Deck 10: Randomized least squares

Math 7870: Topics in Randomized Numerical Linear Algebra

Spring 2026

Akil Narayan

(Linear) Embeddings

Let $V \subset \mathbb{R}^n$ be an arbitrary set. A linear map (matrix) $\boldsymbol{S} : \mathbb{R}^n \to \mathbb{R}^d$ is called an ℓ^2 **embedding of distortion** $\epsilon \in (0, 1)$ on V if,

$$(1 - \epsilon)\|\boldsymbol{x}\|_2 \leqslant \|\boldsymbol{S}\boldsymbol{x}\|_2 \leqslant (1 + \epsilon)\|\boldsymbol{x}\|_2, \qquad \boldsymbol{x} \in V.$$

We know how to generate $\boldsymbol{S}$ randomly to achieve this. In particular:

$$d \gtrsim \frac{w(V)^2}{\epsilon^2}$$

is a sufficiently large embedding dimension for $\boldsymbol{S}$ generated via:

- iid Gaussian entries
- a sparse sign model
- a subsampled randomized Fourier transform model

Now: what can we do with this?

Application 1: Randomized least squares

Here's the generic setup: Let $\boldsymbol{A} \in \mathbb{C}^{m \times n}$, $\boldsymbol{b} \in \mathbb{C}^m$ be given.
We assume $m \gg n$, and that $\boldsymbol{A}$ has full (column) rank (for simplicity).
We seek the solution to,

$$\boldsymbol{x} = \arg\min_{\boldsymbol{y} \in \mathbb{C}^n} \|\boldsymbol{A}\boldsymbol{y} - \boldsymbol{b}\|_2^2 .$$

The "standard" approach to solve this problem when $\boldsymbol{A}$ can be stored in memory:

- (Thin QR decomposition) $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R}$.
  ($\boldsymbol{Q} \in \mathbb{C}^{m \times n}$, $\boldsymbol{R} \in \mathbb{C}^{n \times n}$, $\boldsymbol{Q}^*\boldsymbol{Q} = \boldsymbol{I}_n$, $\boldsymbol{R}$ upper triangular)
- Compute $\boldsymbol{x} = \boldsymbol{R}^{-1}\boldsymbol{Q}^*\boldsymbol{b}$.

This algorithm requires $\mathcal{O}(mn^2)$ complexity.

Strategy 1: sketch-and-solve, I

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2 \leqslant \|\boldsymbol{A}\boldsymbol{y} - \boldsymbol{b}\|_2, \qquad \boldsymbol{y} \in \mathbb{C}^n.$$

Here's the core idea for randomized least squares: Let $\boldsymbol{S}$ be an ϵ-distortion embedding for V defined as,

$$V := \operatorname{span}\{\operatorname{range}(\boldsymbol{A}), \boldsymbol{b}\}.$$

Terminology: the embedding matrix $\boldsymbol{S}$ is also called a (linear) *sketch*.

Now consider the *sketched* least squares problem:

$$\boldsymbol{z} = \operatorname*{arg\,min}_{\boldsymbol{y} \in \mathbb{C}^n} \|\boldsymbol{S}\boldsymbol{A}\boldsymbol{y} - \boldsymbol{S}\boldsymbol{b}\|_2^2.$$

The solution $\boldsymbol{z}$ is a near-optimal solution, essentially by definition:

$$\begin{aligned}(1-\epsilon)\|\boldsymbol{A}\boldsymbol{z} - \boldsymbol{b}\|_2^2 &\leqslant \|\boldsymbol{S}\boldsymbol{A}\boldsymbol{z} - \boldsymbol{S}\boldsymbol{b}\|_2^2 \\ &\leqslant \|\boldsymbol{S}\boldsymbol{A}\boldsymbol{x} - \boldsymbol{S}\boldsymbol{b}\|_2^2 \\ &\leqslant (1+\epsilon)\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2\end{aligned}$$

Strategy 1: sketch-and-solve, II

$$\boldsymbol{z} = \operatorname*{arg\,min}_{\boldsymbol{y}\in\mathbb{C}^n} \|\boldsymbol{SAy} - \boldsymbol{Sb}\|_2^2, \qquad \|\boldsymbol{Az} - \boldsymbol{b}\|_2 \leqslant \sqrt{\frac{1+\epsilon}{1-\epsilon}}\|\boldsymbol{Az} - \boldsymbol{b}\|_2.$$

Note that computing $\boldsymbol{z}$ directly by a QR decomposition requires only $\mathcal{O}(dn^2)$ effort, with $d \ll m$.

However, direct computation of the sketched system requires $\mathcal{O}(mdn)$ complexity. Since $d \gtrsim \dim V \sim n$, we haven't gained anything computationally.

The saving grace is that we can apply FFT-based sketches much faster:

If $\boldsymbol{S}$ is a subsampled random Fourier transform-based sketch, we require $d \gtrsim (n + \log m)\log n$, which achieves an ϵ-embedding for fixed $\epsilon > 0$.
This FFT-based sketch can be applied in $\mathcal{O}(mn\log d)$ complexity.

