Welcome to Ma221! Lecture 12, Fall24

Solving $\underset{x}{\mathrm{argmin}} \|Ax - b\|_2$ $\quad A^{m \times n}$ $\quad m \geq n$

full rank

Normal Equations: $A^T A x = A^T b$

QR : $A = QR$ cols of Q orthonormal

$$x = R^{-1} Q^T b$$

lots of variants: differ in stability

2 goals

$$\frac{\|A - QR\|}{\|A\|} = O(\varepsilon)$$

$$\|Q^T Q - I\| = O(\varepsilon)$$

SVD: $A^{m \times n} = U^{m \times m} \cdot \Sigma \cdot V^{T\, n \times n}$, U, V orthog

$$= \begin{array}{c} \\ m \end{array} \overset{\quad n \quad\;\; m-n}{\begin{bmatrix} U_1 & U_2 \end{bmatrix}} \begin{array}{c} n \\ m-n \end{array} \overset{n}{\begin{bmatrix} \hat{\Sigma} \\ \end{bmatrix}} \cdot \overset{n}{\begin{bmatrix} V^T \end{bmatrix}}$$

$$x = V \hat{\Sigma}^{-1} U_1^T b$$

Moore-Penrose Pseudo inverse

$$A^+ = V \hat{\Sigma}^{-1} U_1^T$$

Perturbation Theory for LS

How much can x change
when A and b change?

When $m=n$, same as $A^{-1}b$
so expect $\kappa(A)$ to appear

Another source of ill conditioning

$$A = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ b_2 \end{bmatrix} \quad x = A^{+}b = [1,0]\begin{bmatrix} 0 \\ b_2 \end{bmatrix} = 0$$

$$A = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \quad x = A^{+}b = [1,0]\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = b_1$$

so large relative change

In general, ill-conditioned if
b (nearly) orthogonal to span(A)

$$x+e = \operatorname{argmin} \| (A+\delta A)(x+e) - (b+\delta b)\|_2$$

$$x = \operatorname{argmin} \| Ax - b\|_2$$

$$e = (x+e) - x = ((A+\delta A)^T(A+\delta A))^{-1}(A+\delta A)(b+\delta b) - (A^TA)^{-1}A^Tb$$

$$(A^TA + \Delta)^{-1} = \left[(A^TA)(I + (A^TA)^{-1}\Delta)\right]^{-1}$$

$$= \underbrace{(I + (A^TA)^{-1}\Delta)^{-1}}_{I - X}(A^TA)^{-1}$$

$$(I-X)^{-1} = I + X + \|X\|^2$$

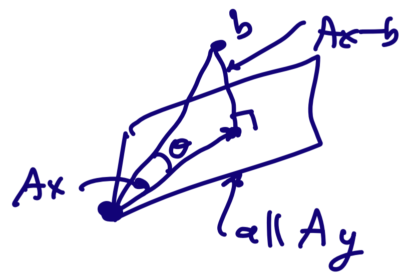
and only keeps terms with one small factor δA, δb

Def: $\varepsilon = \max\left(\frac{\|\delta A\|_2}{\|A\|_2}, \frac{\|\delta b\|_2}{\|b\|_2}\right)$

$$\frac{\|e\|}{\|x\|} \leq \varepsilon\left(2 \cdot \kappa(A) \cdot \frac{1}{\cos\theta} + \tan\theta \cdot \kappa^2(A)\right) + O(\varepsilon^2)$$

$\theta = \text{angle}(b, Ax)$

$$\sin\theta = \frac{\|Ax-b\|_2}{\|b\|_2}$$



$\theta = 0 \Rightarrow$ solve $Ax=b$ exactly

$$\frac{\|e\|}{\|x\|} \leq \varepsilon \cdot (2 \cdot \kappa(A))$$

when is error large?

