AMS526: Numerical Analysis I (Numerical Linear Algebra for Computational and Data Sciences)

Lecture 11: Householder Triangulation; Givens Rotation; Least Squares Problems

Xiangmin Jiao

Stony Brook University

Outline

1. Householder Triangularization (NLA§10)
2. Givens Rotations
3. Linear Least Squares Problems (NLA§11)

Gram-Schmidt as Triangular Orthogonalization

- Every step of Gram-Schmidt can be viewed as multiplication with triangular matrix. For example, at first step:

$$[v_1|v_2|\cdots|v_n]\underbrace{\begin{bmatrix} \frac{1}{r_{11}} & \frac{-r_{12}}{r_{11}} & \frac{-r_{13}}{r_{11}} & \cdots \\ & 1 & & \\ & & 1 & \\ & & & \ddots \end{bmatrix}}_{R_1} = \left[q_1|v_2^{(2)}|\cdots|v_n^{(2)}\right],$$

- Gram-Schmidt therefore multiplies triangular matrices to orthogonalize column vectors, and in turns can be viewed as *triangular orthogonalization*

$$A\underbrace{R_1 R_2\cdots R_n}_{\widehat{R}^{-1}} = \widehat{Q}$$

  where R_i is triangular matrix
- A "dual" approach would be *orthogonal triangularization*, i.e., multiply A by orthogonal matrices to make it triangular matrix

Householder Triangularization

- Method introduced by Alston Scott Householder in 1958
- It multiplies orthogonal matrices to make column triangular, e.g.

$$\underset{A}{\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}} \xrightarrow{Q_1} \underset{Q_1 A}{\begin{bmatrix} \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{x} & \mathbf{x} \end{bmatrix}} \xrightarrow{Q_2} \underset{Q_2 Q_1 A}{\begin{bmatrix} \times & \times & \times \\ & \mathbf{x} & \mathbf{x} \\ & 0 & \mathbf{x} \\ & 0 & \mathbf{x} \\ & 0 & \mathbf{x} \end{bmatrix}} \xrightarrow{Q_3} \underset{Q_3 Q_2 Q_1 A}{\begin{bmatrix} \times & \times & \times \\ & \times & \times \\ & & \mathbf{x} \\ & & 0 \\ & & 0 \end{bmatrix}}$$

- After n steps, we get a product of orthogonal matrices

$$\underbrace{Q_n \cdots Q_2 Q_1}_{Q^T} A = R$$

  and in turn we get full QR factorization $A = QR$
- Q_k introduces zeros below diagonal of kth column while preserving zeros below diagonal in preceding columns
- The key question is how to find Q_k

Householder Reflectors

- First, consider Q_1: $Q_1 a_1 = \|a_1\| e_1$, where $e_1 = (1, 0, \cdots, 0)^T$. Why the length is $\|a_1\|$?
- Q_1 reflects a_1 across hyperplane H orthogonal to $v = \|a_1\| e_1 - a_1$, and therefore
$$Q_1 = I - 2\frac{vv^T}{v^T v}$$
- More generally,
$$Q_k = \begin{bmatrix} I & 0 \\ 0 & F \end{bmatrix}$$
where I is $(k-1) \times (k-1)$ and F is $(m-k+1) \times (m-k+1)$ such that $Fx = \|x\|_2 e_1$, where x is $(a_{k,k}, a_{k,k+1}, \cdots, a_{k,m})^T$
- What is F? It has similar form as Q_1 with $v = \|x\| e_1 - x$.

Choice of Reflectors

- We could choose F such that $Fx = -\|x\|e_1$ instead of $Fx = \|x\|e_1$
  - More generally, $Fx = z\|x\|e_1$ with $|z| = 1$ if $z \in \mathbb{C}$
  - This leads to an infinite number of possible QR factorizations of A
  - If we require $z \in \mathbb{R}$, we still have two choices
- Numerically, it is undesirable for $v^T v$ to be close to zero for $v = z\|x\|e_1 - x$, and $\|v\|$ is larger if $z = -\text{sign}(x_1)$
- Therefore, $v = -\text{sign}(x_1)\|x\|e_1 - x$. Since $I - 2\frac{vv^T}{v^T v}$ is independent of sign, clear out the factor -1 and obtain $v = \text{sign}(x_1)\|x\|e_1 + x$
- For completeness, if $x_1 = 0$, set z to 1 (instead of 0)
- We define $\text{sign}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{otherwise} \end{cases}$

