Efficient Tensor-based Approximations to Kernel Interactions

Rachel Minster[1], Arvind K. Saibaba[1], Misha E. Kilmer[2]

[1]North Carolina State University
[2]Tufts University

5/21/21

Acknowledgements to NSF DMS 1821149 and DMS 1745654 for funding

Motivation: Kernel Methods

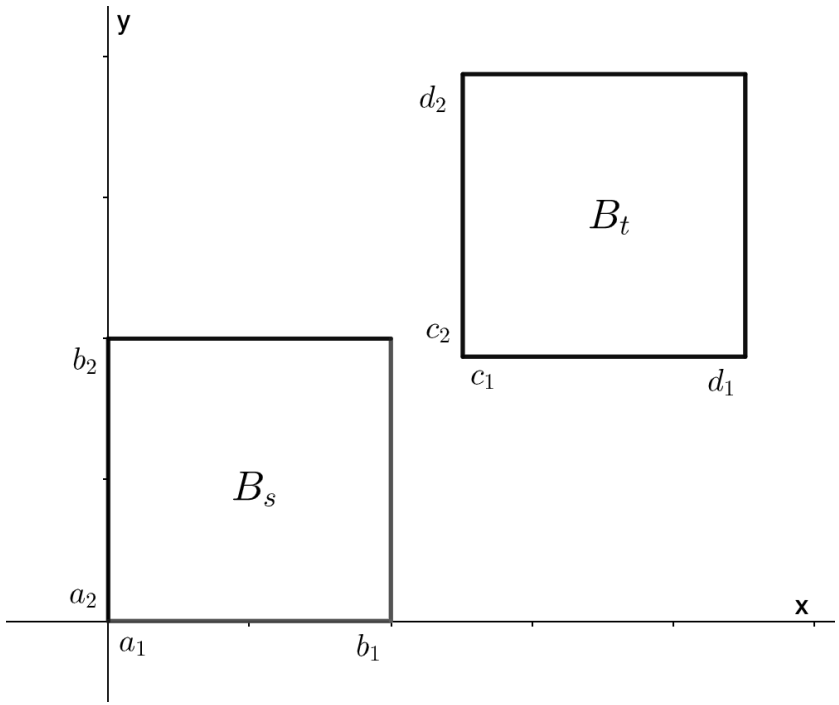
- Model pairwise interactions between sets of points defined by kernel κ
- Applications where kernel methods are used:
  - Integral equations (Green's Function), n-body problems, Gaussian processes
- Major challenges include
  - Number of interaction points often large
  - Kernel matrices are dense, difficult to store and compute with

Motivation: Kernel Methods

- Model pairwise interactions between sets of points defined by kernel κ
- Applications where kernel methods are used:
  - Integral equations (Green's Function), n-body problems, Gaussian processes
- Major challenges include
  - Number of interaction points often large
  - Kernel matrices are dense, difficult to store and compute with
- General approach: store kernel matrix efficiently as a rank-structured matrix in a hierarchical form
  - Forms include $\mathcal{H}$-matrices, $\mathcal{H}^2$-matrices, Hierarchical Semiseparable (HSS) matrices, Hierarchical Off-Diagonal Low Rank (HODLR) matrices
  - Constructed by recursively identifying and compressing off-diagonal blocks in low-rank form

Problem Setup

- N_s source points $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_s}\}$ in $\mathcal{B}_s = [a_1, b_1] \times \cdots \times [a_D, b_D]$
- N_t target points $\boldsymbol{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{N_t}\}$ in $\mathcal{B}_t = [c_1, d_1] \times \cdots \times [c_D, d_D]$
- kernel $\kappa : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$



Problem Setup

- N_s source points $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_s}\}$ in $\mathcal{B}_s = [a_1, b_1] \times \cdots \times [a_D, b_D]$
- N_t target points $\boldsymbol{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{N_t}\}$ in $\mathcal{B}_t = [c_1, d_1] \times \cdots \times [c_D, d_D]$
- kernel $\kappa : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$

Define interaction matrix

$$\mathcal{K}(\boldsymbol{X}, \boldsymbol{Y}) = \begin{bmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{y}_1) & \kappa(\boldsymbol{x}_1, \boldsymbol{y}_2) & \cdots & \kappa(\boldsymbol{x}_1, \boldsymbol{y}_{N_t}) \\ \kappa(\boldsymbol{x}_2, \boldsymbol{y}_1) & \kappa(\boldsymbol{x}_2, \boldsymbol{y}_2) & \cdots & \kappa(\boldsymbol{x}_2, \boldsymbol{y}_{N_t}) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa(\boldsymbol{x}_{N_s}, \boldsymbol{y}_1) & \kappa(\boldsymbol{x}_{N_s}, \boldsymbol{y}_2) & \cdots & \kappa(\boldsymbol{x}_{N_s}, \boldsymbol{y}_{N_t}) \end{bmatrix}$$

Computing requires $N_s N_t$ interactions, compressing with an SVD would be very expensive

Problem Setup

- N_s source points $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_s}\}$ in $\mathcal{B}_s = [a_1, b_1] \times \cdots \times [a_D, b_D]$
- N_t target points $\boldsymbol{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{N_t}\}$ in $\mathcal{B}_t = [c_1, d_1] \times \cdots \times [c_D, d_D]$
- kernel $\kappa : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$

