StreaMRAK a streaming multi-resolution adaptive kernel algorithm☆,☆☆

Andreas Oslandsbotn [a,b,*], Željko Kereta [c], Valeriya Naumova [d], Yoav Freund [e], Alexander Cloninger [e]

[a] University of Oslo, Problemveien 7, Oslo 0315, Norway
[b] Simula School of Research and Innovation, Martin Linges Vei 25, Fornebu 1364, Norway
[c] University College London, Gower St, London WC1E 6BT, England
[d] Simula Research Laboratory, Martin Linges vei 25, Fornebu 1364, Norway
[e] University of California San Diego, 9500 Gilman Dr, La Jolla CA 92093, United States

ARTICLE INFO

ABSTRACT

Kernel ridge regression (KRR) is a popular scheme for non-linear non-parametric learning. However, existing implementations of KRR require that all the data is stored in the main memory, which severely limits the use of KRR in contexts where data size far exceeds the memory size. Such applications are increasingly common in data mining, bioinformatics, and control. A powerful paradigm for computing on data sets that are too large for memory is the *streaming model of computation*, where we process one data sample at a time, discarding each sample before moving on to the next one. In this paper, we propose StreaMRAK - a streaming version of KRR. StreaMRAK improves on existing KRR schemes by dividing the problem into several levels of resolution, which allows continual refinement to the predictions. The algorithm reduces the memory requirement by continuously and efficiently integrating new samples into the training model. With a novel sub-sampling scheme, StreaMRAK reduces memory and computational complexities by creating a *sketch* of the original data, where the sub-sampling density is adapted to the bandwidth of the kernel and the local dimensionality of the data. We present a showcase study on two synthetic problems and the prediction of the trajectory of a double pendulum. The results show that the proposed algorithm is fast and accurate.

1. Introduction

Machine learning algorithms based on kernel ridge regression (KRR) [1] is an active field of research [2–6], with applications ranging from time series prediction in finance [7], parameter inference in dynamical systems [8], to pairwise learning

---

[9], face recognition [10] and drug estimation and gene analysis in biomedicine [11,12]. This paper develops a streaming variation of KRR using a radial kernel, a new sub-sampling scheme, and a multi-resolution formulation of the learning model.

Many popular data analysis software packages, such as Matlab$^{\text{TM}}$ require loading the entire dataset into memory. While computer memory is growing fast, the size of available data sets is growing much faster, limiting the applicability of *in-memory* methods.[1]

Streaming [13] is a computational model where the input size is much larger than the memory size. Streaming algorithms read one item at a time, update their memory, and discard the item. The computer memory is used to store a *model* or a *sketch* of the overall data distribution, which is orders of magnitude smaller than the data itself. The development of streaming algorithms is experiencing increased popularity in the face of big data applications such as data mining [14] and bioinformatics [15], where data sets are typically too large to be kept *in-memory*. Many big data applications call for non-linear and involved models, and thus, the development of non-parametric and non-linear models is critical for successful learning.

Among the most popular non-parametric learning algorithms are kernel methods, which include well-known learning schemes such as the support vector machine (SVM) and KRR, to name a few. The appeal of kernel methods lies in their strong theoretical foundation [1,16], as well as their ability to map complex problems to a linear space without requiring an explicit mapping. A common class of kernels are radial kernels $k(\mathbf{x}, \tilde{\mathbf{x}}) = \Phi(\|\mathbf{x} - \tilde{\mathbf{x}}\|/r)$ for $\mathbf{x}, \tilde{\mathbf{x}} \in \mathcal{X} \subseteq \mathbb{R}^D$ and bandwidth $r > 0$ [17]. These kernels are universal (with a few exceptions [18]), meaning that they can approximate any bounded continuous function on $\mathcal{X}$ arbitrarily well. However, in high dimensions, kernel methods suffer from the "curse of dimensionality" and require large amounts of training data to converge. Furthermore, the computational complexity, memory requirement, and the number of parameters to learn grow unbounded with the number of training samples, a drawback known as the "curse of kernelization" [19]. In the context of streaming, the prospect of unbounded data streams makes this shortcoming even more detrimental.

Although kernel-based learning schemes are typically formulated as convex optimization problems, which do not require tuning hyper-parameters such as learning rate etc., there is still a need to determine the optimal kernel. For the Gaussian kernel, this amounts to selecting the bandwidth. Classically, an optimal kernel is chosen through batch techniques such as leave-one-out and k-fold cross-validation [20–22]. However, these approaches are inefficient as they spend significant time evaluating bad kernel hypotheses and often use multiple runs over the data, which is impossible in a streaming setting.

To meet a need for non-linear non-parametric algorithms for streaming data, we propose the *streaming multi-resolution adaptive kernel algorithm* (StreaMRAK) - a computationally and memory-efficient streaming variation of KRR. StreaMRAK address the kernel selection problem with a multi-resolution kernel selection strategy that adapts the sub-sample density to the kernel bandwidth over several levels of resolution. Furthermore, StreaMRAK addresses the curse of dimensionality and kernelization in a novel way, through the sub-sampling scheme.

1.1. Setting

We consider a finite sample data-cloud $\mathcal{X}$, $|\mathcal{X}| = n$, that is sampled i.i.d. according to a fixed but unknown distribution $\mathcal{P}$ over $\mathbb{R}^D$. The target is a bounded and continuous function $f : \mathbb{R}^D \to \mathbb{R}$. We assume that the points in $\mathcal{X}$ are placed in a *sequence* and that their order is random. [2] Each instance $\mathbf{x}_i \in \mathcal{X}$, for $i \in [n]$, paired with a label y_i where $y_i = f(\mathbf{x}_i) + \varepsilon_i$ and $\varepsilon_i \sim \mathcal{N}(0, \sigma)$ represents noise. The task of learning is to train a model $\widehat{f}$ that is a good approximation of the target function f.

In this work, we think about the intrinsic dimension of $\mathcal{X}$ as a local quantity, meaning it depends on the region $\mathcal{A} \subseteq \mathcal{X}$ and the radius r at which we consider the point cloud. To estimate the local intrinsic dimension in a "location and resolution sensitive" manner, we use the concept of the doubling dimension of a set, defined in Def. 1.2. See [23,24] for related definitions.

Definition 1.1 (Covering number). Consider a set $\mathcal{A}$ and a ball $\mathcal{B}(\mathbf{x}, r)$, with $r > 0$ and $\mathbf{x} \in \mathcal{A}$. We say that a finite set $\mathcal{S} \subset \mathcal{B}(\mathbf{x}, r)$ is a covering of $\mathcal{B}(\mathbf{x}, r)$ in $\mathcal{A}$ if $\mathcal{A} \cap \mathcal{B}(\mathbf{x}, r) \subset \cup_{\mathbf{x}_i \in \mathcal{S}} \mathcal{B}(\mathbf{x}_i, r/2)$. We define the covering number $\kappa(\mathcal{A}, \mathbf{x}, r)$ as the minimum cardinality of any covering of $\mathcal{B}(\mathbf{x}, r)$ in $\mathcal{A}$.

Definition 1.2 (Doubling dimension). The doubling dimension $\mathrm{ddim}(\mathcal{A}, r)$ of a set $\mathcal{A}$ is defined as $\mathrm{ddim}(\mathcal{A}, r) = \lceil \log \kappa(\mathcal{A}, \mathbf{x}, r) \rceil$. For an interval $\mathcal{I} \subset \mathbb{R}_{>0}$ we define the doubling dimension as the least upper bound over $r \in \mathcal{I}$, that is $\mathrm{ddim}(\mathcal{A}, \mathcal{I}) = \max_{r \in \mathcal{I}} \mathrm{ddim}(\mathcal{A}, r)$.

We say that the intrinsic dimension of $\mathcal{X}$ changes with the location if there exist $\mathcal{A}_1, \mathcal{A}_2 \subset \mathcal{X}$ such that $\mathrm{ddim}(\mathcal{A}_1, r) \neq \mathrm{ddim}(\mathcal{A}_2, r)$ for $r > 0$. Similarly, we say that the intrinsic dimensionality of $\mathcal{X}$ changes with the resolution, if there exist $r_1 \neq r_2$ such that the doubling dimension $\mathrm{ddim}(\mathcal{A}, r_1) \neq \mathrm{ddim}(\mathcal{A}, r_2)$ for $\mathcal{A} \subset \mathcal{X}$.

In Fig. 1 we consider three examples to provide further insight on the doubling dimension. In Fig. 1a we see a domain shaped like a dumbbell, where the spheres are high dimensional, and the bar connecting them is lower-dimensional,

---

[1] Simulink$^{\text{TM}}$, a companion software to Matlab$^{\text{TM}}$ supports streaming but has a much more limited computational model, targeted at signal processing applications.

[2] The assumption that the sequence is randomly ordered allows us to draw statistical conclusions from prefixes.
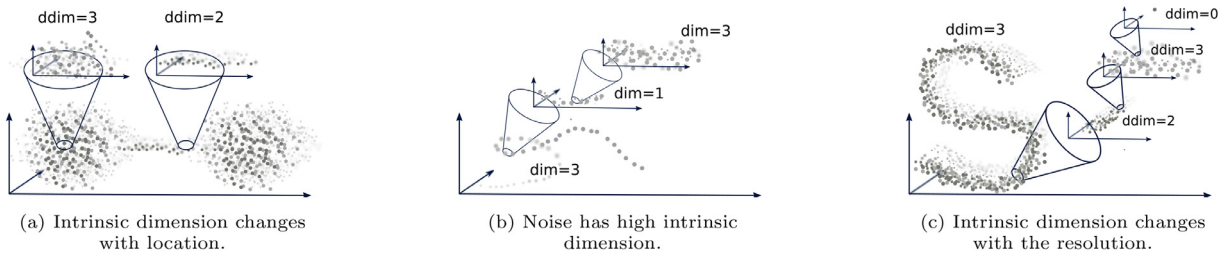
(a) Intrinsic dimension changes with location.

(b) Noise has high intrinsic dimension.

(c) Intrinsic dimension changes with the resolution.

Fig. 1. Three examples of variation in the intrinsic dimension. The coloring of the point clouds illustrates depth.
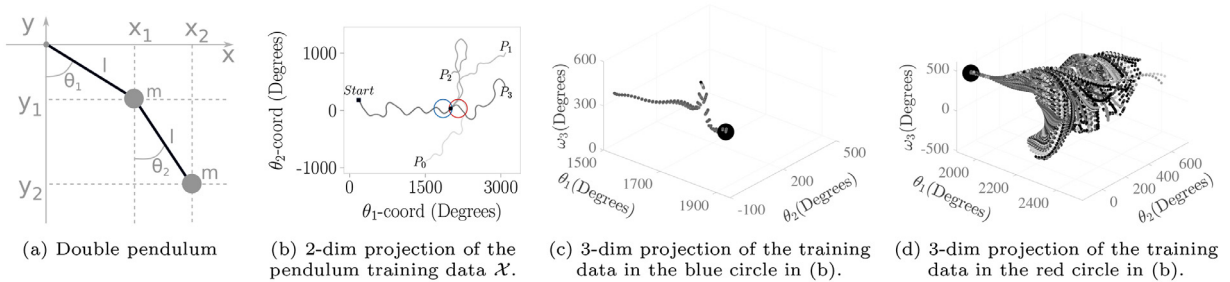

(a) Double pendulum

(b) 2-dim projection of the pendulum training data $\mathcal{X}$.

(c) 3-dim projection of the training data in the blue circle in (b).

(d) 3-dim projection of the training data in the red circle in (b).

Fig. 2. (a) Illustration of a double pendulum. Here l and m are the length and mass of the pendulum rods, and θ_1, θ_2 are the angles. Furthermore, (x_1, y_1) and (x_2, y_2) are the positions of the point masses of the two pendulums. (b) Phase diagram of four double pendulums P_0, P_1, P_2, P_3, iterated for $T = 500$ time steps. The bifurcation point at step $T = 300$ is indicated with a black solid circle. (c) and (d) includes the ω_1 axis and zoom in on respectively the blue and red circles in (b). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

showing how the dimension can change with the location. Meanwhile, Fig. 1b illustrates a lower-dimensional manifold, embedded in $\mathbb{R}^3$, with manifold noise ζ_m. We see that when the resolution is sufficiently small, so that $r \approx \zeta_m$, the doubling dimensionality increases towards the dimension of the ambient space $\mathbb{R}^D$. Furthermore, Fig. 1c shows a point cloud that is locally 2-dimensional, but is embedded in a 3-dimensional space. By reducing r we can resolve this lower dimensionality, but if it is reduced further, we would eventually resolve the noise level, and the doubling dimension increases again.

As an example of how the intrinsic dimension might change with respect to regions and resolutions, we consider a double pendulum system, a well-known chaotic system that depends heavily on its initial conditions [25]. Systems with multiple pendulum elements are well known in engineering applications such as mechanical and robotic systems with several joints and are studied for their chaotic properties [26].

In Fig. 2b we visualize the trajectory of four pendulums P_0, P_1, P_2, P_3, for which the trajectories are indistinguishable until a bifurcation occurs around $T = 300$ time steps, and the trajectories start to diverge. In Fig. 2c and Fig. 2d we zoom in on the trajectory of all 500 pendulums in regions before and after the bifurcation. These two regions, $\mathcal{A}_1$ and $\mathcal{A}_2$, are indicated by a blue and red circle, respectively, in Fig. 2b. From the figures, it is clear that learning the trajectory in $\mathcal{A}_1$ is significantly easier than in $\mathcal{A}_2$, where learning the trajectory is more affected by the curse of dimensionality.

1.2. Contribution and comparison to related work

Contributions of this work can be divided into three components.

(C1) A multi-resolution variation of the state-of-the-art KRR solver **FALKON** [2], using the LP, which refines the predictions at each level of resolution by regressing on the errors from the previous level.
(C2) A novel sub-sampling scheme for kernel methods, tailored for use in combination with the LP, that can handle the curse of dimensionality and does not require the data to be *in-memory*.
(C3) Development of a streaming variation of **FALKON**, where the time and memory requirements depend on the doubling dimensionality and the level of resolution, instead of the number of training points. see Props. 5.3–5.5.

In the following, we give further details on these contributions and compare them to related work. The computational backbone of **StreamRAK** is based on the state-of-the-art KRR solver **FALKON** [2], which among other things combines sub-sampling and preconditioning to process large data sets efficiently. However, **FALKON** relies on selecting an optimal kernel bandwidth, which can be inefficient in streaming.

Our first contribution (C1) addresses the issue of selecting an optimal bandwidth by introducing a multi-resolution reformulation of **FALKON** using a changing bandwidth variation of the LP [27–29]. This strategy is inspired by the success of existing multi-resolution approaches [4,30–36] and, for online learning, adaptive bandwidth approaches [37–39]. Our scheme combines the LP with a localized kernel, which gives a frequency and location-based discretization, similar to wavelet anal-

ysis that have shown great success in numerous applications. However, typical wavelet architectures [40–46] require upfront construction of a wavelet basis, which is not compatible with a data-adaptive kernel.

In this work, we aim to show that the LP is a viable multi-resolution scheme and can be modified to the streaming setting. Furthermore, we provide convergence bounds for the LP in the context of radial kernels and KRR, and show experimentally that it improves the estimation accuracy.

Let us now discuss our second contribution (C2). FALKON addresses the curse of kernelization by combining Nyström sub-sampling, conjugate gradient, and preconditioning, and achieves time and memory requirements of $\mathcal{O}(n\sqrt{n})$ and $\mathcal{O}(n)$ respectively, where n is the number of samples. In recent years there have been several efforts to address the curse of kernelization in similar ways through sub-sampling techniques such as sketching [3,5], randomized features [47–50] and Nyström sub-sampling [51–55]. However, despite their successes, these techniques are in principle *in-memory* type algorithms since they require access to the training data in advance of the training and are not optimized for streaming.

Furthermore, FALKON selects the sub-samples uniformly over the input domain $\mathcal{X}$. However, when learning with a radial kernel, the density of samples should be related to the bandwidth of the kernel. Otherwise, a too-small bandwidth will lead to bad interpolation properties, while a too-large bandwidth gives an ill-conditioned system [34]. Since the LP scheme reduces the kernel bandwidth at each level of resolution, it would be problematic to use the same sub-sample density. Furthermore, due to the curse of dimensionality, the covering number increases exponentially with the doubling dimension. Therefore, if doubling dimensionality varies across different regions of the domain $\mathcal{X}$, as illustrated by Fig. 1a, then the number of sub-samples necessary to maintain the density for a given bandwidth will also vary.

Our second contribution (C2) provides an alternative sub-sampling strategy, adapting the sub-sampling density to the kernel bandwidth. This strategy is based on a damped cover-tree (DCT), which is a modified version of the cover-tree (CT) [24], a tree-based data structure with $\mathcal{O}(n)$ memory and $\mathcal{O}(c^6 n \log n)$ time.

A problem with an adaptive sub-sampling strategy is its vulnerability to the curse of dimensionality. In regions of high doubling dimensions, the number of samples to achieve a certain density increases exponentially, as quantified by Def. 1.2. This means that the number of sub-samples from the CT will quickly grow too large for efficient computing. The danger is to waste resources on samples from subsets and levels where the doubling dimension is so large that good interpolation cannot be achieved for any viable sample sizes. This would only serve to slow down the computation and not increase the precision.