Householder Algorithm

Householder QR Factorization

for $k = 1$ **to** n

$$x = A_{k:m,k}$$
$$v_k = \textbf{sign}(x_1)\|x\|e_1 + x$$
$$v_k = v_k/\|v_k\|$$
$$A_{k:m,k:n} = A_{k:m,k:n} - 2v_k(v_k^T A_{k:m,k:n})$$

- Leave R in place of A
- Matrix Q is not formed explicitly but reflection vector v_k is stored

Householder Algorithm

Householder QR Factorization

for $k = 1$ **to** n

$$x = A_{k:m,k}$$
$$v_k = \textbf{sign}(x_1)\|x\|e_1 + x$$
$$v_k = v_k / \|v_k\|$$
$$A_{k:m,k:n} = A_{k:m,k:n} - 2v_k(v_k^T A_{k:m,k:n})$$

- Leave R in place of A
- Matrix Q is not formed explicitly but reflection vector v_k is stored
- Question: Can A be reused to store both R and v_k completely?

Householder Algorithm

Householder QR Factorization

for $k = 1$ **to** n

$$x = A_{k:m,k}$$
$$v_k = \textbf{sign}(x_1)\|x\|e_1 + x$$
$$v_k = v_k / \|v_k\|$$
$$A_{k:m,k:n} = A_{k:m,k:n} - 2v_k(v_k^T A_{k:m,k:n})$$

- Leave R in place of A
- Matrix Q is not formed explicitly but reflection vector v_k is stored
- Question: Can A be reused to store both R and v_k completely?
- Answer: We can use lower triangular portion of A to store all but one entry in each v_k. An additional array of size n may be needed, or take advantage of $\|v_k\| = 1$

Householder Algorithm

Householder QR Factorization

for $k = 1$ **to** n

$$x = A_{k:m,k}$$
$$v_k = \textbf{sign}(x_1)\|x\|e_1 + x$$
$$v_k = v_k / \|v_k\|$$
$$A_{k:m,k:n} = A_{k:m,k:n} - 2v_k(v_k^T A_{k:m,k:n})$$

- Leave R in place of A
- Matrix Q is not formed explicitly but reflection vector v_k is stored
- Question: Can A be reused to store both R and v_k completely?
- Answer: We can use lower triangular portion of A to store all but one entry in each v_k. An additional array of size n may be needed, or take advantage of $\|v_k\| = 1$
- Question: What happens if v_k is 0 in line 3 of the loop?

Householder Algorithm

Householder QR Factorization

for $k = 1$ **to** n

$$x = A_{k:m,k}$$
$$v_k = \textbf{sign}(x_1)\|x\|e_1 + x$$
$$v_k = v_k / \|v_k\|$$
$$A_{k:m,k:n} = A_{k:m,k:n} - 2v_k(v_k^T A_{k:m,k:n})$$

- Leave R in place of A
- Matrix Q is not formed explicitly but reflection vector v_k is stored
- Question: Can A be reused to store both R and v_k completely?
- Answer: We can use lower triangular portion of A to store all but one entry in each v_k. An additional array of size n may be needed, or take advantage of $\|v_k\| = 1$
- Question: What happens if v_k is 0 in line 3 of the loop?
- Answer: $r_{kk} = 0$

Applying or Forming Q

- Compute $Q^T b = Q_n \cdots Q_1 b$

> Implicit calculation of $Q^T b$
>
> **for** $k = 1$ **to** n
>
> $$b_{k:m} = b_{k:m} - 2v_k(v_k^T b_{k:m})$$

Applying or Forming Q

- Compute $Q^T b = Q_n \cdots Q_1 b$

> Implicit calculation of $Q^T b$
>
> **for** $k = 1$ **to** n
>
> $\qquad b_{k:m} = b_{k:m} - 2v_k(v_k^T b_{k:m})$

- Compute $Qx = Q_1 Q_2 \cdots Q_n x$

> Implicit calculation of Qx
>
> **for** $k = n$ **downto** 1
>
> $\qquad x_{k:m} = x_{k:m} - 2v_k(v_k^T x_{k:m})$

- Question: How to form Q and $\hat{Q}$, respectively?

