Hierarchical Matrices

Jon Cockayne

April 18, 2017

Sources

- "Introduction to Hierarchical Matrices with Applications" [Börm et al., 2003]

Sources

- "Introduction to Hierarchical Matrices with Applications" [Börm et al., 2003]
- "Fast Random Field Generation with $\mathcal{H}$-Matrices" [Feischl et al., 2017]

Introduction to Hierarchical Matrices

Motivation

Consider sampling from the Gaussian process

$$u \sim \mathcal{GP}(0, k)$$
$$k(x, x') = \sigma^2 \exp\left(-\frac{|x - x'|}{\ell}\right)$$

This can be achieved by sampling $\xi \sim N(0, I)$ and computing $K^{\frac{1}{2}}\xi$.

Motivation

Consider sampling from the Gaussian process

$$u \sim \mathcal{GP}(0, k)$$
$$k(x, x') = \sigma^2 \exp\left(-\frac{|x - x'|}{\ell}\right)$$

This can be achieved by sampling $\xi \sim N(0, I)$ and computing $K^{\frac{1}{2}}\xi$.

The kernel is non-smooth at $x = x'$ and globally supported...

Motivation

Consider sampling from the Gaussian process

$$u \sim \mathcal{GP}(0, k)$$
$$k(x, x') = \sigma^2 \exp\left(-\frac{|x - x'|}{\ell}\right)$$

This can be achieved by sampling $\xi \sim N(0, I)$ and computing $K^{\frac{1}{2}}\xi$.

The kernel is non-smooth at $x = x'$ and globally supported...but it is smooth elsewhere.

Motivation

Hierarchical matrices seek to exploit local smoothness to provide:

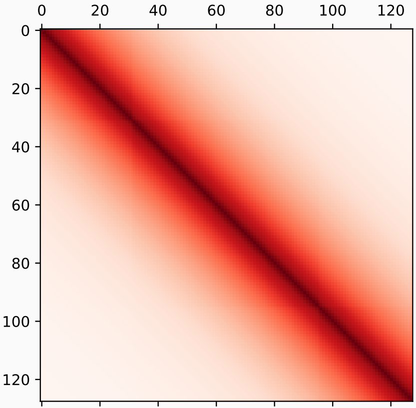
- An approximation to the matrix with low storage requirements...

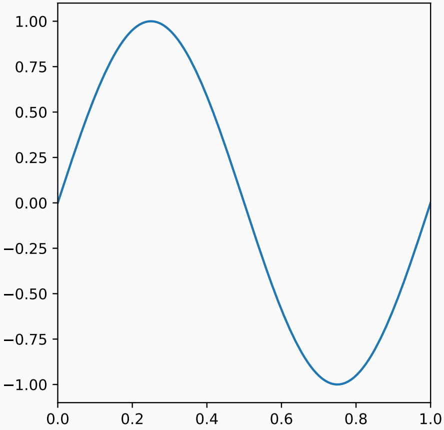
Motivation

Hierarchical matrices seek to exploit local smoothness to provide:

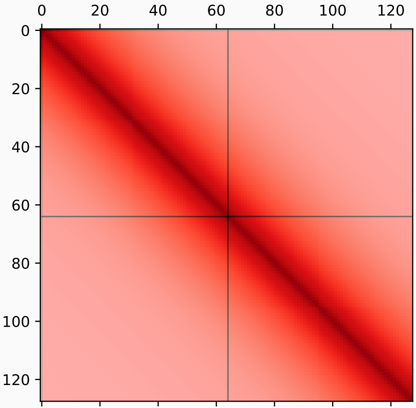
- An approximation to the matrix with low storage requirements...
- ...with which linear algebra can be performed more efficiently.