Due to this, the DCT introduces a damping property, which gradually suppresses the selection of sub-samples where the doubling dimensionality is large. This has the additional advantage of allowing to choose more sub-samples from regions where the doubling dimensionality is small. Thus, the DCT can diminish the impact of the curse of dimensionality. Furthermore, the DCT can be built continuously as new samples come in, making it ideal for a streaming computational model.

Our third contribution (C3), is the streaming capabilities of StreaMRAK. In particular, the sub-sampling and kernel construction allows for continuous integration of new training points. Furthermore, the DCT, the multi-resolution construction, and the KRR solver can all be multi-threaded and parallelized.

1.3. Organization of the paper

The paper is organized as follows. Section 2 introduces kernel methods and the FALKON algorithm, as well as the LP. Section 3 introduces the adaptive sub-sampling scheme and the DCT. StreaMRAK is described in Section 4 and an analysis of the algorithm is given in Section 5. Finally, Section 6 presents several numerical experiments and Section 7 gives an outlook for further work. The Appendix includes further mathematical background and the proofs.

1.4. Notation

We denote vectors $\mathbf{a} \in \mathbb{R}^D$ with boldface and matrices $\mathbf{A} \in \mathbb{R}^{n \times m}$ with bold uppercase, and $\mathbf{A}^\top$ denotes the matrix transpose. We use $\mathbf{K}_{nm}$ for kernel matrices, where the dimensionality is indicated by the subscripts. We reserve n for the number of training samples and m for the number of sub-samples. The ij-th element of a kernel matrix is denoted $[\mathbf{K}_{nm}]_{ij}$, while for other matrices we use $\mathbf{A}_{ij}$. The notation a_i indicates i-th element of a vector $\mathbf{a}$. Furthermore, we use $f([\mathbf{x}_n])$ to denote $(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))^\top \in \mathbb{R}^n$, and $[m]$ to denote $\{i\}_{i=1}^m$. The notation $\mathbf{x}_i$ indicates the i-th training example. We use $\mathbf{a}^{(l)}$ and $\mathbf{A}^{(l)}$, where l refers to a specific level in the LP and the DCT. We take $\|\cdot\|$ to be the L^2 norm and $\|\cdot\|_{\mathcal{H}}$ to be the RKHS norm. We denote the intrinsic dimension of a manifold with d and the dimension of the embedding with D. By $\mathbb{1}_{\mathcal{S}}(\mathbf{x})$ we denote the indicator function, which evaluates to 1 if $\mathbf{x} \in \mathcal{S}$ and 0 otherwise, of a set $\mathcal{S} \subset \mathbb{R}^D$.

2. Kernel methods

Consider a positive definite kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, defined on an input space $\mathcal{X} \subset \mathbb{R}^D$. Given data $\{(\mathbf{x}_i, y_i) : i \in [n]\}$ of samples from $\mathcal{X} \times \mathbb{R}^D$, kernel ridge regression computes an estimator by minimising

$$\widehat{f}_{n,\lambda} = \underset{f \in \widetilde{\mathcal{H}}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} (f(\mathbf{x}_i) - y_i)^2 + \lambda \|f\|_{\mathcal{H}}^2,$$

where $\mathcal{H}$ is the Hilbert space induced by the kernel. This allows to reduce the problem to a linear system

$$(\mathbf{K}_{nn} + \lambda n \mathbf{I}_n)\boldsymbol{\alpha} = \mathbf{y}, \text{ for } [\mathbf{K}_{nn}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j), \text{ and } \mathbf{y} = (y_1, \ldots, y_n)^\top. \tag{2.1}$$

Coefficients $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^\top$ define the estimator by $f(\mathbf{x}) = \sum_{i=1}^n \alpha_i k(\mathbf{x}, \mathbf{x}_i)$. However, solving (2.1) using traditional methods has a time complexity of $\mathcal{O}(n^2)$, which can be costly for large n [2].

FALKON [2] addresses this issue by sub-sampling the columns of $\mathbf{K}_{nn}$, which reduces the effective complexity while maintaining accuracy. Namely, denote $\Gamma_n = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and for $m \ll n$ let $\widetilde{\Gamma}_m = \{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_m\}$ be Nyström centers (i.e. a randomly selected subset of Γ_n). Minimizing

$$\widehat{f}_{n,m,\lambda} = \underset{f \in \widehat{\mathcal{H}}_M}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^n (f(\mathbf{x}_i) - y_i)^2 + \lambda \|f\|_{\mathcal{H}}^2, \tag{2.2}$$

where $\widetilde{\mathcal{H}}_m = \overline{\operatorname{span}}\{k(\cdot, \tilde{\mathbf{x}}_j) : j \in [m]\}$, leads to a linear system

$$\mathbf{H}\widetilde{\boldsymbol{\alpha}} = \mathbf{z}, \text{ for } \mathbf{H} = \mathbf{K}_{nm}^\top \mathbf{K}_{nm} + \lambda n \mathbf{K}_{mm}, \text{ and } \mathbf{z} = \mathbf{K}_{nm}\mathbf{y}.$$

Here $[\mathbf{K}_{nm}]_{ij} = k(\mathbf{x}_i, \tilde{\mathbf{x}}_j) \in \mathbb{R}^{n \times m}$ is the column-subsampled matrix and the estimator is given by $\widehat{f}_{n,m,\lambda}(\mathbf{x}) = \sum_{j=1}^m \widetilde{\alpha}_j k(\mathbf{x}, \tilde{\mathbf{x}}_j)$. To further reduce the time complexity FALKON uses a suitable preconditioner to reduce the condition number. The preconditioner is defined as $\mathbf{B}\mathbf{B}^\top = (n/m\mathbf{K}_{mm}^2 + \lambda n \mathbf{K}_{mm})^{-1}$, which is a natural (lower complexity) approximation of the ideal preconditioner $\mathbf{A}\mathbf{A}^\top = (\mathbf{K}_{nm}^\top \mathbf{K}_{nm} + \lambda n \mathbf{K}_{mm})^{-1}$. We now solve for $\widetilde{\boldsymbol{\alpha}}$ from the system of equations

$$\mathbf{B}^\top \mathbf{H} \mathbf{B}\boldsymbol{\beta} = \mathbf{B}^\top \mathbf{z}, \text{ for } \mathbf{H} = \mathbf{K}_{nm}^\top \mathbf{K}_{nm} + \lambda n \mathbf{K}_{mm}, \ \mathbf{z} = \mathbf{K}_{nm}\mathbf{y}, \text{ and } \widetilde{\boldsymbol{\alpha}} = \mathbf{B}\boldsymbol{\beta}. \tag{2.3}$$

This is solved iteratively, using the conjugate gradients with early stopping. Choosing $m = \mathcal{O}(\sqrt{n})$ still ensures optimal generalisation (i.e. same as KRR), while reducing the computational complexity to $\mathcal{O}(n\sqrt{n})$.

2.1. Streaming adaptation of FALKON

Matrices and vectors involved in the linear system in (2.3) can be separated into two classes: those that depend only on sub-samples in $\widetilde{\Gamma}_m$; and those ($\mathbf{K}_{nm}^\top \mathbf{K}_{nm}$ and $\mathbf{z}$) that also depend on all the training points Γ_n. Critically, terms in both groups are all of size m, which allows to reduce the complexity. Consider now the set of sub-samples $\widetilde{\Gamma}_m$ to be fixed, and assume new training points, in the form $\{(\mathbf{x}_q, y_q) : q = n+1, \ldots, n+t)\}$, are coming in a stream. We can then update the second class of terms according to

$$\left[(\mathbf{K}_{(n+t)m})^\top \mathbf{K}_{(n+t)m}\right]_{ij} = \left[(\mathbf{K}_{nm})^\top \mathbf{K}_{nm}\right]_{ij} + \sum_{q=n+1}^{n+t} k(\mathbf{x}_q, \tilde{\mathbf{x}}_i)k(\mathbf{x}_q, \tilde{\mathbf{x}}_j), \tag{2.4}$$

$$\left[(\mathbf{K}_{(n+t)m})^\top \mathbf{y}\right]_i = z_i + \sum_{q=n+1}^{n+t} k(\mathbf{x}_q, \tilde{\mathbf{x}}_i)y_q. \tag{2.5}$$

Thus, only sub-samples $\widetilde{\Gamma}_m$, matrices $(\mathbf{K}_{nm})^\top \mathbf{K}_{nm}$, $\mathbf{K}_{mm}$ and $\mathbf{z}$, need to be stored. However, in order to continuously incorporate new training points into Eqs. (2.4) and (2.5), sub-samples $\widetilde{\Gamma}_m$ must be determined in advance. Whereas this works if all the data is provided beforehand, it cannot be done if the data arrives sequentially. In this work, we address this through a multi-resolution framework. The overall estimator is composed of a sequence of estimators defined at different resolution levels of the domain. Correspondingly, the set of sub-samples $\widetilde{\Gamma}_m$ consists of smaller sets $\widetilde{\Gamma}_{m^{(l)}}^{(l)}$ that correspond to individual levels of resolution. The sets $\widetilde{\Gamma}_{m^{(l)}}^{(l)}$ are filled as the data streams in, and once a set for a given level is deemed complete, we proceed with updating (2.4) and (2.5).

Further details of how the sets $\widetilde{\Gamma}_{m^{(l)}}^{(l)}$ are constructed, and the corresponding criteria, are provided in Sections 3 and 4. We begin by describing the multi-resolution framework of estimators.

2.2. The laplacian pyramid

The LP [27,29] is a multi-resolution regression method for extending a model $\widehat{f}$ to out-of-sample data points $\mathbf{x} \in \mathcal{X}/\Gamma_n$. The LP can be formulated for radial kernels in the form

$$k(\mathbf{x}_i, \mathbf{x}_j) = \Phi\left(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{r}\right), \tag{2.6}$$

where $r > 0$ is a shape parameter that determines the decay of Φ with respect to $\|\mathbf{x}_i - \mathbf{x}_j\|$, see [17]. The idea underpinning the LP is to approximate the target function sequentially, where at each stage we regress on the errors from the previous stage. In other words, we begin with a rough approximation using a large shape parameter for which Φ decays slowly and

then improve the approximation by fitting the resulting error and reducing the shape parameter. In the LP, the estimator at level $L \in \mathbb{N}$ is defined recursively as

$$\widehat{f}^{(L)}(\mathbf{x}) = \sum_{l=0}^{L} s^{(l)}(\mathbf{x}) = s^{(L)}(\mathbf{x}) + \widehat{f}^{(L-1)}(\mathbf{x}), \tag{2.7}$$

where $\widehat{f}^{(0)} = s^{(0)}$, and $s^{(l)}(\mathbf{x})$ is a correction term defined by

$$s^{(l)}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i^{(l)} k^{(l)}(\mathbf{x}, \mathbf{x}_i). \tag{2.8}$$

The coefficients $\boldsymbol{\alpha}^{(l)} = (\alpha_1^{(l)}, \ldots, \alpha_n^{(l)})^\top$ are computed by conducting KRR on the residuals, i.e. errors, from the estimator at the previous level. Namely, $\boldsymbol{\alpha}^{(l)} = (\mathbf{K}_{nn}^{(l)} + \lambda n \mathbf{I})^{-1} \mathbf{d}^{(l)}$, where

$$\mathbf{d}^{(l)} = \begin{cases} \mathbf{y}, & \text{if } \quad l = 0 \\ \mathbf{y} - \widehat{f}^{(l-1)}([\mathbf{x}_n]), & \text{otherwise} \end{cases}. \tag{2.9}$$

For a FALKON adaption of this scheme, we only need to modify how per-level coefficients are computed. Following (2.3) we iteratively solve

$$(\mathbf{B}^{(l)})^\top \mathbf{H}^{(l)} \mathbf{B}^{(l)} \boldsymbol{\beta}^{(l)} = (\mathbf{B}^{(l)})^\top \left(\mathbf{K}_{nm}^{(l)}\right)^\top \mathbf{d}^{(l)}, \tag{2.10}$$

where $\mathbf{B}^{(l)}$ is the corresponding preconditioner, and $\mathbf{H}^{(l)} = (\mathbf{K}_{nm}^{(l)})^\top \mathbf{K}_{nm}^{(l)} + \lambda n \mathbf{K}_{mm}^{(l)}$, and set $\widetilde{\boldsymbol{\alpha}}^{(l)} = \mathbf{B}^{(l)} \boldsymbol{\beta}^{(l)}$.

Remark 2.1. In this paper, we construct the kernel matrices $\mathbf{K}^{(l)}$ on a particular class of radial kernels, namely the Gaussian kernel

$$k^{(l)}(\mathbf{x}, \tilde{\mathbf{x}}_i) = \exp\left(-\frac{\|\mathbf{x} - \tilde{\mathbf{x}}_i\|^2}{2r_l^2}\right),$$

where $r_l > 0$ is the shape parameter (the kernel bandwidth) at level l.

3. The damped cover tree

This work introduces a data-driven sub-sampling method that we call the damped cover-tree (DCT). The DCT is a modification of the cover-tree (CT) [24], a data structure based on partitioning a metric space, initially designed to facilitate nearest neighbor search. The goal of the DCT is to modify and simplify the CT to allow a viable sub-sampling scheme.

Let $(\mathcal{X}, \|\cdot\|)$ be a normed space where the input domain $\mathcal{X} \subset \mathbb{R}^D$ is bounded, such that the diameter $r_0 = \mathrm{diam}(\mathcal{X})$ is finite. The DCT is a tree structure where each node p of the tree is associated with a point $\mathbf{x}_p \in \mathcal{X}$, and which is built sequentially as data points arrive. Furthermore, let Q_l be a set (herein called a cover-set) containing all the nodes at a level $l \geq 0$ in the given tree. A level is associated with an integer l and a radius $r_l = 2^{-l} r_0$, where $l = 0$ denotes the root level containing only one node and l increases as we descend deeper into the tree. DCT has three invariants, of which the first two are also invariants of the CT.

(I1) (Covering invariant) For all $p \in Q_{l+1}$ there exists $q \in Q_l$ such that $\|\mathbf{x}_q - \mathbf{x}_p\| < r_l$.
(I2) (Separation invariant) For all $q, p \in Q_l$ where $\mathbf{x}_q \neq \mathbf{x}_p$, we have $\|\mathbf{x}_q - \mathbf{x}_p\| > r_l$.

We add that the standard CT includes a third invariant, the so-called nesting invariant, which requires $Q_l \subseteq Q_{l+1}$, but this is not desired for our purpose.

To introduce the last invariant of the DCT, we first need the following definition.

Definition 3.1 (The covering fraction). Let $p \in Q_l$ be a node, and $\mathbf{x}_p$ the associated point in $\mathcal{X}$. Furthermore, let $\widetilde{C}_p = \{c_i\}_{i=1}^{k}$ be the children of p, and $\mathbf{x}_{c_i}$ the corresponding points in $\mathcal{X}$. The covering fraction of a node p is defined as

$$\mathfrak{cf}(p) = \frac{\mathrm{Vol}\left(\mathcal{X} \cap \mathcal{B}(\mathbf{x}_p, r_l) \cap \bigcup_{c_i \in \widetilde{C}_p} \mathcal{B}(\mathbf{x}_{c_i}, r_{l+1})\right)}{\mathrm{Vol}\left(\mathcal{X} \cap \mathcal{B}(\mathbf{x}_p, r_l)\right)}.$$

The covering fraction is the proportion of the volume of $\mathcal{B}(\mathbf{x}_p, r_l)$ that is covered by balls around its children of half the radius. This quantity is directly related to (I2), which enforces the radius r_l to reduce by a factor of 2 for each new level, starting from an initial radius $r_0 > 0$. The covering fraction allows us to capture the vulnerability of the standard CT to the curse of dimensionality.

For example, consider two regions $\mathcal{A}_1, \mathcal{A}_2 \subseteq \mathcal{X}$, for which the doubling dimension at radius r_l is $\mathrm{ddim}(\mathcal{A}_1, r_l) > \mathrm{ddim}(\mathcal{A}_2, r_l)$. A node $p \in \mathcal{A}_1$ at level l will then need exponentially more children to be covered, than a node $q \in \mathcal{A}_2$ at the same level l. This exacerbates the deeper we go into the tree. Therefore, the CT would have significantly more nodes from regions where the doubling dimension is large.

We recall now that sub-sampling is in kernel methods intended to reduce the computational complexity. For this purpose, it is desirable to keep the number of sub-samples from each level within a budget of reasonable size. On the other hand, a too low sub-sample density will lead to poor interpolation performance. Due to the exponential growth of the number of nodes with respect to the doubling dimension, it would be desirable to avoid wasting our budget on sub-samples from regions and radii with a large doubling dimension, as this would require dedicating an (exponentially) large number of points to achieve good interpolation, which is not feasible. Moreover, in high dimensional regions, we likely cannot learn anything more than a simple function, for which a lower sampling density would suffice.

To reduce the number of sub-samples from regions of large doubling dimensionality, we introduce the following damping invariant as the third invariant of the DCT.

(I3) (**Damping invariant**) Let $\mathcal{D}_{\mathfrak{cf}} \in (0, 1)$ be some threshold and let $\widetilde{C}_p$ and $\mathfrak{cf}(p)$ be as in Def. 3.1. Then any node q whose parent node p does not satisfy $\mathfrak{cf}(p) \geq \mathcal{D}_{\mathfrak{cf}}$ does not have children of its own.

