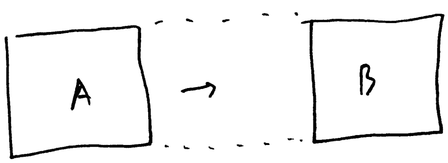
Lecture 09 - Randomized Linear Algebra

First Project ideas & questions.

Motivating Example: SVD-based FMM:



Evaluate $\varphi(\vec{x}) = \int_A K(\vec{x},\vec{y})\, f(\vec{y})\, d\vec{y}$ for $\vec{x} \in B$.

If K ~~is sufficiently smooth and~~ satisfies

$$\iint_{A\times B} |K(\vec{x},\vec{y})|^2\, dx\,dy < \infty$$

then ~~th~~ there exists u_k, v_k, s_k (orthonormal)

$s_k \geq 0$, $s_{k+1} \geq s_k$ such that

$$K(\vec{x},\vec{y}) = \sum_{k=1}^{\infty} u_k(\vec{x})\, s_k\, v_k(\vec{y})$$

(in the least squares sense)

This is the "SVD of an operator",

__If__ $K(\vec{x},\vec{y}) = \sum_{k=1}^{p} u_k(\vec{x})\, s_k\, v_k(\vec{y})$ then

$$\varphi(\vec{x}) = \sum u_k(\vec{x})\, s_k \underbrace{\int_A v_k(\vec{y})\, f(\vec{y})\, d\vec{y}}_{\text{moments of } f \text{ w.r.t. } v_k.}$$

__Idea:__ Construct a numerical, "interpolatory", version of the above decomposition in order to efficiently build translation operators, etc.

To do this we need to do some linear algebra...

__The SVD of a matrix__

For __any__ $m \times n$ matrix A (real or complex), with rank k,

$$\underset{m\times n}{A} = \underset{m\times k}{U}\ \underset{k\times k}{S}\ \underset{n\times k}{V^*}$$

U, V unitary: $U^*U = I_{k\times k}$

$V^*V = I_{k\times k}$

$\operatorname{span}\{U\} = \operatorname{col} A$, $\operatorname{span}\{V^t\} = \operatorname{row} A = \operatorname{col} A^t$

$$S = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{pmatrix}$$ singular values of A

<u>Lemma</u>: The SVD provides the <u>best</u> rank ~~k~~ k approximation to any matrix in the L^2, spectral norm:

$$\| A - VSV^t \| \leq \epsilon$$

where $\epsilon = \sqrt{\sum_{p=k+1}^{\min(m,n)} \sigma_p^2}$

Mention the numerical rank of a matrix

<u>Computation of the SVD</u>:

σ_k's are <u>not</u> the eigenvalues of A (unless A is ~~the~~ selfadjoint, and then $\sigma_k = |\lambda_k|$).

- σ_k^2's are the eigenvalues of $A^* A$.

Modern method: "Householder reflection"

$$A = \begin{pmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{pmatrix}$$

$$U_1^* A = \begin{pmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \end{pmatrix}, \quad U_1^* A V_1 = \begin{pmatrix} x & x & 0 & 0 \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \end{pmatrix}$$

$$\Rightarrow \quad U^* A V = \begin{pmatrix} x & x & 0 & 0 \\ 0 & x & x & 0 \\ 0 & 0 & x & x \\ 0 & 0 & 0 & x \end{pmatrix}$$

bidiagonal form

$\Rightarrow$ $A = U B V^t$

bidiagonal.

Step 2: Bring $B \to \tilde{U} S \tilde{V}^t$ using "Jacobi rotations", for example.

This procedure can be tailored to stop once some precision has been met, or finished and then truncated.

Note: U, V in general have nothing to do with the matrix A, and applying them requires $O(km)$ or $O(kn)$ operations.

~~An alternative, relatively modern factorization~~

$\Rightarrow$ computing "translations" might be done: nk^2, mk^2.

An alternative, relatively modern matrix factorization:

The Interpolation Decomposition (ID):

$$\cancel{P}\; E A T$$

$$A = \cancel{P}\; A_{S}\; P_r \qquad \|\cdot\| \sim 1$$

$k \times k$ submatrix (<u>not</u> contiguous)

$\|\cdot\| \sim 1$

E, P_q contain $k \times k$ permutation matrices

$\Rightarrow$ <u>interpolation matrices</u>

$\Rightarrow$ Each column of A is a linear combination of some k other columns

$\Rightarrow$ Each row of A ...

