Welcome back to Ma221! Lecture 19, Oct 6

Optimizing QR

So far: simple BLAS2, want BLAS3

Same tricks as for LU + Cholesky
+ new ones

Goal: lower bound on

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

① do QR on left part of matrix
(could be b columns, or left half if recursive)

② update right part, using factorization of left part — using matmul

③ Do QR on right part

How to do ② using matmul?

convert $Q = Q_b \cdot Q_{b-1} \cdots Q_1$ to a few matmuls

$Q_i = I - 2 v_i \cdot v_i^T$

Thm: (Q3.17)

$$Q = I - Y \cdot T \cdot Y^T \qquad Y = [v_1, \ldots v_b]$$

T computable from Y

One more case: "Tall-Skinny A" $A^{m \times n}$ $m \gg n$

$n^2 \le M$ ← cache size

Lower bound $= \Theta\left(\frac{\#flops}{\sqrt{M}}\right) = \Theta\left(\frac{mn^2}{\sqrt{M}}\right)$

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

↑ can't move less data than this

new algorithm: Tall-Skinny QR (TSQR)

Suppose we can fit $\frac{1}{3}$ of A in cache

$$A = \begin{bmatrix} A_1 \\ A_2 \\ A_3 \end{bmatrix} \begin{matrix} m/3 \\ m/3 \\ m/3 \end{matrix} \quad \begin{matrix} \\ \\ \\ \end{matrix} \quad \begin{matrix} n \\ \\ \\ \end{matrix}$$

... read A_1 into cache, do QR

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

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

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

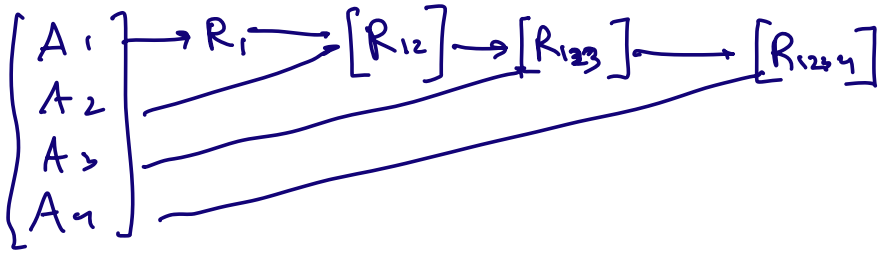
$$= \hat{Q}_1 \cdot \hat{Q}_2 \cdot \begin{bmatrix} R_2 \\ A_3 \end{bmatrix}$$

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

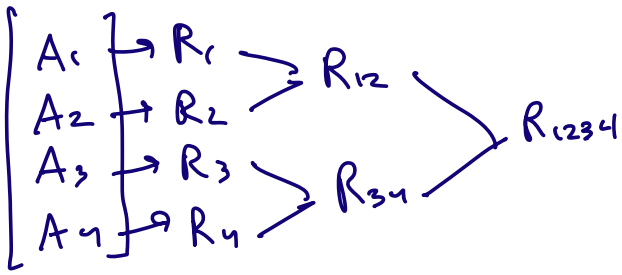
$= \hat{Q}_1 \cdot \hat{Q}_2 \cdot [Q_3 \cdot R_3] = \hat{Q}_1 \cdot \hat{Q}_2 \hat{Q}_3 \cdot R_3$

$= QR$, with implicit Q done!