Motivation

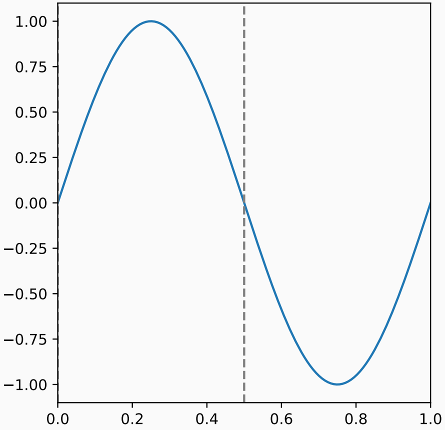

0
20
40
60
80
100
120
0
20
40
60
80
100
120


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
0.2
0.4
0.6
0.8
1.0

Motivation

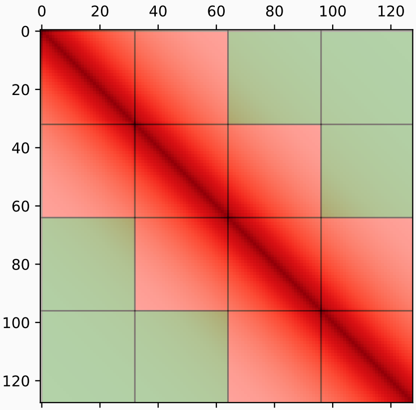


0
20
40
60
80
100
120

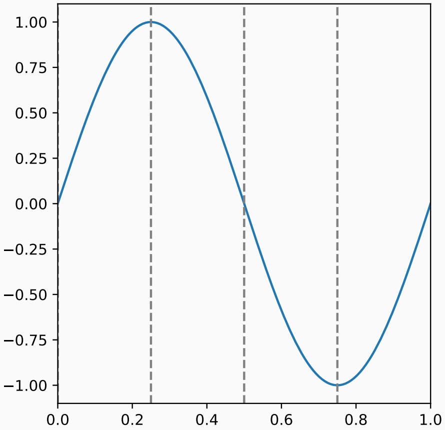


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
0.2
0.4
0.6
0.8
1.0

Motivation

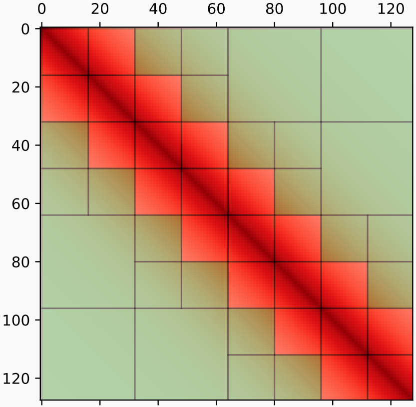


0
20
40
60
80
100
120
0
20
40
60
80
100
120

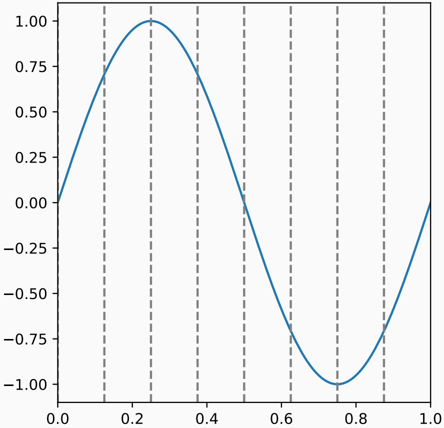


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
0.2
0.4
0.6
0.8
1.0

Motivation

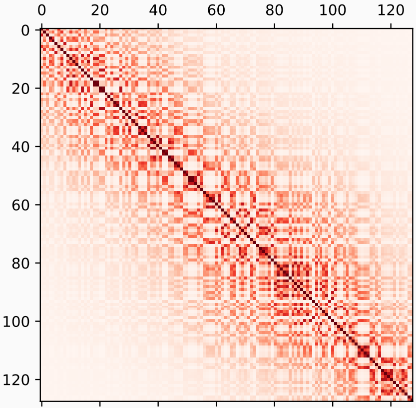


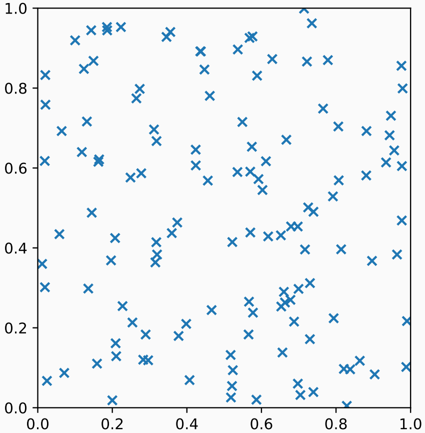
0
20
40
60
80
100
120



1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
0.2
0.4
0.6
0.8
1.0

To generalize a more structured strategy is required...





Strategy

1. Define a cluster tree, which hierarchically splits the points into clusters.
2. Define an admissibility condition to decide how to break the matrix into a block tree.
3. Construct a low-rank approximation to each block.

Cluster Trees

Cluster trees describe how we divide the points into clusters, which we later block together.