Another advantage:

Applying this factorization: $O(\cancel{nm}(n+m-k) \cdot k)$

Applying the SVD: $O((n+m)k)$

In particular:

Thm There exists a factorization

$$A = P_L \begin{bmatrix} I \\ S \end{bmatrix} A_S \begin{bmatrix} I & T \end{bmatrix} P_R^* + X$$

$A_S \sim k \times k$

$S \sim m-k \times k$

$T \sim n-k \times k$

$A_S =$ top left $k \times k$ of $P_L^* A P_R^*$

with

$$\|S\|_F \leq \sqrt{k(m-k)}$$

$$\|T\|_F \leq \sqrt{k(n-k)}$$

$$\|X\|_2 \leq \sigma_{k+1}(A) \sqrt{1 + k(\min(m,n) - k)}$$

small if σ_{k+1} is small.

$$\|A\|_F = \sqrt{\sum_i \sum_j |a_{ij}|^2}$$

To compute this factorization:

① compute $AP_R = Q[R_{11} \mid R_{12}]$

② solve for T:

$$R_{11} T = R_{12}$$

$$\Rightarrow A = A_{cs} [I \; T] P_R^*, \quad A_{cs} = \text{first } k \text{ cols of } AP_R$$

(3) Perform analogous factorization on A_{cs}^T

~~A_s~~ to obtain $A_{cs} = P_L \begin{bmatrix} I \\ S \end{bmatrix} A_s$

$\Rightarrow \quad A_s =$ first k ~~columns~~ rows of $P_L^* A_{cs}$

(4) Then

$$A = A_{cs} \begin{bmatrix} I & T \end{bmatrix} P_R^*$$

$$= P_L \begin{bmatrix} I \\ S \end{bmatrix} A_s \begin{bmatrix} I & T \end{bmatrix} P_R^*$$

Note: Crucial that the QR factorization be computed accurately (modified Gram-Schmidt with reorthogonalization)

Cost: similar to QR : $\mathcal{O}(mnk)$, usually the constant is much smaller than the SVD (and SVD is often $\mathcal{O}(mn \cdot \min(m,n))$)

Why is this factorization useful?

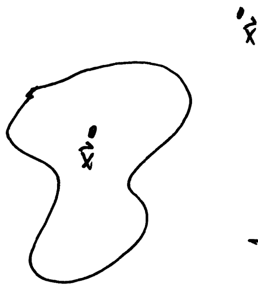
Example "Recursive skeletonization"

From an integral equation

$$\sigma(x) + \int K(x,y)\, \sigma(y)\, dy = f(x)$$

K is $\log|\vec{x}-\vec{y}|$, $\frac{1}{|\vec{x}-\vec{y}|}$, ... "PDE kernel"

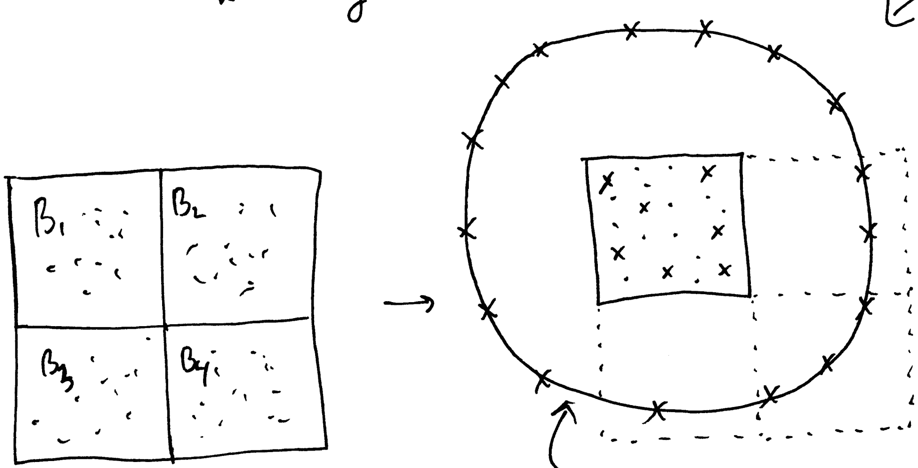
Green's 3rd identity:



$$u(\vec{x}) = \pm\int \frac{\partial G}{\partial n}\, u(\vec{y})\, d\vec{y} \mp \int G(\vec{x},\vec{y})\, \frac{\partial u}{\partial n}(\vec{y})\, dy$$

This means that u inside and outside can be reconstructed from linear combination of G and $\frac{\partial G}{\partial n}$ on the boundary.

