

Fast algorithms for hierarchical matrices

Machine Learning Seminar May 2, 2019

What is fast?

Suppose $A \in \mathbb{C}^{N \times N}$, and, $v \in \mathbb{C}^N$

- Matvec $A \cdot v$: $O(N^2)$
- Inversion A^{-1} : $O(N^3)$
- **Determinants** det A : $O(N^3)$

For a given task, an algorithm is fast if it's runtime beats the asymptotic complexity The dream: $O(N \log^{s} N)$

Examples

- Sparse matrices, matvecs in O(kN), if well-conditioned, inverse in O(kN)
- FFT matrices, matvecs in O(N log N), inverse analytically known, and inverse application in O(N log N)
- FMM matrices, matvecs in O(N)

Dense matrices \neq **Data dense**

$$A_{j,k} = \delta_{j,k} + \cos(t_j - s_k)$$

= $\delta_{j,k} + \cos(t_j)\cos(s_k) + \sin(t_j)\sin(s_k)$
$$f_j$$

• Matvec
$$\boldsymbol{b} = A \cdot \boldsymbol{v}$$
: $O(N^2)$

Step 1:

$$W_1 = \sum_{k=1}^N \cos(s_k) v_k, \quad W_2 = \sum_{k=1}^N \sin(s_k) v_k$$

Step 2:

$$b_j = v_j + \cos(t_j)W_1 + \sin(t_j)W_2$$
 O(N)

$$A = I + UV^{T}, \quad U = \begin{bmatrix} \cos(t_{1}) & \sin(t_{1}) \\ \cos(t_{2}) & \sin(t_{2}) \\ \vdots & \vdots \\ \cos(t_{N}) & \sin(t_{N}) \end{bmatrix}, \quad V = \begin{bmatrix} \cos(s_{1}) & \sin(s_{1}) \\ \cos(s_{2}) & \sin(s_{2}) \\ \vdots & \vdots \\ \cos(s_{N}) & \sin(s_{N}) \end{bmatrix}$$

• Inversion A^{-1} : $O(N^3)$

Sherman Morrison Woodbury formula: $A^{-1} = I - U(I_2 + V^T U)^{-1} V^T$

O(N)!

O(N)!

• **Determinants** det A : $O(N^3)$

Slyvester formula formula: det $A = det (I_2 + V^T U)$

Two-point boundary value problems

Given $p(x), q(x), f(x), \alpha, \beta$, find u(x) which satisfies u''(x) + p(x)u'(x) + q(x)u(x) = f(x) 0 < x < 1. $u(0) = \alpha$, $u(1) = \beta$.

 $u \rightarrow u(x) - \alpha - (\beta - \alpha)x$ satisfies

$$u''(x) + p(x)u'(x) + q(x)u(x) = \tilde{f}(x) \quad 0 < x < 1.$$

$$u(0) = 0, \quad u(1) = 0.$$

G(x, y): Green's function for

$$u''(x) = \delta_y \quad 0 < x < 1.$$

$$u(0) = 0, \quad u(1) = 0.$$

$$G(x, y) = \begin{cases} x(1-y) & 0 \le x < y \le 1\\ y(1-x) & 0 \le y < x < \le 1 \end{cases}$$

Integral formulation: $u(x) = \int_0^1 G(x, y)\sigma(y)$, σ unknown density

- Boundary conditions
- ODE yields following integral equation

$$\sigma(x) + p(x) \int_0^1 \frac{\partial G}{\partial x}(x, y) \sigma(y) \, dy + q(x) \int_0^1 G(x, y) \sigma(y) \, dy = \tilde{f}, \quad 0 < x < 1$$



Structure of off-diagonal blocks

$$\sigma(x) + p(x) \int_{0}^{1} \frac{\partial G}{\partial x}(x, y)\sigma(y) \, dy + q(x) \int_{0}^{1} G(x, y)\sigma(y) \, dy = P\sigma = \tilde{f}$$

$$\begin{bmatrix} P_{AA} & P_{AB} \\ P_{BA} & P_{BB} \end{bmatrix} \begin{bmatrix} \sigma_{A} \\ \sigma_{B} \end{bmatrix} = \begin{bmatrix} \tilde{f}_{A} \\ \tilde{f}_{B} \end{bmatrix}$$

$$P_{AB}\sigma_{B} = p(x) \int_{B} (1 - y)\sigma(y) \, dy + q(x) \cdot x \int_{B} (1 - y)\sigma(y) \, dy$$

$$P_{AB} = \left(p(x) + q(x) \cdot x \right) \int_{B} (1 - y) \cdot * dy$$

$$Rank 1$$

$$u_{A} \qquad v_{B}^{T}$$

Structure of off-diagonal blocks

$$\sigma(x) + p(x) \int_{0}^{1} \frac{\partial G}{\partial x}(x, y)\sigma(y) \, dy + q(x) \int_{0}^{1} G(x, y)\sigma(y) \, dy = P\sigma = \tilde{f}$$