The damping invariant forces the tree to devote more resources to regions of lower doubling dimension by making it harder for nodes in regions with higher doubling dimensions to have children. In other words, the practical effect of the damping invariant is to stop the vertical growth of the DCT if the doubling dimension becomes large. This is because the covering number grows exponentially with the dimensionality, ensuring $\mathfrak{cf}(p) \geq \mathcal{D}_{\mathfrak{cf}}$ gets correspondingly harder to achieve.

Remark 3.2. In Section 5.1, we analyze the damping invariant in more detail and show how the damping suppresses vertical growth of the DCT more for regions of high doubling dimension than for regions of lower doubling dimensionality.

3.1. Construction of the DCT

We now discuss how the DCT is constructed and updated as the data streams in. First, it is important to restate that we use the DCT to replace the Nyström sampling, which was in **FALKON** used to reduce the complexity of the ridge regressor. Consequently, not all of the streamed data (that is, not every training point) will be added to the tree, but only those whose inclusion into the tree would not violate the invariants (I1)-(I3). In other words, the tree consists of only those training points that help resolve the data space at the relevant resolution level. Thus, each node p in the DCT is associated with a unique training sample $\mathbf{x}_p$, but not every training sample will be represented by a node in the tree. Note that this is different from the standard CT, which aims to organize all of the training data into a geometrical leveled data structure.

The construction of the DCT consists of a series of checks which examine whether adding a given data point to the DCT would, or would not, violate invariants (I1)- (I3). When a new point $\mathbf{x}_q$ arrives from the data stream the goal is to identify the deepest level l for which there exists a node p such that $\|\mathbf{x}_q - \mathbf{x}_p\| \leq r_l$. This corresponds to finding the nearest node in the tree that could serve as a parent.

We achieve this in the following way. The first training point is identified as the root node to which we associate the radius r_0. For each new point, we proceed in a top-down manner, starting from the root node[3]. We then check whether $\mathbf{x}_q$ would violate the separation invariant at the next level. In other words, if there exists a node p such that $\|\mathbf{x}_q - \mathbf{x}_p\| < r_l$. If such a node does not exist, then $\mathbf{x}_q$ is added to the set of children of the root node, and we update the covering fraction estimate for the root node. Otherwise, if such a node does exist, we repeat the process, checking the separation invariant among the children of the corresponding node, and proceed further down the tree.

Assume we arrived to a node p at level $l \geq 1$, and we have $\|\mathbf{x}_q - \mathbf{x}_p\| \leq r_l$. We then check if p is allowed to have children, that is if the damping invariant is satisfied. If it is not satisfied, the point $\mathbf{x}_q$ is dismissed (it is not added to the tree). On the other hand, if p is allowed to have children, we check whether the separation invariant holds, i.e., if there exists a child c of the node p such that $\|\mathbf{x}_q - \mathbf{x}_c\| < r_{l+1}$. If that were the case, the separation invariant would be violated, and the recursion is applied again by considering c as the potential parent node. However, if such a child does not exist, that is, if the separation invariant is not violated, then $\mathbf{x}_q$ is added to the set of children of the node p. More details are given in Alg. A.1.

Some comments are needed to elucidate how are the steps described above applied in practice. First, note that the covering fraction from Def. 3.1 cannot be calculated explicitly, since the volume terms require knowing the intrinsic dimensionality. Therefore, it is necessary to use an estimator instead. For this purpose, we interpret $\mathfrak{cf}(p)$ as the probability that a sample $\mathbf{x} \sim \text{Uni}(\mathcal{B}(\mathbf{x}_p, r))$ will be within $\mathcal{B}_c := \bigcup_{c_i \in \widetilde{C}_p} \mathcal{B}(\mathbf{x}_{c_i}, r/2)$, where $\widetilde{C}_p$ are the children of p. This probability can be estimated by considering the checks of the separation invariant (I2), conducted on the last N points that were inside $\mathcal{B}(\mathbf{x}_p, r)$, as a series of independent random trials. We use the following running average as an estimator of the covering fraction

$$(\mathfrak{cf}(p))_t = (1 - \alpha)(\mathfrak{cf}(p))_{t-1} + \alpha \mathbb{1}_{\mathcal{B}_c}(\mathbf{x}_t), \tag{3.1}$$

where $\mathbb{1}_{\mathcal{B}_c}(\mathbf{x}_t)$ is the indicator function, and $\alpha > 0$ is a weighting parameter. This approximates a weighted average of the outcome of the N last draws (cf. Appendix B). Note that this reduces the memory requirements, since instead of storing N trial outcomes for each node in the tree, as required had we used an average of the last N trials, we store only a single value for each node in the tree.

Second, the separation invariant is in practice too strict since it results in too few points added to the tree, and thus a worse kernel estimator. Moreover, checking the separation invariant adds to the computational complexity. Therefore, we

---

[3] We assume that all new points $\mathbf{x}_q$ are within a ball of radius r_0 around this node, which holds for a large enough r_0

introduce the following relaxation. Assume we have a new point $\mathbf{x}_q$ and arrived at a node p at level l. We then first conduct a random Bernoulli trial, with the failure probability

$$q_\mathbf{x} = \frac{1}{1 + \exp\left[h \tan\left(\pi\left(\|\mathbf{x}_q - \mathbf{x}_p\|/r_l - \frac{1}{2}\right)\right)\right]}, \tag{3.2}$$

where h is the hardness of the threshold. In other words, the probability of failure is proportional to the distance between $\mathbf{x}_q$ and $\mathbf{x}_p$ - the larger the distance, the more likely the failure. If the trial's outcome is a failure, then the check for the separation invariant is ignored, and the algorithm continues. If it is a success, we proceed by first checking the separation invariant. This means that the probability to ignore the separation invariant increases as $\mathbf{x}_q$ gets farther from $\mathbf{x}_p$.

3.2. Sub-sampling from the DCT

We now discuss how the DCT is used for sub-sampling the training points. By organizing the training points into cover-sets Q_l the DCT allows a hierarchical sub-sampling. Even though cover-sets Q_l significantly reduce the number of training points, they are for practical purposes still too large for efficient sub-sampling. Due to this, we restrict ourselves to a subset $\widetilde{\Gamma}^{(l)} \subseteq Q_l$ of candidate sub-samples called landmarks.

Definition 3.3 (Landmarks). Let Q_l be the cover-set at level l in a DCT. We define the set of candidate landmarks at level l as $\widetilde{\Gamma}^{(l)} = \{\mathbf{x}_p \mid p \in Q_l \text{ and } \mathfrak{cf}(p) \geq \mathcal{D}_{\mathfrak{cf}}\}$, and the set of landmarks (of size m) as any subset $\widetilde{\Gamma}_m^{(l)} = \{\tilde{\mathbf{x}}_1^{(l)}, \ldots, \tilde{\mathbf{x}}_m^{(l)}\} \subset \widetilde{\Gamma}^{(l)}$ of size m.

Some remarks are in order. First, by Def. 3.3, candidates for landmarks at level l are only those nodes allowed to have children (according to the damping invariant (I3)). This design choice implies that the set of candidate landmarks will contain more points from regions with a lower doubling dimension than points from regions with a higher doubling dimension. This is because the larger the doubling dimension is, the more children nodes are needed to cover a given parent node.

Second, Def. 3.3 suggests using only a subset of candidate landmarks as sub-samples. We refer to a result from [2] which states that good statistical accuracy of the estimator is achieved if the number of sub-samples is proportional to the square root of the number of samples. At level l we therefore use a set of landmarks which is of size $m^{(l)} = \delta_0 \sqrt{|Q_l|}$, where $\delta_0 > 0$ is a constant.

The third point that requires attention concerns the question of when the landmarks should be selected. To that end, we use the covering fraction of a level, which, with a slight abuse of notation, we denote as $\mathfrak{cf}(Q_l)$. Moreover, we compute $\mathfrak{cf}(Q_l)$ as

$$(\mathfrak{cf}(Q_l))_t = (1 - \alpha)(\mathfrak{cf}(Q_l))_{t-1} + \alpha \mathbb{1}_{\mathcal{B}_{\text{level}}}(\mathbf{x}_t), \tag{3.3}$$

where $\mathcal{B}_{level} = \bigcup_{p \in Q_l} \mathcal{B}(\mathbf{x}_p^{(l)}, r_l)$. Moreover, analogously to the damping invariant, let $\mathcal{D}_{level} \in (0, 1)$ be some threshold. We then say that a level l is sufficiently covered when $\mathfrak{cf}(Q_l) \geq \mathcal{D}_{level}$.

Remark 3.4. We note that as the level increases, our estimate of $\mathfrak{cf}(Q_l)$ through Eq. (3.3) will be increasingly more sensitive to subsets $\mathcal{A} \subset \mathcal{X}$ of low doubling dimension than to subsets of large doubling dimension. This is because the damping invariant (I3) makes it harder for nodes in high dimensions to have children. Consequently, we will have fewer points in deeper levels that belong to high dimensional regions. Because of this, the estimator in Eq. (3.3) is biased towards using more sub-samples from lower dimensional regions.

Sub-sampling from a level l goes as follows. As training points arrive, we build the tree and continuously update the covering fraction of a level. Once that level is sufficiently covered, that is, once $\mathfrak{cf}(Q_l) \geq \mathcal{D}_{level}$, we extract the set of landmarks by sub-sampling $m^{(l)}$ points from the pool of candidate landmarks $\widetilde{\Gamma}^{(l)}$.

4. StreaMRAK

In this section, we present StreaMRAK and clarify how it synthesizes concepts from Sections 2 and 3, and utilizes them in a streaming context. The workflow of StreaMRAK can be divided into three threads that can run in parallel, subject to some inter-dependencies. These are the sub-sampling thread, the training thread, and the prediction thread. Overviews of these threads are given next, and the reader is referred to Algorithm A.2 in the Appendix for further details.

4.1. Sub-sampling thread

In the sub-sampling thread StreaMRAK collects and organizes the training data into a DCT. Namely, as new training pairs are collected, the covering (I1) and separation (I2) are checked, and the covering fraction is updated as described in Section 3.1. Moreover, the set of landmarks for each level is updated, as described in Section 3.2. Once the set of landmarks for a given level $\widetilde{\Gamma}_m^{(l)}$ is completed, the landmarks and the estimator for the corresponding level can be used in the remaining two threads.

4.2. Training thread

The model is trained at level l when two conditions are met. First, coefficients of the previous level $l-1$ in the LP must have been calculated, i.e. previous training thread must finish. Second, landmarks $\widetilde{\Gamma}^{(l)}_{m^{(l)}}$ at level l must be ready.

In the first step, we define the kernel matrix on the landmarks by

$$[\mathbf{K}^{(l)}_{mm}]_{ij} = k^{(l)}(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j), \text{ for } \tilde{\mathbf{x}}_i \in \widetilde{\Gamma}^{(l)}. \tag{4.1}$$

In the second step we consider $(\mathbf{K}^{(l)}_{nm})^\top \mathbf{K}^{(l)}_{nm} \in \mathbb{R}^{m^{(l)} \times m^{(l)}}$ and $(\mathbf{K}^{(l)}_{nm})^\top \mathbf{d}^{(l)}_n \in \mathbb{R}^{m^{(l)}}$ which in addition to landmarks depend on the training points. They are updated continuously as new training points come in, according to Eq. (2.4) and Eq. (2.5). However, they are not updated indefinitely, but only until new training points do not significantly alter the matrices according to the following criterion.

Definition 4.1. (Sufficient training points) Let $\mathbf{A}_n := (\mathbf{K}^{(l)}_{nm})^\top \mathbf{K}^{(l)}_{nm}$, and $\mathbf{b}_n := (\mathbf{K}^{(l)}_{nm})^\top \mathbf{d}^{(l)}_n$. Let $\delta_1, \delta_2, \delta_3 > 0$ be three constants. We consider the number of training points at a level l sufficient when either $n \geq \delta_3$ or

$$\left\| \frac{\mathbf{A}_n}{n} - \frac{\mathbf{A}_{n+1}}{n+1} \right\|_\infty \leq \delta_1 \text{ and } \left\| \frac{\mathbf{b}_n}{n} - \frac{\mathbf{b}_{n+1}}{n+1} \right\| \leq \delta_2.$$

After enough training samples are collected according to Def. 4.1, the correction term $s^{(l)}$ is obtained by solving for the coefficients $\tilde{\alpha}^{(l)}_1, \ldots, \tilde{\alpha}^{(l)}_{m^{(l)}}$ using Eq. (2.10). The new prediction model $\widehat{f}^{(L)}$ is obtained by adding $s^{(l)}$ to the previous model, according to Eq. (2.7).

4.3. Prediction thread

In this thread StreaMRAK makes provides the latest version of the trained LP model in Eq. (2.7). This means that if L is currently the highest level that has been trained, the prediction for new points $\mathbf{x}$ is made using the model $\widehat{f}^{(L)}(\mathbf{x})$.

5. Analysis

In this section, we first analyze the damping invariant of the DCT. We then offer theoretical results on the convergence properties of the LP in the context of KRR. Finally, we offer estimates of the time and memory requirements of StreaMRAK. We introduce two auxiliary results.

Lemma 5.1. *Consider a ball $\mathcal{B}(\mathbf{x}, r) \in \mathbb{R}^D$ and let $\delta > 0$. The number of points in any (discrete) set of points within $\mathcal{B}(\mathbf{x}, r)$ that are at least δ apart, $S = \{\mathbf{x}_i \in \mathcal{B}(\mathbf{x}, r) | d(\mathbf{x}_i, \mathbf{x}_j) \geq \delta \text{ for } i \neq j\}$, is bounded by $|S| \leq \left(\frac{2r}{\delta} + 1\right)^D$.*

Proof. Since the points in S are at least δ apart, it follows that the balls $\mathcal{B}(\mathbf{x}_i, \delta/2)$ are disjoint. Consider now the ball $\mathcal{B}(\mathbf{x}, r + \delta/2)$. All of the balls $\mathcal{B}(\mathbf{x}_i, \delta/2)$ are entirely contained within $\mathcal{B}(\mathbf{x}, r + \delta/2)$. Since the balls $\mathcal{B}(\mathbf{x}_i, \delta/2)$ are disjoint, it follows that

$$|S| \leq \text{Vol}\left(\mathcal{B}(\mathbf{x}, r + \delta/2)\right) / \text{Vol}\left(\mathcal{B}(\mathbf{x}_i, \delta/2)\right) = \left(\frac{2r}{\delta} + 1\right)^D.$$

$\square$

Lemma 5.2. *Consider a domain $\mathcal{X} \in \mathbb{R}^D$, a ball $\mathcal{B}(\mathbf{x}_p, r) \subset \mathcal{X}$ and let $S = \{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{B}(\mathbf{x}_p, r) | \|\mathbf{x}_i - \mathbf{x}_j\| \geq \delta \text{ for } i \neq j\}$. Furthermore, let the doubling dimension of the set S be $\texttt{ddim} := \texttt{ddim}(S, r)$. We let $c_d := |S|$ when $\mathfrak{cf}(p) = 1$. We then have $2^{\texttt{ddim}-1} \leq c_d \leq 5^{\texttt{ddim}}$.*

Proof. The upper bound on c_d follows from Lemma 5.1 with $r = r_0$ and $\delta = r_0/2$. The lower bound follows from the definition of the doubling dimension 1.2. $\square$

5.1. Analysis of the DCT

As discussed in Section 3, the DCT adds a given training point to the set of nodes of the tree if conditions (I2) and (I3) are satisfied, and the points are otherwise discarded. In particular, the damping invariant (I3) makes it harder for a node to have children. The guiding idea is that damping should reduce the impact of the curse of dimensionality by making it harder for nodes in regions of higher doubling dimension to have children, and in doing so it should effectively stop the vertical growth of the tree in corresponding regions. Therefore, it is critical to understand how and to what degree the damping affects high dimensional regions more than low dimensional ones.

In a statistical sense, the damping should treat all nodes in regions of the same doubling dimension equally. Therefore, to gain insight into the damping, it suffices to analyze its effects concerning the doubling dimension on a single node p. In this case, the effect of damping can be measured by analyzing how many training points must pass through p, in the sense of Alg. A.1, before children of p are allowed to have children of their own. This can be modeled by considering the expected number of training points $\mathbf{x}_i \sim \text{Uni}(\mathcal{B}(\mathbf{x}_p, r))$ necessary to cover $\mathcal{B}(\mathbf{x}_p, r)$ with balls of radius $r/2$ around points $\mathbf{x}_i$.

Consider $\mathbf{x}_i \sim \text{Uni}\big(\mathcal{B}(\mathbf{x}_p, r)\big)$, and let a set $\mathcal{S}_p$ be built in a succession of trials $i = 1, \ldots, N_t$ so that

$$\mathbf{x}_i \in \mathcal{S}_p \text{ if } \|\mathbf{x}_i - \mathbf{x}\| \geq \frac{r}{2} \text{ for all } \mathbf{x} \in \mathcal{S}_p.$$

In other words, a newly sampled point $\mathbf{x}_i$ will only be added to the set $\mathcal{S}_p$ if its pairwise distances from all the points that are already in $\mathcal{S}_p$ are at least $r/2$.

