

This week: randomized numerical linear algebra

Rand NLA: Use randomization as a resource to develop improved algorithms for large-scale linear algebra computations

when used 'correctly': randomization provides an avenue for computing approximate solutions to LA problems more efficiently than deterministic algorithms.

Example 1 Randomized matrix-matrix multiplication

$$A \in \mathbb{R}^{m \times n}, \quad B \in \mathbb{R}^{n \times p}, \quad C = AB \in \mathbb{R}^{m \times p}$$

$$\text{Let } A = \begin{bmatrix} \alpha_1^T \\ \vdots \\ \alpha_m^T \end{bmatrix}, \quad B = [\beta_1, \dots, \beta_p], \quad \alpha_i, \beta_j \in \mathbb{R}^n$$

$$AB = \begin{bmatrix} \alpha_1^T \beta_1 & \dots & \alpha_1^T \beta_p \\ \vdots & & \vdots \\ \alpha_m^T \beta_1 & \dots & \alpha_m^T \beta_p \end{bmatrix}$$

complexity is $O(mnp)$ - mp inner products, each $O(n)$
 $= O(n^3)$

Fast deterministic algorithm:

Strassen's algorithm ('69): $O(n^{\log_2 7}) \approx O(n^{2.80735\dots})$

$$\begin{matrix} n/2 & n/2 \\ n/2 \end{matrix} \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{matrix} n/2 & n/2 \\ n/2 \end{matrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{matrix} n/2 & n/2 \\ n/2 \end{matrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

Compute M - M multiplication recursively:

- Let $f(n)$ be the M - M multiplication complexity for two $n \times n$ matrices

Then naive $M-M$ multiplication has

$$f(n) = \underset{\substack{\uparrow \\ \text{eight } M-M \text{ product for } n/2 \times n/2 \text{ matrices}}}{8} f\left(\frac{n}{2}\right) + \underset{\substack{\text{four } M-M \text{ addition for } n/2 \times n/2 \text{ matrices}}}{4\left(\frac{n}{2}\right)^2}$$

- Strassen improve 8 to 7:

$$P_1 = (A_{11} + A_{22})(B_{11} + B_{22})$$

$$P_2 = (A_{21} + A_{22})B_{11}$$

$$P_3 = A_{11}(B_{12} - B_{22})$$

$$P_4 = A_{22}(B_{21} - B_{11})$$

$$P_5 = (A_{11} + A_{12})B_{22}$$

$$P_6 = (A_{21} - A_{11})(B_{11} + B_{12})$$

$$P_7 = (A_{12} - A_{22})(B_{21} + B_{22})$$

$$C_{11} = P_1 + P_4 - P_5 + P_7$$

$$C_{12} = P_3 + P_5$$

$$C_{21} = P_2 + P_4$$

$$C_{22} = P_1 + P_3 - P_2 + P_6$$

8 products of $n/2 \times n/2$ matrices

+ 17 addition of $n/2 \times n/2$ matrices

- Strassen is asymptotically better than naive $M-M$ multiplication ($n \geq 10^3$)
- Best known fast $M-M$ multiplication algorithm $O(n^{2.37...})$
- If we can tolerate fairly large errors, can obtain $O(n^2)$ using randomness.

$$A = [a_1, \dots, a_n], \quad B = \begin{bmatrix} b_1^T \\ \vdots \\ b_n^T \end{bmatrix}, \quad a_i, b_j \in \mathbb{R}^n$$

$$AB = \sum_{i=1}^n a_i b_i^T \quad \text{sum of rank-1 matrices}$$

Idea: Sample $|T| \subset [n] = \{1, \dots, n\}$ and use only $a_i b_i^T$, $i \in T$

Algorithm (Drineas - Kannan, 01)