$$\begin{bmatrix} P_{AA} & P_{AB} \\ P_{BA} & P_{BB} \end{bmatrix} \begin{bmatrix} \sigma_{A} \\ \sigma_{B} \end{bmatrix} = \begin{bmatrix} \tilde{f}_{A} \\ \tilde{f}_{B} \end{bmatrix}$$

$$P_{AB}\sigma_{B} = p(x) \int_{B} (1 - y)\sigma(y) \, dy + q(x) \cdot x \int_{B} (1 - y)\sigma(y) \, dy$$

$$G(x, y) = \begin{cases} x(1 - y) & 0 \le x < y \le 1 \\ y(1 - x) & 0 \le y < x < \le 1 \end{cases}$$

$$P_{AB} = (p(x) + q(x) \cdot x) \int_{B} (1 - y) \cdot * \, dy$$

$$Rank 1$$

$$u_{A} \qquad v_{B}^{T}$$

$$P_{BA}\sigma_{A} = -p(x)\int_{A} y\sigma(y) \, dy + q(x) \cdot (1-x)\int_{A} y\sigma(y) \, dy$$
$$P_{AB} = \left(-p(x) + q(x) \cdot (1-x)\right)\int_{A} y \cdot * \, dy$$
Rank 1
$$u_{B}$$
 v_{A}^{T}

Faster? Matvec and inverse

$$\begin{bmatrix} P_{AA} & u_A v_B^T \\ u_B v_A^T & P_{BB} \end{bmatrix} \begin{bmatrix} \sigma_A \\ \sigma_B \end{bmatrix} = \begin{bmatrix} \tilde{f}_A \\ \tilde{f}_B \end{bmatrix}$$

Matvec

Step 1:

Compute $u_B v_A^T \sigma_A, u_A v_B^T \sigma_B$ **O(N)**



Faster? Matvec and inverse

$$\begin{bmatrix} P_{AA} & u_A v_B^T \\ u_B v_A^T & P_{BB} \end{bmatrix} \begin{bmatrix} \sigma_A \\ \sigma_B \end{bmatrix} = \begin{bmatrix} \tilde{f}_A \\ \tilde{f}_B \end{bmatrix}$$

Matvec

Step 1:

Compute $u_B v_A^T \sigma_A, u_A v_B^T \sigma_B$ **O(N)**





Faster? Matvec and inverse

$$\begin{bmatrix} P_{AA} & u_A v_B^T \\ u_B v_A^T & P_{BB} \end{bmatrix} \begin{bmatrix} \sigma_A \\ \sigma_B \end{bmatrix} = \begin{bmatrix} \tilde{f}_A \\ \tilde{f}_B \end{bmatrix}$$

Matvec

Step 1:

Compute
$$u_B v_A^T \sigma_A$$
, $u_A v_B^T \sigma_B$ **O(N)**





Inverse

$$\begin{array}{lll} \mathbf{hverse} & D & UV^{T} \\ \begin{bmatrix} P_{AA} & u_{A}v_{B}^{T} \\ u_{B}v_{A}^{T} & P_{BB} \end{bmatrix} = \begin{bmatrix} P_{AA} & 0 \\ 0 & P_{BB} \end{bmatrix} + \begin{bmatrix} u_{A} & 0 \\ 0 & u_{B} \end{bmatrix} \begin{bmatrix} v_{A}^{T} & 0 \\ 0 & v_{B}^{T} \end{bmatrix} \\ & N \times 2 & 2 \times N \end{array}$$

$$P^{-1} = \begin{bmatrix} P_{AA}^{-1} & 0 \\ 0 & P_{BB}^{-1} \end{bmatrix} - \begin{bmatrix} P_{AA}^{-1}u_{A} & 0 \\ 0 & P_{BB}^{-1}u_{B} \end{bmatrix} \begin{pmatrix} I_{2} + \begin{bmatrix} v_{A}^{T}P_{AA}^{-1}u_{A} & 0 \\ 0 & v_{B}^{T}P_{BB}^{-1}u_{B} \end{bmatrix} \end{pmatrix}^{-1} \begin{bmatrix} v_{A}^{T}P_{AA}^{-1} & 0 \\ 0 & v_{B}^{T}P_{BB}^{-1} \end{bmatrix} O(N^{3}/4)$$

$$D^{-1} - D^{-1}U(I + V^{T}D^{-1}U)^{-1}V^{T}D^{-1}$$

Structure of off-diagonal blocks

$$\begin{bmatrix} P_{AA} & P_{AB} & P_{AC} & P_{AD} \\ P_{BA} & P_{BB} & P_{BC} & P_{BD} \\ P_{CA} & P_{CB} & P_{CC} & P_{CD} \\ P_{DA} & P_{DB} & P_{DC} & P_{DD} \end{bmatrix} \begin{bmatrix} \sigma_A \\ \sigma_B \\ \sigma_C \\ \sigma_D \end{bmatrix} = \begin{bmatrix} \tilde{f}_A \\ \tilde{f}_B \\ \tilde{f}_C \\ \tilde{f}_D \end{bmatrix}$$