Problem 1. Let $\widetilde{C}_p$ denote the set of children of the node p, constructed from the above-described trials. What is the expected number of trials N_t needed to ensure $\mathfrak{cf}(p) = 1$?

Since there is no unique set $\mathcal{S}_p$ such that the corresponding set of children $\widetilde{C}_p$ ensures $\mathfrak{cf}(p) = 1$, the sample space for Problem 1 corresponds to all admissible sets $\mathcal{S}_p$, which vary in both the number and the location of points they contain. Characterizing all such sets corresponds to a disordered sphere packing problem [56], which is an NP-hard combinatorial problem [57]. For a theoretical analysis of this problem, defining a probability measure over the sample space is necessary. However, in this level of generality, neither the sample space nor the probability measure admit a workable definition, with currently available mathematical tools [56]. Although some theoretical insights are possible under simplifications on the sample space, this analysis is restrained to a limited number of spheres and configurations.

Due to these difficulties, we consider a simplified setting where we instead consider an average case. If the set $\mathcal{S}_p$ is such that $\mathcal{B}(\mathbf{x}_i, r) \subset \bigcup_{\mathbf{x}_i \in \mathcal{S}_p} \mathcal{B}(\mathbf{x}_i, r/2)$, which corresponds to $\mathfrak{cf}(p) = 1$, then each of the balls $\mathcal{B}(\mathbf{x}_i, r/2)$ occupies on average $\frac{1}{|\mathcal{S}_p|}$ of the total volume of $\mathcal{B}(\mathbf{x}_p, r)$, assuming none of the balls are covered by a union of other balls. Therefore, as $\mathcal{S}_p$ is being built, adding a point to $\mathcal{S}_p$ will, on average, reduce the unoccupied volume of $\mathcal{B}(\mathbf{x}_p, r)$ by $\frac{1}{|\mathcal{S}_p|}$. Moreover, it can be shown that the number of elements in such a set satisfies $2^{\text{ddim}-1} \leq |\mathcal{S}_p| \leq 5^{\text{ddim}}$, see Lemma 5.2, where $\text{ddim} := \text{ddim}(\mathcal{S}_p, r)$ is the doubling dimension of $\mathcal{S}_p$. Based on these considerations we introduce a simplified setting for the average case of Problem 1.

Assumption 1. Problem 1 can be approximated by dividing the ball $\mathcal{B}(\mathbf{x}_p, r)$ into a union of c_d fixed (and known) disjoint bins $\mathcal{B}_i$ of size $(1/c_d) \text{Vol}\big(\mathcal{B}(\mathbf{x}_p, r)\big)$.

Note that the bins referred to in Assumption 1 correspond to regions around the children of the node p. Assumption 1 reduces the average case of Problem 1 to a form of the classical coupons collector's problem [58], which considers n coupons with the same probability of being drawn. Through a series of randomized trials with replacement, the goal is to obtain a copy of each coupon. Relevant for Problem 1 is estimating the stopping time T, which counts the number of trials before all coupons are collected, and which satisfies $\mathbb{E}[T] = nH_n$, where n denotes the number of coupons and H_n is the n-th harmonic number [58].

In terms of Problem 1, and under Assumption 1, we can therefore identify $T = N_t$, $n = |\mathcal{S}_p|$ and $\mathbb{E}[N_t | \text{Node } p] = |\mathcal{S}_p| H_{|\mathcal{S}_p|}$. Combining the bound $\ln(n) + \frac{1}{2} \leq H_n \leq \ln(n) + 1$ (from [59]), with the bound on $|\mathcal{S}_p|$ from Lemma 5.2 we have

$$2^{\text{ddim}-1}((\text{ddim} - 1)\ln 2 + 1/2) \leq \mathbb{E}[N_t | \text{Node } p] \leq 5^{\text{ddim}}(\text{ddim}\ln 5 + 1). \tag{5.1}$$

With the same strategy, we can bound the number of trials until the cover-fraction of a level reaches 1, as

$$2^{l(\text{ddim}-1)}(l(\text{ddim} - 1)\ln 2 + 1/2) \leq \mathbb{E}[N_t | \text{Level } l] \leq 5^{l\text{ddim}}(l\text{ddim}\ln 5 + 1). \tag{5.2}$$

From Eq. (5.1) we see that the number of training points $\mathbb{E}[N_t | \text{node } p]$ grows exponentially with the doubling dimensionallity d. In other words, significantly more trials are needed to achieve $\mathfrak{cf}(p) = \mathcal{D}_{\mathfrak{cf}}$ for nodes in regions with a large doubling dimension than it is for nodes in regions with a lower doubling dimension. Consequently, through the damping invariant, the DCT restricts the vertical growth of the tree comparatively more the higher the doubling dimension of the local region.

5.2. Time and memory requirements

This section analyzes the memory requirements of StreaMRAK, which involve storing the DCT and the linear system components used to update the coefficients. Furthermore, we consider the computational requirements, which consist in solving the coefficient equations. Both the memory and computational requirements need to be analyzed per level l of the tree due to the multi-resolution nature of the estimator and the tree organization of the data.

For the analysis, we consider a simplified setting where we assume that the doubling dimension is constant for all levels and all subsets of $\mathcal{X}$, and that the number of children c_d is the same for all nodes. At the end of the section we describe a more general setting.

In the following, we assume that the growth of the DCT stops at a level L. In other words, level L is the last level at which there are nodes. In practice, the growth of the DCT slows down exponentially fast with the product of the doubling dimension $\text{ddim} := \text{ddim}(\mathcal{X}, r_L)$ and the level l. This can be seen from Eq. (5.2), which shows that the number of training points necessary to fill up a level grows exponentially with $l\text{ddim}$. Therefore, in practice, no new levels will be added to the DCT when $l\text{ddim}$ is large enough, which effectively makes the last level L independent of the number of training points.

Furthermore, from Lemma 5.2 we know that c_d is bounded by $2^{\mathtt{ddim}-1} \leq c_d \leq 5^{\mathtt{ddim}}$, which shows that also c_d is independent of the number of training points.

Proposition 5.3. *The memory requirement of StreaMRAK is* $\mathcal{O}\big(\sum_{l=0}^{L} c_d^l\big)$.

Proof. The memory requirement of the DCT is determined by the number of nodes in the tree. Given that the number of children is the same for all nodes. If the number of children per node is c_d, then the total number of nodes at level l is c_d^l. Thus, the memory needed to store the DCT with L levels is $\mathcal{O}(\sum_{l=0}^{L} c_d^l)$.

To store the linear system on level l we need the matrices $\big(\mathbf{K}_{nm^{(l)}}\big)^{\top}\mathbf{K}_{nm^{(l)}}$, $\mathbf{K}_{m^{(l)}m^{(l)}} \in \mathbb{R}^{m^{(l)} \times m^{(l)}}$ and the vector $\mathbf{z} \in \mathbb{R}^{m^{(l)}}$. The number of landmarks $m^{(l)}$ at level l is chosen as $m^{(l)} = \delta_0 \sqrt{|Q_l|}$, where $|Q_l|$ is the number of nodes at level l. Since $|Q_l|$ is $\mathcal{O}(c_d^l)$, it follows that $m^{(l)} \times m^{(l)}$ is also $\mathcal{O}(c_d^l)$ per level, and the desired conclusion follows. $\square$

Note that with a fixed L and n larger than $\mathcal{O}\big(\sum_{l=0}^{L} c_d^l\big)$, then the memory requirement is independent of n. We also note that if the deepest level satisfies $L \to \infty$, then the number of nodes is determined by the number of training points, and the memory requirement would thus, in the worst case, become $\mathcal{O}(n)$, the same as for the standard cover-tree.

Next, we discuss the construction of the DCT, where adding a new point to the set of nodes requires a search through the tree.

Proposition 5.4. *Inserting a new point into the DCT, cf. Algorithm A.1, requires* $\mathcal{O}(c_d L)$ *operations.*

Proof. For a point $\mathbf{x}_q \in \mathcal{X}$ to be analyzed at level L, we need to have analyzed it at the previous $l < L$ levels. At each level, we must, in the worst case, check the separation invariant with all children of the current potential parent $p^{(l)}$, before finding a node c such that $\|\mathbf{x}_q - \mathbf{x}_c\| \leq 2^{-l} r_0$, that would serve as the next potential parent. This requires at most c_d operations per level, giving $L c_d$ total operations over the L levels. The same number of operations is necessary if a node is discarded at level L. $\square$

Lastly, we analyze the computational requirements for solving the linear system.

Proposition 5.5. *The time requirement for solving the linear system in Eq. (2.3) is* $\mathcal{O}\big(\delta_3 m^{(l)} + \big(m^{(l)}\big)^3\big)$ *per level, where δ_3 is given in Def. 4.1,*

Proof. The time requirement of FALKON is $\mathcal{O}(nmt + m^3)$ where n is the number of training points, m the number of landmarks and t the number of iterations of the conjugate gradient (which has an upper bound). By Def. 4.1, StreaMRAK uses at most δ_3 training samples at each level. Since $m^{(l)}$ is the number of landmarks at level l, the result follows. $\square$

Assume that the domain $\mathcal{X}$ can be divided into disjoint subsets $\mathcal{A}_1, \ldots, \mathcal{A}_t \subset \mathcal{X}$ for which the doubling dimension $\mathtt{ddim}(\mathcal{A}_i, r_l)$ differs based on $\mathcal{A}_i$ and radius r_l. Let the number of children of a node $\mathbf{x}_p \in \mathcal{A}_i$ at level l be $c_{d,i,l}$. In this scenario, the growth of the DCT will stop at different levels L_i for different subsets $\mathcal{A}_i$. The final time and memory requirements would therefore be the sum of the contribution from each subset $\mathcal{A}_i$. In other words, the memory would be $\mathcal{O}(\sum_{i=1}^{t} \sum_{l=0}^{L_i} c_{d,i,l}^l)$, and similarly the time requirement per point insertion would be $\mathcal{O}(\sum_{i=1}^{t} \sum_{l=0}^{L_i} c_{d,i,l})$. We note that $c_{d,i,l}$ and L_i depend on the dimensionality of the data, but are independent of n. Therefore, so are the time and memory requirements.

5.3. Convergence of the LP formulation of the KRR

This section analyzes the conditions for which the LP approximates the training data $y_i = f(\mathbf{x}_i)$, with respect to the number of levels. A similar analysis was previously done for the LP in the context of kernel smoothers [28]. However, to the best of our knowledge, this is the first time the LP formulation of KRR has been analyzed in this way.

Theorem 5.6. *Let $\widehat{f}^{(l)}$ be the LP estimator defined in Eq. (2.7) and let λ be a regularization parameter. Furthermore, let $0 < \sigma_{l,n} \leq \cdots \leq \sigma_{l,1}$ be the eigenvalues of $\mathbf{K}_{nn}^{(l)}$. For $L > 0$ we then have*

$$\|\widehat{f}^{(L+1)}([\mathbf{x}_n]) - f([\mathbf{x}_n])\| \leq \prod_{l=0}^{L}(1 - \varepsilon(l))\|\widehat{f}^{(0)}([\mathbf{x}_n]) - f([\mathbf{x}_n])\|, \quad \text{where} \quad \varepsilon(l) = \frac{\sigma_{l,n}}{n\lambda + \sigma_{l,n}}.$$

Proof. From the recurrence relationship for the residuals $\mathbf{d}^{(l)}$ in Eq. (2.9) it follows by induction that

$$\widehat{f}^{(l+1)}([\mathbf{x}_n]) - f([\mathbf{x}_n]) = (\mathbf{I} - \mathbf{P}_{nn}^{(l)})(\widehat{f}^{(l)}([\mathbf{x}_n]) - f([\mathbf{x}_n]), \tag{5.3}$$

where $\mathbf{P}_{nn}^{(l)} := \mathbf{K}_{nn}^{(l)}(\mathbf{K}_{nn}^{(l)} + \lambda n \mathbf{I})^{-1}$, cf. Lemma C.1. It follows that

$$\|\widehat{f}^{(l+1)}([\mathbf{x}_n]) - f([\mathbf{x}_n])\| \leq \|\mathbf{I} - \mathbf{P}_{nn}^{(l)}\| \|\widehat{f}^{(l)}([\mathbf{x}_n]) - f([\mathbf{x}_n])\|. \tag{5.4}$$

Consider the SVD $\mathbf{K}_{nn}^{(l)} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^{\top}$ where $\mathbf{\Sigma} = \mathrm{diag}\,(\sigma_{l,i})$ and $\sigma_{l,n} \leq \cdots \leq \sigma_{l,1}$. We then have

$$\|\mathbf{I} - \mathbf{P}_{nn}^{(l)}\| = \left\|\mathbf{U}\,\mathrm{diag}\left(\frac{n\lambda}{n\lambda + \sigma_{l,i}}\right)\mathbf{U}^{\top}\right\| = \left\|\mathrm{diag}\left(\frac{n\lambda}{n\lambda + \sigma_{l,i}}\right)\right\| = \frac{n\lambda}{n\lambda + \sigma_{l,n}} := 1 - \varepsilon(l), \tag{5.5}$$

and Thm. 5.6 follows recursively from Eq. (5.4) and Eq. (5.5). $\quad\square$

From Thm. 5.6 it follows that the LP estimator will converge as $l \to \infty$, since $\sigma_{l,n} > 0$ and therefore $1 - \varepsilon(l) \in (0, 1)$ for all l. In Thm. 5.7 we characterise how $\varepsilon(l)$ depends on the level l to give insight on the nature of this convergence.

Theorem 5.7. *The LP estimator $\widehat{f}^{(l)}$ from Eq. (2.7) converges with increasing level L to the training data $f(\mathbf{x}_i)$, cf. Thm. 5.6, with the rate $\prod_{l=0}^{L}(1 - \varepsilon(l))$, where*

$$1 - \varepsilon(l) \leq \left(1 + C_{1,D}2^{-Dl}\exp\left(-C_{2,D}4^{-l}\right)/n\lambda\right)^{-1}, \tag{5.6}$$

for

$$C_{1,D} = \frac{1}{2}(6\sqrt{2})^{D}\Gamma(D/2 + 1)^{\frac{D-1}{D+1}}\left(\frac{\pi}{9}\right)^{\frac{D}{D+1}}\left(\frac{r_0}{\delta}\right)^{D} \quad \text{and} \quad C_{2,D} = 1152\left(\frac{\pi\,\Gamma^2(D/2 + 1)}{9}\right)^{\frac{2}{D+1}}\left(\frac{r_0}{\delta}\right)^{2},$$

where Γ is the gamma function.

Furthermore, for $l > \log_2(\sqrt{D/2}(r_0/\delta))$ we have the tighter bound

$$1 - \varepsilon(l) < \left(1 + \left(1 - 2^{1 + \frac{1}{\ln 2}(C_3D - g(l))}\right)/n\lambda\right)^{-1}, \tag{5.7}$$

where $g(l) = 4^{l - \log_2 r_0/\delta}$ and $C_3 = (\ln(1 + 1/4) + 2\ln 2)$.

Proof. (Eq. (5.6)) We bound $1 - \varepsilon(l) := n\lambda/(n\lambda + \sigma_{l,n})$ by bounding the smallest eigenvalue of the kernel matrix $[\mathbf{K}_{nn}^{(l)}]_{ij} = \Phi(\|\mathbf{x}_i - \mathbf{x}_j\|)$, namely $\sigma_{l,n}$. To do so we assume that there exists a lower bound on the minimal distance between any two points $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$, defined as $\delta := \min_{i \neq j \in \mathcal{X}} \|\mathbf{x}_i - \mathbf{x}_j\| > 0$.