1. $\kappa(A)$ large
2. θ near $\frac{\pi}{2}$, $\Rightarrow \frac{1}{\cos\theta} \sim \tan\theta \sim \infty$
3. error like $\kappa^2(A)$ when θ not near 0

Stable algorithms for QR

MGS and CGS not stable

$\|Q^TQ - I\|$ can be close to 1

Need Householder factorization (or Givens) also basis for eig( ), svd( )

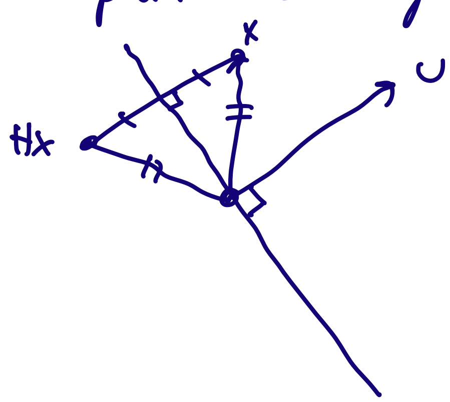
Idea: represent Q as product
$Q = Q_1 \cdot Q_2 \cdots Q_n$ of "simple"
orthogonal matrices, chosen to
make progress in reducing A to R.
since each Q_i orthogonal, so is Q

2 kinds of "simple" Q_i's
Householder Tranforms ("reflections")
Givens Rotations

Householder reflection $\quad H = I - 2uu^T$
$\|u\|_2 = 1$

$$HH^T = (I - 2uu^T)(I - 2uu^T)$$
$$= I - 4uu^T + 4u\underbrace{u^T u}_{1}u^T$$
$$= I$$

Reflection: Hx is reflection of x
in plane orthogonal to u



Given x, want to choose u so Hx has zeroes in certain locations

$$Hx = \begin{bmatrix} c \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix} = c \cdot e_1 \implies c = \pm \|x\|_2$$

$$\|Hx\|_2 = |c| = \|x\|_2$$

$$Hx = (I - 2uu^T)x = x - 2u(u^T x) = c \cdot e_1$$

$$u = \frac{x - ce_1}{2u^T x}$$

denominator scalar, choose it so $\|u\|_2 = 1$:

$$y = x - ce_1 = x \pm \|x\|_2 e_1$$

$$u = \frac{y}{\|y\|_2} = \text{House}(x)$$

How to pick sign: choose sign to avoid cancellation, which could cause numerical instability:

$$y = \begin{bmatrix} x_1 + \text{sign}(x_1) \cdot \|x\|_2 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$

$$\underbrace{\left[\begin{array}{c|c} I_3 & 0 \\ \hline 0 & H^{2\times 2}\end{array}\right]\left[\begin{array}{c|c} I_2 & 0 \\ \hline 0 & H^{3\times 3}\end{array}\right]\left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & H^{4\times 4}\end{array}\right] H^{5\times 5}}_{\text{product of orthogonal matrices so orthogonal}} \begin{bmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{bmatrix} = R$$

$$Q^T \cdot A = R$$

for $i = 1$ to $\min(m-1, n)$ ... only need to do last column if $m > n$

$$u(i) = \text{House}(A(i:m, i))$$

$$A(i:m, i:n) = (I - 2u(i)u(i)^T) \cdot A(i:m, i:n)$$

$$= \underbrace{A(i:m, i:n) - 2u(i)\underbrace{(u(i)^T A(i:m, i:n))}_{\text{vector matrix product}}}_{\text{rank 1 update}}$$

never form $H = I - 2uu^T$ or multiply them to get Q, represent Q implicitly by u vectors

Same trick as GE: R overwrites A, $u(i)$ vectors overwrite "zero entries" below diagonal

$$\text{Cost} = \sum_{i=1}^{n} 4(m-i+1)(n-i+1) = 2n^2 m - \frac{2}{3}n^3$$

+ lower order terms

$m \gg n$: dominated by $2n^2 m$

$m = n$: cost $= \frac{4}{3}n^3$, twice GE

Implicit representation of Q

$$Q_n \cdots Q_2 Q_1 A = R$$

$$\begin{aligned} A &= Q_1^T Q_2^T \cdots Q_n^T R \\ &= Q_1 Q_2 \cdots Q_n R \\ &= QR \end{aligned}$$

Solve LS problem: $x = R^{-1} Q^T b$

for $i = 1$ to n

$$b = Q_i b = (I - 2u(i)u(i)^T)b$$

$$= \underbrace{b + (\underbrace{-2u(i)^T b}_{\text{dot product}}) \cdot u(i)}_{\text{saxpy}}$$

