Welcome to MA221! (Mar 1)

Recall TSQR (sequential)

Same idea for parallel TSQR

"Map Reduce" where QR is reduction operation

Same idea for partial pivoting on a Tall Skinny matrix (TSLU) one communication tree for many columns
Basic operation

\[ A \rightarrow PA \]

select subset of rows of \( A \) chosen by partial pivoting

"most linearly independent rows of \( A \)"

Given rotations: another simple orthogonal transformation used for eigenproblems, SVD

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

\[ R(\theta)x \]
\[ R(i,j, \theta) \text{ applies rotation to } x_i \text{ and } x_j \]

\[
\begin{bmatrix}
\cos \theta & -\sin \theta \\
\sin \theta & \cos \theta
\end{bmatrix}
\]

How to pick \(\theta\) to zero out \(x_d\):
\[
\begin{bmatrix}
\cos \theta & -\sin \theta \\
\sin \theta & \cos \theta
\end{bmatrix}
\begin{bmatrix}
x_i \\
x_j
\end{bmatrix}
= \begin{bmatrix}
x_i^2 + x_j^2 \\
0
\end{bmatrix}
\]

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

can do QR using Givens rotations, no advantage over Householder for dense \(A\), maybe less fill-in 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
\[ \text{fl}(a \circ b) = (a \circ b)(1 + \delta) \quad |\delta| \leq \varepsilon \]
to show that multiplying by one Householder or Givens gets smaller error:

\[ \text{fl}(Q' \cdot A) = Q' \cdot A + E \quad \|E\| = O(\varepsilon) \|A\| \]

\( Q' \) "nearly orthogonal" \( \Rightarrow \)
\[ Q' = Q + F, \quad \|F\| = O(\varepsilon) \]

\( Q \) exactly orthogonal

\[ \text{fl}(Q' \cdot A) = Q' \cdot A + E = (Q + F)A + E \]
\[ = QA + FA + E = QA + G \]

= exact orthogonal transformation + \( G \)

\[ \|G\| = \|FA + E\| \leq \|F\| \|A\| + \|E\| \]
\[ \leq O(\varepsilon) \|A\| + O(\varepsilon) \|A\| = O(\varepsilon) \|A\| \]

\[ \text{fl}(Q' \cdot A) = QA + G = Q(A + QTG) \]
\[ = Q(A + G') \]

\[ \|G'\| = \|G\| \]

\( \Rightarrow \) multiplication by \( Q' \) is backward stable

= exact transform of \( A + G' \)

What if multiply by many \( Q_i' \)?
\[ f(Q'_3(Q'_2(Q'_1A))) \]
\[ = f(Q'_2(Q'_2(Q_1A + E_1))) \]
\[ = f(Q'_2(Q_2(Q_1A + E_1) + E_2)) \]
\[ = (Q_3(Q_2(Q_1A + E_1) + E_2) + E_3) \]
\[ = \frac{Q_3Q_2Q'_1A + Q_3Q_2E_1 + Q_3E_2 + E_3}{Q} \]
\[ \|E_11\| \leq \|E_1\| + \|E_2\| + \|E_3\| = O(\epsilon)\|A\| \]
\[ = Q(A + Q^TE) \quad \|Q^TE\| = \|E\| = O(\epsilon)\|A\| \]

Same idea (pick \( A = I \)) to show that product \( f(Q'_3 Q'_2 Q'_1) \) nearly orthogonal.

One more fast (but unstable)

QR algorithm: used in practice when need \( Q \), know \( A \) well conditioned.

Cholesky QR:

Factor \( A^TA = R^TR \) using Cholesky form \( Q = A \cdot R^{-1} \) since \( A = QR \)

Can fail completely eg if

Some pivot during Cholesky is negative \( \Rightarrow \) 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: $A$ $m \times n$, $m \geq n$, rank $r < n$

$A = U \Sigma V^T = m \begin{bmatrix} U_1 & U_2 & U_3 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1 & V_2 \end{bmatrix}^T$

$\Sigma_1$ is full rank
$\Sigma_2 = 0$ (later: tiny)
The set of vectors minimizing \( \|Ax-b\|_2 \) is
\[
\{ x = V_1 \Xi_1^{-1} U_1^T b + V_2 y_2 \quad \text{any } y_2 \in \mathbb{R}^m \}
\]
Unique \( x \) minimizing both \( \|Ax-b\|_2 \) and \( \|x\|_2 \) is gotten from \( y_2 = 0 \)
\[
x = V_1 \Xi_1^{-1} U_1^T b
\]

Def: \( A^+ = V_1 \Xi_1^{-1} U_1^T \) is Moore-Penrose pseudoinverse of \( A \) (includes full rank case, \( r = n \))
(in practice: \( \Xi_2 \) 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: \( \|Ax-b\|_2 = \| U \Xi V_x^T - b \|_2 \)
\[
= \| \Xi V_x^T - U^T b \|_2 \quad \text{since } U \text{ orthogonal}
\]
\[
= \| y - U^T b \|_2 \quad \text{where } y = V_x^T x
\]
\( \|x\|_2 = \|y\|_2 \) so okay to minimize either one
\[
= \| \begin{bmatrix} \Xi_1 y_1 & -U_1^T b \\ U_1^T b & y_2 \end{bmatrix} \|_2 \quad \text{where } y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_{n-r}
\]
minimized by \( y_1 = \Sigma_1^{-1} U_1^T b \)
and \( \|y_1\|_2^2 = \|y_1\|_2^2 + \|y_2\|_2^2 \)
minimized by \( y_2 = 0 \)

\[ x = V y_- = [V_1, V_2] \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = V_1 y_1 + V_2 y_2 = V_1 \Sigma_1^{-1} U_1^T b \]

---

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

\[ K(A) = \frac{\sigma_{\text{max}}}{\sigma_{\text{min}}} = \infty \text{ if } A \text{ rank deficient} \]

\[ \arg\min_x \| \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \end{bmatrix} \|_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \]

\[ \arg\min_x \| \begin{bmatrix} 1 & 0 \\ 0 & c \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} 1 \\ c \end{bmatrix} \|_2 = \begin{bmatrix} 1 \\ c \end{bmatrix} \] (e tiny)

what does a “solution” mean if it can change discontinuously?