Interaction of B_1 with outside word is rank k — and so with this disc.



proxy surface

~~Choose~~

Skeletonize

$\tilde{K} = \cancel{K(x_i, p_j)}$

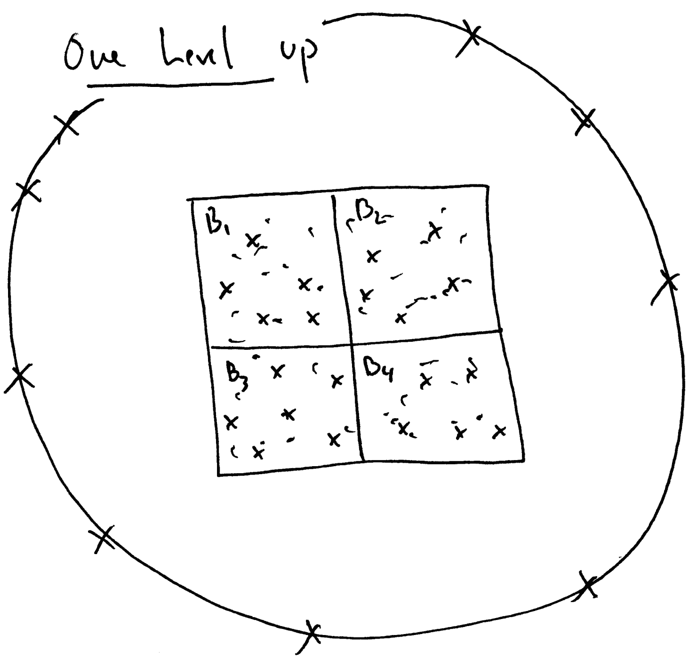
$K(p_i, x_j)$

↑

proxy points

x- skeleton points

One Level up



Re-skeletonize the points selected in each of $B_1, .., B_4$:

$$\tilde{K} = \left[K(\tilde{p}_i, \vec{x}_j^{B_1}) \;\cdots\; K(\tilde{p}_i, \vec{x}_j^{B_4}) \right]$$

We will return to these ideas when we discuss ~~a~~ "fast direct solvers" ....

For now: How do we compute these factorization fast? ~~[illegible]~~

Randomized SVD

~~The algorithm:~~ Goal: compute approximate rank-k SVD $A \simeq USV^t$

Algo

① Find Q s.t. $A \approx QQ^*A$
orthogonal projector

(Assume $A \sim m \times n$
$k \ll m,n$)

② Form $\underset{k \times n}{B} = Q^*A$

③ Compute SVD of B: $B = \tilde{U} S V^*$

④ Form $U = Q\tilde{U}$

$$\Rightarrow \| A - USV^* \| = \| A - Q\tilde{U}SV^* \| = \| A - QB \|$$

$$= \| A - QQ^*A \| \sim \text{small by assumption.}$$

How do we find Q?

~~Randomized svd~~

Directly: Gram-Schmidt process

Indirectly: Randomized sampling

1. Draw a "random matrix" $\Omega_{n \times k}$

$$\Omega = \begin{pmatrix} w_{11} & \cdots & \\ w_{21} & \ddots & \\ \vdots & & \\ w_{n1} & & w_{nk} \end{pmatrix} \qquad w_{ij} \sim N(0,1) \quad \text{for example}$$

2. "Sample" the matrix A:

$$Y = A\Omega$$

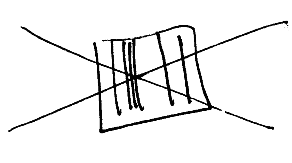
3. Compute Q s.t. $QQ^* Y = Y$

<u>Note</u> If A has <u>exact</u> rank k then

$$\operatorname{span}\{\vec{q}_1 \ldots \vec{q}_k\} = \operatorname{col} A \quad \text{with probability } 1$$

<u>Generally</u>: Y does a pretty good job since it is "mostly" directed in the directions corresponding to the k largest singular vectors.