Cluster Trees

Cluster trees describe how we divide the points into clusters, which we later block together.

Let $\mathcal{T}_I$ be a tree, and T_I its node set. Let I denote a set of particles. $\mathcal{T}_I$ is a cluster tree if:

- Each $\tau \in T_I$ is associated with a subset of I.
- The root node contains all particles.
- Each non-leaf node has two children.
- If a node τ has children τ_1 and τ_2, then $\tau = \tau_1 \cup \tau_2$ and τ_1, τ_2 are disjoint.

Also denote by $\mathcal{L}_I$ the leaf nodes of the tree.

Cluster Trees

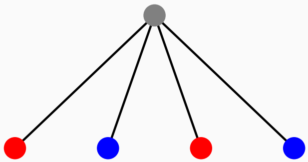
Cluster trees describe how we divide the points into clusters, which we later block together.

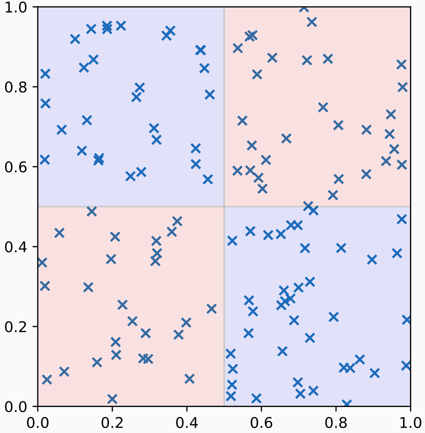
Let $\mathcal{T}_I$ be a tree, and T_I its node set. Let I denote a set of particles. $\mathcal{T}_I$ is a cluster tree if:

- Each $\tau \in T_I$ is associated with a subset of I.
- The root node contains all particles.
- Each non-leaf node has two children.
- If a node τ has children τ_1 and τ_2, then $\tau = \tau_1 \cup \tau_2$ and τ_1, τ_2 are disjoint.

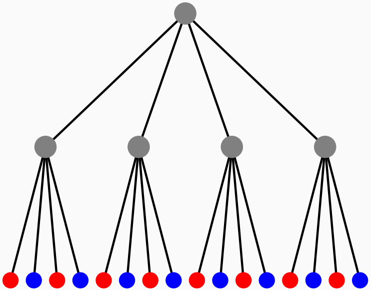
Also denote by $\mathcal{L}_I$ the leaf nodes of the tree.

