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

Lecture 6: Cholesky decomposition (cont'd) and QR factorization

Lecturer: Bo Y.-C. Ning *January 25, 2022*

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

Announcement

- hw1 due tomorrow
- Group members list due this Sunday
  - Please send your members' name and email address to my email

Last time

- Partial pivoting for LU
- Cholesky decomposition

Today

- Examples using Cholesky decomposition
- QR factorization

1 Examples using Cholesky decomposition

1.1 Evaluating a multivariate normal log-likelihood function

Consider the multivariate normal density you learned in statistical methods course

$$y \sim \mathcal{N}(\mu, \Sigma), \quad \mu \in \mathbb{R}^n, \quad \Sigma \in \mathbb{R}^{n \times n}$$

Assuming Σ is p.d., the log-likelihood function is given by

$$-\frac{n}{2}\log(2\pi) - \frac{\log(det(\Sigma))}{2} - \frac{1}{2}(y-\mu)'\Sigma^{-1}(y-\mu).$$

Here are some questions to consider:

- Suppose we know μ and Σ, how do you evaluate the red part? (thinking about taking inverse?) Whenever we see matrix inverse, we should think in terms of solving linear equations. Consider using Cholesky decomposition.

- How about the blue part? Using the fact $\log(det(\Sigma)) = 2\sum_{k=1}^{n} \log L_{kk}$, why?

Therefore, consider Method 1 (for someone does not take this class): Compute $\Sigma^{-1} \rightarrow$ Compute quadratic form $\rightarrow$ Compute determinant

Consider Method 2 using Cholesky: 1) Do Cholesky decomposition $\Sigma = LL'$, 2) solve $x : Lx = y - \mu$ by forward substitution, 3) compute quadratic form $x'x$; 4) Compute determinant from Cholesky factor. How many flops for Method 1 & 2?

- Method 1: 1) $2n^3/3$ flops; 2) $O(n^2)$ flops; 3) $2n^3/3 + O(n^2)$ flops $4n^3/3 + O(n^2) = \mathcal{O}(4n^3/3)$
- Method 2: 1) $n^3/3$ flops; 2) n^2 flops; 3) $O(n)$ flops; 4) $O(n)$ flops $n^3/3 + n^2 + O(n) = \mathcal{O}(n^3/3)$

1.2 Linear regression by Cholesky

The goal is to solve

$$X'X\beta = X'y$$

Assume $X \in \mathbb{R}^{n\times p}$ is a full column rank matrix. It is easy to work with the **augmented matrix**

$$\begin{pmatrix} X'X & X'y \\ y'X & y'y \end{pmatrix} = \begin{pmatrix} L & 0 \\ \ell' & d \end{pmatrix} \times \begin{pmatrix} L' & \ell \\ 0' & d \end{pmatrix} = \begin{pmatrix} LL' & L\ell \\ \ell'L' & \ell'\ell + d \end{pmatrix}$$

By Cholesky, $X'X\beta = LL'\beta = X'y = L\ell$ and $L'\beta = \ell$. This solves β in p^2 flops.

Next,since $\ell = L^{-1}X'y$, we have

$$\ell'\ell = yX(LL')^{-1}X'y = y'X(X'X)^{-1}X'y = y'P_xy = y'P_xP_xy = \hat{y}'\hat{y}$$

and

$$d^2 = y'y - \ell'\ell = y'(I - P_x)y = (y - \hat{y})'(y - \hat{y}) = SSE$$

We thus solved β, $\hat{y}$, and SSE all altogether. In R: chol2inv() and in python: cho_solve()

Some notes: We only do inversion if standard errors are needed, $(X'X)^{-1} = (LL')^{-1} = (L^{-1})'L^{-1}$

To summarize, how many flops for using Cholesky to solve linear regression?

- Form the lower triangular part of $(X, y)'(X, y)$: $n(p + 1)^2/2$ flops (why?)
- Cholesky decomposition of the augmented matrix, $(p + 1)^3/3$ flops
- Solve $L'\beta = l$ for $\hat{\beta}$: p^2 flops
- If need the standard error, estimate $\hat{\sigma} = d^2/(n-p)$ and compute $\hat{\sigma}^2(X'X)^{-1} = \hat{\sigma}^2(LL')^{-1}$: $2p^3/3$ flops

When n, p are large, total cost is $O(p^3/3) + O(np^2/2)$ flops (without s.e.) and $O(p^3) + O(np^2/2)$ flops (with s.e.).

What about using LU? The total flops is $2p^3/3 + p^2 + np^2/2$ (without s.e.) and $> 2p^3 + np^2/2$ (with s.e.)

2 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, $A = Q_1R_1$, $Q_1 \in \mathbb{R}^{n\times p}$, $Q_1'Q_1 = I_p$ 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 6.1 *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 $A'A$.*

2.1 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 $SSE = d^2$.

3 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 and modified Gram-Schmidt
- Householder transformation
- Givens transformation

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

...

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,

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

Classical

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

Modified

for k=1:n,
  $w = a_k$
  for j=1:k-1,

    $r_{jk} = q_j^T w$
    $w = w - r_{jk} q_j$

  end
  $r_{kk} = \|w\|_2$
  $q_k = w/r_{kk}$
end

Figure 6.1: Comparing G-S and modified G-S algorithms. To compare with the notations in Section 3.1, w is u, q_j is e_j and $r_{jk} = \langle a_k, e_j\rangle$.

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

4 The G-S and Modified G-S algorithms

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 $\sum_{k=1}^{p} 2n(k-1) \approx np^2$.

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