Let $p_i \geq 0$, $i \in [n]$, $\sum_{i=1}^n p_i = 1$

for $t = 1, \dots, T$

Sample $i_t \in [n]$ w\ $\mathbb{P}(i_t = j) = p_j$

make $a_{i_t} (\top p_{i_t})^{-1/2}$ col of S

$b_{i_t}^\top (\top p_{i_t})^{-1/2}$ row of R

Lemma: $\mathbb{E}[SR] = AB$

$$\text{Var}[(SR)_{ij}] = \frac{1}{T} \left(\sum_{k=1}^n \frac{A_{ik}^2 B_{kj}^2}{p_k} - (AB)_{ij}^2 \right)$$

$$\text{Pf: } \mathbb{E}[SR] = \sum_{t=1}^T \mathbb{E}_{i_t} [a_{i_t} b_{i_t}^\top / (\top p_{i_t})]$$

$$= \frac{1}{T} \sum_{t=1}^T \sum_{j=1}^n a_j b_j^\top = AB$$

$$\text{Var}[(SR)_{ij}] = \sum_{t=1}^T \text{Var}[(a_{i_t} b_{i_t}^\top / (\top p_{i_t}))_{ij}]$$

$$= \sum_{t=1}^T \left(\mathbb{E}[(a_{i_t(i)} b_{i_t(j)} / (\top p_{i_t}))^2] - (\mathbb{E}[a_{i_t(i)} b_{i_t(j)} / (\top p_{i_t})])^2 \right)$$

$$= \frac{1}{T} \left(\sum_{k=1}^n \frac{A_{ik}^2 B_{kj}^2}{p_k} - \left(\sum_{k=1}^n A_{ik} B_{kj} \right)^2 \right)$$



Thm: for $p_k = \|a_k\|_2 \|b_k\|_2 / \sum_{k=1}^n \|a_k\|_2 \|b_k\|_2$

$$\mathbb{E}[\|AB - SR\|_F^2] = \frac{1}{T} \left[\left(\sum_{k=1}^n \|a_k\|_2 \|b_k\|_2 \right)^2 - \|AB\|_F^2 \right]$$

$$\leq \frac{1}{T} \|A\|_F^2 \|B\|_F^2$$

$$\text{Pf: } \mathbb{E}[\|AB - SR\|_F^2] = \sum_{i,j} \mathbb{E}[(AB)_{ij} - (SR)_{ij}]^2$$

$$= \sum_{i,j} \text{Var}[(SR)_{ij}]$$

$$= \frac{1}{T} \sum_{i,j} \left(\sum_{k=1}^n \frac{A_{ik}^2 B_{kj}^2}{p_k} - (AB)_{ij}^2 \right)$$

$$= \frac{1}{T} \left[\sum_{k=1}^n \|a_k\|_2 \|b_k\|_2 \left(\sum_{\ell=1}^n \|a_\ell\|_2 \|b_\ell\|_2 \right) - \|AB\|_F^2 \right]$$

$$= \frac{1}{T} \left[\left(\sum_{k=1}^n \|a_k\|_2 \|b_k\|_2 \right)^2 - \|AB\|_F^2 \right] \quad \square$$

Choice of p_k also minimizes variance of error

Example 2 Randomized trace estimator

- Given $A \in \mathbb{R}^{n \times n}$, but access only through matrix-vector multiplication, i.e., for queries x_1, \dots, x_m , we can get Ax_1, \dots, Ax_m , how to approximate $\text{tr}(A) = \sum_{i=1}^n A_{ii}$

- For example, we want to compute Laplacian via matrix-vector product

$$\Delta f = \sum_{i=1}^n \frac{\partial^2 f}{\partial x_i^2} = \text{tr}(Hf), \quad Hf = \left(\frac{\partial^2 f}{\partial x_i \partial x_j} \right)_{1 \leq i, j \leq n}$$

Usually, it is easier to compute $Hf x$ via backpropagation

But we don't have access to Hf itself.

- Naive trace estimation

Set $x_i = e_i = (0, \dots, \underset{\substack{\uparrow \\ i\text{-th position}}}{0, 1}, 0, \dots, 0)^T, \quad i=1, \dots, n$

Return $\text{tr}(A) = \sum_{i=1}^n x_i^T A x_i$

Return exact trace via n matrix-vector queries

We want $\ll n$ queries by allowing for approximation.

- Hutchinson's randomized trace estimator

Algorithm

Draw $x_1, \dots, x_m \in \mathbb{R}^n$ w/ i.i.d. random $\{+1, -1\}$ entries.