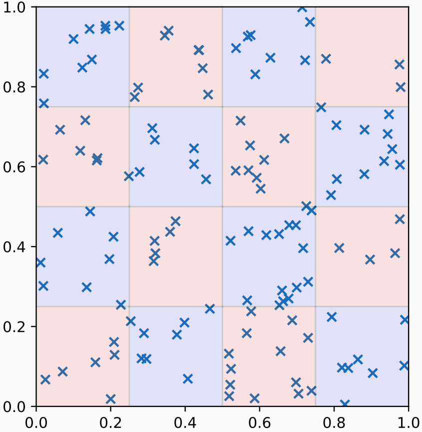
Cluster Trees





Cluster Trees





Admissibility Conditions

Admissibility conditions give a “rule of thumb” for whether a particular pair $(\tau, \sigma) \in T_I \times T_I$ has a suitably low-rank approximation.

Admissibility Conditions

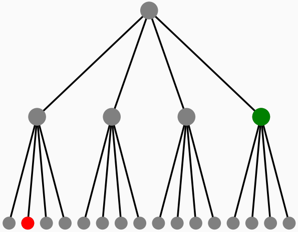
Admissibility conditions give a "rule of thumb" for whether a particular pair $(\tau, \sigma) \in T_I \times T_I$ has a suitably low-rank approximation.

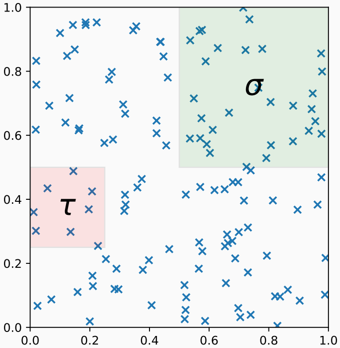
If we have some concept of the "domain" of τ, Ω_τ, use:

$$\min(\text{diam}(\Omega_\tau), \text{diam}(\Omega_\sigma)) \leq \eta \, \text{dist}(\Omega_\tau, \Omega_\sigma)$$

where η controls the tradeoff between complexity and approximation quality.

Admissibility Conditions



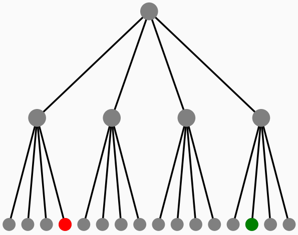


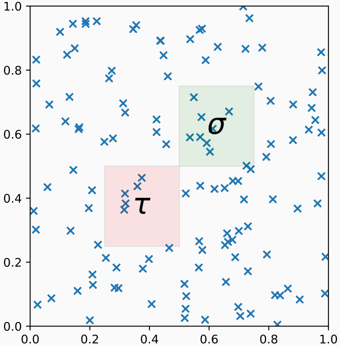
$$\mathrm{diam}(\Omega_\tau) = 0.176$$

$$\mathrm{diam}(\Omega_\sigma) = 0.707$$

$$\mathrm{dist}(\Omega_\tau, \Omega_\sigma) = 1.25$$

Admissibility Conditions





$$\mathrm{diam}(\Omega_\tau) = \mathrm{diam}(\Omega_\sigma) = 0.176$$

$$\mathrm{dist}(\Omega_\tau, \Omega_\sigma) = 0.707$$

Admissibility Conditions

In practise, bounding boxes of points are used to construct Ω_τ.

Block Tree

A Block Tree is a tree $\mathcal{T}_{I \times I}$ constructed recursively.

Block Tree

A Block Tree is a tree $\mathcal{T}_{I \times I}$ constructed recursively.

```
def build_block_tree(tau_times_sigma):
    if admissible(tau_times_sigma):
        tau_times_sigma.children = []
    else:
        tau_times_sigma.children = potential_children(tau_times_sigma)
        for child in tau_times_sigma.children:
            build_block_tree(child)
```

Block Tree

A Block Tree is a tree $\mathcal{T}_{I \times I}$ constructed recursively.

```
def build_block_tree(tau_times_sigma):
    if admissible(tau_times_sigma):
        tau_times_sigma.children = []
    else:
        tau_times_sigma.children = potential_children(tau_times_sigma)
        for child in tau_times_sigma.children:
            build_block_tree(child)
```

i.e. for each submatrix, it has a leaf at the largest block size which is admissible.