All off-diagonal blocks are rank 1

$$\begin{bmatrix} P_{AA} & u_{A,R}v_{B,L}^T & u_{A,R}v_{C,L}^T & u_{A,R}v_{D,L}^T \\ u_{B,L}v_{A,R}^T & P_{BB} & u_{B,R}v_{C,L}^T & u_{B,R}v_{D,L}^T \\ u_{C,L}v_{A,R}^T & u_{C,L}v_{B,R}^T & P_{CC} & u_{C,R}v_{D,L}^T \\ u_{D,L}v_{A,R}^T & u_{D,L}v_{B,R}^T & u_{D,L}v_{C,R}^T & P_{DD} \end{bmatrix} \begin{bmatrix} \sigma_A \\ \sigma_B \\ \sigma_C \\ \sigma_D \end{bmatrix} = \begin{bmatrix} \tilde{f}_A \\ \tilde{f}_B \\ \tilde{f}_C \\ \tilde{f}_D \end{bmatrix}$$

$$u_{I,L} = -p(x) + (1 - x) \cdot q(x), \quad x \in I \qquad \qquad u_{I,R} = p(x) + x \cdot q(x), \quad x \in I$$
$$v_{I,L} = 1 - y, \quad y \in I \qquad \qquad v_{I,R} = y, \quad y \in I$$

$$\begin{bmatrix} P_{AA} & u_{A,R}v_{B,L}^T & u_{A,R}v_{C,L}^T & u_{A,R}v_{D,L}^T \\ u_{B,L}v_{A,R}^T & P_{BB} & u_{B,R}v_{C,L}^T & u_{B,R}v_{D,L}^T \\ u_{C,L}v_{A,R}^T & u_{C,L}v_{B,R}^T & P_{CC} & u_{C,R}v_{D,L}^T \\ u_{D,L}v_{A,R}^T & u_{D,L}v_{B,R}^T & u_{D,L}v_{C,R}^T & P_{DD} \end{bmatrix}$$



$$\begin{bmatrix} P_{AA} & u_A \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} v_B^T & u_A \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} v_C^T & u_A \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} v_D^T \\ u_B \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} v_A^T & P_{BB} & u_B \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} v_C^T & u_B \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} v_D^T \\ u_C \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} v_A^T & u_C \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} v_B^T & P_{CC} & u_C \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} v_D^T \\ u_D \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} v_A^T & u_D \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} v_B^T & u_D \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} v_C^T & P_{DD} \end{bmatrix}$$

$$u_I = \begin{pmatrix} u_{I,L} & u_{I,R} \end{pmatrix}$$

$$v_I^T = \begin{pmatrix} v_{I,L}^T \\ v_{I,R}^T \end{pmatrix}$$

$$P_{i,j} = u_i S_{i,j} v_j^T$$

$$\begin{bmatrix} P_{AA} & u_{A,R}v_{B,L}^T & u_{A,R}v_{C,L}^T & u_{A,R}v_{D,L}^T \\ u_{B,L}v_{A,R}^T & P_{BB} & u_{B,R}v_{C,L}^T & u_{B,R}v_{D,L}^T \\ u_{C,L}v_{A,R}^T & u_{C,L}v_{B,R}^T & P_{CC} & u_{C,R}v_{D,L}^T \\ u_{D,L}v_{A,R}^T & u_{D,L}v_{B,R}^T & u_{D,L}v_{C,R}^T & P_{DD} \end{bmatrix}$$



$$\begin{bmatrix} P_{AA} & u_A \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} v_B^T & u_A \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} v_C^T & u_A \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} v_D^T \\ u_B \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} v_A^T & P_{BB} & u_B \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} v_C^T & u_B \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} v_D^T \\ u_C \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} v_A^T & u_C \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} v_B^T & P_{CC} & u_C \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} v_D^T \\ u_D \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} v_A^T & u_D \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} v_B^T & u_D \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} v_C^T & P_{DD} \end{bmatrix}$$

$$u_I = \begin{pmatrix} u_{I,L} & u_{I,R} \end{pmatrix}$$

$$v_I^T = \begin{pmatrix} v_{I,L}^T \\ v_{I,R}^T \end{pmatrix}$$

$$P_{i,j} = u_i S_{i,j} v_j^T$$

$$\begin{bmatrix} P_{AA} & u_A \begin{pmatrix} 1 & 0 \end{pmatrix} v_B^* & u_A \begin{pmatrix} 1 & 0 \end{pmatrix} v_C^* & u_A \begin{pmatrix} 1 & 0 \end{pmatrix} v_D^* \\ 1 & 0 \end{pmatrix} v_D^T \\ u_B \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} v_A^T & P_{BB} & u_B \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} v_C^T & u_B \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} v_D^T \\ u_C \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} v_A^T & u_C \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} v_B^T & P_{CC} & u_C \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} v_D^T \\ u_D \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} v_A^T & u_D \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} v_B^T & u_D \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} v_C^T & P_{DD} \end{bmatrix}$$



