:,:]=Matrix[:,:]
    R=np.mat(np.zeros((column_no,column_no)))
    for i in range(column_no):
        for j in range(i):
            R[j,i]=np.matmul(Q[:,j].T,Q[:,i])[0,0]
            # delete projection
            Q[:,i]=Q[:,i]-R[j,i]*Q[:,j]
        R[i,i]=np.matmul(Q[:,i].T,Q[:,i])[0,0]**(0.5)
        # normalization
        Q[:,i]=Q[:,i]/R[i,i]
    return [Q,R]
```

II. Householder reflections

1. Zero out Entries in the First Column of a Matrix using a Householder Reflection

For a $\mathrm{n} \times \mathrm{n}$ Matrix A, the first column $x = [x_1 \quad x_2 \quad \cdots \quad x_n]^{\mathrm{T}}$.

(1) Calculate the maximum Column element $\mathrm{Colmax} = \max(|x_1|, |x_2|, \cdots, |x_n|)$

(2) $\bar{x} = x/\mathrm{Colmax}$

(3) Compute u and β

$$u = \bar{x} \pm \|\bar{x}\|_2 e_1, \beta = \frac{2}{u^{\mathrm{T}} u}$$

The sign $\pm$ depends on whether $x_1 > 0$.

(4) Householder reflections

$$H_{u_1} = I - \beta u u^{\mathrm{T}}$$

$$A_1 = H_{u_1} A$$

2. Repeat this process until all Entries under diagonal element of other Columns of the Matrix are zero out

We will apply the process above to a $\mathrm{n} - 1 \times \mathrm{n} - 1$ submatrix of Matrix A_1 by deleting the first column and row of Matrix A_1. And repeat the process up to all Entries under diagonal element of Matrix A are zero out.

Then we have:

$$H_{n-1} \cdots H_2 H_1 A = R$$

Where R is a upper-triangle matrix and H_{u_i} is the Householder reflections in step i, H_i is a $\mathrm{n} \times \mathrm{n}$ Matrix with $H_i = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & H_{u_i} \end{bmatrix}$.

Then the QR decomposition of A:

$$R = H_{n-1} \cdots H_2 H_1 A, Q = H_1 H_2 \cdots H_{n-1}$$

3. Code:

```
# input: nxn Matrix
def Householder(Matrix):
    # col_no=n
    column_no=Matrix.shape[0]
    # nxn identity matrix
    R=np.mat(np.identity(column_no))
    Q=np.mat(np.identity(column_no))
    A=Matrix[:,:]
    for i in range(column_no-1):
        # n-i x n-i identity matrix
        identity=np.mat(np.identity(column_no-i))
        x=A[:,0]/A[:,0].max()
        if x[0,0]>=0:
```

```
            u=x+(np.matmul(x.T,x)[0,0]**(0.5))*identity[:,0]
        else:
            u=x-(np.matmul(x.T,x)[0,0]**(0.5))*identity[:,0]
        b=2/np.matmul(u.T,u)[0,0]
        # Householder reflection (n-i x n-i)
        Hsmall=identity-b*np.matmul(u,u.T)
        # result matrix (n-i x n-i)
        Asmall=np.matmul(Hsmall,A)
        A=Asmall[1:,1:]
        H=np.mat(np.identity(column_no))
        H[i:,i:,]=Hsmall
        # orthogonal matrix
        Q=np.matmul(Q,H)
        # upper-triangle matrix
        R[i:,i:]=Asmall
    # output Q and R
    return [Q,R]
```

III. Some example and results

1.

$$B = \begin{bmatrix} 1 & 2 & 0 \\ -1 & 4 & 1 \\ -3 & 1 & 2 \end{bmatrix}$$

```
Q,R=GM(B)  # Gram-Smith
print(Q.round(10))
print(R.round(10))
print(np.matmul(Q,R).round(10))
```

```
[[ 0.30151134  0.56719685  0.76640632]
 [-0.30151134  0.81928434 -0.48771311]
 [-0.90453403 -0.08402916  0.41803981]]
[[ 3.31662479 -1.50755672 -2.11057941]
 [ 0.          4.3275019   0.65122601]
 [ 0.          0.          0.34836651]]
[[ 1.  2.  0.]
 [-1.  4.  1.]
 [-3.  1.  2.]]
```

```
Q,R=Householder(B)  # Householder
print(Q.round(10))
print(R.round(10))
print(np.matmul(Q,R).round(10))
```

```
[[-0.30151134 -0.56719685  0.76640632]
 [ 0.30151134 -0.81928434 -0.48771311]
 [ 0.90453403  0.08402916  0.41803981]]
[[-3.31662479  1.50755672  2.11057941]
 [-0.         -4.3275019  -0.65122601]
 [-0.         -0.          0.34836651]]
[[ 1.  2.  0.]
 [-1.  4.  1.]
 [-3.  1.  2.]]
```

As we can see in the figure above, our two approaches of QR decomposition give us the equivalent results (same except for some plus minus signs that will cancel out with each other) up to at least 10 d.p and we can return to B by B=QR.

2. 15X15 Hilbert Matrix

```
X = 1. / (np.arange(1, 16) + np.arange(0, 15)[:, np.newaxis])
H=np.matrix(X) # 15x15 Hilbert Matrix
np.linalg.cond(H)
```

```
2.495951750009794e+17
```

```
I=np.mat(np.identity(15))
Q,R=GM(H) # Gram-Smith
print(np.linalg.norm(I-np.matmul(Q,Q.T),ord=2)) # 2-norm
```

```
0.9782962275967905
```

```
Q,R=Householder(H) # Householder
print(np.linalg.norm(I-np.matmul(Q,Q.T),ord=2)) # 2-norm
```

```
8.617771840179688e-16
```

As we can see in the figure above, the Q computed by Gram-Smith is not orthogonal, but Q computed by Householder is still orthogonal.
This is because 15×15 Hilbert Matrix H has a condition number of 2.4959×10^{17}, and Gram-Smith algorithm fails when the matrix is ill-conditioned but Householder algorithm still holds even the matrix is ill-conditioned.