Define interaction matrix

$$\mathcal{K}(\boldsymbol{X}, \boldsymbol{Y}) = \begin{bmatrix} \kappa(\boldsymbol{x}_1, \boldsymbol{y}_1) & \kappa(\boldsymbol{x}_1, \boldsymbol{y}_2) & \cdots & \kappa(\boldsymbol{x}_1, \boldsymbol{y}_{N_t}) \\ \kappa(\boldsymbol{x}_2, \boldsymbol{y}_1) & \kappa(\boldsymbol{x}_2, \boldsymbol{y}_2) & \cdots & \kappa(\boldsymbol{x}_2, \boldsymbol{y}_{N_t}) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa(\boldsymbol{x}_{N_s}, \boldsymbol{y}_1) & \kappa(\boldsymbol{x}_{N_s}, \boldsymbol{y}_2) & \cdots & \kappa(\boldsymbol{x}_{N_s}, \boldsymbol{y}_{N_t}) \end{bmatrix}$$

Computing requires $N_s N_t$ interactions, compressing with an SVD would be very expensive

Main idea: approximate κ using Chebyshev interpolation and tensor compression methods

Multivariate Chebyshev Interpolation

For two spatial dimensions ($D = 2$):

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = f(x_1, x_2, y_1, y_2) = f(\xi_1, \xi_2, \xi_3, \xi_4)$$
$$\approx \sum_{j_1=1}^{n} \cdots \sum_{j_4=1}^{n} f(\eta_{j_1}^{(1)}, \eta_{j_2}^{(2)}, \eta_{j_3}^{(3)}, \eta_{j_4}^{(4)}) \left(\prod_{k=1}^{4} S_n^{[\alpha_k, \beta_k]}(\eta_{j_k}^{(k)}, \xi_k) \right)$$

with

- $\xi_j = x_j$, $\xi_{j+2} = y_j$
- $\eta_{j_k}^{(k)}$ Chebyshev points in interval $[\alpha_k, \beta_k]$
  - $\alpha_j = a_j, \beta_j = b_j, \alpha_{j+2} = c_j, \beta_{j+2} = d_j$
- n number of Chebyshev points per dimension
- $S_n^{[a,b]}$ Chebyshev polynomial interpolant

Multivariate Chebyshev Interpolation

For two spatial dimensions ($D = 2$):

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = f(x_1, x_2, y_1, y_2) = f(\xi_1, \xi_2, \xi_3, \xi_4)$$
$$\approx \sum_{j_1=1}^{n} \cdots \sum_{j_4=1}^{n} f(\eta_{j_1}^{(1)}, \eta_{j_2}^{(2)}, \eta_{j_3}^{(3)}, \eta_{j_4}^{(4)}) \left(\prod_{k=1}^{4} S_n^{[\alpha_k, \beta_k]}(\eta_{j_k}^{(k)}, \xi_k) \right)$$

with

- $\xi_j = x_j$, $\xi_{j+2} = y_j$
- $\eta_{j_k}^{(k)}$ Chebyshev points in interval $[\alpha_k, \beta_k]$
  - $\alpha_j = a_j, \beta_j = b_j, \alpha_{j+2} = c_j, \beta_{j+2} = d_j$
- n number of Chebyshev points per dimension
- $S_n^{[a,b]}$ Chebyshev polynomial interpolant

Chebyshev Interpolation as a Tensor

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = f(x_1, x_2, y_1, y_2) = f(\xi_1, \xi_2, \xi_3, \xi_4)$$
$$\approx \sum_{j_1=1}^{n} \cdots \sum_{j_4=1}^{n} f(\eta_{j_1}^{(1)}, \eta_{j_2}^{(2)}, \eta_{j_3}^{(3)}, \eta_{j_4}^{(4)}) \left(\prod_{k=1}^{4} S_n^{[\alpha_k, \beta_k]}(\eta_{j_k}^{(k)}, \xi_k) \right)$$

Can be written in tensor form:

$$f(\xi_1, \xi_2, \xi_3, \xi_4) \approx \mathcal{M} \bigtimes_{k=1}^{4} \boldsymbol{s}_k(\xi_k)$$

where

- $\mathcal{M}_{j_1, j_2, j_3, j_4} = f(\eta_{j_1}^{(1)}, \eta_{j_2}^{(2)}, \eta_{j_3}^{(3)}, \eta_{j_4}^{(4)})$, with $\mathcal{M} \in \mathbb{R}^{n \times n \times n \times n}$
- $\boldsymbol{s}_k(\xi_k) = \begin{bmatrix} S_n^{[\alpha_k, \beta_k]}(\eta_1^{(k)}, \xi_k) & \cdots & S_n^{[\alpha_k, \beta_k]}(\eta_n^{(k)}, \xi_k) \end{bmatrix} \in \mathbb{R}^{1 \times n}$

Our Tensor-based Compression Approach

Idea: Use tensor compression methods on $\mathcal{M}$ to obtain Tucker approximation $\widehat{\mathcal{M}} = [\mathcal{G}; A_1, A_2, A_3, A_4]$

Three new randomized methods for compressing $\mathcal{M}$

- Method 1: uses row interpolatory decomposition to approximate mode unfoldings
- Method 2: Method 1 but with a subsampled tensor
- Method 3: uses a Kronecker product of random Gaussian matrices instead of a single Gaussian matrix