Applying or Forming Q

- Compute $Q^T b = Q_n \cdots Q_1 b$

> Implicit calculation of $Q^T b$
>
> **for** $k = 1$ **to** n
>
> $$b_{k:m} = b_{k:m} - 2v_k(v_k^T b_{k:m})$$

- Compute $Qx = Q_1 Q_2 \cdots Q_n x$

> Implicit calculation of Qx
>
> **for** $k = n$ **downto** 1
>
> $$x_{k:m} = x_{k:m} - 2v_k(v_k^T x_{k:m})$$

- Question: How to form Q and $\hat{Q}$, respectively?
- Answer: Apply $x = I_{m\times m}$ or first n columns of I, respectively

Operation Count

- Most work done at step $A_{k:m,k:n} = A_{k:m,k:n} - 2v_k(v_k^T A_{k:m,k:n})$
- Flops per iteration:
  - $\sim 2(m-k)(n-k)$ for dot products $v_k^T A_{k:m,k:n}$
  - $\sim (m-k)(n-k)$ for outer product $2v_k(\cdots)$
  - $\sim (m-k)(n-k)$ for subtraction
  - $\sim 4(m-k)(n-k)$ total
- Including outer loop, total flops is

$$\begin{aligned}\sum_{k=1}^{n} 4(m-k)(n-k) &= 4\sum_{k=1}^{n}(mn - km - kn + k^2) \\ &\sim 4mn^2 - 4(m+n)n^2/2 + 4n^3/3 \\ &= 2mn^2 - \frac{2}{3}n^3\end{aligned}$$

If $m \approx n$, it is more efficient than Gram-Schmidt method, but is similar to Gram-Schmidt if $m \gg n$

Outline

1. Householder Triangularization (NLA§10)
2. Givens Rotations
3. Linear Least Squares Problems (NLA§11)

Givens Rotations

- Instead of using reflection, we can rotate x to obtain $\|x\| e_1$
- A Given rotation $R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ rotates $x \in \mathbb{R}^2$ counterclockwise by θ
- Choose θ to be angle between $(x_i, x_j)^T$ and $(1,0)^T$, and we have

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_i \\ x_j \end{bmatrix} = \begin{bmatrix} \sqrt{x_i^2 + x_j^2} \\ 0 \end{bmatrix}$$

  where

$$\cos\theta = \frac{x_i}{\sqrt{x_i^2 + x_j^2}}, \ \sin\theta = \frac{-x_j}{\sqrt{x_i^2 + x_j^2}}$$

Givens QR

- Introduce zeros in column bottom-up, one zero at a time

$$\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \overset{(4,5)}{\longrightarrow} \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{x} & \mathbf{x} \end{bmatrix} \overset{(3,4)}{\longrightarrow} \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{x} & \mathbf{x} \\ & \times & \times \end{bmatrix} \cdots$$

- To zero a_{ij}, left-multiply matrix F with $F_{i:i+1,i:i+1}$ being rotation matrix and $F_{kk} = 1$ for $k \neq i, i+1$
- Flop count of Givens QR is $3mn^2 - n^3$, which is about 50% more expensive than Householder triangularization

Adding a Row

- Suppose $A \in \mathbb{R}^{m \times n}$ with $m \geq n$, and A has full rank
- Let $\tilde{A} = \begin{bmatrix} A_1 \\ z^T \\ A_2 \end{bmatrix}$, where $A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$ and z^T is a new row inserted
- Obtain $\tilde{A} = \tilde{Q}\tilde{R}$ from $A = QR$ efficiently using Givens rotation:
  - Suppose $A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} R$.
  - Then $\tilde{A} = \begin{bmatrix} A_1 \\ z^T \\ A_2 \end{bmatrix} = \begin{bmatrix} 0 & Q_1 \\ 1 & 0^T \\ 0 & Q_2 \end{bmatrix} \begin{bmatrix} z^T \\ R \end{bmatrix}$
  - Perform series of Givens rotation $\tilde{R} = U_n^T \dots U_2^T U_1^T \begin{bmatrix} z^T \\ R \end{bmatrix}$, and

    then $\tilde{Q} = \begin{bmatrix} 0 & Q_1 \\ 1 & 0^T \\ 0 & Q_2 \end{bmatrix} U_1 U_2 \dots U_n$
  - Updating $\tilde{R}$ costs $3n^2$ flops, and updating $\tilde{Q}$ costs $6mn$ flops