$x = R^{-1} b$ by substitution

cost $= O(m \cdot n)$, much less than QR

Optimizing QR

so far: simple BLAS2 + BLAS1 version

use same optimizations as for LU + Cholesky:

Goal: # words moved $= \Omega\left(\frac{\#\text{flops}}{\sqrt{\text{cache size}}}\right)$

1. do QR on left part of matrix
   (could be block of b columns, or left half, if recursive)
2. update right part using Q from left part
3. do QR on right part

Need to do 2 using matmul

Need to multiply $Q = Q_b Q_{b-1} \cdots Q_1$ using a few matmuls

Thm (see Q3.17 for details)

if $Q_i = I - 2u_iu_i^T$, then

$$Q_b \cdots Q_1 = Q = \underset{n\times n}{I} - \underset{n\times b}{Y}\,\underset{b\times b}{T}\,\underset{b\times n}{Y^T}$$

$Y = [u_1, \dots, u_b]$

T can be computed from u_i

One more case to optimize
"Tall Skinny QR" TSQR $m >> n$

$n^2 \le$ cache size $= M$

$$\text{lower bound} = O\left(\frac{\#\text{flops}}{\sqrt{M}}\right) = O\left(\frac{mn^2}{\sqrt{M}}\right)$$

$$\le O\left(\frac{mn^2}{n}\right) = O(m \cdot n) = O(\text{size of input})$$

can't do better than $m \cdot n$, need to read whole input

New goal: read data once, get QR

Ex: suppose we can fit $\frac{1}{3}$ of A into cache

$$A = \begin{array}{c} n \\ \begin{bmatrix} A_1 \\ A_2 \\ A_3 \end{bmatrix} \end{array} \begin{array}{l} \\ m/3 \\ m/3 \\ m/3 \end{array}$$

... read A_1 into cache, do QR

$$= \begin{bmatrix} Q_1 R_1 \\ A_2 \\ A_3 \end{bmatrix} = \begin{bmatrix} Q_1 & & \\ & I & \\ & & I \end{bmatrix} \begin{bmatrix} R_1 \\ A_2 \\ A_3 \end{bmatrix} = \hat{Q}_1 \begin{bmatrix} R_1 \\ A_2 \\ A_3 \end{bmatrix}$$

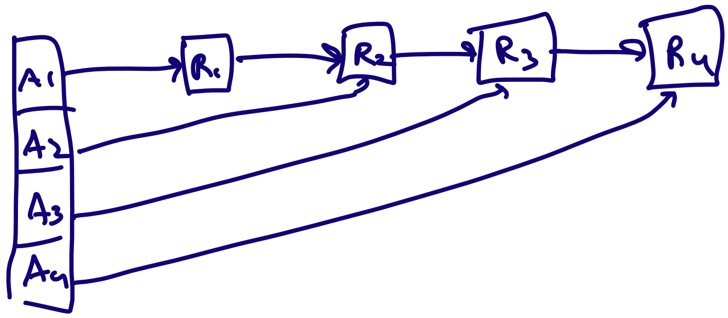
... read A_2 into cache, do QR on $\begin{bmatrix} R_1 \\ A_2 \end{bmatrix}$

$$= \hat{Q}_1 \begin{bmatrix} \begin{bmatrix} R_1 \\ A_2 \end{bmatrix} \\ A_3 \end{bmatrix} = \hat{Q}_1 \begin{bmatrix} Q_2 R_2 \\ A_3 \end{bmatrix} = \hat{Q}_1 \begin{bmatrix} Q_2 & \\ & I \end{bmatrix} \begin{bmatrix} R_2 \\ A_3 \end{bmatrix}$$