$$P_{i,j} = u_i S_{i,j} v_j^T$$



$$\begin{bmatrix} P_{AA} & u_A \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} v_B^T & u_A \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} v_C^T & u_A \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} v_D^T \\ u_B \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} v_A^T & P_{BB} & u_B \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} v_C^T & u_B \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} v_D^T \\ u_C \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} v_A^T & u_C \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} v_B^T & P_{CC} & u_C \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} v_D^T \\ u_D \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} v_A^T & u_D \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} v_B^T & u_D \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} v_C^T & P_{DD} \end{bmatrix} \qquad \begin{array}{c} \mathbf{A} \quad \mathbf{B} \quad \mathbf{C} \quad \mathbf{D} \\ \mathbf{A} \quad \mathbf{A} \quad \mathbf{B} \quad \mathbf{C} \quad \mathbf{D} \\ \mathbf{A} \quad \mathbf{B} \quad \mathbf{C} \quad \mathbf{C} \quad \mathbf{C} \\ \mathbf{A} \quad \mathbf{A} \quad \mathbf{B} \quad \mathbf{C} \quad \mathbf{C} \\ \mathbf{A} \quad \mathbf{A} \quad \mathbf{C} \quad \mathbf{C} \quad \mathbf{C} \\ \mathbf{A} \quad \mathbf{C} \quad \mathbf{C} \quad \mathbf{C} \quad \mathbf{C} \quad \mathbf{C} \\ \mathbf{C} \quad \mathbf$$



 $\hat{D} = (V^T D^{-1} U)^{-1}, \quad E = D^{-1} U \hat{D}, \quad F = (\hat{D} V^T D^{-1})^T, \quad G = D^{-1} - D^{-1} U \hat{D} V^T D^{-1}$ Recall inverse with two intervals: $P^{-1} = \begin{bmatrix} P_{AA}^{-1} & 0 \\ 0 & P_{BB}^{-1} \end{bmatrix} - \begin{bmatrix} P_{AA}^{-1} u_A & 0 \\ 0 & P_{BB}^{-1} u_B \end{bmatrix} \begin{pmatrix} I_2 + \begin{bmatrix} v_A^T P_{AA}^{-1} u_A & 0 \\ 0 & v_B^T P_{BB}^{-1} u_B \end{bmatrix} \end{pmatrix}^{-1} \begin{bmatrix} v_A^T P_{AA}^{-1} & 0 \\ 0 & v_B^T P_{BB}^{-1} \end{bmatrix}$

Sparse matrix embedding of P





Matvec: $O(N^2/4)$

Inverse: $O(N^3/32)$

Not fast enough!

S admits a similar factorization

Structure of S



Hierarchical block separable (HBS) form of P

$$P = U^{(3)} \left(U^{(2)} \left(U^{(1)} D^{(0)} \left(V^{(1)} \right)^T + D^{(1)} \right) \left(V^{(2)} \right)^T + D^{(2)} \right) \left(V^{(3)} \right)^T + D^{(3)}$$
$$P^{-1} = E^{(3)} \left(E^{(2)} \left(E^{(1)} G^{(0)} \left(F^{(1)} \right)^T + G^{(1)} \right) \left(F^{(2)} \right)^T + G^{(2)} \right) \left(F^{(3)} \right)^T + G^{(3)}$$



All matrices except $G^{(0)}$ are block diagonal

Charge embedding of URC form of D



Sparse embedding of HRS form of P - inversion

$$P^{-1} = E^{(3)} \left(E^{(2)} \left(E^{(1)} G^{(0)} \left(F^{(1)} \right)^T + G^{(1)} \right) \left(F^{(2)} \right)^2 + G^{(2)} \right) \left(F^{(3)} \right)^2 + G^{(3)}$$



Inversion: O(N)

Sparse embedding of HBS form of P - inversion

$$P^{-1} = E^{(3)} \left(E^{(2)} \left(E^{(1)} G^{(0)} \left(F^{(1)} \right)^T + G^{(1)} \right) \left(F^{(2)} \right)^2 + G^{(2)} \right) \left(F^{(3)} \right)^2 + G^{(3)}$$



Inversion: O(N)

Higher dimensions?