Adding a Column

- Suppose $A \in \mathbb{R}^{m\times n}$ with $m \geq n$, and A has full rank
- Let $\tilde{A} = \begin{bmatrix} A_1 & z & A_2 \end{bmatrix}$, where $A = \begin{bmatrix} A_1 & A_2 \end{bmatrix}$ and z is new column
- Obtain $\tilde{A} = \tilde{Q}\tilde{R}$ from $A = QR$ efficiently using Givens rotation:
  - Suppose $A = \begin{bmatrix} A_1 & A_2 \end{bmatrix} = Q \begin{bmatrix} R_1 & R_2 \end{bmatrix}$
  - Then $\tilde{A} = \begin{bmatrix} A_1 & z & A_2 \end{bmatrix} = Q \begin{bmatrix} R_1 & w & R_2 \end{bmatrix}$, where $w = Q^T z$
  - Perform series of Givens rotation $\tilde{R} = U_{k+1} \cdots U_n \begin{bmatrix} R_1 & w & R_2 \end{bmatrix}$, where U_n performs on rows n and $n-1$, U_{n-1} performs on rows $n-1$ and $n-2$, etc.
  - $\tilde{Q} = QU_n^T \cdots U_{k+1}^T$
  - It takes $O(mn)$ time overall

Outline

1. Householder Triangularization (NLA§10)
2. Givens Rotations
3. Linear Least Squares Problems (NLA§11)

Linear Least Squares Problems

- Overdetermined system of equations $Ax \approx b$, where A has more rows than columns and has full rank, in general has no solutions
- Example application: Polynomial least squares fitting
- In general, minimize the residual $r = b - Ax$
- In terms of 2-norm, we obtain linear least squares problem: Given $A \in \mathbb{R}^{m \times n}$, $m \geq n$, and $b \in \mathbb{R}^m$, find $x \in \mathbb{R}^n$ such that $\|b - Ax\|_2$ is minimized
- If A has full rank, the minimizer x is the solution to the normal equation

$$A^T A x = A^T b$$

or in terms of the *pseudoinverse* A^+,

$$x = A^+ b, \quad \text{where } A^+ = (A^T A)^{-1} A^T \in \mathbb{R}^{n \times m}$$

Geometric Interpretation

- Ax is in range(A), and the point in range(A) closest to b is its orthogonal projection onto range(A)
- Residual r is then orthogonal to range(A), and hence $A^T r = A^T(b - Ax) = 0$
- Ax is orthogonal projection of b, where $x = A^+ b$, so $P = AA^+ = A(A^T A)^{-1}A^T$ is orthogonal projection

Solution of Lease Squares Problems

- One approach is to solve normal equation $A^T Ax = A^T b$ directly using Cholesky factorization
  - Is unstable, but is very efficient if $m \gg n$ $(mn^2 + \frac{1}{3}n^3)$
- More robust approach is to use QR factorization $A = \hat{Q}\hat{R}$
  - b can be projected onto range(A) by $P = \hat{Q}\hat{Q}^T$, and therefore $\hat{Q}\hat{R}x = \hat{Q}\hat{Q}^T b$
  - Left-multiply by $\hat{Q}^T$ and we get $\hat{R}x = \hat{Q}^T b$ (note $A^+ = \hat{R}^{-1}\hat{Q}^T$)

> Least squares via QR Factorization
>
> Compute reduced QR factorization $A = \hat{Q}\hat{R}$
> Compute vector $c = \hat{Q}^T b$
> Solve upper-triangular system $\hat{R}x = c$ for x

- Computation is dominated by QR factorization $(2mn^2 - \frac{2}{3}n^3)$
- Question: If Householder QR is used, how to compute $\hat{Q}^T b$?

Solution of Lease Squares Problems

- One approach is to solve normal equation $A^T Ax = A^T b$ directly using Cholesky factorization
  - Is unstable, but is very efficient if $m \gg n$ ($mn^2 + \frac{1}{3}n^3$)
- More robust approach is to use QR factorization $A = \hat{Q}\hat{R}$
  - b can be projected onto range(A) by $P = \hat{Q}\hat{Q}^T$, and therefore $\hat{Q}\hat{R}x = \hat{Q}\hat{Q}^T b$
  - Left-multiply by $\hat{Q}^T$ and we get $\hat{R}x = \hat{Q}^T b$ (note $A^+ = \hat{R}^{-1}\hat{Q}^T$)

> Least squares via QR Factorization
>
> Compute reduced QR factorization $A = \hat{Q}\hat{R}$
>
> Compute vector $c = \hat{Q}^T b$
>
> Solve upper-triangular system $\hat{R}x = c$ for x

- Computation is dominated by QR factorization ($2mn^2 - \frac{2}{3}n^3$)
- Question: If Householder QR is used, how to compute $\hat{Q}^T b$?
- Answer: Compute $Q^T b$ (where Q is from full QR factorization) and then take first n entries of resulting $Q^T b$