Our Tensor-based Compression Approach

Idea: Use tensor compression methods on $\mathcal{M}$ to obtain Tucker approximation $\widehat{\mathcal{M}} = [\mathcal{G}; A_1, A_2, A_3, A_4]$

Three new randomized methods for compressing $\mathcal{M}$

- **Method 1**: uses row interpolatory decomposition to approximate mode unfoldings
- **Method 2**: Method 1 but with a subsampled tensor
- Method 3: uses a Kronecker product of random Gaussian matrices instead of a single Gaussian matrix

Our Tensor-based Compression Approach

Idea: Use tensor compression methods on $\mathcal{M}$ to obtain Tucker approximation $\widehat{\mathcal{M}} = [\mathcal{G}; A_1, A_2, A_3, A_4]$

Three new randomized methods for compressing $\mathcal{M}$

- **Method 1**: uses row interpolatory decomposition to approximate mode unfoldings
- **Method 2**: Method 1 but with a subsampled tensor
- Method 3: uses a Kronecker product of random Gaussian matrices instead of a single Gaussian matrix

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = f(\xi_1, \xi_2, \xi_3, \xi_4) \approx \widehat{\mathcal{M}} \bigtimes_{k=1}^{4} \boldsymbol{s}_k(\xi_k)$$

$$= \mathcal{G} \bigtimes_{i=1}^{4} A_i \bigtimes_{k=1}^{4} \boldsymbol{s}_k(\xi_k) = \mathcal{G} \bigtimes_{k=1}^{4} \boldsymbol{s}_k(\xi_k) A_k$$

Low-rank Approximation to Kernel Matrix

For a single pair of points,

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \mathcal{G} \bigtimes_{k=1}^{4} \boldsymbol{s}_k(\xi_k) A_k = \mathcal{G} \bigtimes_{k=1}^{4} \widehat{\boldsymbol{s}}_k(\xi_k)$$

$$= \left(\widehat{\boldsymbol{s}}_2(\xi_2) \otimes \widehat{\boldsymbol{s}}_1(\xi_1)\right) G_{(1:2)} \left(\widehat{\boldsymbol{s}}_4^{\top}(\xi_4) \otimes \widehat{\boldsymbol{s}}_3^{\top}(\xi_3)\right)$$

Low-rank Approximation to Kernel Matrix

For a single pair of points,

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \mathcal{G} \bigtimes_{k=1}^{4} \boldsymbol{s}_k(\xi_k) A_k = \mathcal{G} \bigtimes_{k=1}^{4} \widehat{\boldsymbol{s}}_k(\xi_k)$$
$$= \left(\widehat{\boldsymbol{s}}_2(\xi_2) \otimes \widehat{\boldsymbol{s}}_1(\xi_1)\right) G_{(1:2)} \left(\widehat{\boldsymbol{s}}_4^\top(\xi_4) \otimes \widehat{\boldsymbol{s}}_3^\top(\xi_3)\right)$$

Collect $\widehat{\boldsymbol{s}}$ matrices for all points:

- $U_j = \begin{bmatrix} \widehat{\boldsymbol{s}}_j(x_1) & \dots & \widehat{\boldsymbol{s}}_j(x_{N_s}) \end{bmatrix}^\top \in \mathbb{R}^{N_s \times r}, \quad j = 1, 2$
- $V_j = \begin{bmatrix} \widehat{\boldsymbol{s}}_j(y_1) & \dots & \widehat{\boldsymbol{s}}_j(y_{N_t}) \end{bmatrix}^\top \in \mathbb{R}^{N_t \times r}, \quad j = 3, 4$

Low-rank Approximation to Kernel Matrix

For a single pair of points,

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \mathcal{G} \bigtimes_{k=1}^{4} \boldsymbol{s}_k(\xi_k) A_k = \mathcal{G} \bigtimes_{k=1}^{4} \widehat{\boldsymbol{s}}_k(\xi_k)$$
$$= \left(\widehat{\boldsymbol{s}}_2(\xi_2) \otimes \widehat{\boldsymbol{s}}_1(\xi_1)\right) G_{(1:2)} \left(\widehat{\boldsymbol{s}}_4^\top(\xi_4) \otimes \widehat{\boldsymbol{s}}_3^\top(\xi_3)\right)$$

Collect $\widehat{\boldsymbol{s}}$ matrices for all points:

- $U_j = \begin{bmatrix} \widehat{\boldsymbol{s}}_j(x_1) & \dots & \widehat{\boldsymbol{s}}_j(x_{N_s}) \end{bmatrix}^\top \in \mathbb{R}^{N_s \times r}, \quad j = 1, 2$
- $V_j = \begin{bmatrix} \widehat{\boldsymbol{s}}_j(y_1) & \dots & \widehat{\boldsymbol{s}}_j(y_{N_t}) \end{bmatrix}^\top \in \mathbb{R}^{N_t \times r}, \quad j = 3, 4$

Then interaction matrix approximation is

$$\mathcal{K}(\boldsymbol{X}, \boldsymbol{Y}) \approx (U_2 \ltimes U_1) G_{(1:2)} (V_4 \ltimes V_3)^\top$$

Randomized Row Interpolatory Decomposition (RRID)[1]

