# Welcome to Ma221! Lecture 14, Fall 24

Finish QRCP

(see Matlab demos in class notes)

How to fix failure of QRCP

Gu / Eisenstat - Strong RRQR alg

- avoids failures to pivot correctly
- more complicated pivoting

  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
- more expensive, still $O(m \cdot n \cdot k)$

  like QRCP, but bigger constant

Avoiding Communication in RRQR

so far algorithm touches all trailing columns at every step $\Rightarrow O(mn^2)$ words moved instead of $O(mn^2/\sqrt{\text{cache size}})$

- try to use BLAS3, only half flops in BLAS3
- tournament pivoting as in TSLU for GEPP
  - pick b columns at a time instead of 1

- Randomization (Duersch + Gu, 2017,
Martinsson, 2015 ...)

---

Low rank factorization without orthogonal factors — why avoid orthogonal factors?

- explainability in data

Eg: $A$: rows : represent people

cols : represent characteristics: age, weight, income, ...

Suppose a column of $Q$ is

$.2 \cdot \text{age} - .3 \cdot \text{height} + .1 \cdot \text{income} \ldots$

what does that mean?

Instead want to approximate other columns of $A$ with as few selected columns as possible

Def: CUR decomposition of $A$

$C$ = subset of $k$ columns of $A$

$R$ = " " $k$ rows of $A$

$U$ = $k \times k$ "connector"

where $\|A - CUR\|_2$ is "small"

close to lower bound from SVD: $\sigma_{k+1}$

(1) choose $C$: perform QR with some kind of column pivoting to pick "k most linearly independent cols"

$J = \{j_1, j_2, \ldots, j_k\}$ be indices of selected columns, so $C = A(:, J)$

(2) perform GEPP, or TSLU, on $C$ to pick "k most linearly independent rows", $I = \{i_1, i_2, \ldots, i_k\}$ be indices

$R = A(I, :)$

Still need $U$.

(3) Use HW 3.12: choose $U$ to minimize $\|A - CUR\|_F$ : $U = C^+ A R^+$

(4) Cheaper: choose $U$ so that $CUR$ matches $A$ in rows $I$ and cols $J$

$C(I, 1:k) = R(1:k, J) = A(I, J)$

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

$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ oops, if we pick $C = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $R = [0, 1]$

(2) above won't do this

---

Randomized Linear Algebra

LS and SVD on $A^{m \times n}$ $\quad m \gg n$

Let $Q$ be a random matrix $m \times k$

$Q$ orthogonal $\quad k \ll n$

Approximate $A$ by $Q\underbrace{(Q^T A)}_{k\times n} = \underbrace{(QQ^T)A}_{m\times n}$

rank $\le k$

Cost? $Q^T A$ costs $2m\cdot n\cdot k$,

only 2x faster than QRCP for k steps

$\Rightarrow$ can also use cheap structured $Q$ to make multiplication cheaper

First big speedup over QR for LS in 2010, Blendenpik (Toledo, Avron, Maymounkov)

Best theoretical result so far, for $\text{argmin}_x \|Ax-b\|_2$ is $O(nnz(A))$ for $A$ sparse

Clarkson, Woodruff 2012

Examples in low dimensions of why random projections work:

Ex: $x\in\mathbb{R}^2$, $q\in\mathbb{R}^2$ random unit vector

$q = \begin{bmatrix}\sin t\\ \cos t\end{bmatrix}$ $t$ uniform in $[0,2\pi)$

How well does $|x^T q|^2$ approximate $\|x\|_2^2$?

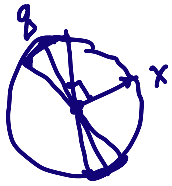
What is distribution of

$$|x^T q|^2 = \|x\|_2^2 \cdot \cos^2 \angle(x,q) \ ?$$

Easy to see $\angle(x,q)$ also uniform in $[0,2\pi)$

$$E(|x^T q|^2) = .5\,\|x\|_2^2$$

What is Prob($|x^T q|^2$) underestimates $\|x\|_2^2$ by a factor $e \ll 1$?




$\text{Prob}(|\cos(t)|^2 < e) \sim 2\sqrt{e}/\pi$

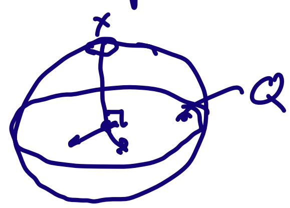
Ex: $x \in \mathbb{R}^3$, Q random plane

i.e. random 3x2 orthogonal matrix

$x^T Q$ = projection of x onto plane

How well does $\|x^T Q\|_2^2$ approximate $\|x\|_2^2$?

$\|x^T Q\|_2^2 < e\|x\|_2^2$ requires x nearly parallel to perpendicular of Q




what is chance that x lies in little circle of radius $\sqrt{e}$?

prob = $O(e)$ versus $\sqrt{e}$ before

Johnson-Lindenstrauss (J-L) Lemma

$0 < \varepsilon < 1$, $x_1, \ldots, x_n$ any n vectors in $\mathbb{R}^m$

$k \geq 8 \cdot \ln(n)/\varepsilon^2$

Let F be random $k \times m$ orthogonal matrix, multiplied by $\sqrt{m/k}$

Then with probability $\geq \frac{1}{n}$ for all $1 \leq i, j \leq n$ $i \neq j$

$$1 - \varepsilon \leq \frac{\|F(x_i - x_j)\|_2^2}{\|x_i - x_j\|_2^2} \leq 1 + \varepsilon$$

Probability $\frac{1}{n}$ seems small, but being positive means $F$ exists (original goal of JL)

proof sketch: Think of $F$ fixed $x = x_i - x_j$ as random

$F = \begin{bmatrix} I \\ 0 \end{bmatrix}$, each entry of $x$ i.i.d $N(0,1)$

$$\frac{\|Fx\|_2^2}{\|x\|_2^2} = \frac{\text{sum of squares of } N(0,1) \text{ r.v.}}{\text{sum of more squares of } N(0,1) \text{ r.v.}}$$

(Dasgupta + Gupta on web page)

---

Approximate $\|x\|_2$ by $\|Fx\|_2$ $x \in \mathbb{R}^m$

$F^{k \times m}$ $k \ll m$

What choices of $F$ are there?