Given f(x), find u(x) which satisfies

 $\Delta u(x) = 0 \quad x \in \Omega.$ $u(x) = f(x), \quad x \in \Gamma.$

G(x, y): Green's function for

$$\Delta u(x) = \delta_{y}$$



Boundary conditions yields following integral equation

$$\sigma(x) + 2 \int_{\Gamma} \frac{\partial}{\partial \nu} G(x, y) \sigma(y) \, dS_y = 2f(x) \qquad \qquad \mathbf{P}\boldsymbol{\sigma} = \mathbf{f}$$

P is also HBS compressible!

Off-diagonal blocks

$$\begin{bmatrix} P_{AA} & P_{AB} & P_{AC} & P_{AD} \\ P_{BA} & P_{BB} & P_{BC} & P_{BD} \end{bmatrix} \begin{bmatrix} \sigma_A \\ \sigma_B \end{bmatrix} \begin{bmatrix} \tilde{f}_A \\ \tilde{f}_B \end{bmatrix}$$
$$\begin{bmatrix} \tilde{f}_A \\ \tilde{f}_B \end{bmatrix}$$
$$\begin{bmatrix} \tilde{f}_A \\ \tilde{f}_B \end{bmatrix} \begin{bmatrix} \sigma_B \\ \tilde{f}_B \end{bmatrix} \begin{bmatrix} \sigma_B \\ \tilde{f}_B \end{bmatrix} \begin{bmatrix} \sigma_B \\ \tilde{f}_B \end{bmatrix}$$





$$P_{i,j} = u_i S_{i,j} v_j^T$$

 u_i, v_j , now rank-k matrices

How to compute u_i, v_j ?

Option 1: Use analytical FMM expansions — matrix no longer HBS then, but \mathscr{H}^2

Option 2: Use numerical compression techniques, like SVD, ID

Low-rank approximations - Functional SVDs

Suppose
$$P\sigma = \sum_{j} K(x_i, y_j)\sigma_j$$
, $x_i \in B$ $y_j \in B_0 \setminus B$
 $Tf = \int_B K(x, y)f(y)dy$, $T : \mathbb{L}^2(B_0 \setminus B) \to \mathbb{L}^2(B)$ with

$$\int_{B_0 \setminus B} \int_B |K(x, y)|^2 \, dx \, dy < \infty$$

Then,
$$K(x, y) = \sum_{i=1}^{p} u_i(x) s_i v_i(y) + O(\varepsilon)$$
.

- Computing the functional SVD can be numerically intensive, particularly beyond d=2,3
- Costs can be amortized for translationally invariant kernels K(|x y|) and/or homogeneous kernels $K(\lambda x, \lambda y) = \lambda^r K(x, y)$
- Computational savings if kernel satisfies Green's identities (Proxy surfaces)
- FMM-like translation operators through SVDs for recompressing S

Low rank approximations - Randomized algorithms

$$\begin{bmatrix} P_{AA} & P_{AB} & P_{AC} & P_{AD} \\ P_{BA} & P_{BB} & P_{BC} & P_{BD} \\ P_{CA} & P_{CB} & P_{CC} & P_{CD} \\ P_{DA} & P_{DB} & P_{DC} & P_{DD} \end{bmatrix} \begin{bmatrix} \sigma_A \\ \sigma_B \\ \sigma_C \\ \sigma_D \end{bmatrix} = \begin{bmatrix} \tilde{f}_A \\ \tilde{f}_B \\ \tilde{f}_C \\ \tilde{f}_D \end{bmatrix}$$



$$P_B$$

$$\begin{bmatrix} P_{AB} & 0 & P_{CB} & P_{DB} \end{bmatrix} = U_B \begin{bmatrix} \tilde{V}_{BA}^T & 0 & \tilde{V}_{BC}^T & \tilde{V}_{BD} \end{bmatrix}$$

$$p \times N \qquad k \times N \qquad k \times p$$

Randomized algorithms:

$W = \mathbb{R}^{N \times (k+r)},$	random Gaussian matrix, FFT matrix					
$Y = P_B W$	$Y \in \mathbb{R}^{p \times (k+r)}$	Sample range of matrix				
Y = QR		Orthogonalize sampled range				
$T = Q^* P_B$	$T \in \mathbb{R}^{(k+r) \times p}$	Change of basis				
$T = \hat{U}SV^T$		SVD of reduced matrix				
$P_B \approx Q \hat{U} S V^T$						

Randomized algorithms - error analysis and performance

$$P_B$$

$$\begin{bmatrix} P_{AB} & 0 & P_{CB} & P_{DB} \end{bmatrix} = U_B \begin{bmatrix} \tilde{V}_{BA}^T & 0 & \tilde{V}_{BC}^T & \tilde{V}_{BD}^T \end{bmatrix}$$

$$p \times N \qquad k \times N \qquad k \times p$$

Randomized algorithms:

$$\begin{split} W &= \mathbb{R}^{N \times (k+r)}, \quad \text{random Gaussian matrix, FFT matrix} \\ Y &= P_B W \qquad Y \in \mathbb{R}^{p \times (k+r)} \qquad \text{Sample range of matrix} \qquad O(N \cdot (k+r) \cdot p) \\ Y &= Q R \qquad \qquad \text{Orthogonalize sampled range} \qquad O(p \cdot (k+r)^2) \\ T &= Q^* P_B \qquad T \in \mathbb{R}^{(k+r) \times N} \qquad \text{Change of basis} \qquad O(N \cdot (k+r) \cdot p) \\ T &= \hat{U} S V^T \qquad \qquad \text{SVD of reduced matrix} \qquad O((k+r)^2 \cdot N) \\ P_B &\approx Q \hat{U} S V^T \end{split}$$

