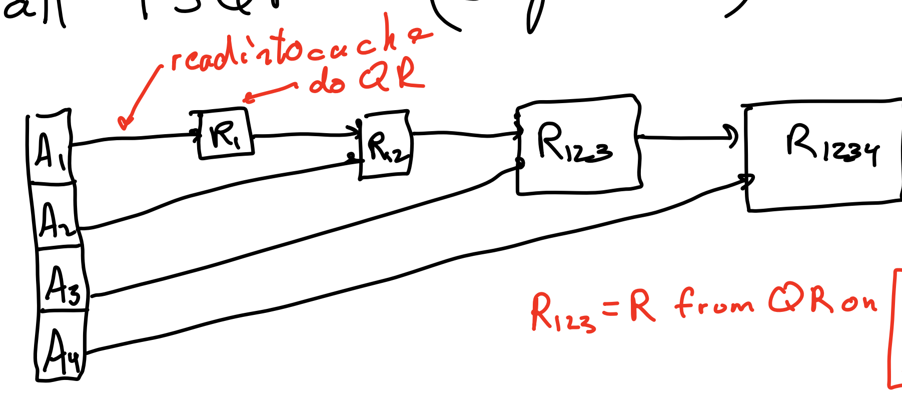
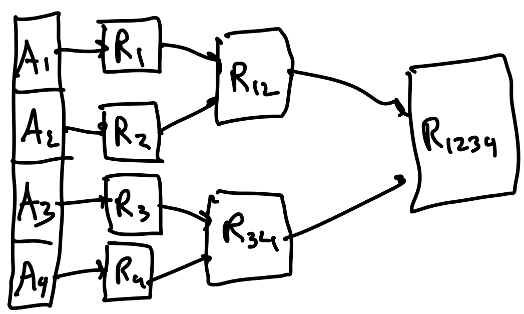
Welcome to Ma221! (Mar 1)

Recall TSQR (sequential)



$R_{123} = R$ from QR on $\begin{bmatrix} A_1 \\ A_2 \\ A_3 \end{bmatrix}$ etc

Same idea for parallel TSQR



"MapReduce" where QR is reduction operation

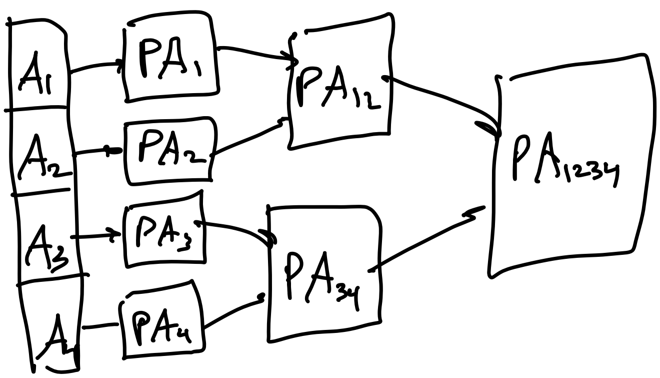
Same idea for partial pivoting on a Tall Skinny matrix (TSLU)

one communication tree for many columns

Basic operation

$A \to PA$

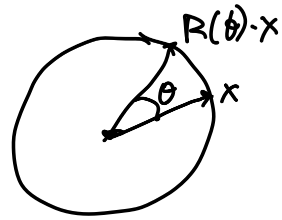
select subset of rows of A chosen by partial pivoting
"most linearly independant rows of A"



Givens rotations: another simple orthogonal tranformation:
used for 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 rotation to x_i and x_j

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

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 do QR using Given rotations, no advantage over Householder for dense A, maybe less fillin for sparse A

Stability of applying Orthogonal matrices

Summary: Any algorithm that just multiplies by a sequence of orthogonal matrices is backward stable

Proof sketch: use basic rule

$$fl(a \text{ op } b) = (a \text{ op } b)(1+\delta) \quad |\delta| \le \varepsilon$$

to show that multiplying by one Householder or Givens gets small error:

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

Q' "nearly orthogonal" $\Rightarrow$

$$Q' = Q + F, \quad \|F\| = O(\varepsilon)$$

Q exactly orthogonal

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

= exact orthogonal transformation + G

$$\|G\| = \|FA + E\| \le \|F\| \cdot \|A\| + \|E\|$$
$$\le O(\varepsilon)\|A\| + O(\varepsilon) \cdot \|A\| = O(\varepsilon) \cdot \|A\|$$

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

$$\|G'\| = \|G\|$$

$\Rightarrow$ multiplication by Q' is backward stable

= exact transform of $A + G'$

What if multiply by many Q_i' ?