Low-Rank Approximation

In the admissible leaf nodes we replace $k(x, y)$ by an approximation $\tilde{k}(x, y)$.

Low-Rank Approximation

In the admissible leaf nodes we replace $k(x, y)$ by an approximation $\tilde{k}(x, y)$.

Standard to use a degenerate approximation

$$\tilde{k}(x, y) = \sum_{v=1}^{k} p_v^x \, q_v(y)$$

p_v^x are coefficients, estimated for each block. $q_v(y)$ are basis functions.

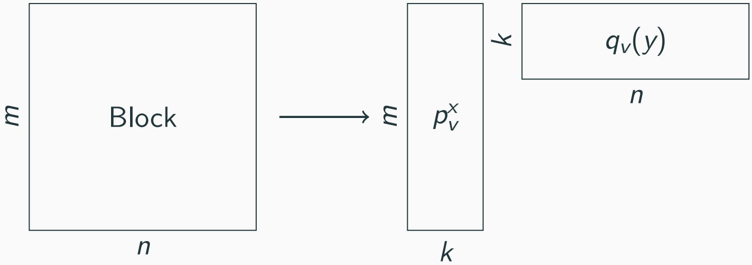
Low-Rank Approximation

In the admissible leaf nodes we replace $k(x, y)$ by an approximation $\tilde{k}(x, y)$.

Standard to use a degenerate approximation

$$\tilde{k}(x, y) = \sum_{v=1}^{k} p_v^x \, q_v(y)$$

p_v^x are coefficients, estimated for each block. $q_v(y)$ are basis functions.



Low-Rank Approximation

Common choices:

- Taylor expansion
- Interpolating polynomials (e.g. Lagrange Polynomials)

Definition 1 (Hierarchical Matrix)

An Hierarchical Matrix with blockwise rank k is a Matrix L associated with a block tree $\mathcal{T}_{I \times I}$ for which all admissible leaves $\tau \times \sigma \in T_{I \times I}$ have $\text{rank}(L|_{\tau \times \sigma}) \leq k$

Introduction to Hierarchical Matrices

$\mathcal{H}^2$-matrices

Uniform $\mathcal{H}$-matrices

$\mathcal{H}^2$ matrices use a particular approximation $\tilde{k}$ which gives improved efficiency:

$$\tilde{k}(x,y) = \sum_{i=1}^{k_\tau} \sum_{j=1}^{k_\sigma} k(x_i^\tau, x_j^\sigma) p_i^\tau(x) p_j^\tau(y)$$

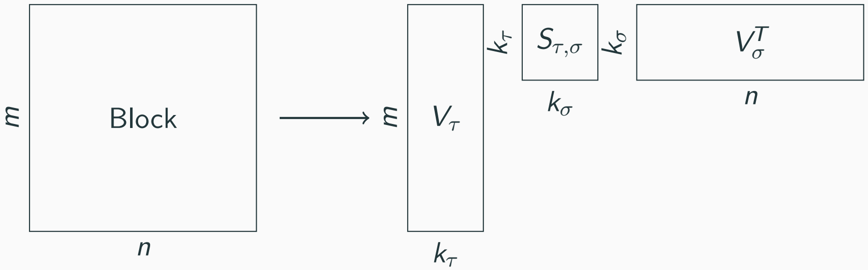
i.e. we use an tensor approximation of k in both arguments.

Uniform $\mathcal{H}$-matrices

$\mathcal{H}^2$ matrices use a particular approximation $\tilde{k}$ which gives improved efficiency:

$$\tilde{k}(x,y) = \sum_{i=1}^{k_\tau} \sum_{j=1}^{k_\sigma} k(x_i^\tau, x_j^\sigma) p_i^\tau(x) p_j^\tau(y)$$

i.e. we use an tensor approximation of k in both arguments.