Gives low-rank approximation using randomized range finder and subset selection

[1] Halko, Martinsson, Tropp, SIAM Review, 2011

Randomized Row Interpolatory Decomposition (RRID)[1]

Gives low-rank approximation using randomized range finder and subset selection

Main Steps

For a matrix $X \in \mathbb{R}^{m \times n}$,
target rank r, oversampling
parameter $p > 0$ such that
$r + p < m$,

1. Estimate range of X
2. Subset selection to identify indices $\mathcal{J}$
3. Compute approximation matrix F so $X \approx FX(\mathcal{J}, :)$

[1] Halko, Martinsson, Tropp, SIAM Review, 2011

Randomized Row Interpolatory Decomposition (RRID)[1]

Gives low-rank approximation using randomized range finder and subset selection

Main Steps

For a matrix $X \in \mathbb{R}^{m \times n}$, target rank r, oversampling parameter $p > 0$ such that $r + p < m$,

1. **Estimate range of X**
2. Subset selection to identify indices $\mathcal{J}$
3. Compute approximation matrix F so $X \approx FX(\mathcal{J}, :)$

Details

- Draw $\Omega \in \mathbb{R}^{n \times (r+p)}$ a Gaussian random matrix
- Form product $Y = X\Omega$
- Compute thin QR $Y = QR$
  - $\mathcal{R}(Q) \approx \mathcal{R}(X)$
- Use pivoted QR on $Q^\top$
  - Gives indices $\mathcal{J}$ of well-conditioned rows of Q
- Compute $F = Q\left(Q(\mathcal{J}, :)\right)^{-1}$

[1] Halko, Martinsson, Tropp, SIAM Review, 2011

Randomized Row Interpolatory Decomposition (RRID)[1]

Gives low-rank approximation using randomized range finder and subset selection

Main Steps

For a matrix $X \in \mathbb{R}^{m \times n}$, target rank r, oversampling parameter $p > 0$ such that $r + p < m$,

1. Estimate range of X
2. **Subset selection to identify indices $\mathcal{J}$**
3. Compute approximation matrix F so $X \approx FX(\mathcal{J}, :)$

Details

- Draw $\Omega \in \mathbb{R}^{n \times (r+p)}$ a Gaussian random matrix
- Form product $Y = X\Omega$
- Compute thin QR $Y = QR$
  - $\mathcal{R}(Q) \approx \mathcal{R}(X)$
- Use pivoted QR on $Q^\top$
  - Gives indices $\mathcal{J}$ of well-conditioned rows of Q
- Compute $F = Q\left(Q(\mathcal{J}, :)\right)^{-1}$

[1] Halko, Martinsson, Tropp, SIAM Review, 2011

Randomized Row Interpolatory Decomposition (RRID)[1]

Gives low-rank approximation using randomized range finder and subset selection

Main Steps

For a matrix $X \in \mathbb{R}^{m \times n}$, target rank r, oversampling parameter $p > 0$ such that $r + p < m$,

1. Estimate range of X
2. Subset selection to identify indices $\mathcal{J}$
3. **Compute approximation matrix F so $X \approx FX(\mathcal{J}, :)$**

Details

- Draw $\Omega \in \mathbb{R}^{n \times (r+p)}$ a Gaussian random matrix
- Form product $Y = X\Omega$
- Compute thin QR $Y = QR$
  - $\mathcal{R}(Q) \approx \mathcal{R}(X)$
- Use pivoted QR on $Q^\top$
  - Gives indices $\mathcal{J}$ of well-conditioned rows of Q
- Compute $F = Q\left(Q(\mathcal{J}, :)\right)^{-1}$

[1] Halko, Martinsson, Tropp, SIAM Review, 2011

Method 1: Randomized Interpolatory Tensor Decomposition

Compresses tensor $\mathcal{M} \in \mathbb{R}^{n \times n \times n \times n}$

- Easily extendable to d-dimensional tensors

Process:

- For mode 1:
  1. Unfold $\mathcal{M} \rightarrow M_{(1)}$
  2. Apply RRID with target rank r to $M_{(1)} \approx A_1 M_{(1)}(\mathcal{J}_1, :)$
- Repeat for modes 2-4 to obtain matrices A_2, A_3, A_4 and index sets $\mathcal{J}_2, \mathcal{J}_3, \mathcal{J}_4$
- Compute core $\mathcal{G} = \mathcal{M}(\mathcal{J}_1, \mathcal{J}_2, \mathcal{J}_3, \mathcal{J}_4)$
- Gives approximation $\widehat{\mathcal{M}} = \mathcal{G} \times_1 A_1 \times_2 A_2 \times_3 A_3 \times_4 A_4$

Method 1: Randomized Interpolatory Tensor Decomposition

Compresses tensor $\mathcal{M} \in \mathbb{R}^{n \times n \times n \times n}$

- Easily extendable to d-dimensional tensors

Process:

- For mode 1:
  1. Unfold $\mathcal{M} \rightarrow M_{(1)}$
  2. Apply RRID with target rank r to $M_{(1)} \approx A_1 M_{(1)}(\mathcal{J}_1, :)$
- Repeat for modes 2-4 to obtain matrices A_2, A_3, A_4 and index sets $\mathcal{J}_2, \mathcal{J}_3, \mathcal{J}_4$
- Compute core $\mathcal{G} = \mathcal{M}(\mathcal{J}_1, \mathcal{J}_2, \mathcal{J}_3, \mathcal{J}_4)$
- Gives approximation $\widehat{\mathcal{M}} = \mathcal{G} \times_1 A_1 \times_2 A_2 \times_3 A_3 \times_4 A_4$