Return $\tilde{T} = \frac{1}{m} \sum_{i=1}^m x_i^T A x_i$ as approximation to $\text{tr}(A)$

We can also draw x_1, \dots, x_m w/ i.i.d. Gaussian entries.

Thm $\mathbb{E} \tilde{T} = \text{tr}(A)$

Let $S = \frac{A+A^T}{2}$ be the symmetric part of A

$$\text{Var}(\tilde{T}) = \frac{1}{m} \text{Var}(x_1^T A x_1) = \frac{2}{m} \left(\|S\|_F^2 - \sum_{i=1}^n S_{ii}^2 \right) \leq \frac{2}{m} \|A\|_F^2$$

pf Since $\mathbb{E}[(x_i)_k (x_i)_l] = \delta_{kl}$

we have $\mathbb{E}[x_i^T A x_i] = \sum_{i,j} a_{ij} \mathbb{E}[(x_i)_k (x_i)_l] = \text{tr}(A)$

$$\text{Var}(x_1^T A x_1) = \text{Var}\left(\sum_{i,j} a_{ij} (x_1)_i (x_1)_j\right)$$

$$= \mathbb{E}\left[\left(\sum_{i,j} a_{ij} (x_1)_i (x_1)_j - \text{tr}(A)\right)^2\right]$$

$$= \mathbb{E}\left[\left(\sum_{i \neq j} a_{ij} (x_1)_i (x_1)_j\right)^2\right]$$

$$= \sum_{i \neq j} \sum_{k \neq l} a_{ij} a_{kl} \mathbb{E}[(x_1)_i (x_1)_j (x_1)_k (x_1)_l]$$

$$= \sum_{i \neq j} a_{ij}^2 + \sum_{i \neq j} a_{ij} a_{ji}$$

$$= \frac{1}{2} \sum_{i \neq j} S_{ij}^2 = 2 \left(\|S\|_F^2 - \sum_{i=1}^n S_{ii}^2 \right) \leq 2 \|S\|_F^2 \leq 2 \|A\|_F^2 \quad \square$$

- Roughly, to achieve ε (relative) error in trace, we need $\frac{1}{\varepsilon^2}$ queries
- Hutchison estimator performs much better when A has a flat spectrum. In this case, $\|A\|_F \ll \text{tr}(A)$ and the relative error could be much smaller.

In the extreme case $\lambda_1 \approx \lambda_2 \approx \dots \approx \lambda_n > 0$

we have $\|A\|_F = \left(\sum_{i=1}^n \lambda_i^2 \right)^{1/2} \approx \frac{1}{\sqrt{n}} \sum_{i=1}^n \lambda_i = \frac{1}{\sqrt{n}} \text{tr}(A)$

So $\text{Var}(\tilde{T}) \approx \frac{1}{mn} (\text{tr}(A))^2$

To achieve ε error in trace, we need $\frac{1}{\varepsilon^2}$ queries.

- Fast decaying spectrum: Hutch++



"truncate" the spectrum such that the fast decaying rank part is handled by sketching (or other low-rank techniques)

Example 3 Low-Rank Approximation via Randomized Algorithms

- We've learned a lot of methods to handle matrices:
 - When A has no structure: LU / QR / SVD (general but expensive)
 - When A is sparse (or Ax easy to evaluate): Krylov (sparse A usually arises in PDE problems)
- One of the most significant shifts in numerical analysis / applied math in recent years is the need to handle massive volume of data.

Challenge : 1) massive high-dim data sets / matrices

2) The structure is less explicit in many cases

3) Presence of noise and corruption in matrix entries

• How do we deal with high-dim data?

Observation : high-dim data can often be approximated with low-rank matrices

$$\begin{array}{ccccc} A & \approx & B & \cdot & C \\ m \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix} & & m \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix} & & \begin{bmatrix} \\ \\ \\ \\ \end{bmatrix}^k \end{array} \quad \leftarrow \begin{array}{l} \text{cheaper to} \\ \text{store and operate} \end{array}$$

$k < \min\{m, n\}$

Finding such B and C is not a new math problem.