 $\|P_B - Q\hat{U}SV^T\| = \|P_B - QT\| = \|P_B - QQ^*P_B\|$ $\|P_B - QQ^*P_B\| \le \left(1 + C\sqrt{N}\right)s_{k+1} \quad \text{with probability } 1 - 6r^{-r}$

Issues:

Cost of compressing one block of rows: $O(N \cdot (k + r) \cdot p)$ N such factorizations needed \implies cost of factorization: $O(N^2)$ Lack of interpretability of S at next layer

Interpolative Decomposition (ID)



Low rank approximation that uses columns of input matrix

$$\begin{aligned} |P_B - \tilde{P}_B Z|| &\leq (1 + \sqrt{k(n-k)})s_{k+1} \\ Z_{i,j}| &\leq 1 \end{aligned}$$

Combinatorial search, exponential cost

$$\begin{split} \|P_B - \tilde{P}_B Z\| &\leq (1 + \sqrt{k(n-k)})s_{k+1} \\ |Z_{i,j}| &\leq 2 \\ O(N \cdot p^2 \log N) \end{split}$$

In practice, rank revealing QR works fine



Randomized approach for computing ID

Interpolative Decomposition (ID)



Randomized approach for computing ID

Issues:

Cost of compressing one block of rows: $O(N \cdot (k + r) \cdot p)$

N such factorizations needed \implies cost of factorization: $O(N^2)$

-Lack of interpretability of S at next layer

Entries of S are sub-blocks of the original matrix



Works when matrix entries from Kernel satisfying Green's identity

General idea: identify smaller collection of columns which approximate bulk of matrix

Issues:

- Cost of compressing one block of rows: $O(N \cdot (k+r) \cdot p)$
- Cost of compressing one block of rows: $O((n_{proxy} + n_{near}) \cdot (k + r) \cdot p)$
- **N** such factorizations needed \implies cost of factorization: $O(N^2)$
- Lack of interpretability of S at next layer
- Entries of S are sub-blocks of the original matrix

The zoo of matrix factorizations



HODLR/HSS matrices

 FMM/\mathcal{H}^2 matrices



Low-rank structure

	$\stackrel{\rm Nested \ basis}{\longrightarrow}$						
	No	Yes					
Strong	HODLR	HSS					
Weak	\mathcal{H}	\mathcal{H}^2					

Butterfly/FFT matrices



A multiscale neural network based on hierarchical matrices

Yuwei Fan
* Lin Lin † Lexing Ying † Leonardo Zepeda-Núñez
§ Using ${\mathscr H}$ in layers of locally connected networks