$$= \hat{Q}_1 \hat{Q}_2 \begin{bmatrix} R_2 \\ A_3 \end{bmatrix}$$

... read A_3 into cache, do QR on $\begin{bmatrix} R_2 \\ A_3 \end{bmatrix}$

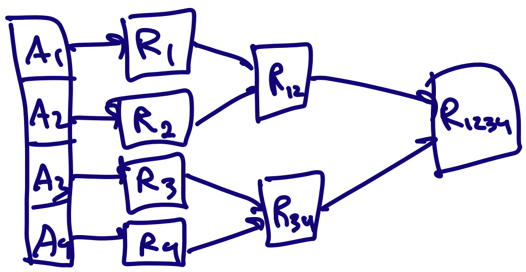
$$= \hat{Q}_1 \hat{Q}_2 \hat{Q}_3 R_3$$



sequential QR

Same idea for parallel TSQR

A_i on proc. P_i

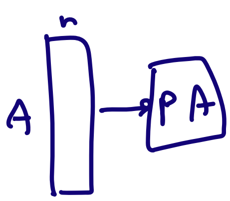


"map reduce" in cloud

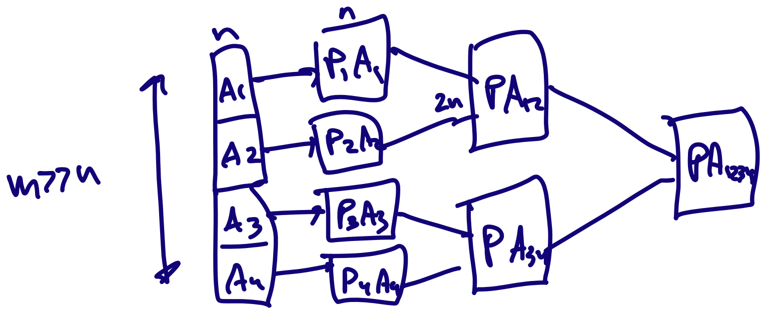
Same idea for partial pivoting on a Tall Skinny matrix (TSLU)

one communication for many columns versus finding max in column by column

Basis operation



select subset of rows of A by usual partial pivoting "most linearly independent rows of subset of A it owns"



"Tournament Pivoting"

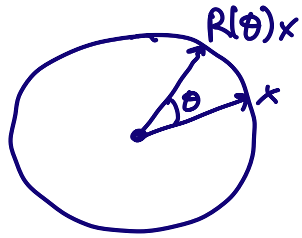
TSLU

Givens rotations:

simple orthogonal transform

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$R(\theta)x$



$R(i,j,\theta)x$ applies rotation to x_i and x_j

$$\begin{array}{cc} & \begin{array}{cc} \quad i & \quad j \end{array} \\ \begin{array}{c} \\ i \\ j \\ \\ \end{array} & \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & \cos\theta & -\sin\theta & \\ & & \sin\theta & \cos\theta & \\ & & & & \ddots \\ & & & & & 1 \end{bmatrix} \end{array}$$

How to pick θ to zero out x_j

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

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

Can use Givens to do QR, no advantage over Householder in dense, maybe less fill-in in sparse case

---

Stability of Applying Orthogonal Matrices

Summary: Any algorithm that only multiplies by orthogonal matrices is backward stable

proof sketch: use basic rule $fl(a \text{ op } b) = (a \text{ op } b)(1+\delta)$, $|\delta| \le \varepsilon$

to show that applying one Householder or one Givens rotation gets small error

$$fl(Q' \cdot A) = Q' A + E, \quad \|E\| = O(\varepsilon) \cdot \|A\|$$

Q' "nearly orthogonal" ($\|u\|_2$ close to 1)

$$\Rightarrow Q' = Q + F, \quad \|F\| = O(\varepsilon), \quad Q^T Q = I$$

$$\begin{aligned} fl(Q' \cdot A) &= Q' \cdot A + E \\ &= (Q+F) \cdot A + E \\ &= Q \cdot A + F \cdot A + E \\ &= Q \cdot A + G \end{aligned}$$

= exact orthog transform of A + error G

$$\|G\| = \|FA + E\| \le \|FA\| + \|E\|$$

$$\leq \|F\| \cdot \|A\| + \|E\|$$
$$= O(\varepsilon) \cdot \|A\| + O(\varepsilon) \|A\|$$
$$= O(\varepsilon) \cdot \|A\|$$

$$fl(Q'A) = QA + G$$
$$= Q(A + Q^T G)$$
$$= Q(A + G')$$
$$\|G'\| = \|Q^T G\| = \|G\|$$

$\Rightarrow$ multiplication by Q' backward stable

What if we multiply by many Q's?