STA 141C - Big Data & High Performance Statistical Computing **Spring 2022**

Week 5-1: QR decomposition

Lecturer: Bo Y.-C. Ning *April 26, 2022*

Disclaimer: *My notes may contain errors, distribution outside this class is allowed only with the permission of the Instructor.*

Announcement

- hw2 posted

Last time

- Examples using Cholesky decomposition

Today

- QR decomposition

1 QR factorization

For $A \in \mathbb{R}^{n \times p}$, an n-by-p rectangle matrix,

$$A = QR,$$

where $Q \in \mathbb{R}^{n \times n}$, $Q'Q = I_n$, and $R \in \mathbb{R}^{n \times p}$.

Usually, we use the thin QR, $A = Q_1 R_1$, $Q_1 \in \mathbb{R}^{n \times p}$, $Q_1' Q_1 = I_p$ and $R_1 \in \mathbb{R}^{p \times p}$, where Q_1 has orthogonal columns and R_1 is an invertible upper triangular matrix with positive diagonal entries.

Theorem 5.1 *If $A \in \mathbb{R}^{n \times p}$, then there exists a matrix $Q \in \mathbb{R}^{n \times p}$ has orthogonal columns and $R \in \mathbb{R}^{p \times p}$ is upper triangular matrix. In particular, $R = G'$, where G is the lower triangular Cholesky factor of $A'A$.*

1.1 Linear regression with thin QR

Our goal is to solve $\beta, \hat{y}, SSE$ from $X'X\beta = X'y$. Let's consider the augmented matrix:

$$\begin{pmatrix} X & y \end{pmatrix} = \begin{pmatrix} Q & q \end{pmatrix} \begin{pmatrix} R & r \\ 0_p' & d \end{pmatrix} = \begin{pmatrix} QR & Qr + dq \end{pmatrix}$$

Thus, $X'X\beta = X'y$ implies $R\beta = {R^{-1}}'X'y = {R^{-1}}'R'Q'y = Q'y = r$, $\hat{y} = X\hat{\beta} = QRR^{-1}r = Qr$, $\hat{e} = y - X\hat{\beta} = y - Qr = dq$, and $SSE = d^2$.

2 Numerical methods to solve QR:

Two things one may want to consider:

- Is QR faster than LU or Cholesky?
- Why QR?

Today and the next lecture, we introduce three popular numerical methods to solve QR:

- Gram-Schmidt
- Householder transformation
- Givens transformation
- Modified Gram-Schmidt

2.1 Gram-Schmidt method

Assume $A = (a_1, \ldots, a_p) \in \mathbb{R}^{n \times p}$ has full column rank

Consider a matrix $A = (a_1, \ldots, a_n)$, the steps of the Gram-Schmidt algorithm is given as follows:

1) $u_1 = a_1, \quad e_1 = u_1/\|u_1\|$

2) $u_2 = a_2 - \langle a_2, e_1 \rangle e_1, \quad e_2 = u_2/\|u_2\|$

$\ldots$

k) $u_k = a_k - \langle a_k, e_1 \rangle e_1 - \cdots - \langle a_k, e_{k-1} \rangle e_{k-1}, \quad e_k = u_k/\|u_k\|$

Then,

$$A = (a_1, \ldots, a_n) = (e_1, \ldots, e_n) \begin{pmatrix} \langle a_1, e_1 \rangle & \langle a_2, e_1 \rangle & \cdots & \langle a_n, e_1 \rangle \\ 0 & \langle a_2, e_2 \rangle & \cdots & \langle a_n, e_2 \rangle \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \langle a_n, e_n \rangle \end{pmatrix} = QR$$

Example:

Consider the matrix $A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$

- What e_1, e_2, e_3?

Step 1 $u_1 = a_1 = (1,1,0)'$, $e_1 = u_1/\|u_1\| = (1,1,0)'/\sqrt{2} = (1/\sqrt{2}, 1/\sqrt{2}, 0)'$

Step 2 $u_2 = a_2 - \langle a_2, e_1 \rangle e_1 = (1,0,1)' - \frac{1}{\sqrt{2}}(1/\sqrt{2}, 1/\sqrt{2}, 0)' = (1/2, -1/2, 1)$

$e_2 = u_2/\|u_2\| = (1/\sqrt{6}, -1/\sqrt{6}, 2/\sqrt{6})'$.