Consider the Gaussian $\Phi(\mathbf{x}) = \exp(-\beta\|\mathbf{x}\|_2^2)$, $\beta > 0$, with the Fourier transform $\widehat{\Phi}(\omega) = (\pi/\beta)^{D/2}\exp(-\|\omega\|_2^2/4\beta)$. From [60, Corollary 12.4] we have the bound

$$\sigma_{l,n} \geq C_D 2^D(2\beta)^{-D/2}\delta^{-D}\exp(-4M_D^2/(\delta^2\beta)),$$

where

$$M_D = 12\left(\frac{\pi\,\Gamma^2(D/2 + 1)}{9}\right)^{1/(D+1)} \quad \text{and} \quad C_D = \frac{1}{2\Gamma(D/2 + 1)}\left(\frac{M_D}{2^{3/2}}\right)^{D}.$$

With $\beta = (\sqrt{2}2^{-l}r_0)^{-2}$ and inserting for M_D and C_D we then have

$$\begin{aligned}
\sigma_{l,n} &\geq C_D 2^D 2^{-Dl}\left(\frac{r_0}{\delta}\right)^{D}\exp\left(-(2\sqrt{2}M_D)^2(r_0/\delta)^2 4^{-l}\right) \\
&= \tfrac{1}{2}(6\sqrt{2})^{D}\Gamma(D/2 + 1)^{\frac{D-1}{D+1}}\left(\frac{\pi}{9}\right)^{\frac{D}{D+1}}\left(\frac{r_0}{\delta}\right)^{D}2^{-Dl} \\
&\quad \cdot \exp\left(-1152\left(\frac{\pi\Gamma^2(D/2+1)}{9}\right)^{\frac{2}{D+1}}\left(\frac{r_0}{\delta}\right)^2 4^{-l}\right) := B(l),
\end{aligned} \tag{5.8}$$

The bound in Eq. (5.6) follows from this result. $\quad\square$

Proof. (Eq. (5.7)) When the level l becomes sufficiently large, the kernel matrix $\mathbf{K}_{nn}^{(l)}$ becomes diagonally dominant, and we can therefore bound the eigenvalues using Garschgorins Theorem [61, Thm. 1.1], which gives

$$|\sigma_{l,i} - [\mathbf{K}_{nn}^{(l)}]_{jj}| = |\sigma_{l,i} - 1| < \sum_{\substack{q=1, \\ q \neq j}}^{n} |[\mathbf{K}_{nn}^{(l)}]_{jq}| \quad \text{for} \quad i, j \in [n]. \tag{5.9}$$

To find a more explicit bound, we analyze the sum on the right-hand side. Consider a family of annuli $\{R_t\}_{t=0}^{\infty}$ where $R_t = \mathcal{B}(\mathbf{x}_j, 2^{t+1}\delta)\backslash\mathcal{B}(\mathbf{x}_j, 2^t\delta)$. Inspired by [28], we can interpret the right hand side of Eq. (5.9) as a sum over $\{R_t\}_{t=0}^{\infty}$. The entries of $\mathbf{K}_{nn}^{(l)}$ are defined as

$$[\mathbf{K}_{nn}^{(l)}]_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2r_l^2}\right), \quad \forall i, j \in [n],$$

where $r_l = 2^{-l}r_0$ for $r_0 > 0$. It follows

$$\sum_{\substack{q=1, \\ q \neq j}}^{n} |[\mathbf{K}_{nn}^{(l)}]_{jq}| = \sum_{t=0}^{\infty}\sum_{\mathbf{x}_q \in R_t} k^{(l)}(\mathbf{x}_j, \mathbf{x}_q) \leq \sum_{t=0}^{\infty}\left(\frac{2^{t+2}\delta}{\delta} + 1\right)^{D}\exp\left(-(2^t\delta 2^{-1/2}r_l^{-1})^2\right),$$

where in the first term on the right-hand side we bound the number of summands using Lemma 5.1, and in the second we use $\|\mathbf{x}_q - \mathbf{x}_j\| \geq 2^t \delta$ for $\mathbf{x}_q \in R_t$. Note now that for all $T \geq 1$ there exists $C_T > 0$ such that $\exp(-r^2) \leq C_T r^{-T}$ holds for all $r > 0$. Such a constant is given by the Lambert W function and satisfies $C_T = \left(\frac{T}{2e}\right)^{T/2}$. Moreover, $2^{t+2} + 1 \leq 2^{t+2+\alpha}$, for $\alpha \geq \ln(1 + 1/4)/\ln(2)$. Thus,

$$\sum_{\substack{q=1, \\ q \neq j}}^{n} |[\mathbf{K}_{nn}^{(l)}]_{jq}| \leq C_T \left(\frac{r_l}{\delta}\right)^T 2^{(2+\alpha)D + T/2} \sum_{t=0}^{\infty} 2^{t(D-T)} \leq 2 \cdot 2^{D(2+\alpha) - T/2(1 + 1/\ln(2))} T^{T/2} \left(\frac{r_l}{\delta}\right)^T,$$

where in the last step we have used $\exp(1) \geq 2^{1+1/\ln(2)}$ along with $\sum_{t=0}^{\infty} 2^{t(D-T)} \leq 2$, which holds for $D - T < 0$. We let $r_l = r_0 2^{-l}$ and define $F(T) := 2^{-T/2(1+1/\ln(2))} T^{T/2} 2^{-lT} \left(\frac{r_0}{\delta}\right)^T$. From Lemma C.4 we have the minimum of $F(T)$, which together with the choice $\alpha = \ln(1 + 1/4)/\ln 2$, gives

$$\sigma_{l,n} > 1 - \sum_{\substack{q=1, \\ q \neq j}}^{n} \left|\left[\mathbf{K}_{nn}^{(l)}\right]_{jq}\right| \geq 1 - 2^{1 + \frac{1}{\ln 2}((\ln(1+1/4) + 2\ln 2)D - g(l))}, \quad g(l) = 4^{l - \log_2 r_0/\delta}.$$

By defining $C_3 = (\ln(1 + 1/4) + 2\ln 2)$ and using that $1 - \varepsilon(l) := n\lambda/(n\lambda + \sigma_{l,n})$ this leads to the bound in Eq. (5.7). We note that this bound holds for $T^* > D$ which means that $l > \log_2(\sqrt{D/2} r_0/\delta)$. $\square$

We note that the bound in Eq. (5.6) underestimates the rate of convergence for lower levels but improves as the levels increase. Furthermore, Thm. 5.7 shows that the convergence rate increases with the level l. In fact, the bound in Eq. (5.6) can be simplified with an *a fortiori* bound of the same form, where $C_{1,D} = \frac{1}{2} \left(\frac{12.76}{2^{3/2}}\right)^D \left(\frac{D^D}{\Gamma(D/2+1)}\right) \left(\frac{r_0}{\delta}\right)^D$ and $C_{2,D} = (12.76\sqrt{2}D)^2 (r_0/\delta)^2$, which ensures that $1 - \varepsilon(l)$ decreases monotonically for $l < \log_2(\sqrt{D/2}(r_0/\delta)) + \log_2(25.52\sqrt{2})$. see Remark C.2 and Corollary C.3.

On the other hand, when $l > \log_2(\sqrt{D/2}(r_0/\delta))$ the tighter bound from Eq. (5.7) ensures that $1 - \varepsilon(l)$ continues to decreases monotonically. Moreover, as $l \to \infty$ each new level reduces the residual error by $(1 + 1/n\lambda)^{-1}$. We can also observe that the convergence rate is reduced by the number of training points n, but this effect can be mitigated by reducing the regularization parameter λ. We also note that Thm. 5.6 and Thm. 5.7 are derived for a vector of numbers on the training data $\Gamma_n \subset \mathcal{X}$, without assumptions on the target function. In other words, the LP estimator can approximate the training data for any function $f : \Gamma_n \to \mathbb{R}$, to arbitrary precision, by including sufficiently many levels.

Corollary 5.8. *If the residual $\mathbf{d}^{(l)} = (\widehat{f}^{(l)}([\mathbf{x}_n]) - f([\mathbf{x}_n]))$ at level l only projects non-trivially onto the eigenvectors with eigenvalue $\sigma_{l,n} \geq \sigma_{\text{cut-off}}$, then we say the residual is spectrally band-limited with respect to the kernel. If the residual $\mathbf{d}^{(l)}$ is spectrally band-limited, then $1 - \epsilon(l) < n\lambda/(n\lambda + \sigma_{\text{cut-off}})$.*

Proof. Follows from Eq. (5.3)-(5.5) with $P_{nn}^{(l)} = P_{nn,k}^{(l)} + (P_{nn,k}^{(l)})^\perp$, where $P_{nn,k}^{(l)}$ is the projection on the eigenvectors associated with the k largest eigenvalues and $(P_{nn,k}^{(l)})^\perp (\widehat{f}^{(l)}([\mathbf{x}_n]) - f([\mathbf{x}_n])) = 0$. $\square$

6. Experiments

This section presents comparative numerical experiments of the proposed estimator on three problems. In Section 6.1 we consider a one-dimensional regression problem, and in Section 6.2 we consider a dumbbell-shaped domain that consists of two 5-dimensional spheres connected by a 2-dimensional plane. Lastly, in Section 6.3, we forecast the trajectory of a double pendulum, which is a well-known chaotic system [25].

We compare StreaMRAK with FALKON [2] and an LP modification of KRR (LP-KRR). Both FALKON and LP-KRR rely on the standard Nyström sub-sampling [51,52]. Furthermore, FALKON does not rely on a multi-resolution scheme but uses instead a single bandwidth, found by cross-validation.

Throughout the experiments, we set the threshold for the number of sub-samples (landmarks) in StreaMRAK to be $10\sqrt{|Q_l|}$, where Q_l is the set of nodes at level l in the DCT. We note that to choose the sub-sample size, FALKON and LP-KRR require n to be known beforehand. For FALKON we let the number of Nyström landmarks be $10\sqrt{n}$, where n is the number of training samples. Meanwhile, for LP-KRR we sub-sample $\sqrt{n}$ Nyström landmarks, which are then used for all levels.

We also need to pre-select the number of training points for LP-KRR and FALKON. For FALKON we use the entire training set, as in [2]. Similarly, it is also common for the LP to use the entire training set at each level [27,28]. However, for large data sets, it might be better to include fewer data points. Therefore, we also use a version of the LP-KRR where we divide the total training data equally between the levels.

Throughout the experiments, we measure the performance of StreaMRAK, FALKON, and LP-KRR by estimating the mean square error

$$MSE(y, y_{pred}) = \frac{1}{\Upsilon \Lambda} \sum_{k=1}^{\Upsilon} \frac{1}{n_k} ||\mathbf{y}_k - \mathbf{y}_k^{pred}||^2, \text{ with } \Lambda = \max_{\substack{k \in [\Upsilon] \\ i \in [n_k]}} [\mathbf{y}_k]_i - \min_{\substack{k \in [\Upsilon] \\ i \in [n_k]}} [\mathbf{y}_k]_i, \tag{6.1}$$

Table 1

Comparison of StreaMRAK, LP-KRR and FALKON for the target in Eq. (6.2). For each level l we show the number of landmarks, the mean square error (MSE), and the accumulated time to train the prediction model (Time). In parenthesis, in the time column of the FALKON row, is the time to find the optimal bandwidth through cross-validation.

	Level	# Landmarks	MSE	Time
StreaMRAK	5	47	2.55×10^{-1}	77 s
	10	392	3.69×10^{-2}	116 s
	15	1525	$8.63 \times 10-6$	497 s
	16	2302	$6.18 \times 10-6$	1194 s
LP-KRR (1) $n_l = 1.1 \times 10^5$	5	1483	2.56×10^{-1}	143 s
	10	1483	3.65×10^{-2}	413 s
	15	1483	$8.72 \times 10-6$	825 s
	16	1483	$6.85 \times 10-6$	922 s
	18	1483	$6.55 \times 10-6$	1136 s
LP-KRR (2) $n_l = 2.2 \times 10^6$	5	1483	2.56×10^{-1}	2963 s
	10	1483	3.64×10^{-2}	8704 s
	18	1483	$8.91 \times 10-6$	23113 s
StreaMRAK	–	14830	5.7×10^{-3}	4642 s+(27930 s)
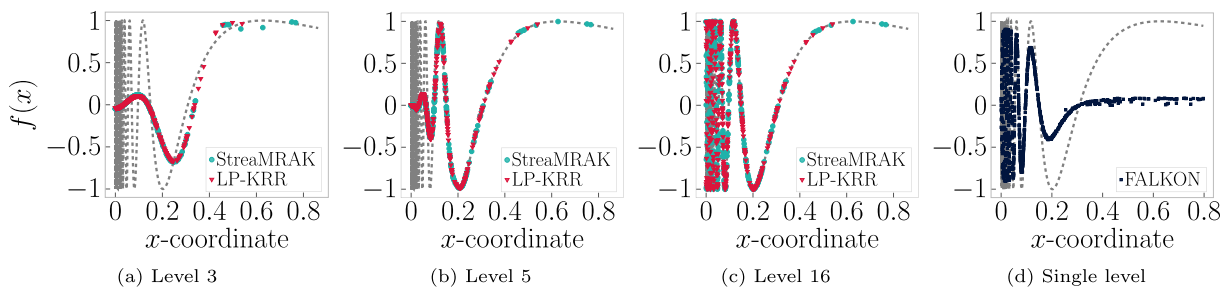
Fig. 3. (a)-(d) shows the target function $f(x)$ from Eq. (6.2) as a grey dotted line. The light-blue circles indicates the predicted values made by StreaMRAK. Similarly the red triangles indicates the predictions made by LP-KRR and the dark blue squares the predictions made by FALKON. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

where Υ is the number of test runs we average over, n_k is the number of test points at test run k, and $\mathbf{y}_k, \mathbf{y}_k^{pred} \in \mathbb{R}^{n_t}$ are the target values and predictions respectively, and Λ is the normalisation factor.

6.1. Multi-resolution benchmark

We consider the function,

$$f(x) = \sin\left(\frac{1}{x + 0.01}\right), \text{ for } x \in \left[0, \frac{\pi}{4}\right]. \tag{6.2}$$

In the experiment we use a training set of $n = 2.2 \times 10^6$ samples and a test set of 1.3×10^5 samples. We use the non-uniform gamma distribution $\Gamma(\alpha, \beta)$ with $\alpha = 1$, $\beta = 2$ to sample the training data.

The number of training points used at each level in StreaMRAK is determined by setting δ_1 and δ_2 from Def. 4.1 to 10^{-3}. With this choice, StreaMRAK selects between 30244 and 40100 training points for each level. For comparison, FALKON uses all the 2.2×10^6 training points. Furthermore, for LP-KRR we run two experiments: LP-KRR (1) using 1.1×10^5 training points at each level and LP-KRR (2) using 2.2×10^6 training points at each level.

Results are presented in Table 1, and the prediction results are illustrated in Fig. 3a-3 d. The results show that StreaM-RAK and both LP-KRR schemes perform much better than FALKON. The reason is that FALKON uses only one bandwidth r, while the multi-resolution schemes StreaMRAK and LP-KRR, utilize a bandwidth regime $r_l = 2^{-l}r_0$ that varies with the level l. The consequence is that StreaMRAK and LP-KRR approximate the low-frequency components of f when the bandwidth is large, and then target the high-frequency components of $f(x)$ gradually as the bandwidth decreases. These results illustrate the benefits of a multi-resolution scheme over a single bandwidth scheme.

From Table 1, we also observe that LP-KRR (2) is significantly slower than StreaMRAK and LP-KRR (1). This is because it uses the entire training set at each level. Therefore, since LP-KRR (1) and LP-KRR (2) achieve comparable precision, we see that including all training points at each level is not always necessary.

A closer comparison of StreaMRAK and LP-KRR is given in Fig. 4. In particular, in Fig. 4a we see that the two algorithms achieve very similar precision. However, comparing the training times in Fig. 4b, we see that StreaMRAK trains each level faster and therefore achieves better precision earlier than LP-KRR (1).
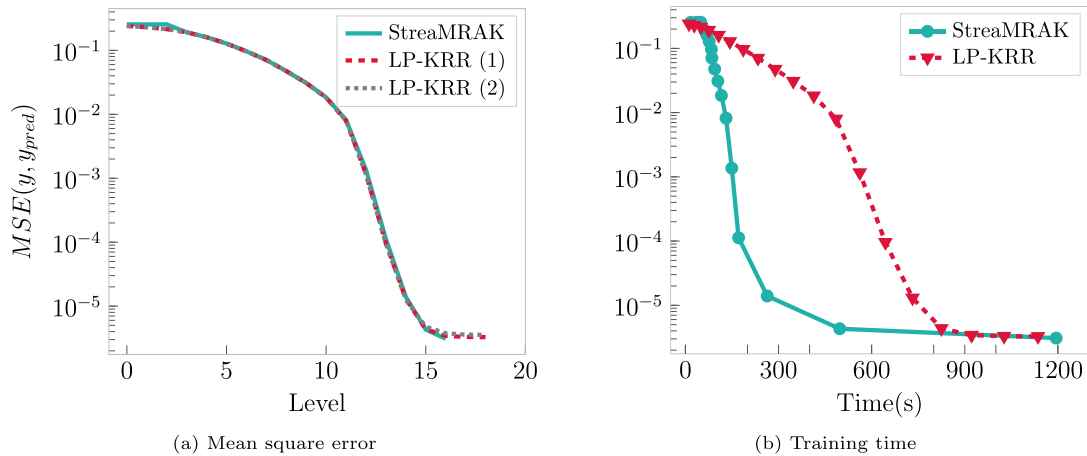
(a) Mean square error (b) Training time

Fig. 4. Comparison of StreaMRAK and LP-KRR. (a) shows the mean square error calculated according to Eq. (6.1) with the target function from Eq. (6.2). Along the x-axis is the number of levels included in the model. (b) The x-axis shows the accumulated training time until a level in the LP is completed. The y-axis shows the MSE of the prediction using the currently available model. The blue circles indicate the prediction error of StreaMRAK and the red triangles indicate the prediction error of LP-KRR (1). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

(a) Level 3 (b) Level 5 (c) Level 16
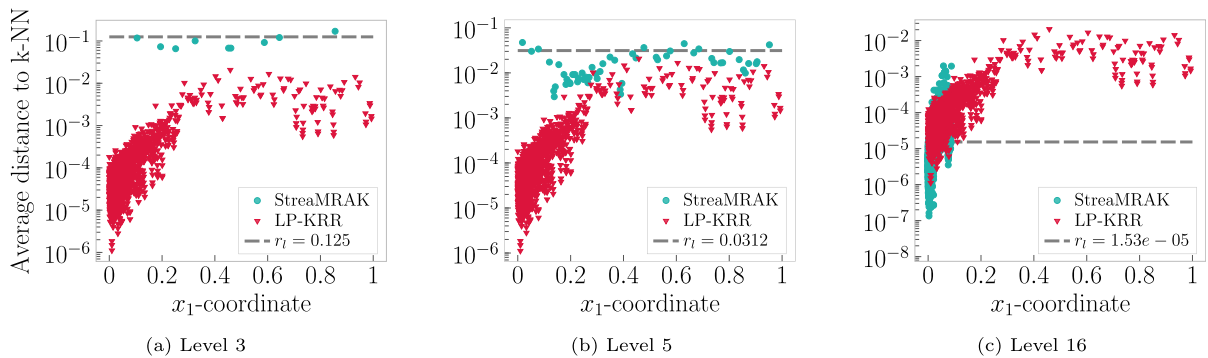
Fig. 5. (a)-(c) shows the landmarks with their position along the x_1 axis and the average distance to their 2 nearest neighbors along the y-axis. Here the red triangles are the Nyström landmarks of LP-KRR and the light-blue circles the landmarks of StreaMRAK. The grey dotted line is the bandwidth at the given level. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

In Fig. 5 we show the average distance of each landmark to their 2 nearest neighbors (2-NN distance). Two aspects of the selection require attention. As opposed to LP-KRR, StreaMRAK selects landmarks such that the 2-NN distance is comparable to the bandwidth used at a specific level. In addition, StreaMRAK saves computational power by not choosing landmarks in regions where the 2-NN distance is too low compared to the bandwidth. In Fig. 5c this can be observed for level $l = 16$ for landmarks with $x \geq 0.2$. Due to the non-uniform sample distribution with a higher density around $x = 0$, the adaptive sub-sampling is able to select more landmarks in the region close to $x = 0$, where f oscillates with high frequency. Furthermore, StreaMRAK stops predicting at level 16 because level 17 is not yet covered with a high enough density of landmarks. Meanwhile, LP-KRR continues, but as seen from Fig. 4a the improvements after level 15 are not significant because the density of Nyström samples is too low compared to the bandwidth.