Method 1: Randomized Interpolatory Tensor Decomposition

Compresses tensor $\mathcal{M} \in \mathbb{R}^{n \times n \times n \times n}$

- Easily extendable to d-dimensional tensors

Process:

- For mode 1:
  1. Unfold $\mathcal{M} \rightarrow M_{(1)}$
  2. Apply RRID with target rank r to $M_{(1)} \approx A_1 M_{(1)}(\mathcal{J}_1, :)$
- Repeat for modes 2-4 to obtain matrices A_2, A_3, A_4 and index sets $\mathcal{J}_2, \mathcal{J}_3, \mathcal{J}_4$
- Compute core $\mathcal{G} = \mathcal{M}(\mathcal{J}_1, \mathcal{J}_2, \mathcal{J}_3, \mathcal{J}_4)$
- Gives approximation $\widehat{\mathcal{M}} = \mathcal{G} \times_1 A_1 \times_2 A_2 \times_3 A_3 \times_4 A_4$

Method 1: Randomized Interpolatory Tensor Decomposition

Compresses tensor $\mathcal{M} \in \mathbb{R}^{n \times n \times n \times n}$

- Easily extendable to d-dimensional tensors

Process:

- For mode 1:
  1. Unfold $\mathcal{M} \rightarrow M_{(1)}$
  2. Apply RRID with target rank r to $M_{(1)} \approx A_1 M_{(1)}(\mathcal{J}_1, :)$
- Repeat for modes 2-4 to obtain matrices A_2, A_3, A_4 and index sets $\mathcal{J}_2, \mathcal{J}_3, \mathcal{J}_4$
- Compute core $\mathcal{G} = \mathcal{M}(\mathcal{J}_1, \mathcal{J}_2, \mathcal{J}_3, \mathcal{J}_4)$
- Gives approximation $\widehat{\mathcal{M}} = \mathcal{G} \times_1 A_1 \times_2 A_2 \times_3 A_3 \times_4 A_4$

Method 2: Randomized Interpolatory Tensor Decomp. with Block Selection

Similar to Method 1, but working with subsampled tensor instead of $\mathcal{M}$ to decrease computational cost

Process:

- For mode 1:
  1. Sample tensor: form index set $\mathcal{I}$ of size b, and extract subsampled tensor $\mathcal{X} = \mathcal{M}(:, \mathcal{I}, \mathcal{I}, \mathcal{I})$
  2. Unfold $\mathcal{X} \to X_{(1)}$
  3. Apply RRID with target rank r to $X_{(1)} \approx A_1 X_{(1)}(\mathcal{J}_1, :)$
- Repeat for modes 2-4 to obtain matrices A_2, A_3, A_4 and index sets $\mathcal{J}_2, \mathcal{J}_3, \mathcal{J}_4$
- Compute core $\mathcal{G} = \mathcal{M}(\mathcal{J}_1, \mathcal{J}_2, \mathcal{J}_3, \mathcal{J}_4)$
- Gives approximation $\widehat{\mathcal{M}} = \mathcal{G} \times_1 A_1 \times_2 A_2 \times_3 A_3 \times_4 A_4$

Method 2: Randomized Interpolatory Tensor Decomp. with Block Selection

Similar to Method 1, but working with subsampled tensor instead of $\mathcal{M}$ to decrease computational cost

Process:

- For mode 1:
  1. Sample tensor: form index set $\mathcal{I}$ of size b, and extract subsampled tensor $\mathcal{X} = \mathcal{M}(:, \mathcal{I}, \mathcal{I}, \mathcal{I})$
  2. Unfold $\mathcal{X} \to X_{(1)}$
  3. Apply RRID with target rank r to $X_{(1)} \approx A_1 X_{(1)}(\mathcal{J}_1, :)$
- Repeat for modes 2-4 to obtain matrices A_2, A_3, A_4 and index sets $\mathcal{J}_2, \mathcal{J}_3, \mathcal{J}_4$
- Compute core $\mathcal{G} = \mathcal{M}(\mathcal{J}_1, \mathcal{J}_2, \mathcal{J}_3, \mathcal{J}_4)$
- Gives approximation $\widehat{\mathcal{M}} = \mathcal{G} \times_1 A_1 \times_2 A_2 \times_3 A_3 \times_4 A_4$

Method 2: Randomized Interpolatory Tensor Decomp. with Block Selection

Similar to Method 1, but working with subsampled tensor instead of $\mathcal{M}$ to decrease computational cost

Process:

- For mode 1:
  1. Sample tensor: form index set $\mathcal{I}$ of size b, and extract subsampled tensor $\mathcal{X} = \mathcal{M}(:, \mathcal{I}, \mathcal{I}, \mathcal{I})$
  2. Unfold $\mathcal{X} \to X_{(1)}$
  3. Apply RRID with target rank r to $X_{(1)} \approx A_1 X_{(1)}(\mathcal{J}_1, :)$
- Repeat for modes 2-4 to obtain matrices A_2, A_3, A_4 and index sets $\mathcal{J}_2, \mathcal{J}_3, \mathcal{J}_4$
- Compute core $\mathcal{G} = \mathcal{M}(\mathcal{J}_1, \mathcal{J}_2, \mathcal{J}_3, \mathcal{J}_4)$
- Gives approximation $\widehat{\mathcal{M}} = \mathcal{G} \times_1 A_1 \times_2 A_2 \times_3 A_3 \times_4 A_4$