<u>NOTE</u> Randomized sampling does <u>not</u> pick rows/columns



<u>Do not do this (usually)</u>

How about the interpolation decomposition?

Goal: Compute $A \approx L A_R$ ← "row skeleton"

<u>Algorithm</u>

1. Draw $n \times k$ random matrix Ω
2. Form $Y = A\Omega$ ← $m \times k$
3. Compute ID of Y: $Y = L Y_R$ ← select the rows $i_1, \dots, i_k$

Then $A \approx L A_R$ ← select the rows $i_1, \dots, i_k$ from A

(Matlab notation: $A_R = A(\vec{i}, :)$
$\vec{i} = (i_1 \cdots i_k)$)

Why is this an "analysis-based" algorithm?

In general, A will come from some continuous integral operator $K(x,y)$. We know a lot about K, including how its singular values decay. The previous algorithms' accuracy depends on the behavior of the sing.-vals, can be shown using probabilistic arguments.

Computational savings

Randomized SVD:

$$Y = A\Omega \qquad \sim \quad O(mnk)$$

$$\underset{m\times k}{Y} = \underset{m\times k}{Q}\,\underset{k\times k}{R} \qquad \sim \quad O(mk^2)$$

$$\underset{k\times n}{B} = Q^*A \qquad \sim \quad O(mnk)$$

$$\underset{k\times n}{B} = \tilde{U}SV^* \qquad \sim \quad O(nk^2)$$

$$U = \underset{m\times k}{Q}\,\underset{k\times k}{\tilde{U}} \qquad \sim \quad O(mk^2)$$

$$\text{Total}: \quad O(2mnk + (m+n)k^2) \quad \sim \quad \underbrace{O(2mnk)}$$

the constant in this is <u>very</u> small, ~1 since it is only matrix-matrix multiplies.

$Y = A\Omega$

<u>Case 1</u> Direct: $O(mnk)$

<u>Case 2</u> Fast apply: $O((m+n)\ell)$

<u>Case 3</u> Use SRFT : $O(mn \log k)$

An even faster algorithm

Idea sample the row and column space separately using a subsampled-random-Fourier Transform: (SRFT)

1. Form SRFT $\underset{n\times k}{\Omega_1}, \underset{m\times k}{\Omega_2}$

Choice (i)

$$\Omega = \underset{\substack{n\times n\\ \text{random diagonal } e^{i\theta}}}{D}\ \underset{\substack{n\times n\\ \text{FFT}}}{F}\ \underset{\substack{n\times k\\ \text{random permutation}}}{S}$$

2. Form $Y = A\Omega_1$

$\underbrace{Z}_{n\times m} = A^*\Omega_2 \iff Z^* = \Omega_2^* A = Z^* W W^*$

3. Find Q, W s.t. $Y = QQ^*Y$, $Z = \underset{n\times k}{W}\underset{k\times k}{W^*}Z$ $\quad \|A - QQ^*AWW^*\|$

4. Solve for $k\times k$ matrix T:

$$\left.\begin{aligned} Q^*Y &= T(W^*\Omega_1) \\ W^*Z &= T^*(Q^*\Omega_2) \end{aligned}\right\} \Rightarrow \begin{aligned} QQ^*Y &= QTW^*\Omega_1 = Y \\ WW^*Z &= WT^*Q^*\Omega_2 = Z \\ \hookrightarrow Z^*WW^* &= \Omega_2^*QTW^* = Z^* \end{aligned}$$

$$\left.\begin{aligned} Q^*A\Omega_1 &= TW^*\Omega_1 \\ W^*A^*\Omega_2 &= T^*Q^*\Omega_2 \end{aligned}\right\} \quad \begin{aligned} Q^*A &\sim TW^* \\ Q^*AW &\sim T \end{aligned}$$

5. SVD compute: $T = \tilde{U}S\tilde{V}^t$

6. Form $U = Q\tilde{U}$, $V = W\tilde{V}$ $\Rightarrow$

$$\begin{aligned} \|A - USV^*\| &= \|A - Q\tilde{U}S\tilde{V}^*W^*\| \\ &= \|A - QTW^*\| \\ &= \|A - QQ^*AWW^*\| \\ &\sim \text{small} \end{aligned}$$