Step 3 $u_3 = (-1/\sqrt{3}, 1/\sqrt{3}, 1/\sqrt{3})'$ and $e_3 = (-1/\sqrt{3}, 1/\sqrt{3}, 1/\sqrt{3})'$

Thus,

$$Q = (e_1, e_2, e_3), \quad R = \begin{pmatrix} \sqrt{2} & 1/\sqrt{2} & 1/\sqrt{2} \\ 0 & 3/\sqrt{6} & 1/\sqrt{6} \\ 0 & 0 & 2/\sqrt{3} \end{pmatrix}$$

3 Householder transformation

Let's first take a look at an example for getting QR for $A \in \mathbb{R}^{5\times 4}$ using the Householder transformation. Given a matrix A, where

$$A = \begin{pmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{pmatrix}$$

In Step 1, we choose H_1 such that

$$A = \begin{pmatrix} \color{red}\times & \times & \times & \times \\ \color{red}\times & \times & \times & \times \\ \color{red}\times & \times & \times & \times \\ \color{red}\times & \times & \times & \times \\ \color{red}\times & \times & \times & \times \end{pmatrix} \longrightarrow H_1 \begin{pmatrix} \color{red}\times \\ \color{red}\times \\ \color{red}\times \\ \color{red}\times \\ \color{red}\times \end{pmatrix} = \begin{pmatrix} \times \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Thus,

$$H_1 A = \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{pmatrix}$$

In Step 2, we choose $\tilde{H}_2$ such that

$$H_1 A = \begin{pmatrix} \times & \times & \times & \times \\ 0 & \color{red}\times & \times & \times \\ 0 & \color{red}\times & \times & \times \\ 0 & \color{red}\times & \times & \times \\ 0 & \color{red}\times & \times & \times \end{pmatrix} \longrightarrow \tilde{H}_2 \begin{pmatrix} \color{red}\times \\ \color{red}\times \\ \color{red}\times \\ \color{red}\times \end{pmatrix} = \begin{pmatrix} \times \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

and let

$$H_2 = \begin{pmatrix} 1 & 0_4' \\ 0_4 & \tilde{H}_2 \end{pmatrix}$$

We then obtain

$$H_2 H_1 A = \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{pmatrix}$$

In Step 3, we choose $\tilde{H}_3$ such that

$$H_2 H_1 A = \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{pmatrix} \longrightarrow \tilde{H}_3 \begin{pmatrix} \times \\ \times \\ \times \end{pmatrix} = \begin{pmatrix} \times \\ 0 \\ 0 \end{pmatrix}.$$

We let

$$H_3 = \begin{pmatrix} 1 & 0 & 0_3' \\ 0 & 1 & 0_3' \\ 0_3 & 0_3 & \tilde{H}_3 \end{pmatrix}$$

We continue this for the last column, finally we obtain $R = H_4 H_3 H_2 H_1 A$, which is an upper triangular matrix and $Q = H_1 H_2 H_3 H_4$, which is an orthogonal matrix. Can you verify $A = QR$?

In general, we have the following definition for the householder matrix.

Definition 5.2 (Householder matrix) *Let v be a $k \times 1$ vector. The Householder matrix associated to v is the $k \times k$ matrix H_k defined as follows:*

$$H_k = I_k - \frac{2vv'}{\|v\|^2},$$

where I is the $k \times k$ identity matrix, $\|v\|$ is the norm of v.

Assume $A = (a_1, \dots, a_p) \in \mathbb{R}^{n \times p}$ has full column rank, we construct matrices $H_1, \dots, H_p$ in $\mathbb{R}^{n \times n}$ such that

$$H_p \dots H_2 H_1 A = \begin{pmatrix} R_1 \\ 0 \end{pmatrix},$$

then the matrices H_js are called *Householder matrices*.

How many flops for QR via Householder transformation? The total is $2p^2(n - p/3)$ flops for an $n \times p$ matrix.

QR via Householder transformation is built in LAPACK. In python: `scipy.linalg.qr` is a wrapper of the LAPACK routines `dgeqrf`, `zgeqrf`, `dorgqr`, and `zungqr`, note that `dgeqrf` uses Householder transformation for real matrices In R, the default option for `qr` is `DQRDC` in LINPACK, which also uses the Householder transformation.

4 Givens rotation

Again, assuming $A \in \mathbb{R}^{n \times p}$ has full column rank, Givens rotation provides another approach for getting $A = QR$.