Simpler notation:



Same idea for parallel QR

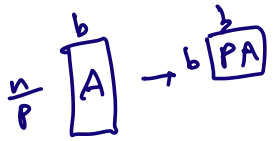


"map reduce"

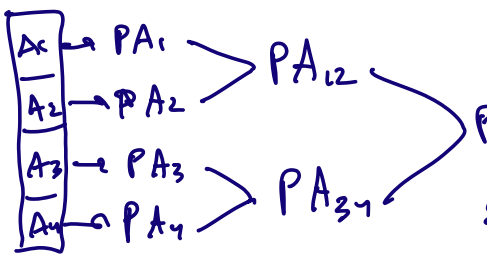
Same idea for accelerating GE"PP"

given n×b submatrix of A

choose b pivot rows



choose b "most independent" rows using GEPP



Tournament Pivoting

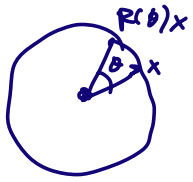
similar analysis for GEPP

slightly weaker

Givens Rotations: sparse QR
eigenproblems + SVD

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

$R(\theta)x$



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

Pick θ to zero out x_2

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \sqrt{x_1^2 + x_2^2} \\ 0 \end{bmatrix}$$

$$c = \cos\theta = \frac{x_1}{\sqrt{x_1^2 + x_2^2}} \qquad s = \frac{-x_2}{\sqrt{x_1^2 + x_2^2}}$$

Could do QR with Givens, no advantage over Householder for dense A, but could be less fill-in for sparse A

Stability of applying orthog. matrices
- one error analysis for QR, eigenproblems, SVD

Thm: Any algorithm that multiplies by orthog. matrices is backward stable

Proof sketch: use basic rule $fl(a \mathbin{op} b) = (a \mathbin{op} b)(1+\delta)$ to show that applying one orthog matrix

gives small error:

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

Ex: $(I - 2uu^T) \cdot A = A - 2u \cdot (u^T \cdot A)$

error $\varepsilon\|A\|$

error $\varepsilon\|A\|$

error $\varepsilon\|A\|$

Q' "nearly orthog" $\quad Q' = Q + F \qquad \|F\| = O(\varepsilon)$

$Q^T \cdot Q = I$

$$fl(Q' \cdot A) = Q' \cdot A + E = (Q + F) \cdot A + E$$

$$= Q \cdot A + F \cdot A + E = Q \cdot A + G$$

$$= \text{exact orthog trans} + G$$

$$\|G\| = \|F \cdot A + E\| \le \|F \cdot A\| + \|E\|$$

$$\le \|F\| \cdot \|A\| + \|E\|$$

$$= O(\varepsilon) \cdot \|A\|$$

$$\rightarrow fl(Q' \cdot A) = Q \cdot A + G = Q \cdot (A + Q^T G)$$

$$= Q \cdot (A + G')$$

$$\|G'\| = \|G\| = O(\varepsilon) \cdot \|A\|$$

Next: multiply by many Q_i'

$$fl(Q_3' \cdot (Q_2' \cdot (Q_1' \cdot A)))$$

$$= fl(Q_3' \cdot (Q_2' \cdot (Q_1 \cdot A + G_1)))$$

$$= fl(Q_3' \cdot (Q_2 \cdot (Q_1 \cdot A + G_1) + G_2))$$

$$= fl(Q_3 \cdot (Q_2 \cdot (Q_1 \cdot A + G_1) + G_2) + G_3)$$

$$= Q_3 Q_2 Q_1 A + \underbrace{Q_3 \cdot Q_2 \cdot G_1 + Q_3 \cdot G_2 + G_3}_{G'}$$

$$\|G'\| \le \|G_1\| + \|G_2\| + \|G_3\| = O(\varepsilon)\|A\| \quad \text{QED}$$

One more fast (but unstable) QR alg used in practice when we "know" A well-conditioned

Cholesky QR:

Factor $A^T A = R^T R$ using Cholesky

Form $Q = A \cdot R^{-1}$

can fail if during Cholesky a pivot is negative

Compare: Householder

CGS and CGS2

MGS and MGS2

$$A = Q_1 R_1 = (Q_2 R_2) R_1 = Q_2 (R_2 R_1)$$

CholQR and CholQR2

Measure: $\dfrac{\|A - QR\|}{\|A\|}$ $\|Q^T Q - I\|$

(Matlab demo)