Method 2: Randomized Interpolatory Tensor Decomp. with Block Selection

Similar to Method 1, but working with subsampled tensor instead of $\mathcal{M}$ to decrease computational cost

Process:

- For mode 1:
  1. Sample tensor: form index set $\mathcal{I}$ of size b, and extract subsampled tensor $\mathcal{X} = \mathcal{M}(:, \mathcal{I}, \mathcal{I}, \mathcal{I})$
  2. Unfold $\mathcal{X} \to X_{(1)}$
  3. Apply RRID with target rank r to $X_{(1)} \approx A_1 X_{(1)}(\mathcal{J}_1, :)$
- Repeat for modes 2-4 to obtain matrices A_2, A_3, A_4 and index sets $\mathcal{J}_2, \mathcal{J}_3, \mathcal{J}_4$
- Compute core $\mathcal{G} = \mathcal{M}(\mathcal{J}_1, \mathcal{J}_2, \mathcal{J}_3, \mathcal{J}_4)$
- Gives approximation $\widehat{\mathcal{M}} = \mathcal{G} \times_1 A_1 \times_2 A_2 \times_3 A_3 \times_4 A_4$

Computational Cost

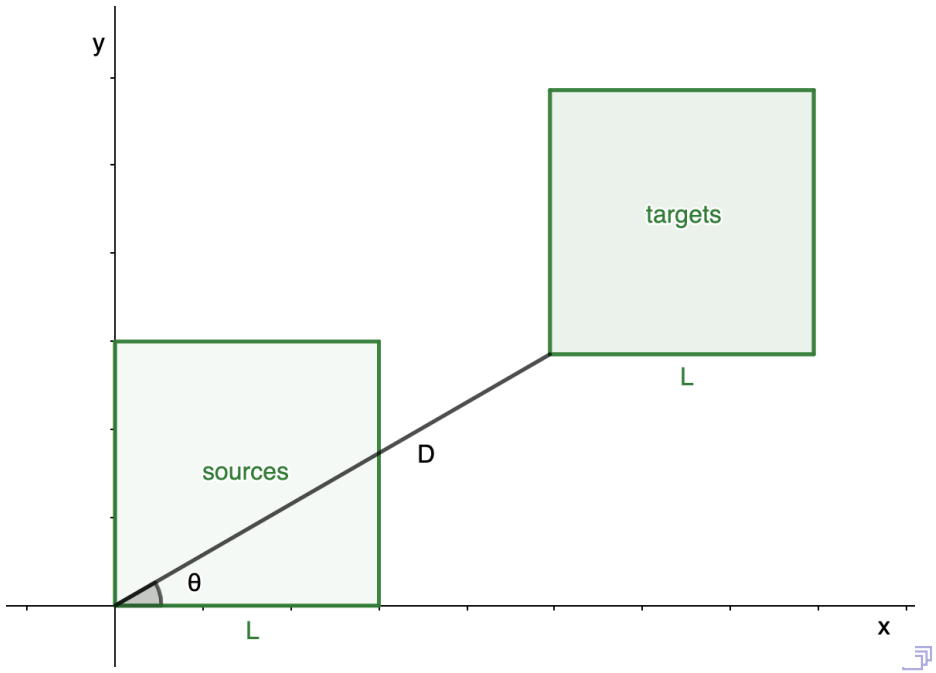
Parameters:

- n: number of Chebyshev nodes
- r: target rank
- p: oversampling parameter
- b: size of index set for Method 2