Uniform $\mathcal{H}$-matrices

The matrices $\{V_\tau\}$ are called a cluster basis.

Definition 2 (Uniform $\mathcal{H}$-matrix)

Let $L \in \mathbb{R}^I \times \mathbb{R}^I$ be an $\mathcal{H}$-matrix. Let V be a cluster basis. L is a uniform $\mathcal{H}$-matrix with respect to V and the coefficient family $\{S_{\tau,\sigma} : \tau \times \sigma \in \mathcal{L}_{I\times I}\}$ if:

$$L|_{\tau\times\sigma} = V_\tau S_{\tau,\sigma} V_\sigma^T$$

The matrices $\{V_\tau\}$ are called a cluster basis.

Definition 2 (Uniform $\mathcal{H}$-matrix)

Let $L \in \mathbb{R}^I \times \mathbb{R}^I$ be an $\mathcal{H}$-matrix. Let V be a cluster basis. L is a uniform $\mathcal{H}$-matrix with respect to V and the coefficient family $\{S_{\tau,\sigma} : \tau \times \sigma \in \mathcal{L}_{I \times I}\}$ if:

$$L|_{\tau\times\sigma} = V_\tau S_{\tau,\sigma} V_\sigma^T$$

This has lower storage requirements:

- Storage per-cluster instead of per-block
- The matrices $S_{\tau,\sigma}$ are small.

Uniform $\mathcal{H}$-matrices

The matrices $\{V_\tau\}$ are called a cluster basis.

Definition 2 (Uniform $\mathcal{H}$-matrix)

Let $L \in \mathbb{R}^I \times \mathbb{R}^I$ be an $\mathcal{H}$-matrix. Let V be a cluster basis. L is a uniform $\mathcal{H}$-matrix with respect to V and the coefficient family $\{S_{\tau,\sigma} : \tau \times \sigma \in \mathcal{L}_{I \times I}\}$ if:

$$L|_{\tau\times\sigma} = V_\tau S_{\tau,\sigma} V_\sigma^T$$

This has lower storage requirements:

- Storage per-cluster instead of per-block
- The matrices $S_{\tau,\sigma}$ are small.

$\mathcal{H}^2$-matrices

Suppose that the approximation order k_τ is fixed to k_0 for all τ.

$\mathcal{H}^2$-matrices

Suppose that the approximation order k_τ is fixed to k_0 for all τ.

In this case we can express blocks of V_τ in terms of its children $V_{\tau'}$:

$$V_\tau = \begin{bmatrix} V_{\tau_1} B_{\tau_1,\tau} \\ V_{\tau_2} B_{\tau_2,\tau} \\ \vdots \\ V_{\tau_n} B_{\tau_n,\tau} \end{bmatrix}$$

$\mathcal{H}^2$-matrices

Suppose that the approximation order k_τ is fixed to k_0 for all τ.

In this case we can express blocks of V_τ in terms of its children $V_{\tau'}$:

$$V_\tau = \begin{bmatrix} V_{\tau_1} B_{\tau_1,\tau} \\ V_{\tau_2} B_{\tau_2,\tau} \\ \vdots \\ V_{\tau_n} B_{\tau_n,\tau} \end{bmatrix}$$

This means that we only need to store the V_τ corresponding to leaf nodes and the transfer matrices $B_{\tau',\tau}$.

Definition 3 ($\mathcal{H}^2$-matrix)

A uniform $\mathcal{H} - \textit{matrix}$ is called an $\mathcal{H}^2 - \textit{matrix}$ if the cluster basis V is nested.

References

Steffen Börm, Lars Grasedyck, and Wolfgang Hackbusch. Introduction to hierarchical matrices with applications. *Engineering Analysis with Boundary Elements*, 27(5):405–422, 2003.

Michael Feischl, Frances Kuo, and Ian H. Sloan. Fast random field generation with *h*-matrices, 2017.