Sketching
Lecture 8

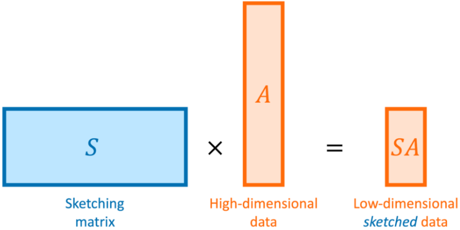
F. M. Faulstich

02/02/2024

What is sketching?

It is a dimension reduction:

- Let $\mathbf{A} \in \mathbb{F}^{n \times m}$.
  A matrix $\mathbf{S} \in \mathbb{F}^{d \times n}$ with $d \ll n$ is called a sketching matrix. We sketch $\mathbf{A}$ by applying $\mathbf{SA}$



What is sketching?

It is a dimension reduction:

- Let $\mathbf{A} \in \mathbb{F}^{n \times m}$.
  A matrix $\mathbf{S} \in \mathbb{F}^{d \times n}$ with $d \ll n$ is called a sketching matrix. We sketch $\mathbf{A}$ by applying $\mathbf{SA}$
- Consider $\mathbf{A} = [\mathbf{a}_n | ... | \mathbf{a}_m]$. The matrix $\mathbf{S}$ is a good sketch if

$$(1 - \varepsilon)\|\mathbf{a}_i\| \leq \|\mathbf{S}\mathbf{a}_i\| \leq (1 + \varepsilon)\|\mathbf{a}_i\|$$

  the lengths of the vectors are preserved.
  (Distortion condition)
- In linear algebra, we want to sketch

$$\mathrm{Im}(\mathbf{A}) = \{\mathbf{A}\mathbf{x} \mid \mathbf{x} \in \mathbb{F}^m\}$$

- There exists sketching matrices that achieve ε distortion for $\mathrm{Im}(\mathbf{A})$ with an output dimension

$$d \approx m/\varepsilon^2$$

Sketching matrices

Sketching is **not** unique!

Sketching matrices

- Random projections
- Johnson-Lindenstrauss lemma:
  Given $0 < \varepsilon < 1$, a set X of $m \in \mathbb{Z}_{\geq 1}$ points in $\mathbb{R}^N$ ($N \in \mathbb{Z}_{\geq 0}$), and an integer $n > 8(\ln m)/\varepsilon^2$, there exists a linear map $f : \mathbb{R}^N \to \mathbb{R}^n$ such that

  $$(1-\varepsilon)\|u-v\|^2 \leq \|f(u)-f(v)\|^2 \leq (1+\varepsilon)\|u-v\|^2$$

  for all $u, v \in X$.

> "a small set of points in high-dimensional space can be embedded into a lower-dimensional space in such a way that the distances between the points are nearly preserved."

Gaussian Embeddings

- $\mathbb{F}^{d \times n} \ni \mathbf{S} \sim \mathcal{N}(\mathbf{0}, \frac{1}{d}\mathbf{I})$, i.e., the entries of $\mathbf{S}$ are i.i.d. $\mathcal{N}(0, \frac{1}{d})$
- Sketches $\mathrm{Im}(\mathbf{A})$ well
- Benefits:
  Easy to code
  Requires only the standard matrix product
  choose $d \approx m/\varepsilon^2$
- Downsides:
  Sketching a vector $\mathbf{a} \in \mathbb{F}^n$ costs $\mathcal{O}(dn)$
  Additional storage required for $\mathbf{S}$

Subsampled Randomized Trigonometric Transforms (SRTT)