Therefore, the full cost of a subsampled randomized Fourier transform-based sketch-and-solve procedure is:

- $\mathcal{O}(mn\log d)$ complexity to generate and apply the sketch $\boldsymbol{S}$ to $\boldsymbol{A}$ and $\boldsymbol{b}$.
- $\mathcal{O}(dn^2)$ complexity to directly solve the sketched system.

Total $\mathcal{O}(mn\log d + dn^2)$ complexity.
Cf.: Direct least squares is $\mathcal{O}(mn^2)$. Significant savings when $d \sim n \ll m$.

Strategy 2: iterative sketching, I

Before we use embeddings to sketch, here's the deterministic idea: Notice first that the minimizer of $\|\boldsymbol{A}\boldsymbol{y} - \boldsymbol{b}\|_2^2$ is the same as the minimizer of,

$$f(\boldsymbol{y}) = \frac{1}{2}\boldsymbol{y}^*\boldsymbol{A}^*\boldsymbol{A}\boldsymbol{y} - (\boldsymbol{b}^*\boldsymbol{A})\boldsymbol{y}.$$

$$\boldsymbol{x} = \arg\min_{\boldsymbol{y}} f(\boldsymbol{y})$$

(In addition, the stationary point of f is the solution to the normal equations.)

One key realization from this is that we can make this process iterative simply by changing $\boldsymbol{b}$ above.

Suppose we have an approximation $\boldsymbol{x}$: $\boldsymbol{x} \approx \boldsymbol{y}$. Writing $\boldsymbol{x} = \boldsymbol{w} + \boldsymbol{y}$, the exact solution is given by,

$$\min_{\boldsymbol{v}} \|\boldsymbol{A}(\boldsymbol{v} + \boldsymbol{y}) - \boldsymbol{b}\|_2^2 = \min_{\boldsymbol{v}} \|\boldsymbol{A}\boldsymbol{v} - (\boldsymbol{b} - \boldsymbol{A}\boldsymbol{y})\|_2^2,$$

i.e.,:

$$\boldsymbol{w} = \arg\min_{\boldsymbol{v}} \frac{1}{2}\boldsymbol{v}^*\boldsymbol{A}^*\boldsymbol{A}\boldsymbol{v} - ((\boldsymbol{b} - \boldsymbol{A}\boldsymbol{y})^*\boldsymbol{A})\,\boldsymbol{v}.$$

Strategy 2: iterative sketching, II

To make this iterative, we presume that each minimization is performed approximately. For $k \geqslant 1$:

$$\boldsymbol{x}_0 = \boldsymbol{0}, \boldsymbol{w}_k \qquad \approx \arg\min_{\boldsymbol{v}} \frac{1}{2}\boldsymbol{v}^*\boldsymbol{A}^*\boldsymbol{A}\boldsymbol{v} - \left((\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_{k-1})^*\boldsymbol{A}\right)\boldsymbol{v}$$

$$\boldsymbol{x}_k \boldsymbol{x}_{k-1} + \boldsymbol{w}_k.$$

Why should we solve the minimization step approximately? This is the expensive part, requiring $\mathcal{O}(mn^2)$ complexity.

The core improvement: we can *approximately* solve this equation via sketching. We need only sketch the quadratic term:

$$\boldsymbol{w}_k = \arg\min_{\boldsymbol{v}} \frac{1}{2}\boldsymbol{v}^*\boldsymbol{A}^*\boldsymbol{S}^*\boldsymbol{S}\boldsymbol{A}\boldsymbol{v} - \left((\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_{k-1})^*\boldsymbol{A}\right)\boldsymbol{v}$$

Since we're only sketching $\boldsymbol{A}$, the embedding $\boldsymbol{S}$ needs to be an embeeding only for range($\boldsymbol{A}$).

Strategy 2: iterative sketching, III

As before, we need to sketch with a subsampled randomized Fourier transform to yield asymptotic computational gains.

Without presenting the details of the analysis:

- To achieve ϵ-proximity to the solution, strategy 1 (sketch-and-solve) requires $d \sim (n \log n)\epsilon^{-2}$. This strategy, iterative sketching, can circumvent this.
- We need only take $d \sim Cn \log n$ for a universal C here.
- Through an inductive argument: the iterative scheme is linearly convergent, so the error decays like c^k after k iterations, where $c < 1$.
- To achieve ϵ-proximity to the solution, the number of *iterations* must be $\mathcal{O}(D/\epsilon)$, where D is the "dynamic range", i.e., the relative amount of $\boldsymbol{b}$ that lies outside range($\boldsymbol{A}$):

$$D = \frac{\|\boldsymbol{P}\boldsymbol{b}\|_2}{\|\boldsymbol{b} - \boldsymbol{P}\boldsymbol{b}\|_2}, \qquad \boldsymbol{P} = \boldsymbol{Q}\boldsymbol{Q}^*, \qquad \boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R}.$$

- Overall, we need $\mathcal{O}(mn \log n)$ complexity to apply a sketch, $\mathcal{O}(dn^2)$ to directly solve a sketched least squares system, and $\mathcal{O}(mn)$ complexity for $\log(D/\epsilon)$ iterations to update the "$\boldsymbol{b}$" vector in f.