Welcome back to Ma221! Lecture 21, Oct 11

Solving Low rank LS using QR with column pivoting

Ex: $A = \begin{bmatrix} e & 1 \\ 0 & 0 \end{bmatrix}$ e tiny $\quad \operatorname{argmin}_x \|Ax-b\|_2$

$\Rightarrow x = \begin{bmatrix} (b_1 - x_2)e \\ x_2 \end{bmatrix}$ e tiny $\Rightarrow$ very sensitive to b_1 or x_2

$AP = \begin{bmatrix} 1 & e \\ 0 & 0 \end{bmatrix} \Rightarrow x = \begin{bmatrix} b_1 - e \cdot x_2 \\ x \end{bmatrix}$

insensitive to changes in b, x_2

What does a good R look like in general?

Compare $\begin{matrix} k \\ n-k \end{matrix}\begin{bmatrix} \overset{k}{R_{11}} & \overset{n-k}{R_{12}} \\ 0 & R_{22} \end{bmatrix}$ with $\begin{matrix} k \\ n-k \end{matrix}\begin{bmatrix} \overset{k}{\Sigma_1} & \overset{n-k}{0} \\ 0 & \Sigma_2 \end{bmatrix}$

Def Rank Revealing QR (RRQR) is $A \cdot P = QR$ P = permutation s.t.

① R_{22} is "small", ideally $\|R_{22}\|_2 = \sigma_{k+1}(A)$

R_{22} "contains" smallest $n-k$ singular values

② R_{11} is "large", ideally $\sigma_{min}(R_{11}) = \sigma_k(A)$, not too much smaller

If in addition

③ $\|R_{11}^{-1} R_{12}\|_2$ not "too large"

then $AP = QR$ called "strong RRQR"

Thm: if ①, ②, ③ hold then

$$\sigma_i(A) \geq \sigma_i(R_{11}) \geq \frac{\sigma_i(A)}{\sqrt{1 + \|R_{11}^{-1} R_{12}\|_2^2}} \quad \text{for } i = 1..k$$

$$\sigma_i(A) \leq \sigma_{max}(R_{22}) \cdot \sqrt{1 + \|R_{11}^{-1} \cdot R_{12}\|_2^2} \quad i = k+1..n$$

Leading k columns of AP contain most information about range(A)

$$AP = [\overset{k}{A_1}, \overset{n-k}{A_2}] = QR = \overset{k}{\underset{n-k}{\begin{bmatrix} Q_1, Q_2 \end{bmatrix}}} \overset{k}{\underset{n-k}{\begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix}}}$$

$$\approx [Q_1, Q_2] \cdot \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix} \quad \text{where } \sigma_{k+1}(A) \text{ small}$$

$$= [Q_1] [R_{11}, R_{12}] = Q_1 R_1 [I, R_{11}^{-1} R_{12}]$$

$$= A_1 [\overset{k}{I}, \underbrace{R_{11}^{-1} R_{12}}_{\text{not large}}]$$

How to compute P

Algorithm: QR with column pivoting

QRCP for short

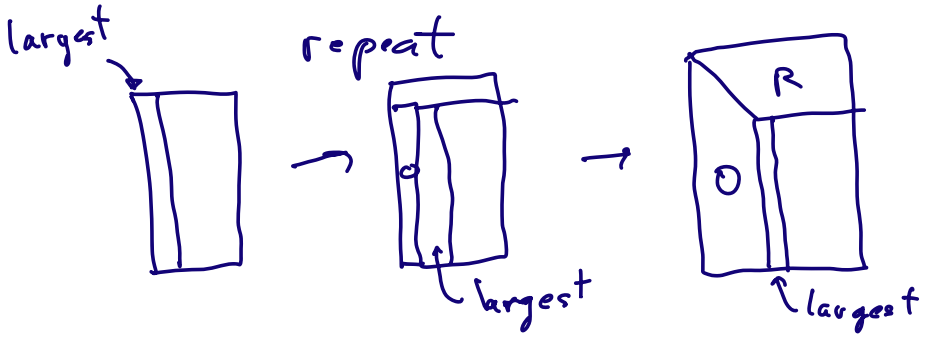
analogous to partial pivoting:

simple greedy algorithm

often works; like GEPP, can fail in rare cases by factor 2^n

First step: pick largest (in norm) column of A
move it to front
take one step of QR
among remaining columns, pick one
with largest orthogonal component
to first column

repeat



for $i = 1$ to $\min(m-1, n)$ or
when R_{22} small enough
("truncated QR")

choose largest column in norm
in trailing matrix
$\operatorname{argmin}_{j \geq i} \| A(i:m, j) \|_2$

if $i \neq j$, swap columns i & j
multiply $A(i:m, i:n)$ by Householder
matrix to zero out $A(i+1:m, i)$

Suppose we stop after k steps
because $\|R_{22}\| <$ user threshold
$\Rightarrow$ cost $= 4 \cdot m \cdot n \cdot k$ vs. $O(mn^2)$ for
full QR

Naive way to compute column norms
costs $O(mn^2)$, too much
cheaper: $O(mn)$ by subtracting out
one row at each step
oops: get roundoff if norm small
at later step $\Rightarrow$ if norm
small enough, recompute, continue

Drawbacks: can fail in rare cases
maximizes communication

Lapack: geqpf (truncated QR
in next release)

Matlab: [Q, R, P] = qr(A)

Matlab demo: (code in typed notes)

How to fix failure, communication (i.e. all
BLAS 2 so far)

Strong RRQR alg: Gu, Eisenstat
- avoids failures by pivoting correctly
- more complicated pivoting rule
exchange a column in R_{11} and one
not in R_{11} to maximize $\det(R_{11})$
- guarantees strong RRQR i.e. $\|R_{11}^{-1} R_{12}\|$
bounded by poly(n)
- more expensive, still $O(m \cdot n \cdot k)$ if
stop after k steps

Avoiding communication:

All BLAS2 so far ⇒
touch whole trailing submatrix
at each steps via Householder
⇒ data movement = $O(mn^2)$
vs. $O(\frac{mn^2}{\sqrt{\text{cache size}}})$

Use BLAS3 as in QR ⇒ can only do
half flops in BLAS3

Hit lower bound:

Tournament Pivoting: pick b columns
at a time, not 1

Randomization (Duersch, Gu, 2017)

Low Rank Factorization without
Orthogonal Factors

Why avoid orthogonality:

① maintain sparsity

② explainability in data science

Ex: A: rows represent people
cols " characteristics
age, weight, etc

Suppose a column of Q is

$= .2 \cdot age - .3 \cdot height + .1 \cdot income$

hard to interpret as a predictor of another column, say "been treated for some disease"

Instead: approximate each column by linear combinations of as few other columns as possible:

Def: CUR decomposition of A

C = subset of k columns of A

R = " " k rows of A

U = $k \times k$ matrix

where $\|A - C \cdot U \cdot R\|$ is small, close to σ_{k+1}

lots of algorithms (see web page)

(1) Choose C: perform QR with column pivoting: "k most linearly independent columns"

$J = \{j_1, j_2, \ldots, j_k\}$ = indices of cols

$C = A(:, J)$

(2) Perform: GEPP on C to pick "k most linearly independent rows" of C

$I = \{i_1, i_2, \ldots, i_k\}$ $R = A(I, :)$

(3) pick U: (HW Q 3.12)

choose U to minimize $\|A - CUR\|_F$

answer: $U = C^+ A R^+$

(4) cheaper: choose U so $CUR = A$ in rows I and columns J

$$U = (A(I, J))^{-1}$$