- Ansatz

$$\mathbf{S} = \sqrt{\frac{n}{d}}\mathbf{RFD}$$

  where:
  - $\mathbf{D} \in \mathbb{F}^{n\times n}$ diagonal with Rademacher i.i.d. entries $[\mathcal{O}(n)]$
  - $\mathbf{F} \in \mathbb{F}^{n\times n}$ fast trigonometric transform
    e.g. discrete cosine transform
  - $\mathbf{R} \in \mathbb{F}^{d\times n}$ is a selection matrix. $[\mathcal{O}(d)]$
    Let $\{i_1, ..., i_d\} \subset [\![n]\!]$, then $\mathbf{Rb} := (b_{i_1}, ..., b_{i_d})$
- Benefits:
  Sketching a vector $\mathbf{a} \in \mathbb{F}^n$ costs $\mathcal{O}(n\log(n))$
- Drawbacks:
  SRTT requires a good implementation of a fast trigonometric transform.
  choose $d \approx (m\log(m))/\varepsilon^2$

Discrete cosine transform (DCT)

- Similar to discrete Fourier transform but real valued coefficients
- DCT–II: Let $\mathbf{x} \in \mathbb{R}^n$

$$\mathbf{y}_k = \sum_{i=0}^{n-1} \mathbf{x}_i \cos\left(\frac{\pi}{n}\left(i + \frac{1}{2}\right)k\right) \qquad \text{for } k = 0, ..., n-1$$

- Can be implemented fast! $\mathcal{O}(n \log(n))$

SRTT (MATLAB)

```matlab
function [c] = SRTT_sketch(b,d)

    n = length(b);
    signs = 2*randi(2,n,1)-3; % diagonal entries of D (Rademacher)
    idx = randsample(n,d); % indices i_1,...,i_d defining R

    % Multiply S against b
    c = signs .* b; % multiply by D
    c = dct(c); % multiply by F
    c = c(idx); % multiply by R
    c = sqrt(n/d) * c; % scale

end
```

Sparse Sign Embeddings (SSE)

- Ansatz

$$\mathbf{S} = \frac{1}{\sqrt{\zeta}}[\mathbf{s}_1|...|\mathbf{s}_n]$$

  $\mathbf{s}_i \in \mathbb{F}^d$ are random vectors with $\mathbb{N} \ni \zeta$ many Rademacher entries.
  In practice, ζ is small like 8.

- Benefits:
  Using a sparse library **S** can applied super fast! $\mathcal{O}(n)$ or $\mathcal{O}(n\log(\mathrm{d}))$

  With a good sparse matrix library, sparse sign embeddings are often the fastest sketching matrix by a wide margin

- Drawbacks:
  Larger storage than SRTT: $\mathcal{O}(\zeta n)$ vs $\mathcal{O}(n)$

Comparison (time)

We compare:

- Construction: The time required to generate the sketching matrix $\mathbf{S}$.
- Vector apply. The time to apply the sketch to a single vector
- Matrix apply. The time to apply the sketch to an $n \times 200$ matrix

Settings and parameters:

- We will test with input dimension $n = 10^6$ and $d = 400$.
- We use SRTT with DCT
- We use $\zeta = 8$ for SSE

Comparison (time)

Averaged times over 20 runs:

Time (sec)	Gaussian	SRTT	Sparse sign
Construction	2.7	0.0052	0.038
Vector apply	0.32	0.011	0.0031
Matrix apply	5.9	1.63	0.079

Conclusion:

- SSE are the fastest sketching matrices by a wide margin!
- For an "end-to-end" workflow involving generating the sketching matrix $\mathbf{S} \in \mathbb{R}^{400 \times 10^6}$ and applying it to a matrix $\mathbf{A} \in \mathbb{R}^{10^6 \times 200}$, SSE are 14x faster than SRTTs and 73x faster than Gaussian embeddings.

How to use sketching?

Sketch-and-solve:

- Apply sketch to perform a dimension reduction
- Apply conventional numerical linear algebra tools

Example: Least-squares problem

$$\min_{\mathbf{x}\in\mathbb{R}^m} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|$$

- What do we sketch?
- We sketch: $\mathbf{A}$ and $\mathbf{b}$
- Then solve

$$\min_{\hat{\mathbf{x}}\in\mathbb{R}^m} \|(\mathbf{S}\mathbf{A})\hat{\mathbf{x}} - \hat{\mathbf{b}}\|$$

Does this work?