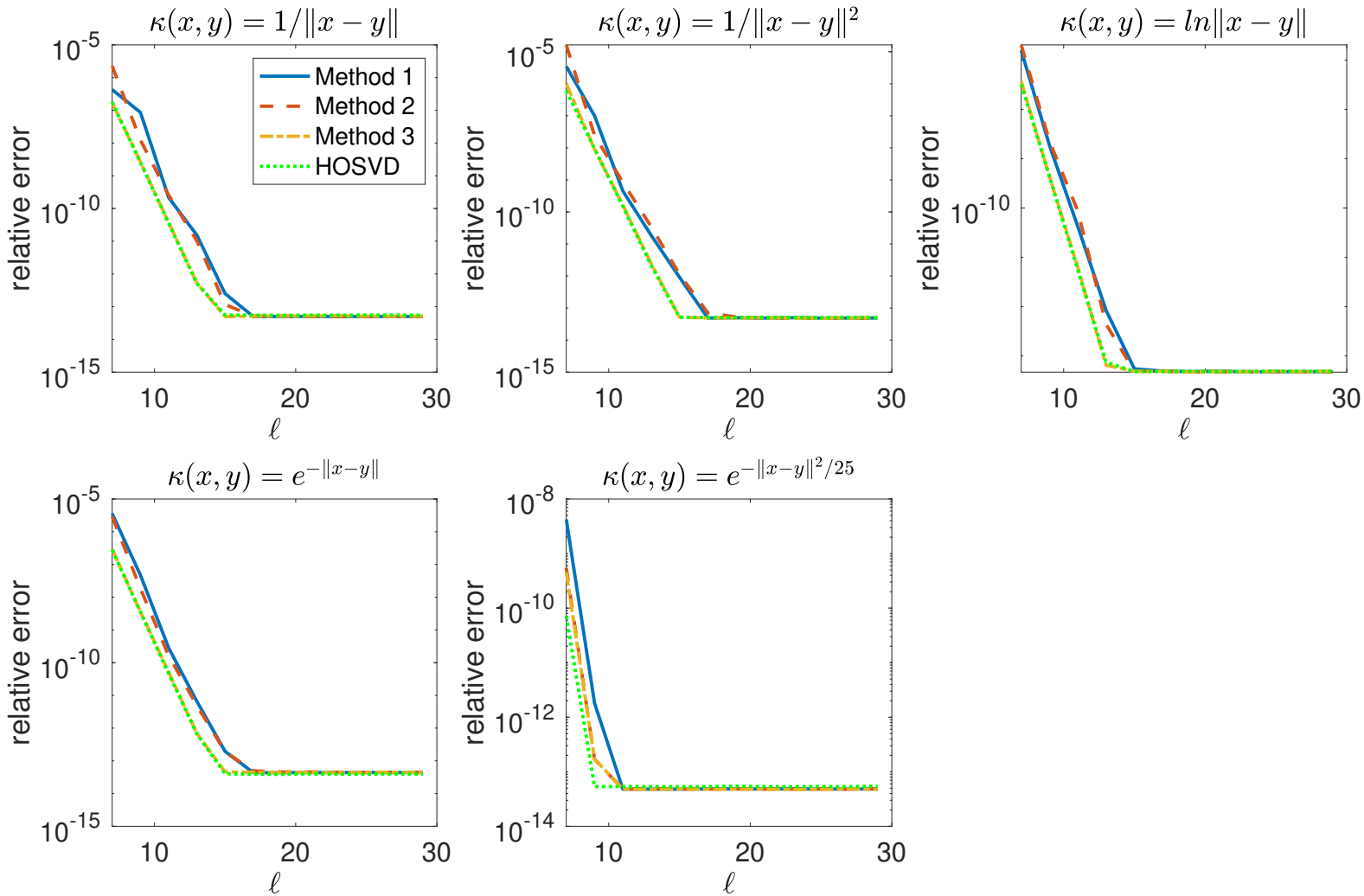
Compression Method	Computational Cost (flops)	Kernel Evals.
Method 1	$\mathcal{O}\left(rn^4\right)$	n^4
Method 2	$\mathcal{O}(b^3 n)$	$b^3 n + (r+p)^4$
Method 3	$\mathcal{O}\left(rn^4\right)$	n^4

Numerical Results Setup

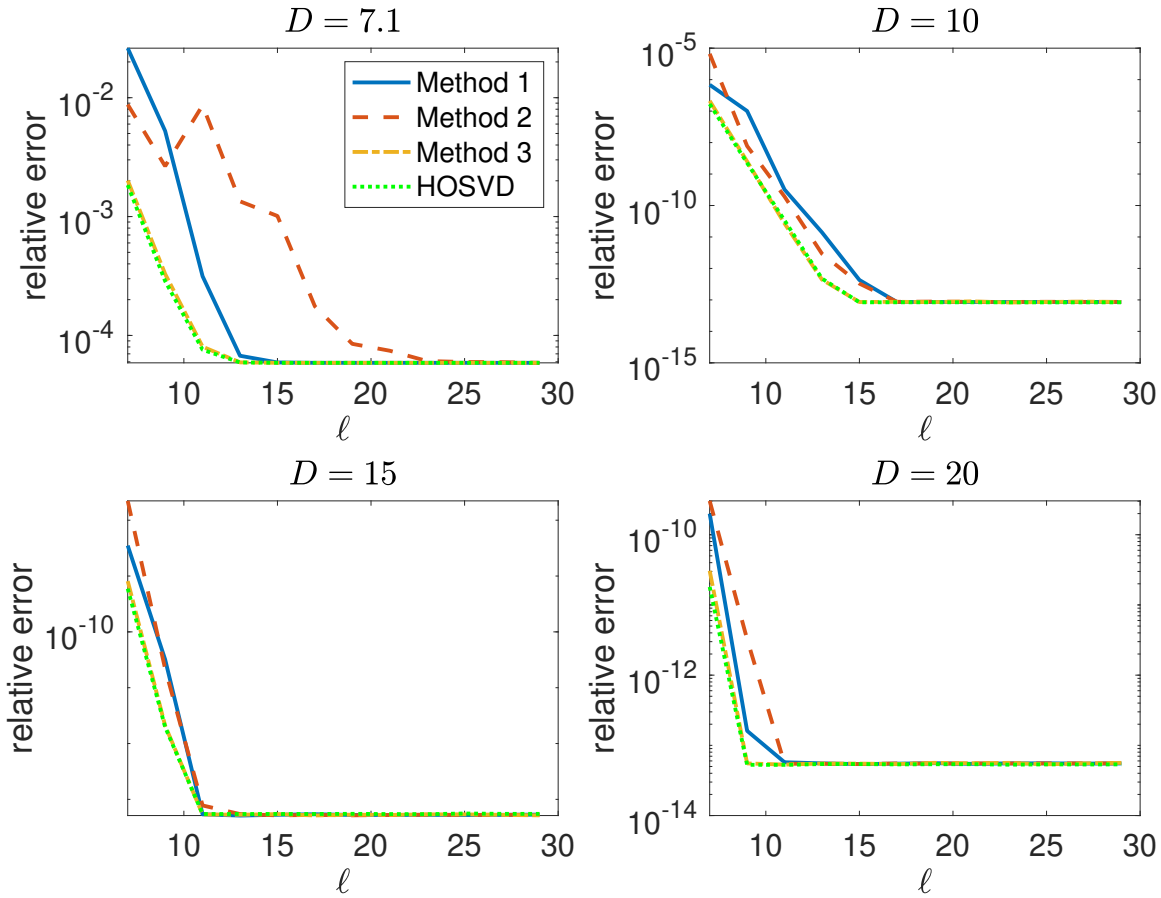
- Generate $N_s = N_t = 5000$ random points within boxes of length $L = 5$, $D = 10$ units apart, with angle $\theta = \pi/4$
- $n = 30$ Chebyshev nodes
- oversampling parameter $p = 5$, $\ell = r + p$
- kernel $\kappa(x, y) = 1/\|x - y\|_2$
- Error is computed in the ∞-norm