We can formalize it as follows.

Goal : Given $A \in \mathbb{R}^{m \times n}$, $k < n$ (assume $m \geq n$)

$$\text{Find } \min_{\text{rank}(\hat{A}) \leq k} \|A - \hat{A}\| \quad (*)$$

Here we take $\|\cdot\|$ to be 2-norm or Frobenius norm

Solution to (*) is given by the truncated SVD of A

Let $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0$ be singular value of A

$\mathbb{R}^{m \times m} \ni U = [\vec{u}_1, \vec{u}_2, \dots, \vec{u}_m]$ be left-singular vectors of A

$\mathbb{R}^{n \times n} \ni V = [\vec{v}_1, \vec{v}_2, \dots, \vec{v}_n]$ be right-singular vectors of A

$$A = U \Sigma V^T, \quad \Sigma = \text{diag}(\sigma_1, \dots, \sigma_n) \in \mathbb{R}^{m \times n}$$

$$= \sum_{i=1}^n \sigma_i u_i v_i^T$$

$$\leftarrow \text{rank}(A) = \# \{ \text{nonzero } \sigma_i \text{'s} \}$$

Now we take the truncated SVD of A

$$A_k = \sum_{i=1}^k \sigma_i u_i v_i^T$$

$$= \begin{bmatrix} u_1 & \dots & u_k \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix} \begin{bmatrix} -v_1^T \\ \vdots \\ -v_k^T \end{bmatrix} \in \mathbb{R}^{m \times n}$$

$$= \begin{bmatrix} u_1 & \dots & u_m \end{bmatrix} \underbrace{\begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k & & 0 \\ & & & \ddots & \\ & & & & 0 \end{bmatrix}}_{=: \Sigma_k} \begin{bmatrix} -v_1^T \\ \vdots \\ -v_n \end{bmatrix}$$

clearly $\text{rank}(A_k) \leq k$,

$$\|A - A_k\|_2 = \|U(\Sigma - \Sigma_k)V^T\|_2 = \|\Sigma - \Sigma_k\|_2 = \sigma_{k+1}$$

$$\|A - A_k\|_F = \|\Sigma - \Sigma_k\|_F = \left(\sum_{j=k+1}^n \sigma_j^2 \right)^{1/2}$$

Thm (Eckart - Young)

$$\min_{\text{rank}(\hat{A}) \leq k} \|A - \hat{A}\|_2 = \sigma_{k+1}$$

$$\min_{\text{rank}(\hat{A}) \leq k} \|A - \hat{A}\|_F = \left(\sum_{j=k+1}^n \sigma_j^2 \right)^{1/2}$$

Pf: We prove the 2-norm case only.

It suffices to show that $\|A - \hat{A}\|_2 \geq \sigma_{k+1}$

for any $\text{rank}(\hat{A}) \leq k$

It suffices to show that $\exists x \in \mathbb{R}^n$, s.t. $\frac{\|(A - \hat{A})x\|_2}{\|x\|_2} \geq \sigma_{k+1}$

I want to find $x \in \mathbb{R}^n$ such that $\hat{A}x = 0$

and $x \in \text{span}\{v_1, \dots, v_{k+1}\}$ ($x = \sum_{i=1}^{k+1} \alpha_i v_i$, $\sum_{i=1}^{k+1} \alpha_i^2 = 1$)

$$\Rightarrow \frac{\|(A-\hat{A})x\|_2}{\|x\|_2} = \frac{\|Ax\|_2}{\|x\|_2} = \left\| \sum_{i=1}^{k+1} \sigma_i u_i x_i \right\|_2 \geq \sigma_{k+1}$$

Such x always exists:

Since $\text{rank}(\hat{A}) \leq k$, we know $\dim \text{Null}(\hat{A}) \geq n-k$

but $\dim \text{span}\{v_1, \dots, v_{k+1}\} = k+1$

$$\Rightarrow \text{Null}(\hat{A}) \cap \text{span}\{v_1, \dots, v_{k+1}\} \neq \emptyset$$



∴ The best rank k approximation of A is given by
 k -truncated SVD of A