Often A not known exactly, just up to some tolerance, \( \| A - A' \|_2 \leq \epsilon_0 \)

What to do?
Def: truncated SVD

\[ A(\text{tol}) = U \cdot \Sigma(\text{tol}) \cdot V^T \]

\[ \Sigma(\text{tol}) = \text{diag}(\sigma_1(\text{tol}), \sigma_2(\text{tol}), \ldots, \sigma_n(\text{tol})) \]

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

\[ A(\text{tol}) = \text{lowest rank matrix within distance tol of } A \]

Using \( A(\text{tol}) \) for LS reduces

\[ k = \frac{\sigma_{\text{max}}}{\sigma_{\text{min}}} \text{ to } \frac{\sigma_{\text{max}}}{\text{tol}} \]

\( \text{tol} \) is a “knob” for user to trade off sensitivity \( k \) 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: \( x_i = \arg \min_x \| A(tol)x - b_i \|_2 \)
\( x_2 = \arg \min_x \| A(tol)x - b_2 \|_2 \)

Choose \( \| x_i \|_2 \) of smallest norm

Then \( \| x_i - x_2 \|_2 \leq \frac{\| b_i - b_2 \|_2}{tol} \)

Proof: \( \| x_i - x_2 \|_2 = \| (A(tol))^{-1} (b_i - b_2) \|_2 \)

\[= \| U \Sigma (U^T) (b_i - b_2) \|_2 \]
\[= \| \text{diag}(\sigma_1^{-1}, \sigma_2^{-1}, \ldots, \sigma_k^{-1}, 0, \ldots) \cdot U^T (b_i - b_2) \|_2 \]

\[\leq \frac{1}{\sigma_k} \| U^T (b_i - b_2) \|_2 \]
\[\leq \frac{1}{tol} \| b_i - b_2 \|_2 \]

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

Precisely, constant changes when \( tol = \sigma_i \)

Other advantages of \( A(tol) \):

Setting more \( \sigma \) 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 \( \arg \min_x \| Ax - b \|_2^2 \)
by \( \arg \min_x \| Ax - b \|_2^2 + \lambda \| x \|_2^2 \)

\( \lambda \) "penalizes" very large \( x \)
\( \lambda \) user parameter

\[ \arg \min_x \| Ax - b \|_2^2 + \lambda \| x \|_2^2 \]
\[ = \arg \min_x \| [A \overset{\lambda}{\| x \|_2} - [0] \|_2 \]

full rank for any \( \lambda > 0 \)

\( \text{NE} \Rightarrow x = (A^T A + \lambda I)^{-1} A^T b \)

just add \( \lambda \) to diagonal of \( A^T b \)

How does \( \lambda \) change SVD?

plug \( A = U \Sigma V^T \) into (\#)
\[ x = V \left( \Sigma \left( \frac{\lambda^2 + \lambda I}{\sigma_i^2 + \lambda} \right)^{-1} \right) U^\top b \]

\[ = V \text{ diag} \left( \frac{\sigma_i}{\sigma_i^2 + \lambda} \right) U^\top b \]

usual SVD if \( d = 0 \)

\[ \sigma_i \gg \lambda^{1/2} \Rightarrow \frac{\sigma_i}{\sigma_i^2 + \lambda} \approx \frac{1}{\sigma_i} \]

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

i.e. \( d \) and tol in \( A(\text{tol}) \)
play analogous roles

---

Solving low rank LS using QR

QR with column pivoting

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

if leading \( r \) columns of \( A \) were linearly independent (true for
"almost all" low rank \( A \)?)
\[
R = r \begin{bmatrix}
R_{11} & R_{12} \\
0 & 0
\end{bmatrix} \quad \text{R}_{11} \text{ full rank}
\]

so \( R_{22} = 0 \)

If \( A \) nearly low rank, hope that \( \| R_{22} \| \ll 1 \), set \( R_{22} = 0 \)

Assuming this works, solve LS:

\[
A = mQ \times \hat{R} = \begin{bmatrix}
\hat{R}_{11} & \hat{R}_{12} \\
0 & 0
\end{bmatrix}
\begin{bmatrix}
\hat{R} \\
0
\end{bmatrix}
\]

\[m \begin{bmatrix}
\hat{R}_{11} & \hat{R}_{12} \\
0 & 0
\end{bmatrix} = m \begin{bmatrix}
R_{11} & R_{12} \\
0 & 0
\end{bmatrix}
\]

\[
\arg \min_x \|Ax - b\|_2
\]

\[
= \arg \min_x \| [Q_1, Q_2, Q'] [R] x - b \|_2
\]

\[
= \arg \min_x \| \begin{bmatrix}
R_{11} & R_{12} \\
0 & 0
\end{bmatrix} \begin{bmatrix}
X_1 \\
X_2
\end{bmatrix} - \begin{bmatrix}
Q_{1}^T b \\
Q_{2}^T b \\
Q_{3}^T b
\end{bmatrix} \|_2
\]

\[
= \arg \min_x \| \begin{bmatrix}
R_{11} \chi_1 + R_{12} X_2 - Q_{1}^T b \\
0 \\
0
\end{bmatrix} \|_2
\]

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 \( \|x\|_2 \)?