Software

Lib

HLib Literature FAQs Patches Contact





https://github.com/klho/FLAM

		Declaration 1	Madam Hala		()								
Sarari	File Edit View Histo	bry Bookmarks	, <mark>P</mark> =	Not Secure — hilb org		••••••••••••••••••••••••••••••••••••••	5% <u></u> wed 5-32 PM	v ≔ N a					
Anil Dan	ale Adr	rianna Gillman	ChenhanYu/hmln: High-Perf	Hierarchical Matrices: HI ib	Software	https://apdiv.org/pdf/1403.6	siyaramambikasaran/HODI	ив. +		Safari File Edit	View History	Bookmarks Wind	ow He
										Anil Damle Adrianna Gil	ChenhanYu/		
	Hierarchic	al Matric	es									Branch: master - Ne	ew pull req
ИйЫ												oneilm name change	
	HLib Package	9										iii bin	ad
HLib	What is HLib?											ill doc	cle
<u>FAQs</u>	ut the is a program libr	ary for hierarchical	matrices and H ² -matrices. It pr	ovidee								include	na
Patches Contact	HEIB IS a program nor	ary for merarcinear	matrices and <i>H</i> -matrices. It pro	ovides								STC	re
	 routines for the discretization fu 	inctions that fill thes	e structures by approximations	of FEM or BEM operators,	w-rank matrices and block	k matrices),						in tmp	ad
	 arithmetic algor conversion routi 	ithms that perform a	pproximative matrix operation matrices and dense matrices in	s like addition, multiplication, factor to H-matrices and H-matrices or H ² .	zations and inversion, matrices into H^2 -matrices	and							Ad
	 service function 	is that display matrix	structures, perform numerical	quadrature or handle files.	marices into 17 - marices	and						Makefile	sm
	Requirements											README.md	rea
	The library is written	in the c programs	ning language and requires pr	and tabler to perform lower l	wal algebraic operations	(lika parforming dansa matri	x matrix multiplications or	or colving				size.txt	ed
	eigenvalue problems).	i ili ule e programi		and LAPACE to perform lower-	ver algebraic operations	(like performing dense maur	A-matrix multiplications of	n solving				I README.md	
	On Unix-type systems If GTK+, OpenGL an working on a compret	a, we use Autoconf, d GTKGLArea are nensive GUI that wil	Automake and Libtool in order installed, a visualization progra l offer the more interesting fun-	to provide the user with a fully auto am for surface meshes and cluster t ctions of the library.	es is provided that can b	e quite useful for debugging	BEM applications. We are	currently				Interpo	lativ
	Licensing											This is a fork o	of the ID
	Since HLib was devel interested in doing res	loped (at least parti earch if they sign ar	ally) at the Max-Planck-Institu appropriate license agreement	tt Leipzig, we cannot use an open	ource license and turn it	into free software. But we ca	an give it away for free to	o scientist				 P. Martins via interpo 	son, V. R plative de
	Staffan Börm and Lar	e Grasadvek										Original dis	tributio
	Max-Planck-Institut f	<u>ur Mathematik in de</u>	n Naturwissenschaften	?								Please see the	docume
	Inseistrasse 22-20, 04	103 Leipzig, Germa	ny									At the minimur	m, pleas
												source code c well").	omment
									pr	es Extract		🔊 🌍 🄇	
pres Extract		2 🔇) 🖉 🔟 🜔 [

https://github.com/sivaramambikasaran/HODLR

	···· ¹² =							۵
Anil Damle Adrianna Gil ChenhanYu/	. Hierarchical	Software https://arxiv	sivaramamb	kiho/FLAM:	Fast Algorit	https://www	Mark Tygert	fastalgorith
	Branch: master - New pu	ll request	Create ne	w file Upload files	Find File Clor	e or download +		
	oneilm name change				Latest commit 4	455d51 on Feb 4		
	iii bin	added bin				3 months ago		
	in doc	cleaning up the repo				3 months ago		
	include	name change				3 months ago		
	in src	removed more save statements, and	orin statements			3 months ago		
	ille test	initial commit				6 months ago		
	in tmp	added tmp				3 months ago		
	LICENSE	Add LICENSE				3 months ago		
	Makefile	small edit makefile				3 months ago		
	README.md	readme				3 months ago		
	size.txt	editined				3 months ago		
	E README.md					1		
	Interpolat This is a fork of the • P. Martinsson, via interpolativ Original distrib Please see the door	ive Decomposition ID library V. Rokhlin, Y. Shkolnisky, M. Tygert e decompositions, http://tygert.com ution readme umentation in subdirectory doc of	, ID: a software packag n/software.html	e for low-rank app	roximation of ma	atrices		
	At the minimum, p	ease read Subsection 2.1 and Section	on 3 in the documental	tion, and beware th N.B. stands for no	nat the N.B.'s in 1 Inta bene (Latin fe	the or "note		