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

$$= fl(Q_3'(Q_2'(Q_1 A + E_1)))$$

$$= fl(Q_3'(Q_2(Q_1 A + E_1) + E_2))$$

$$= (Q_3(Q_2(Q_1 A + E_1) + E_2) + E_3)$$

$$= \underbrace{Q_3 Q_2 Q_1}_{Q} \cdot A + \underbrace{Q_3 Q_2 E_1 + Q_3 E_2 + E_3}_{E}$$

$$\|E\| \le \|E_1\| + \|E_2\| + \|E_3\| = O(\varepsilon) \cdot \|A\|$$

$$= Q(A + Q^T E) \qquad \|Q^T E\| = \|E\| = O(\varepsilon) \cdot \|A\|$$

same idea (pick $A = I$) to show that product $fl(Q_3' Q_2' Q_1')$ nearly orthogonal

One more fast (but unstable) QR algorithm: used in practice when need Q, know A well conditioned

Cholesky QR:

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

form $Q = A \cdot R^{-1}$ since $A = QR$

can fail completely eg if some pivot during Cholesky is negative → fails, can't do twice

Dealing with (nearly) low rank matrices

Motivation: Real data often low rank (nearly redundant)

(1) take precautions to avoid inaccurate LS

(2) use it to compress data, go faster, both deterministically and randomized

Use LS as example of compression, but many applications

Solving a LS problem when A rank deficient

Thm: A $m \times n$, $m \ge n$, rank $r < n$

$$A = U \Sigma V^T = \begin{array}{c} \\ m \end{array} \begin{array}{c} \begin{array}{ccc} r & n-r & m-n \end{array} \\ \left[U_1, U_2, U_3 \right] \end{array} \begin{array}{c} \\ r \\ n-r \\ m-n \end{array} \begin{array}{c} \begin{array}{cc} r & n-r \end{array} \\ \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \\ 0 & 0 \end{bmatrix} \end{array} \begin{array}{c} \\ n \end{array} \begin{array}{c} \begin{array}{cc} r & n-r \end{array} \\ \left[V_1, V_2 \right]^T \end{array}$$

Σ_1 is full rank

$\Sigma_2 = 0$ (later: tiny)

The set of vectors minimizing
$\|Ax-b\|_2$ is

$$\{x = V_1 \Sigma_1^{-1} U_1^T b + V_2 y_2, \text{ any } y_2 \in \mathbb{R}^{n-r}\}$$

Unique x minimizing both $\|Ax-b\|_2$ and $\|x\|_2$ is gotten from $y_2=0$

$$x = V_1 \Sigma_1^{-1} U_1^T b$$

Def: $A^+ = V_1 \Sigma_1^{-1} U_1^T$ is Moore-Penrose pseudoinverse of A (includes full rank case, $r=n$)

(in practice: Σ_2 will be all singular values less than some user defined tolerance)

So square or not, full rank or not "best" solution is $x = A^+ b$

Proof: $\|Ax-b\|_2 = \|U\Sigma V^T x - b\|_2$

$= \|\Sigma V^T x - U^T b\|_2$ since U orthogonal

$= \|\Sigma y - U^T b\|_2$ where $y = V^T x$

$\|x\|_2 = \|y\|_2$ so okay to minimize either one

$$= \left\| \begin{bmatrix} \Sigma_1 y_1 - U_1^T b \\ -U_2^T b \\ -U_3^T b \end{bmatrix} \right\|_2 \quad \text{where } y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \begin{matrix} r \\ n-r \end{matrix}$$

minimized by $y_1 = \Sigma_1^{-1} U_1^T b$

and $\|y\|_2^2 = \|y_1\|_2^2 + \|y_2\|^2$

minimized by $y_2 = 0$

$$x = Vy = [V_1, V_2]\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = V_1 y_1 + V_2 y_2$$

$$= V_1 \Sigma_1^{-1} U_1^T b$$

Solving LS when A (nearly) rank deficient, using truncated SVD

$$\kappa(A) = \frac{\sigma_{max}}{\sigma_{min}} = \infty \text{ if } A \text{ rank deficient}$$

$$\underset{x}{\operatorname{argmin}} \left\| \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\|_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\underset{x}{\operatorname{argmin}} \left\| \begin{bmatrix} 1 & 0 \\ 0 & e \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\|_2 = \begin{bmatrix} 1 \\ 1/e \end{bmatrix}$$ e tiny

what does a "solution" mean if it can change discontinuously?

Often A not known exactly, just up to some tolerance, $\|A - A'\|_2 \le tol$

What to do?

Def: truncated SVD

if $A = U \Sigma V^T$

$$A(tol) = U \cdot \Sigma(tol) \cdot V^T$$

$$\Sigma(tol) = diag(\sigma_1(tol), \sigma_2(tol), \ldots, \sigma_n(tol))$$

$$\sigma_i(tol) = \begin{cases} \sigma_i & \text{if } \sigma_i \geq tol \\ 0 & \text{if } \sigma_i < tol \end{cases}$$

$A(tol)$ = lowest rank matrix within distance tol of A

Using $A(tol)$ for LS reduces

$$\kappa = \frac{\sigma_{max}}{\sigma_{min}} \quad \text{to} \quad \frac{\sigma_{max}}{tol}$$

tol is a "knob" for user to trade off sensitivity κ and how well LS problem can be solved (how small you can make residual)