(see webpage for link to RandBLAS)

To construct random orthogonal $Q$, as in J-L

1. $A^{m \times k}$ each entry $N(0,1)$ i.i.d

   $A = QR$, $Q$ random orthogonal

   $F = \sqrt{\frac{m}{k}}\, Q$

   too expensive: cost $= O(mk^2)$ for QR

2. represent $Q = \prod (I - 2 v_i v_i^T)$

   pick each $v_i$ randomly

   cost reduces to $O(mk)$

③ Some applications let us use
$F(i,j) = N(0,1)$ i.i.d., save doing
QR for later in algorithm,
when cheaper

Alternatives to random orthogonal

① Subsampled randomized trig transform
(SRTT) trig ∋ FFT

$F \cdot x$ will cost $O(m \log m)$ or even $O(m \log k)$

$F = R \cdot FFT \cdot D$ so $Fx = R(FFT(Dx))$

$D$ = $m \times m$ diagonal matrix where
$D(i,i)$ uniformly distributed
on unit circle

$R^{k \times m}$ = a random subset of
$k$ rows of $m \times m$ $I$ (sampling)

Real case: SRHT =

Subsampled randomized Hadamard Transform

FFT replaced by $H$ = Hadamard

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad H_4 = \begin{bmatrix} H_2 & H_2 \\ H_2 & -H_2 \end{bmatrix}$$

$$H_{2^n} = \begin{bmatrix} H_{2^{n-1}} & H_{2^{n-1}} \\ H_{2^{n-1}} & -H_{2^{n-1}} \end{bmatrix}$$

cost$(Hx)$ =
$O(m \log m)$
like FFT

Intuition for why $\|Fx\|_2 \approx \|x\|_2$ :

FFT, or H, "mixes" rows of x
so that sampling k of them good enough

So far: x dense, want speed up if sparse

Goal: cost $(F \cdot x) = O(nnz(x))$

$F = S \cdot D$ $\quad\quad$ $F \cdot x = S(Dx)$

$D =$ m×m diagonal $\quad D_{ii} = \pm 1$

$S = k \times m$, each column is a
random column of $I_{k \times k}$

$y = S \cdot D$:
for each nonzero in x: $x_i$
pick random $y_j$
$y_j = y_j \pm x_i$

called: Randomized Sparse Embedding

F is not as "statistically strong"
as previous F's, so need larger k

---

Apply these choices of F to LS

"sketch and solve": project onto
smaller problem, solve it

"sketch and iterate": use randomization
to build a "preconditioner", iterate
(Chap 6)

$x_{true} = \arg\min_x \|Ax-b\|_2 \quad A^{m\times n} \quad m>n$

$x_{approx} = \arg\min_x \|F(Ax-b)\|_2$

Use J-L for $F^{k\times m}$ (random orthogonal)

choose $k = n \log n / \varepsilon^2$ rows

in order to get

$\|A x_{approx} - b\|_2 \le (1+\varepsilon) \|Ax_{true} - b\|_2$

No bound on $\|x_{approx} - x_{true}\|_2$

Cost: if $F$ dense, computing $F\cdot A$

using dense GEMM cost $O(k\cdot m\cdot n)$

$= O(mn^2 \log n/\varepsilon^2)$, worse than QR: $O(mn^2)$

Use cheaper $F$:

SRTT with dense $A$

$FA$ costs $O(n\cdot m\cdot \log m)$

$FA$ has size $k\times n$ so solving smaller LS

problem $\arg\min_x \|FAx - Fb\|_2$

costs $O(kn^2) = O(n^3 \log n/\varepsilon^2)$

total cost $= O(m\cdot n \log m + n^3 \log n/\varepsilon^2)$

potentially much cheaper than

QR $O(m\cdot n^2)$ when $m \gg n$

and $\varepsilon$ not too small,

otherwise use "sketch and iterate"

Sparse LS: goal: cost $= O(nnz(A))$

+ "lower order terms"

See papers by Clarkson + Woodruff
Meng + Mahoney

$F$ = Randomized Sparse Embedding
$k = O\left(\left(\frac{n}{\varepsilon}\right)^2 \cdot \log^6\left(\frac{n}{\varepsilon}\right)\right)$

Forming $FA$ and $Fb$ costs
$nnz(A)$ and $nnz(b)$

but since $k = \Omega(n^2)$ much larger
than SRFT for which $k = O(n)$

If we solved $\arg\min_x \|(FA)x - Fb\|_2$
using dense QR, would cost
$O(kn^2) = O(n^4 \log^2(\frac{n}{\varepsilon})/\varepsilon^2)$
much more than SRFT

Trick: randomize again to solve
smaller LS problem, using SRFT

Thm: With probability $\geq \frac{2}{3}$
$\|Ax_{approx} - b\|_2 \leq (1+\varepsilon) \cdot \|Ax_{true} - b\|_2$

To make probability of success close to 1
run $s$ times, pick result with
smallest residual, prob(success) $= 1 - \frac{1}{3^s}$