https://github.com/fastalgorithms/libid

http://www.hlib.org

More resources

			Safari File Edit View	History Bookmarks Window Help	6-0 🚥 🐺 🖶 🗛 🗔 💭 🗐 🛜 33% 🗔 Thu 6:26
🗯 Safari File Edit View His	story Bookmarks Window Help	6Ə 🚥 👯 🖆 🙆 😱 🥅 🎅 34% 🗊 🛛 Thu 6:21 AM 🔍 😑		••• ¹² =	ei github.com 🖒
	🚥 🎴 🗐 🔒 github.com	Ċ Ó	http https://anxiv	gunnar mart WhatsApp Portland Tra interpolativ	https://www https://web Yuwei Fan victorminde Finding the i
http https://arxiv gunna	ar mart. WhatsApp Portland Tra interpolativ https://www https://www // Pull requests issues Marketplace Explore	Yuwel Fan victominde Finding the L Introduction +	Search or jump to	Pull requests issues Marketplace Explore	
	ChenhanYu / hmlp O Code Code Table 19 Pull requests (a) Pull Projects (s) Bit Wild Insights	- 8 Xtar 6 Y Fork 6		Y victorminden / GPMLE teried from asdank/GPMLE ↔ Code □ Pull requests @ □ Projects @ □ Wiki	Ø Watch • 1 ¥ Star 1 ¥ Fork 4 Insights
	Introduction to GOFMM Chenhan D. Yu edited this page on Jul 20, 2017 - 7 revisions			Code implementing the methodology in arXiv:1603:08057 fo from a kernelized Gaussian process in two spatial dimension	r maximum likelihood estimation for parameter-fitting given observations s.
	GOFMM stands for Geometry-Oblivious Fast Multipole Method, which is a novel method that creates a hierarchical low-rank approximation, or "compression," of an arbitrary dense symmetric positive denite (SPD) matrix. For many applications. GOFMM enables an approximate	▼ Pages () Find a Page		G 4 commits J 1 branch Branch: master • New pull request	© O releases ▲ 2 contributors d₂ MIT Create new file Upload files Find File Clone or download -
	matrix-vector multiplication (MATVEC) in Q(NiogN) or even Q(N) time, where N is the matrix size. Compression requires N log N storage and work. In general, our scheme belongs to the family of hierarchical matrix approximation methods. In particular, it generalizes the fast	Home Introduction to GKMX		This branch is 1 commit behind asdamle:master.	Pull request ② Compare Latest commit 5e29912 on Apr 19, 2016
	multipole method (FMM) to a purely algebraic setting by only requiring the ability to sample matrix entries. Neither geometric information (6.e., point coordinates) on knowledge of how the matrix entries have been generated is required, thus the term "geometry-oblivious."	Introduction to GOFMM Introduction to GSKNN		in ex typo in kernels cleaned up	3 years ago 3 years ago
	Compress (Constructing an H-Matrix)	Introduction to GSKS Introduction to STRASSEN		opt_rskelf commit peel commit	3 years ago 3 years ago
	Evaluate (Matrix-Vector Multiplication)	Introduction to TREE Microkernels		verify_error commit	3 years ago SE.md 3 years ago
	Factorize (H-Matrix Factorization for Fast Solvers)	RUNTIME and COMMUNICATOR		README.md commit	3 years ago
	Solve (Solve a Shifted Linear System)	Use Strasen Clone this wiki locally		E startup.m commit	3 years ago
	HMLP is currently not an open source project. Do not distribute!!!	https://github.com/Chenha 😫		III README.md	
				README	
pres Extract	🛂 🚷 🧭 🧭 🔁 📒 📁 🔛 🔍 🐣 🎯		pres Extract	N 🕄 🗞 🐼 🏈 🗾 🚺 🚺 🗎] 🔜 🔍 🔇 💽 Ţ 🗐 🚫 🦉
https://gi	thub.com/ChenhanYu/ ion-to-GOFMM	/hmlp/wiki/	https://	/github.com/vic	torminden/GPMLE

- Video lectures by Gunnar https://www.youtube.com/playlist? list=PLPDZ9rclfxyOrlpcu_D1PRcyK-o2iofwW
- Excellent review article on randomized methods for low rank approximations Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions: <u>https://arxiv.org/pdf/0909.4061.pdf</u>
- Some of the illustrations courtesy: Sivaram Ambikasaran, Per-Gunnar Martinsson, Ken Ho, Lesliie Greengard, Lexing Ying, Adrianna Gillman

References

- Ho, Kenneth L., and Leslie Greengard. "A fast direct solver for structured linear systems by recursive skeletonization." *SIAM Journal on Scientific Computing* 34.5 (2012): A2507-A2532.
- Yu, Chenhan D., et al. "Geometry-oblivious FMM for compressing dense SPD matrices." *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis.* ACM, 2017.
- Ambikasaran, S., Foreman-Mackey, D., Greengard, L., Hogg, D. W., & O'Neil, M. (2016). Fast direct methods for Gaussian processes. *IEEE transactions on pattern analysis and machine intelligence*, *38*(2), 252-265.
- Gillman, A., Young, P. M., & Martinsson, P. G. (2012). A direct solver with O (N) complexity for integral equations on one-dimensional domains. Frontiers of Mathematics in China, 7(2), 217-247.
- Halko, N., Martinsson, P. G., & Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, *53*(2), 217-288.
- Greengard, L., & Rokhlin, V. (1991). On the numerical solution of two-point boundary value problems. Communications on Pure and Applied Mathematics, 44(4), 419-452.
- Fan, Y., Lin, L., Ying, L., & Zepeda-Núnez, L. (2018). A multiscale neural network based on hierarchical matrices. arXiv preprint arXiv:1807.01883.
- Minden, V., Damle, A., Ho, K. L., & Ying, L. (2017). Fast spatial gaussian process maximum likelihood estimation via skeletonization factorizations. *Multiscale Modeling & Simulation*, 15(4), 1584-1611.
- Martinsson, P. G., Rokhlin, V., Shkolnisky, Y., & Tygert, M. (2008). ID: A software package for low-rank approximation of matrices via interpolative decompositions, Version 0.2.
- Corona, E., Martinsson, P. G., & Zorin, D. (2015). An O (N) direct solver for integral equations on the plane. *Applied and Computational Harmonic Analysis*, 38(2), 284-317.
- Gimbutas, Z., & Rokhlin, V. (2003). A generalized fast multipole method for nonoscillatory kernels. SIAM Journal on Scientific Computing, 24(3), 796-817.

Not an exhaustive list