Replacing A by an "easier" matrix called regularization, several mechanisms

Lemma: $x_1 = \arg\min_x \|A(tol)x - b_1\|_2$

$x_2 = \arg\min_x \|A(tol)x - b_2\|_2$

choose $\|x_i\|_2$ of smallest norm

Then $\|x_1 - x_2\| \le \dfrac{\|b_1 - b_2\|}{tol}$

proof: $\|x_1 - x_2\|_2 = \|(A(tol))^+ (b_1 - b_2)\|_2$

$= \|V(\Sigma(tol))^+ U^T(b_1 - b_2)\|_2$

$= \|\mathrm{diag}(\sigma_1^{-1}, \sigma_2^{-1}, \dots \sigma_k^{-1}, 0 \cdots 0) \cdot U^T(b_1 - b_2)\|_2$

↑ last nonzero σ_k where $\sigma_k \ge tol$

$\le \dfrac{1}{\sigma_k} \|U^T(b_1 - b_2)\|_2$

$\le \dfrac{1}{tol} \|b_1 - b\|_2$

How does $A(tol)$ depend on tol?

piecewise constant, changes when $tol = \sigma_i$

Other advantage of $A(tol)$:

setting more σ_i to 0, compresses A

$A(tol)$ rank $k \Rightarrow$ need $m \cdot k + k + n \cdot k$ words to store SVD, can be $\ll m \cdot n$

Solving (nearly) low rank LS using Tikhonov regularization, or ridge regression

Replace $\underset{x}{\operatorname{argmin}} \|Ax-b\|_2^2$

by $\underset{x}{\operatorname{argmin}} \|Ax-b\|_2^2 + \lambda\|x\|_2^2$

$\lambda > 0$

λ "penalizes" very large x

λ user parameter

$$\underset{x}{\operatorname{argmin}} \|Ax-b\|_2^2 + \lambda\|x\|_2^2$$

$$= \underset{x}{\operatorname{argmin}} \left\| \begin{bmatrix} A \\ \lambda^{\frac{1}{2}} I \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|_2^2$$

full rank for any $\lambda > 0$

$$NE \Rightarrow x = \left(\begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix}^T \begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix} \right)^{-1} \begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix}^T \begin{bmatrix} b \\ 0 \end{bmatrix}$$

$$(*) \quad = (A^TA + \lambda I)^{-1} A^T b$$

just add λ to diagonal of A^Tb

How does λ change SVD?

plug $A = U\Sigma V^T$ into $(*)$

$$x = V\left(\Sigma(\Sigma^2 + \lambda I)^{-1}\right)U^T b$$

$$= V \operatorname{diag}\left(\frac{\sigma_i}{\sigma_i^2 + \lambda}\right) U^T b$$

usual SVD if $\lambda = 0$

$$\sigma_i \gg \lambda^{1/2} \Rightarrow \frac{\sigma_i}{\sigma_i^2 + \lambda} \sim \frac{1}{\sigma_i}$$

$$\sigma_i < \lambda^{1/2} \Rightarrow \frac{\sigma_i}{\sigma_i^2 + \lambda} \leq \frac{1}{\lambda^{1/2}}$$

i.e. λ and tol in A(tol) play analogous roles

Solving low rank LS using QR

QR with column pivoting

Suppose we did $A = QR$ exactly, rank$(A) = r < n$, what would R look like?

If leading r columns of A were linearly independent (true for "almost all" low rank A)

$$R = \begin{array}{c|cc} & r & n-r \\ \hline r & R_{11} & R_{12} \\ n-r & 0 & 0 \end{array}$$

R_{11} full rank so $R_{22} = 0$

If A nearly low rank, hope that $\|R_{22}\| < tol$, set $R_{22} = 0$

Assuming this works, solve LS:

$$A = {}_m Q_n R = {}_m\left[\overset{n}{Q}, \overset{m-n}{Q'}\right] \begin{bmatrix} R \\ 0 \end{bmatrix}$$

$${}_m\left[\overset{n}{Q}, \overset{m-n}{Q'}\right] = {}_m\left[\overset{r}{Q_1}, \overset{n-r}{Q_2}, \overset{m-n}{Q'}\right]$$

$$\operatorname{argmin}_x \|Ax - b\|_2$$

$$= \operatorname{argmin}_x \left\| [Q_1, Q_2, Q'] \begin{bmatrix} R \\ 0 \end{bmatrix} x - b \right\|_2$$

$$= \operatorname{argmin}_x \left\| \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{array}{c} r \\ n-r \end{array}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} Q_1^T b \\ Q_2^T b \\ Q'^T b \end{bmatrix} \right\|_2$$

$$= \operatorname{argmin}_x \left\| \begin{bmatrix} R_{11} x_1 + R_{12} x_2 - Q_1^T b \\ - Q_2^T b \\ - Q_1' b \end{bmatrix} \right\|_2$$

solution $x_1 = R_{11}^{-1} Q_1^T b - R_{11}^{-1} R_{12} x_2$ for any x_2

How to pick x_2 to minimize $\|x\|_2$?