- Let $\mathbf{x}_*$ be the solution to

$$\min_{\mathbf{x}\in\mathbb{R}^m} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|$$

  and let $\hat{\mathbf{x}}$ be the sketch-and-solve solution
- Using the distortion condition we get

$$\|\mathbf{A}\hat{\mathbf{x}} - \mathbf{b}\| \leq \frac{1+\varepsilon}{1-\varepsilon}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|$$

- for $\varepsilon = 1/3$ this yields

$$\|\mathbf{A}\hat{\mathbf{x}} - \mathbf{b}\| \leq 2\|\mathbf{A}\mathbf{x}_* - \mathbf{b}\|$$

$$\Rightarrow \text{Good? Bad?}$$

Numerics

Experiment:

- Consider a least-squares problem of size 10,000 by 100 with condition number 10^8 and residual norm 10^{-4}
- Generate SSE $d = 400$ with $\varepsilon \approx 1/2$

Findings:

- Rsidual norms:
  - sketch-and-solve: 1.13e-4
  - direct: 1.00e-4
- Forward errors:
  - sketch-and-solve: 1.06e+3
  - direct: 8.08e-7

Conclusion:

If a small enough residual is all that is needed, then sketch-and-solve is perfectly adequate. If a small forward error is needed, sketch-and-solve can be quite bad.

Can we do better?

- Sketch-and-solve is a fast way to get a low-accuracy solution to a least-squares problem
- How about iterative methods?
- Observer that

$$\mathbf{SA} = \mathbf{QR} \Rightarrow \mathbf{A}^\top \mathbf{A} \approx (\mathbf{SA})^\top (\mathbf{SA}) = \mathbf{R}^\top \mathbf{Q}^\top \mathbf{QR} = \mathbf{R}^\top \mathbf{R}$$

- Using normal equations we can then solve the LSP iteratively

  i) Solving
  $$(\mathbf{A}^\top \mathbf{A})\mathbf{x} = \mathbf{A}^\top \mathbf{b} \Rightarrow \mathbf{x} \approx \mathbf{x}_1 = \mathbf{R}^{-1}\mathbf{R}^{-\top}\mathbf{A}^\top \mathbf{b}$$

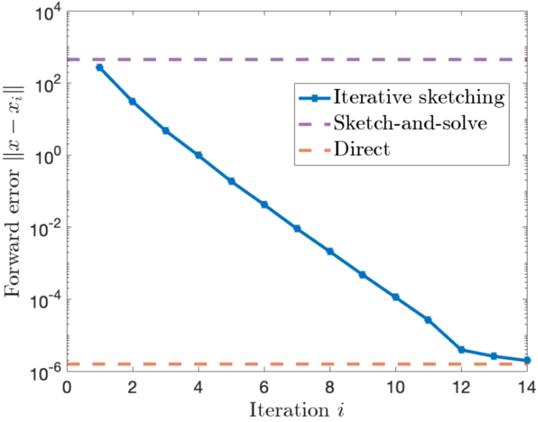
  ii) Solve for the resiudal
  $$\mathbf{A}^\top \mathbf{A}(\mathbf{x} - \mathbf{x}_0) = \mathbf{A}^\top (\mathbf{b} - \mathbf{A}\mathbf{x}_0) \Rightarrow \mathbf{x} \approx \mathbf{x}_2 = \mathbf{x}_1 + \mathbf{R}^{-1}\mathbf{R}^{-\top}\mathbf{A}^\top (\mathbf{b} - \mathbf{A}\mathbf{x}_1)$$

  $$\vdots$$

  n) $\mathbf{x}_n = \mathbf{x}_{n-1} + \mathbf{R}^{-1}\mathbf{R}^{-\top}\mathbf{A}^\top (\mathbf{b} - \mathbf{A}\mathbf{x}_{n-1})$

$$\Rightarrow \text{Iterative sketching}$$

Comparison



Iterative sketching
Sketch-and-solve
Direct
Forward error $\|x - x_i\|$
Iteration i
10^4
10^2
10^0
10^{-2}
10^{-4}
10^{-6}
0
2
4
6
8
10
12
14