6.2. Adaptive sub-sampling benchmark

We consider a dumbbell-shaped domain embedded in $\mathbb{R}^5$, consisting of two 5-dimensional spheres connected by a 2-dimensional plane. A projection of the input domain in $\mathbb{R}^3$ is shown in Fig. 7 (a)-(c). Furthermore, as target we consider the following function,

$$f(\mathbf{x}) = \begin{cases} A\sin(Bx_1 + \phi) + (x_1 + 2), & 1 < x_1 < 3 \\ 1, & \text{otherwise} \end{cases}, \text{ for } \mathbf{x} \in [-1, 5] \times [-1, 1]^4, \tag{6.3}$$

where A, B and ϕ are chosen so that $f \in \mathcal{C}^1([-1, 5] \times [-1, 1]^4, \mathbb{R}^5)$. For the experiments, we consider a training set of 1.9×10^6 samples and a test set of 6×10^5 samples, all sampled uniformly at random from the input domain. We note that

Table 2

Comparison of StreaMRAK, LP-KRR, and FALKON predictions of the target function in Eq. (6.3). For each level l we show the number of landmarks, the mean square error (MSE), and the accumulated time to train the prediction model (Time). In parenthesis, in the time column of the FALKON row, is the time to find the optimal bandwidth through cross-validation.

	Level	# Landmarks	MSE	Time
	4	352	1.29×10^{-3}	64 s
StreaMRAK	5	2667	1.27×10^{-3}	1398 s
	6	1858	8.31×10^{-4}	1462 s
	8	1329	2.75×10^{-5}	2307 s
	4	1375	1.28×10^{-3}	386 s
LP-KRR (1) $n_l = 1.8 \times 10^5$	5	1375	1.26×10^{-3}	520 s
	6	1375	9.10×10^{-4}	671 s
	8	1375	3.30×10^{-4}	1064 s
	9	1375	3.10×10^{-4}	1287 s
	4	1375	1.34×10^{-3}	4160 s
LP-KRR (2) $n_l = 1.9 \times 10^6$	5	1375	1.30×10^{-3}	5570 s
	6	1375	9.44×10^{-4}	7168 s
	8	1375	3.16×10^{-4}	11125 s
	9	1375	3.01×10^{-4}	13334 s
FALKON	–	14830	6.8×10^{-4}	6590 s+(37561 s)

we purposefully chose a simple function in the high dimensional regions because complicated functions in high dimensions require far too many points to be satisfactorily learned.

To determine the number of training points for StreaMRAK, we let $\delta_1 = 1 \times 10^{-3}$ and $\delta_2 = 1 \times 10^{-4}$, cf. Def. 4.1. With this choice StreaMRAK selects between 30100 and 40100 training points for each level. FALKON again uses all the 1.9×10^6 training points and for LP-KRR we consider two settings: LP-KRR (1) using 1.8×10^5 training points at each level, and LP-KRR (2) using 1.9×10^6 training points at each level.

The results for StreaMRAK, LP-KRR, and FALKON are presented in Table 2. We observe that StreaMRAK achieves a better prediction than both FALKON and LP-KRR because it adapts the sub-sampling density to the level of resolution.

To understand the improvement in prediction accuracy, we need to discuss the effects of landmark selection. In Fig. 7a-7 c we show the projections of landmarks for StreaMRAK and LP-KRR on $\mathbb{R}^3$, and in Fig. 7d-7 f the average distance of each landmark to its 7 nearest neighbors. These distances are compared with the bandwidth r_l selected for the given level l. We see that StreaMRAK selects landmarks in regions where the average distance to nearest neighbors is comparable to the bandwidth. This means that in high dimensional regions, which correspond to $x_1 \in [-1, 1] \cup [3, 5]$, the algorithm effectively stops collecting landmarks since it cannot maintain high enough density. On the other hand, LP-KRR uses Nystrom subsampling, which imposes a uniform selection of landmarks. Consequently, a significant number of landmarks come from high-dimensional regions.

Moreover, Fig. 7 shows that in the case of LP-KRR, the average distance between the landmarks in high dimensional regions is larger than the bandwidth r_l when $l \geq 5$. As a knock-on effect, LP-KRR makes only small improvements in high dimensional regions for $l \geq 5$, as seen from Fig. 6b. Analogous behavior can be observed for StreaMRAK. However, since StreaMRAK devotes fewer resources to high dimensional regions, it sub-samples more from the low dimensional region, as illustrated in Fig. 6a. The consequence is that StreaMRAK makes bigger improvements in the low dimensional region than LP-KRR, as seen from Fig. 6b. Note that this was not the case in Section 6.1, where the two methods had similar behavior, but unlike here, the input domain in Section 6.1 did not consist of regions with different dimensionalities.

6.3. Forecasting the trajectory of a double pendulum

We consider the double pendulum, illustrated in Fig. 2a, which we model by the Lagrangian system

$$\mathcal{L} = ml^2(\omega_1^2 + \frac{1}{2}\omega_2^2) + ml^2\omega_1\omega_2 \cos(\theta_1 - \theta_2) + mgl(2\cos\theta_1 + \cos\theta_2), \tag{6.4}$$

under the assumption that the pendulums are massless rods of length $l_1 = l_2 = l$ with masses $m_1 = m_2 = m$ centered at the end of each rod. Here g is the standard gravity, $\omega_1 := \dot{\theta}_1$, $\omega_2 := \dot{\theta}_2$ are the angular velocities, and the angles θ_1, θ_2 are as indicated in Fig. 2a. For the experiments we let $m = 1$, $l = 1$ and $g = 10$.

The learning task is to forecast the trajectory of the pendulum, given only its initial conditions. We let $\mathbf{s}_t = [\theta_1(t), \theta_2(t), \omega_1(t), \omega_2(t)] \in \mathbb{R}^4$ be the state of the system at step $t \in \mathbb{N}$ and train StreaMRAK, LP-KRR and FALKON to learn how $\mathbf{s}_t$ maps to a later state $\mathbf{s}_{t+\Delta}$, for $\Delta \in \mathbb{N}$. The trained model $\hat{f}$ is used to forecast the state $\mathbf{s}_T$ for $T >> 0$ by recursively predicting $\mathbf{s}_{t+\Delta} = \hat{f}(\mathbf{s}_t)$ from the initial state $\mathbf{s}_0$ until $t = T$.

For the experiments we consider two settings: a low energy system $\mathbf{s}_0^{low} = [-20°, -20°, 0°, 0°]$ and a high energy system $\mathbf{s}_0^{high} = [-120°, -20°, -7.57°, 7.68°]$. For these systems, we initialize 8000 pendulums as $\mathbf{s}_0 \sim \mathcal{N}(\mathbf{s}, \sigma(\mathbf{s}))$ for $\mathbf{s} = \mathbf{s}_0^{low}, \mathbf{s}_0^{high}$
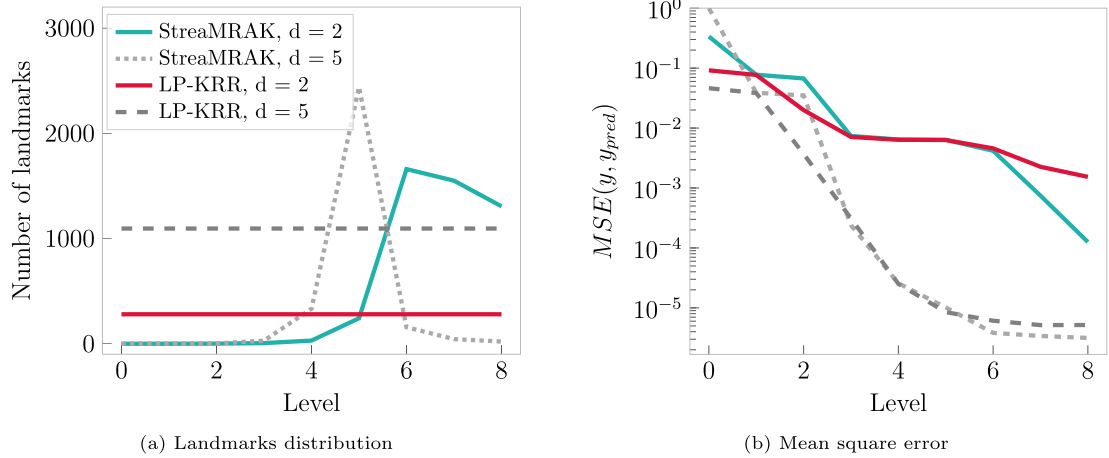
(a) Landmarks distribution   (b) Mean square error

Fig. 6. Comparison of StreaMRAK and LP-KRR (1) in the 2-dim and 5-dim regions of the Dumbbell domain. The solid blue line is StreaMRAK for dimension $d = 2$ while the solid red line is LP-KRR (1) for dimension $d = 2$. The grey dotted line is StreaMRAK for dimension $d = 5$ and the dark-grey dashed line is LP-KRR (1) for dimension $d = 5$ (a) shows the mean square error calculated according to Eq. (6.1). (b) shows the number of landmarks in the 2-dimensional and the 5- dimensional regions. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

(a) Level 4   (b) Level 5   (c) Level 6

(d) Level 4   (e) Level 5   (f) Level 6
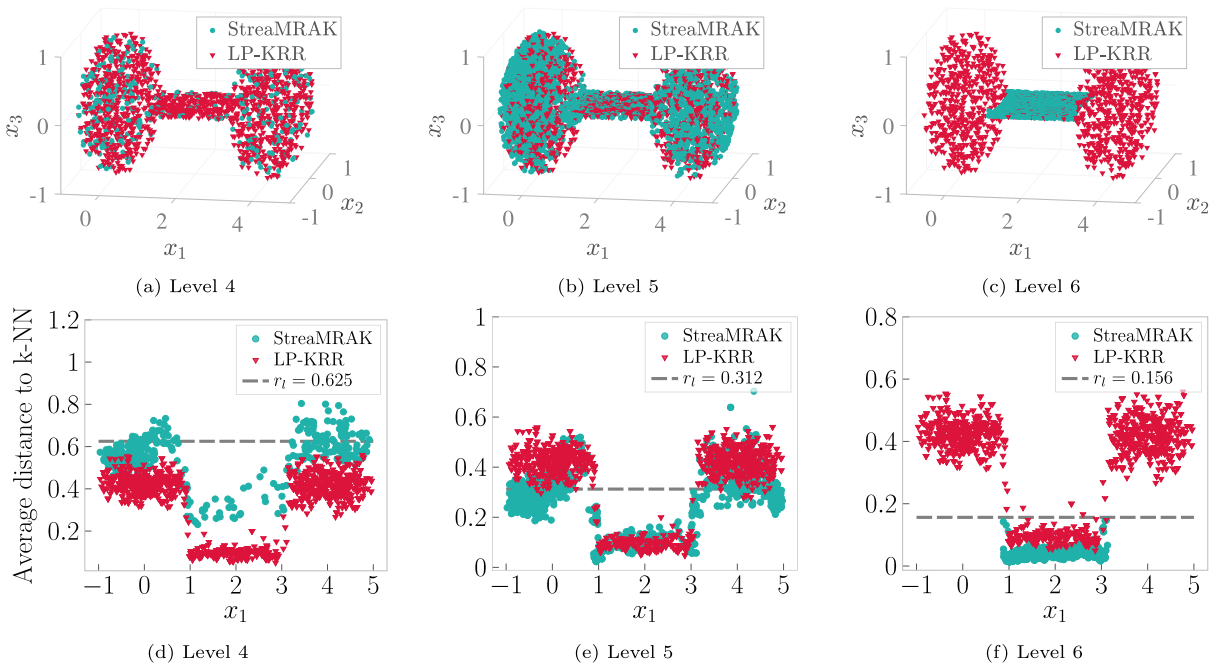
Fig. 7. In the figure, red triangles correspond to LP-KRR and light-blue circles to StreaMRAK. (a)-(c) shows the landmark distributions projected on $\mathbb{R}^3$ at level $l = 4, 5, 6$ respectively. (d)-(f) shows the average distance between the 7 nearest neighbors; bandwidth r_l is indicated with a dotted line. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

respectively, where $\sigma(\mathbf{s}) = [0.025|\theta_1|, 0.15|\theta_2|, 0.3|\omega_1|, 0.3|\omega_2|]$. Each pendulum is iterated for 500 steps, which results in 5×10^6 training points distributed in $\mathbb{R}^4$. Furthermore, for the test data we consider 100 pendulums $\mathbf{s}_0 \sim \mathcal{N}(\mathbf{s}, 0.01|\mathbf{s}|)$ for $\mathbf{s} = \mathbf{s}_0^{low}, \mathbf{s}_0^{high}$, iterated for 500 steps.

To determine the number of training points for StreaMRAK, we let $\delta_1, \delta_2 = 10^{-4}$, cf. Def. 4.1. With this choice StreaM-RAK selects between 30219 and 70282 training points for each level for the low energy system, and between 36300 and 130200 for the high energy system. Meanwhile, FALKON uses all 5.0×10^6 training points and LP-KRR use 3.9×10^5 training points at each level.

Results are presented in Table 3 and Table 4. Furthermore, to illustrate the prediction results we consider the center of mass $\overline{M}_x(\mathbf{s}_t) = \frac{1}{2}(x_1(t) + x_2(t)) \in \mathbb{R}$ at state $\mathbf{s}_t$, where $x_1, x_2 \in \mathbb{R}$ are the positions of the two pendulum masses as seen in

Table 3
Comparison of StreaMRAK, LP-KRR, and FALKON for the low energy system. For each level l we show the number of landmarks, the MSE at step T=50, and the accumulated time to train the prediction model (Time). In parenthesis, in the time column of the FALKON row, is the time to find the optimal bandwidth through cross-validation.

	Level	# Landmarks	MSE(T=50)	Time
StreaMRAK	2	1	1.12×10^{-1}	31 s
	5	66	3.39×10^{-5}	498 s
	7	659	1.44×10^{-6}	534 s
	9	4085	3.01×10^{-7}	812 s
LP-KRR	2	1979	5.93×10^{-2}	490 s
	5	1979	2.73×10^{-5}	1463 s
	7	1979	2.16×10^{-7}	2395 s
	9	1979	1.16×10^{-8}	3550 s
FALKON	–	19790	5×10^{6}	2934 s+(1498 s)

Table 4
Comparison of the StreaMRAK, LP-KRR, and FALKON for the high energy system. For each level l we show the number of landmarks, the MSE at step T=50, and the accumulated time to train the prediction model (Time). In parenthesis, in the time column of the FALKON row, is the time to find the optimal bandwidth through cross-validation.