The next example illustrates how givens rotation works (from Page 252 of Golub & van Loan, 4th edition): Consider a 4×3 matrix,

$$
\begin{pmatrix} \times & \times & \times \\ \times & \times & \times \\ {\color{red}\times} & \times & \times \\ {\color{red}\times} & \times & \times \end{pmatrix}
\xrightarrow{(1;\ 3,4)}
\begin{pmatrix} \times & \times & \times \\ {\color{red}\times} & \times & \times \\ {\color{red}\times} & \times & \times \\ 0 & \times & \times \end{pmatrix}
\xrightarrow{(1;\ 2,3)}
\begin{pmatrix} {\color{red}\times} & \times & \times \\ {\color{red}\times} & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{pmatrix}
\xrightarrow{(1;\ 1,2)}
$$

$$
\begin{pmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & {\color{red}\times} & \times \\ 0 & {\color{red}\times} & \times \end{pmatrix}
\xrightarrow{(2;\ 3,4)}
\begin{pmatrix} \times & \times & \times \\ 0 & {\color{red}\times} & \times \\ 0 & {\color{red}\times} & \times \\ 0 & 0 & \times \end{pmatrix}
\xrightarrow{(2;\ 2,3)}
\begin{pmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & {\color{red}\times} \\ 0 & 0 & {\color{red}\times} \end{pmatrix}
\xrightarrow{(3;\ 3,4)} R
$$

Definition 5.3 *Given a vector with length two, the Givens rotation matrix is given by G such that*

$$G' \begin{pmatrix} a & b \end{pmatrix} = \begin{pmatrix} \gamma & -\sigma \\ \sigma & \gamma \end{pmatrix}' \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} r \\ 0 \end{pmatrix},$$

where $r = \sqrt{a^2 + b^2}$, $\sigma = \sin\theta$, and $\gamma = \cos\theta$. To compute γ and σ, a straightforward approach is

$$\gamma = \frac{a}{\sqrt{a^2 + b^2}}, \quad \sigma = \frac{b}{\sqrt{a^2 + b^2}}.$$

In practice, we should avoid calculating $\sqrt{a^2 + b^2}$ especially when a, b are small. Instead, if $a^2 > b^2$, we shall take $\tau = b/a < 1$, $\gamma = \frac{1}{\sqrt{1+\tau^2}}$, and $\sigma = \gamma\tau$; and if $a^2 < b^2$, we shall take $\tau = a/b < 1$, $\sigma = \frac{1}{\sqrt{1+\tau^2}}$, and $\gamma = \sigma\tau$.

Notice that $Q = G_1 \ldots, G_t$ with t number of total rotations. QR by givens rotation takes $3p^2(n - p/3)$ flops.

In R, givens() function in the pracma package (see here) uses givens rotation.

In python, the build-in function scipy.linalg.qr uses Householder as default. The givens implementation is available here

5 Modified G-S algorithm

The regular G-S is unstable (as one may loose orthogonality due to roundoff errors) when columns of A are collinear. The modified G-S comes to the rescue. The figure below highlight the difference between G-S and modified G-S.

Flop counts for both GS and MGS are $\sum_{k=1}^{p} 2n(k-1) \approx np^2$.

[Source: `https://arnold.hosted.uark.edu/NLA/Pages/CGSMGS.pdf`]

More things to read:

- Difference between GS and MGS: click here
- MGS method with sample code: click here
- See why MGS is more stable than GS: click here

Classical	**Modified**
for k=1:n,	for k=1:n,
$w = a_k$	$w = a_k$
for j = 1:k-1,	for j=1:k-1,
$r_{jk} = q_j^T w$	$r_{jk} = q_j^T w$
end	$w = w - r_{jk} q_j$
for j = 1:k-1,	
$w = w - r_{jk} q_j$	
end	end
$r_{kk} = \|w\|_2$	$r_{kk} = \|w\|_2$
$q_k = w/r_{kk}$	$q_k = w/r_{kk}$
end	end

Figure 5.1: Comparing G-S and modified G-S algorithms. To compare with the notations in Section 2.1, w is u, q_j is e_j and $r_{jk} = \langle a_k, e_j \rangle$.

6 Review of numerical linear algebra

Consider solving the problem $Ax = b$, $A \in \mathbb{R}^{n \times n}$ is symmetric and p.s.d.

So far, we have learned GE/LU, Cholesky, QR, and soon we will learn SVD.

Let's summarize the numerical methods we learned for solving linear regression problem

Method	Flops	Software	Stability
GE/LU	$O(2n^3/3)$	R & Python	less stable
Cholesky	$O(n^3/3)$	R & Python	stable than GE/LU, lesser stable than QR
QR by HH	$O(2n^3)$	R & Python	stable
QR by Givens	$O(3n^3)$	R & Python	stable
QR by MGS	$O(n^3)$		more stable

A few notes:

- Cholesky are twice faster than QR and need less space
- QR are more stable and produce numerically more accurate solution
- MGS is slower than HH but yields Q_1 (thin QR)
- There is simply no such thing as a universal 'gold standard' when it comes to algorithms.
- Flop counts is not everything. GE/LU has a higher memory traffic and vectorization overheads.
- QR is comparable in efficiency and more stable