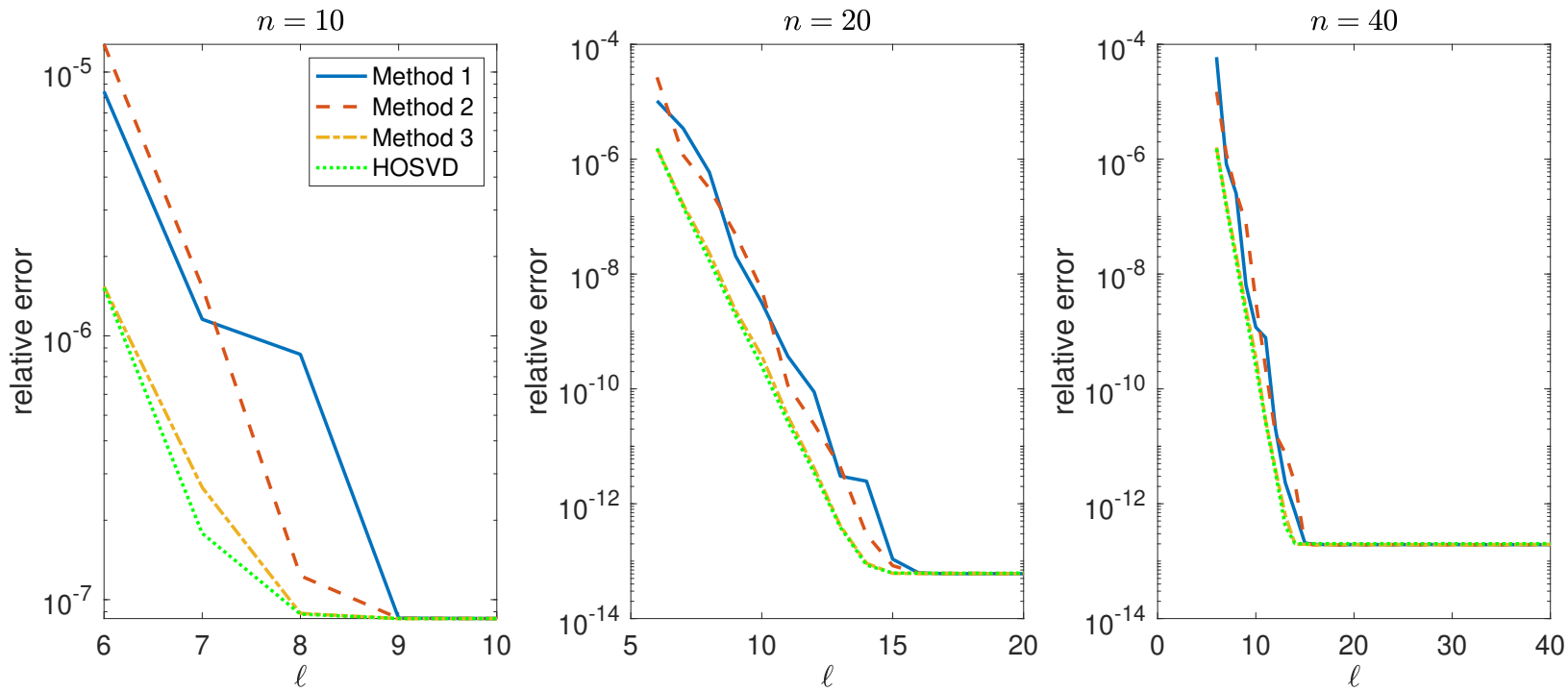
Accuracy with different kernels



Accuracy while varying distance D between boxes



Accuracy with increasing n



Conclusions

Contributions:

- Use of tensor-based methods for computing efficient low-rank kernel approximations
  - for any number of spatial dimensions
- New randomized tensor compression methods for low-rank Tucker approximations
  - Reduce computational costs of standard algorithms
  - Similar accuracy to standard algorithms
  - Error analysis included in preprint

In preparation: Minster, Saibaba, Kilmer, *Efficient Tensor-based Approximation to Kernel Interactions*