	Level	# Landmarks	MSE(T=50)	Time
StreaMRAK	2	1	2.70×10^{-1}	49 s
	5	1106	8.53×10^{-3}	915 s
	7	6376	2.16×10^{-4}	1999 s
LP-KRR	2	1979	1.72×10^{-2}	522 s
	5	1979	5.09×10^{-3}	1474 s
	7	1979	1.39×10^{-4}	2431 s
FALKON	–	19790	5×10^{6}	23830 s+(11050 s)

(a) Mean square error (b) Forecasting trajectory (c) Training time
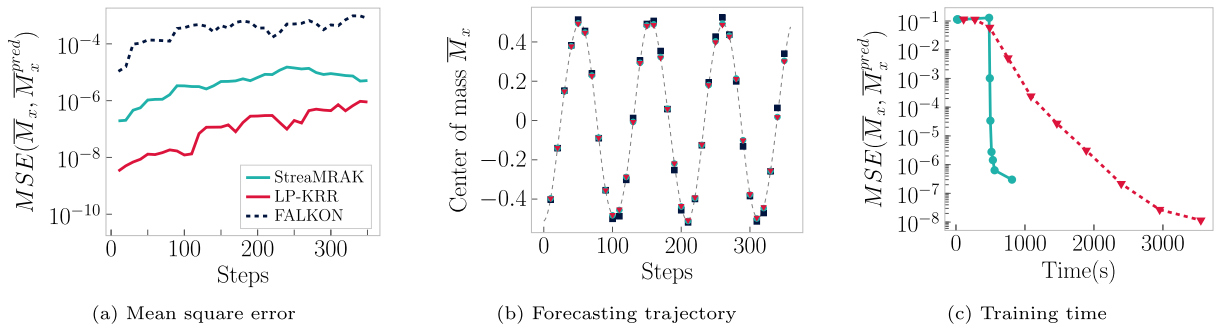
Fig. 8. Comparison of StreaMRAK (light blue lines and circles), LP-KRR (red lines and triangles), and FALKON (dark blue dotted lines and squares) for the low energy pendulum. (a) Shows the mean square error of the center of mass $\overline{M}_x(\mathbf{s}_t)$ for the level 7 prediction, with step T along the x-axis. (b) shows the true center of mass trajectory as a grey dotted line and the predictions of StreaMRAK, LP-KRR, and FALKON at level 9. (c) The x-axis shows the accumulated training time until a level in the LP is completed. The y-axis shows the MSE of the predicted system state after $T = 50$ steps. We note that StreaMRAK includes 7 levels, while LP-KRR includes 9. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Fig. 2a. The prediction results are illustrated in Fig. 8 and Fig 9 for the low and high energy pendulums respectively. We calculate the MSE at each step t separately, such that for a given t we use Eq. 6.1 with $\mathbf{y}_k = \overline{M}_x(\mathbf{s}_t)$, $\mathbf{y}_k^{pred} = \overline{M}_x(\mathbf{s}_t^{pred})$ and $\Upsilon = 100$.

For the low energy system, we see from Fig. 8c how StreaMRAK is trained significantly faster than LP-KRR, although at a cost of reduced precision. The reduced training time of StreaMRAK is a consequence of the low doubling dimension of the training data, which allows the selection of far fewer landmarks for StreaMRAK than what is used at each level in LP-KRR.

For the high-energy pendulum, we see from Fig. 9c that StreaMRAK is again able to achieve good precision faster than LP-KRR. Furthermore, we see that the number of landmarks selected for StreaMRAK increases abruptly with the levels, reflecting the high doubling dimension of the training data. Due to this StreaMRAK stops the training after level 7, as
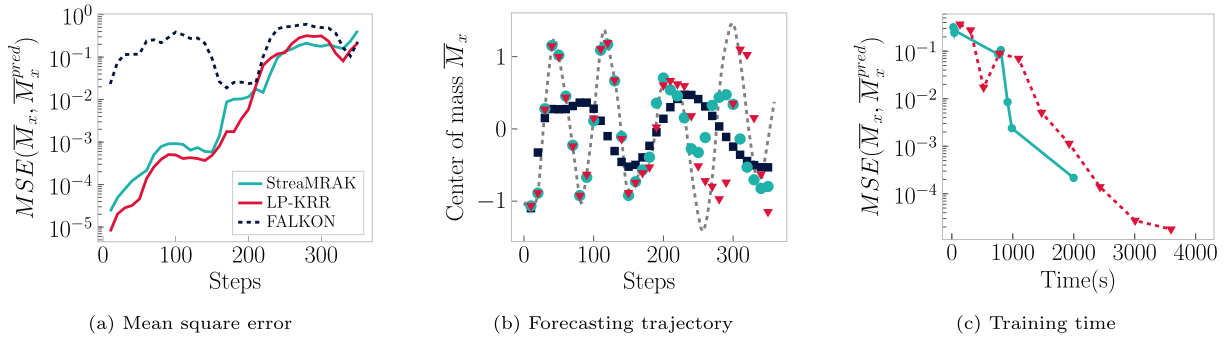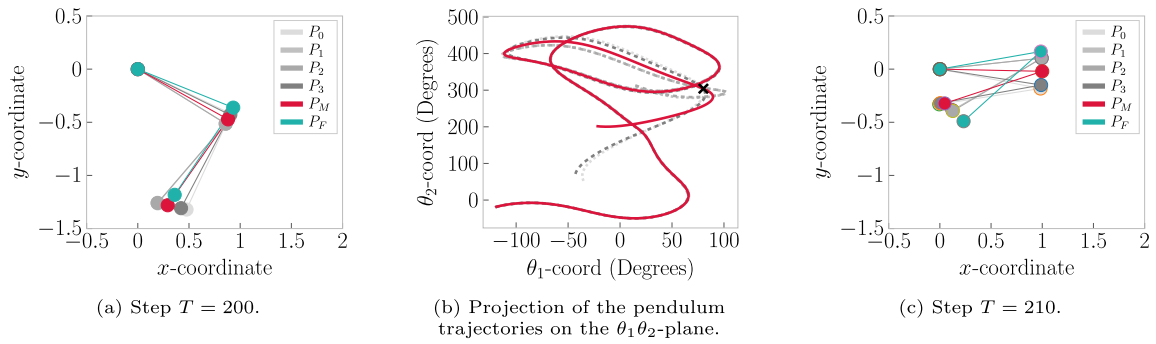
(a) Mean square error   (b) Forecasting trajectory   (c) Training time

Fig. 9. Comparison of StreaMRAK (light blue lines and circles), LP-KRR (red lines and triangles), and FALKON (dark blue dotted lines and squares) for the high-energy pendulum. (a) Shows the mean square error of the center of mass $\overline{M}_x(\mathbf{s}_t)$ for the level 7 prediction, with step T along the x-axis. (b) shows the true center of mass trajectory as a grey dotted line and the predictions of StreaMRAK, LP-KRR, and FALKON at level 9. (c) The x-axis shows the accumulated training time until a level in the LP is completed. The y-axis shows the MSE of the predicted system state after $T = 50$ steps. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

(a) Step $T = 200$.   (b) Projection of the pendulum   (c) Step $T = 210$.
                              trajectories on the $\theta_1\theta_2$-plane.

Fig. 10. (a) Pendulum positions at $T = 200$ and (c) The positions at $T = 210$. In (a) and (c), P_M is the main pendulum with initial conditions $\mathbf{s}_0^{high}$, while P_F is the StreaMRAK forecast of the pendulum position. Similarly, $P_0 - P_3$ are four training pendulums with a perturbation of 0.5% on the initial angles θ_1 and θ_2 of the main pendulum. (b) Projection of the training data on the $\theta_1\theta_2$-plane. The thick red line is the main pendulum corresponding to P_M and the four grey dotted lines are the test pendulums $P_0 - P_3$, where the X indicates the time $T = 205$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

the next levels require too many landmarks. By continuing for 2 more levels LP-KRR is able to achieve marginally better precision but at increased computational cost.

As seen in Fig. 9b, the forecasting of StreaMRAK and LP-KRR breaks down after $T \approx 200$ steps. In Fig. 10b we observe the trajectory of a pendulum with initial condition $\mathbf{s}_0^{high}$, as well as four pendulums with a 0.5% perturbation on the angles θ_1 and θ_2 in $\mathbf{s}_0^{high}$. We observe that after roughly $T = 205$ time steps the trajectory of the five pendulums diverge significantly from each other. Therefore, it seems that a bifurcation point occurs around this time, which may explain why all the algorithms are unable to make good forecasting beyond this point.

7. Outlook

Further development of StreaMRAK is intended with focus on four objectives.

(O1) Augmentation of the DCT to track the error at each node
(O2) Improve the estimator in Def. 4.1 and Eq. 3.1.
(O3) Refinement of previously fitted levels in the LP as new data arrives.
(O4) Further theoretical analysis of the LP.

Considering objective (O1) we intend to develop the DCT to track the error at each node. This way the growth can be restricted in regions where the error is small, which allows for more focus on regions where the error is large. The intention is that this will reduce the problem complexity even further, while also increasing the precision. Regarding objective (O2), a drawback with the estimator in Eq. 3.1 was already mentioned in Remark B.1 in Appendix B. Furthermore, for the estimator in Def. 4.1, we intend to implement and evaluate alternative ways to estimate the convergence of the matrices. Another focus area will be objective (O3), as we believe new information may be revealed as new training data arrive, and refinement of previously fitted levels can therefore be beneficial. Finally, the theoretical analysis in objective (O4) will focus on analyzing the generalization error for the LP, particularly in combination with the adaptive sub-sampling scheme.

Acknowledgement

Appendix A. Algorithms

We here denote nodes by p, q, c and $\mathbf{x}_p, \mathbf{x}_q, \mathbf{x}_c \in \mathcal{X} \subset \mathbb{R}^D$ are the corresponding points.

Algorithm A.1 INSERT(point q, node p, level l).

1: We assume q already satisfies $\|\mathbf{x}_q - \mathbf{x}_p\| \leq 2^{-l}r_0$.
2: **if** $\|\mathbf{x}_q - \mathbf{x}_c\| > 2^{-(l+1)}r_0$ for all $c \in Children(p)$ **then**
3:    Insert q into $Children(c)$.
4:    UPDATE_COVERFRACTION($Parent(Q_l)$, "No parent found")
5:    **Break**
6: **else if** $\|\mathbf{x}_q - \mathbf{x}_c\| < 2^{-(l+1)}r_0$ for some $c \in Children(p)$ **then**
7:    Consider all children of c, namely $Children(c)$
8:    **if** $Children(c)$ is empty **then**
9:       **if** Covering fraction of p, Def. 3.1, satisfy $\mathfrak{cf}(p) \geq \mathcal{D}_{\mathfrak{cf}}$ for some threshold $\mathcal{D}_{\mathfrak{cf}}$ **then**
10:          Insert q into $Children(c)$
11:          **Break**
12:       **else**
13:          UPDATE_COVERFRACTION(p, "parent found"){c is found to be a potential parent. However, since $\mathfrak{cf}(p) < \mathcal{D}_{\mathfrak{cf}}$ we can not add q to $Children(c)$}
14:       **end if**
15:    **else**
16:       INSERT($q, c, l+1$)
17:    **end if**
18: **end if**

Algorithm A.2 STREAMRAK(point $\mathbf{x}$, target y).

1: Let l be the level. Let $p_{(0)}$ be the root node, r_0 the radius of the root node.
2: **Sub-sampling thread**
3: Insert $\mathbf{x}$ into the cover tree with INSERT($\mathbf{x}, p_{(0)}, l = 0$). {See Alg. A.1}
4: **if** a new level has $\mathfrak{cf}(Q_l) \geq \mathcal{D}_{level}$. **then**
5:    Extract the landmarks at level l as sub-samples, namely $\Gamma^{(l)}_{m^{(l)}}$.
6: **end if**
7: **Training thread**
8: Consider level l and assume that the landmarks $\Gamma^{(l)}_{m^{(l)}}$ are extracted.
9: **while** l is not sufficiently covered with training points according to Def. 4.1. **do**
10:    Update $\left[(\mathbf{K}^{(l)}_{nm})^\top \mathbf{K}^{(l)}_{nm}\right]_{ij}$ and $\mathbf{z}^{(l)}_i$ according to Eq. 2.4 and Eq. 2.5 as new samples $(\mathbf{x}, y)$ arrive, using the landmarks in $\widetilde{\Gamma}^{(l)}_m$ from Def. 3.3.
11:    Continuously check if matrices have converged.
12:    **if** Matrices converge according to Def. 4.1 **then**
13:       Update the StreaMRAK regression model $\widetilde{f}^{(L)}$, by including the correction term $s^{(l)}$ into the Laplacian pyramid, as described in Section 2.2. Let $L = l$ and update $l = l + 1$.
14:    **end if**
15: **end while**

Algorithm A.3 UPDATE_COVERFRACTION(node p, string s).

1: **if** s="No parent found" **then**
2:    Update covering fraction of p with $\mathfrak{cf}(p) = (1 - \alpha)\mathfrak{cf}(p)$
3: **else if** s= "parent found" **then**
4:    Update covering fraction of p with $\mathfrak{cf}(p) = (1 - \alpha)\mathfrak{cf}(p) + \alpha$
5: **end if**

Appendix B. Preparatory material

We offer preparatory material on the damped cover-tree and kernel methods.

B1. Preparatory material on the damped cover-tree

This section shows how the recursive formula in Eq. 3.1 approximates the weighted average of the outcome of the last N random trails. Where the trails are as described in Section 3.1. By expanding Eq. 3.1 we have $(\mathfrak{cf}(p))_t = (1 - \alpha)^t (\mathfrak{cf}(p))_1 + \alpha \sum_{i=1}^{t-1}(1 - \alpha)^i \mathbb{1}_{\mathcal{B}_c}(\mathbf{x}_{t-i})$. Since $(1 - \frac{1}{N})^N \approx 1/e$, the first term becomes negligible when $t \gg N$. Similarly, all terms $i > N$ in the sum becomes negligible. This leaves,

$$(\mathfrak{cf}(p))_t \approx \frac{1}{N} \sum_{i=1}^{N} \left(1 - \frac{1}{N}\right)^i \mathbb{1}_{\mathcal{B}_c}(\mathbf{x}_{t-i})$$

which is a weighted average of the outcome of the N last draws as claimed.

Remark B.1. We mention a weakness of the estimator in Eq. (3.1). As follows from Algorithm A.1, every time a new point $\mathbf{x}$ is not covered by the existing children, a new child is added. This consequently updates $\mathcal{B}_c$, leading to the posterior distribution $\text{Prob}(\mathbb{1}_{\mathcal{B}_c}(\mathbf{x}) = 0 | \mathbf{x})$ to changed every time $\mathbb{1}_{\mathcal{B}_c}(\mathbf{x}) = 0$.

B2. Preparatory material on Kernel methods

Kernel methods in the context of reproducing kernel Hilbert spaces (RKHS) offer a powerful approach to machine learning with a well-established mathematical foundation [1,62]. In this paper we consider an input space $\mathcal{X} \subset \mathbb{R}^D$, a corresponding target space $\mathcal{Y} \subset \mathbb{R}$ and let ρ be the probability distribution on $\mathcal{X} \times \mathcal{Y}$. Furthermore, we assume an RKHS $\mathcal{H}_k$ generated by a positive definite kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. In other words, the eigenvalues $\sigma_i, \ldots, \sigma_n$ of the corresponding kernel matrix $\mathbf{K}_{nn} = (k(\mathbf{x}_i, \mathbf{x}_j))_{i,j=1}^n$ satisfies $\sigma_i > 0$ for all $i \in n$. In this setting the inner product between two feature vectors $\phi(\mathbf{x}), \phi(\mathbf{x}') \in \mathcal{H}_k$ satisfies the property that $< \phi(\mathbf{x}), \phi(\mathbf{x}') >_{\mathcal{H}_k} = k(\mathbf{x}, \mathbf{x}')$. This relation, known as the "kernel trick" [63,64], effectively circumvents the need for explicit construction of non-linear mappings ϕ.

Given a training set $\{(\mathbf{x}_i, y_i) : i \in [n]\}$ sampled according to ρ with $\Gamma_n = \{\mathbf{x}_i : i \in [n]\}$, we formulate the kernel ridge regression (KRR) problem as

$$\widehat{f}_{n,\lambda} = \underset{f \in \widehat{\mathcal{H}}_n}{\text{argmin}} \frac{1}{n} \sum_{i=1}^{n}(f(\mathbf{x}_i) - y_i)^2 + \lambda \|f\|_{\mathcal{H}}^2, \tag{B.1}$$

where $\lambda > 0$ is a regularisation parameter and $\widehat{\mathcal{H}}_n = \overline{\text{span}}\{k(\cdot, \mathbf{x}_i) : i \in [n]\}$ is a finite-dimensional subspace of $\mathcal{H}_k$. What is more, for all $f \in \widehat{\mathcal{H}}_n$ the Representer theorem [65,66] guarantees that there exists coefficients $\alpha_1, \ldots, \alpha_n$ such that the solution to Eq. (B.1) is on the form

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}, \mathbf{x}_i).$$

Computing the KRR estimator is therefore reduced to solving the linear system

$$(\mathbf{K}_{nn} + \lambda \mathbf{I}_n)\boldsymbol{\alpha} = \mathbf{y},$$

where $\mathbf{y} = (y_1, \ldots, y_n)^\top$, $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^\top$, and $[\mathbf{K}_{nn}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.

Appendix C. Auxiliary results

Lemma C.1. *Let* $\mathbf{d}^{(l)}$ *be the residual at level l as defined in Eq. (2.9). We then have,*

$$\mathbf{d}^{(l+1)} = (\mathbf{I} - \mathbf{K}_{nn}^{(l)}(\mathbf{K}_{nn}^{(l)} + \lambda n\mathbf{I})^{-1})\mathbf{d}^{(l)}$$

Proof. Denote $\mathbf{s}^{(l)} = s^{(l)}([\mathbf{x}_n])$, and note that $\mathbf{s}^{(l)} = \mathbf{K}_{nn}^{(l)}(\mathbf{K}_{nn}^{(l)} + \lambda n\mathbf{I})^{-1}\mathbf{d}^{(l)}$. For $l = 1$, we have

$$\mathbf{d}^{(1)} = \mathbf{y} - \mathbf{s}^{(0)} = \mathbf{y} - \mathbf{K}_{nn}^{(l)}\boldsymbol{\alpha}^{(0)} = (\mathbf{I} - \mathbf{K}_{nn}^{(0)})(\mathbf{K}_{nn}^{(0)} + \lambda n\mathbf{I})^{-1}\mathbf{y}.$$

We proceed by induction. Assume the statement holds for an $l \geq 2$. We now have

$$\mathbf{d}^{(l+1)} = \mathbf{y} - \sum_{j=0}^{l}\mathbf{s}^{(j)} = \mathbf{d}^{(l)} - \mathbf{s}^{(l)} = \mathbf{d}^{(l)} - \mathbf{K}_{nn}^{(l)}(\mathbf{K}_{nn}^{(l)} + \lambda n\mathbf{I})^{-1}\mathbf{d}^{(l)} = (\mathbf{I} - \mathbf{K}_{nn}^{(l)}(\mathbf{K}_{nn}^{(l)} + \lambda n\mathbf{I})^{-1})\mathbf{d}^{(l)}.$$

$\square$

Remark C.2. In [60, Thm. 12.3] they also offer an a fortiori bound corresponding to $M_D = 6.38D$, $C_{1,D} = \frac{1}{2}\left(\frac{12.76}{2^{3/2}}\right)^D\left(\frac{D^D}{\Gamma(D/2+1)}\right)\left(\frac{r_0}{\delta}\right)^D$ and $C_{2,D} = (12.76\sqrt{2}D)^2(r_0/\delta)^2$.

Corollary C.3. *We note that $B(l)$ from Eq. (5.8) has a maximum at*

$$l^* = \frac{1}{2}\log_2\left(\frac{C_{2,D}\log 4}{D\log 2}\right) = \log_2\left(\sqrt{\frac{D}{2}}\left(\frac{r_0}{\delta}\right)\right) + \log_2\left(\frac{4M_D}{D}\sqrt{2}\right)$$

and is monotonically increasing with l on the interval $l \in (0, l^*)$. Furthermore, with the a fortiori expression for M_D from Remark C.2 we have

$$l^* = \log_2\left(\sqrt{\frac{D}{2}}\left(\frac{r_0}{\delta}\right)\right) + \log_2\left(25.52\sqrt{2}\right).$$

Lemma C.4. *The function*

$$F(T) := 2^{-\frac{T}{2}(1+1/\ln 2)}2^{-lT}T^{\frac{T}{2}}\left(\frac{r_0}{\delta}\right)^T$$

has minimum

$$F(T^*) = 2^{-\frac{1}{\ln 2}}4^{l-\log_2(r_0/\delta)}.$$

Proof. We can write $F(T)$ as

$$F(T) = 2^{-\frac{T}{2}(1+1/\ln 2)}2^{-lT}2^{T/2\log_2 T}2^{T\log_2(r_0/\delta)} = 2^{-T/2(B-\log_2 T)} = 2^{f(T)},$$

where $B = 1 + \frac{1}{\ln 2} + 2l - 2\log_2\left(\frac{r_0}{\delta}\right)$ and $f(T) = -T/2(B - \log_2 T)$. $F(T)$ is therefore minimized when $f(T)$ is minimized. Namely when

$$T^* = 2^{B-1/\ln 2} = 2^{1+1/\ln 2+2l-2\log_2(r_0/\delta)-1/\ln 2} = 2 \cdot 4^{l-\log_2(r_0/\delta)}.$$

Inserting this back into the expression for $F(T)$ gives the desired result. $\square$

References

[1] B. Schölkopf, A.J. Smola, Learning with kernels: Support vector machines, optimization, and beyond, 1st, MIT press, 2002.
[2] A. Rudi, L. Carratino, L. Rosasco, FALKON: An optimal large scale kernel method, in: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Proc. 31th Int. Conf. Neural Inf. Process. Syst., volume 30, 2017, pp. 3889–3899.
[3] A.E. Alaoui, M.W. Mahoney, Fast randomized kernel ridge regression with statistical guarantees, in: Proc. 28th Int. Conf. Neural Inf. Process. Syst., volume 1, 2015, pp. 775–783.
[4] Y. Zhang, J. Duchi, M. Wainwright, Divide and conquer kernel ridge regression: adistributed algorithm with minimax optimal rates, J. Mach. Learn. Res. 16 (2015) 3299–3340.
[5] H. Avron, K.L. Clarkson, D.P. Woodruff, Faster kernel ridge regression using sketching and preconditioning, J. Matrix. Anal. Appl. 38 (4) (2017) 1116–1138, doi:10.1137/16M1105396.
[6] E. Burnaev, I. Nazarov, Conformalized kernel ridge regression, in: Proc. 15th Int. Conf. Mach. Learn. Appl., 2017, pp. 45–52, doi:10.1109/ICMLA.2016.65.
[7] P. Exterkate, P.J. Groenen, C. Heij, D. van Dijk, Nonlinear forecasting with many predictors using kernel ridge regression, Int. J. Forecas 32 (3) (2016) 736–753, doi:10.1016/j.ijforecast.2015.11.017.
[8] M. Niu, S. Rogers, M. Filippone, D. Husmeier, Fast parameter inference in nonlinear dynamical systems using iterative gradient matching, in: Proc. 33rd Int. Conf. Mach. Learn. Res., 2016, pp. 1699–1707.
[9] M. Stock, T. Pahikkala, A. Airola, B. De Baets, W. Waegeman, A comparative study of pairwise learning methods based on kernel ridge regression, Neural Comput. 30 (8) (2018) 2245–2283, doi:10.1162/neco_a_01096.
[10] S. An, W. Liu, S. Venkatesh, Face recognition using kernel ridge regression, in: Proc. Conf. Comput. Vis. Recognit., 2007, pp. 1–7, doi:10.1109/CVPR.2007.383105.
[11] B.Y.S. Li, L.F. Yeung, K.T. Ko, Indefinite kernel ridge regression and its application on QSAR modelling, Neurocomputing 158 (2015) 127–133, doi:10.1016/j.neucom.2015.01.060.
[12] P. Mohapatra, S. Chakravarty, P.K. Dash, Microarray medical data classification using kernel ridge regression and modified cat swarm optimization based gene selection system, Swarm. Evol. Comput. 28 (2016) 144–160, doi:10.1016/j.swevo.2016.02.002.
[13] S. Muthukrishnan, Data streams: algorithms and applications, Found. Trends Theor. Comput. Sci. 1 (2) (2005) 117–236, doi:10.1561/0400000002.
[14] W. Fan, A. Bifet, Mining big data, ACM SIGKDD Explor. Newsl. 14 (2) (2013) 1–5, doi:10.1145/2481244.2481246.
[15] K. Lan, D.-T. Wang, S. Fong, L.-S. Liu, K.K.L. Wong, N. Dey, A survey of data mining and deep learning in bioinformatics, J. Med. Syst. 42 (8) (2018), doi:10.1007/s10916-018-1003-9.
[16] J. Kivinen, A.J. Smola, R.C. Williamson, Online learning with kernels, in: Proc. 14th Int. Conf. Neural Inf. Process. Syst., 2001, pp. 785–792, doi:10.7551/mitpress/1120.003.0105.

[17] C. Scovel, D. Hush, I. Steinwart, J. Theiler, Radial kernels and their reproducing kernel hilbert spaces, J. Complex. 26 (6) (2010) 641–660, doi:10.1016/j.jco.2010.03.002.
[18] C.A. Micchelli, Y. Xu, H. Zhang, Universal kernels, J. Mach. Learn. Res. 7 (2006) 2651–2667.
[19] Z. Wang, K. Crammer, S. Vucetic, Breaking the curse of kernelization: budgeted stochastic gradient descent for large-scale SVM training, J. Mach. Learn. Res. 13 (2012) 3103–3131.
[20] C.R. Loader, Bandwidth selection: classical or plug-in? Ann. Stat. 27 (2) (1999) 415–438.
[21] G.C. Cawley, N.L. Talbot, Fast exact leave-one-out cross-validation of sparse least-squares support vector machines, Neural Netw. 17 (10) (2004) 1467–1475, doi:10.1016/j.neunet.2004.07.002.
[22] S. Arlot, A. Celisse, A survey of cross-validation procedures for model selection, Stat. Surv. 4 (2010) 40–79, doi:10.1214/09-SS054.
[23] R. Krauthgamer, J.R. Lee, Navigating nets: Simple algorithms for proximity search, in: Proc. 15th Annu. ACM-SIAM Symp. Discrete Algorithms, 2004, pp. 798–807.
[24] A. Beygelzimer, S. Kakade, J. Langford, Cover trees for nearest neighbor, in: Proc. 23th Int. Conf. Mach. Learn., 2006, pp. 97–104, doi:10.1145/1143844.1143857.
[25] T. Shinbrot, C. Grebogi, J. Wisdom, J.A. Yorke, Chaos in a double pendulum, Am. J. Phys. 60 (6) (2016) 491–499, doi:10.1119/1.16860.
[26] A. Marcelo Tusset, V. Piccirillo, A.M. Bueno, J. Manoel Balthazar, D. Sado, J.L.P. Felix, R.M.L. Brasil, Chaos control and sensitivity analysis of a double pendulum arm excited by an RLC circuit based nonlinear shaker, J. Vib. Control 22 (17) (2016) 3621–3637, doi:10.1177/1077546314564782.
[27] N. Rabin, R.R. Coifman, Heterogeneous datasets representation and learning using diffusion maps and Laplacian pyramids, in: Proc. 12th Int. Conf. Data Min., 2012, pp. 189–199, doi:10.1137/1.9781611972825.17.
[28] W. Leeb, Properties of Laplacian pyramids for extension and denoising, 2019, arXiv:1909.07974
[29] P.J. Burt, E.H. Adelson, The laplacian pyramid as a compact image code, IEEE Trans. commun. 31 (4) (1983) 532–540.
[30] G.R. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, M.I. Jordan, Learning the kernel matrix with semidefinite programming, J. Mach. Learn. Res. 5 (2004) 27–72.
[31] F.R. Bach, G.R. Lanckriet, M.I. Jordan, Multiple kernel learning, conic duality, and the SMO algorithm, in: Proc. 21th Int. Conf. Mach. Learn., 2004, pp. 41–48, doi:10.1145/1015330.1015424.
[32] S. Sonnenburg, G. Rätsch, C. Schäfer, B. Schölkopf, Large scale multiple kernel learning, J. Mach. Learn. Res. 7 (2006) 1531–1565.
[33] E.G. Băzăvan, F. Li, C. Sminchisescu, Fourier kernel learning, in: Eur. Conf. Comput. Vis., 2012, pp. 459–473, doi:10.1007/978-3-642-33709-3_33.
[34] A. Bermanis, A. Averbuch, R.R. Coifman, Multiscale data sampling and function extension, Appl. Comput. Harmon. Anal. 34 (1) (2013) 15–29, doi:10.1016/j.acha.2012.03.002.
[35] N. Rabin, D. Fishelov, Multi-scale kernels for nyström based extension schemes, Appl. Math. Comput. 319 (2018) 165–177, doi:10.1016/j.amc.2017.02.025.
[36] W. Liao, M. Maggioni, S. Vigogna, Multiscale regression on unknown manifolds, Mathematics in Engineering 4 (4) (2022) 1–25, doi:10.3934/mine.2022028.
[37] H. Fan, Q. Song, S.B. Shrestha, Kernel online learning with adaptive kernel width, Neurocomputing 175 (2015) 233–242, doi:10.1016/j.neucom.2015.10.055.
[38] B. Chen, J. Liang, N. Zheng, J.C. Príncipe, Kernel least mean square with adaptive kernel size, Neurocomputing 191 (2016) 95–106, doi:10.1016/j.neucom.2016.01.004.
[39] J. Zhang, H. Ning, X. Jing, T. Tian, Online kernel learning with adaptive bandwidth by optimal control approach, IEEE Trans. Neural Netw. Learn. Syst. 32 (5) (2021) 1920–1934, doi:10.1109/TNNLS.2020.2995482.
[40] A. Graps, An introduction to wavelets, IEEE Comput. Sci. Eng. 2 (2) (1995) 50–61, doi:10.1109/99.388960.
[41] A.N. Akansu, W.A. Serdijn, I.W. Selesnick, Emerging applications of wavelets: a review, Phys. Commun. 3 (1) (2010) 1–18, doi:10.1016/j.phycom.2009.07.001.
[42] R.R. Coifman, M. Maggioni, Diffusion wavelets, Appl. Comput. Harmon. Anal. 21 (1) (2006) 53–94, doi:10.1016/j.acha.2006.04.004.
[43] M. Maggioni, H.N. Mhaskar, Diffusion polynomial frames on metric measure spaces, Appl. Comput. Harmon. Anal. 24 (3) (2008) 329–353, doi:10.1016/j.acha.2007.07.001.
[44] D.K. Hammond, P. Vandergheynst, R. Gribonval, Wavelets on graphs via spectral graph theory, Appl. Comput. Harmon. Anal. 30 (2) (2011) 129–150, doi:10.1016/j.acha.2010.04.005.
[45] A. Cloninger, H. Li, N. Saito, Natural graph wavelet packet dictionaries, J. Fourier Anal. Appl. 27 (3) (2021) 1–33, doi:10.1007/s00041-021-09832-3.
[46] E. De Vito, Z. Kereta, V. Naumova, L. Rosasco, S. Vigogna, Wavelet frames generated by a reproducing kernel, J. Fourier Anal. Appl. 27 (2) (2021) 1–39, doi:10.1007/s00041-021-09835-0.
[47] A. Rahimi, B. Recht, Random features for large-scale kernel machines, in: Adv. Neural Inf. Process Syst., volume 20, 2008, pp. 1177–1184.
[48] Q.V. Le, T. Sarlos, A. Smola, Fastfood-computing hilbert space expansions in loglinear time, in: Proc. 30th Int. Conf. Mach. Learn., volume 28, 2013, pp. 244–252.
[49] Z. Yang, A.J. Smola, L. Song, A.G. Wilson, A la carte | learning fast kernels, in: Proc. 18th Int. Conf. Artif. Intell. Stat., volume 38, 2015, pp. 1098–1106.
[50] J. Zhang, A. Cloninger, R. Saab, Sigma-delta and distributed noise-shaping quantization methods for random fourier features, 2021, arXiv:2106.02614
[51] C. Williams, M. Seeger, Using the Nyström method to speed up kernel machines, in: Proc. 14th Annu. Conf. Neural Inf. Process Syst., volume 13, 2001, pp. 682–688.
[52] A.J. Smola, B. Schölkopf, Sparse greedy matrix approximation for machine learning, in: Proc. 1th Int. Conf. Mach. Learn., 2000, pp. 911–918.
[53] A. Cloninger, Prediction models for graph-linked data with localized regression, in: Proc. SPIE Int. Soc. Opt. Eng., volume 10394, 2017, doi:10.1117/12.2271840.
[54] S. Ma, R. Bassily, M. Belkin, The power of interpolation : Understanding the effectiveness of SGD, in: Proc. 35th Int. Conf. Mach. Learn., 2018, pp. 3331–3340.
[55] S. Ma, M. Belkin, Kernel machines that adapt to GPUs for effective large batch training, 2018, arXiv:1806.06144
[56] J. Picka, Statistical inference for disordered sphere packings, Stat. Surv. 6 (2012) 74–112, doi:10.1214/09-SS058.
[57] M. Hifi, R. M'Hallah, A literature review on circle and sphere packing problems: models and methodologies, Adv. Oper. Res. (2009), doi:10.1155/2009/150624.
[58] P. Flajolet, D. Gardy, L. Thimonier, Birthday paradox, coupon collectors, caching algorithms and self-organizing search, Discrete Appl. Math. 39 (3) (1992) 207–229, doi:10.1016/0166-218X(92)90177-C.
[59] G. Klambauer, Problems and propositions in analysis, Marcel Dekker, New York, 1979.
[60] H. Wendland, Scattered data approximation, Cambridge University Press, 2004, doi:10.1017/cbo9780511617539.
[61] D. Gómez, A more direct proof of gerschgorinós theorem, Mat: Enseñanza Univ. 14 (2) (2006) 119–122.
[62] T. Hofmann, B. Schölkopf, A.J. Smola, Kernel methods in machine learning, Ann. Stat. 36 (3) (2008) 1171–1220, doi:10.1214/009053607000000677.
[63] M. Aiserman, E.M. Braverman, L.I. Rozonoer, Theoretical foundations of the potential function method in pattern recognition, Avtomat. i Telemeh. 25 (6) (1964) 917–936.
[64] B.E. Boser, I.M. Guyon, V.N. Vapnik, A training algorithm for optimal margin classifiers, in: Proc. 5th Annu. Workshop. Comput. Learn. Theory, 1992, pp. 144–152, doi:10.1145/130385.130401.
[65] G.S. Kimeldorf, G. Wahba, A correspondence between bayesian estimation on stochastic processes and smoothing by splines, Ann. Math. Stat. 41 (2) (2011) 495–502, doi:10.1214/aoms/1177697089.
[66] B. Schölkopf, R. Herbrich, A.J. Smola, A generalized representer theorem, in: Int. Conf. Comput. Learn. Theory, 2001, pp. 416–426, doi:10.1007/3-540-44581-1_27.